Pentesting Web checklist

Recon phase

* Large: a whole company with multiple domains

* Medium: a single domain

* Small: a single website

Large scope

- [ ] Get [ASN](../recon/public-info-gathering.md#amass) for IP ranges ([amass](https://github.com/OWASP/Amass), [asnlookup](https://github.com/yassineaboukir/Asnlookup), [metabigor](https://github.com/j3ssie/metabigor), [bgp] (https://bgp.he.net/))
- [ ] Review latest [acquisitions](https://www.crunchbase.com/)
- [ ] Get relationships by registrants ([viewdns](https://viewdns.info/reversewhois/))
- [ ] Go to medium scope for each domain

Medium scope

- [ ] [Enumerate subdomains](../recon/subdomain-enum/) ([amass](https://github.com/OWASP/Amass) or [subfinder](https://github.com/projectdiscovery/subfinder) with all available API keys)
- [ ] Subdomain bruteforce ([puredns](https://github.com/d3mondev/puredns) with [wordlist](https://gist.github.com/six2dez/a307a04a222fab5a57466c51e1569acf))
- [ ] Permute subdomains ([gotator](https://github.com/Josue87/gotator) or [ripgen](https://github.com/resyncgg/ripgen) with [wordlist](https://gist.github.com/six2dez/ffc2b14d283e8f8eff6ac83e20a3c4b4))
- [ ] Identify alive subdomains ([httpx](https://github.com/projectdiscovery/httpx))
- [ ] [Subdomain takeovers](../recon/subdomain-enum/subdomain-takeover.md) ([nuclei-takeovers](https://github.com/projectdiscovery/nuclei-templates/tree/master/takeovers))
- [ ] Check for [cloud assets](../enumeration/cloud/cloud-info-recon.md) ([cloudenum](https://github.com/initstring/cloud_enum))
- [ ] [Shodan](../recon/public-info-gathering.md#shodan) search
- [ ] [Transfer zone](https://six2dez.gitbook.io/pentest-book/enumeration/ports#port-53-dns)
- [ ] Subdomains recursive search
- [ ] Take screenshots ([gowitness](https://github.com/sensepost/gowitness), [webscreenshot](https://github.com/maaaaz/webscreenshot), [aquatone](https://github.com/michenriksen/aquatone))

Small scope

- [ ] Identify web server, technologies and database ([httpx](https://github.com/projectdiscovery/httpx))
- [ ] Try to locate `/robots.txt` , `/crossdomain.xml` `/clientaccesspolicy.xml` `/sitemap.xml` and `/.well-known/`
- [ ] Review comments on source code (Burp Engagement Tools)
- [ ] [Directory enumeration](../enumeration/web/crawl-fuzz.md)
- [ ] Web fuzzing ([ffuf](https://github.com/ffuf/ffuf) and [wordlist](https://github.com/six2dez/OneListForAll))
- [ ] Find[leaked ids, emails](../recon/public-info-gathering.md) ([pwndb](https://github.com/davidtavarez/pwndb))
- [ ] Identify WAF ([whatwaf](https://github.com/Ekultek/WhatWaf), [wafw00f](https://github.com/EnableSecurity/wafw00f))
- [ ] [Google dorking](../recon/public-info-gathering.md#google)
- [ ] [GitHub dorking](../recon/public-info-gathering.md#github)/Github tools ([githound](https://github.com/tillson/git-hound), [gitdorks_go](https://github.com/damit5/gitdorks_go))
- [ ] Get urls ([gau](https://github.com/lc/gau) , [waybackurls](https://github.com/tomnomnom/waybackurls), [gospider] (https://github.com/jaeles-project/gospider))
- [ ] Check potential vulnerable urls ([gf-patterns](https://github.com/1ndianl33t/Gf-Patterns))
- [ ] Automatic XSS finder ([dalfox](https://github.com/hahwul/dalfox))
- [ ] Locate admin and login panel
- [ ] Broken link hijacking ([blc](https://github.com/stevenvachon/broken-link-checker))
- [ ] Get all JS files ([subjs](https://github.com/lc/subjs), [xnLinkFinder](https://github.com/xnl-h4ck3r/xnLinkFinder))
- [ ] JS hardcoded APIs and secrets ([nuclei-tokens](https://github.com/projectdiscovery/nuclei-templates/tree/4e3f843e15c68f816f0ef6abce5d30b6cf6d4a30/exposures/tokens))

- [ ] JS analysis ([subjs](https://github.com/lc/subjs), [JSA](https://github.com/w9w/JSA), [xnLinkFinder] (https://github.com/xnl-h4ck3r/xnLinkFinder), [getjswords] (https://github.com/m4ll0k/BBTz))
- [ ] Run automated scanner ([nuclei](https://github.com/projectdiscovery/nuclei))
- [ ] Test CORS ([CORScanner](https://github.com/chenjj/CORScanner), [corsy] (https://github.com/s0md3v/Corsy))

Network

- [ ] Check ICMP packets allowed
- [ ] Check DMARC/SPF policies ([spoofcheck](https://github.com/BishopFox/spoofcheck))
- [ ] Open ports with [Shodan](https://www.shodan.io/)
- [ ] [Port scan](../recon/network-scanning.md#nmap) to all ports
- [ ] Check UDP ports ([udp-proto-scanner](https://github.com/CiscoCXSecurity/udp-proto-scanner) or nmap)
- [ ] Test [SSL](../enumeration/ssl-tls.md)([testssl](https://github.com/drwetter/testssl.sh))
- [ ] If got creds, try password [spraying](https://github.com/x90skysn3k/brutespray)for all the services discovered

Preparation

- [ ] Study site structure
- [ ] Make a list with all possible test cases
- [ ] Understand the business area and what their customer needs
- [ ] Get a list of every asset (all_subdomains.txt, live_subdomains.txt, waybackurls.txt, hidden_directories.txt, nmap_results.txt, GitHub_search.txt, altdns_subdomain.txt, vulnerable_links.txt, js_files.txt)

User management

Registration

- [ ] Duplicate registration (try with uppercase, +1@..., dots in name, etc)
- [ ] Overwrite existing user (existing user takeover)
- [ ] Username uniqueness
- [ ] Weak password policy (user=password, password=123456,111111,abcabc,qwerty12)
- [ ] [Insufficient email verification process](../enumeration/web/email-attacks.md) (also my%00email@mail.com for account tko)
- [ ] Weak registration implementation or allows disposable email addresses
- [ ] Fuzz after user creation to check if any folder have been overwritten or created with your profile name
- [ ] Add only spaces in password
- [ ] Long password (>200) leads to DoS
- [ ] Corrupt authentication and session defects: Sign up, don't verify, request change password, change, check if account is active.
- [ ] Try to re-register repeating same request with same password and different password too
- [ ] If JSON request, add comma {“email”:“victim@mail.com”,”hacker@mail.com”,“token”:”xxxxxxxxxx”}
- [ ] Lack of confirmation -> try to register with company email.
- [ ] Check OAuth with social media registration
- [ ] Check state parameter on social media registration
- [ ] Try to capture integration url leading integration takeover
- [ ] Check redirections in register page after login
- [ ] Rate limit on account creation
- [ ] XSS on name or email

Authentication

- [ ] Username enumeration
- [ ] Resilience to password guessing
- [ ] Account recovery function
- [ ] "Remember me" function
- [ ] Impersonation function
- [ ] Unsafe distribution of credentials
- [ ] Fail-open conditions and Multi-stage mechanisms
- [ ] [SQL Injections](../enumeration/web/sqli.md)
- [ ] Auto-complete testing
- [ ] Lack of password confirmation on change email, password or 2FA (try change response)
- [ ] Weak login function over HTTP and HTTPS if both are available
- [ ] User account lockout mechanism on brute force attack
- [ ] Check for password wordlist ([cewl](https://github.com/digininja/CeWL) and [burp-goldenNuggets](https://github.com/GainSec/GoldenNuggets-1))
- [ ] Test 0auth login functionality for [Open Redirection](../enumeration/web/ssrf.md)
- [ ] Test response tampering in [SAML](../enumeration/webservices/onelogin-saml-login.md)authentication
- [ ] In OTP check guessable codes and race conditions
- [ ] OTP, check response manipulation for bypass
- [ ] OTP, try bruteforce
- [ ] If [JWT](../enumeration/webservices/jwt.md), check common flaws
- [ ] Browser cache weakness (eg Pragma, Expires, Max-age)
- [ ] After register, logout, clean cache, go to home page and paste your profile url in browser, check for "login?next=accounts/profile" for open redirect or XSS with "/login?next=javascript:alert(1);//"
- [ ] Try login with common [credentials](https://github.com/ihebski/DefaultCreds-cheat-sheet)

Session

- [ ] Session handling
- [ ] Test tokens for meaning and Test tokens for predictability
- [ ] Insecure transmission of tokens
- [ ] Disclosure of tokens in logs
- [ ] Mapping of tokens to sessions
- [ ] Session termination and Session fixation
- [ ] [Cross-site request forgery](../enumeration/web/csrf.md)
- [ ] Cookie scope
- [ ] Decode Cookie (Base64, hex, URL etc.)
- [ ] Cookie expiration time
- [ ] Check HTTPOnly and Secure flags
- [ ] Use same cookie from a different effective IP address or system
- [ ] Access controls
- [ ] Effectiveness of controls using multiple accounts
- [ ] Insecure access control methods (request parameters, Referer header, etc)
- [ ] Check for concurrent login through different machine/IP
- [ ] Bypass [AntiCSRF](../enumeration/web/csrf.md#csrf-token-bypass)tokens
- [ ] Weak generated security questions
- [ ] Path traversal on cookies
- [ ] Reuse cookie after session closed
- [ ] Logout and click browser "go back" function (Alt + Left arrow)
- [ ] 2 instances open, 1st change or reset password, refresh 2nd instance
- [ ] With privileged user perform privileged actions, try to repeat with unprivileged user cookie.

Profile/Account details

- [ ] Find parameter with user id and try to tamper in order to get the details of other users
- [ ] Create a list of features that are pertaining to a user account only and try [CSRF](../enumeration/web/csrf.md)
- [ ] Change email id and update with any existing email id. Check if its getting validated on server or not.
- [ ] Check any new email confirmation link and what if user doesn't confirm.
- [ ] File [upload](../enumeration/web/upload-bypasses.md): [eicar](https://secure.eicar.org/eicar.com.txt), No Size Limit, File extension, Filter Bypass, [burp](https://github.com/portswigger/upload-scanner) extension, RCE
- [ ] CSV import/export: Command Injection, XSS, macro injection
- [ ] Check profile picture URL and find email id/user info or [EXIF Geolocation Data](http://exif.regex.info/exif.cgi)
- [ ] Imagetragick in picture profile upload
- [ ] [Metadata](https://github.com/exiftool/exiftool)of all downloadable files (Geolocation, usernames)
- [ ] Account deletion option and try to reactivate with "Forgot password" feature
- [ ] Try bruteforce enumeration when change any user unique parameter.
- [ ] Check application request re-authentication for sensitive operations
- [ ] Try parameter pollution to add two values of same field
- [ ] Check different roles policy

Forgot/reset password

- [ ] Invalidate session on Logout and Password reset
- [ ] Uniqueness of forget password reset link/code
- [ ] Reset links expiration time
- [ ] Find user id or other sensitive fields in reset link and tamper them
- [ ] Request 2 reset passwords links and use the older
- [ ] Check if many requests have sequential tokens
- [ ] Use username@burp_collab.net and analyze the callback
- [ ] Host header injection for token leakage
- [ ] Add X-Forwarded-Host: evil.com to receive the reset link with evil.com
- [ ] Email crafting like victim@gmail.com@target.com
- [ ] IDOR in reset link
- [ ] Capture reset token and use with other email/userID
- [ ] No TLD in email parameter
- [ ] User carbon copy email=victim@mail.com%0a%0dcc:hacker@mail.com
- [ ] Long password (>200) leads to DoS
- [ ] No rate limit, capture request and send over 1000 times
- [ ] Check encryption in reset password token
- [ ] Token leak in referer header
- [ ] Append second email param and value
- [ ] Understand how token is generated (timestamp, username, birthdate,...)
- [ ] Response manipulation

Input handling

- [ ] Fuzz all request parameters (if got user, add headers to fuzzer)
- [ ] Identify all reflected data
- [ ] [Reflected XSS](../enumeration/web/xss.md)
- [ ] HTTP[header injection](../enumeration/web/header-injections.md) in GET & POST (X Forwarded Host)
- [ ] RCE via Referer Header
- [ ] SQL injection via User-Agent Header
- [ ] Arbitrary redirection
- [ ] Stored attacks
- [ ] OS command injection
- [ ] Path [traversal](../enumeration/web/lfi-rfi.md), LFI and RFI
- [ ] Script injection
- [ ] File inclusion
- [ ] SMTP injection
- [ ] Native software flaws (buffer overflow, integer bugs, format strings)
- [ ] SOAP injection
- [ ] LDAP injection
- [ ] SSI Injection
- [ ] XPath injection
- [ ] [XXE](../enumeration/web/xxe.md) in any request, change content-type to text/xml
- [ ] Stored [XSS](../enumeration/web/xss.md)
- [ ] [SQL](../enumeration/web/sqli.md)injection with ' and '--+-
- [ ] [NoSQL](../enumeration/webservices/nosql-and-and-mongodb.md)injection
- [ ] HTTP Request [Smuggling](../enumeration/web/request-smuggling.md)

- [ ] [Open redirect](../enumeration/web/ssrf.md)
- [ ] Code Injection (\<h1>six2dez\</h1> on stored param)
- [ ] [SSRF](../enumeration/web/ssrf.md)in previously discovered open ports
- [ ] xmlrpc.php DOS and user enumeration
- [ ] HTTP dangerous methods OPTIONS PUT DELETE
- [ ] Try to discover hidden parameters ([arjun](https://github.com/s0md3v/Arjun)or [parameth](https://github.com/maK-/parameth))

Error handling

- [ ] Access custom pages like /whatever_fake.php (.aspx,.html,.etc)
- [ ] Add multiple parameters in GET and POST request using different values
- [ ] Add "\[]", "]]", and "\[\[" in cookie values and parameter values to create errors
- [ ] Generate error by giving input as "/\~randomthing/%s" at the end of URL
- [ ] Use Burp Intruder "Fuzzing Full" List in input to generate error codes
- [ ] Try different HTTP Verbs like PATCH, DEBUG or wrong like FAKE

Application Logic

- [ ] Identify the logic attack surface
- [ ] Test transmission of data via the client
- [ ] Test for reliance on client-side input validation
- [ ] Thick-client components (Java, ActiveX, Flash)
- [ ] Multi-stage processes for logic flaws
- [ ] Handling of incomplete input
- [ ] Trust boundaries
- [ ] Transaction logic
- [ ] Implemented CAPTCHA in email forms to avoid flooding
- [ ] Tamper product id, price or quantity value in any action (add, modify, delete, place, pay...)
- [ ] Tamper gift or discount codes
- [ ] Reuse gift codes
- [ ] Try parameter pollution to use gift code two times in same request
- [ ] Try stored XSS in non-limited fields like address
- [ ] Check in payment form if CVV and card number is in clear text or masked
- [ ] Check if is processed by the app itself or sent to 3rd parts
- [ ] IDOR from other users details ticket/cart/shipment
- [ ] Check for test credit card number allowed like 4111 1111 1111 1111 ([sample1](https://www.paypalobjects.com/en_GB/vhelp/paypalmanager_help/credit_card_numbers.htm) [sample2] (http://support.worldpay.com/support/kb/bg/testandgolive/tgl5103.html))
- [ ] Check PRINT or PDF creation for IDOR
- [ ] Check unsubscribe button with user enumeration
- [ ] Parameter pollution on social media sharing links
- [ ] Change POST sensitive requests to GET

Other checks
Infrastructure

- [ ] Segregation in shared infrastructures
- [ ] Segregation between ASP-hosted applications
- [ ] Web server vulnerabilities
- [ ] Dangerous HTTP methods
- [ ] Proxy functionality
- [ ] [Virtual](../enumeration/webservices/vhosts.md)hosting misconfiguration ([VHostScan](https://github.com/codingo/VHostScan))
- [ ] Check for internal numeric IP's in request
- [ ] Check for external numeric IP's and resolve it
- [ ] Test [cloud](../enumeration/cloud/cloud-info-recon.md)storage
- [ ] Check the existence of alternative channels (www.web.com vs m.web.com)

CAPTCHA

- [ ] Send old captcha value.
- [ ] Send old captcha value with old session ID.
- [ ] Request captcha absolute path like www.url.com/captcha/1.png
- [ ] Remove captcha with any adblocker and request again
- [ ] Bypass with OCR tool ([easy one](https://github.com/pry0cc/prys-hacks/blob/master/image-to-text))
- [ ] Change from POST to GET
- [ ] Remove captcha parameter
- [ ] Convert JSON request to normal
- [ ] Try header injections

Security Headers

- ☐ X-XSS-Protection
- ☐ Strict-Transport-Security
- ☐ Content-Security-Policy
- ☐ Public-Key-Pins
- ☐ X-Frame-Options
- ☐ X-Content-Type-Options
- ☐ Referer-Policy
- ☐ Cache-Control
- ☐ Expires