Appx B
(Ref to para 6(b))

DESKTOP APPLICATION CHECKLIST

Ser		Controls	Test Conducted	Vulnerability Detected	Remarks
INFORMATION GATHERING					
1.	Information Gathering				
	(a)	Find Out the application architecture (two- tier or three-tier)			
	(b)	Find out the technologies used (languages and frameworks)			
	(c)	Identify network communication			
	(d)	Observe the application process			
	(e)	Observe each functionality and behavior of the application			
	(f)	Identify all the entry points			
	(g)	Analyze the security mechanism (authorization and authentication)			
	(h)	Tools Used : CFF Explorer, Sysinternals Suite, Wireshark, PEid, Detect It Easy (DIE), Strings			
GUI TESTING					
2.	Test For GUI Object Permission				
	(a)	Display hidden form object			
	(b)	Try to activate disabled functionalities			
	(c)	Try to uncover the masked password			
3.	Test GUI Content				
	(a)	Look for sensitive information			
4.	Test For GUI Logic				
	(a)	Try for access control and injection-based vulnerabilities			
	(b)	Try for access control and injection-based vulnerabilities			
	(c)	Check improper error handling			
	(d)	Check weak input sanitization			
	(e)	Try privilege escalation (unlocking admin features to normal users)			
	(f)	Try payment manipulation			
	(h)	Tools Used : UISpy, Winspy++, Window Detective, Snoop WPF			

FILE TESTING					
5.	Test For Files Permission				
	(a)	Check permission for each and every file and folder			
6.	Test For File Continuity				
	(a)	Check strong naming			
	(b)	Authenticate code signing			
7.	Test For File Content Debugging				
	(a)	Look for sensitive information on the file system (symbols, sensitive data, passwords, configurations)			
	(b)	Look for sensitive information on the config file			
	(c)	Look for Hardcoded encryption data			
	(d)	Look for Clear text storage of sensitive data			
	(e)	Look for side-channel data leakage			
	(f)	Look for unreliable log			
8.	Test For File And Content Manipulation				
	(a)	Try framework backdooring			
	(b)	Try DLL preloading			
	(c)	Perform Race condition check			
	(d)	Test for Files and content replacement			
	(e)	Test for Client-side protection bypass using reverse engineering			
9.	Test For Function Exported				
	(a)	Try to find the exported functions			
	(b)	Try to use the exported functions without authentication			
10.	Test For Public Methods				
	(a)	Make a wrapper to gain access to public methods without authentication			
11.	Test For Decompile And Application Rebuild				
	(a)	Try to recover the original source code, passwords, keys			
	(b)	Try to decompile the application			
	(c)	Try to rebuild the application			
	(d)	Try to patch the application			
12.	Test For Decryption And DE obfuscation				
	(a)	Try to recover original source code			
	(b)	Try to retrieve passwords and keys			
	(c)	Test for lack of obfuscation			

13.	Test For Decryption And DE obfuscation				
	(a)	Try to recover original source code			
	(b)	Try to retrieve passwords and keys			
	(c)	Test for lack of obfuscation			
	Test For Disassemble and Reassemble				
	(a)	Try to build a patched assembly			
	(b)	Tools Used : Strings, dnSpy, Procmon, Process Explorer, Process Hacker			
REGISTRY TESTING					
14.	Test For Registry Permissions				
	(a)	Check read access to the registry keys			
	(b)	Check to write access to the registry keys			
15.	Test For Registry Contents				
	(a)	Inspect the registry contents			
	(b)	Check for sensitive info stored on the registry			
	(c)	Compare the registry before and after executing the application			
16.	Test For Registry Manipulation				
	(a)	Try for registry manipulation			
	(b)	Try to bypass authentication by registry manipulation			
	(c)	Try to bypass authorization by registry manipulation			
	(d)	Tools Used : Reshot, Procmon, Accessenum			
NETWORK TESTING					
17.	Test For Network				
	(a)	Check for sensitive data in transit			
	(b)	Try to bypass firewall rules			
	(c)	Try to manipulate network traffic			
	(d)	Tools Used : Wire shark, TCPview			
ASSEMBLY TESTING					
18.	Test For Assembly				
	(a)	Verify Address Space Layout Randomization (ASLR)			
	(b)	Verify SafeSEH			
	(c)	Verify Data Execution Prevention (DEP)			
	(d)	Verify strong naming			
	(e)	Verify ControlFlowGuard			
	(f)	Verify HighentropyVA			
	(g)	Tools Used : PE Security			
MEMORY TESTING					
19.	Test For Memory Content				

	(a)	Check for sensitive data stored in memory			
20.	Test For Memory Manipulation				
	(a)	Test For Memory Manipulation			
	(b)	Try to bypass authentication by memory manipulation			
	(c)	Try to bypass authorization by memory manipulation			
21.	Test For Run Time Manipulation				
	(a)	Try to analyze the dump file			
	(b)	Check for process replacement			
	(c)	Check for modifying assembly in the memory			
	(d)	Try to debug the application			
	(e)	Try to identify dangerous functions			
	(f)	Use breakpoints to test each and every functionality			
	(h)	Tools Used : Process Hacker, HxD, Strings			
TRAFFIC TESTING					
22.	Test For Traffic				
	(a)	Analyze the flow of network traffic			
	(b)	Try to find sensitive data in transit			
	(c)	Tools Used : Echo Mirage, MITM Relay, Burp Suite			
COMMON VULNERABILITIES TESTING					
23.	Test For Common Vulnerabilities				
	(a)	Try to decompile the application			
	(b)	Try for reverse engineering			
	(c)	Try to test with OWASP WEB Top 10			
	(d)	Try to test with OWASP API Top 10			
	(e)	Test for DLL Hijacking			
	(f)	Test for signature checks (Use Sigcheck)			
	(g)	Test for binary analysis (Use Binscope)			
	(h)	Test for business logic errors			
	(i)	Test for TCP/UDP attacks			
	(j)	Test with automated scanning tools (Use Visual Code Grepper - VCG)			

Appx C
(Ref to para 6(b))

API SECURITY CHECKLIST

Ser no.	Controls	Test Conducted	Vulnerability Detected	Remarks
Authentication				
1.	Don't use Basic Auth . Use standard authentication instead (e.g., JWT).			
2.	Don't reinvent the wheel in Authentication token generation, password storage. Use the standards.			
3.	Use Max Retry and jail features in Login.			
4.	Use encryption on all sensitive data.			
JWT (JSON Web Token)				
5.	Use a random complicated key (JWT Secret) to make brute forcing the token very hard.			
6.	Don't extract the algorithm from the header. Force the algorithm in the backend (HS256 or RS256).			
7.	Make token expiration (TTL, RTTL) as short as possible.			
8.	Don't store sensitive data in the JWT payload, it can be decoded easily.			
9.	Avoid storing too much data. JWT is usually shared in headers and they have a size limit.			
Access				
10.	Limit requests (Throttling) to avoid DDoS / brute-force attacks.			
11.	Use HTTPS on server side with TLS 1.2+ and secure ciphers to avoid MITM (Man in the Middle Attack).			
12.	Use HSTS header with SSL to avoid SSL Strip attacks.			
13.	Turn off directory listings.			
14.	For private APIs, allow access only from safe listed IPs/hosts			
Authorization				
OAuth				

15.	Always validate redirect_uri server-side to allow only safe listed URLs.			
16.	Always try to exchange for code and not tokens (don't allow response_type=token).			
17.	Use state parameter with a random hash to prevent CSRF on the OAuth authorization process.			
18.	Define the default scope, and validate scope parameters for each application.			
Input				
19.	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.			
20.	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.			
21.	Validate content-type of posted data as you accept (e.g., application/x-www-form-urlencoded , multipart/form-data , application/json , etc.)			
22.	Validate user input to avoid common vulnerabilities (e.g., XSS , SQLInjection , Remote Code Execution , etc.)			
23.	Don't use any sensitive data (credentials, Passwords, security tokens, or API keys) in the URL, but use standard Authorization header.			
24.	Use only server-side encryption.			
25.	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				
26.	Check if all the endpoints are protected behind authentication to avoid broken authentication process.			
27.	User own resource ID should be avoided. Use /me/orders instead of /user/654321/orders.			
28.	Don't auto-increment IDs. Use UUID instead.			

29.	If you are parsing XML data, make sure entity parsing is not enabled to avoid XXE (XML external entity attack).			
30.	If you are parsing XML, YAML or any other language with anchors and refs, make sure entity expansion is not enabled to avoid Billion Laughs/XML bomb via exponential entity expansion attack.			
31.	Use a CDN for file uploads			
32.	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.			
33.	Do not forget to turn the DEBUG mode OFF.			
34.	Use non-executable stacks when available.			
Output				
35.	Send X-Content-Type-Options: nosniff header.			
36.	Send X-Frame-Options: deny header.			
37.	Send Content-Security-Policy: default-src 'none' header.			
38.	Remove fingerprinting headers - X-Powered-By , Server , X-AspNetVersion , etc.			
39.	Force content-type for your response. If you return application/json , then your content-type response is application/json.			
40.	Don't return sensitive data like credentials, passwords, or security tokens.			
41.	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				
42.	Audit your design and implementation with unit/integration tests coverage.			
43.	Use a code review process and disregard self-approval.			
44.	Ensure that all components of your services are statically scanned by AV software before pushing to production, including vendor libraries and other			

	dependencies.			
45.	Continuously run security tests (static/dynamic analysis) on your code.			
46.	Check your dependencies (both software and OS) for known vulnerabilities.			
47.	Design a rollback solution for deployments.			
Monitoring				
48.	Use centralized logins for all services and components			
49.	Use agents to monitor all traffic, errors, requests, and responses			
50.	Use alerts for SMS, Slack, Email, Telegram, Kibana, Cloudwatch, etc			
51.	Ensure that you aren't logging any sensitive data like credit cards, passwords, PINs, etc			
52.	Use an IDS and/or IPS system to monitor your API requests and instances.			
See also :				
53.	yosriady/api-development-tools - A collection of useful resources for building RESTful HTTP+JSON APIs.			
Contribution				
54.	Feel free to contribute by forking this repository, making some changes, and submitting pull requests. For any questions drop us an email at team@shieldfy.io .			

Appx D
(Ref to para 6(b))

MOBILE APPLICATION PENTESTING CHECKLIST

Ser	Controls	Test Conducted	Vulnerability Detected	Remarks
STATIC ANALYSIS				
1.	Reverse Engineering the Application Code (Code Obfuscating Checking)			
2.	Information leakage/Hardcoded credential in the binaries			
3.	Unauthorized Code Modification			
4.	Misuse of App permissions			
5.	Insecure version of OS Installation Allowed			
6.	Abusing Android Components through IPC intents ("exported" and "intent-filter")			
7.	Unrestricted Backup file			
8.	Cryptographic Based Storage Strength			
9.	Poor key management process			
10.	Use of custom encryption protocols			
11.	Debuggable Application			
DYNAMIC AND RUNTIME ANALYSIS				
12.	Misuse of Keychain , Touch ID and other			
13.	Minimum Device Security Requirements absent			
14.	Unencrypted Database files			
15.	Insecure Shared Storage			
16.	Insecure Application Data Storage			
17.	Information Disclosure through Logcat/Apple System Log (ASL)			
18.	Application Backgrounding (Screenshot)			
19.	Copy/Paste Buffer Caching			
20.	Keyboard Press Caching			
21.	Unrestricted Backup file			
22.	Remember Credentials Functionality (Persistent authentication)			
23.	Client Side Based Authentication Flaws			
24.	Client Side Authorization Breaches			
25.	Content Providers: SQL Injection and			

	Local File Inclusion			
26.	Broadcast Receiver			
27.	Service component			
28.	Insufficient WebView hardening			
29.	Injection (SQLite Injection, XML Injection)			
30.	Local File Inclusion through Webviews			
31.	Abusing URL schemes or Deeplinks			
32.	Sensitive Information Masking			
33.	Runtime Manipulation			
34.	Rooted or Jail-broken device checking			
35.	Passwords/ Connection String disclosure			
36.	Hidden and Unscrutinised functionalities			
COMMUNICATION CHANNEL				
37.	Insecure Transport Layer Protocols			
38.	Use of Insecure and Deprecated algorithms			
39.	Use of Disabling certificate validation			
40.	SSL pinning Implementation			
41.	End-to-end encryption			
SERVER SIDE - WEBSERVICES AND API				
42.	Excessive port opened at Firewall			
43.	Default credentials on Application Server			
44.	Weak password policy Implementation			
45.	Exposure of Webservices through WSDL document			
46.	Security Misconfiguration on Server API			
47.	Security Patching on Server API			
48.	Input validation on API			
49.	Information Exposure through API response message			
50.	Control of interaction frequency on API (Replay Attack)			
51.	Session invalidation on Backend			
52.	Session Timeout Protection			
53.	Cookie Rotation			
54.	Multiple concurrent logins			
55.	Exposing Device Specific Identifiers in Attacker Visible Elements			
56.	Token/Session Creation and handling			
57.	Insecure Direct Object references			
58.	Missing function level access control			
59.	Bypassing business logic flaws			

Appx E
(Ref to para 6(e))

Feedback

Name *	
Email *	
Feedback *	