**Mobile Vulnerabilities List**

1. **Insecure Data Storage**

* **Threat Agents**

Threats agents include the following: an adversary that has attained a lost/stolen mobile device; malware or another repackaged app acting on the adversary’s behalf that executes on the mobile device.

In the event that an adversary physically attains the mobile device, the adversary hooks up the mobile device to a computer with freely available software. These tools allow the adversary to see all third-party application directories that often contain stored personally identifiable information (PII) or other sensitive information assets. An adversary may construct malware or modify a legitimate app to steal such information assets.

Insecure data storage vulnerabilities occur when development teams assume that users or malware will not have access to a mobile device’s filesystem and subsequent sensitive information in data-stores on the device. Filesystems are easily accessible. Organizations should expect a malicious user or malware to inspect sensitive data stores. Usage of poor encryption libraries is to be avoided. Rooting or jailbreaking a mobile device circumvents any encryption protections. When data is not protected properly, specialized tools are all that is needed to view application data.

* **Technical Impacts**

This can result in data loss, in the best case for one user, and in the worst case for many users. It may also result in the following technical impacts: extraction of the app’s sensitive information via mobile malware, modified apps or forensic tools. The nature of the business impact is highly dependent upon the nature of the information stolen. Insecure data may result in the following business impacts:

* Identity theft;
* Privacy violation;
* Fraud;
* Reputation damage;
* External policy violation (PCI); or
* Material loss.

1. **Insecure Communication**

* **Threat Agents**

When designing a mobile application, data is commonly exchanged in a client-server fashion. When the solution transmits its data, it must traverse the mobile device’s carrier network and the internet. Threat agents might exploit vulnerabilities to intercept sensitive data while it’s traveling across the wire. The following threat agents exist:

An adversary that shares your local network (compromised or monitored Wi-Fi); Carrier or network devices (routers, cell towers); or Malware on your mobile device.

The exploitability factor of monitoring a network for insecure communications ranges. Monitoring traffic over a carrier’s network is harder than that of monitoring a local coffee shop’s traffic. In general, targeted attacks are easier to perform.

Mobile applications frequently do not protect network traffic. They may use SSL/TLS during authentication but not elsewhere. This inconsistency leads to the risk of exposing data and session IDs to interception. The use of transport security does not mean the app has implemented it correctly. To detect basic flaws, observe the phone’s network traffic. More subtle flaws require inspecting the design of the application and the applications configuration.

* **Technical Impacts**

This flaw exposes an individual user’s data and can lead to account theft. If the adversary intercepts an admin account, the entire site could be exposed. Poor SSL setup can also facilitate phishing and MITM attacks.

1. **Insecure Authentication**

* **Threat Agents**

Threat agents that exploit authentication vulnerabilities typically do so through automated attacks that use available or custom-built tools. Once the adversary understands how the authentication scheme is vulnerable, they fake or bypass authentication by submitting service requests to the mobile app’s backend server and bypass any direct interaction with the mobile app. This submission process is typically done via mobile malware within the device or botnets owned by the attacker.

Poor or missing authentication schemes allow an adversary to anonymously execute functionality within the mobile app or backend server used by the mobile app. Weaker authentication for mobile apps is fairly prevalent due to a mobile device’s input form factor. The form factor highly encourages short passwords that are often purely based on 4-digit PINs. Authentication requirements for mobile apps can be quite different from traditional web authentication schemes due to availability requirements.

In traditional web apps, users are expected to be online and authenticate in real-time with a backend server. Throughout their session, there is a reasonable expectation that they will have continuous access to the Internet.

In mobile apps, users are not expected to be online at all times during their session. Mobile internet connections are much less reliable or predictable than traditional web connections. Hence, mobile apps may have uptime requirements that require offline authentication. This offline requirement can have profound ramifications on things that developers must consider when implementing mobile authentication.

To detect poor authentication schemes, testers can perform binary attacks against the mobile app while it is in ‘offline’ mode. Through the attack, the tester will force the app to bypass offline authentication and then execute functionality that should require offline authentication. As well, testers should try to execute any backend server functionality anonymously by removing any session tokens from any POST/GET requests for the mobile app functionality.

* **Technical Impacts**

The technical impact of poor authentication is that the solution is unable to identify the user performing an action request. Immediately, the solution will be unable to log or audit user activity because the identity of the user cannot be established. This will contribute to an inability to detect the source of an attack, the nature of any underlying exploits, or how to prevent future attacks.

Authentication failures may expose underlying authorization failures as well. When authentication controls fail, the solution is unable to verify the user’s identity. This identity is linked to a user’s role and associated permissions. If an attacker is able to anonymously execute sensitive functionality, it highlights that the underlying code is not verifying the permissions of the user issuing the request for the action. Hence, anonymous execution of code highlights failures in both authentication and authorization controls.

1. **Insecure Authorization**

* **Threat Agents**

Threat agents that exploit authorization vulnerabilities typically do so through automated attacks that use available or custom-built tools. Once the adversary understands how the authorization scheme is vulnerable, they login to the application as a legitimate user. They successfully pass the authentication control. Once past authentication, they typically force-browse to a vulnerable endpoint to execute administrative functionality. This submission process is typically done via mobile malware within the device or botnets owned by the attacker.

Testers should try to execute any privileged functionality using a low-privilege session token within the corresponding POST/GET requests for the sensitive functionality to the backend server. Poor or missing authorization schemes allow an adversary to execute functionality they should not be entitled to using an authenticated but lower-privilege user of the mobile app. Authorization requirements are more vulnerable when making authorization decisions within the mobile device instead of through a remote server. This may be a requirement due to mobile requirements of offline usability.

* **Technical Impacts**

The technical impact of poor authorization is similar in nature to the technical impact of poor authentication. The technical impact can be wide ranging in nature and dependent upon the nature of the over-privileged functionality that is executed. For example, over-privileged execution of remote or local administration functionality may result in destruction of systems or access to sensitive information.