Outpost24

# Certification of Security testing at WhistleB Whistleblowing Centre AB

**2021-04-23**

## Web Application Security Testing

Outpost24 has, on behalf of WhistleB Whistleblowing Centre AB performed an one month (24th of February, 2021 to 23rd of March, 2021) of Manual Penetration Tests.

Main concern of the test was:
*Our key concern is client loss of data*

1. Methodology
Snapshot uses the methodology as described by OWASP, OSSTMM and best practices as described by several standards, like e.g. the ISO27001 standard and PCI DSS. The testing focuses on Application, Presentation and Session layers of the OSI. This includes examination of the implementation of the HTTP protocol, WebSockets, TLS, other encryption layers, caching and other mechanisms utilized by the application. This however doesn't mean that all the other layers are omitted, for instance HTTP relies on the Transport layer which is depending on the Network layer, so any eventual issues related to these layers will also be reported. Test activities and descriptions are presented below.

2. Report
The Manual Web Application Security Tests have been performed according to OWASP Top10 2017 in order to assist WhistleB Whistleblowing Centre AB in identifying vulnerabilities and misconfigurations of websites and applications. At the end of the assessment no vulnerabilities of severity high or medium according to OWASP Top10 Testing guidelines were identified.

Karlskrona, Sweden, 2021-04-23

Nils Thulin

Nils Thulin
Director Product Delivery

**About Outpost24**
Outpost24 is a leading cyber assessment company focused on enabling its customers to achieve maximum value from their evolving technology investments. By leveraging our full

stack security insights to reduce attack surface for any architecture, Outpost24 customers continuously improve their security posture with the least effort.

Over 2,000 customers in more than 40 countries around the world trust Outpost24 to assess their devices, networks, applications, cloud and container environments and report compliance status for government, industry sector, or internal regulations. Founded in 2001, Outpost24 serves leading organizations across a wide range of segments including financial and insurance, government, healthcare, retail, telecommunications, technology, and manufacturing.

**Testing Methodology**

The test-cases are oriented around the OWASP TESTING GUIDE, and for the application the following controls has been performed.

| **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 applicable |
| Search for: | | Not applicable |
| Network diagrams and configurations | | Not applicable |
| Archived posts and emails by administrators and other key staff | | Not applicable |
| Log on procedures and username formats | | Not applicable |
| Usernames and passwords | | Not applicable |
| Error message content | | Not applicable |
| Development, test, UAT and staging versions of the website | | Not applicable |
| | | |
| **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 applicable |
| 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 | | Audited |

| | | |
|---|---|---|
| component or node | | |
| - 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 applicable |
| - 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 applicable |
| Is there any verification, vetting and authorization of provisioning requests? | | Not applicable |
| Is there any verification, vetting and authorization of de-provisioning requests? | | Not applicable |
| 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 applicable |
| | | |
| **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 applicable |
| - Change requirements - when can the password be changed? How often can it be changed? | | Not applicable |
| - Reuse requirements - can old passwords be reused? How many old passwords does the application keep track of? | | Not applicable |
| - Difference requirements - how different must two passwords be in order to be accepted? Are any comparisons done at all? | | Not applicable |
| - 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- | Audited |

| | | |
|---|---|---|
| | 008 | |
| - 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-** | OTG-INPVAL- | Audited |

| | | |
|---|---|---|
| **017)** | 017 | |
| - 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 |
| | | |

Outpost24

| | | |
|---|---|---|
| **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 |