

Pentest Book

Pentesting Web checklist

Recon phase

Small scope

- Identify web server, technologies and database ([whatweb](#), [webanalyze](#))
- Try to locate /robots.txt /crossdomain.xml /clientaccesspolicy.xml /sitemap.xml and /.well-known/
- Review comments on source code (Burp Engagement Tools)
- [Directory enumeration](#)
- Find [leaked ids, emails](#) ([pwndb](#))
- Identify WAF ([whatwaf](#), [wafw00f](#))
- [Google dorking](#)
- [GitHub dorking](#)/Github tools ([githound](#), [git-search](#))
- Get urls ([gau](#) , [waybackurls](#), [hakrawler](#))
- Check potential vulnerable urls ([gf-patterns](#))
- Find hidden parameters ([paramspider](#))
- Automatic XSS finder ([dalfox](#))
- Check for backup files ([bfac](#))
- Locate admin and login panel
- Broken link hijacking ([blc](#))
- Get all JS files ([subjs](#), [linkfinder](#))
- JS hardcoded APIs and secrets ([secretfinder](#))
- JS analysis ([JSParser](#), [JSFScan](#), [JSScanner](#), [jshole](#))
- Run automated scanner ([nuclei](#))
- Test CORS ([CORScanner](#), [corsy](#))

Medium scope

- [Enumerate subdomains](#) ([subfinder](#), [assetfinder](#), [amass](#), [sudomy](#), [crobat](#), [SubDomainizer](#))
- Permute subdomains ([dnsgen](#))
- Subdomain bruteforce ([shuffledns](#), [subbrute](#))
- Identify alive subdomains ([httpx](#))

- Subdomain takeovers (SubOver)
- Check for cloud assets (cloudeenum, cloudscraper, cloudlist)
- Shodan
- Transfer zone
- Subdomains from subdomains (altdns, flydns, goaltdns)
- Take screenshots (gowitness, webscreenshot, aquatone)

Large scope

- Get ASN for IP ranges (amass, asnlookup, metabigor, bgp)
- Review latest acquisitions

Network

- Check ICMP packets allowed
- Check DMARC/SPF policies (spooftest)
- Open ports with Shodan
- Port scan to all ports
- Check UDP ports (udp-proto-scanner or nmap)
- Test SSL (testssl)
- If got creds, try password spraying 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
- Overwrite existing user (existing user takeover)
- Username uniqueness
- Weak password policy (user=password, password=123456,111111,abcabc,qwerty12)
- Insufficient email verification process** (also my%00email@mail.com for account takeover)
- 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

Authentication

- Username enumeration
- Resilience to password guessing
- Account recovery function
- "Remember me" function
- Impersonation function
- Unsafe distribution of credentials
- Fail-open conditions
- Multi-stage mechanisms
- SQL Injections**
- 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](#) and [burp-goldenNuggets](#))
 - Test OAuth login functionality for [Open Redirection](#)
 - Test response tampering in [SAML](#) authentication
 - In OTP check guessable codes and race conditions
 - OTP, check response manipulation for bypass
 - OTP, try bruteforce
 - If [JWT](#), 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](#)

Session

- Session handling
- Test tokens for meaning
- Test tokens for predictability
- Insecure transmission of tokens
- Disclosure of tokens in logs
- Mapping of tokens to sessions
- Session termination
- Session fixation
- [Cross-site request forgery](#)
- 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](#) 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**
- 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**: **eicar**, No Size Limit, File extension, Filter Bypass, **burp** extension, RCE
- CSV import/export: Command Injection, XSS, macro injection
- Check profile picture URL and find email id/user info or **EXIF Geolocation Data**
- Imagetragick in picture profile upload
- Metadata** 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
-

Input handling

- Fuzz all request parameters (if got user, add headers to fuzzer)
- Identify all reflected data
- Reflected XSS
- HTTP header injection 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, 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 in any request, change content-type to text/xml
- Stored XSS
- SQL injection with ' and '-+-
- NoSQL injection
- HTTP Request Smuggling

- Open redirect
- Code Injection (<h1>six2dez</h1> on stored param)
- SSRF in previously discovered open ports
- xmlrpc.php DOS and user enumeration
- HTTP dangerous methods OPTIONS PUT DELETE
- Try to discover hidden parameters (arjun or 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](#) [sample2](#))
 - 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](#) hosting misconfiguration ([VHostScan](#))
- Check for internal numeric IP's in request
- Check for external numeric IP's and resolve it
- Test [cloud](#) 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](#))
- 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