# SWAT Detailed Report

All Findings - acc.i-talent.eu

2020-01-24

## Report Information

| | |
|---|---|
| **Report type** | SWAT Vulnerability |
| **Report ID** | FE82C07A26763B77A7BB10AEF7AF0804 |
| **Date report was created** | 2020-01-24 16:38 |
| **Timezone for dates** | GMT+1 |
| **Report created for** | e-Progress |
| **Report generated by** | Wouter Olde Weghuis |
| **Report filtering** | Report has filtering active and this can result in a report with partial findings.<br>Fixed: No |
| **Report template** | All Findings |
| **Findings sorted by** | CVSS |

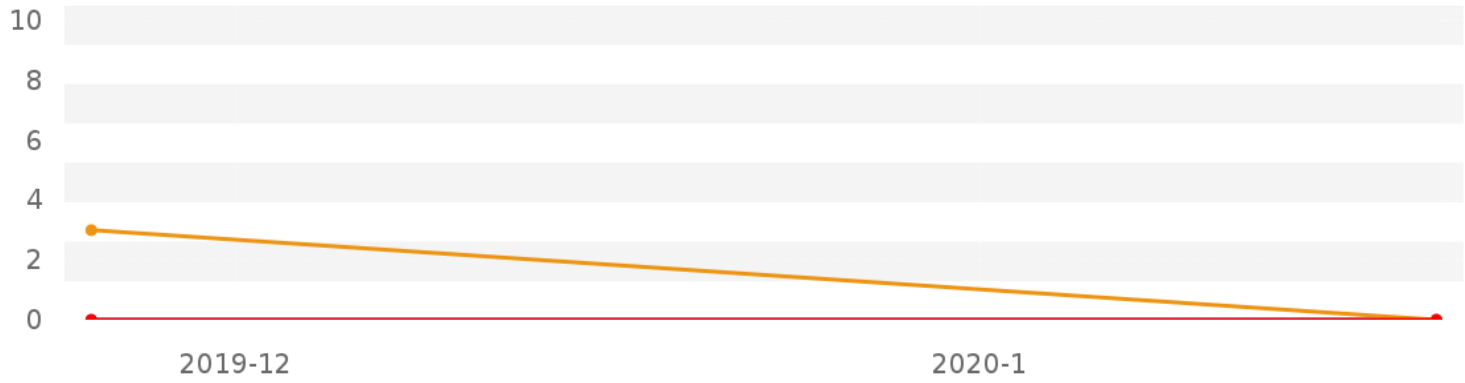
## Executive Summary

### Risk Level Overview

| | |
|---|---|
| High risk | 0 findings |
| Medium risk | 0 findings |
| Low risk | 0 findings |

### Trend

Number of findings for each risk level between 2019-01-24 and 2020-01-24




### OWASP Top 10

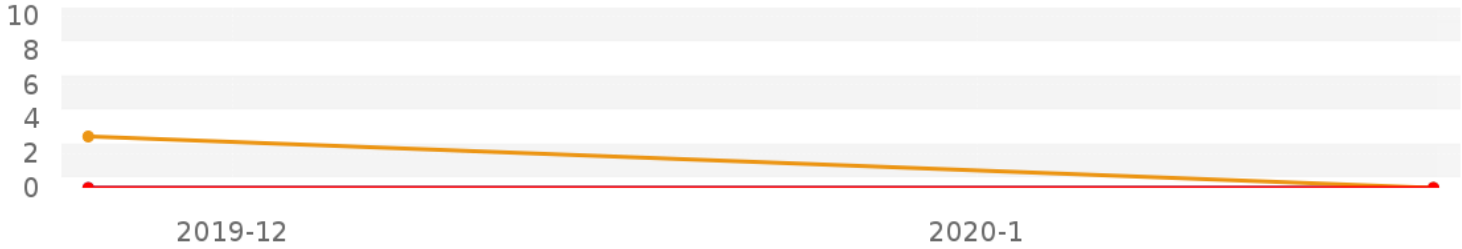
| | | |
|---|---|---|
| A01 | Injection | ✓ |
| A02 | Broken Authentication | ✓ |
| A03 | Sensitive Data Exposure | ✓ |
| A04 | XML External Entities (XXE) | ✓ |
| A05 | Broken Access Control | ✓ |
| A06 | Security Misconfiguration | ✓ |
| A07 | Cross-Site Scripting (XSS) | ✓ |
| A08 | Insecure Deserialization | ✓ |
| A09 | Using Components with Known Vulnerabilities | ✓ |
| A10 | Insufficient Logging&Monitoring | ✓ |

✓ No issues
✖ One or more issues

## SWAT Application Summary

### acc.i-talent.eu

| | | | |
|---|---|---|---|
| High risk | 0 | Average CVSS score | 0.0 |
| Medium risk | 0 | | |
| Low risk | 0 | | |




### OWASP Top 10

| | | |
|---|---|---|
| A01 | Injection | ✓ |
| A02 | Broken Authentication | ✓ |
| A03 | Sensitive Data Exposure | ✓ |
| A04 | XML External Entities (XXE) | ✓ |
| A05 | Broken Access Control | ✓ |
| A06 | Security Misconfiguration | ✓ |
| A07 | Cross-Site Scripting (XSS) | ✓ |
| A08 | Insecure Deserialization | ✓ |
| A09 | Using Components with Known Vulnerabilities | ✓ |
| A10 | Insufficient Logging&Monitoring | ✓ |

✓ No issues
✖ One or more issues

## Component "Bootstrap 3.4.1" End-of-Life

| | |
|---|---|
| **Port** | 443/TCP - http |
| **Finding Id** | 346517787 |
| **Description** | The detected version 3.4.1 of Bootstrap is no longer supported by the vendor, as it has reached its end-of-life. This means that the component will (in addition to regular updates) not receive any security patches from the vendor.<br><br>Right now there is no vulnerabilities associated with this version of Bootstrap, however as no further updates will come to Bootstrap 3.x this may be a risk in the future.<br><br>For reference, please see:<br>https://github.com/twbs/release |
| **Solution** | Upgrade Bootstrap to the latest secure version. |
| **Category** | Not classified |
| **Recreation Flow** | 1. Visit the following URL:<br><br>https://pentest.portal.acc.i-talent.eu<br><br>2. Review the response HTML source code, and note the following line:<br><br><script type="text/javascript" src="/Scripts/dist/main.js?cacheVersion=1.24.0.79"></script><br><br>3. This line includes an End-of-Life version of Bootstrap, which has known security issues associated with it |
| **Explanation** | Attackers may be able to stage attacks using known vulnerabilities in the component, as the component will not receive any security updates. |
| **Age** | 56.0 days - First detected: 2019-11-29 07:26 |
| **Last detected** | 2019-11-29 15:18 |
| **Script Id** | 2000106 - Added: 2017-01-02 |
| **Attachments** | Bootstrap_3.4.1.PNG — See appendix 1.1 |

## Missing "X-XSS-Protection" Response Header

| | |
|---|---|
| **Port** | 443/TCP - http |
| **Finding Id** | 346323645 |
| **Description** | The HTTP X-XSS-Protection response header is a feature of Internet Explorer, Chrome and Safari that stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. Although these protections are largely unnecessary in modern browsers when sites implement a strong Content-Security-Policy that disables the use of inline JavaScript ('unsafe-inline'), they can still provide protection for users of older web browsers that do not yet support CSP.<br><br>Note that Internet Explorer 8 (which is end-of-life) misinterprets this directive and becomes more vulnerable should the header be set. As such, it might not always be the best idea to enable it. |
| **Solution** | Consider setting the 'X-XSS-Protection:' HTTP response header on all responses for browsers that are not IE8. |
| **Category** | Not classified |
| **Recreation Flow** | 1. Navigate to the following URL:<br><br>https://pentest.performance.acc.i-talent.eu/not.found<br><br>2. Review the HTTP response headers:<br><br>HTTP/1.1 404 Not Found<br>Content-Type: text/html<br>Server: Microsoft-IIS/10.0<br>X-Powered-By: ASP.NET<br>Date: Thu, 28 Nov 2019 08:38:29 GMT<br>Connection: close<br>Content-Length: 1245<br><br>3. Note that the "X-XSS-Protection" header is not set |
| **Explanation** | Without the "X-XSS-Protection" response header, XSS attacks may be successfully executed. |
| **CWE** | **CWE-16** |
| **WASC** | **WASC-14** |
| **OWASP Top 10 2017** | **A06: Security Misconfiguration** |
| **OWASP Top 10 2013** | **A05: Security Misconfiguration** |
| **OWASP Top 10 2010** | A06: Security Misconfiguration |
| **OWASP Top 10 2004** | A10: Insecure configuration Management |
| **Age** | 57.0 days - First detected: 2019-11-28 07:18 |
| **Last detected** | 2019-11-29 07:18 |
| **Script Id** | 2000106 - Added: 2017-01-02 |

## Missing "X-Content-Type-Options: nosniff" Response Header

| | |
|---|---|
| **Port** | 443/TCP - http |
| **Finding Id** | 346323644 |
| **Description** | The X-Content-Type-Options response HTTP header is a marker used by the server to indicate that the MIME types advertised in the Content-Type headers should not be changed and be followed.<br>This enables one to opt-out of MIME type sniffing. |
| **Solution** | Consider setting the 'X-Content-Type-Options: nosniff' HTTP response header on all responses. |
| **Category** | Not classified |
| **Recreation Flow** | 1. Navigate to the following URL:<br><br>https://pentest.performance.acc.i-talent.eu/not.found<br><br>2. Note the server HTTP response headers:<br><br>HTTP/1.1 404 Not Found<br>Content-Type: text/html<br>Server: Microsoft-IIS/10.0<br>X-Powered-By: ASP.NET<br>Date: Thu, 28 Nov 2019 08:38:29 GMT<br>Connection: close<br>Content-Length: 1245<br><br>3. Note that the "X-Content-Type-Options: nosniff" header is not set |
| **Explanation** | An attacker could cause the web browser to interpret files as something else than declared by the content type. |
| **Age** | 57.0 days - First detected: 2019-11-28 07:18 |
| **Last detected** | 2019-11-29 07:18 |
| **Script Id** | 2000106 - Added: 2017-01-02 |

## HTTP Strict Transport Security not Configured

| | |
|---|---|
| **Port** | 443/TCP - http |
| **Finding Id** | 346323643 |
| **Description** | The web application accepts connections over encrypted HTTPS, but it does not instruct the client user agent to enforce HTTPS for future connections. This could by extension lead to that a user either directly or inadvertently accesses the application over unencrypted HTTP - a downgrade which could subsequently be exploited by an adjacent attacker.<br><br>For example, an attacker could trick the victim user agent to perform a request over HTTP, and then intercept all non-Secure client cookies.<br><br>Implementing HSTS require careful consideration of the security/convenience implications, as it would also prevent users from legitimately downgrading their connections. For reference, please see: https://tools.ietf.org/html/rfc6797 |
| **Solution** | Set the "Strict-Transport-Security" HTTP response header on all HTTPS responses. |
| **Category** | Not classified |
| **Recreation Flow** | 1. Navigate to the following URL:<br><br>https://pentest.performance.acc.i-talent.eu<br><br>2. Inspect the server HTTP response headers, and note that the "Strict-Transport-Security" HSTS header is not present<br><br>HTTP/1.1 302 Found<br>Cache-Control: private<br>Location: https://pentest.portal.acc.i-talent.eu/Portal/Account/DoLogin?returnUrl=https%3a%2f%2fpentest.performance.acc.i-talent.eu%2f<br>Server: Microsoft-IIS/10.0<br>X-AspNetMvc-Version: 5.2<br>X-AspNet-Version: 4.0.30319<br>X-Powered-By: ASP.NET<br>Date: Thu, 28 Nov 2019 08:35:42 GMT<br>Connection: close<br>Content-Length: 0 |
| **Explanation** | Inadvertently connecting once over an insecure connection could allow attackers to perform Man-in-the-Middle-Attacks. |
| **CWE** | CWE-311<br>CWE-523 |
| **WASC** | WASC-04 |
| **OWASP Top 10 2017** | A03: Sensitive Data Exposure |
| **OWASP Top 10 2013** | A06: Sensitive Data Exposure |
| **OWASP Top 10 2010** | A09: Insufficient Transport Layer Protection |
| **OWASP Top 10 2007** | A09: Insecure Communications |
| **Age** | 57.0 days - First detected: 2019-11-28 07:18 |
| **Last detected** | 2019-11-29 07:18 |
| **Script Id** | 2000106 - Added: 2017-01-02 |

## Missing "X-XSS-Protection" Response Header

| | |
|---|---|
| **Port** | 443/TCP - http |
| **Finding Id** | 346323637 |
| **Description** | The HTTP X-XSS-Protection response header is a feature of Internet Explorer, Chrome and Safari that stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. Although these protections are largely unnecessary in modern browsers when sites implement a strong Content-Security-Policy that disables the use of inline JavaScript ('unsafe-inline'), they can still provide protection for users of older web browsers that do not yet support CSP.<br><br>Note that Internet Explorer 8 (which is end-of-life) misinterprets this directive and becomes more vulnerable should the header be set. As such, it might not always be the best idea to enable it. |
| **Solution** | Consider setting the 'X-XSS-Protection:' HTTP response header on all responses for browsers that are not IE8. |
| **Category** | Not classified |
| **Recreation Flow** | 1. Navigate to the following URL:<br><br>https://pentest.portal.acc.i-talent.eu/Portal/Account/DoLogin?ReturnUrl=%2F<br><br>2. Review the HTTP response headers:<br><br>HTTP/1.1 200 OK<br>Cache-Control: no-cache, no-store, must-revalidate<br>Pragma: no-cache<br>Content-Type: text/html; charset=utf-8<br>Expires: -1<br>Vary: Accept-Encoding<br>Server: Microsoft-IIS/10.0<br>X-AspNetMvc-Version: 5.2<br>X-Frame-Options: SAMEORIGIN<br>X-AspNet-Version: 4.0.30319<br>X-Powered-By: ASP.NET<br>Date: Thu, 28 Nov 2019 06:28:32 GMT<br>Connection: close<br>Content-Length: 6218<br><br>3. Note that the "X-XSS-Protection" header is not set |
| **Explanation** | Without the "X-XSS-Protection" response header, XSS attacks may be successfully executed. |
| **CWE** | CWE-16 |
| **WASC** | WASC-14 |
| **OWASP Top 10 2017** | A06: Security Misconfiguration |
| **OWASP Top 10 2013** | A05: Security Misconfiguration |
| **OWASP Top 10 2010** | A06: Security Misconfiguration |
| **OWASP Top 10 2004** | A10: Insecure configuration Management |
| **Age** | 57.0 days - First detected: 2019-11-27 20:20 |
| **Last detected** | 2019-11-29 07:24 |
| **Script Id** | 2000106 - Added: 2017-01-02 |

## Missing "X-Content-Type-Options: nosniff" Response Header

| | |
|---|---|
| **Port** | 443/TCP - http |
| **Finding Id** | 346323294 |
| **Description** | The X-Content-Type-Options response HTTP header is a marker used by the server to indicate that the MIME types advertised in the Content-Type headers should not be changed and be followed.<br>This enables one to opt-out of MIME type sniffing. |
| **Solution** | Consider setting the 'X-Content-Type-Options: nosniff' HTTP response header on all responses. |
| **Category** | Not classified |
| **Recreation Flow** | 1. Navigate to the following URL:<br><br>https://pentest.portal.acc.i-talent.eu/Portal/Account/DoLogin?ReturnUrl=%2F<br><br>2. Note the server HTTP response headers:<br><br>HTTP/1.1 200 OK<br>Cache-Control: no-cache, no-store, must-revalidate<br>Pragma: no-cache<br>Content-Type: text/html; charset=utf-8<br>Expires: -1<br>Vary: Accept-Encoding<br>Server: Microsoft-IIS/10.0<br>X-AspNetMvc-Version: 5.2<br>X-Frame-Options: SAMEORIGIN<br>X-AspNet-Version: 4.0.30319<br>X-Powered-By: ASP.NET<br>Date: Thu, 28 Nov 2019 06:28:32 GMT<br>Connection: close<br>Content-Length: 6218<br><br>3. Note that the "X-Content-Type-Options: nosniff" header is not set |
| **Explanation** | An attacker could cause the web browser to interpret files as something else than declared by the content type. |
| **Age** | 57.0 days - First detected: 2019-11-27 20:20 |
| **Last detected** | 2019-11-29 07:24 |
| **Script Id** | 2000106 - Added: 2017-01-02 |

## HTTP Strict Transport Security not Configured

| | |
|---|---|
| **Port** | 443/TCP - http |
| **Finding Id** | 346320318 |
| **Description** | The web application accepts connections over encrypted HTTPS, but it does not instruct the client user agent to enforce HTTPS for future connections. This could by extension lead to that a user either directly or inadvertently accesses the application over unencrypted HTTP - a downgrade which could subsequently be exploited by an adjacent attacker.<br><br>For example, an attacker could trick the victim user agent to perform a request over HTTP, and then intercept all non-Secure client cookies.<br><br>Implementing HSTS require careful consideration of the security/convenience implications, as it would also prevent users from legitimately downgrading their connections. For reference, please see: https://tools.ietf.org/html/rfc6797 |
| **Solution** | Set the "Strict-Transport-Security" HTTP response header on all HTTPS responses. |
| **Category** | Not classified |
| **Recreation Flow** | 1. Navigate to the following URL:<br><br>https://pentest.portal.acc.i-talent.eu/<br><br>2. Inspect the server HTTP response headers, and note that the "Strict-Transport-Security" HSTS header is not present<br><br>HTTP/1.1 302 Found<br>Cache-Control: private<br>Location: https://pentest.portal.acc.i-talent.eu/Portal/Account/DoLogin?ReturnUrl=%2F<br>Server: Microsoft-IIS/10.0<br>X-AspNetMvc-Version: 5.2<br>X-AspNet-Version: 4.0.30319<br>X-Powered-By: ASP.NET<br>Date: Thu, 28 Nov 2019 06:28:31 GMT<br>Connection: close<br>Content-Length: 0 |
| **Explanation** | Inadvertently connecting once over an insecure connection could allow attackers to perform Man-in-the-Middle-Attacks. |
| **CWE** | CWE-311<br>CWE-523 |
| **WASC** | WASC-04 |
| **OWASP Top 10 2017** | A03: Sensitive Data Exposure |
| **OWASP Top 10 2013** | A06: Sensitive Data Exposure |
| **OWASP Top 10 2010** | A09: Insufficient Transport Layer Protection |
| **OWASP Top 10 2007** | A09: Insecure Communications |
| **Age** | 57.0 days - First detected: 2019-11-27 20:20 |
| **Last detected** | 2019-11-29 07:24 |
| **Script Id** | 2000106 - Added: 2017-01-02 |

# Methodology

## Approach

The manual components of the testing in based on the methodology set forth in the OWASP Testing Guide to the extent where so is applicable, as well as takes influence in risk management guidelines as ISO27005, testing guidelines as published by the PCI SSC in regards to testing of PCI DSS environments, the OSSTMM, and NIST publications on penetration testing.

The scope is established, application content, behavior and context mapped, interactions identified and then subjected to testing. Risks are noted and explored further.
All identified risks are scored according to business best practice using the Common Vulnerability Scoring System. When known vulnerabilities are discovered, their reference and CVE-id are included in the reporting.

Each risk identified by an analyst or the supporting technical platforms and scanners is manually verified, recreated and documented, including explanations of its impact and a recommendations in regards to problem resolution.

## Scope and Vectors

The scope is defined as the targets of the tests. The locations of the tester and the routes from the tester to the scope is defined as the vector. The scope was supplied by the customer and contains the application(s) to be tested.

**Application**

acc.i-talent.eu

The vector is shown below. The list was supplied by Outpost24 AB and contains the remote IP addresses which were used during the test:

| Vectors | IP Addresses |
| --- | --- |
| Public Internet | |

## Activities

The following activities were executed during the testing:

| | |
|---|---|
| **Network and Application Scans** | Determining host, IP, route and location information of the network systems related to the applications to be tested, as well as initial configuration of the continuous monitoring. |
| **Component and Service Identification** | After the manners to communicate with the server are inventoried, it will be attempted to identify which services, operating systems, patches and components are running on the targets to be tested. Further it will be inspected if security devices or redundant systems can be detected. |
| **Vulnerability Research and Configuration Inspection** | The detected components will be checked against known vulnerabilities, problems, and configuration issues. |
| **Application Exploiting** | Testing will attempt to tamper/exploit the detected weaknesses. If such tests are expected to endanger the confidentiality, integrity, or availability of data, then such testing may only occur in consultation with the customer. |
| **Analysis and Reporting** | The vulnerability information, evidence, and other data collected are analyzed and compiled into the vulnerability report. |

## Explicit Exceptions

The following tests were not executed during the testing:

| | |
|---|---|
| **Denial of Service Attacks** | The result of a denial of service attack might cause the application to cease normal behavior. Therefore, attacks of this type will not be executed, unless explicitly requested by the customer. |
| **Social Engineering** | In social engineering, an adversary attempts to gain access or otherwise manipulate an application by attacking the people and employees with privileged access, e.g. by enticing them to divulge information. |
| **Physical Security** | Physical security is the protection of personnel, hardware, programs, networks, and data from physical circumstances and events that could cause serious loss or damage to an enterprise, agency, or institution. |

## Purpose

Outpost24 AB has executed web application tests for the customer.

The objective of these tests was to get an impression about the information security of the web applications and the environment. Based on the test results Outpost24 AB will compile a vulnerability report, and give recommendations for improvements where applicable.

The end result will be that the customer will gain insight about the robustness and security of their applications. Conclusions will be provided with clear suggestions for operational solutions and managerial focus, leading to heightened IT security.

## OWASP Top 10 2017 Description

The OWASP Top Ten is a powerful awareness document for web application security, which represents a broad consensus about what the most critical web application security flaws are.

| | | |
|---|---|---|
| A01 | Injection | Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization. |
| A02 | Broken Authentication | Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently. |
| A03 | Sensitive Data Exposure | Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser. |
| A04 | XML External Entities (XXE) | Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks. |
| A05 | Broken Access Control | Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc. |
| A06 | Security Misconfiguration | Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion. |
| A07 | Cross-Site Scripting (XSS) | XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites. |
| A08 | Insecure Deserialization | Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks. |
| A09 | Using Components with Known Vulnerabilities | Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts. |
| A10 | Insufficient Logging&Monitoring | Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring. |

# Common Vulnerability Scoring System (CVSS) v2 Description

The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities. CVSS attempts to assign severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to threat. Scores are calculated based on a formula that depends on several metrics that approximate ease of exploit and the impact of exploit. Scores range from 0 to 10, with 10 being the most severe. While many utilize only the CVSS Base score for determining severity, Temporal and Environmental scores also exist, to factor in availability of mitigation and how widespread vulnerable systems are within an organization, respectively.

## Access Complexity (AC)

| Metric Value | Description |
|---|---|
| High (H) | Specialized access conditions exist. For example:<br>In most configurations, the attacking party must already have elevated privileges or spoof additional systems in addition to the attacking system (e.g., DNS hijacking).<br>The attack depends on social engineering methods that would be easily detected by knowledgeable people. For example, the victim must perform several suspicious or atypical actions.<br>The vulnerable configuration is seen very rarely in practice. |
| Medium (M) | The access conditions are somewhat specialized; the following are examples:<br>The attacking party is limited to a group of systems or users at some level of authorization, possibly untrusted.<br>Some information must be gathered before a successful attack can be launched.<br>The affected configuration is non-default, and is not commonly configured (e.g., vulnerability present when a server performs user account authentication via a specific scheme, but not present for another authentication scheme).<br>The attack requires a small amount of social engineering that might occasionally fool cautious users (e.g., phishing attacks that modify a web browser's status bar to show a false link, having to be on someone's "buddy" list before sending an IM exploit). |
| Low (L) | Specialized access conditions or extenuating circumstances do not exist. The following are examples:<br>The affected product typically requires access to a wide range of systems and users, possibly anonymous and untrusted (e.g., Internet-facing web or mail server).<br>The affected configuration is default or ubiquitous.<br>The attack can be performed manually and requires little skill or additional information gathering.<br>The "race condition" is a lazy one (i.e., it is technically a race but easily winnable). |

## Access Vector (AV)

| Metric Value | Description |
|---|---|
| Local (L) | Vulnerability exploitable with only local access requires the attacker to have either physical access to the vulnerable system or a local (shell) account. Examples of locally exploitable vulnerabilities are peripheral attacks such as Firewire/USB DMA attacks, and local privilege escalations (e.g., sudo). |
| Adjacent Network (A) | Vulnerability exploitable with adjacent network access requires the attacker to have access to either the broadcast or collision domain of the vulnerable software. Examples of local networks include local IP subnet, Bluetooth, IEEE 802.11, and local Ethernet segment. |
| Network (N) | A vulnerability exploitable with network access means the vulnerable software is bound to the network stack and the attacker does not require local network access or local access. Such vulnerability is often termed "remotely exploitable". An example of a network attack is an RPC buffer overflow. |

## Authentication (Au)

| Metric Value | Description |
|---|---|
| Multiple (M) | Exploiting the vulnerability requires that the attacker authenticate two or more times, even if the same credentials are used each time. An example is an attacker authenticating to an operating system in addition to providing credentials to access an application hosted on that system. |
| Single (S) | One instance of authentication is required to access and exploit the vulnerability. |
| None (N) | Authentication is not required to access and exploit the vulnerability. |

## Confidentiality Impact (C)

| Metric Value | Description |
|---|---|
| Partial (P) | There is considerable informational disclosure. Access to some system files is possible, but the attacker does not have control over what is obtained, or the scope of the loss is constrained. An example is a vulnerability that divulges only certain tables in a database. |
| Complete (C) | There is total information disclosure, resulting in all system files being revealed. The attacker is able to read all of the system's data (memory, files, etc.) |
| None (N) | There is no impact to the confidentiality of the system. |

## Integrity Impact (I)

| Metric Value | Description |
|---|---|
| Partial (P) | Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited. For example, system or application files may be overwritten or modified, but either the attacker has no control over which files are affected or the attacker can modify files within only a limited context or scope. |

| | |
|---|---|
| Complete (C) | There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised. The attacker is able to modify any files on the target system. |
| None (N) | There is no impact to the integrity of the system. |

## Availability Impact (A)

| Metric Value | Description |
|---|---|
| Partial (P) | There is reduced performance or interruptions in resource availability. An example is a network-based flood attack that permits a limited number of successful connections to an Internet service. |
| Complete (C) | There is total information disclosure, resulting in all system files being revealed. The attacker is able to read all of the system's data (memory, files, etc.) |
| None (N) | There is no impact to the availability of the system. |

## Test case appendix - SWAT

SWAT is a hybrid service delivery covering automated monitoring and web application scanning as well as at least quarterly penetration testing, including application logics, of web applications under service.

The test-cases are oriented around the OWASP TESTING GUIDE, and for the application the following controls has been performed. Note that a control will be marked as audited either if found present and audited, or were found not present and hence not auditable - This to show that the application has been audited for this class of risks.

| Test Activities and Descriptions | OWASP testing guide | Audit note |
|---|---|---|
| **Information Gathering** | | |
| 4.2.1 Conduct Search Engine Discovery and Reconnaissance for Information Leakage (OTG-INFO-001) | OTG-INFO-001 | Not in scope |
| Search for: | | Not in scope |
| Network diagrams and configurations | | Not in scope |
| Archived posts and emails by administrators and other key staff | | Not in scope |
| Log on procedures and username formats | | Not in scope |
| Usernames and passwords | | Not in scope |
| Error message content | | Not in scope |
| Development, test, UAT and staging versions of the website | | Not in scope |
| 4.2.2 Fingerprint Web Server (OTG-INFO-002) | OTG-INFO-002 | Audited |
| Determine web server software and (if possible) version | | Audited |
| 4.2.3 Review Webserver Metafiles for Information Leakage (OTG-INFO-003) | OTG-INFO-003 | Audited |
| Locate the robots.txt file(s) and review their content | | Audited |
| 4.2.4 Enumerate Applications on Webserver (OTG-INFO-004) | OTG-INFO-004 | Audited |
| Enumerate and identify all available applications | | Audited |
| Check each available web server for applications | | Audited |
| Create a list of possible virtual hosts and check if they are accepted as such (DNS enumeration, rDNS, identify domains which map to the same IP) | | Not in scope |
| Check if applications are situated in a directory other than root by: Spider server, Forceful browsing, Search engines, etc. | | Audited |
| 4.2.5 Review Webpage Comments and Metadata for Information Leakage (OTG-INFO-005) | OTG-INFO-005 | Audited |
| Review all source comments and note useful information. | | Audited |
| 4.2.6 Identify application entry points (OTG-INFO-006) | OTG-INFO-006 | Audited |
| Identify entry points / gates / input vectors: | | Audited |
| - Query (GET) parameters | | Audited |
| - Body parameters | | Audited |
| - Cookies | | Audited |
| - Request headers | | Audited |
| - REST-style parameters | | Audited |
| Review regular responses | | Audited |
| - Where are cookies set? | | Audited |
| - Does the application fail during normal operation (i.e. HTTP 500, 404) | | Audited |
| - Load balancers in place (might mean that exploits have to be repeated until vulnerable back-end server is hit) | | Audited |

| | | |
|---|---|---|
| 4.2.7 Map execution paths through application (OTG-INFO-007) | OTG-INFO-007 | Audited |
| Map the application structure and paths | | Audited |
| Note what parts of the application might share server-side components and code | | Audited |
| Note which parts might contain unique functionality | | Audited |
| Note which functionality might not be exposed | | Audited |
| 4.2.8 Fingerprint Web Application Framework (OTG-INFO-008) | OTG-INFO-008 | Audited |
| For each identified web application, determine if it is based upon one or multiple frameworks | | Audited |
| For each framework, determine the name and vendor, as well as the version | | Audited |
| 4.2.9 Fingerprint Web Application (OTG-INFO-009) | OTG-INFO-009 | Audited |
| - For each identified web application, determine if the application is (or is based upon) a standard application | | Audited |
| - Determine the name and vendor of the application, as well as the version | | Audited |
| 4.2.10 Map Application Architecture (OTG-INFO-010) | OTG-INFO-010 | Audited |
| - Determine if any firewalls or web application firewalls are in place | | Audited |
| - Determine if a reverse proxy, cache, or load balancer is in use | | Audited |
| - Determine if there are multiple web servers handling requests | | Audited |
| - Determine the name, vendor, and version for each component or node | | Audited |
| - Draft a network topology map from the determined structure | | Audited |
| - Determine if URL rewrites can lead to cache poisoning – Not OTG testcase | | Audited |
| 4.3 Configuration and Deployment Management Testing | | |
| 4.3.1 Test Network/Infrastructure Configuration (OTG-CONFIG-001) | OTG-CONFIG-001 | |
| - Leverage the map established in 4.2.10 Map Application Architecture and check for known vulnerabilities | | Not in scope |
| - Determine the location of administrative interface and test for configuration issues | | Audited |
| 4.3.2 Test Application Platform Configuration (OTG-CONFIG-002) | OTG-CONFIG-002 | Audited |
| - Enumerate known files and directories, and determine if any platform-provided components are vulnerable | | Audited |
| - Review source comments for useful information | | Audited |
| - Determine how error reporting is handled | | Audited |
| 4.3.3 Test File Extensions Handling for Sensitive Information (OTG-CONFIG-003) | OTG-CONFIG-003 | Audited |
| - Assert how the web server presents files according to their file extension | | Audited |
| - Determine if any server-side code can be discovered by forceful browsing | | Audited |
| - Determine if file upload or file access restrictions based on file extensions can be circumvented | | Audited |
| 4.3.4 Review Old, Backup and Unreferenced Files for Sensitive Information (OTG-CONFIG-004) | OTG-CONFIG-004 | Audited |
| Attempt to reveal unreferenced files through: | | Audited |
| - Forceful browsing, "blind guessing" | | Audited |
| - Server misconfigurations or vulnerabilities (such as enabled directory listing, or IIS short name) | | Audited |
| - Search engines and public information | | Audited |
| - File name bypass (using IIS short name to circumvent filter) | | Audited |
| - HTML (or other) source comments | | Audited |
| - Extrapolating from detected or derived naming schemes (e.g. "/2016/08" => "/2016/07") | | Audited |

| | | |
|---|---|---|
| 4.3.5 Enumerate Infrastructure and Application Admin Interfaces (OTG-CONFIG-005) | OTG-CONFIG-005 | Audited |
| - Determine the location of available administrative interfaces | | Audited |
| - Determine whether or not the administrative interfaces performs proper checks in regards to authentication and authorisation | | Audited |
| - Determine is default credentials are in use | | Audited |
| 4.3.6 Test HTTP Methods (OTG-CONFIG-006) | OTG-CONFIG-006 | Audited |
| - Determine which HTTP methods are supported, and to what extent | | Audited |
| - Determine if TRACE is enabled (XST) | | Audited |
| - Determine if regular (such as HEAD) or arbitrary (such as ASDF) methods can be used in order to bypass authorisation or cause other issues | | Audited |
| 4.3.7 Test HTTP Strict Transport Security (OTG-CONFIG-007) | OTG-CONFIG-007 | Audited |
| - Determine if HSTS is properly configured for the application | | Audited |
| - Determine whether or not HSTS preloading is properly configured | | Audited |
| 4.3.8 Test RIA cross domain policy (OTG-CONFIG-008) | OTG-CONFIG-008 | Audited |
| - Determine if crossdomain.xml and clientaccesspolicy.xml exists, and if so, if they are properly set up | | Audited |
| 4.4 Identity Management Testing | | |
| 4.4.1 Test Role Definitions (OTG-IDENT-001) | OTG-IDENT-001 | Audited |
| - Map user roles and their intended permissions for various objects | | Audited |
| - Verify that user roles can not exceed their intended permissions | | Audited |
| 4.4.2 Test User Registration Process (OTG-IDENT-002) | OTG-IDENT-002 | Audited |
| - Verify that the registration requirements are properly implemented and can not be circumvented or altered | | Audited |
| - Verify that the registration process aligns with the business requirements | | Audited |
| 4.4.3 Test Account Provisioning Process (OTG-IDENT-003) | OTG-IDENT-003 | Audited |
| Determine which accounts or user roles may create other accounts | | Audited |
| Determine if the account creation process aligns with business and security requirements: | | Not in scope |
| Is there any verification, vetting and authorization of provisioning requests? | | Not in scope |
| Is there any verification, vetting and authorization of de-provisioning requests? | | Not in scope |
| Can an administrator provision other administrators or just users? | | Audited |
| Can an administrator or other user provision accounts with privileges greater than their own? | | Audited |
| Can an administrator or user de-provision themselves? | | Audited |
| How are the files or resources owned by the de-provisioned user managed? Are they deleted? Is access transferred? | | Not in scope |
| 4.4.4 Testing for Account Enumeration and Guessable User Account (OTG-IDENT-004) | OTG-IDENT-004 | Audited |
| Determine if it is possible to enumerate user accounts: | | Audited |
| - Log in as known user with known password | | Audited |
| - Log in as known user with the wrong password | | Audited |
| - Log in as non-existing user with wrong password | | Audited |
| - Find other entry points accepting user name or user reference input and test them as well, e.g. password reset | | Audited |
| 4.4.5 Testing for Weak or unenforced username policy (OTG-IDENT-005) | OTG-IDENT-005 | Audited |
| - Determine whether or not there is a naming scheme in place for usernames | | Audited |

| | | |
|---|---|---|
| - Evaluate application response in regards to usernames following or breaking the scheme | | Audited |

### 4.5 Authentication Testing

| | | |
|---|---|---|
| **4.5.1 Testing for Credentials Transported over an Encrypted Channel (OTG-AUTHN-001)** | OTG-AUTHN-001 | Audited |
| - Assert whether or not all credentials are transmitted over an encrypted channel | | Audited |
| - Test if credentials are accepted over plaintext connections | | Audited |
| **4.5.2 Testing for default credentials (OTG-AUTHN-002)** | OTG-AUTHN-002 | Audited |
| - Determine if access can be achieved using standard credentials | | Audited |
| - Determine if a common or guessable set of credentials are in use | | Audited |
| - Determine if a default or guessable password is set for new accounts | | Audited |
| **4.5.3 Testing for Weak lock out mechanism (OTG-AUTHN-003)** | OTG-AUTHN-003 | Audited |
| - Determine if password brute forcing is possible (lacking automation protection) | | Audited |
| - Determine if there is an account lockout in place, and the boundaries associated with it | | Audited |
| - Determine if the lockout can be circumvented | | Audited |
| **4.5.4 Testing for bypassing authentication schema (OTG-AUTHN-004)** | OTG-AUTHN-004 | Audited |
| Determine if authentication can be bypassed by: | | Audited |
| - Forced browsing, direct navigation | | Audited |
| - Parameter or cookie modification | | Audited |
| - Session token prediction | | Audited |
| - Injection vulnerabilities (such as SQLi) | | Audited |
| **4.5.5 Test remember password functionality (OTG-AUTHN-005)** | OTG-AUTHN-005 | Audited |
| Determine if there are any sensitive fields with autocomplete=on set | | Audited |
| Assert whether or not the application has a "remember me"-function. If so: | | Audited |
| - Determine how the feature is implemented and how it functions | | Audited |
| - Determine if any sensitive data is stored client-side (perhaps in a cookie) | | Audited |
| Verify that credentials are only sent when authenticating, not for each request | | Audited |
| **4.5.6 Testing for Browser cache weakness (OTG-AUTHN-006)** | OTG-AUTHN-006 | Audited |
| - Determine if user agents are allowed to store sensitive documents in the history storage | | Audited |
| - Determine if user agents are allowed to cache sensitive documents | | Audited |
| **4.5.7 Testing for Weak password policy (OTG-AUTHN-007)** | OTG-AUTHN-007 | Audited |
| Determine the specifics of the in-use password policy | | Audited |
| Assert whether or not users are able to (if willing) create strong passwords given the password policy | | Audited |
| Assert whether or not users are able to create weak passwords: | | Audited |
| - Character set requirements - what sets must be present? | | Audited |
| - Age requirements - how old can a password be? How often must it be changed? | | Not in scope |
| - Change requirements - when can the password be changed? How often can it be changed? | | Not in scope |
| - Reuse requirements - can old passwords be reused? How many old passwords does the application keep track of? | | Not in scope |
| - Difference requirements - how different must two passwords be in order to be accepted? Are any comparisons done at all? | | Not in scope |
| - Dictionary words - can dictionary words, or easily guessable strings such as the username or first name be present in the new password? | | Audited |

| | | |
|---|---|---|
| 4.5.8 Testing for Weak security question/answer (OTG-AUTHN-008) | OTG-AUTHN-008 | Audited |
| Check whether or not answers to pre-generated security questions: | | Audited |
| - Can be known by family members or friends (e.g. date of birth) | | Audited |
| - Can easily be guessed (e.g. favourite colour) | | Audited |
| - Can be publicly discovered (e.g. favourite movie, listed on Facebook) | | Audited |
| Check whether or not self-generated questions can be weak ("What is 1 + 1?") | | Audited |
| Check whether or not secret question answer can be found by brute force | | Audited |
| 4.5.9 Testing for weak password change or reset functionalities (OTG-AUTHN-009) | OTG-AUTHN-009 | Audited |
| - Determine if one user can change the password of another user (unless this is expected, e.g. administrator) | | Audited |
| - Determine if existing password reset functionality can be leveraged to change the password of other user accounts | | Audited |
| - Determine if the password reset functionality has any flaws, e.g. guessable tokens | | Audited |
| - Determine whether or not the password change or reset functions can be attacked via CSRF or similar vectors | | Audited |
| 4.5.10 Testing for Weaker authentication in alternative channel (OTG-AUTHN-010) | OTG-AUTHN-010 | Audited |
| - Identify and understand the primary authentication method and channel | | Audited |
| - Identify other authentication channels and map their scope | | Audited |
| - Determine if the alternative channels undermine the primary channel | | Audited |
| 4.6 Authorization Testing | | |
| 4.6.1 Testing Directory traversal/file include (OTG-AUTHZ-001) | OTG-AUTHZ-001 | Audited |
| - From the list of entry points, determine which could potentially be used to refer to local or remote resources | | Audited |
| - For these entry points, determine whether or not directory traversal or file inclusion can occur | | Audited |
| 4.6.2 Testing for bypassing authorization schema (OTG-AUTHZ-002) | OTG-AUTHZ-002 | Audited |
| For each unique role or privilege, assert whether or not: | | Audited |
| - It is possible to access a restricted resource without authorizing | | Audited |
| - It is possible to access a restricted resource after logging out | | Audited |
| - If is possible to access a restricted resource using an unauthorised account (lacking the Tested privilege) | | Audited |
| Determine whether or not there are flaws in the administrative functionality, using the same checks | | Audited |
| 4.6.3 Testing for Privilege Escalation (OTG-AUTHZ-003) | OTG-AUTHZ-003 | Audited |
| - For all functionality associated with sessions, or specifically assigned privileges, determine whether or not it is possible to access or modify it using an unauthorised account | | Audited |
| - Determine if the authorisation flaw can be used to escalate privileges | | Audited |
| 4.6.4 Testing for Insecure Direct Object References (OTG-AUTHZ-004) | OTG-AUTHZ-004 | Audited |
| - Enumerate all object references exposed throughout the application | | Audited |
| - Determine if these references can be altered to access data not intended for the current user | | Audited |
| 4.7 Session Management Testing | | |
| 4.7.1 Testing for Bypassing Session Management Schema (OTG-SESS-001) | OTG-SESS-001 | Audited |
| Enumerate all cookies set by the application, and determine: | | Audited |

| | | |
|---|---|---|
| - How many cookies are set? | | Audited |
| - Which cookies could have value to an attacker? | | Audited |
| - Which parts of the application generate or modify the cookies? | | Audited |
| - Which parts of the application requires cookies to be accessed? | | Audited |
| - Which subset of cookies are Tested? Which cookies can be discarded? | | Audited |
| - Whether or not the HTTPOnly and Secure flags are set for all cookies. | | Audited |
| - Whether or not cookies are (or can be) sent over an unencrypted channel. | | Audited |
| - Which cookies are temporary, and which are permanent | | Audited |
| - What HTTP/1.1 and HTTP/1.0 Cache-Control settings are used to protect cookies | | Audited |
| Analysis: | | Audited |
| - Determine if sensitive data is exposed through the cookie | | Audited |
| - Determine if there is any obfuscation in place of the cookie name or value | | Audited |
| - Determine if there are any patterns to the cookie data structure | | Audited |
| - Are the Session IDs provably random in nature? Can the resulting values be reproduced? | | Audited |
| - Do the same input conditions produce the same ID on a subsequent run? | | Audited |
| - Are the Session IDs provably resistant to statistical or cryptanalysis? | | Audited |
| - What elements of the Session IDs are time-linked? | | Audited |
| - What portions of the Session IDs are predictable? | | Audited |
| - Can the next ID be deduced, given full knowledge of the generation algorithm and previous IDs? | | Audited |
| - Does the cookie have sufficient entropy and unpredictability? | | Audited |
| - Is the cookie tamper resistant? Will the application reject modified cookies? | | Audited |
| - Does the cookie expire within a sane time period? | | Audited |
| Determine if it is feasible to gain access to a valid cookie by brute force | | Audited |
| **4.7.2 Testing for Cookies attributes (OTG-SESS-002)** | OTG-SESS-002 | Audited |
| - Determine whether or not the cookie attributes (HTTPOnly, Secure, Domain, Path) are properly set | | Audited |
| **4.7.3 Testing for Session Fixation (OTG-SESS-003)** | OTG-SESS-003 | Audited |
| - Determine whether or not a fresh session token (cookie) is set upon successful authentication. | | Audited |
| **4.7.4 Testing for Exposed Session Variables (OTG-SESS-004)** | OTG-SESS-004 | Audited |
| Assert whether or not the session tokens (cookies) are always transmitted securely | | Audited |
| Determine if new temporary tokens are generated for HTTP requests, or if leaked tokens can be re-used | | Audited |
| Determine if the caching directives provide sufficient protection | | Audited |
| Determine if any credentials or session tokens are transmitted as query parameter | | Audited |
| **4.7.5 Testing for Cross Site Request Forgery (CSRF) (OTG-SESS-005)** | OTG-SESS-005 | Audited |
| - For each unique request or function call, establish whether or not it can be triggered via CSRF, and whether or not that has any impact | | Audited |
| **4.7.6 Testing for logout functionality (OTG-SESS-006)** | OTG-SESS-006 | Audited |
| - Determine if the application features a logout function | | Audited |
| - Determine if the logout function properly terminates the session client-side | | Audited |
| - Determine if the logout function properly terminates the session server-side | | Audited |
| - Determine whether or not inactive sessions are terminated after a certain period of time | | Audited |
| - Assert whether or not it is possible to invalidate all user sessions (if multiple sessions are allowed) | | Audited |
| - If SSO, determine if there is a single sign-off implemented | | Audited |

| | | |
|---|---|---|
| 4.7.7 Test Session Timeout (OTG-SESS-007) | OTG-SESS-007 | Audited |
| - Determine whether or not an inactive session expires, and if so, the specific duration | | Audited |
| - Assert if the session is invalidated by the client, by the server, or both | | Audited |
| 4.7.8 Testing for Session puzzling (OTG-SESS-008) | OTG-SESS-008 | Audited |
| - Enumerate what session information is set where | | Audited |
| - Assert if it is possible to gain or escalate privileges by leveraging ("puzzling" together) a partial session | | Audited |
| 4.8 Input Validation Testing | | |
| 4.8.1 Testing for Reflected Cross Site Scripting (OTG-INPVAL-001) | OTG-INPVAL-001 | Audited |
| - Identify and test entry points which have the potential to echo user input for content and script injection issues | | Audited |
| 4.8.2 Testing for Stored Cross Site Scripting (OTG-INPVAL-002) | OTG-INPVAL-002 | Audited |
| - Identify and test entry points which have the potential to echo user input for content and script injection issues | | Audited |
| 4.8.3 Testing for HTTP Verb Tampering (OTG-INPVAL-003) | OTG-INPVAL-003 | Audited |
| - Determine what HTTP methods are supported by the application | | Audited |
| - If methods other than GET+POST are accepted, determine whether or not they are in use | | Audited |
| - Establish whether or not authentication and authorisation is properly implemented for the non-standard HTTP methods | | Audited |
| 4.8.4 Testing for HTTP Parameter pollution (OTG-INPVAL-004) | OTG-INPVAL-004 | Audited |
| - Determine if setting two parameters with identical name has any impact on the server response or filter validation | | Audited |
| 4.8.5 Testing for SQL Injection (OTG-INPVAL-005) | OTG-INPVAL-005 | Audited |
| - Identify and test entry points which have potential database interaction for injection issues | | Audited |
| 4.8.6 Testing for LDAP Injection (OTG-INPVAL-006) | OTG-INPVAL-006 | Audited |
| - Identify and test entry points which have potential LDAP interaction for injection issues | | Audited |
| 4.8.7 Testing for ORM Injection (OTG-INPVAL-007) | OTG-INPVAL-007 | Audited |
| - Identify and test entry points which have potential database interaction for injection issues | | Audited |
| 4.8.8 Testing for XML Injection (OTG-INPVAL-008) | OTG-INPVAL-008 | Audited |
| - Identify and test entry points which might be handled by XML parsers for injection issues | | Audited |
| 4.8.9 Testing for SSI Injection (OTG-INPVAL-009) | OTG-INPVAL-009 | Audited |
| - Identify and test entry points which have the potential to echo user input for SSI injection issues | | Audited |
| 4.8.10 Testing for XPath Injection (OTG-INPVAL-010) | OTG-INPVAL-010 | Audited |
| - Identify and test entry points which might be a part of an XPath expression for injection issues | | Audited |
| 4.8.11 IMAP/SMTP Injection (OTG-INPVAL-011) | OTG-INPVAL-011 | Audited |
| - Identify and test entry points that may (directly or indirectly) be used as parameters related to email handling | | Audited |

| | | |
|---|---|---|
| 4.8.12 Testing for Code Injection (OTG-INPVAL-012) | OTG-INPVAL-012 | Audited |
| - Identify and test entry points which could potentially evaluate the input as code or commands | | Audited |
| 4.8.13 Testing for Command Injection (OTG-INPVAL-013) | OTG-INPVAL-013 | Audited |
| - Identify and test entry points which could potentially evaluate the input as operating system commands | | Audited |
| 4.8.14 Testing for Buffer overflow (OTG-INPVAL-014) | OTG-INPVAL-014 | Audited |
| - Determine whether or not heap and stack overflow can be achieved by submitting larger input data than expected to entry points | | Audited |
| - Determine if format string expressions are evaluated | | Audited |
| 4.8.15 Testing for incubated vulnerabilities (OTG-INPVAL-015) | OTG-INPVAL-015 | Audited |
| Identify controls that can be leveraged in order to stage a new attack, and assert the possibility of doing so | | Audited |
| 4.8.16 Testing for HTTP Splitting/Smuggling (OTG-INPVAL-016) | OTG-INPVAL-016 | Audited |
| - Assert if HTTP request splitting is possible by leveraging echoed values present in the HTTP response header section | | Audited |
| - Assert if HTTP request smuggling is possible in the target environment | | Audited |
| 4.8.17 Testing for HTTP Incoming Requests (OTG-INPVAL-017) | OTG-INPVAL-017 | Audited |
| - Review HTTP request/response interaction between the client and server | | Audited |
| 4.9 Testing for Error Handling | | |
| 4.9.1 Analysis of Error Codes (OTG-ERR-001) | OTG-ERR-001 | Audited |
| Review all error messages generated by activities | | Audited |
| Determine how the application responds to: | | Audited |
| - Resource not found or forbidden | | Audited |
| - Accessing application without credentials | | Audited |
| - Bad request | | Audited |
| - Methods not allowed, and methods not implemented | | Audited |
| - Request time-out | | Audited |
| 4.9.2 Analysis of Stack Traces (OTG-ERR-002) | OTG-ERR-002 | Audited |
| Review all error messages generated by testing | | Audited |
| For all input vectors, establish application behaviour in regards to: | | Audited |
| - Invalid input | | Audited |
| - Input that contains non-alphanumeric characters | | Audited |
| - Empty inputs | | Audited |
| - Too long inputs | | Audited |
| - Accessing application in an unexpected way (bypassing regular flow) | | Audited |
| 4.10 Testing for weak Cryptography | | |
| 4.10.1 Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection (OTG-CRYPST-001) | OTG-CRYPST-001 | Audited |
| - Determine if any sensitive (including credentials) data is transmitted in clear text | | Audited |
| - Determine if any weak SSL/TLS ciphers are in use, and if any weak protocols are in use | | Audited |
| - Assert whether or not BEAST, POODLE, HeartBleed, FREAK or CRIME is applicable | | Audited |
| - Determine if the certificate is signed by a recognized CA | | Audited |

| | | |
|---|---|---|
| - Determine if the certificate is valid | | Audited |
| - Determine if Surf Jacking and SSL Strip are applicable | | Audited |
| **4.10.2 Testing for Padding Oracle (OTG-CRYPST-002)** | OTG-CRYPST-002 | Audited |
| - Identify parameter values which may be encrypted, and determine whether or not a padding oracle is present in the receiving implementation | | Audited |
| **4.10.3 Testing for Sensitive information sent via unencrypted channels (OTG-CRYPST-003)** | OTG-CRYPST-003 | Audited |
| - Determine if any sensitive (including credentials) data is transmitted in clear text | | Audited |
| **4.11 Business Logic Testing** | | |
| **4.11.1 Test Business Logic Data Validation (OTG-BUSLOGIC-001)** | OTG-BUSLOGIC-001 | Audited |
| Determine how the application front-end and back-end validates data, and note any discrepancies | | Audited |
| Identify what assumptions the application makes about decision-relevant data, and determine if this can be leveraged | | Audited |
| **4.11.2 Test Ability to Forge Requests (OTG-BUSLOGIC-002)** | OTG-BUSLOGIC-002 | Audited |
| Attempt to enumerate functions and function-changing parameters by guessing for predictable names and by using project/application documentation | | Audited |
| - Identify interesting function and parameter names | | Audited |
| Forge HTTP request to leverage these parameters and functions, and determine if any impact can be established | | Audited |
| **4.11.3 Test Integrity Checks (OTG-BUSLOGIC-003)** | OTG-BUSLOGIC-003 | Audited |
| Identify controls that dynamically generate output based on some criteria, and determine how the functionality or parameters presented differs | | Audited |
| - For each different parameter or function, determine the impact of unexpected or unauthorised input or access | | Audited |
| Identify what data is accepted by the various components/functions, and determine if the business logic aligns with this | | Audited |
| **4.11.4 Test for Process Timing (OTG-BUSLOGIC-004)** | OTG-BUSLOGIC-004 | Audited |
| Determine if there is a meaningful difference in response time between various inputs, function calls, or results | | Audited |
| **4.11.5 Test Number of Times a Function Can be Used Limits (OTG-BUSLOGIC-005)** | OTG-BUSLOGIC-005 | Audited |
| - For each function with a call limit, determine if it is possible to circumvent the limit | | Audited |
| - For each function with no limit, determine if the lack of restriction can result in some form of impact | | Audited |
| **4.11.6 Testing for the Circumvention of Work Flows (OTG-BUSLOGIC-006)** | OTG-BUSLOGIC-006 | Audited |
| - Identify work flows and procedures within the application and determine if it is possible to navigate non linearly or skip steps | | Audited |
| **4.11.7 Test Defenses Against Application Mis-use (OTG-BUSLOGIC-007)** | OTG-BUSLOGIC-007 | Audited |
| Determine how the application handles abuse of intended functionality: | | Audited |
| - Rejecting input containing certain characters | | Audited |
| - Locking out an account temporarily after a number of authentication failures | | Audited |
| - Forced browsing | | Audited |
| - Bypassing presentation layer input validation | | Audited |
| - Multiple access control errors | | Audited |
| - Additional, duplicated or missing parameter names | | Audited |

| | | |
|---|---|---|
| - Multiple input validation or business logic verification failures with values that cannot be the result user mistakes or typos | | Audited |
| - Structured data (e.g. JSPN, XML) of an invalid format is received | | Audited |
| - Blatant cross-site scripting or SQL injection payloads are received | | Audited |
| - Using the application faster than one could do manually | | Audited |
| - Change in continental geo-location of a user | | Audited |
| - Change of user agent | | Audited |
| - Accessing a multi-stage business process in the wrong order | | Audited |
| - Large number of, or high rate of use of, application-specific functionality (e.g. voucher code submission, failed credit card payments, file uploads, file downloads, log outs, etc). | | Audited |
| **4.11.8 Test Upload of Unexpected File Types (OTG-BUSLOGIC-008)** | OTG-BUSLOGIC-008 | Audited |
| - For each file upload feature, determine whether or not only intended file types can be uploaded (both for file name, and actual file type) | | Audited |
| **4.11.9 Test Upload of Malicious Files (OTG-BUSLOGIC-009)** | OTG-BUSLOGIC-009 | Audited |
| - Determine which file types should be considered malicious within the context of the application | | Audited |
| - Upload the known "malicious" EICAR anti-malware test file and determine how the application responds | | Audited |
| **4.12 Client Side Testing** | | |
| **4.12.1 Testing for DOM based Cross Site Scripting (OTG-CLIENT-001)** | OTG-CLIENT-001 | Audited |
| - Enumerate objects that supply or are used as input to JavaScript functions, and determine if it is possible to cause attacker-supplied code to be evaluated | | Audited |
| **4.12.2 Testing for JavaScript Execution (OTG-CLIENT-002)** | OTG-CLIENT-002 | Audited |
| - Enumerate objects that supply or are used as input to JavaScript functions, and determine if it is possible to cause attacker-supplied code to be evaluated | | Audited |
| **4.12.3 Testing for HTML Injection (OTG-CLIENT-003)** | OTG-CLIENT-003 | Audited |
| - Enumerate objects that supply or are used as input to JavaScript functions, and determine if it is possible to inject HTML indistinguishable from site content | | Audited |
| **4.12.4 Testing for Client Side URL Redirect (OTG-CLIENT-004)** | OTG-CLIENT-004 | Audited |
| - Enumerate objects that supply or are used as input to JavaScript functions, and determine if it is possible to redirect the user to an arbitrary destination | | Audited |
| **4.12.5 Testing for CSS Injection (OTG-CLIENT-005)** | OTG-CLIENT-005 | Audited |
| - Enumerate objects used as input to dynamically generate CSS, and determine if it is possible to leverage the generation to cause an impact | | Audited |
| **4.12.6 Testing for Client Side Resource Manipulation (OTG-CLIENT-006)** | OTG-CLIENT-006 | Audited |
| Enumerate objects used to determine a URL, and assert whether or not this can be modified to load arbitrary content into the page | | Audited |
| **4.12.7 Test Cross Origin Resource Sharing (OTG-CLIENT-007)** | OTG-CLIENT-007 | Audited |
| - Determine if the application implements proper behaviour in regards to CORS, or if the policy in place is too permissive | | Audited |
| - Determine if XHR controls can be used to load arbitrary content by allowing the scope origin in the CORS headers | | Audited |
| **4.12.8 Testing for Cross Site Flashing (OTG-CLIENT-008)** | OTG-CLIENT-008 | Audited |
| - Determine which parameters are passed to the flash object, and whether or not they can be leveraged in order to inject code or alter the object logic | | Audited |
| - Determine whether or not the flash object loads remote flash objects, and whether or not any arbitrary object can be loaded | | Audited |

| | | |
|---|---|---|
| 4.12.9 Testing for Clickjacking (OTG-CLIENT-009) | OTG-CLIENT-009 | Audited |
| - Determine if it is possible to frame the target application | | Audited |
| - Determine if a malicious impact can be caused by framing the application | | Audited |
| 4.12.10 Testing WebSockets (OTG-CLIENT-010) | OTG-CLIENT-010 | Audited |
| - Determine whether or not WebSockets are in use | | Audited |
| - Determine if the origin is properly verified | | Audited |
| - Determine if the WS is secure | | Audited |
| - Determine if authentication is properly set up | | Audited |
| - Determine if proper authorisation is performed | | Audited |
| - Determine that proper input sanitisation is performed | | Audited |
| 4.12.11 Test Web Messaging (OTG-CLIENT-011) | OTG-CLIENT-011 | Audited |
| - Determine if any event listeners for Cross Document Messaging are implemented, and if they can be leveraged in order to cause an impact | | Audited |
| 4.12.12 Test Local Storage (OTG-CLIENT-012) | OTG-CLIENT-012 | Audited |
| - Enumerate controls taking their input from either localStorage or sessionStorage | | Audited |
| - Determine if an impact can be achieved by manipulating the storage | | Audited |
| - Determine if any sensitive data is stored in either localStorage or sessionStorage | | Audited |

## Appendix 1.1 - Bootstrap_3.4.1.PNG