|  |  |
| --- | --- |
|  |  |
|  | Penetration Testing Report |
|  | 03/03/2016 |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Customer Information | | | | | |
| Company Name: |  | | | | | |
| City: |  | State: |  | Zip Code: |  | |
| URL: |  |  |  |  |  | |

|  |  |
| --- | --- |
| Customer Contact Information | |
| Contact Name: |  |
| Title: |  |
| Telephone: |  |
| E-mail: |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Consultant Information | | | | | |
| Company Name: | EY | | | | |
| Contact Name: |  | | | | |
| Title: |  | | | | |
| Telephone: |  | | | | |
| E-mail: |  | | | | |
| Business Address: | EY Building, Tunis | | | | |
| City | Tunisia | State: | MA | Zip Code: | 11111 |
| URL: | <http://www.ey.com> | | | | |

# Introduction

# Context and objectives

EY was responsible for conducting technical tests to evaluate the security level of . In this document, will be reported the results of audit tests through the enumeration of identified vulnerabilities and the specification of recommendations in order to mitigate the related risks.

## Scope and limits

EY was responsible for conducting technical tests to evaluate the security level of Portal and underlying IT infrastructure services accessible from the public networks at .

The objective of the intrusion tests is to assess the security level of the in scope perimeter towards external attacks. This evaluation involves the following steps:

* Mapping of the audited scope,
* Vulnerability discovery and its existence validation through exploitation tests,
* Identification and assessment of risks associated to these vulnerabilities,
* Proposal of action plan to cover the identified risks.

## Testing conditions

Penetration Test approach used:

* **Full Testing***:*
* **Black Box** as a first step it will be undertaken tests on the target with no prior knowledge of the security system, its components or its flows. The target is prepared to be tested while knowing in advance the details of this assessment
* **Grey Box** in a second time will be undertaken tests on target with limited knowledge of the security system, its components and flows. The target is prepared to be tested while knowing in advance the details of this assessment tested environment.

Tested environment:

* **Production**

The following logical accesses were provided by :

# Executive summary

The penetration test of had the objective of evaluating its level of security with respect to best practices and security standards. The audit, conducted from the Internet has put forward several observations:

* Point 1
* Point 2
* Point 3

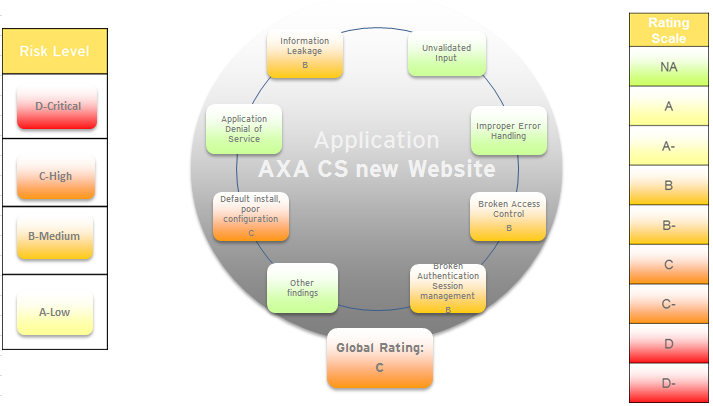
However, EY was able to achieve the goals of the assessment and exfiltrate the targeted data. There were a number of critical findings during the assessment including the following:

|  |  |
| --- | --- |
| Finding Name | Risk |
| SQL Injection | D-Critical |
| Cross Site Scripting (XSS) | D-Critical |
| Direct Object References | D-Critical |
| XML External Entity (XXE) Processing | D-Critical |

MOAR STUFF

Therefore, we consider the level of security of this application insufficient because it has exploitable vulnerabilities that might have an impact on the confidentiality and integrity of the data it hosts.

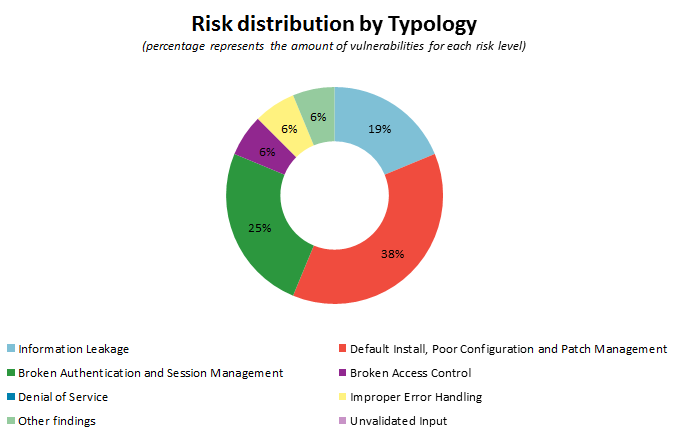
The findings on the filtering mechanisms and access controls lead us to give a rating of **C**to



In conclusion, the identified weaknesses expose Website to attacks that can have a **strong impact** on its operations, particularly through the leakage of sensitive business information and the possible alteration of alert status by malicious users.

|  |  |
| --- | --- |
| Risk distribution by severity  *(percentage represents the amount of vulnerabilities for each risk level)* | Risk distribution by location authentication *(percentage represents the amount of vulnerabilities)* |
| High risks were found during this test. | Most of the vulnerabilities don’t require a valid user account. |

Risk distribution by Typology



Most of the risks are related to Default Install, Poor Configuration and Patch Management

# Audit Methodology

## Types of test

The six types of test described below are different depending on the amount of information that the auditor has on the target and the knowledge possessed by the target on the auditor.

As part of the mission, purpose of this report, a specific methodology adapted to the context has been implemented inspired from several elements described below



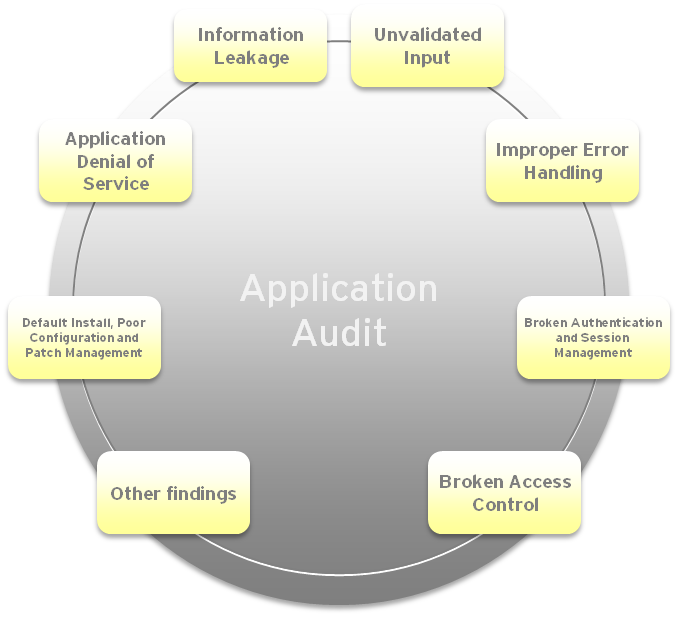
* **Blind**
* The auditor undertakes tests on the target with no prior knowledge of the security system, its components or its flows. The target is prepared to be audited while knowing in advance the details of this audit.
* **Double Blind**
* The auditor undertakes tests on the target with no prior knowledge of the security system, its components or its flows. The target is not notified in advance on the scope of this audit. Channels and vectors used for testing are also not announced.
* **Grey Box**
* The auditor undertakes tests on target with limited knowledge of the security system, its components and flows. The target is prepared to be audited while knowing in advance the details of this audit.
* **Double Grey Box**
* The auditor undertakes tests on target with limited knowledge of the security system, its components and has complete visibility of flows. The target is prepared to be audited while knowing in advance the details of this audit.
* **Tandem**
* The auditor and the target are prepared to audit and know in advance all the details of the audit.
* **Reversal**
* The auditor undertakes an audit of the target with full knowledge of all its processes and its operational security system. On the other hand, the target has no information on the nature of the tests or the date of their launches and how they are realized.

**Reference:**

* [OSSTMM 3](http://www.isecom.org/) The Open Source Security Testing Methodology Manual,
* [OWASP](https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents) Testing Guide Methodology.

## Risk assessment methodology

Our evaluation methodology is based on eight application areas:



You will find below a preview (the listed items are not exhaustive) of controls for each area:

* **A1 - Unvalidated Input**
* Identification of parameters
* Inserting unexpected values (negative, zero ...) and unexpected parameters
* Parsing errors
* Reflected values
* Unsanitized input transmitted to other application layers
* **A2 - Broken Access Control**
* Identification of access profiles
* What screens are authenticated?
* Who has access to what screen?
* Evaluation of horizontal segregation
* Attempt to access the data of another user
* Evaluation of vertical segregation
* Elevation of privilege attempt
* Functionalities analysis (identification of all functions and sensitive functions)
* **A3 - Broken Authentication and Session Management**
* Identify the authentication method
* Presence of trivial account
* Password policy Analysis
* Analysis of password recovery
* Certificates Verification
* Tests of bypass replay
* Session Management (inactivity, multiple login, ...)
* **A4 - Improper Error Handling**
* Information disclosure through technical errors
* Faulty log management
* **A5 - Information Leakage**
* Technical information disclosure
* Local File Inclusion (LFI)
* Full Path Disclosure
* Secrets disclosure
* Passwords
* Encryption keys
* **A6 - Application Denial of Service**
* Resistance to Denial of Service
* Identifying points latency
* Stress testing simultaneously latency points
* Sending unexpected data
* Resistance to reverse engineering
* **A7 - Default Install, Poor Configuration and Patch Management**
* Known and published vulnerabilities
* Testing pages with vulnerable functions
* Defaults users and passwords
* **A8 - Other findings**
* Static Analysis (Hex Editor, configuration file, Decompilation)
* Dynamic Analysis (Debugger)

## Risk Metrics Rating

#### *CVSS Rating*

The vulnerability rating is based on the CVSS v2 standard. Please refer to [https://www.first.org/cvss/v2/guide](http://www.first.org/cvss/cvss-guide.html) for full details.

Only the base metric is evaluated.

#### EY Risk Rating Evaluation

The EY grading is based on the risk associated with the vulnerability defined by its impact and by the level of exploitability, and in some cases, based on the authentication and the location parameters.

For each vulnerability, we estimate a risk rating defined by all the previous criteria using the following method:

* **Risk = Impact \* Exploitability \* (Authentication + Location)**
  + Ex1: Risk1 = (Critical impact \* Difficult exploitability \* (Anonymous + distant) = C-High.
  + Ex2: Risk2 = (High impact \* Standard exploitability \* (Authenticated + Local) = B-Medium.

NB: In some cases, the two criteria: **Authentication** and **Location** may not have any repercussion upon the risk. If the application being tested is not accessible from the internet, the **location** parameter does not have any impact on the risk evaluation.

|  |  |
| --- | --- |
| **D-Critical** | The risk associated with the vulnerability allows the total control of the system or the application. |
| **C-High** | The risk associated with the vulnerability allows partial control of the system or the application. |
| **B-Medium** | The risk associated with the vulnerability allows information disclosure which could facilitate advanced and targeted attacks. |
| **A-Low** | The risk associated with the vulnerability is very limited. It usually represents a lack of good practices follow-up, but doesn't have impacts on the company's stakes. |

#### Remediation effort

For each vulnerability identified, we estimate the ease of correction in order to correct or limit the vulnerability. We use the following possible values: Complex, Moderate, and Low. Below is a detailed description for each case.

|  |  |
| --- | --- |
| **Complex** | The correction of the identified vulnerability requires a complete modification of the code. This correction has an important impact on the system or the application audited. |
| **Medium** | The correction of the identified vulnerability requires some modification of the infrastructure supporting the application. Its impact needs to be identified in order to ensure that there will not be any side effects. |
| **Low** | The correction of the identified vulnerability requires a simple modification which will have little to no impact on the system or the application audited. |

#### Remediation Priority Level

For each vulnerability identified, we estimate the priority of remediation in order to correct or limit the vulnerability. We use the following possible values: Urgent, Standard, and Low. Below is a detailed description for each case.

|  |  |
| --- | --- |
| **Urgent** | The priority of remediation for the identified vulnerability requires a specific process besides the maintenance standard process. The correction of this vulnerability has to be immediate and treated as an important incident on the production platform. |
| **Standard** | The correction has to be integrated in the next release of the application/platform. |
| **Low** | The remediation priority is low and needs to be considered by the concerned teams. |

# Findings and Remediation:

## Full list of vulnerabilities:

| **EY reference** | **Title** | **Description** | **Severity** | **Location** | **Authentication** | **Host** | **Fix Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | SQL Injection |  | **D-Critical** | Network | None |  | KO |
|  | Cross Site Scripting (XSS) |  | **D-Critical** | Network | None |  | KO |
|  | Direct Object References |  | **D-Critical** | Network | None |  | KO |
|  | XML External Entity (XXE) Processing |  | **D-Critical** | Network | None |  | KO |
|  | Path Traversal |  | **C-High** | Network | None |  | KO |

Legend:

|  |  |
| --- | --- |
| **KO** | The fix in place does not correct the vulnerability or no patch has been applied. |
| **OK** | The fix implemented efficiently corrects the vulnerability. |
| **N/A** | No fix has been implemented. |

## Full list of remediation

| **EY reference** | **Title** | **Description** | **Severity** | **Remediation effort** | **Recommendation (summary)** | **Host** |
| --- | --- | --- | --- | --- | --- | --- |
|  | SQL Injection |  | **D-Critical** | SQL Injection | The following is recommended to prevent SQL Injection:  Use of Prepared Statements (Parameterized Queries)  Use of Stored Procedures  Never trust user input, Escaping all User Supplied Input |  |
|  | Cross Site Scripting (XSS) |  | **D-Critical** | Cross Site Scripting (XSS) | The following is recommended to remediate XSS vulnerabilities:  Never trust user input  Never insert untrusted data except in allowed locations  HTML escape before inserting untrusted data into HTML element content  Use whitelists in place for Black lists for input filtering |  |
|  | Direct Object References |  | **D-Critical** | Direct Object References | Use per user or session indirect object references. This prevents attackers from directly targeting unauthorized resources. For example, instead of using the resource’s database key, a drop down list of six resources authorized for the current user could use the numbers 1 to 6 to indicate which value the user selected. The application has to map the per-user indirect reference back to the actual database key on the server.  Check access. Each use of a direct object reference from an untrusted source must include an access control check to ensure the user is authorized for the requested object. |  |
|  | XML External Entity (XXE) Processing |  | **D-Critical** | XML External Entity (XXE) Processing |  |  |
|  | Path Traversal |  | **C-High** | MEDIUM |  |  |

1. Detailed description of Vulnerabilities :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Vulnerability** | SQL Injection | | | | | **ID** | 2015T114R-12 |
| **Brief Description** | | | | | | | |
| A Clickjacking vulnerability was detected. | | | | | | | |
| **Risk** | **D-Critical** | | **Category** | |  | | |
| **Assets** |  | | | | | | |
| **CVSS** |  | | | | | | |
| **CVSS Vector** |  | | | | | | |
| **Description** | | | | | | | |
| The OWASP guide [1] gives the following description for SQL Injection:  A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands. | | | | | | | |
| Impact | | | | | | | |
|  | | | | | | | |
| **Recommendation** | | | | | | | |
| **Urgency** | |  | | **Complexity** | | MEDIUM | |
| The following is recommended to prevent SQL Injection:   * Use of Prepared Statements (Parameterized Queries) * Use of Stored Procedures * Never trust user input, Escaping all User Supplied Input | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Vulnerability** | Cross Site Scripting (XSS) | | | | | **ID** | 2015T114R-12 |
| **Brief Description** | | | | | | | |
| A Clickjacking vulnerability was detected. | | | | | | | |
| **Risk** | **D-Critical** | | **Category** | |  | | |
| **Assets** |  | | | | | | |
| **CVSS** |  | | | | | | |
| **CVSS Vector** |  | | | | | | |
| **Description** | | | | | | | |
| The OWASP guide [1] gives the following description for Cross-Site Scripting:  Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.  An attacker can use XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. | | | | | | | |
| Impact | | | | | | | |
|  | | | | | | | |
| **Recommendation** | | | | | | | |
| **Urgency** | |  | | **Complexity** | | LOW | |
| The following is recommended to remediate XSS vulnerabilities:   * Never trust user input * Never insert untrusted data except in allowed locations * HTML escape before inserting untrusted data into HTML element content * Use whitelists in place for Black lists for input filtering | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Vulnerability** | Direct Object References | | | | | **ID** | 2015T114R-12 |
| **Brief Description** | | | | | | | |
| A Clickjacking vulnerability was detected. | | | | | | | |
| **Risk** | **D-Critical** | | **Category** | |  | | |
| **Assets** |  | | | | | | |
| **CVSS** |  | | | | | | |
| **CVSS Vector** |  | | | | | | |
| **Description** | | | | | | | |
| The OWASP guide [1] gives the following description for Insecure Direct Object Reference:  Applications frequently use the actual name or key of an object when generating web pages. Applications do not always verify the user is authorized for the target object. This results in an insecure direct object reference flaw. Testers can easily manipulate parameter values to detect such flaws and code analysis quickly shows whether authorization is properly verified. | | | | | | | |
| Impact | | | | | | | |
|  | | | | | | | |
| **Recommendation** | | | | | | | |
| **Urgency** | |  | | **Complexity** | | LOW | |
| Use per user or session indirect object references. This prevents attackers from directly targeting unauthorized resources. For example, instead of using the resource’s database key, a drop down list of six resources authorized for the current user could use the numbers 1 to 6 to indicate which value the user selected. The application has to map the per-user indirect reference back to the actual database key on the server.  Check access. Each use of a direct object reference from an untrusted source must include an access control check to ensure the user is authorized for the requested object. | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Vulnerability** | Path Traversal | | | | | **ID** | 2015T114R-12 |
| **Brief Description** | | | | | | | |
| A Clickjacking vulnerability was detected. | | | | | | | |
| **Risk** | **C-High** | | **Category** | |  | | |
| **Assets** |  | | | | | | |
| **CVSS** |  | | | | | | |
| **CVSS Vector** |  | | | | | | |
| **Description** | | | | | | | |
| Quoting from [1], a Path Traversal attack aims to access files and directories that are stored outside the web root folder. By browsing the application, the attacker looks for absolute links to files stored on the web server. By manipulating variables that reference files with “dot-dot-slash (../)” sequences and its variations, it may be possible to access arbitrary files and directories stored on file system, including application source code, configuration and critical system files, limited by system operational access control. The attacker uses “../” sequences to move up to root directory, thus permitting navigation through the file system.  This attack can be executed with an external malicious code injected on the path, like the Resource Injection attack. To perform this attack it’s not necessary to use a specific tool; attackers typically use a spider/crawler to detect all URLs available.  This attack is also known as “dot-dot-slash”, “directory traversal”, “directory climbing” and “backtracking”. | | | | | | | |
| Impact | | | | | | | |
|  | | | | | | | |
| **Recommendation** | | | | | | | |
| **Urgency** | |  | | **Complexity** | | MEDIUM | |
|  | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Vulnerability** | XML External Entity (XXE) Processing | | | | | **ID** | 2015T114R-12 |
| **Brief Description** | | | | | | | |
| A Clickjacking vulnerability was detected. | | | | | | | |
| **Risk** | **D-Critical** | | **Category** | |  | | |
| **Assets** |  | | | | | | |
| **CVSS** |  | | | | | | |
| **CVSS Vector** |  | | | | | | |
| **Description** | | | | | | | |
| An XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, port scanning from the perspective of the machine where the parser is located, and other system impacts. | | | | | | | |
| Impact | | | | | | | |
|  | | | | | | | |
| **Recommendation** | | | | | | | |
| **Urgency** | |  | | **Complexity** | | LOW | |
|  | | | | | | | |