Final Security Requirements Report

Mobile Plataform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Channel-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	Local Storage (Centralized Database)
Which DB	
Type of data stored	Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	Java ; HTML5
Input Forms	Yes
Upload Files	Yes
The system has logs	Yes
The system has regular updates	Yes
The system has third-party	Yes
System Cloud Environments	Public Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Bluetooth ; Wi-Fi ; GPS ; NFC
Data Center Phisical Access	Yes

Confidentiality

The property that ensures that information is not disclosed or made available to any unauthorized entity. In other words, information cannot be accessed by an unauthorized third party.

Note *This requirement is applied were the information is stored*.

Failure to guarantee this security requirement can lead to the leakage or loss of confidential data shared among authorized users of the application e a aplicação poderá estar sujeita aos seguintes ataques:

1. Brute Force

The attacker attempts to gain access to systems' asset (information, functionality, identity, etc.) protected by a finite secret value by using trial-and-error to exhaustively explore all the possible secret values in the hope of finding the secret (or a value that is functionally equivalent) that will unlock the asset.

2. Eavesdropping

Eavesdropping is a type of attack where the attacker tries to gain access to sensitive information of legitimate users from the messages (text, voice and video) exchanged between two or more users of Instant Messaging (IM) applications. The same applies to recorded calls, call logs and multimedia stored in clear text on memory cards.

3. Session Hijacking

This type of attack involves an adversary exploiting weaknesses in the use of an application's authentication sessions. Put another way, this type of attack results from the successful exploitation of improper authentication. Its main purpose is account hijacking. The attacker may be able to steal or manipulate an active session and use it to gain unauthorised access to the application.

4. Session Fixation

The aim of this attack is account hijacking, with the aim of fraudulently accessing the area restricted to legitimate users of a web app or a hybrid mobile app. The success of this attack depends on prior login, as it requires the session ID of a legitimate user of the target application, impersonating the victim, which grants privileges and results in the violation of confidentiality, access control and authorisation of sensitive user data, and theft of money from mobile banking app or mobile interbank application.

5. Cross Site Request Forgery

Cross-site Request Forgery (CSRF) is an attack that forces the user to execute unintended actions in an application for which it is already authenticated in a given moment. These attacks target unrequested status changes, and not directly data theft, as the attacker cannot see the answer to the forged request.

6. MITM Attacks:

In this type of attack, an attacker attempts to intrude on a mail exchange or continuous message between two users or clients of a cloud-based mobile application (client-server).

7. Server Side Request Forgery

We are in the presence of a Server Side Request Forgery Attack (SSRF) as long as the web application is vulnerable and redirects the attacker's requests to the internal network and exposes local services to the remote attacker, which introduces different forms of risks.

8. Sniffing Attacks

Sniffing is the act of obtaining data in real time from data transmitted between smartphones (or tablets) and with the Cloud, through a network (Bluetooth, Wireless Sensor Network (WSN), Wi-Fi, 3G/4G/5G, etc.)

9. Buffer Overflow

Buffer overflows is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. It can be triggered by non-validated inputs that are designed to execute code.

10. Spoofing

Spoofing attacks are a fraudulent act in which an entity fakes its identity to attempt to access resources and critical data. There are four variants of the spoofing attack, namely, content spoofing, identity spoofing, resource location spoofing and action spoofing.

11. Code Inclusion

Unlike code injection, in this type of attack, an attacker exploits a weakness in the target in order to force arbitrary code to be retrieved locally or from a remote location and executed.

12. Malware-as-a-Service

Malicious code, or simply malware, are scripts or programs that can be injected in the Cloud and Mobile ecosystem. Malware are traditionally mainly classified based on two main factors: propagation strategy (e.g., virus or worm) and malicious activity (trojan horse, spyware, adware, rootkits, botnets, ransomware, backdoors and key-loggers).

13. Code Injection

This type of attack targets injecting malicious code into user inputs from the interface (web application forms or hybrid mobile applications) of applications written in HTML5.

14. Command Injection Attacks

This is a class of attacks to which web applications are susceptible, resulting from the semantic gap existing between database interpretation and web application interpretation, as well as from the inappropriate handling of user input.

15. SQL Injection Attack

In this attack the perpetrator injects malicious command in the system to gain access to information or even to gain control of the entire system.

16. Reverse Enginnering

Reverse engineering attacks target the assets embedded in software. In such an attack scenario, the attacker by reverse engineering attempts to steal confidential information, such as embedded cryptographic keys or intellectual property in the form of algorithms.

17. Cryptanalysis

Cryptanalysis focuses on finding vulnerabilities in cryptographic algorithms and using these weaknesses to decrypt the ciphertext without knowing the secret key.

18. Cellular Rogue Base Station Attacks

Cellular Rogue Base Station is a security threat targeting a mobile phone network that can exploit the radio interface between smartphones and base stations, potentially launching passive or active attacks against user equipment. Such attacks range from acquiring the International Mobile Subscriber Identifier (IMSI) of subscribers, DoS, leaking private information on 4G networks and eavesdropping.

19. Rogue Access Points

The Access Points (AP) in a Wi-Fi networks are subject to the attack of access point spoofing, usually called Rogue Access Point (RAP). This attack consists of cloning the Media Access Control (MAC) address and Service Set IDentifier (SSID) of an AP, giving rise to the emergence of a fake access point posing as a genuine one, leading users to connect to this new network as if they were connecting to the genuine network. After connection, an attack is able to eavesdrops on its communication to hijack client's communication, re-direct clients to malicious websites, steal credentials (session hijacking) of the clients connecting to it.

20. GPS Spoofing Attacks

GPS has grown to such an extent that today it is considered a ubiquitous technology that provides positioning, navigation and timing services (PNT). As an essential element of the global information infrastructure, GPS cybersecurity faces serious challenge issues. Although some mission-critical systems even rely on

GPS as a security measure, GPS in civilian version has no protection against malicious acts such as spoofing. According, "GPS spoofing breaches authentication by forging satellite signals to mislead users with wrong location/timing data that threatens homeland security".

21. Access Point hijacking

This type of attack is a variant of the session hijacking attack and targets the AP access credentials of legitimate administrators. These credentials can be extracted through a sniffing, brute force or MiTM attack. After this, the attacker is able to carry out other types of attacks, such as DoS and Rogue Access Points.

22. NFC Payment replay attacks

This type of attack targets the exploitation of vulnerabilities in the European Visa and Mastercad (EMV) wireless communication protocol between the smartcard and the payment terminal, namely, the authenticity of the payment terminal is not guaranteed to the customer's payment device and the banking data exchanged between the customer's payment device (smartcard) and the point of sale terminal are not encrypted and are transferred in clear text.

23. Wi-Fi SSID Tracking Attacks

This type of attack aims to obtain sensitive data (location, routine, trajectory, etc.) of users of mobile devices using Wi-Fi networks to access the Internet. Furthermore, it consists of using sophisticated sniffing devices to bypass authentication (for closed networks), extract and identify the MAC address of the mobile device and establish a match with its potential owner.

24. Phishing Attacks

In this type of attack scenario, an attacker can perform a phishing attack by manipulating a web link to attempt to redirect users to a false one and capture user information and account access, with the final objective of stealing sensitive data. Main attacks vectors are e-mail keyloggers through trojan horses and Man-in-the-Middle Attack of data proxies.

25. Pharming Attacks

Pharming is a special type of phishing attack or DNS poisoning attack in which the user is redirected to a fake website by changing the IP address on the DNS server. The target of the attack is the same as the phishing attack, i.e. theft of sensitive data and money from legitimate users of the applications.

26. Malicious QR Code

Although this type of attack also affects the category or application layer, it also falls under the category of Social Engineering attacks as it mainly relies on human behaviour and its vulnerabilities. For example, the victim may be tricked into reading a malicious QR code advertising a fake promotional campaign for a product from a supermarket car park.

27. On-Off Attacks

This type of attack targets a wireless sensor network, aiming to disrupt a trust redemption scheme, behaving alternately good and bad, in order to ensure immediate trust redemption before another attack occurs. On the other hand, this occurs because there is a security vulnerability that has to do with the fact that not all trust redemption schemes are able to effectively discriminate an On-Off attack and temporary errors, especially when it is good most of the attacker's behaviour, making him able to remain active in the system, as he has the ability to disguise attacks as temporary errors.

28. Mobile SIM Swapping

This attack targets the SIM card of a smartphone user, i.e. swapping the victim user's SIM card. SIM swapping can happen remotely. A cybercriminal, with a few important details about your life in hand, can answer security questions correctly, impersonate you, and convince your mobile carrier to reassign your phone number to a new SIM card. At that point, the criminal can get access to your phone’s data and start changing your account passwords to lock you out of your online banking profile, email, and more.

References

1. [https://capec.mitre.org/data/definitions/651.html];
2. [https://capec.mitre.org/data/definitions/112.html].

Integrity

Is the property of safeguarding the correctness and completeness of assets in a Cloud & Mobile system. In other words it involves maintaining the data consistent, trustworthy and accurate during it life-cycle.

Note: *This requirements is applied in the Cloud and Mobile Ecosystem.*

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. SQL Injection Attacks:

In this type of attack, the attacker inserts malicious code with the intention of accessing the unauthorized database for the purpose of obtaining confidential or critical data from the legitimate user.

2. Wrapping Attacks:

In a wrapping attack scenario, the attacker duplicates the SOAP message in the course of the translation and sends it to the server as a legitimate user. Therefore, the attacker may interfere with the malicious code.

3. MITM Attacks:

In this type of attack, an attacker attempts to intrude on a mail exchange or continuous message between two users or clients of a cloud-based mobile application (client-server).

4. Cookie Poisoning:

This type of attack consists of replacing or modifying cookie content in ways to gain unauthorized access to applications or Web pages.

5. Session Hijacking

This type of attack involves an adversary exploiting weaknesses in the use of an application's authentication sessions. Put another way, this type of attack results from the successful exploitation of improper authentication. Its main purpose is account hijacking. The attacker may be able to steal or manipulate an active session and use it to gain unauthorised access to the application.

6. Cross Site Request Forgery

Cross-site Request Forgery (CSRF) is an attack that forces the user to execute unintended actions in an application for which it is already authenticated in a given moment. These attacks target unrequested status changes, and not directly data theft, as the attacker cannot see the answer to the forged request.

7. Server Side Request Forgery

We are in the presence of a Server Side Request Forgery Attack (SSRF) as long as the web application is vulnerable and redirects the attacker's requests to the internal network and exposes local services to the remote attacker, which introduces different forms of risks.

8. Buffer Overflow

Buffer overflows is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. It can be triggered by non-validated inputs that are designed to execute code.

9. Spoofing

Spoofing attacks are a fraudulent act in which an entity fakes its identity to attempt to access resources and critical data. There are four variants of the spoofing attack, namely, content spoofing, identity spoofing, resource location spoofing and action spoofing.

10. Audit Log Manipulation Attacks (ALM)

This type of attack targets log files for the purpose of manipulating (deleting, reading, and altering) them.

11. Code Inclusion

In this type of attack, an attacker exploits a weakness in the target in order to force arbitrary code to be retrieved locally or from a remote location and executed.

12. Malware-as-a-Service

Malicious code, or simply malware, are scripts or programs that can be injected in the Cloud and Mobile ecosystem. Malware are traditionally mainly classified based on two main factors: propagation strategy (e.g., virus or worm) and malicious activity (trojan horse, spyware, adware, rootkits, botnets, ransomware, backdoors and key-loggers).

13. Code Injection

This type of attack targets injecting malicious code into user inputs from the interface (web application forms or hybrid mobile applications) of applications written in HTML5.

14. Command Injection Attacks

This is a class of attacks to which web applications are susceptible, resulting from the semantic gap existing between database interpretation and web application interpretation, as well as from the inappropriate handling of user input

15. Reverse Enginnering

Reverse engineering attacks target the assets embedded in software. In such an attack scenario, the attacker by reverse engineering attempts to steal confidential information, such as embedded cryptographic keys or intellectual property in the form of algorithms.

16. Access Point hijacking

This type of attack is a variant of the session hijacking attack and targets the AP access credentials of legitimate administrators. These credentials can be extracted through a sniffing, brute force or MiTM attack. After this, the attacker is able to carry out other types of attacks, such as DoS and Rogue Access Points.

17. Phishing

In this type of attack scenario, an attacker can perform a phishing attack by manipulating a web link to attempt to redirect users to a false one and capture user information and account access, with the final objective of stealing sensitive data. Main attacks vectors are e-mail keyloggers through trojan horses and Man-in-the-Middle Attack of data proxies.

18. Malicious QR Code

Although this type of attack also affects the category or application layer, it also falls under the category of Social Engineering attacks as it mainly relies on human behaviour and its vulnerabilities. For example, the victim may be tricked into reading a malicious QR code advertising a fake promotional campaign for a product from a supermarket car park.

19. Malicious Insider

This type of attack occurs when there a malicious entity (client, employee, hypervisor, Cloud provider/corrector, etc.) takes advantage of its privileges to secretly realize a malicious activity, such as information theft and data or physical infrastructure destruction. This type of attack also occurs from client to server, when the person, employee or team with insider knowledge on how the system is built can implant malicious code to fully destroy the Cloud solution.

20. On-Off Attacks

This type of attack targets a wireless sensor network, aiming to disrupt a trust redemption scheme, behaving alternately good and bad, in order to ensure immediate trust redemption before another attack occurs. On the other hand, this occurs because there is a security vulnerability that has to do with the fact that not all trust redemption schemes are able to effectively discriminate an On-Off attack and temporary errors, especially when it is good most of the attacker's behaviour, making him able to remain active in the system, as he has the ability to disguise attacks as temporary errors.

21. Mobile SIM Swapping

This attack targets the SIM card of a smartphone user, i.e. swapping the victim user's SIM card. SIM swapping can happen remotely. A cybercriminal, with a few important details about your life in hand, can answer security questions correctly, impersonate you, and convince your mobile carrier to reassign your phone number to a new SIM card. At that point, the criminal can get access to your phone’s data and start changing your account passwords to lock you out of your online banking profile, email, and more.

Availability

Refers to the property which ensures that a mobile device or system is accessible and usable upon demand by authorized entities. In other words the mobile cloud-based application need to be always available to access by authorized people.

Note: *This requirement is applied were the information is stored.*

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. Flooding

In this type of attacks, commonly referred to as denial of service (DoS) or distribuited denial of service (DDoS), the attacker attempts to negatively impact of the service or resource availability from authorized users, by using different types vulnerability exploitation or flooding - SYN flooding attacks, User Datagram Protocol (UDP) flooding, Internet Control Message Protocol (ICMP) flooding, etc. - on the server.

2. Session Hijacking

This type of attack involves an adversary exploiting weaknesses in the use of an application's authentication sessions. Put another way, this type of attack results from the successful exploitation of improper authentication. Its main purpose is account hijacking. The attacker may be able to steal or manipulate an active session and use it to gain unauthorised access to the application.

3. Server Side Request Forgery

We are in the presence of a Server Side Request Forgery Attack (SSRF) as long as the web application is vulnerable and redirects the attacker's requests to the internal network and exposes local services to the remote attacker, which introduces different forms of risks.

4. Buffer Overflow

Buffer overflows is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. It can be triggered by non-validated inputs that are designed to execute code.

5. Spoofing

Spoofing attacks are a fraudulent act in which an entity fakes its identity to attempt to access resources and critical data. There are four variants of the spoofing attack, namely, content spoofing, identity spoofing, resource location spoofing and action spoofing.

6. Malware-as-a-Service

Malicious code, or simply malware, are scripts or programs that can be injected in the Cloud and Mobile ecosystem. Malware are traditionally mainly classified based on two main factors: propagation strategy (e.g., virus or worm) and malicious activity (trojan horse, spyware, adware, rootkits, botnets, ransomware, backdoors and key-loggers).

7. Code Injection

This type of attack targets injecting malicious code into user inputs from the interface (web application forms or hybrid mobile applications) of applications written in HTML5.

8. Command Injection Attacks

This is a class of attacks to which web applications are susceptible, resulting from the semantic gap existing between database interpretation and web application interpretation, as well as from the inappropriate handling of user input

9. SQL Injection Attack

In this attack the perpetrator injects malicious command in the system to gain access to information or even to gain control of the entire system.

10. GPS Jamming Attacks

This attack aims to interrupt or obstruct the communication between the emitting satellite and the device (smartphone/tablet) receiving the GPS signal. Normally, the attack consists of blocking the signal from the receiver, since the receiving signal is weaker compared to the broadcasting signal, and can be carried out in two different ways, namely, blanket jamming, and deception jamming.

11. Orbital Jamming Attacks

This type of attack targets low-orbit satellites because, although these low-orbit satellites are attractive due to the low power levels required for communications links from terrestrial terminals, they can also be vulnerable to jamming attacks when used in some applications. In fact, a jammer of reasonable power could easily saturate the RF front-end of a low-orbit satellite, resulting in disabling the link across the entire frequency band.

12. DoS (Cellular) Jamming Attack

Interference attacks target radio communication technology (communication between smart devices and base stations). This attack can be caused by noise, interference, disruption or by sending corrupted data packets, with the purpose of causing DoS in the physical transmission of signals on certain routes.

13. Bluejacking and Bluesnarfing Attacks

These are DDoS-type attacks that target a Bluetooth wireless network in order to shut down activity on it. It usually occurs through an attack coming from a connection of malicious entities in a target network.

14. Wi-Fi Jamming Attacks

This is a denial-of-service attack that blocks the radio frequency, making access to the Wi-Fi network and consequently to the Internet unavailable.

15. Access Point hijacking

This type of attack is a variant of the session hijacking attack and targets the AP access credentials of legitimate administrators. These credentials can be extracted through a sniffing, brute force or MiTM attack. After this, the attacker is able to carry out other types of attacks, such as DoS and Rogue Access Points.

16. Phishing

In this type of attack scenario, an attacker can perform a phishing attack by manipulating a web link to attempt to redirect users to a false one and capture user information and account access, with the final objective of stealing sensitive data. Main attacks vectors are e-mail keyloggers through trojan horses and Man-in-the-Middle Attack of data proxies.

17. Malicious QR Code

Although this type of attack also affects the category or application layer, it also falls under the category of Social Engineering attacks as it mainly relies on human behaviour and its vulnerabilities. For example, the victim may be tricked into reading a malicious QR code advertising a fake promotional campaign for a product from a supermarket car park.

18. Malicious Insider

This type of attack occurs when there a malicious entity (client, employee, hypervisor, Cloud provider/corrector, etc.) takes advantage of its privileges to secretly realize a malicious activity, such as information theft and data or physical infrastructure destruction. This type of attack also occurs from client to server, when the person, employee or team with insider knowledge on how the system is built can implant malicious code to fully destroy the Cloud solution.

19. Mobile SIM Swapping

This attack targets the SIM card of a smartphone user, i.e. swapping the victim user's SIM card. SIM swapping can happen remotely. A cybercriminal, with a few important details about your life in hand, can answer security questions correctly, impersonate you, and convince your mobile carrier to reassign your phone number to a new SIM card. At that point, the criminal can get access to your phone's data and start changing your account passwords to lock you out of your online banking profile, email, and more.

Authenticity

Is the assurance that information transaction is from the source it claims to be from. The device authenticates itself prior to receiving or transmitting any information. It assures that the information received is authentic. It is assumed that communications may be intercepted by an unauthorized entity and data at rest may be subject to unauthorized access during transport and rest, taking into account the nature of the cloud and mobile ecosystem.

Note: *This security requirement is applied across all layers of the ecosystem under consideration, i.e., communication, transport and storage of information shared or exchanged between authorized entities.*

Security Verification Requirements

- If the app provides users access to a remote service, some form of authentication, such as username/password authentication, is perfumed at the remote endpoint;
- if stateful session management is used, the remote andpoint uses rendomly generated session identifiers to authenticate client requests without sending the user's crendentials;
- If stateless token-base authentication is used, the server provides a token that has been signed using a secure algorithm;
- The remote endpoint terminates the existing session when the user logs out;
- A password policy exists and is enforcer at the remote endpoint;
- The remote endpoint implements a mechanism to protect against the submission of credentials an excessive number of times;
- Sessions are invalidated at the remote endpoint after a predefined period of inactivity and access number of times;
- Biometric authentication, if any, is not event-bound (i.e. using an API that simply returns "true" or "false"). Instead, it is based on unlocking the keychain/keystore;
- A second factor of authentication exists at the remote endpoint and the 2FA requirements is consistently enforced;
- Sensitive transactions require set-up authentication;
- The app informas the user of all login activities with their account. Users are able view a list of devices used to access the account, and to block specific devices.

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. Botnets Attack

A botnet is a collection of compromised devices that can be remotely controlled by an attacker, i.e. the bot master. Its main purpose is to steal business information, remote access, online fraud, phishing, malware distribution, spam emails, etc.

2. Phishing

In this type of attack scenario, an attacker can perform a phishing attack by manipulating a web link to attempt to redirect users to a false one and capture user information and account access, with the final objective of stealing sensitive data. Main attacks vectors are e-mail keyloggers through trojan horses and Man-in-the-Middle Attack of data proxies.

3. DNS Attack

DNS attacks always occur in the case where the attacker makes use of the translation of the domain name in an Internet Protocol (IP) address, in order to access the confidential data of the user in an unauthorized way

4. MITM Attack

In this type of attack, an attacker attempts to intrude on a mail exchange or continuous message between two users or clients of a cloud-based mobile application (client-server).

5. Reused IP Address Attack:

This type of attack occurs whenever a IP address is reused on a network. This occurs because in a network the number of IP addresses is usually limited, which causes an address assigned to one user to be assigned to another, so that it leaves the network.

6. Wrapping Attacks

In a wrapping attack scenario, the attacker duplicates the SOAP message in the course of the translation and sends it to the server as a legitimate user. Therefore, the attacker may interfere with the malicious code.

7. Cookie Poisoning Attack

This type of attack consists of replacing or modifying cookie content in ways to gain unauthorized access to applications or Web pages.

8. Google Hacking Attacks

This type of attack involves the use of the Google search engine for the purpose of discovering confidential information that a hacker or wrongdoer can use for their benefit by hacking the account of a used.

9. VM Escape

This type of attack occurs whenever an application escapes the VM and obtains control over the VMM, as it escalates its VM privileges to root level. The malicious application accesses the host machine, bypassing the hypervisor.

10. Session Fixation

The aim of this attack is account hijacking, with the aim of fraudulently accessing the area restricted to legitimate users of a web app or a hybrid mobile app. The success of this attack depends on prior login, as it requires the session ID of a legitimate user of the target application, impersonating the victim, which grants privileges and results in the violation of confidentiality, access control and authorisation of sensitive user data, and theft of money from mobile banking app or mobile interbank application.

11. Brute Force

The attacker attempts to gain access to systems' asset (information, functionality, identity, etc.) protected by a finite secret value by using trial-and-error to exhaustively explore all the possible secret values in the hope of finding the secret (or a value that is functionally equivalent) that will unlock the asset.

12. Buffer Overflow

Buffer overflows is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. It can be triggered by non-validated inputs that are designed to execute code.

13. Spoofing

Spoofing attacks are a fraudulent act in which an entity fakes its identity to attempt to access resources and critical data. There are four variants of the spoofing attack, namely, content spoofing, identity spoofing, resource location spoofing and action spoofing.

14. Cache Poisoning

This type of attack has to do with any attack whereby an attacker caches incorrect or harmful material. The cache targeted can be an application's cache (e.g., a web browser cache) or a public cache (fe.g., a DNS or ARP cache).

15. Reverse Enginnering

Reverse engineering attacks target the assets embedded in software. In such an attack scenario, the attacker by reverse engineering attempts to steal confidential information, such as embedded cryptographic keys or intellectual property in the form of algorithms.

15. NFC Payment replay attacks

This type of attack targets the exploitation of vulnerabilities in the European Visa and Mastercad (EMV) wireless communication protocol between the smartcard and the payment terminal, namely, the authenticity of the payment terminal is not guaranteed to the customer's payment device and the banking data exchanged between the customer's payment device (smartcard) and the point of sale terminal are not encrypted and are transferred in clear text.

16. Bypassing Physical Security

This type of attack aims to circumvent or avoid detection by physical security and building surveillance systems and use methods to bypass electronic or physical locks protecting entry points. It also results in other types of attacks aimed at accessing, altering or destroying sensitive user information or making a service or resource unavailable.

17. Physical Theft

This type of attack aims to access and steal the target user's device in order to perform a malicious action, such as altering, deleting, leaking, inserting and destroying data, as well as stealing money through banking transactions, posing as the rightful owner. The attacker can simply destroy the device, preventing the user from accessing their data and the services provided in the form of an application as a service.

18. On-Off Attacks

This type of attack targets a wireless sensor network, aiming to disrupt a trust redemption scheme, behaving alternately good and bad, in order to ensure immediate trust redemption before another attack occurs. On the other hand, this occurs because there is a security vulnerability that has to do with the fact that not all trust redemption schemes are able to effectively discriminate an On-Off attack and temporary errors, especially when it is good most of the attacker's behaviour, making him able to remain active in the system, as he has the ability to disguise attacks as temporary errors.

19. Mobile SIM Swapping

This attack targets the SIM card of a smartphone user, i.e. swapping the victim user's SIM card. SIM swapping can happen remotely. A cybercriminal, with a few important details about your life in hand, can answer security questions correctly, impersonate you, and convince your mobile carrier to reassign your phone

number to a new SIM card. At that point, the criminal can get access to your phone’s data and start changing your account passwords to lock you out of your online banking profile, email, and more.

References

- In general - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04e-Testing-Authentication-and-Session-Management.md;
- For Android - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05f-Testing-Local-Authentication.md;
- For iOS - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06f-Testing-Local-Authentication.md;
- CAPEC - https://capec.mitre.org/data/definitions/141.html.

Authorization

The property that determines whether the user or device has rights/privileges to access a resource, or issue commands.

Note: *These requirements or assumptions apply to the secure coding of PHP, C/C++, Java, C#, PHP, HTML, JavaScript, Swift programming languages in building mobile Android application and were the information might be accessed from and between the communications in the cloud and mobile ecosystem.*

Not addressing this requirement may lead to vulnerabilities explored by attacks, such as:

1. SQL Injection Attack

In this attack the perpetrator injects malicious command in the system to gain access to information or even to gain control of the entire system.

2. XSS Attack

In this attack the perpetrator injects malicious code in the system to gain access to information or even to gain control of the entire system.

3. Reused IP Address

This type of attack occurs whenever a IP address is reused on a network. This occurs because in a network the number of IP addresses is usually limited, which causes an address assigned to one user to be assigned to another, so that it leaves the network.

4. Botnet Attacks

A botnet is a collection of compromised devices that can be remotely controlled by an attacker, i.e. the bot master. Its main purpose is to steal business information, remote access, online fraud, phishing, malware distribution, spam emails, etc.

5. Sniffing Attacks

This type of attack is carried out by attackers using applications that can capture data packets in transit on a network, and if they are not heavily encrypted, can be read or interpreted.

6. Wrapping Attacks

In this attack scenario, the attacker duplicates the SOAP message in the course of the translation and sends it to the server as a legitimate user. Therefore, the attacker may interfere with the malicious code.

7. Google Hacking Attacks

This type of attack involves the use of the Google search engine for the purpose of discovering confidential information that a hacker or wrongdoer can use for their benefit by hacking the account of a used.

8. Hypervisor Attacks

Neste tipo de ataque o atacante tem como alvo comprometer a autenticidade dos dados sensíveis dos utilizadores e a disponibilidade de serviços a partir da cloud ao nível das VMs.

9. OS Command Injection

Applications are considered vulnerable to the OS command injection attack if they utilize non validated user input in a system level command what can lead to the invocation of scripts injected by the attacker.

10. Buffer Overflow

Buffer overflows is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. It can be triggered by non-validated inputs that are designed to execute code.

11. Session Fixation

The aim of this attack is account hijacking, with the aim of fraudulently accessing the area restricted to legitimate users of a web app or a hybrid mobile app. The success of this attack depends on prior login, as it requires the session ID of a legitimate user of the target application, impersonating the victim, which grants privileges and results in the violation of confidentiality, access control and authorisation of sensitive user data, and theft of money from mobile banking app or mobile interbank application.

12. Brute Force

The attacker attempts to gain access to systems' asset (information, functionality, identity, etc.) protected by a finite secret value by using trial-and-error to exhaustively explore all the possible secret values in the hope of finding the secret (or a value that is functionally equivalent) that will unlock the asset.

13. Cross Site Request Forgery

Cross-site Request Forgery (CSRF) is an attack that forces the user to execute unintended actions in an application for which it is already authenticated in a given moment. These attacks target unrequested status changes, and not directly data theft, as the attacker cannot see the answer to the forged request.

14. MITM Attacks:

In this type of attack, an attacker attempts to intrude on a mail exchange or continuous message between two users or clients of a cloud-based mobile application (client-server).

15. VM Escape

This type of attack occurs whenever an application escapes the VM and obtains control over the VMM, as it escalates its VM privileges to root level. The malicious application accesses the host machine, bypassing the hypervisor.

16. Malware-as-a-Service

Malicious code, or simply malware, are scripts or programs that can be injected in the Cloud and Mobile ecosystem. Malware are traditionally mainly classified based on two main factors: propagation strategy (e.g., virus or worm) and malicious activity (trojan horse, spyware, adware, rootkits, botnets, ransomware, backdoors and key-loggers).

17. Reverse Enginnering

Reverse engineering attacks target the assets embedded in software. In such an attack scenario, the attacker by reverse engineering attempts to steal confidential information, such as embedded cryptographic keys or intellectual property in the form of algorithms.

18. Phishing

In this type of attack scenario, an attacker can perform a phishing attack by manipulating a web link to attempt to redirect users to a false one and capture user information and account access, with the final objective of stealing sensitive data. Main attacks vectors are e-mail keyloggers through trojan horses and Man-in-the-Middle Attack of data proxies.

19. Malicious Insider

This type of attack occurs when there a malicious entity (client, employee, hypervisor, Cloud provider/corrector, etc.) takes advantage of its privileges to secretly realize a malicious activity, such as information theft and data or physical infrastructure destruction. This type of attack also occurs from client to server, when the person, employee or team with insider knowledge on how the system is built can implant malicious code to fully destroy the Cloud solution.

20. Mobile SIM Swapping

This attack targets the SIM card of a smartphone user, i.e. swapping the victim user's SIM card. SIM swapping can happen remotely. A cybercriminal, with a few important details about your life in hand, can answer security questions correctly, impersonate you, and convince your mobile carrier to reassign your phone number to a new SIM card. At that point, the criminal can get access to your phone’s data and start changing your account passwords to lock you out of your online banking profile, email, and more.

References

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transaction_Authorization_Cheat_Sheet.md

Non-Repudiation

The security property that ensures that the transfer of messages or credentials between 2 mobile users entities is undeniable .

Note: *This requirement is applied between information transactions, between information transactions over the Internet in the Cloud and in the database.*

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. VM Escape

This type of attack occurs whenever an application escapes the VM and obtains control over the VMM, as it escalates its VM privileges to root level. The malicious application accesses the host machine, bypassing the hypervisor.

2. Mobile SIM Swapping

This attack targets the SIM card of a smartphone user, i.e. swapping the victim user's SIM card. SIM swapping can happen remotely. A cybercriminal, with a few important details about your life in hand, can answer security questions correctly, impersonate you, and convince your mobile carrier to reassign your phone number to a new SIM card. At that point, the criminal can get access to your phone’s data and start changing your account passwords to lock you out of your online banking profile, email, and more.

Accountability

The property that ensures that every action can be traced back to a single user or device.

Note: *This requirement is applied over Internet transactions.*

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. DNS Attacks

DNS attacks always occur in the case where the attacker makes use of the translation of the domain name in an Internet Protocol (IP) address, in order to access the confidential data of the user in an unauthorized way.

2. MITM Attacks

In this type of attack, an attacker attempts to intrude on a mail exchange or continuous message between two users or clients of a cloud-based mobile application (client-server).

3. VM Escape

This type of attack occurs whenever an application escapes the VM and obtains control over the VMM, as it escalates its VM privileges to root level. The malicious application accesses the host machine, bypassing the hypervisor.

Reliability

Refers to the property that guarantees consistent intended behavior of an a general system, in this case applied to cloud and mobile ecosystem.

Note: *This requirement is applied over Internet transactions in the cloud and mobile ecosystem.*

Physical Security

Refers to the security measures designed to deny unauthorized physical access to mobile devices and equipment, and to protect them from damage or in other words gaining physical access to the device won't give access to it's information.

Note: *This requirement is applied were the information is stored in the device.*

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. Bypassing Physical Security

This type of attack aims to circumvent or avoid detection by physical security and building surveillance systems and use methods to bypass electronic or physical locks protecting entry points. It also results in other types of attacks aimed at accessing, altering or destroying sensitive user information or making a service or resource unavailable.

2. Physical Theft

This type of attack aims to access and steal the target user's device in order to perform a malicious action, such as altering, deleting, leaking, inserting and destroying data, as well as stealing money through banking transactions, posing as the rightful owner. The attacker can simply destroy the device, preventing the user from accessing their data and the services provided in the form of an application as a service.

Forgery Resistance

Is the propriety that ensures that the contents shared between entities cannot be forged by a third party trying to damage or harm the system or its users. In other words no one can try to forge content and send it in the name of another entities.

Note: *This requirement is applied in the device, in the cloud, and in the database.*

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. Tampering

This type of attacks occurs when an attacker preforms physical modifications on the hardware where the software is implemented.

2. Reused IP Address Attack

In this attack some nods are made more attractive than others by tampering with the routing information, when arriving to the sinkhole node the messages may be dropped or altered.

Tamper Detection

Ensures all devices are physically secured, such that any tampering attempt is detected.

Note: *This requirement is applied were the information in the device.*

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. Tampering

Is when an attacker performs physical modifications on the hardware where the software is implemented.

Data Freshness

Status that ensures that data is the most recent, and that old messages are not mistakenly used as fresh or purposely replayed by perpetrators. In other words this requirement provides the guarantee that the data displayed is the most recent.

Note: *This requirement is applied to the cloud, since it says that messages sent between components of the cloud and mobile ecosystem can be captured and forwarded, by hypothesis and between the communications.*

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. Tampering

This type of attacks occurs when a attacker preforms physical modifications on the hardware where the software is implemented.

2. Reused IP Address Attack

In this attack some nods are made more attractive than others by tampering with the routing information, when arriving to the sinkhole node the messages may be dropped or altered.

Confinement

Ensures that even if a party is corrupted, the spreading of the effects of the attack is as confined as possible.

Note: *This requirement is applied in the entire system.*

Interoperability

Is the propriety that ensures that different software communicates and works well with each-other, sharing resources such as network, processing and memory without constraints.

Note: *This requirement is applied in the entire system.*

Data Origin Authentication

Ensures that the data being received by the software comes from the source it claims to be. In other words it ensures that the data being received is authentic and from a trusted party.

Note: *This requirement is applied between the communications.*

Not addressing this requirement may lead to vulnerabilities explored by attacks such as:

1. MITM attack:

This type of attacks occurs when an attacker gains access to a packet and re-sends it when it’s beneficial to him, resulting in him gaining the trust of the system.

2. Brute Force

The attacker attempts to gain access to systems' asset (information, functionality, identity, etc.) protected by a finite secret value by using trial-and-error to exhaustively explore all the possible secret values in the hope of finding the secret (or a value that is functionally equivalent) that will unlock the asset.

3. VM Escape

This type of attack occurs whenever an application escapes the VM and obtains control over the VMM, as it escalates its VM privileges to root level. The malicious application accesses the host machine, bypassing the hypervisor.

4. Spoofing

Spoofing attacks are a fraudulent act in which an entity fakes its identity to attempt to access resources and critical data. There are four variants of the spoofing attack, namely, content spoofing, identity spoofing, resource location spoofing and action spoofing.

5. Cache Poisoning

This type of attack has to do with any attack whereby an attacker caches incorrect or harmful material. The cache targeted can be an application's cache (e.g., a web browser cache) or a public cache (fe.g., a DNS or ARP cache).

Final Security Good Practices

Mobile Plataform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Channel-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	Local Storage (Centralized Database)
Which DB	
Type of data stored	Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	Java ; HTML5
Input Forms	Yes
Upload Files	Yes
The system has logs	Yes
The system has regular updates	Yes
The system has third-party	Yes
System Cloud Environments	Public Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Bluetooth ; Wi-Fi ; GPS ; NFC
Data Center Phisical Access	Yes

SQL Injection

An SQL injection attack consists of insertion or "injection" of either a partial or complete SQL query via the data input or transmitted from the client (browser) to the web application. SQL Injection flaws are introduced when software developers create dynamic database queries that include user supplied input. To avoid SQL injection flaws is simple. Developers need to either:

- stop writing dynamic queries;
- prevent user supplied input which contains malicious SQL from affecting the logic of the executed query.

Primary Defenses:

- Option 1: Use of Prepared Statements (with Parameterized Queries)
- Option 2: Use of Stored Procedures
- Option 3: White List Input Validation
- Option 4: Escaping All User Supplied Input

Additional Defenses:

- Also: Enforcing Least Privilege
- Also: Performing White List Input Validation as a Secondary Defense

Input Validation

Input validation is performed to ensure only properly formed data is entering the workflow in an information system, preventing malformed data from persisting in the database and triggering malfunction of various downstream components.

Implementing input validation

- Data type validators available natively in web application frameworks;
- Validation against JSON Schema and XML Schema (XSD) for input in these formats;
- Type conversion (e.g. Integer.parseInt() in Java, int() in Python) with strict exception handling;
- Minimum and maximum value range check for numerical parameters and dates;
- Minimum and maximum length check for strings;
- Array of allowed values for small sets of string parameters (e.g. days of week);
- Regular expressions for any other structured data covering the whole input string (^...$) and not using "any character" wildcard (such as . or \S)

If it's well structured data, like dates, social security numbers, zip codes, e-mail addresses, etc. then the developer should be able to define a very strong validation pattern, usually based on regular expressions, for validating such input. If the input field comes from a fixed set of options, like a drop down list or radio buttons, then the input needs to match exactly one of the values offered to the user in the first place. Free-form text, especially with Unicode characters, is perceived as difficult to validate due to a relatively large space of characters that need to be whitelisted. The primary means of input validation for free-form text

input should be:

- Normalization: Ensure canonical encoding is used across all the text and no invalid characters are present;
- Character category whitelisting: Unicode allows whitelisting categories such as "decimal digits" or "letters" which not only covers the Latin alphabet but also various other scripts used globally (e.g. Arabic, Cyryllic, CJK ideographs etc);
- Individual character whitelisting: If you allow letters and ideographs in names and also want to allow apostrophe ' for Irish names, but don't want to allow the whole punctuation category.

Client Side vs Server Side Validation

Be aware that any JavaScript input validation performed on the client can be bypassed by an attacker that disables JavaScript or uses a Web Proxy. Ensure that any input validation performed on the client is also performed on the server.

Email Validation Basics

Many web applications do not treat email addresses correctly due to common misconceptions about what constitutes a valid address. Specifically, it is completely valid to have an mailbox address which:

- Is case sensitive in the local portion of the address (left of the rightmost @ character);
- Has non-alphanumeric characters in the local-part (including + and @);
- Has zero or more labels.

Following RFC 5321, best practice for validating an email address would be to:

- Check for presence of at least one @ symbol in the address;
- Ensure the local-part is no longer than 64 octets;
- Ensure the domain is no longer than 255 octets;
- Ensure the address is deliverable.

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input_Validation_Cheat_Sheet.md

Session Management

A web session is a sequence of network HTTP request and response transactions associated to the same user. Modern and complex web applications require the retaining of information or status about each user for the duration of multiple requests. Therefore, sessions provide the ability to establish variables - such as access rights and localization settings - which will apply to each and every interaction a user has with the web application for the duration of the session.

Additionally, web applications will make use of sessions once the user has authenticated. This ensures the ability to identify the user on any subsequent requests as well as being able to apply security access controls, authorized access to the user private data, and to increase the usability of the application. Therefore, current web applications can provide session capabilities both pre and post authentication.

Session ID Properties

In order to keep the authenticated state and track the users progress within the web application, applications provide users with a session identifier (session ID or token) that is assigned at session creation time, and is shared and exchanged by the user and the web application for the duration of the session. The session ID is a name=value pair.

With the goal of implementing secure session IDs, the generation of identifiers (IDs or tokens) must meet the following properties:

Session ID Name Fingerprinting
The name used by the session ID should not be extremely descriptive nor offer unnecessary details about the purpose and meaning of the ID. It is recommended to change the default session ID name of the web development framework to a generic name, such as id.

Session ID Length
The session ID must be long enough to prevent brute force attacks, where an attacker can go through the whole range of ID values and verify the existence of valid sessions. The session ID length must be at least 128 bits (16 bytes).

Session ID Entropy
The session ID must be unpredictable (random enough) to prevent guessing attacks, where an attacker is able to guess or predict the ID of a valid session through statistical analysis techniques. For this purpose, a good PRNG (Pseudo Random Number Generator) must be used. The session ID value must provide at least 64 bits of entropy (if a good PRNG is used, this value is estimated to be half the length of the session ID).

Session ID Content (or Value)
The session ID content (or value) must be meaningless to prevent information disclosure attacks, where an attacker is able to decode the contents of the ID and extract details of the user, the session, or the inner workings of the web application.

The session ID must simply be an identifier on the client side, and its value must never include sensitive information. The meaning and business or application logic associated to the session ID must be stored on the server side, and specifically, in session objects or in a session management database or repository.

The stored information can include the client IP address, User-Agent, e-mail, username, user ID, role, privilege level, access rights, language preferences, account ID, current state, last login, session timeouts, and other internal session details. If the session objects and properties contain sensitive information, such as credit card numbers, it is required to duly encrypt and protect the session management repository. It is recommended to create cryptographically strong session IDs through the usage of cryptographic hash functions such as SHA256.

Inactivity Time Out*
Authenticated sessions should timeout after determined period of inactivity - 15 minutes is recommended.

Login & Logout
New session IDs should be created on login (to prevent session fixation via XSS on sibling domains or subdomains). Upon logout the session ID should be invalidated on the server side and deleted on the client via expiration/overwriting the value.

Cookies

The session ID exchange mechanism based on cookies provides multiple security features in the form of cookie attributes that can be used to protect the exchange of the session ID:

Secure Attribute The Secure cookie attribute instructs web browsers to only send the cookie through an encrypted HTTPS (SSL/TLS) connection. This session protection mechanism is mandatory to prevent the disclosure of the session ID through MitM (Man-in-the-Middle) attacks. It ensures that an attacker cannot simply capture the session ID from web browser traffic.

HttpOnly Attribute The HttpOnly cookie attribute instructs web browsers not to allow scripts an ability to access the cookies via the DOM document.cookie object. This session ID protection is mandatory to prevent session ID stealing through XSS attacks.

SameSite Attribute SameSite allows a server define a cookie attribute making it impossible to the browser send this cookie along with cross-site requests. The main goal is mitigate the risk of cross-origin information leakage, and provides some protection against cross-site request forgery attacks.

Domain and Path Attributes The Domain cookie attribute instructs web browsers to only send the cookie to the specified domain and all subdomains. If the attribute is not set, by default the cookie will only be sent to the origin server. The Path cookie attribute instructs web browsers to only send the cookie to the specified directory or subdirectories (or paths or resources) within the web application. If the attribute is not set, by default the cookie will only be sent for the directory (or path) of the resource requested and setting the cookie.

Expire and Max-Age Attributes Typically, session management capabilities to track users after authentication make use of non-persistent cookies. This forces the session to disappear from the client if the current web browser instance is closed. Therefore, it is highly recommended to use non-persistent cookies for session management purposes, so that the session ID does not remain on the web client cache for long periods of time, from where an attacker can obtain it.

- Ensure that sensitive information is not comprised, by ensuring that sensitive information is not persistent / encrypting /stored on a need basis for the duration of the need
- Ensure that unauthorized activities cannot take place via cookie manipulation Ensure secure flag is set to prevent accidental transmission over "the wire" in a non-secure manner
- Determine if all state transitions in the application code properly check for the cookies and enforce their use
- Ensure entire cookie should be encrypted if sensitive data is persisted in the cookie
- Define all cookies being used by the application, their name and why they are needed

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md

Cross Site Scripting (XSS)

Given the way browsers parse HTML, each of the different types of slots has slightly different security rules. When you put untrusted data into these slots, you need to take certain steps to make sure that the data does not break out of that slot into a context that allows code execution.

HTML entity encoding is okay for untrusted data that you put in the body of the HTML document, such as inside a "div" tag. It even sort of works for untrusted data that goes into attributes, particularly if you're religious about using quotes around your attributes. But HTML entity encoding doesn't work if you're putting untrusted data inside a "script" tag anywhere, or an event handler attribute like onmouseover, or inside CSS, or in a URL.

XSS Prevention Rules

- Never Insert Untrusted Data Except in Allowed Locations - The first rule is to deny all;
- HTML Escape Before Inserting Untrusted Data into HTML Element Content;
- Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes;
- JavaScript Escape Before Inserting Untrusted Data into JavaScript Data Values;
- HTML escape JSON values in an HTML context and read the data with JSON.parse;
- Ensure returned Content-Type header is application/json and not text/html;
- CSS Escape And Strictly Validate Before Inserting Untrusted Data into HTML Style Property Values;
- URL Escape Before Inserting Untrusted Data into HTML URL Parameter Values;
- Sanitize HTML Markup with a Library Designed for the Job;
- Prevent DOM-based XSS;

- Use HTTPOnly cookie flag;
- Implement Content Security Policy;
- Use an Auto-Escaping Template System;
- Use the X-XSS-Protection Response Header;
- Properly use modern JS frameworks like Angular (2+) or ReactJS.

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md

Cryptography

An architectural decision must be made to determine the appropriate method to protect data at rest. There are such wide varieties of products, methods and mechanisms for cryptographic storage.The general practices and required minimum key length depending on the scenario listed below:

Good practices

- Cryptographic algorithms are up to date and in-line with industry standards. This includes, but is not limited to outdated block ciphers (e.g. DES), stream ciphers (e.g. RC4), as well as hash functions (e.g. MD5) and broken random number generators like Dual_EC_DRBG (even if they are NIST certified). All of these should be marked as insecure and should not be used and removed from the application and server.
- Key lengths are in-line with industry standards and provide protection for sufficient amount of time. A comparison of different key lengths and protection they provide taking into account Moore's law is available online.
- Cryptographic means are not mixed with each other: e.g. you do not sign with a public key, or try to reuse a keypair used for a signature to do encryption.
- Cryptographic parameters are well defined within reasonable range. This includes, but is not limited to: cryptographic salt, which should be at least the same length as hash function output, reasonable choice of password derivation function and iteration count (e.g. PBKDF2, scrypt or bcrypt), IVs being random and unique, fit-for-purpose block encryption modes (e.g. ECB should not be used, except specific cases), key management being done properly (e.g. 3DES should have three independent keys) and so on.

Recommended Algorithms

- Confidentiality algorithms: AES-GCM-256 or ChaCha20-Poly1305;
- Integrity algorithms: SHA-256, SHA-384, SHA-512, Blake2;
- Digital signature algorithms: RSA (3072 bits and higher), ECDSA with NIST P-384;
- Key establishment algorithms: RSA (3072 bits and higher), DH (3072 bits or higher), ECDH with NIST P-384;
- Application must be capable of using end-to-end encryption via SSL / TLS in relation to sensitive data in transit and at rest.

Additionally, you should always rely on secure hardware (if available) for storing encryption keys, performing cryptographic operations, etc.

Secure Cryptographic Storage Design

- All protocols and algorithms for authentication and secure communication should be well vetted by the cryptographic community.
- Ensure certificates are properly validated against the hostnames users whom they are meant for.
- Avoid using wildcard certificates unless there is a business need for it
- Maintain a cryptographic standard to ensure that the developer community knows about the approved ciphersuits for network security protocols, algorithms, permitted use, cryptoperiods and Key Management.
- Only store sensitive data that you need

Use strong approved Authenticated Encryption

CCM or GCM are approved Authenticated Encryption modes based on AES algorithm.

Use strong approved cryptographic algorithms

- Do not implement an existing cryptographic algorithm on your own, no matter how easy it appears. * Instead, use widely accepted algorithms and widely accepted implementations.
- Only use approved public algorithms such as AES, RSA public key cryptography, and SHA-256 or better for hashing.
- Do not use weak algorithms, such as MD5 or SHA1.
- Avoid hashing for password storage,instead use Argon2, PBKDF2, bcrypt or scrypt.
- See NIST approved algorithms or ISO TR 14742 "Recommendations on Cryptographic Algorithms or Algorithms", key size and parameters by European Union Agency for Network and Information Security.
- If a password is being used to protect keys then the password strength should be sufficient for the strength of the keys it is protecting. * When 3DES is used, ensure K1 != K2 != K3, and the minimum key length must be 192 bits .
- Do not use ECB mode for encrypting lots of data (the other modes are better because they chain the blocks of data together to improve the data security).

Use strong random numbers

- Ensure that all random numbers, especially those used for cryptographic parameters (keys, IV's, MAC tags), random file names, random GUIDs, and random strings are generated in a cryptographically strong fashion.
- Ensure that random algorithms are seeded with sufficient entropy.
- Tools like NIST RNG Test tool can be used to comprehensively assess the quality of a Random Number Generator by reading e.g. 128MB of data from the RNG source and then assessing its randomness properties with the tool.

The following libraries are considered weak random numbers generators and should not be used:

- C library: random(), rand(), use getrandom(2) instead;
- Java library: java.util.Random() instead use java.security.SecureRandom instead.

For secure random number generation, refer to NIST SP 800-90A. CTR-DRBG, HASH-DRBG, HMAC-DRBG are recommended.

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cryptographic_Storage_Cheat_Sheet.md

Access Control

Authorization is the process where requests to access a particular resource should be granted or denied. It should be noted that authorization is not equivalent to authentication - as these terms and their definitions are frequently confused. Authentication is providing and validating identity. Authorization includes the execution rules that determines what functionality and data the user (or Principal) may access, ensuring the proper allocation of access rights after authentication is successful.

Web applications need access controls to allow users (with varying privileges) to use the application. They also need administrators to manage the applications access control rules and the granting of permissions or entitlements to users and other entities.

Role Based Access Control

Access decisions are based on an individual's roles and responsibilities within the organization or user base. An RBAC access control framework should provide web application security administrators with the ability to determine who can perform what actions, when, from where, in what order, and in some cases under what relational circumstances.

Advantages:

- Roles are assigned based on organizational structure with emphasis on the organizational security policy
- Easy to use
- Easy to administer
- Built into most frameworks
- Aligns with security principles like segregation of duties and least privileges

Problems:

- Documentation of the roles and accesses has to be maintained stringently.
- Multi-tenancy can not be implemented effectively unless there is a way to associate the roles with multi-tenancy capability requirements e.g. OU in Active Directory
- There is a tendency for scope creep to happen e.g. more accesses and privileges can be given than intended for. Or a user might be included in two roles if proper access reviews and subsequent revocation is not performed.
- Does not support data based access control

Areas of caution:

- Roles must be only be transferred or delegated using strict sign-offs and procedures.
- When a user changes his role to another one, the administrator must make sure that the earlier access is revoked such that at any given point of time, a user is assigned to only those roles on a need to know basis.
- Assurance for RBAC must be carried out using strict access control reviews.

Discretionary Access Control

Discretionary Access Control is a means of restricting access to information based on the identity of users and/or membership in certain groups. Access decisions are typically based on the authorizations granted to a user based on the credentials he presented at the time of authentication. The owner of information or any resource is able to change its permissions at his discretion.

Advantages:

- Easy to use;
- Easy to administer;
- Aligns to the principle of least privileges;
- Object owner has total control over access granted.

Problems:

- Documentation of the roles and accesses has to be maintained stringently;
- Multi-tenancy can not be implemented effectively unless there is a way to associate the roles with multi-tenancy capability requirements;
- There is a tendency for scope creep to happen e.g. more accesses and privileges can be given than intended for.

Areas of caution: * While granting trusts; * Assurance for DAC must be carried out using strict access control reviews.

Mandatory Access Control

Ensures that the enforcement of organizational security policy does not rely on voluntary web application user compliance. MAC secures information by assigning sensitivity labels on information and comparing this to the level of sensitivity a user is operating at.MAC is usually appropriate for extremely secure systems including multilevel secure military applications or mission critical data applications.

Advantages:

- Access to an object is based on the sensitivity of the object;
- Access based on need to know is strictly adhered to and scope creep has minimal possibility;
- Only an administrator can grant access.

Problems:

- Difficult and expensive to implement;
- Not agile.

Areas of caution:

- Classification and sensitivity assignment at an appropriate and pragmatic level;
- Assurance for MAC must be carried out to ensure that the classification of the objects is at the appropriate level.

Permission Based Access Control

Is the abstraction of application actions into a set of permissions. A permission may be represented simply as a string based name, for example "READ". Access decisions are made by checking if the current user has the permission associated with the requested application action.

The has relationship between the user and permission may be satisfied by creating a direct relationship between the user and permission (called a grant), or an indirect one. In the indirect model the permission grant is to an intermediate entity such as user group.

A user is considered a member of a user group if and only if the user inherits permissions from the user group. Systems that provide fine-grained domain object level access control, permissions may be grouped into classes. The system can be associated with a class which determines the permissions applicable to the respective domain object.

> In such a system a "DOCUMENT" class may be defined with the permissions "READ", "WRITE" and "DELETE"; a "SERVER" class may be defined with the permissions "START", "STOP", and "REBOOT".

File Uploading

Into web applications, when we expect upload of working documents from users, we can expose the application to submission of documents that we can categorize as malicious. We use the term "malicious" here to refer to documents that embed malicious code that will be executed when another user (admin, back office operator...) will open the document with the associated application reader.
Usually, when an application expect his user to upload a document, the application expect to receive a document for which the intended use will be for reading/printing/archiving. The document should not alter is content at opening time and should be in a final rendered state.

The most common file types used to transmit malicious code into file upload feature are the following:

- Microsoft Office document: Word/Excel/Powerpoint
- Adobe PDF document: Insert malicious code as attachment.
- Images: Malicious code embedded into the file or use of binary file with image file extension.

For Word/Excel/Powerpoint/Pdf documents:

- Detect when a document contains "code"/OLE package, if it's the case then block the upload process. For Images document:
- Sanitize incoming image using re-writing approach and then disable/remove any "code" present (this approach also handle case in which the file sent is not an image).

Upload Verification

- Use input validation to ensure the uploaded filename uses an expected extension type;

- Ensure the uploaded file is not larger than a defined maximum file size;

Upload Storage

- Use a new filename to store the file on the OS. Do not use any user controlled text for this filename or for the temporary filename;
- Store all user uploaded files on a separate domain. Archives should be analyzed for malicious content (anti-malware, static analysis, etc).

Public Serving of Uploaded Content

- Ensure the image is served with the correct content-type (e.g. image/jpeg, application/x-xpinstall).

Beware of "special" files

The upload feature should be using a whitelist approach to only allow specific file types and extensions. However, it is important to be aware of the following file types that, if allowed, could result in security vulnerabilities;

"crossdomain.xml" allows cross-domain data loading in Flash, Java and Silverlight. If permitted on sites with authentication this can permit cross-domain data theft and CSRF attacks. Note this can get pretty complicated depending on the specific plugin version in question, so its best to just prohibit files named "crossdomain.xml" or "clientaccesspolicy.xml".

".htaccess" and ".htpasswd" provides server configuration options on a per-directory basis, and should not be permitted.

Logging and Error Handling

Purpose of logging

Application logging should be always be included for security events. Application logs are invaluable data for:

- Identifying security incidents;
- Monitoring policy violations;
- Establishing baselines;
- Assisting non-repudiation controls;
- Providing information about problems and unusual conditions;
- Contributing additional application-specific data for incident investigation which is lacking in other log sources;
- Helping defend against vulnerability identification and exploitation through attack detection.

Each log entry needs to include sufficient information for the intended subsequent monitoring and analysis. It could be full content data, but is more likely to be an extract or just summary properties.

The application logs must record "when, where, who and what" for each event.

Where to record event data

- When using the file system, it is preferable to use a separate partition than those used by the operating system, other application files and user generated content:
- For file-based logs, apply strict permissions concerning which users can access the directories, and the permissions of files within the directories;
- In web applications, the logs should not be exposed in web-accessible locations, and if done so, should have restricted access and be configured with a plain text MIME type (not HTML).
- When using a database, it is preferable to utilize a separate database account that is only used for writing log data and which has very restrictive database , table, function and command permissions;
- Use standard formats over secure protocols to record and send event data, or log files, to other systems e.g. Common Log File System (CLFS) or Common Event Format (CEF) over syslog; standard formats facilitate integration with centralised logging services.

Which events to log

- Input validation failures e.g. protocol violations, unacceptable encodings, invalid parameter names and values;
- Output validation failures e.g. database record set mismatch, invalid data encoding
- Authentication successes and failures;
- Authorization (access control) failures;
- Session management failures e.g. cookie session identification value modification
- Application errors and system events e.g. syntax and runtime errors, connectivity problems, performance issues, third party service error messages, file system errors, file upload virus detection, configuration changes;
- Application and related systems start-ups and shut-downs, and logging initialization (starting, stopping or pausing);

- Use of higher-risk functionality e.g. network connections, addition or deletion of users, changes to privileges, assigning users to tokens, adding or deleting tokens, use of systems administrative privileges, access by application administrators, all actions by users with administrative privileges, access to payment cardholder data, use of data encrypting keys, key changes, creation and deletion of system-level objects, data import and export including screen-based reports, submission of user-generated content - especially file uploads.

Data to exclude

- Application source code;
- Session identification values (consider replacing with a hashed value if needed to track session specific events);
- Access tokens;
- Sensitive personal data and some forms of personally identifiable information (PII) e.g. health, government identifiers, vulnerable people;
- Authentication passwords;
- Database connection strings;
- Encryption keys and other master secrets;
- Bank account or payment card holder data;
- Data of a higher security classification than the logging system is allowed to store;
- Commercially-sensitive information;
- Information it is illegal to collect in the relevant jurisdictions;
- Information a user has opted out of collection, or not consented to e.g. use of do not track, or where consent to collect has expired.

Error Handling

User Facing Error Messages

Error messages displayed to the user should not contain system, diagnostic or debug information.

Formatting Error Messages

Error messages are often logged to text files or files viewed within a web browser.

- Text based log files: Ensure any newline characters (%0A%0C) are appropriately handled to prevent log forging;
- Web based log files: Ensure any logged html characters are appropriately encoded to prevent XSS when viewing logs.

Recommended Error Handling Design

- Log necessary error data to a system log file;
- Display a generic error message to the user;
- If necessary provide an error code to the user which maps to the error data in the log file. A user reporting an error can provide this code to help diagnose issue.

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Logging_Cheat_Sheet.md

Application Regular Updates

Mobile devices and platforms, such as, for example, smartphones, typically provide the capability for operating system (OS), firmware (FW) and applications updates or re-installations with reduced user involvement. The user involvement may often be limited to clicking an icon or accepting an agreement. While this reduced level of involvement may provide convenience and an improved user experience, it fails to address the issue of secure user authentication.

Mobile devices and platforms, such as smartphones, typically provide features for operating system (OS), firmware (FW) upgrades, and applications or reinstallations with reduced user engagement. User engagement may be limited to clicking an icon or accepting a contract. While this reduced level of engagement can provide convenience and enhance the user experience, it does not address the issue of secure user authentication. Thus, it is necessary to create a secure channel that provides confidentiality, integrity, authentication and data updating.

Requirements for a secure software update

Data Confidentiality: the contents of transmitted data should be kept confidential. This also includes software updates. Thus, secure channels between the mobile device and the network manager must be set up. The standard approach to keep sensitive data secret is to encrypt the data with a key that is shared only between the intended receivers;

Data integrity: it must be possible to ensure that data packets have not been modified in transit. For mobile devices, control requests, and software updates it is critically important to verify that the contents in the packets have not been tampered with;

Data Authentication: To prevent an attacker from injecting packets it is important to make sure that the receiver can verify the sender of the packets. Data authentication ensures this property such that the re- ceiver can verify that the received packets really are from the claimed sender. For example, for software updates, data authentication is needed such that the device can verify that the received software comes from a trusted source. Data authentication can be achieved using a MAC or Digital Signature;

Data Freshness: to protect against replay attacks, e.g., during the key establishment phase, the proto- col must ensure that the messages are fresh. Data freshness ensures the security property that the data is recent and that an attacker is not replaying old data.

Third-Party Applications

Many social networks also offer the possibility to create additional applications that extend the functionality of the network. The two major platforms for such applications are the Facebook Platform and Open Social. While applications designed for the Facebook Platform can only be executed in Facebook, Open Social is a combined effort to allow developers to run their applications on any social network that supports the Open Social platform (e.g., MySpace and Orkut).

Requirements for a secure third-party applications

- Data Privacy;
- Data Authentication;
- Data Authorization.

Apps that process or query sensitive information should run in a trusted and secure environment. To create this environment, the app can check the device for the following:

- PIN - or password-protected device locking;
- Recent Mobile Plataform or OS version;
- USB Debugging activation;
- Device encryption;
- Device rooting (see also "Testing Root Detection").

Final Security Mechanisms Report

Mobile Plataform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Channel-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	Local Storage (Centralized Database)
Which DB	
Type of data stored	Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	Java ; HTML5
Input Forms	Yes
Upload Files	Yes
The system has logs	Yes
The system has regular updates	Yes
The system has third-party	Yes
System Cloud Environments	Public Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Bluetooth ; Wi-Fi ; GPS ; NFC
Data Center Phisical Access	Yes

In order to guarantee the confidentiality, availability and privacy of shared data and data freshness, at rest, in use or in transit by legitimate users and communications, as well as the integrity and authenticity of data and communications, developers are recommended of apps for the cloud & mobile platform incorporate *secure backup mechanisms* in the implementation and codification phase of the software development process, as described below.

Req	Mechanism	Mechanism Type	Description	Layer
Integrity, authenticity and privacy, authorization, availability, data freshness	Backup	Local and remote encrypted storage using modern and secure encryption schemes	To incorporate remote authentication mechanisms, that is, access to stored data should only be possible through remote authentication	Data Link

				Using NIDS, NIPS, HIDS, HIPS			Allow to guarantee the Network defense in depth
				To incorporate hybrid authentication mechanisms for accessing applications from the mobile device (e.g., fingerprint and PIN, face recognition and PIN or voice and PIN recognition, iris recognition and PIN or PIN)			Application
				To incorporate access control mechanisms that ensure application data isolation and user session management			Application
				Installing IPS and IDS on mobile devices, in order to guarantee the perimeter security of user data stored locally			Network

In order to guarantee the integrity and availability of user data stored in the cloud and consequently their leakage or loss, it is recommended that developers of mobile applications incorporate *audit mechanisms*, based on the illustration below.

Ref	Mechanism	Mechanism Type	Description	Layer
	Confidentiality, Integrity, availability, auditing, accountability	Reliability, inspection, Identity-based public cloud auditing scheme, analysis mechanisms		Data Link

	An identity-based distributed probable data ownership scheme
	Audit scheme for public cloud storage based on authorized identity with hierarchical structure for large-scale user groups

In order to guarantee the confidentiality and privacy of data shared, at rest or in transit by legitimate users and communications, as well as the integrity, authenticity of data and communications, it is recommended to developers of apps for the cloud & mobile platform to incorporate the algorithms cryptographic and related mechanisms in the implementation and codification phase of the software development process, as described below.

Ref	Mechanism	Mechanism Type	Description	Layer
	Privacy and confidentiality authentication	Cryptographic algorithms Confidentiality, related authorization mechanisms	TCP/TLS, HTTPS, XMPP, AES256-RSA, SSL/TLS, HTTPSCurve25519, AES-256, AES256-RSA2048	Encrypted communications Presentation and Application
		MAC, Digital Signatures	Authentication and Application	Presentation and Application
		AES-GCM-256 or ChaCha20-Poly1305	Confidentiality and Application Algorithms	Presentation and Application
		RSA (3072 bits and higher), ECDSA with NIST P-384	Digital Signature Algorithms	Presentation and Application
	Integrity	SHA-256, SHA-384, SHA-512, Blake2		Presentation and Application
		RSA (3072 bits and higher), DH (3072 bits or higher), ECDH with NIST P-384	Key establishment algorithms	Presentation and Application

In order to ensure that personal data, applications and servers are authentic and that they are only accessed by legitimate or authorized entities, it is recommended to incorporate the authentication and backup mechanisms in the implementation and codification phase of the software development process, as described in the table below.

Requirement	Plataform	Mechanism	Mechanism Type	Description	Layer
Authenticity	Both	Authentication	Biometric-based authentication	Gaze Gesture, Electrocardiogram, Voice recognition, Signature recognition, Gait recognition, Behavior profiling, Fingerprint, Smart card, Multi-touch interfaces, Graphical password, Face recognition, Iris recognition, Rhythm, Capacitive touch-screen, Ear Shape, Arm Gesture, Keystroke Dinamics, Touch dinamics	Application

In order to ensure that personal data, applications and servers are authentic and that they are only accessed by legitimate or authorized entities, it is recommended to incorporate the authentication and backup mechanisms in the implementation and codification phase of the software development process, as described in the table below.

Requirement	Plataform	Mechanism	Mechanism Type	Description	Layer

Authenticity	Both	Authentication	Channel-based authentication	Physical proximity, Electronic voting, Seamless roaming, Transitive authentication, Atribute-based authentication, User-habit-oriented authentication, Handover authentication	Application
	Both	Secure Boot	Digital Signature, checksums, Trusted Plataform Module	Boot verification of hardware, software and firmware integrity	Application

In order to ensure that personal data, applications and servers are authentic and that they are only accessed by legitimate or authorized entities, it is recommended to incorporate the authentication and backup mechanisms in the implementation and codification phase of the software development process, as described in the table below.

Requirement	Plataform	Mechanism	Mechanism Type	Description	Layer
Authenticity	Both	Authentication	Factors-based authentication	Two-factor, Three-factor, Multi-factor	Application
	Both	Secure Boot	Digital Signature or checksums	Boot verification of hardware, software and firmware integrity	Application

In order to ensure that the data shared and exchanged between two or more authorized entities are reliable, complete, authentic and only accessible to these entities, it is recommended that software developers for the mobile ecosystem incorporate *cryptographic protocols* in the implementation and codification phase of the software development process, as described below.

Requirement	Plataform	Mechanism	Mechanism Type	Description	Layer

	Both	Cryptographic Protocols over SCTP/UDP	SSL/TLS, DTLS	Protocols that can be used or implemented over a network to ensure secure data transmission over UDP and SCTP	Application, Presentation, Session
	Both	Wireless Cryptographic Protocols	WEP, WPA, 802.11i (WPA2), EAP, PSK, TKIP, PEAP, EAP-TTLS, EAP-PSK, EAP-SIM, EAP-AKA, AES-CCMP	Security Protocols that must be used or implemented specifically according to the mobile platform or operating system for wireless networks	Transport
	Both	Cryptographic Protocols over IP Protocol	IPSec, PEAP, EAP-TLS	Protocols that ensure data packet encryption and authentication over the IP Protocol	Network and Data Link

In order to ensure that applications and users access only and only the resources allowed, safeguarding the principle of minimum privileges, it is recommended that developers of apps for the cloud & mobile ecosystem incorporate *access control mechanisms* in the coding implementation phase in the software development process, according to the suggestions described below.

Require	Plataform	Mechar	Mechanism Type	Desc	Layer
	Both	Authorization, audit, authenticity, interoperability	Access Control	RBAC, ABAC, ABE	Applic
	Android		DR BACA, CA-ARBAC, RBACA		

To ensure a permanent or almost permanent observation of the system, in order to detect any unexpected activity or detect abuses by privileged users, app developers for the cloud & mobile ecosystem are recommended to incorporate inspection mechanisms in the implementation and coding phase in the software development process, as described below.

Requirement	Plataform	Mechanism	Mechanism Type	Description	Layer
Privacy, authorization, immunity, Tampering Detection		Inspection	IDS, IPS, NIDS, NIPS, HIDS, HIPS, IDPS, DIDS, VMM based IDS		Network

In order to ensure non-repudiation, audit and accountability by all legitimate or illegitimate entities in the cloud & mobile ecosystem, it is recommended that mobile app developers incorporate *logging mechanisms* during the implementation and coding in the software development process, as described below.

Requirement	Mechanism	Mechanism Type	Description	Layer
Non repudiation, audit, accountability	System, Boot, Logging, System log files or event log		It is recommended that developers during the coding phase, use the native APIs of each Data Link of the mobile device platform that allow incorporation of Logging into application during the software development process	

			All mechanisms related to storage or secure backup apply		

In order to ensure that the application and confidential data of legitimate users are not accessed by third parties from the device or remotely from the data center, it is recommended that users incorporate *tampering detention mechanisms* on the device, as illustrated below.

Requirement	Mechanism	Mechanism Type	Description	Layer
Authorization, authenticity, privacy, immunity	Device authentication, Application authentication, Detection	Incorporation of hybrid authentication schemes to the application		Application
		Incorporation of access control and session management mechanisms that guarantee the sending of notifications whenever there is new access from a new device or browser		Session

In order to ensure that user data stored in remote databases is safe and reliable, app developers for the cloud & mobile ecosystem are recommended to incorporate data *location physical mechanisms* for data centers.

Requirement	Plataform	Mechanism	Mechanism Type	Description	Layer
Physical security	Both	Physical security location	Smartcards, mobile surveillance cameras with 360 degree night vision, motion sensors and detectors, facial recognition identification cameras, etc.		Physical

In order to ensure that applications are resilient to an eventual attack and that they do not violate the principle of minimum requirements when sharing resources locally or remotely, app developers for the cloud & mobile ecosystem are recommended to incorporate *confinement mechanisms*, as well as those of access

control or secure permissions.

Requirement	Plataform	Mechanism	Mechanism Type	Desc	Layer
Privacy, integrity, authenticity, immunity	Both	Containerizing	Sandboxing, TPM, MTM, TEE		Its purpose is to guarant the privacy, integrity and authent of the applic data of the end users and the integrity of the system
	Both		Firewall		
	Both		DMZ		
	iOS		Unix Permissions		
	iOS		iOS Capabilities		
	iOS		Hard-Coded Checks		

In order to ensure that legitimate or illegitimate users or machines do not access users' confidential data or potentially unsafe resources or harmful content to sensitive users or children, app developers for the cloud & mobile ecosystem are recommended to incorporate filtering mechanisms , such as those listed below.

Requirement	Plataform	Mechanism	Mechanism Ty	Description	Layer

Integrity, authenticity, access Control, Privacy	Both	Filtering	Firewall and Cryptographic Techniques		Network

Final Attack Models Report

Mobile Plataform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Channel-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	Local Storage (Centralized Database)
Which DB	
Type of data stored	Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	Java ; HTML5
Input Forms	Yes
Upload Files	Yes
The system has logs	Yes
The system has regular updates	Yes
The system has third-party	Yes
System Cloud Environments	Public Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Bluetooth ; Wi-Fi ; GPS ; NFC
Data Center Phisical Access	Yes

Man-in-the-Middle Attack

In this type of attack an active man listen and change communications between Mobile Device and Cloud. In other hand, in this attack an intruder enters in the ongoing conversation between sender and the receiver and makes them believe that conversation is taking place between them only.

Definition

This type of attack occurs whenever an attacker intends to intercept communications in order to interpret or alter the original data in transit between the sender and the receiver establishing a conversation.

Technical Impact

- An attacker is able to decrypt and read all SSL/TLS traffic between the client and server;
- Gain Privileges or Assume Identity.

Risk Analysis

- Critical Risk.

Likelihood of Exploit

- Medium.

Attacker Powers

The attacker generally and depending on whether the communication situation is encrypted or not, is able to modify the cryptographically unprotected communication or modify the cryptographically protected communication. More specifically, it will have the following powers:

- Steal encryption key;
- Discover cryptographic key using cryptanalysis;
- Exploit vulnerabilities in cryptographic algorithm;
- Exploit vulnerabilities in cryptographic protocol.

Recommendations

To ensure that the mobile application is resilient or immune to malicious MitM attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity, privacy and authenticity of the data.

Reference

1. [https://cwe.mitre.org/data/definitions/300.html];

2. [https://www.first.org/cvss/v3.1/examples].

Man-in-the-Middle Attack Diagram

Brute Force Attacks

This type of attack consists in trying to access a system using some mechanism or simply using trial-and-error, aiming to guess the password of a legitimate user of that system. The success of this attack depends largely on the cryptographic scheme used for authentication and access control to the system, as well as the nature of the password set by the legitimate user.

Description

In this attack, some asset, namely, information, functionality, identity, etc., is protected by a finite secret value. The attacker attempts to gain access to this asset by using trial-and-error to exhaustively explore all the possible secret values in the hope of finding the secret (or a value that is functionally equivalent) that will unlock the asset. Examples of secrets can include, but are not limited to, passwords, encryption keys, database lookup keys, and initial values to one-way functions. The key factor in this attack is the attackers' ability to explore the possible secret space rapidly. This, in turn, is a function of the size of the secret space and the computational power the attacker is able to bring to bear on the problem. If the attacker has modest resources and the secret space is large, the challenge facing the attacker is intractable. Assuming a finite secret space, a brute force attack will eventually succeed. The defender must rely on making sure that the time and resources necessary to do so will exceed the value of the information.

This type of attack can be carried out in two different ways: 1. Encryption Brute Forcing; 2. Password Brute Forcing.

Technical Impact

- Read Data:
- Gain Privileges.

Likelihood Of Attack

- Medium

Typical Severity

- High

Risk Analysis

- Critical

Likelihood of Exploit

- High

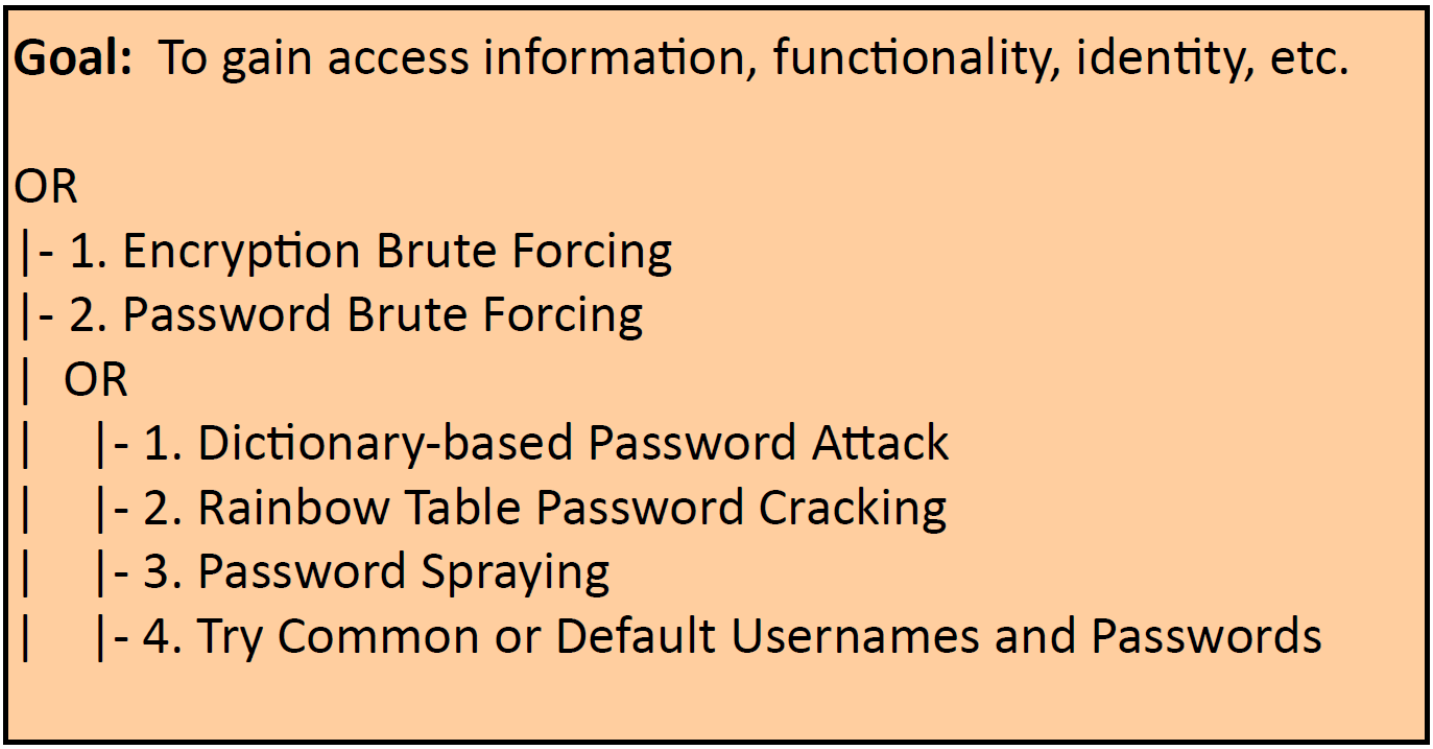
Recommendations

In order to mitigate the Brute Force type attacks it is convenient to follow the good practice guidelines, aiming at incorporating the security mechanisms during the coding and implementation phase and carrying out the security tests suggested and present in the report during the verification phase, with the purpose of ensuring that the functional requirements linked to security and the non-functional requirements of the application to be developed or deployed are met.

References

1. [https://capec.mitre.org/data/definitions/112.html];
2. [https://cwe.mitre.org/data/definitions/521.html]

Brute Force Attack Tree Diagram



<<<<<<< HEAD

Eavesdropping Attacks

Eavesdropping is a type of attack where the attacker tries to gain access to sensitive information of legitimate users from the messages (text, voice and video) exchanged between two or more users of Instant Messaging (IM) applications. The same applies to recorded calls, call logs and multimedia stored in clear text in memory cards.

Description

An adversary intercepts a form of communication (e.g. text, audio, video) by way of software (e.g., microphone and audio recording application), hardware (e.g., recording equipment), or physical means (e.g., physical proximity). The goal of eavesdropping is typically to gain unauthorized access to sensitive information about the target for financial, personal, political, or other gains. It entails listening in on the raw audio source of a conversation between two or more parties. This type of attack can be carried out in two different ways: 1. Shoulder Surfing (Physical Eavesdropping); 2. Probe Audio and Video Peripheralsn (Software Eavesdropping).

Technical Impact

- Read Data

Likelihood Of Attack

- High

Typical Severity

- High

Risk Analysis

- High

Likelihood of Exploit

- Medium

Recommendations

In order to mitigate the espionage type attacks it is convenient to follow the good practice guidelines, aiming at incorporating the security mechanisms during the coding and implementation phase and carrying out the security tests suggested and present in the report during the verification phase, with the purpose of ensuring that the functional requirements linked to security and the non-functional requirements of the application to be developed or deployed are met.

References

1. [https://capec.mitre.org/data/definitions/651.html];
2. [https://cwe.mitre.org/data/definitions/200.html];
3. [https://www.first.org/cvss/calculator/3.1#CVSS:3.1/].

Eavesdropping Attack Tree Diagram

=======

Eavesdropping Attacks

Eavesdropping is a type of attack where the attacker tries to gain access to sensitive information of legitimate users from the messages (text, voice and video) exchanged between two or more users of Instant Messaging (IM) applications. The same applies to recorded calls, call logs and multimedia stored in clear text in memory cards.

Description

An adversary intercepts a form of communication (e.g. text, audio, video) by way of software (e.g., microphone and audio recording application), hardware (e.g., recording equipment), or physical means (e.g., physical proximity). The goal of eavesdropping is typically to gain unauthorized access to sensitive information about the target for financial, personal, political, or other gains. It entails listening in on the raw audio source of a conversation between two or more parties. This type of attack can be carried out in two different ways: 1. Shoulder Surfing (Physical Eavesdropping); 2. Probe Audio and Video Peripheralsn (Software Eavesdropping).

Technical Impact

- Read Data

Likelihood Of Attack

- High

Typical Severity

- High

Risk Analysis

- High

Likelihood of Exploit

- Medium

Recommendations

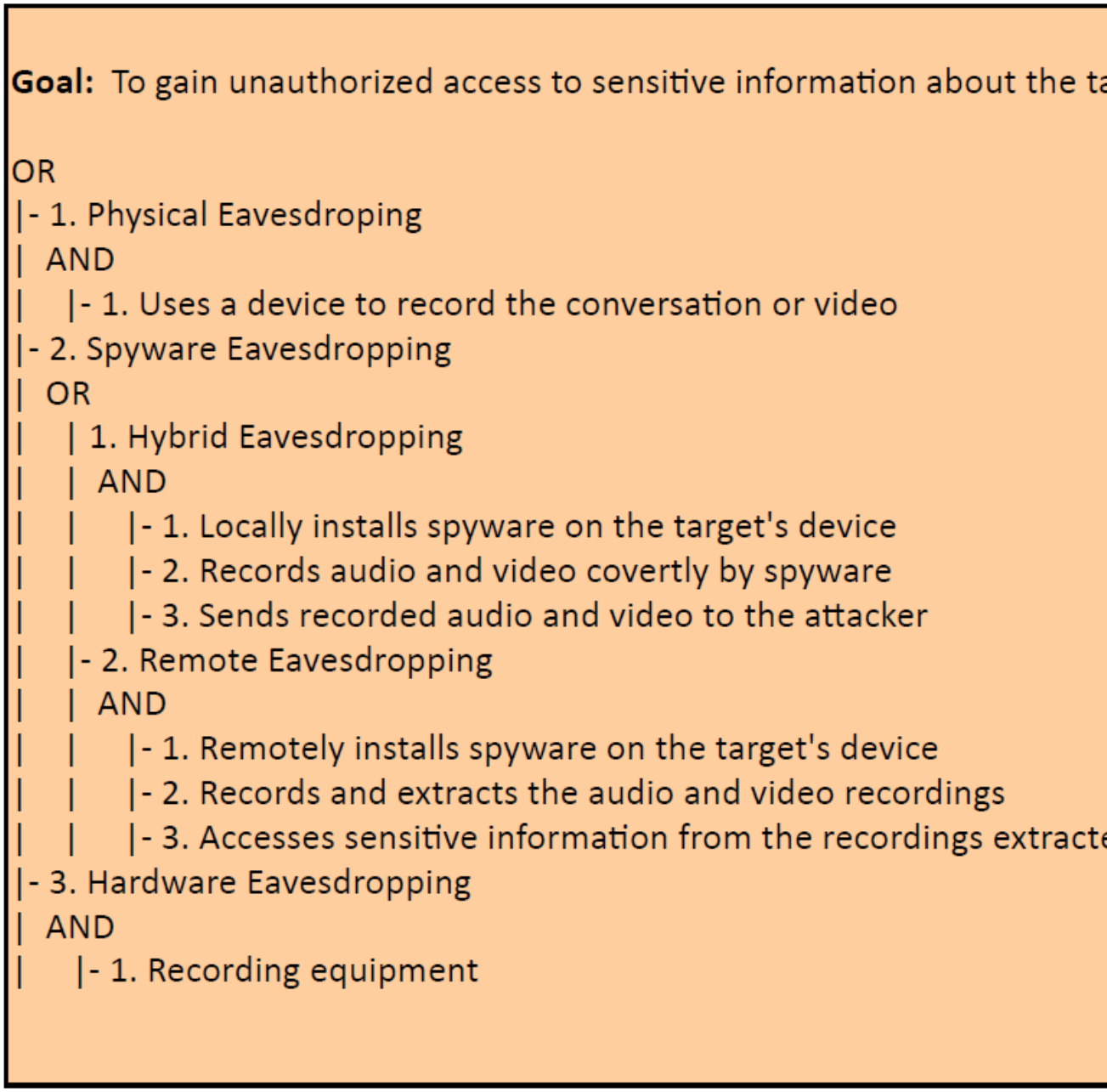
In order to mitigate the espionage type attacks it is convenient to follow the good practice guidelines, aiming at incorporating the security mechanisms during the coding and implementation phase and carrying out the security tests suggested and present in the report during the verification phase, with the purpose of ensuring that the functional requirements linked to security and the non-functional requirements of the application to be developed or deployed are met.

References

1. [https://capec.mitre.org/data/definitions/651.html];
2. [https://cwe.mitre.org/data/definitions/200.html];
3. [https://www.first.org/cvss/calculator/3.1#CVSS:3.1/].

Eavesdropping Attack Tree Diagram

756a6f658e94ba3c11c4cd25afbb448e80592558



Cross Site Scripting Attacks

In short, Cross Site Scripting (XSS) allows an attacker to execute a browser script bypassing access control mechanisms such as the same origin policy. During this attack a malicious script is injected into web content and user considering it to be authentic executes it over its own machine, thus giving either control of the machine or exposure of confidential information to the attacker.

Definition

Being an attack that exploits vulnerabilities in web applications, the attacker in this type of attack executes malicious database claims, exploiting improper validation of data flowing from the user to the database. The attacker's goal is to access the intended party's confidential data by inserting malicious code into the user's web page in order to redirect them to their site. There are two ways to forge this type of attack:

- Stored XSS (uninterruptedly stores malicious code in a resource managed by the web application);
- Reflective XSS (promptly reflects malicious code against the user and therefore does not store it permanently;
- XSS based on DOM (Document Object Model).

Technical Impact

- Gain Privileges or Assume Identity;
- Bypass Protection Mechanism;
- Read Application Data;
- Modify Application Data;

- DoS: Crash, Exit, or Restart.

Risk Analysis

- Critical Risk.

Likelihood of Exploit

- Medium.

Attacker Powers

- Circumvent the policy of same origin;
- Impersonate you to websites and/or web applications you regularly use by obtaining/altering/destroying various types of content.

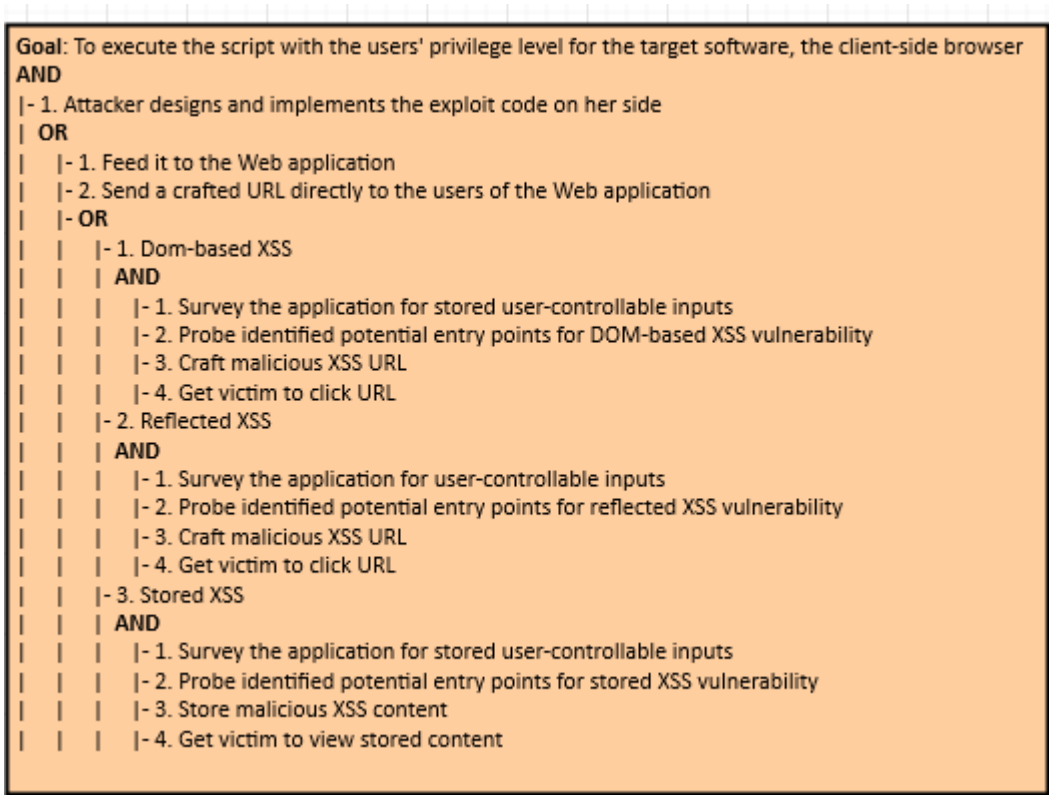
Recommendations

To ensure that the mobile application is resilient or immune to XSS attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity, privacy and authenticity of the data.

References

1. [https://cwe.mitre.org/data/definitions/352.html];
2. [https://www.first.org/cvss/v3.1/examples]

Cross Site Scripting Attacks Diagram



Cross Site Request Forgery Attacks

O Cross Site Request Forgery (CSRF) é um ataque que força um utilizador final a executar acções indesejadas numa aplicação na qual está autenticado naquele momento.

Definition

Este tipo de ataque tem como finalidade a mudança de estado e não o roubo de dados, dado que o invasor fica impedido de ver a resposta à solicitação falsificada. A condição necessária para que este tipo de ataque tenha sucesso é a existência da permissão de alterações através de solicitações GET.

Technical Impact

- Bypass Protection Mechanism;
- Gain Privileges;
- DoS: Crash, Exit, or Restart;
- Read and Modify Data.

Risk Analysis

- High.

Likelihood of Exploit

- High.

Attacker's Powers

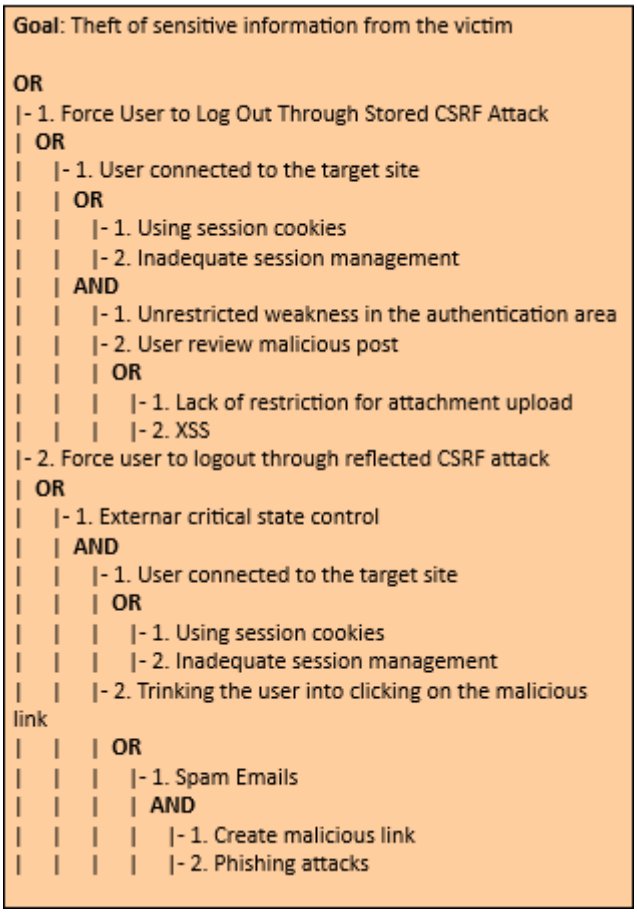
- Furtar valores monetários de forma simulada;
- Realização de outros tipos de ataques;
- Acesso a dados confidenciais (histórico da vítima) ou criticos (número de cartão de crédito) do utilizador.

Recommendations

In order to ensure that the mobile application is resilient or immune to the CSRF attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed.

References

1. [https://capec.mitre.org/data/definitions/62.html];
2. [https://cwe.mitre.org/data/definitions/352.html]



Cache Poisoning Attacks

In this type of attack the attacker uses DNS to convert the domain name to an IP address for the purpose of accessing the user's confidential data. On the other hand, sender and a receiver get rerouted through some evil connection.

Definition

Cache poisoning is the act of introducing false information into a Domain Name System (DNS) cache in order to cause DNS queries to return an incorrect response and, e.g., redirect users to malicious websites. This type of attack can target the cache of an application (e.g., a web browser cache) or a public cache (e.g. a DNS or Address Resolution Protocol (ARP) cache), exposing the application to a variety of attacks, such as redirection to malicious websites and malware injection.

Technical Impact

- Gain Privileges or Assume Identity;
- Bypass Protection Mechanism.

Risk

- Medium.

Likelihood of Exploit

- Low.

Attacker Powers

- Access confidential information from legitimate/authorized users;
- Perpetrate other types of attacks like DDoS and Main-in-the-Middle.

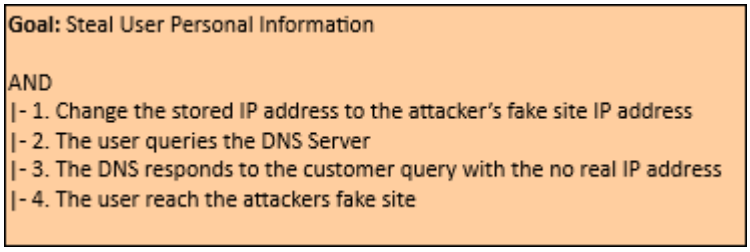
Recommendations

In order to ensure that the mobile application is resilient or immune to the DNS attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed.

Reference

1. [https://cwe.mitre.org/data/definitions/350.html];
2. [https://capec.mitre.org/data/definitions/141.html].

Cache Poisoning Attacks Diagram



Malicious QR Code Attacks

In this type of attack, one of the strategies used by the attackers, after coding the malicious links, is to take them to phishing sites or execute fraudulent codes. In addition, in order to end this type of attack, the attackers often print the malicious QR codes on small stickers that are pasted on pre-existing QR codes. On the other hand, attackers often change selected modules from white to black and vice versa in order to replace the original encoded content.

Definition

QR code-based attack is defined as an attack that attempts to lure victims into scanning a QR code that directs them to malicious websites. The key idea behind QR code attacks is that victims might trust the web page or the printed material on which the QR code is displayed, and assume that the associated code is harmless. In addition, attackers use malicious QR codes to direct users to fraudulent web sites, which masquerade as legitimate web sites aiming to steal sensitive personal information such as usernames, passwords or credit card information.

Technical Impact

- Execute Unauthorized Code or Commands.

Risk Analysis

- High Risk.

Likelihood Exploits

- Low.

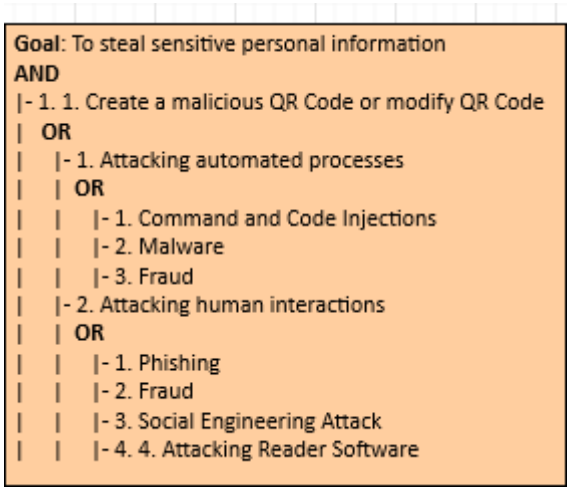
Attacker Powers

- Direct the user to an exploit or phishing site;
- Perform other attacks such as phishing, farming and botnet; * Distribute malware; * Extraction of personal and confidential data from smartphones and tablets via command injection or traditional buffer overflows by reader software;
- Steal users' Money via fraud;
- Social Engineering attacks via spear phishing e.g. leaving a poster of a QR Code on the parking lot of a company (instead of the traditional attack with an USB drive) offering discount in a nearby restaurant is a new attack vector which is likely to be successful.

Recommendations

To ensure that the mobile application is resilient or immune to malicious QR Code attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity and authenticity of the data.

Malicious QR Code Attacks Diagram



CAPTCHA Breaking Attacks

CAPTCHAs were developed in order to prevent the usage of internet resources by bots or computers. They are used to prevent spam and overexploitation of network resources by bots. But recently, it has been found that the spammers (attackers) are able to break the CAPTCHA. In this case, we will be in the presence of an attack of this nature, Captcha Breaking.

Definition

In this type of attacks, the attacker can break the CAPTCHAs by using an audio system, can read the CAPTCHAs by using speech to text conversion software and can also break image-based scheme and video-based scheme.

Technical Impact

- Bypass Protection Mechanism;
- Alter Execution Logic.

Risk Analysis

- High Risk.

Likelihood of Exploit

- Low.

Attacker Powers

- Spamming;
- Conducting DoS and DDoS attacks;

- Excessive exploitation of network resources by bots.

Recommendations

In order to ensure that the mobile application is resilient or immune to the CAPTCHA Breaking attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed.

References

1. [https://cwe.mitre.org/data/definitions/804.html];
2. [https://www.first.org/cvss/calculator/3.1#CVSS:3.1/].

CAPTCHA Breaking Attacks Diagram

Flooding or Distributed Denial of Services (DDoS) Attacks

Flooding is an enhanced Denial of Service (DoS) attack type, originating from multiple network attack surfaces that were previously compromised to disrupt the services or resources provided by the target server. It differs from DoS in that it generates more traffic, so that the targeted server cannot handle requests. This type of attack generally exposes a weakness in rate limiting or flow.

Definition

The Flooding attack attempts to make a service unavailable to intended users by draining the system or network resource. Attackers can now launch various DDoS attacks, including resource-focused attacks (eg, network bandwidth, memory, and CPU) and app-focused attacks (e.g., mobile applications, database service) from almost every attack places. This type of attack can be executed as follows:

- TCP Flood;
- UDP Flood;
- ICMP Flood;
- HTTP Flood;
- SSL Flood;
- Amplification;
- ML Flood;
- BlueSmacking.

Technical Impact

- Crash, Exit, or Restart;
- Bypass protection mechanism;
- Other.

Typical Severity

- Medium.

Risk

- High.

Likelihood of Exploit

- High.

Attacker's Powers

- Make features and services unavailable to authorized users;
- Perpetrate other types of attacks and even extract sensitive and critical data.

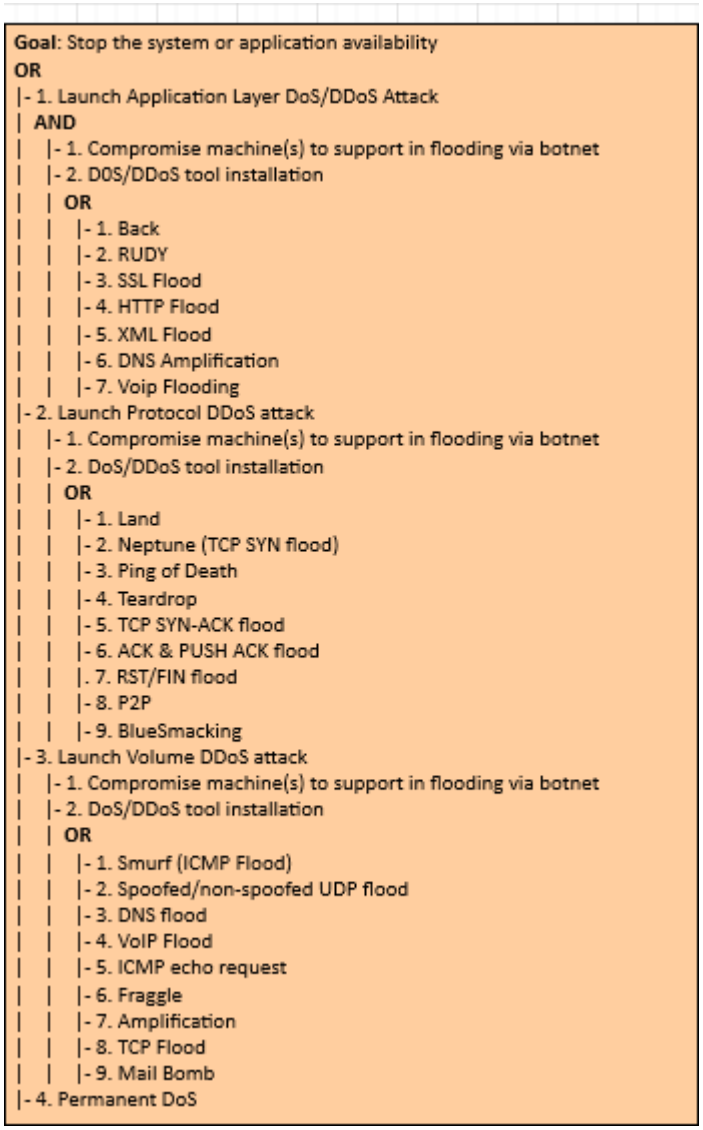
Recommendations

In order to ensure that the mobile application is resilient or immune to the Flooding attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed.

References

1. [[https://cwe.mitre.org/data/definitions/400.html]

Flooding Attacks Diagram



Eavesdropping or Sniffing

This type of attack is carried out by attackers who use applications that can capture data packets in transit over a network, and if they are not heavily encrypted, can be read or interpreted. The goal of the attacker is to spy on all kinds of conversations and recordings and to listen to communication channels.

Definition

This type of attack consists of implant eavesdropping tools in specific network for spying on communication channels, capturing the network traffic behavior and getting the network map. Eavesdropping is dangerous threat that leads to break down the integrity and confidentiality which causes financial and personal failures. There are several ways to get a sniffing attack on a smartphone, as there is a vulnerability in GSM's encryption function for call and SMS privacy, A5 / 1 (it can be stopped second). This vulnerability puts all GSM subscribers at risk of sniffing attacks.

Technical Impact

- Read Application Data;
- Modify Files or Directories.

Risk Analysis

- Critical Risk.

Likelihood of Exploit

- High.

Attacker Powers

- Tracking, capture and theft of confidential information.

References

1. [https://cwe.mitre.org/data/definitions/319.html].

Sniffing Attacks Diagram

```
Goal: To Intercept information transmitted, observing, reading, and/or hearing the communication traffic between two third parties

OR
|- 1. Sniffing Network Traffic
|  AND
|    |- 1. Obtain a network sniffer tool such as Wireshark
|    |- 2. Sniff communication using Wireshark tool
|- 2. Accessing/Intercepting/Modifying HTTP Cookies
|  AND
|    |- 1. Obtain copy of cookie
|    |  OR
|    |    |- 1. Sniff cookie using a network sniffer such as Wireshark
|    |    |- 2. Obtain cookie using a utility
|    |    |- 3. Steal cookie via a cross-site scripting attack
|    |    |- 4. Guess cookie contents if it contains predictable information
|    |- 2. Obtain sensitive information from cookie
|    | -3. Modify cookie to subvert security controls
|    |  OR
|    |    |- 1. Modify logical parts of cookie and send it back to server to observe the effects
|    |    |- 2. Modify numeric parts of cookie arithmetically and send it back to server to observe the effects
|    |    |- 3. Modify cookie bitwise and send it back to server to observe the effects
|    |    |- 4. Replace cookie with an older legitimate cookie and send it back to server to observe the effects
|- 3. Utilizing REST's Trust in the System Resource to Obtain Sensitive Data
|  AND
|    |- 1. Find a REST-style application that uses SSL
|    |- 2. Insert a listener to sniff client-server communication
|    |- 3. Gather information passed in the clear
|- 4. Cellular Traffic Intercept
|- 5. Sniff Application Code
|  AND
|    |- 1. Set up a sniffer
|    |- 2. Capturing Application Code Bound During Patching
```

Phishing Attack

In phishing attack, an adversary sets up a fake URL identical to real Web application fooling the users to enter a valid credentials and certificates.

Definition

Phishing is the attempt to acquire sensitive information or to make somebody act in a desired way by masquerading as a trustworthy entity in an electronic communication medium. They are usually targeted at large groups of people. Phishing attacks can be performed over almost any channel, from physical presence of the attacker to websites, social networks or even cloud services. On the other hand, phishing attacks are typically fraudulent email messages which directs to spoofed website. In PaaS cloud environment, these attacks affect both enterprise and users. This is a type of social engineering attack. These attackers convince the customers to reveal their most important data like password or other sensitive information by using bogus web pages, emails, or bloggers.

Attacker Powers

- Access confidential information from legitimate users by collecting data through malware; * Perpetrate other types of attacks like Botnet.

Recommendations

To ensure that the mobile application is resilient or immune to malicious Phishing attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity, privacy and authenticity of the data.

Phishing Attack Diagram

Goal: Extract and steal user confidential information

AND
|- 1. Create malicious scripts (email, website, etc.)
|- 2. Send or inject malicious scripts
| **OR**
| |- 1. Phishing
| | **OR**
| | |- 1. Spear Phishing
| | |- 2. Whaling
| | |- 3. Vishing
| | |- 4. Smishing
| |- 2. Dumpster Diving
| |- 3. Shoulder Surging
| |- 4. RSE
| |- 5. Water Holing
| |- 6. APT
| |- 7. Baiting

Botnet Attacks

In a nutshell, in a botnet attack scenario the attacker hijacks a set of mobile devices, creating a network of remote controlled zombie devices. This network is called Botnet, from which various types of attacks can be carried out, such as denial of service attacks, malware distribution, phishing, etc.

Definition

A botnet is a set of compromised mobile devices. A necessary condition for these devices to be compromised is their infection by malware. This allows attackers/hackers to remotely control this botnet and launch other types of attacks, such as DoS, Phishing, malware injection, etc.

Technical Impact

- Gain priviliegies or assume identity.

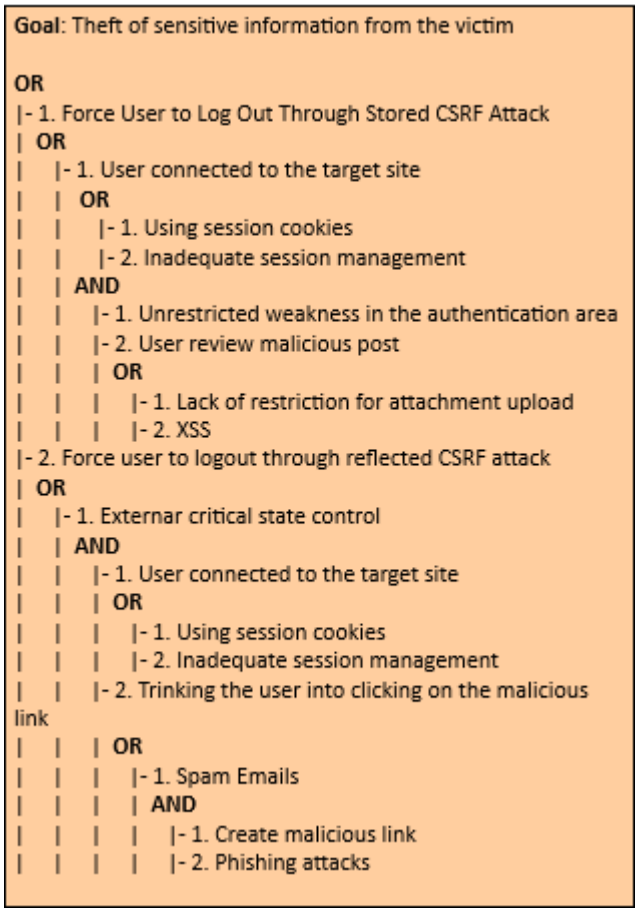
Risk Analysis

- Critical.

Attacker's Powers

- Sending spam;
- Perform attacks like DoS;
- Collecting information that can be used for illegal purposes;

Botnet Attacks Diagram



XML Injection Attacks

It is an attacking technique used against XML-based applications to modify or compromise their normal operation.

Definition

XML Injection (XMLi) attacks are carried out by injecting pieces of XML code along with malicious content into user inputs in order to produce harmful XML messages. The aim of this type of attacks is to compromise the system or system component that receives user inputs, making it malfunction (e.g. crash), or to attack other systems or subsequent components that process those injected XML messages. This type of attack can be classified into 4 categories:

- Deforming: Attack input values of Type 1 are XML meta-characters, such as <, >,]] >, that are intro- duced to compromise the structure of generated XML messages;
- Random closing tags: Attack input values of Type 2 are random XML closing tags (e.g., < /test>), aiming at deforming the generated XML messages to reveal their structure;
- Replicating: Attack input values of Type 3 are strings of characters consisting of XML tag names and malicious content;
- Replacing: Attack input values of Type 4 are similar to those of Type 3 but they involve multiple input fields in order to comment out some existing XML elements and inject new ones with malicious content.

Attacker Powers

- Obtain confidential information;
- Change the underlying business logic of the destination.

Recommendations

To ensure that the mobile application is resilient or immune to Spoofing attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity, privacy and authenticity of the data.

XML Injection Attacks Diagram

Goal: Disclosure or modification of the internal data

AND
|- 1. Create malicious content
|- 2. Inject pieces of XML code along with malicious content into user inputs
|- **OR**
|- |- 1. Deforming
|- |- 2. Random closing tags
|- |- 3. Replicating
|- |- 4. Replacing

Session Hijacking Attack

An attacker impersonates a legitimate user through stealing or predicting a valid session ID.

Definition

The necessary condition for the session hijacking attack to occur is the existence of architectural vulnerabilities in the absence of protection for the storage of session identifiers. This vulnerability generally occurs in web applications written in PHP in previous versions (e.g., PHP 4.0 to PHP 4.1.2), As described in CVE-2002-0121.

Technical Impact

- Read Application Data;
- Gain Privileges or Assume Identity;
- Execute Unauthorized Code or Commands.

Risk Analysis

- Critical.

Likelihood of Exploit

- High.

Attacker Powers

- Steal Session ID;
- Impersonation of a legitimate user and confidential information from a legitimate user.

References

1. [https://www.cvedetails.com/cve/CVE-2002-0121/];
2. [https://cwe.mitre.org/data/definitions/287.html];
3. [https://capec.mitre.org/data/definitions/593.html].

Session Hijacking Attack Diagram

Goal: To gain unathorized access to the application

OR
|- 1. Reusing Session IDs (aka Session Replay)
| **AND**
| |- 1. The attacker interacts with the target host and finds authenticate users session IDs
| |- 2. The attacker steals a session ID from a valid user
| |- 3. The attacker tries to use the stolen session ID to gain access to the system
|- 2. Session Fixation
| **AND**
| |- 1. Setup the Attack (setup a session)
| | **OR**
| | |- 1. The attacker chooses a predefined identifier that they know
| | |- 2. The attacker creates a trap session for the victim
| |- 2. Attract a victim by fixate the session
| | **OR**
| | |- 1. Attackers can put links on web sites
| | |- 2. Attacker establish rogue proxy servers that give out the session ID and then redirect the connection to the legitimate service
| | |- 3. Attackers can email attack URLs to potential victims through spam and phishing techniques
| |- 3. Abuse the victim's Session via takeover the fixated session
| | **OR**
| | |- 1. The attacker loads the predefined session ID into their browser and browses to protected data or functionality.
| | |- 2. The attacker loads the predefined session ID into their software and utilizes functionality with the rights of the victim.
|- 3. Session Sidejacking
| **AND**
| |- 1. The attacker uses sniffing tools to capture a session token from traffic
| |- 2. The attacker attempts to insert a captured session token into communication with the targeted application to confirm viability for exploitation
| |- 3. The attacker leverages the captured session token to interact with the targeted application in a malicious fashion, impersonating the victim
|- 4. Cross Site Tracing
| **AND**
| |- 1. Determine if HTTP Trace is enabled at the web server with which the victim has an active session
| |- 2. Identify mechanism to launch HTTP Trace reques
| |- 3. Create a malicious script that pings the web server with HTTP TRACE request
| |- 4. Execute malicious HTTP Trace launching script
| |- 5. Intercept HTTP TRACE response

Spoofing Attacks

In a nutshell, spoofing attacks consist of spoofing the caller ID in order to impersonate a trusted entity and thus obtain confidential information in a disguised manner.

Definition

In this type of attack, the attacker can spoof the "Caller ID" and impersonate him as a legitimate user, i.e., an attacker could spoof the "Caller ID" and impersonate a trusted party. Recent studies have also shown how to spoof MMS messages that appeared to be messages from a number that operators use to send alerts or update notifications. In addition, base stations can also be counterfeited. On the other hand, there is also the mobile application spoofing attack, which consists of an attack where a malicious mobile application mimics the visual appearance of another one. The goal of the adversary is to trick the user into believing that she is interacting with a genuine application while she interacts with one controlled by the adversary. If such an attack is successful, the integrity of what the user sees as well as the confidentiality of what she inputs into the system can be violated by the adversary.

Technical Impact

- Bypass Protection Mechanism;
- Gain Privileges or Assume Identity.

Risk Analysis

- Critical Risk.

Likelihood of Exploit

- High.

Attacker Powers

- Faker caller ID;
- Monitoring of calls and access to the confidential information of legitimate users from voice or text messages.

Recommendations

To ensure that the mobile application is resilient or immune to Spoofing attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity, privacy and authenticity of the data.

References

1. [https://cwe.mitre.org/data/definitions/290.html].

Spoofing Attacks Diagram

Goal: Accessing resources or obtain banking and other critical information

OR
|- 1. Content Spoofing
| **OR**
| |- 1. Checksum Spoofing
| |- 2. Spoofing ofUDDI/ebXML Messages
| |- 3. Counterfeit GPS Signals
|- 2. Action Spoofing
| **OR**
| |- 1. Clickjacking
| | **OR**
| | |- 1. Flash File Overlay
| | |- 2. iFrame Overlay
| |- 2. Android Activity Hijack
| |- 3. Credential Prompt Impersonation
| |- 4. Tapjacking
|- 3. Resource Location Spoofing
| **OR**
| |- 1. Redirect Access to Libraries
| | **OR**
| | |- 1. Symlink Attack
| | |- 2. Leveraging/Manipulating Configuration File Search Paths
| | |- 3. Search Order Hijacking
| | |- 4. DLL Side-Loading
| |- 2. Establish Rogue Location
| | **OR**
| | |- 1. Evil Twin Wi-Fi Attack
| | |- 2. Cellular Rogue Base Station
| | |- 3. Scheme Squatting
| | |- 4. BitSquatting
| | |- 5. TypoSquatting
| | |- 6. SoundSquatting
| | |- 7. Homograph Attackvia Homoglyphs
| | |- 8. Bluetooth Impersonation AttackS
|- 4. Identity Spoofing
| **OR**
| |- 1. Pharming
| |- 2. Phishing
| | **OR**
| | |- 1. Spear Phishing
| | |- 2. Mobile Phishing
| | |- 3. Voice Phishing
| |- 3. Fake the Source of Data
| |- 4. Principal Spoof
| |- 5. Signature Spoof

VM Migration Attacks

A malicious user can start or redirect the migration process to a different network in which he has access or untrusted host, or it can just be copied and used elsewhere, which compromise the VM with the passwords, credentials on it and in case of coping it makes it difficult to trace the attacker.

Definition

VMs roll back to their previous state if an error occurs. Unfortunately, this factor can re-expose them to security vulnerabilities, and attackers can gain benefit to attack on this compromised hypervisor. It is important to protect the data during migration. In fact, this is the defending of data privacy and integrity from various network attacks during migration. Live migration might be susceptible to many attacks like ”man-in-the-middle”, ”denial-of-service” and ”replay. The data during

the migration can be sniffed or tampered easily as it is not encrypted.

Technical Impact

- Read Application Data (lack of confidentiality);
- Modify Application Data (lack of integrity and confidentiality).

Risk Analisys

- High Risk.

Likelihood of Exploit

- High.

Attacker Powers

- Launch attacks such as man-in-the-middle, DoS and replay;
- Detect or tamper with data during migration as it is not encrypted.

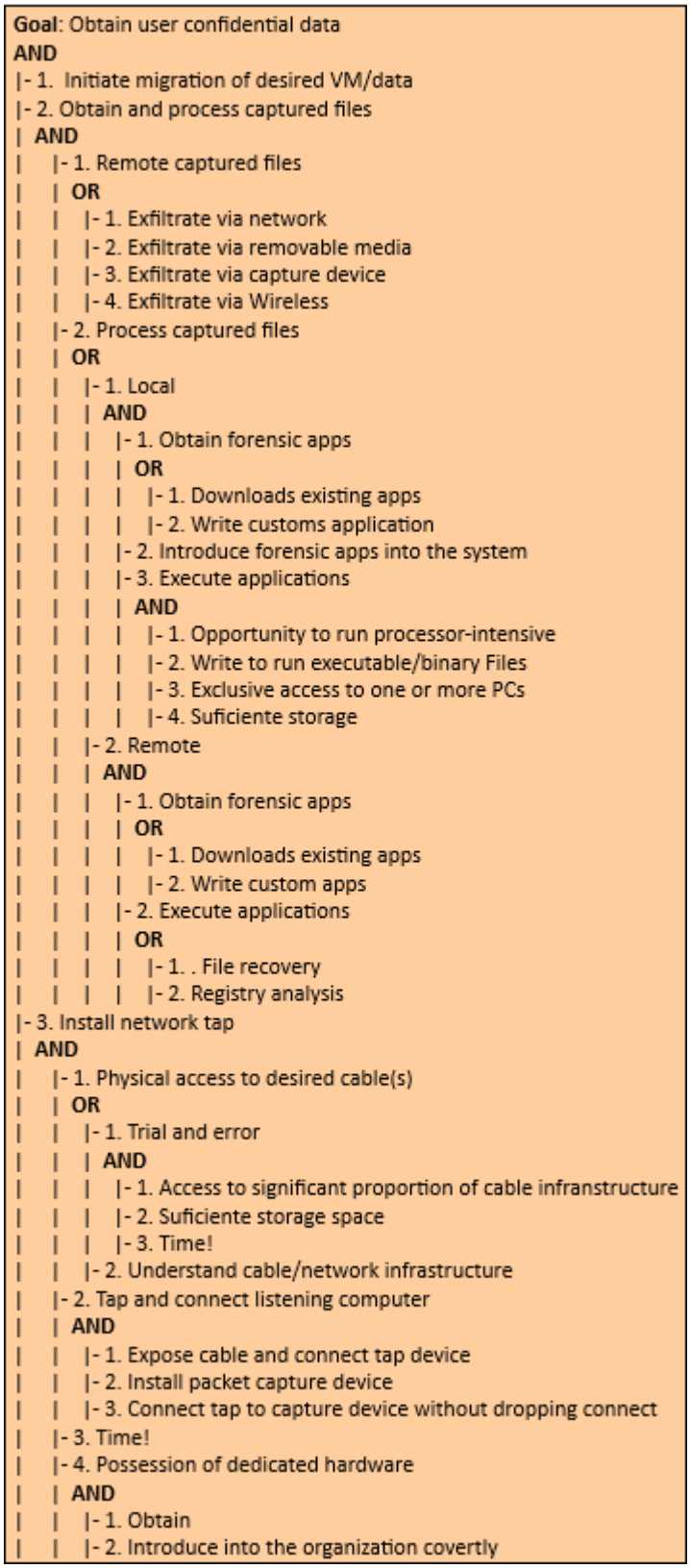
Recommendations

To ensure that the mobile application is resilient or immune to VM Migration attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity, privacy, confinement, and authenticity of the data.

References

1. [https://cwe.mitre.org/data/definitions/311.html].

VM Migration Attacks Diagram



Malicious Insiders Attacks

This type of attacks ocurre when there is a malicious entity (client, employee, Hypervisor, Cloud Provider/Broker, etc.) takes advantage of its privileges to covertly carry out any malicious activity such as information theft and data destruction or physical infrastructures.

Definition

Malicious Hypervisor, Malicious Clients, Malicious Cloud Provider/Broker, etc. are all the other terms which can also be used as an alternative to malicious insiders. This kind of attack occurs from client to server when the person, employee or staffs who know how the system runs, can implant malicious codes to destroy everything in the cloud system.

Technical Impact

- Read Application Data;
- Read Files or Directories;
- Modify Application Data;
- Modify Files or Directories;
- Gain Privileges or Assume Identity.

Analysis of Risk

- High.

Likelihood Of Exploit

- High.

Attacker Powers

- Implants malicious codes to destroy everything in the cloud system; * Steals confidential data.

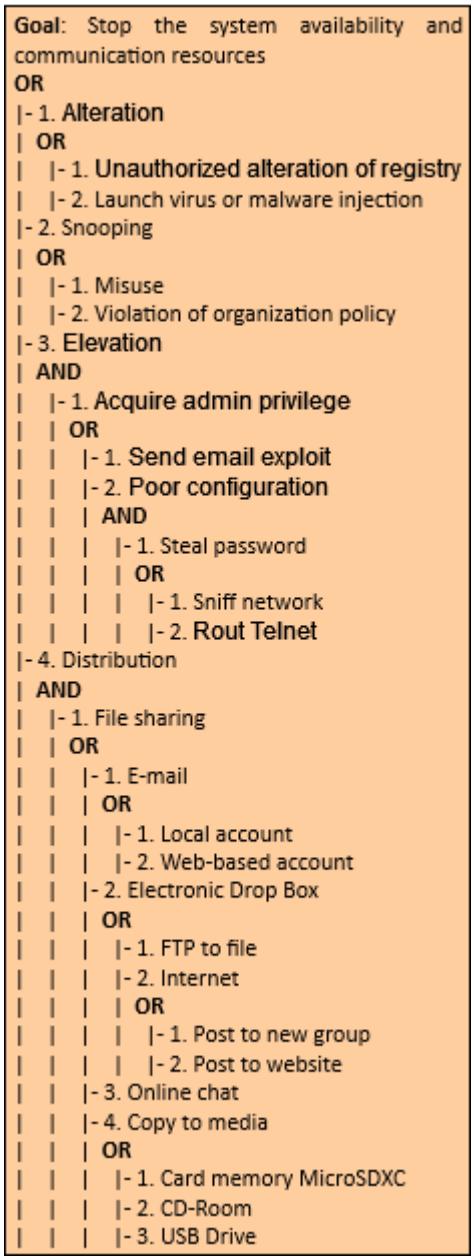
Recommendations

In order to ensure that the mobile application is resilient or immune to Malicious Insiders attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed.

References

1. [https://cwe.mitre.org/data/definitions/285.html].

Malicious Insiders Attacks Diagram



VM Escape Attacks

This type of attack occurs when an application escapes from the VM and gains control of VMM, as it escapes the VM privilege and obtains the root privilege.

Definition

VM escape is where an application running on a VM can directly have access to the host machine by bypassing the hypervisor, being the root of the system it makes this application escape the VM privilege and gain the root privilege. In this type of attack the attackers attempt to break down the guest OS in order to access the hypervisor or to penetrate the functionalities of other guest OS and underlying host OS. This breaking of the guest OS is called as escape. If the attackers escapes the guest OS it may compromise the hypervisor and as a result it may control over the entire guest OS. In this way the security breach in single point in hypervisor may break down all the hypervisor. If the attacker controls the hypervisor, it can do anything to the VM on the host system.

Risk Analysis

- Critical Risk.

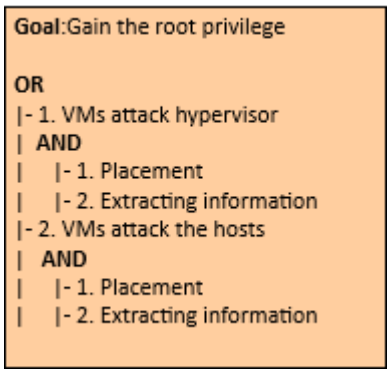
Attacker Powers

- Shutdown and eliminate target or victim VMs, resulting in the loss and destruction of data or information;
- Compromise the hypervisor and other resources.

Recommendations

To ensure that the mobile application is resilient or immune to VM Escape attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity, privacy, authenticity and confinement of the data.

VM Escape Attacks Diagram



Side-Channel Attacks

It is a type of attack enabled by leakage of information from a physical cryptosystem.

Definition

Side-channel attacks use statistical models such as differential analysis and correlation analysis on the information leaked from the cryptographic device during runtime. While early attacks required attackers to be in physical possession of the device, newer side-channel attacks such as cache-timing attacks or DRAM row buffer attacks are conducted remotely by executing malicious software in the targeted cloud environment. Regarding smartphones/tablets, they have developed more sophisticated side-channel attacks that target the built-in sensors of these devices, allowing them to infer keyboard input on touchscreens through sensor readings of native applications and websites, infer a user's location by the power consumption available in the proc file system (procfs), and infer a user's identity, location and diseases through procfs.

- Time-driven side-channel attack;
- Trace-driven side-channel attacks;
- Access-driven side-channel attacks.
- Power Analysis;
- Electromagnetic Analysis;
- Laser/optical;
- Clock/power Glitch;
- Temperature Variation;
- EMFI;
- Differential Computation Analysis
- Reflection/hands;
- Smudges;
- Network Traffic Analysis;
- USB Power Analysis;
- Wi-Fi Signal Monitoring;
- Figerprinting Devices;
- Data-usage Statistics;
- Page Deduplication;
- Procfs Leaks;
- Microarchitectural Attacks;
- Location Inference;
- Speech Recognition;
- Soundcomber;
- Sensor-based Keyloggers;
- Rowhammer.

Technical Impact

- Modify and Read Memory;
- Read Files or Directories;
- Modify Files or Directories;

- Execute Unauthorized Code or Commands;
- Gain Privileges or Assume Identity;
- Bypass Protection Mechanism;
- Read Application Data;
- Modify Application Data;
- Hide Activities.

Risk Analysis

- High Risk.

Likelihood of Exploit

- Low.

Attacker Powers

- Steal cryptographic information;
- Extract cryptographic key;
- Obtains confidential data or sensitive information.

Recommendations

In order to ensure that the mobile application is resilient or immune to the side-channel attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. Grassi, P.A., et al., 2017. Digital identity guidelines. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP. 800-63-3.pdf, doi:https://doi.org/10.6028/NIST.SP.800-63-3.
2. Spreitzer, R., et al., 2018. Systematic classification of side-channel attacks: A case study for mobile devices. IEEE Communications Surveys Tutorials 20, 465–488. doi:10.1109/COMST.2017.2779824.

Cross VM Attacks Diagram

Goal: Stealing and accessing sensitive user or system information
OR
|- 1. Passive
| **OR**
| |- 1. Physical
| | **OR**
| | |- 1. Local (Chip, Device)
| | | **OR**
| | | |- 1. Power Analysis Attacks
| | | |- 2. Electromagnetic Analysis Attacks
| | | |- 3. Smudge Attacks
| | | |- 4. Shoulder Surfing and Reflections
| | | |- 5. Hand/Device Movements
| | |- 2. Vicinity (Wire/Communication)
| | | **OR**
| | | |- 1. Electromagnetic Analysis Attacks
| | | |- 3. Shoulder Surfing and Reflections
| | | |- 3. USB Power Analysis
| | | |- 4. Wi-Fi Signal Monitoring
| | |- 3. Remote (Sofware, Web)
| | | **OR**
| | | |- 1. Linux-inherited procfs Leaks
| | | |- 2. Data-Usage Statistics
| | | |- 4. Microarchitectural Attacks
| | | |- 4. Sensor-based Keyloggers
| | | |- 5. Fingerprinting Devices/Users
| | | |- 6. Location Inference
| | | |- 7. Speech Recognition
| | | |- 8. Soundcomber
| |- 2. Logical
| | |- 1. Local (Chip, Device)
| | | **OR**
| | | |- 1. Differential Computation Analysis
| | |- 2. Vicinity (Wire/Communication)
| | | **OR**
| | | |- 1. Network Traffic Analysis
| | |- 3. Remote (Sofware, Web)
| | | **OR**
| | | |- 1. Fingerprinting Devices/Users
| | | |- 2. Page Deduplication
| | | |- 3. Linux-inherited procfs Leaks
|- 2. Active
| **OR**
| |- 1. Physical
| | **OR**
| | |- 1. Local (Chip, Device)
| | | **OR**
| | | |- 1. Clock/Power Glitching
| | | |- 2. Electromagnetic Fault Injection
| | | |- 3. Laser/Optical Faults
| | | |- 4. Temperature Variation
| | | |- 5. NAND mirroring
| | |- 2. Vicinity (Wire/Communication)
| | |- 3. Remote (Sofware, Web)
| | | **OR**
| | | |- 1. Microarchitectural Attacks
| | | |- 2. Rowhammer
| |- 2. Logical
| | |- 1. Local (Chip, Device)
| | | **OR**
| | | |- 1. Differential Computation Analysis
| | |- 2. Vicinity (Wire/Communication)
| | | **OR**
| | | |- 1. Network Traffic Analysis
| | |- 3. Remote (Sofware, Web)

Malware-as-a-Service

This type of attack occurs whenever a user can install malware on a mobile device. In addition, this type of attack can be carried out remotely or locally.

Definition

Attacks on the cloud and mobile application-level ecosystem can affect the integrity and confidentiality of data and applications through different strategies. E.g., by injecting malware. Malware can be virus, worm, trojan, rootkit and botnet.

Technical Impact

- Execute Unauthorized Code or Commands;
- Read Application Data.

Risk Analysis

- Critical Risk.

Likelihood of Exploit

- Medium.

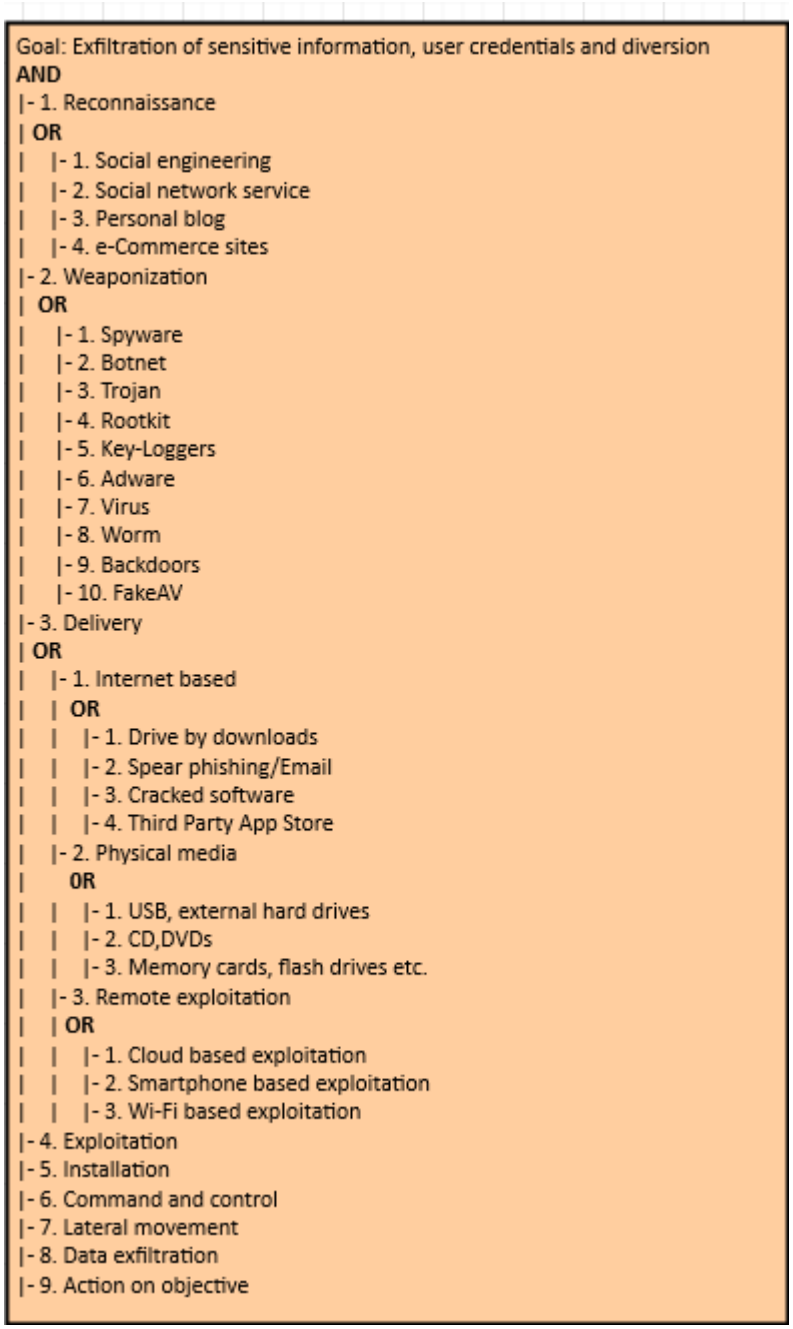
Attacker Powers

- Access and steal users confidential data;
- Obtain root permissions on mobile devices and control the mobile device;
- Directly affect the computational integrity of mobile platforms along with the application.

Recommendations

To ensure that the mobile application is resilient or immune to malicious Malware Injection attacks, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity and authenticity of the data.

Malware-as-a-Service Diagram



Tampering Attacks

In this type of attack an attacker preforms physical modifications on the hardware where the software is implemented.

Definition

This type of attack occurs whenever an unauthorized user has physical access to the device. When this access is realized, it is possible to loss, leakage, access or unintentionally disclose of the data or applications to unauthorized users, if the mobile devices are misplaced, lost or theft.

Technical Impact

- Read and Modify Application Data.

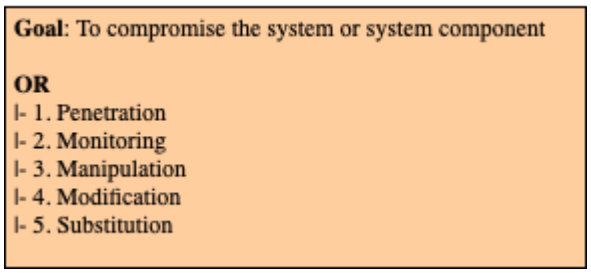
Attacker Powers

- Sending high malicious traffic stream;
- Huge messages to targeting mobile devices to make unused or reducing the capability;
- Access and steal users confidential data.

Recommendations

To ensure that the mobile application is resilient or immune to malicious Tampering attack, it is recommended that the measures described in the good practice report and the security tests present in the full report are followed to ensure authenticity, integrity, privacy and authenticity of the data.

Tampering Attacks Diagram



Bluejacking, Bluesnarfing and BlueSmacking Attacks

These are DDoS-type attacks that target a Bluetooth wireless network in order to shut down activity on it. It usually occurs through an attack coming from a connection of malicious entities in a target network.

Definition

Bluejacking occurs by sending unsolicited messages between the mobile devices (host nodes) over the Bluetooth connection. The unauthorized information can be accessed from a mobile device through Bluesnarfing to Bluetooth enabled devices using OBject EXchange (OBEX) protocol. Through the Bluejacking attack, attackers can send unwanted sounds, videos to other Bluetooth enabled devices. Bluesnarfing attack consists of using Bluetooth connection for the purpose of stealing sensitive information (contacts, emails, passwords, photos, and other useful data) from wireless devices such as smartphones, tablets and IoT. In a Bluesmacking cenario attack, an adversary uses Bluetooth flooding to transfer large packets to Bluetooth enabled devices over the L2CAP protocol with the goal of creating a DoS.

Technical Impact

- Resource Consumption;
- Malware Injection;
- Unreliable Execution;
- Read Data.

Typical Severity

- Medium.

Risk Analysis

- High Risk.

Likelihood of Exploit

- Medium.

Recommendations

In order to ensure that the mobile application is resilient or immune to the Bluejacking, Bluesnarfing and BlueSmacking attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. atel, N., Wimmer, H., Rebman, C.M., 2021. Investigating bluetooth vulnerabilities to defend from attacks, in: 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), IEEE, Ankara, Turkey. pp. 549–554. doi:10.1109/ISMSIT52890.2021.9604655.

Bluejacking, Bluesnarfing and BlueSmacking Attacks Diagram

Goal: To make services unavailable, to inject malware and to steal user/system sensitive information

AND
|- 1. Adquire and setup penetration testing tool
|- 2. Turn on Bluetooth on penetration testing tool, such as Kali Linux
|- 3. Installing “hcitool” and executing “hciconfig”
|- 4. From the terminal, scan the connected Bluetooth (hcitool scan) devices
|- 5. Make sure you can ping all device by using <l2ping 'MAC address'> command
| **OR**
| |- 1. Bluesmack/DoS Attack
| | **AND**
| | |- 1. Clone DOS attack script from GitHub or another repository
| | |- 2. Implement DoS attack using python3 Bluetooth-DOS-Attack.py from DoS attack script folder
| | |- 3. Specify the target address of the device using MAC address of them (Target addr > 'MAC address')
| | |- 4. Setup packages size and Treads counts (e.g., Packages size > 600; Treads counts > 512)
| |- 2. Bluejacking
| | **AND**
| | |- 1. Execute bluetooth-sendto --device='MAC Address' 'file.mp3' command from mobile device to target mobile device
| |- 3. Bluesnarfing
| | **AND**
| | |- 1. Execute bluesarfing attack using bluesnarfer –r 1-100 –b (Bluetooth device address) command from Kali Linux

GPS Jamming Attacks

This is a DoS attack that targets the GPS sensor, aiming to make this service (position, path, speed, direction, time, and distance) unavailable to users of the target mobile devices.

Definition

This attack aims to interrupt or obstruct the communication between the emitting satellite and the device (smartphone/tablet) receiving the GPS signal. Normally, the attack consists of blocking the signal from the receiver, since the receiving signal is weaker compared to the broadcasting signal, and can be carried out in two different ways:

- Blanket Jamming;
- Deception Jamming.

Technical Impact

- Service unavailability.

Typical Severity

- High

Risk Analysis

- High Risk.

Likelihood of Exploit

- Low.

Recommendations

In order to ensure that the mobile application is resilient or immune to the GPS Jamming attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. CAPEC-627: Counterfeit GPS Signals.

GPS Jamming Attacks Diagram

Goal: Disrupt and make GPS services unavailable

AND
|- 1. Acquire jammer tool (e.g.,)
|- 2. Setup jammer tool
OR
|- 1. Cigarette lighter COTS GPS jammers
|- 2. SMA-battery COTS GPS jammers
|- 3. Non SMA-battery COTS GPS jammers
|- 4. Wide-band COTS GPS jamming attacks on maritime systems
|- 5. GPS jamming attack on PMUs

DoS (Cellular) Jamming Attacks

This type of attack aims to dominate and disrupt communication between a user's mobile device and the cell tower by actively transmitting signals.

Definition

Interference attacks target radio communication technology (communication between smart devices and base stations). This attack can be caused by noise, interference, disruption or by sending corrupted data packets, with the purpose of causing DoS in the physical transmission of signals on certain routes.

Technical Impact

- Resource Consumption.

Typical Severity

- Low.

Risk Analysis

- High Risk.

Likelihood of Exploit

- Low.

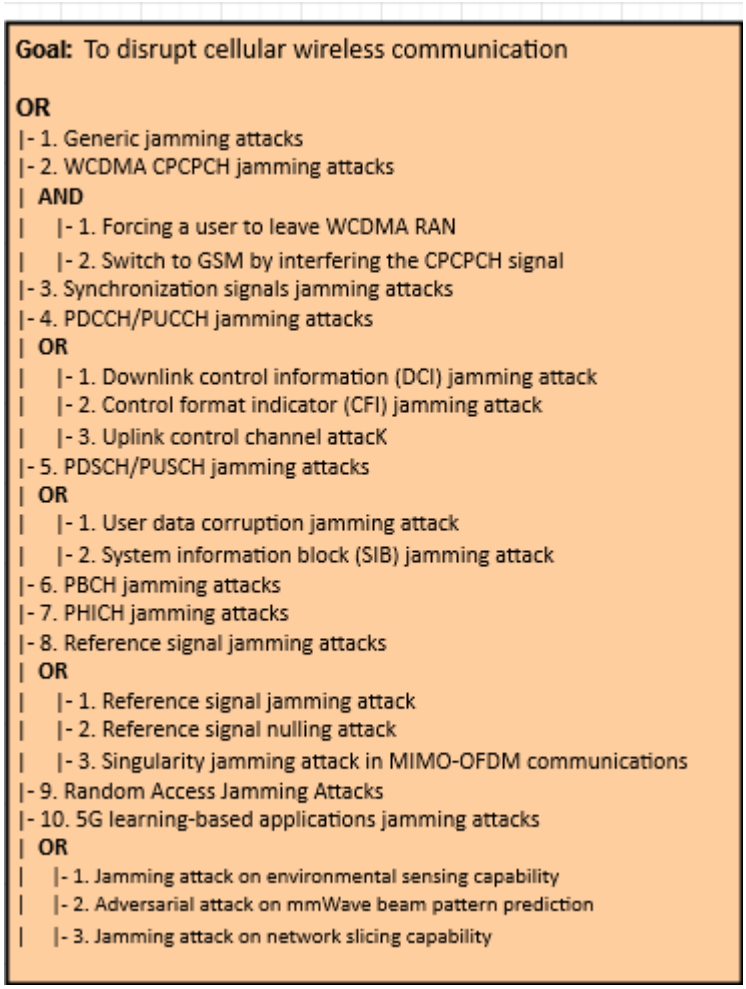
Recommendations

In order to ensure that the mobile application is resilient or immune to the DoS Jamming attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. CAPEC-605: Cellular Jamming. 2.Moorthy, V., Venkataraman, R., Rama Rao, T., 2020. Security and privacy attacks during data communication in software defined mobile clouds. Computer Communications 153, 515–526. URL: https://www.sciencedirect.com/science/article/pii/S0140366419317268, doi:https://doi.org/10.1016/j.comcom.2020.02.030.

DoS Jamming Attacks Diagram



Cryptanalysis Attacks

This attack consists in deciphering a ciphered message without knowing the decryption key by exploiting vulnerabilities in the cryptographic algorithm.

Definition

Cryptanalysis focuses on finding vulnerabilities in cryptographic algorithms and using these weaknesses to decrypt the ciphertext without knowing the secret key. In addition, this can have other purposes such as Total Breach, Global Deduction, Information Deduction, and Distinguishing Algorithm.

Technical Impact

- Read Data.

Typical Severity

- Very High.

Risk Analysis

- Very High Risk.

Likelihood of Exploit

- Low.

Recommendations

In order to ensure that the mobile application is resilient or immune to the Cryptanalysis Attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. CAPEC-97: Cryptanalysis.

Cryptanalysis Attacks Diagram

Goal: To discover hidden texts using cipher or discover secrets used in cipher systems to hide one or more texts

OR
|- 1. Passive Attacks
| **OR**
| |- 1. Ciphertext-only attack
| |- 2. Known-plaintext attack
| |- 3. Chosen-plaintext attack
| |- 4. Adaptive chosen-plaintext attack
| |- 5. Chosen-ciphertext attack
| |- 6. Adaptive chosen-ciphertext attack
|- 2. Active Attacks
| **OR**
| |- 1. Brute-force attacks
| |- 2. Man-in-the-Middle attacks

Reverse Engineering Attacks

Typically, this attack consists of using specific tools to analyze the target application (feature or resources), within its own local environment, downloaded from a mobile application store, such as Apple's App Store and Google's Play Store, respectively.

Definition

Reverse engineering attacks (REA) target the assets embedded in software. In such an attack scenario, the attacker by reverse engineering attempts to steal confidential information, such as embedded cryptographic keys or intellectual property in the form of algorithms. There are two ways of carrying out this type of attack:

- White Box Reverse Engineering;
- Black Box Reverse Engineering.

Technical Impact

- Reveal information about back end servers;
- Reveal cryptographic constants and ciphers;
- Steal intellectual property;
- Perform attacks against back end systems;
- Gain intelligence needed to perform subsequent code modification.

Risk Analysis

- High Risk.

Likelihood of Exploit

- Low.

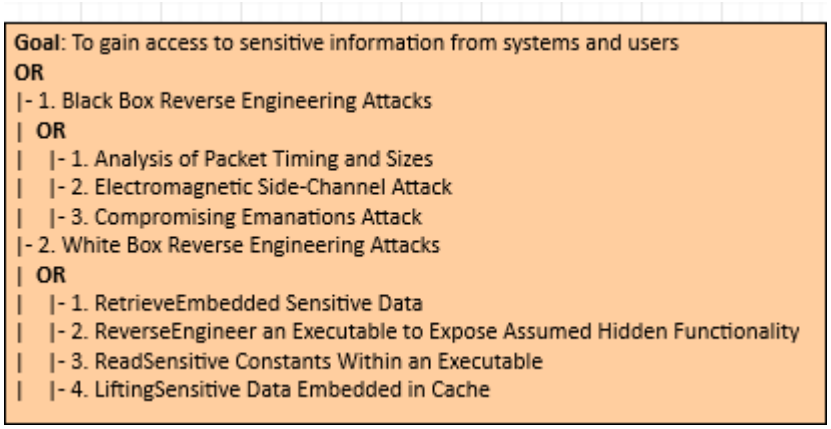
Recommendations

In order to ensure that the mobile application is resilient or immune to the Reverse Engineering attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. Basile, C., et al., 2019. A meta-model for software protections and reverse engineering attacks. Journal of Systems and Software 150, 3–21. URL: https://www.sciencedirect.com/science/article/pii/ S0164121218302838, doi:https://doi.org/10.1016/j.jss.2018.12.025.
2. M9: Reverse Engineering.

Reverse Engineering Attacks Diagram



Audit Log Manipulation Attacks

This type of attack targets log files for the purpose of manipulating (deleting, reading, and altering) them.

Definition

In a log file audit manipulation attack scenario, an attacker injects, manipulates, deletes, or forges malicious entries in the log file in an attempt to deceive a log file audit or to cover impressions of an attack. The success of this type of attack depends on the insufficiency of log file access controls mechanisms.

Technical Impact

- Modify Data.

Typical Severity

- High.

Risk Analysis

- High Risk.

Likelihood of Exploit

- High.

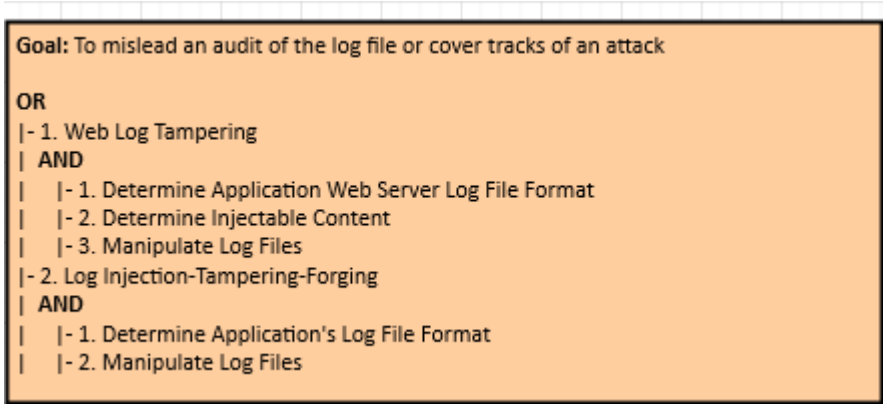
Recommendations

In order to ensure that the mobile application is resilient or immune to the Audit Log Manipulation Attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. CAPEC-268: Audit Log Manipulation.

Audit Log Manipulation Attacks Diagram



Wi-Fi Jamming Attacks

In a scenario of this type of attack, the attacker targets the wireless network made available from the access point, with the aim of making it unavailable.

Definition

This is a denial-of-service attack that blocks the radio frequency, making access to the Wi-Fi network and consequently to the Internet unavailable. Generally, two techniques are used to carry out this type of attack, namely: 1) The attacker may flood the Wi-Fi access point (e.g. the retransmission device) with deauthentication frames; 2) Another method is to transmit high levels of noise on the RF band used by the Wi-Fi network.

Technical Impact

- Resource Consumption.

Typical Severity

- High.

Risk Analysis

- High Risk.

Likelihood of Exploit

- Medium.

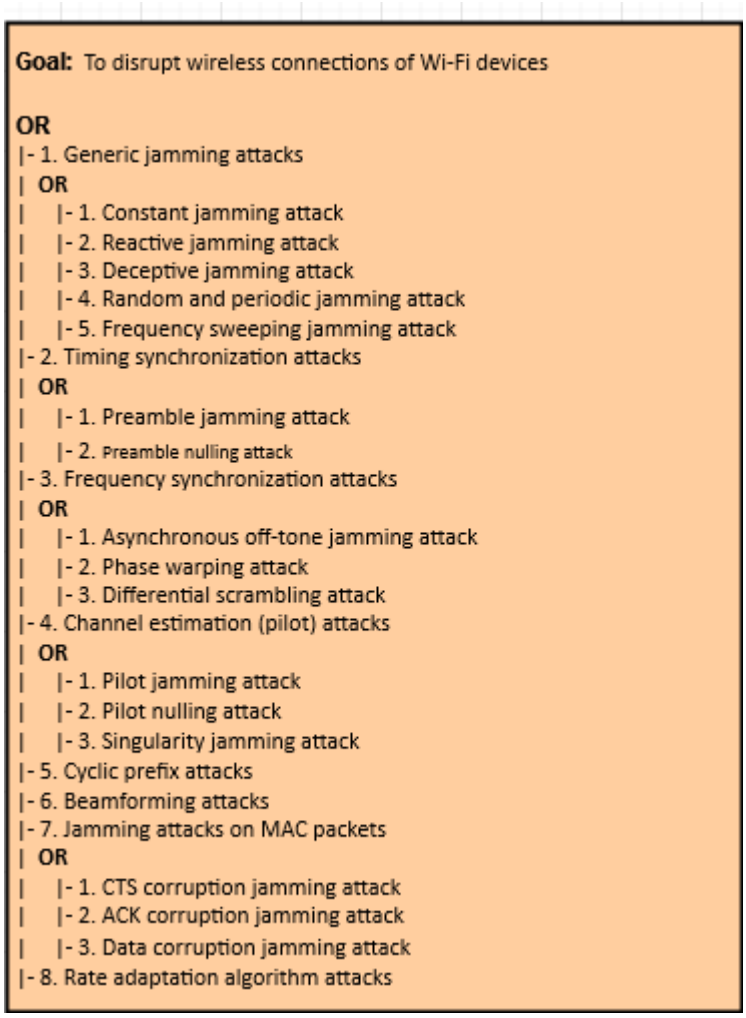
Recommendations

In order to ensure that the mobile application is resilient or immune to the Code Inclusion attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. CAPEC-175: Code Inclusion.

Code Inclusion Attacks Diagram



Wi-Fi SSID Tracking Attacks

Unlike code injection, in this type of attack, an attacker exploits a weakness in the target in order to force arbitrary code to be retrieved locally or from a remote location and executed.

Definition

This type of attack aims to obtain sensitive data (location, routine, trajectory, etc.) of users of mobile devices using Wi-Fi networks to access the Internet. Furthermore, it consists of using sophisticated sniffing devices to bypass authentication (for closed networks), extract and identify the MAC address of the mobile device and establish a match with its potential owner.

Technical Impact

- Read Data;
- Bypass Protection Mechanism.

Typical Severity

- Low.

Risk Analysis

- Very High Risk.

Likelihood of Exploit

- Medium.

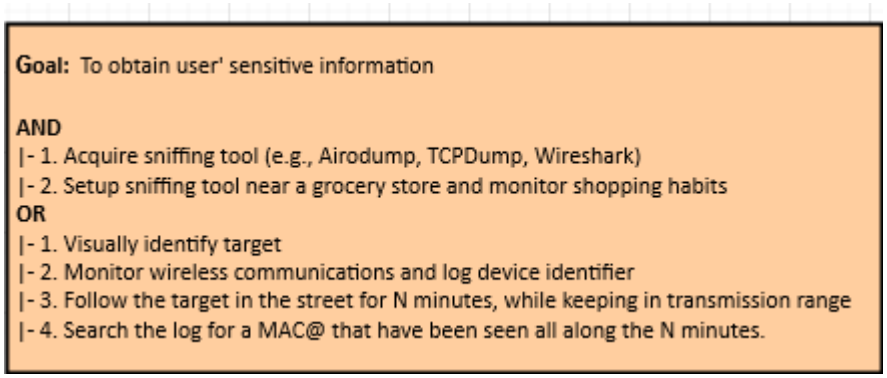
Recommendations

In order to ensure that the mobile application is resilient or immune to the Wi-Fi SSID Tracking attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. Matte, C., 2017. Wi-Fi tracking: Fingerprinting attacks and countermeasures. Ph.D. thesis. Université de Lyon.

Wi-Fi SSID Tracking Attacks Diagram



Byzantine Attacks

In a Byzantine attack scenario, the attacker targets the routing protocols of an ad hoc wireless network, aiming to access and modify sensitive data exchanged between two or more entities on this network.

Definition

In a mobile ad hoc wireless network, Byzantine attacks are defined as attacks that target routing protocols, in which two or more routers collude to drop, fabricate, modify, or divert packets in an attempt to disrupt routing services.

Technical Impact

- Read Data;
- Modify Data;
- Denial of service.

Typical Severity

- High.

Risk Analysis

- High Risk.

Likelihood of Exploit

- Medium.

Recommendations

In order to ensure that the mobile application is resilient or immune to the Byzantine Attacks, it is recommended that the measures described in the good practice report and the security testing present in the full report are followed.

References

1. Yu, M., et al., 2009. A secure routing protocol against byzantine attacks for manets in adversarial environments. IEEE Transactions on Vehicular Technology 58, 449–460. doi:10.1109/TVT.2008.923683.

Byzantine Attacks Diagram

Goal: To disrupt the whole network communication and transmission of data

AND
|- 1. Making the active set of internal network nodes malicious
|- 2. Control the entire network
| **OR**
| |- 1. Black hole or sink hole attacks
| |- 2. Byzantine Wormhole attacks
| |- 3. Byzantine Overlay Network Wormhole Attack
| |- 4. Gray hole attacks
| |- 5. Flood Rushing attack
| |- 6. Selfish Node Attack

Final Security Test Specification and Tools Report

Mobile Plataform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Channel-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	Local Storage (Centralized Database)
Which DB	
Type of data stored	Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	Java ; HTML5
Input Forms	Yes
Upload Files	Yes
The system has logs	Yes
The system has regular updates	Yes
The system has third-party	Yes
System Cloud Environments	Public Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Bluetooth ; Wi-Fi ; GPS ; NFC
Data Center Phisical Access	Yes

In order to avoid or prevent *DoS Jamming, Wi-Fi Jamming, Orbital Jamming, GPS Jamming, Flooding* attacks, the following security tests should be performed.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
DoS and DDoS Attacks	Black Box	Dynamic Analysis	Penetration Testing	NMAP, SlowBot Net, MetaSploit, LOIC, Kali Linux		
Web Server Authentication	Black Box	Dynamic Analysis	Proxies	Wireshark	tPacketCapturepro	
DoS and DDoS Attacks	Grey Box	Static Analysis	Penetration Testing	Cydia Substrate		Cycript

In order to avoid or prevent *Malicious Insider, Sniffing, MiTM, Eavesdropping* attacks, the following security tests should be performed.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Data leakage and Breach	Grey Box	Dynamic analysis	Proxies	Wireshark	tPacketCapturepro, AFWall+	
	Grey Box	Dynamic Analysis	Penetration Testing	VASTO		
	White Box	Dynamic Analysis	Stressing Testing (fuzzing)	Webfuzz, Wfuzz		
	Grey Box	Dynamic analysis	Vulnerability Scanner	Acunetix, W3af, Nikto, Fortify WebInspect		
	Grey Box	Dynamic Analysis	Penetration Testing	TCPDump, Wireshark		
Secure backup, logging and Insecure Data Storage	Black Box	Dynamic Analysis	Proxies		adb	

In order to avoid *MiTM, Eavesdropping, Side-Channel, VM Escape, Wi-Fi SSID Tracking, Rogue Access Point, Cellular Rogue Base Station, Sniffing, Cryptanalysis, Audit Log Manipulation Attacks, Byzantine, On-Off, Brute Force*, the following security tests should be perform.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Proper SSL usage and Insecure TLS Protection, Use of encryption	White Box	Static analysis	Forensic Mobile	XRY, UFED Touch, OpenSSL	AndroGuard, MalloDroid, apktool, Amandroid	
Interception of network	Grey Box	Hybrid	Penetration Testing	Burp Suite, Wireshark, bettercap		
Interception of network	Black Box	Dynamic Analysis	Proxy	mitm-relay, Kali Linux, Burp Suite		

Poor use of certificate parameters	Black Box	Dynamic analysis	Proxies	NMAP, Nessus, Metasploit Framework		
Data leakage	Grey Box	Dynamic analysis	Proxies	Wireshark	tPacketCapturepro, AFWall+	
Secure backup, logging and Insecure Data Storage	Grey Box	Dynamic Analysis	Proxies, Penetration Testing	Frida	adb	PassFab iPhone Backup Unlocker
Secure backup, logging and Insecure Data Storage	White Box	Dynamic Analysis	Mobile Forensic		Logcat	
Web Server connection	Black Box	Manual Dynamic Analysis	Proxies	OWASP WebScarab, OWASP ZAP, Paros		
Web Server Authentication	Black Box	Dynamic Analysis	Proxies	Wireshark, CERT Tapioca	tPacketCapturepro	
Dynamic binary analysis	Black Box	Dinamic Analysis	Penetration Testing		Introspy-Android	Introspy-iOS

In order to avoid or prevent *Malware as a Service, Malicious QR Code, Botnet, Spoofing and Eavesdroping, NFC Payment Replay, Byzantine, Bluesnarfing, Bluejacking* attacks, the following security tests should be performed.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Malware and Privacy Scanners	Grey Box	Static Analysis	Forensic Mobile	Bitdefender, Norton, McAfee, Kaspersky	SandDroid	
Data Leakage	Black Box	Dinamic Analysis	Proxies	Wireshark	tPacketCapturepro	
Authentication and Authorization, Use of Encryption Encryption, Authentication and Authorization, Web Server Authentication, Access Control	Black Box	Dinamic Analysis	Proxies		NFCSpy	
	Black Box	Dinamic Analysis	Penetration Testing	Kali Linux, hcitool		

In order to avoid or prevent *Bypassing Physical Security, Physical Theft and VM Migration attacks*, the following security tests should be performed.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Data leakage and Breach	White Box	Static Analysis	Forensic Mobile	BlackBag Blacklight, Encase Forensics, Oxygen Forensic Suite	Androguard , Drozer, SpotBugs, Andriller	Elcomsoft iOS Forensic Toolkit

In order to avoid or prevent *Malware as a Service, Malicious QR Code, Botnets, Spoofing, Eavesdroping, NFC Payment Replay, Byzantine, Bluesnarfing, Bluejacking, Side-Channel, Flooding attacks*, the following security tests should be performed.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Malware and Privacy Scanners	Grey Box	Static Analysis	Forensic Mobile	Bitdefender, Norton, McAfee, Kaspersky	SandDroid	
Data Leakage	Black Box	Dinamic Analysis	Proxies	Wireshark	tPacketCapturepro, AFWall+	
Authentication and Authorization, Use of Encryption Encryption, Authentication and Authorization, Web Server Authentication, Access Control	Black Box	Dinamic Analysis	Proxies		NFCSpy	
	Black Box	Dinamic Analysis	Penetration Testing	Kali Linux, hcitool		
Use of encryption, Secure backup, logging and Insecure Data Storage	White Box	Static Analysis	Forensic Mobile	Slueth Kit + Autopsy Browser	AndroGuard, Drozer, apktool, Amandroid	
Dynamic binary analysis: debugging, tracing	White Box	Hybrid Analysis	Vulnerability Scanner	RMS	Drozer, Sieve	

Secure backup, logging and Insecure Data Storage	Grey Box	Static Analysis	Mobile Forensic			iOSbackup

In order to avoid or prevent *Spoofing, Eavesdropping, Sniffing, Botnets, MiTM, Flooding, Reverse Enginnering attacks*, the following security tests should be performed.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Mobile decryption, unpacking & conversion	White Box	Static Analysis	Penetration Testing	Ghidra	Dex2jar, JD-GUI, Dextra	Clutch
Mobile decryption, unpacking & conversion	Black Box	Static Analysis	Penetration Testing	MobSF	APKEnum	Damn Vulnerable iOS App
Secure backup, logging and Insecure Data Storage	Grey Box	Dynamic Analysis	Proxies		adb	
Static binary analysis: disassembly, decompilation	Grey Box	Static Analysis	Manual (Reversed) Code Review	r2ghidra-dec,r2frida, Radare2		Hooper

In order to avoid or prevent *Malware as a Service, Side-Channel and Botnets* attacks, the following security tests should be performed.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Use of encryption, Secure backup, logging and Insecure Data Storage	White Box	Static Analysis	Forensic Mobile	Slueth Kit + Autopsy Browser	AndroGuard, Drozer, apktool, Amandroid	

In order to avoid or prevent *Phishing, Botnet, Malware as a Service* attacks, the following security tests should be performed.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Add-ons	White Box	Static Analysis	Forensic Mobile		Addons Detector	

In order to avoid or prevent *Spoofing, Eavesdrooping, Botnets, Flooding* attacks, the following security tests should be performed.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Web Server Authentication	Black Box	Dynamic Analysis	Proxies	Wireshark	tPacketCapturepro	
DoS and DDoS Attacks	Grey Box	Static Analysis	Penetration Testing	Cydia Substrate		Cycript

In order to avoid *SQLi, Command Injection, Session Hijacking, Botnets, AP Hijacking, Brute Force, Phishing, Spoofing, MiTM, Buffer Overflow, Sniffing, CSRF, VM Migration* attacks, the following security tests should be perform.

Test Parameter	Testing Types	Testing Analysis	Method	Tools		
				Both	Android	iOS
Dynamic binary analysis	Black Box	Dinamic Analysis	Penetration Testing		Introspy-Android	Introspy-iOS