Web Application Penetration Testing Checklist

# More than 200 custom test cases

**Recon Phase**

Identify web server, technologies and database Subsidiary and Acquisition Enumeration Reverse Lookup

ASN & IP Space Enumeration and Service Enumeration Google Dorking

Github Recon Directory Enumeration IP Range Enumeration JS Files Analysis

Subdomain Enumeration and Bruteforcing Subdomain Takeover

Parameter Fuzzing Port Scanning

Template-Based Scanning(Nuclei) Wayback History

Broken Link Hijacking

Internet Search Engine Discovery Misconfigured Cloud Storage

# Registration Feature Testing

Check for duplicate registration/Overwrite existing user Check for weak password policy

Check for reuse existing usernames

Check for insufficient email verification process

Weak registration implementation-Allows disposable email addresses Weak registration implementation-Over HTTP

Overwrite default web application pages by specially crafted username registrations. ⇒ After registration, does your profile link appears something as [www.tushar.com/](http://www.chintan.com/chintan)tushar?

1. If so, enumerate default folders of web application such as /images, /contact, /portfolio
2. Do a registration using the username such as images, contact, portfolio
3. Check if those default folders have been overwritten by your profile link or not."

# Session Management Testing

Identify actual session cookie out of bulk cookies in the application

Decode cookies using some standard decoding algorithms such as Base64, hex, URL, etc

Modify cookie.session token value by 1 bit/byte. Then resubmit and do the same for all tokens. Reduce the amount of work you need to perform in order to identify which part of the token is actually being used and which is not

If self-registration is available and you can choose your username, log in with a series of similar usernames containing small variations between them, such as A, AA, AAA, AAAA, AAAB, AAAC, AABA, and so on. If another user-specific data is submitted at login or stored in user profiles (such as an email address)

Check for session cookies and cookie expiration date/time Identify cookie domain scope

Check for HttpOnly flag in cookie

Check for Secure flag in cookie if the application is over SSL

Check for session fixation i.e. value of session cookie before and after authentication

Replay the session cookie from a different effective IP address or system to check whether the server maintains the state of the machine or not

Check for concurrent login through different machine/IP

Check if any user pertaining information is stored in cookie value or not If yes, tamper it with other user's data Failure to Invalidate Session on Email Change,2FA Activation)

# Authentication Testing

Username enumeration

Bypass authentication using various SQL Injections on username and password field Lack of password confirmation on

Change email address Change password Manage 2FA

Is it possible to use resources without authentication? Access violation Check if user credentials are transmitted over SSL or not

Weak login function HTTP and HTTPS both are available Test user account lockout mechanism on brute force attack

Variation : If server blocks instant user requests, then try with time throttle option from intruder and repeat the

process again.

Bypass rate limiting by tampering user agent to Mobile User agent Bypass rate limiting by tampering user agent to Anonymous user agent Bypass rate liniting by using null byte

Create a password wordlist using cewl command Test Oauth login functionality

OAuth Roles

Resource Owner → User Resource Server → Twitter

Authorization Server → Twitter

client\_id → Twitterdeck ID This is a public, non-secret unique identifier\_

client\_secret → Secret Token known to the Twitter and Twitterdeck to generate access\_tokens response\_type → Defines the token type e.g (code, token, etc.)

scope → The requested level of access Twitterdeck wants

redirect\_uri → The URL user is redirected to after the authorization is complete

state → Main CSRF protection in OAuth can persist data between the user being directed to the authorization server and back again

grant\_type → Defines the grant\_type and the returned token type

code → The authorization code twitter generated, will be like ?code= , the code is used with client\_id and client\_secret to fetch an access\_token

access\_token → The token twitterdeck uses to make API requests on behalf of the user refresh\_token → Allows an application to obtain a new access\_token without prompting the user

Code Flaws

Re-Using the code

Code Predict/Bruteforce and Rate-limit

Is the code for application X valid for application Y?

Redirect\_uri Flaws

URL isn't validated at all: ?redirect\_uri=https://attacker.com

Subdomains allowed Subdomain Takeover or Open redirect on those subdomains): ? redirect\_uri=https://sub.twitterdeck.com

Host is validated, path isn't Chain open redirect): ?redirect\_uri=https://twitterdeck.com/callback? redirectUrl=https://evil.com

Host is validated, path isn't Referer leakages): Include external content on HTML page and leak code via Referer

Weak Regexes

Bruteforcing the URL encoded chars after host: redirect\_uri=https://twitterdeck.com§FUZZ§

Bruteforcing the keywords whitelist after host (or on any whitelist open redirect filter): ? redirect\_uri=https://§FUZZ§.com

URI validation in place: use typical open redirect payloads State Flaws

Missing State parameter? CSRF Predictable State parameter?

Is State parameter being verified?

Misc

Is client\_secret validated?

Pre ATO using facebook phone-number signup No email validation Pre ATO

Test 2FA Misconfiguration Response Manipulation Status Code

Manipulation

2FA Code Reusability

Lack of Brute-Force Protection Missing 2FA Code Integrity Validation With null or 000000

# My Account Post Login) Testing

Find parameter which uses active account user id. Try to tamper it in order to change the details of the other accounts

Create a list of features that are pertaining to a user account only. Change Email Change Password Change account details Name, Number, Address, etc.) Try CSRF

Post login change email id and update with any existing email id. Check if its getting validated on server side or not. Does the application send any new email confirmation link to a new user or not? What if a user does not confirm the link in some time frame?

Open profile picture in a new tab and check the URL. Find email id/user id info. EXIF Geolocation Data Not Stripped From Uploaded Images.

Check account deletion option if application provides it and confirm that via forgot password feature Change email id, account id, user id parameter and try to brute force other user's password

Check whether application re authenticates for performing sensitive operation for post authentication features

# Forgot Password Testing

Failure to invalidate session on Logout and Password reset Check if forget password reset link/code uniqueness

Check if reset link does get expire or not if its not used by the user for certain amount of time

Find user account identification parameter and tamper Id or parameter value to change other user's password Check for weak password policy

Weak password reset implementation Token is not invalidated after use

If reset link has another param such as date and time, then. Change date and time value in order to make active & valid reset link

Check if security questions are asked? How many guesses allowed? ⟶ Lockout policy maintained or not? Add only spaces in new password and confirmed password. Then Hit enter and see the result

Does it display old password on the same page after completion of forget password formality? Ask for two password reset link and use the older one from user's email

Check if active session gets destroyed upon changing the password or not? Weak password reset implementation Password reset token sent over HTTP Send continuous forget password requests so that it may send sequential tokens

# Contact Us Form Testing

Is CAPTCHA implemented on contact us form in order to restrict email flooding attacks? Does it allow to upload file on the server?

Blind XSS

# Product Purchase Testing

Buy Now

Tamper product ID to purchase other high valued product with low prize

Tamper product data in order to increase the number of product with the same prize Gift/Voucher

used

Tamper gift/voucher value to increase/decrease the value of the voucher in terms of money. (e.g. $100 is given as a voucher, tamper value to increase, decrease money)

Reuse gift/voucher by using old gift values in parameter tampering

Check the uniqueness of gift/voucher parameter and try guessing other gift/voucher code

Use parameter pollution technique to add the same voucher twice by adding same parameter name and value again with & in the BurpSuite request

Add/Delete Product from Cart

Tamper user id to delete products from other user's cart Tamper cart id to add/delete products from other user's cart

Identify cart id/user id for cart feature to view the added items from other user's account Address

Tamper BurpSuite request to change other user's shipping address to yours Try stored XSS by adding XSS vector on shipping address

Use parameter pollution technique to add two shipping address instead of one trying to manipulate application to send same item on two shipping address

Place Order

Tamper payment options parameter to change the payment method. E.g. Consider some items cannot be ordered for cash on delivery but tampering request parameters from debit/credit/PayPal/net banking option to cash on delivery may allow you to

place order for that particular item

Tamper the amount value for payment manipulation in each main and sub requests and responses Check if CVV is going in cleartext or not

Check if the application itself processes your card details and then performs a transaction or it calls any third-party payment processing company to perform a transaction

Track Order

Track other user's order by guessing order tracking number

Brute force tracking number prefix or suffix to track mass orders for other users Wish list page testing

Check if a user A can add/remote products in Wishlist of other user B’s account

Check if a user A can add products into user B’s cart from his/her (user A’s) Wishlist section.

Post product purchase testing

Check if user A can cancel orders for user B’s purchase

Check if user A can view/check orders already placed by user B

Check if user A can modify the shipping address of placed order by user B Out of band testing

Can user order product which is out of stock?

# Banking Application Testing

Billing Activity

Check if user 'A' can view the account statement for user 'B' Check if user 'A' can view the transaction report for user 'B' Check if user 'A' can view the summary report for user 'B'

Check if user 'A' can register for monthly/weekly account statement via email behalf of user 'B'

summary

Deposit/Loan/Linked/External Account Checking

Check if user 'A' can view the deposit account summary of user 'B' Check for account balance tampering for Deposit accounts

Tax Deduction Inquiry Testing

Check if user 'A' with it's customer id 'a' can see the tax deduction details of user 'B' by tampering his/her customer id 'b'

Check parameter tampering for increasing and decreasing interest rate, interest amount, and tax refund Check if user 'A' can download the TDS details of user 'B’

Check if user 'A' can request for the cheque book behalf of user ‘B.’ Fixed Deposit Account Testing

Check if is it possible for user 'A' to open FD account behalf of user 'B'

Check if Can user open FD account with the more amount than the current account balance Stopping Payment on basis of cheque/date range

Can user 'A' stop the payment of user 'B' via cheque number Can user 'A' stop the payment on basis of date range for user 'B’

Status Enquiry Testing

Can user 'A' view the status enquiry of user 'B' Can user 'A' modify the status enquiry of user 'B'

Can user 'A' post and enquiry behalf of user 'B' from his own account Fund transfer testing

Is it possible to transfer funds to user 'C' instead of user 'B' from the user 'A' which was intended to transfer from user 'A' to user 'B'

Can fund transfer amount be manipulated?

Can user 'A' modify the payee list of user 'B' by parameter manipulation using his/her own account

Is it possible to add payee without any proper validation in user 'A' 's own account or to user 'B' 's account Schedule transfer testing

Can user 'A' view the schedule transfer of user 'B'

Can user 'A' change the details of schedule transfer for user 'B’ Testing of fund transfer via NEFT

Amount manipulation via NEFT transfer

Check if user 'A' can view the NEFT transfer details of user 'B’ Testing for Bill Payment

Check if user can register payee without any checker approval Check if user 'A' can view the pending payments of user 'B' Check if user 'A' can view the payment made details of user 'B'

# Open Redirection Testing

Common injection parameters

/{payload}

?next={payload}

?url={payload}

?target={payload}

?rurl={payload}

?dest={payload}

?destination={payload}

?redir={payload}

?redirect\_uri={payload}

?redirect\_url={payload}

?redirect={payload}

/redirect/{payload}

/cgi-bin/redirect.cgi?{payload}

/out/{payload}

/out?{payload}

?view={payload}

/login?to={payload}

?image\_url={payload}

?go={payload}

?return={payload}

?returnTo={payload}

?return\_to={payload}

?checkout\_url={payload}

?continue={payload}

?return\_path={payload}

Use burp 'find' option in order to find parameters such as URL, red, redirect, redir, origin, redirect\_uri, target etc Check the value of these parameter which may contain a URL

Change the URL value to [www.tushar.com](http://www.chintan.com/) and check if gets redirected or not Try Single Slash and url encoding

Using a whitelisted domain or keyword Using // to bypass http blacklisted keyword

Using https: to bypass // blacklisted keyword Using \\ to bypass // blacklisted keyword Using \/\/ to bypass // blacklisted keyword Using null byte %00 to bypass blacklist filter Using ° symbol to bypass

# Host Header Injection

Supply an arbitrary Host header Check for flawed validation Send ambiguous requests

Inject duplicate Host headers Supply an absolute URL

Add line wrapping

Inject host override headers

# SQL Injection Testing

Entry point detection Simple characters Multiple encoding Merging characters Logic Testing Weird characters

Use SQLmap to identify vulnerabile parameters Fill form in browser GUI submit it normally

Go to history tab in burpsuite and find the relevent request Right click and select the option "copy to file"

Save file as anyname.txt

python [sqlmap.py](http://sqlmap.py/) r ~/Desktop/textsqli.txt proxy= [http://127.0.0.1](http://127.0.0.1/)8080 Run SQL injection scanner on all requests

Bypassing WAF

Using Null byte before SQL query Using SQL inline comment sequence URL encoding

Changing Cases (uppercase/lowercase) Use SQLMAP tamper scripts

Time Delays

Oracle

dbms\_pipe.receive\_message(('a'),10)

Microsoft WAITFOR DELAY '0:0:10'

PostgreSQL SELECT pg\_sleep(10)

MySQL

SELECT sleep(10)

Conditional Delays

Oracle

SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 'a'||dbms\_pipe.receive\_message(('a'),10) ELSE NULL END FROM d

Microsoft IF (YOUR-CONDITION-HERE) WAITFOR DELAY '0:0:10'

PostgreSQL SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN pg\_sleep(10) ELSE pg\_sleep(0) END

MySQL

SELECT IF(YOUR-CONDITION-HERE,sleep(10),'a')

# Cross-Site Scripting Testing

Try XSS using QuickXSS tool by theinfosecguy

Upload file using '"><img src=x onerror=alert(document.domain)>.txt If script tags are banned, use <h1 and other HTML tags

If output is reflected back inside the JavaScript as a value of any variable just use alert(1) if " are filtered then use this payload /><img src=d onerror=confirm(/tushar/);>

Upload a JavaScript using Image file

Unusual way to execute your JS payload is to change method from POST to GET. It bypasses filters sometimes Tag attribute value

Input landed -<input type=”text” name=”state” value=”INPUT\_FROM\_ USER”> Payload to be inserted -“ onfocus=”alert(document.cookie)"

Syntax Encoding payload “%3cscript%3ealert(document.cookie)%3c/script%3e" XSS filter evasion

< and > can be replace with html entities &lt; and &gt;

You can try an XSS polyglot.Eg:-javascript:/*→</title></style></textarea></script></xmp>*

*<svg/onload='+/"/+/onmouseover=1/+/[*/[]/+alert(1)//'> XSS Firewall Bypass

Check if the firewall is blocking only lowercase Try to break firewall regex with the new line(\r\n) Try Double Encoding

Injecting anchor tag without whitespaces Try to bypass whitespaces using Bullet Try to change request method

# CSRF Testing

Validation of CSRF token depends on request method Validation of CSRF token depends on token being present CSRF token is not tied to the user session

CSRF token is tied to a non-session cookie

Validation of Referer depends on header being present

# SSO Vulnerabilities

If internal.company.com Redirects You To SSO e.g. auth.company.com, Do FUZZ On Internal.company.com

If company.com/internal Redirects You To SSO e.g. Google login, Try To Insert public Before internal e.g. company.com/public/internal To Gain Access Internal

Try To Craft SAML Request With Token And Send It To The Server And Figure Out How Server Interact With This

If There Is AssertionConsumerServiceURL In Token Request Try To Insert Your Domain e.g. [http://me.com](http://me.com/) As Value To Steal The Token

If There Is AssertionConsumerServiceURL In Token Request Try To Do FUZZ On Value Of AssertionConsumerServiceURL If It Is Not Similar To Origin

If There Is Any UUID, Try To Change It To UUID Of Victim Attacker e.g. Email Of Internal Employee Or Admin Account etc

Try To Figure Out If The Server Vulnerable To XML Signature Wrapping OR Not? Try To Figure Out If The Server Checks The Identity Of The Signer OR Not?

Try To Inject XXE Payloads At The Top Of The SAML Response

Try To Inject XSLT Payloads Into The Transforms Element As A Child Node Of The SAML Response

If Victim Can Accept Tokens Issued By The Same Identity Provider That Services Attacker, So You Can Takeover Victim Account

While Testing SSO Try To search In Burp Suite About URLs In Cookie Header e.g. Host=IP; If There Is Try To Change IP To Your IP To Get SSRF

# XML Injection Testing

Change the content type to text/xml then insert below code. Check via repeater

<?xml version="1.0" encoding="ISO 8859 1"?>

<!DOCTYPE tushar [

<!ELEMENT tushar ANY

<!ENTITY xxe SYSTEM "file:///etc/passwd" >]><tushar>&xxe;</

<!ENTITY xxe SYSTEM "file:///etc/hosts" >]><tushar>&xxe;</

<!ENTITY xxe SYSTEM "file:///proc/self/cmdline" >]><tushar>&xxe;</

<!ENTITY xxe SYSTEM "file:///proc/version" >]><tushar>&xxe;</

Blind XXE with out-of-band interaction

**Cross-origin resource sharing CORS** Errors parsing Origin headers Whitelisted null origin value

# Server-side request forgery SSRF

Common injection parameters

"access=", "admin=",

"dbg=",

"debug=",

"edit=",

"grant=",

"test=",

"alter=",

"clone=", "create=", "delete=", "disable=", "enable=", "exec=", "execute=", "load=",

"make=", "modify=", "rename=", "reset=",

"shell=", "toggle=", "adm=",

"root=",

"cfg=",

"dest=", "redirect=", "uri=",

"path=", "continue=", "url=", "window=", "next=",

"data=", "reference=", "site=",

"html=",

"val=", "validate=", "domain=", "callback=", "return=", "page=",

"feed=",

"host=",

"port=",

"to=",

"out=",

"view=",

"dir=",

"show=", "navigation=", "open=",

"file=", "document=", "folder=", "pg=", "php\_path=", "style=",

"doc=",

"img=", "filename="

Try basic localhost payloads Bypassing filters

Bypass using HTTPS Bypass with [::]

Bypass with a domain redirection Bypass using a decimal IP location

Bypass using IPv6/IPv4 Address Embedding

Bypass using malformed urls

Bypass using enclosed alphanumerics Cloud Instances

AWS

[http://instance-data](http://instance-data/) [http://169.254.169.254](http://169.254.169.254/)

<http://169.254.169.254/latest/user-data>

<http://169.254.169.254/latest/user-data/iam/security-credentials/>[ROLE NAME] <http://169.254.169.254/latest/meta-data/>

<http://169.254.169.254/latest/meta-data/iam/security-credentials/>[ROLE NAME] <http://169.254.169.254/latest/meta-data/iam/security-credentials/PhotonInstance> <http://169.254.169.254/latest/meta-data/ami-id> <http://169.254.169.254/latest/meta-data/reservation-id> <http://169.254.169.254/latest/meta-data/hostname> <http://169.254.169.254/latest/meta-data/public-keys/> <http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key> <http://169.254.169.254/latest/meta-data/public-keys/>[ID]/openssh-key <http://169.254.169.254/latest/meta-data/iam/security-credentials/dummy> <http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access> <http://169.254.169.254/latest/dynamic/instance-identity/document>

Google Cloud

<http://169.254.169.254/computeMetadata/v1/> <http://metadata.google.internal/computeMetadata/v1/> <http://metadata/computeMetadata/v1/> <http://metadata.google.internal/computeMetadata/v1/instance/hostname> <http://metadata.google.internal/computeMetadata/v1/instance/id> <http://metadata.google.internal/computeMetadata/v1/project/project-id>

Digital Ocean

curl <http://169.254.169.254/metadata/v1/id> <http://169.254.169.254/metadata/v1.json> <http://169.254.169.254/metadata/v1/> <http://169.254.169.254/metadata/v1/id> <http://169.254.169.254/metadata/v1/user-data> <http://169.254.169.254/metadata/v1/hostname> <http://169.254.169.254/metadata/v1/region> <http://169.254.169.254/metadata/v1/interfaces/public/0/ipv6/address>

Azure

<http://169.254.169.254/metadata/v1/maintenance> <http://169.254.169.254/metadata/instance?api-version=2017-04-02>

<http://169.254.169.254/metadata/instance/network/interface/0/ipv4/ipAddress/0/publicIpAddress?api-version=2017-04-02&f> ormat=text

Bypassing via open redirection

# File Upload Testing

upload the malicious file to the archive upload functionality and observe how the application responds upload a file and change its path to overwrite an existing system file

Large File Denial of Service Metadata Leakage ImageMagick Library Attacks Pixel Flood Attack

Bypasses

Null Byte %00 Bypass Content-Type Bypass

Client-Side Validation Bypass Blacklisted Extension Bypass Homographic Character Bypass

# CAPTCHA Testing

Missing Captcha Field Integrity Checks HTTP Verb Manipulation

Content Type Conversion Reusuable Captcha

Check if captcha is retrievable with the absolute path such as [www.tushar.com/internal/captcha/images/24.png](http://www.chintan.com/internal/captcha/images/24.png)

Check for the server side validation for CAPTCHA.Remove captcha block from GUI using firebug addon and submit request to the server

Check if image recognition can be done with OCR tool?

# JWT Token Testing

Brute-forcing secret keys

Signing a new token with the “none” algorithm

Changing the signing algorithm of the token (for fuzzing purposes)

Signing the asymmetrically-signed token to its symmetric algorithm match (when you have the original public key)

# Websockets Testing

Intercepting and modifying WebSocket messages Websockets MITM attempts

Testing secret header websocket Content stealing in websockets

Token authentication testing in websockets

**GraphQL Vulnerabilities Testing** Inconsistent Authorization Checks Missing Validation of Custom Scalars Failure to Appropriately Rate-limit Introspection Query Enabled/Disabled

# WordPress Common Vulnerabilities

XSPA in wordpress Bruteforce in wp-login.php

Information disclosure wordpress username Backup file wp-config exposed

Log files exposed

Denial of Service via load-styles.php Denial of Service via load-scripts.php DDOS using xmlrpc.php

# Denial of Service

Cookie bomb

Frame flood, using GIF with a huge frame ReDoS Regex DoS

CPDoS Cache Poisoned Denial of Service)

# Other Test Cases All Categories)

Check for security headers and at least X Frame Options

XXSS header HSTS header CSP header Referrer Policy Cache Control Public key pins

Testing for Role authorization

Check if normal user can access the resources of high privileged users? Forced browsing

Insecure direct object reference

Parameter tampering to switch user account to high privileged user Blind OS command injection

using time delays

by redirecting output

with out-of-band interaction

with out-of-band data exfiltration

Command injection on CSV export Upload/Download) CSV Excel Macro Injection

If you find phpinfo.php file, check for the configuration leakage and try to exploit any network vulnerability. Parameter Pollution Social Media Sharing Buttons

Broken Cryptography

Cryptography Implementation Flaw Encrypted Information Compromised Weak Ciphers Used for Encryption

Web Services Testing

Test for directory traversal

Web services documentation disclosure Enumeration of services, data types, input types boundaries and limits

**Created by: Tushar Verma(e11i0t\_4lders0n)**

# Contact Me: [LinkedIn](http://www.linkedin.com/in/tushars25) ,[Twitter](https://twitter.com/e11i0t_4lders0n)