TELECOMMUNICATION ENGINEERING CENTRE
दूरसंचार अभियांत्रिक केन्द्र
MINISTRY OF COMMUNICATIONS AND INFORMATION TECHNOLOGY
संचार एवं सूचना प्रौद्योगिकी मंत्रालय
DEPARTMENT OF TELECOMMUNICATIONS
दूरसंचार विभाग
GOVERNMENT OF INDIA
भारत सरकार
ISO 9001:2008

STUDY PAPER ON

Application Security

Table of Contents

1. Introduction:

Application security is a very important issues nowadays. Thousands of new applications hit the market each week. Application either through Mobile, web, PCs, tablets provides array of services which contains user personal information. With new applications, new security vulnerabilities are also discovered every day in commonly used applications. This vulnerability can put personal data of user at risk.

An application vulnerability is a system flaw or weakness in an application that could be exploited to compromise the security of the application. Once an attacker has found a flaw, or application vulnerability, and determined how to access it, it can exploit the application vulnerability to facilitate a cyber-crime. These crimes target the confidentiality, integrity, or availability (known as the "CIA") of resources possessed by an application, its creators, and its users.

Application security, or "AppSec," is what an organization does to protect its critical data from external threats by ensuring the security of all of the software used to run the business, whether built internally, bought or downloaded. Application security helps identify, fix and prevent security vulnerabilities in any kind of software application

Mainly there are two types of applications:

i. Mobile Applications
ii. Web Based applications

Security measures in mobile apps involves security at various levels i.e. Application, network and host levels. However, this study paper mainly covers security aspects of applications which will directly effect user's security and privacy.

2. Need for Application Security:

Security of applications is critical due to the following reasons:

i. Storage and Processing of Sensitive Data:

Mobile devices are being used to access a range of services, from social networking, banking, ticketing, and shopping to corporate applications such as email, enterprise resource planning (ERP), customer relationship management (CRM), and calendar and address book applications. The applications store and transmit a lot of sensitive personal and corporate information, such as login credentials, credit card details, private contact entries, invoices, and purchase orders. If developed insecurely, these applications could potentially disclose sensitive information.

ii. Non transparent Use of Mobile Devices:

Using personal phones for corporate purposes makes it difficult to enforce corporate policies and restrictions on these devices. Also, an attacker can more easily compromise personal devices than corporate- issued devices, which are locked down using far more draconian measures. Sensitive corporate applications and data on unmanaged personal devices open up security risks, such as exposure of confidential

corporate information through lost or stolen phones, data interception and manipulation through Wi-Fi sniffing, and man-in-the-middle attacks at public Wi-Fi hotspots.

iii. Regulatory requirements:

Around the world, countries have their own regulatory requirements for enterprises that manage sensitive and confidential customer data such as personally identifiable information, personal health information, cardholder information, and financial information. Hence organizations dealing with such information must mandate use of minimum security requirements.

3. Mobile Application Security:

Currently Smartphone have become incredibly important among people around the world are more than communication devices. They are like a personal computer, with more memory and processing power than your laptop of just a few years ago. Hence they have become the most targetable sources for hackers and malicious program.

As per GSMA statistics, total number of mobile subscription worldwide is 5108 million approx. till November 2017. A recent study from Veracode found the average global enterprise has approximately 2,400 unsafe applications installed on employees' mobile devices. Security researchers from antivirus software firm G Data have discovered that more than 750,000 new malicious apps have sprung out during the first quarter of this year, with estimates the total number will grow up to a staggering 3.5 million by the end of 2017.That's alarming, especially if each of these applications could serve as the entry point for hackers seeking to access a corporate network and obtain sensitive data.

3.1 Security Issues in Mobile Applications

3.1.1. Threat Vectors:

Many threat vectors for infecting personal computers arise from social-engineering attacks that bypass anti-virus defenses. Similar techniques are used in the smartphone and tablet world by deceiving users into installing malicious apps. Examples include apps that gather personal information, track location, and charge accounts by sending text messages to premium-rate numbers. Using a mobile device to access corporate email or other resources also extends the threat to the organization, including the theft of sensitive data.

While viruses and malware targeting mobile devices would share many of the same goals as on the PC, the enhanced capabilities of these devices present expanded attack surfaces through sensors such as GPS, accelerometer, camera, microphone, and gyroscope. Recently, Kaspersky Lab discovered a new threat involving the photo-scanning of Quick Response(QR) codes. QR codes are 2-D matrix barcodes increasingly used in advertising and merchandising to direct mobile-phone users to a website for further information on the tagged item. In this case, users download a seem to be a legitimate app, but instead was malware that sent SMS to a premium-rate number that charged for each message. This app could have easily been

reconfigured to send covert copies of emails and text messages to an intelligence gatherer instead. Hence mobile inbuilt features can act as threat vector effecting security of user.

3.1.2. Security Issues in Mobile Applications Platforms:

Since more sensitive data now resides in mobile platforms, hackers are gradually shifting their attention towards mobile environment and its platforms. Currently various types of platforms exist to deploy mobile applications with different private policies**.** Development of Mobile application on various platforms is based on functional and non-functional requirements. Also the security within each platform such as Motion BlackBerry OS, Apple iOS, Google Android, Microsoft Windows Phone is different from one another. Hence mobile platforms play an important role for maintaining, effecting the aspect of application security.

a. Security Issues in Android:

As per reports, it has been stated that 2016 was the year with the most malware appearances on Android devices. Experts estimate that 9468 new malware appear daily on average. This means that a new malware appears every 9 seconds.

Android is developed as an open source model. Android developers are free to add to the API, use third-party APIs, and distribute apps through any means as per their convenience. While all Android apps must be signed with a certificate, developers can create their own certificates without using a signature from duly certified certificate authority. Android provides the capability for greater application security than iOS, but based on security model of "trust them". Due to which a number of back doored/malicious applications have been published to the Android app market and have been distributed to users.

Android grants permissions to resources on a per-application basis during the installation of the application. The user is given a one-time option to install/not install the application after reviewing the resources requested by the application, thereby granting all the permissions or not installing the application at all. Some apps require dangerous combination of permissions which can create threat to user privacy data. For example, it may be legitimate for an application designed to provide current weather conditions to request access to GPS and networking so the user does not have to continuously input a location; but if the application also requests access to telephony (i.e., dialling phone numbers), a red flag should be raised.

In android, preferences are stored unencrypted: application and system preferences, including in some cases authentication information, credentials, and tokens, are also stored completely unencrypted on the device's file system. While this is not normally available without the user being prompted, an attacker with access to the device file system would be able to read this sensitive information

All android applications have their own unique identity, and there was a vulnerability that allowed identities to be copied so that one application could impersonate

another. This Fake ID breach allowed malicious applications to be recognized as a trusted one by the user without the user knowing about it. This could potentially allow malicious software to steal user information from a trusted application and even take control of the security mechanisms on a device. This problem arises from the Android package installer not verifying the validity of a chain of certificates.

Including these, various real world vulnerabilities have been identified in Android OS and developers are trying to fix those vulnerabilities in security updates

b. Security Issues in IOS:

Apple's "trust us" model controls security from malicious apps by providing only one outlet for app distribution and by tightly controlling the iOS Software Development Kit (SDK). Developers submitting apps for distribution must register with Apple to obtain certificates to build and deploy apps. All apps must be signed with the certificate assigned by Apple. Apps must be built using Xcode, Apple's own development tool, and apps may use only the official iOS SDK—no third-party software APIs. Apple's development program requires a yearly fee which must be kept up-to-date. Apple reserves the right to revoke the developer's certificate at any time, which will take any apps developed off the App Store and prevent the developer from distributing any further apps until restoration the certificate.

An additional security measure is sandboxing applications and their data stores. Sandboxing provides an app with its own process space and prevents the app from accessing other process spaces. Apple sandboxing does not prevent malicious attacks against an app, but it does limit the damage done by the hacked app to other parts of the device. iOS apps are not allowed to start or execute other apps.

It is said that IOS Security model is based on concept of "Trust Us" as it possesses complete control of app development for their platform, from the APIs and tools available for app development, to the distribution process, to the device itself.

But in IOS all default security is at risk if the device is jail broken and it enables download of any unauthorized application which can create risk for system. Password protection mechanisms as adopted by IOS is mainly user interface based rather than enforced by kernel.

Including this, certain vulnerabilities are also discovered in IOS and listed in CVE catalogue which allows remote attacker to execute arbitrary code or denial of service.

3.1.3. Insecure Data Storage:

This risk occurs when sensitive data is stored on a device or when cloud-synced data is left unprotected. It's generally the result of not encrypting sensitive data, caching information not intended for long-term storage, allowing global file permissions, or failing to leverage best practices for a particular platform. This
in turn leads to the exposure of sensitive information, privacy violations, and noncompliance.

3.1.4. Weak Server side controls:

Failure to implement proper security controls such as patches and updates or secure configurations, changing default accounts, or disabling unnecessary backend services can compromise data confidentiality and integrity.

3.1.5. Insufficient Transport Layer Protection:

Mobile applications use HTTP protocol for client-server communication which communicates all information in plain text. Even when they provide transport-layer security through use of the HTTPS protocol, if they ignore certificate validation errors or revert to plain-text communication after a failure, they can jeopardize security by revealing data or facilitating data tampering through man-in-the middle attacks.

3.1.6. Client Side injection:

Apart from the known injection attacks such as XSS, HTML injection, and SQL injection applicable to mobile Web and hybrid apps, mobile apps are witnessing newer attacks such as abusing the phone dialler, SMS, and in-app payments.

3.1.7. Poor Authorization & Authentication:

Authentication and authorization are critical controls for every mobile payment application since they not only authenticate the user but also authorize the payment. Weaknesses in authorization might allow impersonation attacks with stolen data such as stolen tokens being used by a different device and user other than that they were intended for.

Attackers attempt to bypass authentication by spoofing biometric data, attempting to reset user credentials when they are lacking strong verification of the user with additional validation before reissuance of PINs and passwords.

3.1.8. Improper Session Handling:

Improper session handling can be caused by failing to validate the session at logout, poor implementations of session expiration, issues with session tokens/cookies such as replay and hijacking of the sessions because of lack of protection of session data such as cookies in transit between client and servers.

3.1.9. Side Channel Data Leakage:

Attackers could access sensitive data using side channels such as by installing malware on the device in order to control it remotely or to steal data from the device.

3.1.10. Broken Cryptography:

This risk emanates from insecure development practices, such as using custom instead of standard cryptographic algorithms, assuming that encoding and obfuscation are equivalent to encryption, or hardcoding cryptographic keys into the application code itself. Such practices can result in a loss of data confidentiality or privilege escalation.

3.1.11. Escalated Privileges:

Additional or excessive permissions acquired by application can sometimes effect privacy of individual and can cause application to get access to personal information. Also lack of proper implementation of sandboxing can cause any application to get access to unauthorised data.

3.1.12. Unlicensed and unmanaged apps:

These apps can act as a blind spot for admins and thus placing the system and privacy of users at risk.

3.2 Countermeasures/Guidelines for Mobile Application Security:

Security in an application should be considered during development phase of application itself. Application development process can be prepared specifying the phases needs to be adopted while building secure application. It also reduces the cost of security implementation and fixing issues later. The countermeasures/guidelines which should be adopted for building a secure application are mentioned below:

3.2.1 Secure Application Development Cycle:

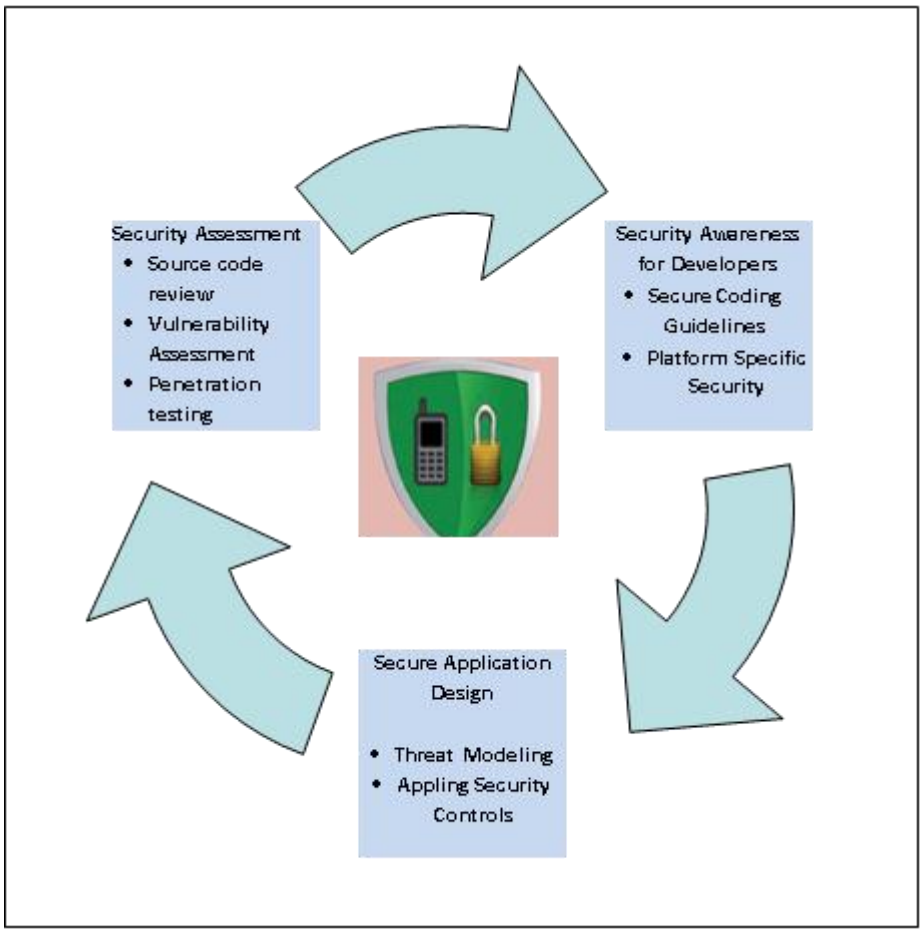

Fig1: Secure Application development process

i. **Secure Application design:**

As per business requirements: (i)users,(ii) data sensitivity and (iii) device types, Design considerations pertaining to (a) access control and privileges, (b) data encryption (in transit and local storage), (c) strong password and (d) account-lockout policies should be included in any mobile application design. An adequate threat-risk modelling of the application should also be considered in this phase to identify any security risks and determine the adequate security controls such as multifactor authentication, digital signature and TLS/SSL encryption based on criticality of application.

ii. **Secure coding guidelines:**

Developer of applications should be aware of following guidelines while developing an app:

a. Perform secure logging and error handling
b. Follow the principles of least privilege with proper sandboxing implementation and isolation.
c. Validate input data both on client as well as server side and proper security controls must be implemented for input validations.
d. Avoid storing sensitive data on client devices unless absolutely necessary and use standard encryption algorithms with strong key values instead of default ones to encrypt sensitive data residing on devices or at the server backend.

iii. **Security Assessment of applications:**

Security assessment of mobile applications must be conducted before being released to production to attack surface or threats discovered during threat modelling have been addressed. This also addresses the issue arising from integration of different modules in application. This is done through security testing of applications. Security Testing is focused on the inspection of the application in the runtime environment in order to find security problems. Organizations should specify security test cases based on known requirements and common vulnerabilities, and also perform application penetration testing before each major release.
Generally, for this purpose source code review tool (Static and dynamic analysis) along with vulnerability assessment and penetration testing tools can also be used.

3.2.2 Mobile device Security capabilities:

a. Trusted Execution Environment:

A TEE is a secure area that resides in the application processor of an electronic device, that runs a secure operating system in the main processor of a mobile device. The TEE includes key-storage and management functionalities. It also includes secure storage, which can be used to store transaction logs and authentication credentials in a private

area. Separated by hardware from the main operating system, a TEE ensures the secure storage and processing of sensitive data and trusted applications. It

- implements an isolated computing resource in the mobile device,
- takes control over mobile vulnerable resources, particularly peripherals
- provides security properties to the executing critical application so that the application is immune to malicious software threats, and
- complements the security execution environment offered by an SE, including the provision of any encryption functionality

It protects the integrity and confidentiality of key resources, such as the user interface and service provider assets. A TEE manages and executes trusted applications built in by device makers, as well as trusted applications installed as people demand them. Trusted applications running in a TEE have access to the full power of a device's main processor and memory, while hardware isolation protects these applications from user-installed apps running in a main operating system. The software and cryptographic isolation inside the TEE protect the trusted applications contained within from each other.

A typical mobile with TEE has Rich OS & TEE OS, only TEE OS can access the application process which needs to be secured. Rich OS will access TEE OS through API which in turn will carry out the secure processing within TEE (in protected service area). The following figure depicts such a mechanism:

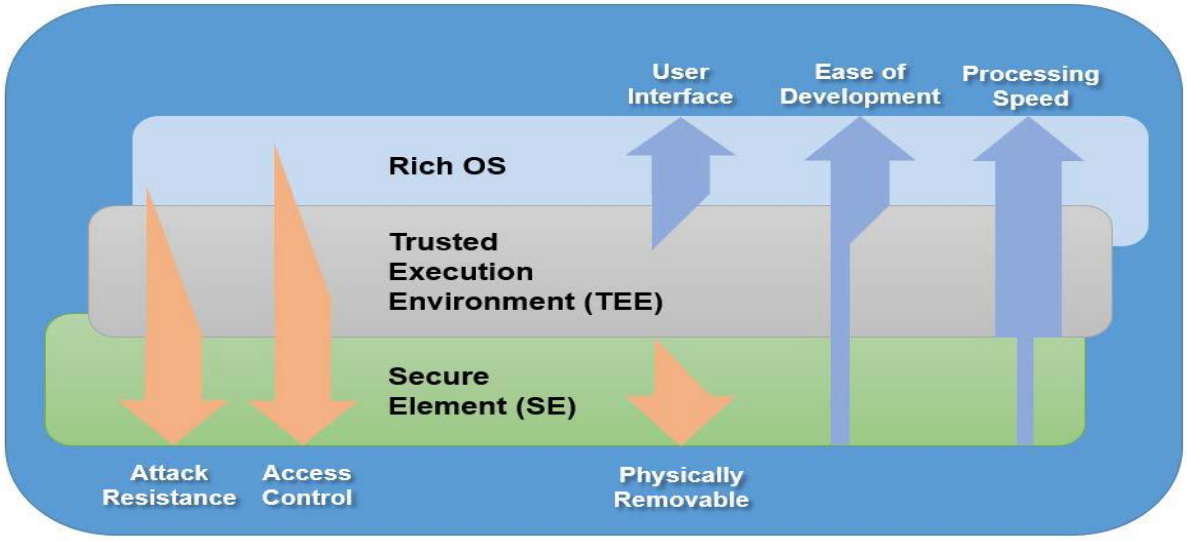

Fig2:Rich OS,TEE OS and SE positioning

b. Secure Element / TPM(Trusted Platform Module)

A Secure Element (SE) is a tamper-resistant platform (typically a one chip secure microcontroller) capable of securely hosting applications and their confidential and cryptographic data (e.g. key management) in accordance with the rules and security requirements set forth by a set of well-identified trusted authorities. The TPM is an implementation of the functions defined in the TCG TPM 2.0 Library Specification. The TPM includes some RoT, shielded locations, and protected capabilities. In general, the TPM provides a set of functions and data that enable platforms to be trustworthy. The

TPM provides methods for collecting and reporting the identities of hardware and software components of a platform for the purposes of establishing trust in that platform.

3.2.3 Guidelines for Mobile Application Security:

Risk factor in mobile device mainly has four stages such as application coding, Application distribution, Application Configuration and Configuration of device. Following steps should be adopted to maintain security in application:

i. Mobile have trusted execution environment inbuilt in their architecture. This can also be used to store critical applications such as financial applications etc.
ii. Secure element also along with TEE can be used to provide separate and secure space for critical and important applications.
iii. Mobile applications should be designed with built-in capabilities of Root/Jailbreak detection, tamper resistance against reverse engineering, multilayer authentication leveraging voice, fingerprinting, image, and geolocation.
iv. Application stores for different mobile device vendors use different security vetting processes. It's important to make sure applications aren't corrupted during the distribution process.
v. Temper detection and code obfuscation mechanisms should be implemented strictly in mobile handsets.
vi. Store the sensitive data properly
vii. Proper authentication mechanisms should be in place.
viii. Leveraging code obfuscation and anti-tampering to prevent reverse engineering.
ix. Use security testing tools such as Vulnerability assessment and penetration testing etc. in regular manner
x. Security and authenticity of 3rd party libraries should be validated. Binary testing tool can also help in security testing of 3rd party codes/libraries.
xi. A white-list of suitable, applicable and safe applications and software should be published within the organization and centrally imposed on all devices.
xii. Mobile devices should be controlled centrally to enable entity-wide configurations, remote data management, remote data recovery and data wipe.

4. Web Applications Security:

In the recent years, web applications have become primary targets of attacks. The National Vulnerability Database (NVD) maintained by the National Institute of Standards and Technology (NIST) has more than 18,500 vulnerabilities. These include 2,757 buffer overflow, 2,147 cross-site scripting (XSS), and 1,600 SQL injection vulnerabilities. XSS and SQL injection vulnerabilities occur mostly in web-based applications.

Web application security is difficult because these applications are, by definition, exposed to the general public, including malicious users. Additionally, input to web applications

comes from within HTTP requests. Correctly processing this input is difficult. The incorrect or missing input validation causes most vulnerabilities in web applications.

Network firewalls, network vulnerability scanners, and the use of Secure Socket Layer (SSL) are generally used but do not make a web site secure fully. It is estimated that over 70% of attacks against a company's web site or web application come at the application layer, not the network or system layer.

4.1 Web Application: Definition

The Web Application Security Consortium (WASC) defines a web application as "a software application, executed by a web server, which responds to dynamic web page requests over HTTP." A web application is comprised of a collection of scripts, which reside on a web server and interact with databases or other sources of dynamic content. Using the infrastructure of the Internet, web applications allow service providers and clients to share and manipulate information in a platform-independent manner.

4.2 Security issues in Web applications:

The Open Web Application Security Project (OWASP) publishes the list of the most critical web application vulnerabilities. Some of common vulnerabilities and attacks (application level) as identified in CWE catalogue are:

i. **Cross-site scripting (XSS) vulnerabilities**:

The vulnerability occurs when an attacker submits malicious data to a web application. Examples of such data are client-side scripts and hyperlinks to an attacker's site. If the application gathers the data without proper validation and dynamically displays it within its generated web pages, it will display the malicious data in a legitimate user's browser. As a result, the attacker can manipulate or steal the credentials of the legitimate user, impersonate the user, or execute malicious scripts on the user's machine. This attack mainly targets application user and uses application as vehicle for attack.

ii**. Injection vulnerabilities:**

This includes data injection, command injection, resource injection, and SQL injection. SQL Injection occurs when a web application does not properly filter user input and places it directly into a SQL statement. This can allow disclosure and/or modification of data in the database. Another possible object of injection is executable scripts, which can be coerced into doing unanticipated things.

iii. **Cookie Poisoning:**

Cookie poisoning is a technique mainly for achieving impersonation and breach of privacy through manipulation of session cookies, which maintain the identity of the client. By forging these cookies, an attacker can impersonate a valid client, and thus gain information and perform actions on behalf of the victim.

iv. **Unvalidated input:**

XSS, SQL Injection, and cookie poisoning vulnerabilities are some of the specific instances of this problem. In addition, it includes tainted data and forms, improper use of hidden

fields, use of unvalidated data in array index, in function call, in a format string, in loop condition, in memory allocation and array allocation.

v. **Broken authentication and session management**:

Account credentials and session tokens not properly protected, allowing attackers to compromise passwords, keys, session cooker or tokens, and can determine the identities of other users. This can also cause data tampering.

vii. **Improper Error handling:**
Incorrect error handling and reporting may reveal information thus opening doors for malicious users to guess sensitive information. This includes catch NullPointerException, empty catch block, overly-broad catch block and overly-broad "throws" declaration.

viii. **Denial of service:**

Attackers consume Web application resources to the point where other legitimate users can no longer access or use the application. Attackers can also lock users out of their accounts or cause an application to fail.

ix. **Other Vulnerabilities:**

Authentication, authorization and access control vulnerabilities could allow malicious user to gain control of the application or backend servers. This includes weak password management, use of poor encryption methods, use of privilege elevation, use of insecure macro for dangerous functions, use of unintended copy, authentication errors, and cryptographic errors.

4.3 Countermeasures for Web Applications Security

Security of web applications should also be kept in mind during development phase itself. The Secure Application Development Cycle as mentioned in fig. 1 above will also be applicable to web applications. The points as explained above needs to be considered while developing web applications.

4.3.1. Countermeasures to prevent web application vulnerabilities is as follows:

i. Countermeasures to prevent XSS include:

a Perform thorough input validation. Applications must ensure that input from query strings, form fields, and cookies are valid for the application. Consider all user input as possibly malicious, and filter or sanitize for the context of the downstream code. All inputs must be validated for known valid values and then reject all other input. Use regular expressions to validate input data received via HTML form fields, cookies, and query strings.

b Use HTMLEncode and URLEncode functions to encode any output that includes user input. This converts executable script into harmless HTML.

ii. Countermeasures to prevent SQL injection:

a. Perform thorough input validation. Your application should validate its input prior to sending a request to the database.

b. Parameterized stored procedures must be used for database access to ensure that input strings are not treated as executable statements. If stored procedures could not be used, then SQL parameters should be used during building SQL commands.
c. Use least privileged accounts to connect to the database.

iii. Countermeasures to prevent network eavesdropping include:

a. Use authentication mechanisms that do not transmit the password over the network such as Kerberos protocol or Windows authentication.
b. Make sure passwords are encrypted (if you must transmit passwords over the network) or use an encrypted communication channel, for example with SSL.

iv. Countermeasures to prevent cookie replay include:

a. Use an encrypted communication channel provided by SSL whenever an authentication cookie is transmitted.
b. Use a cookie timeout to a value that forces authentication after a relatively short time interval. Although this doesn't prevent replay attacks, it reduces the time interval in which the attacker can replay a request without being forced to re-authenticate because the session has timed out.

v. Countermeasures to prevent data tampering include:

a. Use strong access controls to protect data in persistent stores to ensure that only authorized users can access and modify the data.
b. Use role-based security to differentiate between users who can view data and users who can modify data.

vi. Countermeasures to prevent unauthorized access to administration interfaces include:

a. Minimize the number of administration interfaces.
b. Use strong authentication, for example, by using certificates.
c. Use strong authorization with multiple gatekeepers.
d. Consider supporting only local administration. If remote administration is absolutely essential, use encrypted channels, for example, with VPN technology or SSL, because of the sensitive nature of the data passed over administrative interfaces. To further reduce risk, also consider using IPSec policies to limit remote administration to computers on the internal network.

vii. Countermeasures to prevent session hijacking/broken session management:

a. Use SSL to create a secure communication channel and only pass the authentication cookie over an HTTPS connection.
b. Implement logout functionality to allow a user to end a session that forces authentication if another session is started.
c. Limit the expiration period on the session cookie if you do not use SSL. Although this does not prevent session hijacking, it reduces the time window available to the attacker.

4.3.2. **Security testing:**

Different security testing techniques such as Vulnerability assessment and penetration testing of web applications also help web developers discover and locate system vulnerabilities and may remediate those vulnerabilities.

5. Conclusions:

The emergence of open mobile platforms and the convergence of mobile and the 'web' has created a vibrant and dynamic mobile ecosystem that enables individuals to shape and present rich personal identities online, connect with communities of their choice, and engage with innovative, applications and services. Much of this relies on the real-time access and use of personal information that is often transferred globally between applications, devices, and companies. This also poses and create individual and private information of user at risk.

Mobile application security not only includes just code running on devices but also other multiple factors such as device platform, cloud services, web services etc. A proper software development lifecycle should be adopted before app development and subsequently its deployment. TEC is also going to establish security test lab which is having vulnerability assessment and penetration testing tool, Source Code review tool and binary testing tool which are capable of performing application tests during its various development and deployment phases.

By adapting proper development lifecycle and subsequent proper security testing of any app, application vulnerabilities can be identified and can be eliminated well in advance before deploying and thereby resulting in considerable saving in investment.

6. Glossary

ERP	Enterprise Resource Planning
CRM	Customer relationship Management
QR	Quick Response
SDK	Software Development Kit
HTTP	Hyper Text Transfer Protocol
XSS	Cross site Scripting
HTML	Hypertext Markup Language
SQL	Structured Query Language
NVD	National Vulnerability Database
SSL	Secure Socket Layer
CWE	Common Weakness Enumeration
OWASP	Open Web Application Security Project

7. References:

a. NIST paper on Web Application Scanners: Definitions and Functions
b. Category: OWASP Top Ten Project
c. http:wwww.syamantec.com/connect/articles/Five common Web application vulnerabilities
d. GSMA Mobile privacy guidelines for mobile application development
e. International Journal of Computer Applications (Mobile Application Security Platforms Survey)
f. Journal on Addressing Security and Privacy Risks in Mobile Applications by TCS
g. https://msdn.microsoft.com/enus/library/ff648641.aspx#c02618429_0081/Threats and countermeasures.
h. NIST Special Publication 800-163 Vetting the Security of Mobile Applications
i. NISTIR 8144 1-Assessing Threats to 2 Mobile Devices & Infrastructure
j. Guidelines for mobile device Security-Meity