

Final Security Requirements Report

Mobile Platform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	SQL
Which DB	MySQL
Type of data stored	Personal Information ; Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	HTML5 ; PHP ; Javascript
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	Private Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Wi-Fi ; GPS
Data Center Physical 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 where 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.

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 its life-cycle.

Note: *This requirement 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. # 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 needs to be always available to access by authorized people.

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

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

1. DoS Attacks

In this type of attacks, the attacker attempts to prevent the provision of a service or resource that are signed by authorized users by launching various types of flood.

2. DDoS Attacks

It is an improved case of DoS attacks in terms of flooding the target server with server with a huge amount of packets.

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 performed at the remote endpoint;
- if stateful session management is used, the remote endpoint uses randomly generated session identifiers to authenticate client requests without sending the user's credentials;
- If stateless token-based 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 enforced 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 informs 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. Botnet 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 Attack

In a scenario of this type of attack, when using cloud services, an attacker can conduct phishing attacks by manipulating the web link to redirect it to a false link and hijack the user account for the purpose of stealing the your sensitive data.

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 user.

9. Hypervisor Attacks:

In this type of attack the attacker aims to compromise the authenticity of sensitive user data and the availability of services from the cloud at the VM level.

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#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 where 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 code 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 user.

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 Overflows

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 Hijacking

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

12. Session Fixation

An attacker has a valid session ID and forces the victim to use this ID.

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.*

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). # 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.*

Privacy

In the context of cloud and mobile, privacy refers to the control of the user over the disclosure of his data. In other words only the user has control of the sharing of is personal information and his data is only made public if the user allowed it.

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

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. Physical Attack

This type of attack occurred when the perpetrator gains physical access to the location where the system is operating and tries to gain information stored in the system using his physical access.

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 nuds 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 nuds 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.

Final Security Good Practices

Mobile Platform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	SQL
Which DB	MySQL
Type of data stored	Personal Information ; Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	HTML5 ; PHP ; Javascript
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	Private Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Wi-Fi ; GPS
Data Center Physical Access	Yes

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, Cyrillic, 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 $K1 \neq K2 \neq K3$, 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

Authentication and Integrity

Introduction

This cheat sheet provides a simple model to follow when implementing transport layer protection for an application. Although the concept of SSL is known to many, the actual details and security specific decisions of implementation are often poorly understood and frequently result in insecure deployments. This article establishes clear rules which provide guidance on securely designing and configuring transport layer security for an application. This article is focused on the use of SSL/TLS between a web application and a web browser, but we also encourage the use of SSL/TLS or other network encryption technologies, such as VPN, on back end and other non-browser based connections.

Architectural Decision

An architectural decision must be made to determine the appropriate method to protect data when it is being transmitted. The most common options available to corporations are Virtual Private Networks (VPN) or a SSL/TLS model commonly used by web applications. The selected model is determined by the business needs of the particular organization. For example, a VPN connection may be the best design for a partnership between two companies that includes mutual access to a shared server over a variety of protocols. Conversely, an Internet facing enterprise web application would likely be best served by a SSL/TLS model.

TLS is mainly a defence against man-in-the-middle attacks. An TLS Threat Model is one that starts with the question "What is the business impact of an attacker's ability to observe, intercept and manipulate the traffic between the client and the server".

This cheat sheet will focus on security considerations when the SSL/TLS model is selected. This is a frequently used model for publicly accessible web applications.

Providing Transport Layer Protection with SSL/TLS

Benefits

The primary benefit of transport layer security is the protection of web application data from unauthorized disclosure and modification when it is transmitted between clients (web browsers) and the web application server, and between the web application server and back end and other non-browser based enterprise components;

The server validation component of TLS provides authentication of the server to the client. If configured to require client side certificates, TLS can also play a role in client authentication to the server. However, in practice client side certificates are not often used in lieu of username and password based authentication models for clients;

TLS also provides two additional benefits that are commonly overlooked; integrity guarantees and replay prevention. A TLS stream of communication contains built-in controls to prevent tampering with any portion of the encrypted data. In addition, controls are also built-in to prevent a captured stream of TLS data from being replayed at a later time.

It should be noted that TLS provides the above guarantees to data during transmission. TLS does not offer any of these security benefits to data that is at rest. Therefore appropriate security controls must be added to protect data while at rest within the application or within data stores.

Good Practices

- Use TLS, as SSL is no longer considered usable for security;
- All pages must be served over HTTPS. This includes css, scripts, images, AJAX requests, POST data and third party includes. Failure to do so creates a vector for man-in-the-middle attacks;
- Just protecting authenticated pages with HTTPS, is not enough. Once there is one request in HTTP, man-in-the-middle attacks are possible, with the attackers being able to prevent users from reaching the secured pages.

The HTTP Strict Transport Security Header must be used and pre loaded into browsers. This will instruct compatible browsers to only use HTTPS, even if requested to use HTTP. Cookies must be marked as Secure.

Basic Requirements *

Access to a Public Key Infrastructure (PKI) in order to obtain certificates;

- Access to a directory or an Online Certificate Status Protocol (OCSP) responder in order to check certificate revocation status;
- Agreement/ability to support a minimum configuration of protocol versions and protocol options for each version.

[https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_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 its 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 receiver 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 protocol 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 Platform or OS version;
- USB Debugging activation;
- Device encryption;
- Device rooting (see also "Testing Root Detection").

Final Security Mechanisms Report

Mobile Platform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	SQL
Which DB	MySQL
Type of data stored	Personal Information ; Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	HTML5 ; PHP ; Javascript
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	Private Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Wi-Fi ; GPS
Data Center Physical 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.

Requirement	Platform	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 defense in depth	Network
			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.

Requirement	Platform	Mechanism	Mechanism Type	Description	Layer
-------------	----------	-----------	----------------	-------------	-------

Confiability,
Integrity,
authenticity,
audit,
accountability

Both

Audit

Record inspection
and analysis
mechanisms

Identity-based public cloud
auditing scheme 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.

Requirement	Plataform	Mechanism	Mechanism Type	Description	Layer
Privacy and confidentiality, authenticity, authorization	Both	Cryptographic algorithms and related mechanisms	TCP/TLS, HTTPS, XMPP, AES256-RSA, SSL/TLS, HTTPSCurve25519, AES-256, AES256-RSA2048	Encrypted communications	Presentation and Application
			MAC, Digital Signatures	Authentic communications	Presentation and Application
			AES-GCM-256 or ChaCha20- Poly1305	Confidentiality 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	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	Platform	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 can be used or implemented specifically 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.

Requirement	Platform	Mechanism	Mechanism Type	Description	Layer
Authorization, audit, authenticity, interoperability	Both	Access Control	RBAC, ABAC, ABE		Application
	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	Platform	Mechanism	Mechanism Type	Description	Layer
Privacy, authorization, immunity, Tampering Detection		Inspection	IDS, IPS, NIDS, NIPS, HIDS, HIPS		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	Platform	Mechanism	Mechanism Type	Description	Layer
Non repudiation, audit, accountability	Both	Logging	System log files or event log	It is recommended that developers, during the coding phase, use the native APIs of each of the mobile device platforms that allow incorporating Logging into applications during the software development process.	Data Link
			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	Platform	Mechanism	Mechanism Type	Description	Layer
Authorization, authenticity, privacy, immunity		Device Adulterion Detection	Incorporation of hybrid authentication schemes into 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	Platform	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	Platform	Mechanism	Mechanism Type	Description	Layer
Privacy, integrity, authenticity, immunity	Both	Confinement	Sandboxing	Its purpose is to guarantee the privacy, integrity and authenticity of the data of the end users and the integrity of the system	Application
	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	Platform	Mechanism	Mechanism Type	Description	Layer
Integrity, authenticity, access Control, Privacy	Both	Filtering	Firewall and Cryptographic Techniques		Network

Final Attack Models Report

Mobile Platform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	SQL
Which DB	MySQL
Type of data stored	Personal Information ; Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	HTML5 ; PHP ; Javascript
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	Private Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Wi-Fi ; GPS
Data Center Physical Access	Yes

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).

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.

Cross Site Scripting Attacks Diagram

Goal: To write an alert message in a victim's browser
OR
|- 1. Make reflected XSS
| **AND**
| |- 1. Make malicious URL
| |- 2. Send malicious URL to victim
|- 2. Make stored XSS
| **AND**
| |- 1. Make malicious URL
| |- 2. Put malicious script into vulnerable page
|- 3. Make DOM-Based XSS
| **AND**
| |- 1. Make malicious URL
| |- 2. Send maliciously scripted URL message
|- 4. Modify HTTP datagrams
| **AND**
| |- 1. Capture datagrams
| | **OR**
| | |- 1. Listen to the wireless network
| | |- 2. Access a network router
| |- 2. Modify package contents
| |- 3. Inject or send new packages

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.

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

Goal: To steal sensitive personal information
OR
|- 1. The attacker replaces the entire QR Code
| **AND**
| |- 1. Create a new QR Code with a malicious link
| |- 2. Pastes it over an already existing one on
| | **AND**
| | |- 1. Redirecting a user to a fraudulent page
|- 2. Encrypted Connection
| **OR**
| |- 1. Modifying individual modules of a QR Code
| | **AND**
| | |- 1. Attacker modifies the coded content
| | |- 2. Redirecting a user to a fraudulent page

SQL Injection Attacks

In this type of attack, an attacker could provide malicious input with a clever mix of characters and meta characters from a form (e.g., login form) to alter the logic of the SQL command.

Definition

Structured Query Language (SQL) Injection Attack is a code injection technique commonly used to attack web applications where an attacker enters SQL characters or keywords into an SQL statement through superuser input parameters for the purpose. to change the logic of the desired query.

Attacker Powers

- Identify parameters vulnerable to injection; * Discover DBMS and version; * Discover relational scheme; * Extract data; * Add / modify data; * Cause denial of service; * Evade detection; * Bypass authentication; * Execute commands; * Elevate privileges.

Recommendations

To ensure that the mobile application is resilient or immune to SQLi 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.

SQL Injection Attacks Diagram

Goal: Get full access to data base
AND
|- 1. Identify the vulnerability
|- 2. Create attack script
|- 3. Inject attack script
|- **OR**
| |- 1. Use input fields in the web
| |- 2. Use URL
|- 4. Execute attack script on the server

Denial of Services

In a DoS attack scenario, the attacker attempts to disrupt the network or disable services provisioned by a server by sending uninterrupted data packets to the target server and without changing nodes, data packets, or decrypting encrypted data. Typically, these data packets take up bandwidth and consume server resources.

Definition

In such attacks, the attacker attempts to prevent a service or feature that is signed by authorized users from being released by launching various types of floods - SYN flooding, User Datagram Protocol (UDP) flooding, Internet Control Message Protocol (ICMP) attacks) flooding, etc - on the server.

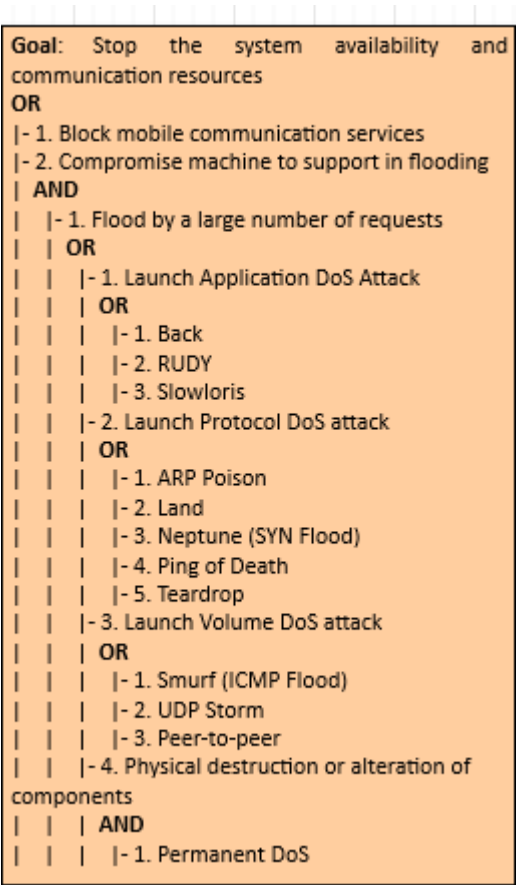
Attacker Powers

- Prevent the availability of a service or resource to authorized users;
- Perpetrating other types of attacks while services or features are unavailable, such as Spoofing.

Recommendations

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

Denial of Services Attacks Diagram



Distributed Denial of Services Attacks

Distributed Denial of Services (DDoS) is an enhanced 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.

Definition

The DDoS 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 (eg, mobile applications, database service) from almost every attack. places.

Attacker Powers

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

Recommendations

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

Distributed Denial of Services Attacks Diagram

```

Goal: Stop the system availability and
communication resources
OR
|- 1. Block mobile communication services
|- 2. Compromise machines to support in flooding
| AND
|   |- 1. Flood by a large number of requests
|   | OR
|   |   |- 1. Launch Application DDoS Attack
|   |   | OR
|   |   |   |- 1. Back
|   |   |   |- 2. RUDY
|   |   |   |- 3. Slowloris
|   |   |- 2. Launch Protocol DDoS attack
|   |   | OR
|   |   |   |- 1. ARP Poison
|   |   |   |- 2. Land
|   |   |   |- 3. Neptune (SYN Flood)
|   |   |   |- 4. Ping of Death
|   |   |   |- 5. Teardrop
|   |   |- 3. Launch Volume DDOS attack
|   |   | OR
|   |   |   |- 1. Smurf (ICMP Flood)
|   |   |   |- 2. UDP Storm
|   |   |   |- 3. Peer-to-peer
|   |   |- 4. Physical destruction or alteration of
components
|   | AND
|   |   |- 1. Permanent DDoS

```

Domain Name Server 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

In DNS reflection attacks, attackers send DNS requests toward multiple open DNS servers with spoofed source address of the target, which results in a large number of DNS responses to the target from DNS servers. Since the cloud has its own DNS servers to answer DNS queries from hosted tenants, there should not be any DNS responses from the Internet to the cloud. Therefore, any activity of inbound DNS responses may signify a potential DNS reflection attack.

Inbound DNS reflection attacks often come from up to 6K distinct sources (with 1500 byte full-size packets). We only observed outbound DNS responses from a single VIP hosting a DNS server at 5666 packets per second for a couple of days repeatedly.

Attacker Powers

- Access confidential information from legitimate/authorized users; * Perpetrate other types of attacks like DDoS and Man-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.

DNS Attacks Diagram

Reused IP Address Attacks

IP address is reassigned and reused by other customer. The address still exists in the DNS cache, it violating the privacy of the original user.

Definition

Each node of a network has an IP address which is allocated to a particular user when that user leaves the network, the IP address associated with him is assigned to a new user. The chances of accessing previous user data by the new user exist as the address still exist in DNS cache and hence the data belonging to one person can be accessed by another.

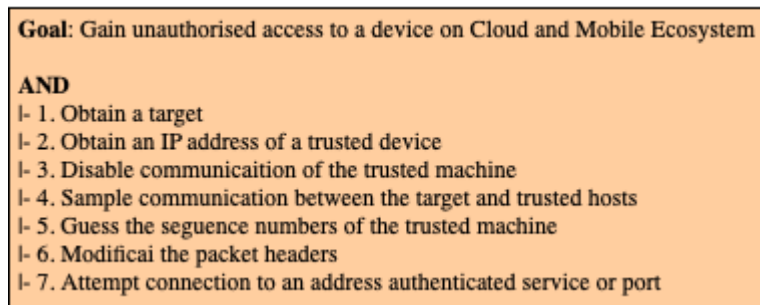
Attacker Powers

- Access confidential information from legitimate/authorized users.

Recommendations

To ensure that the mobile application is resilient or immune to malicious Reused IP Address 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.

Reused IP Address 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.

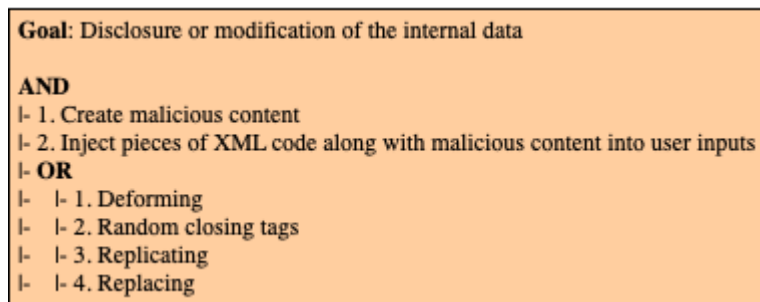
Attacker Powers

- 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



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.

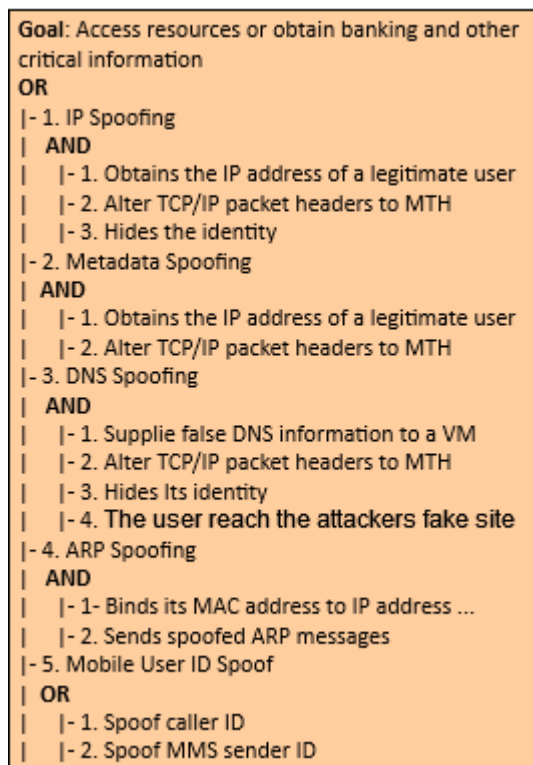
Attacker Powers

- 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.

Spoofing Attacks Diagram



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.

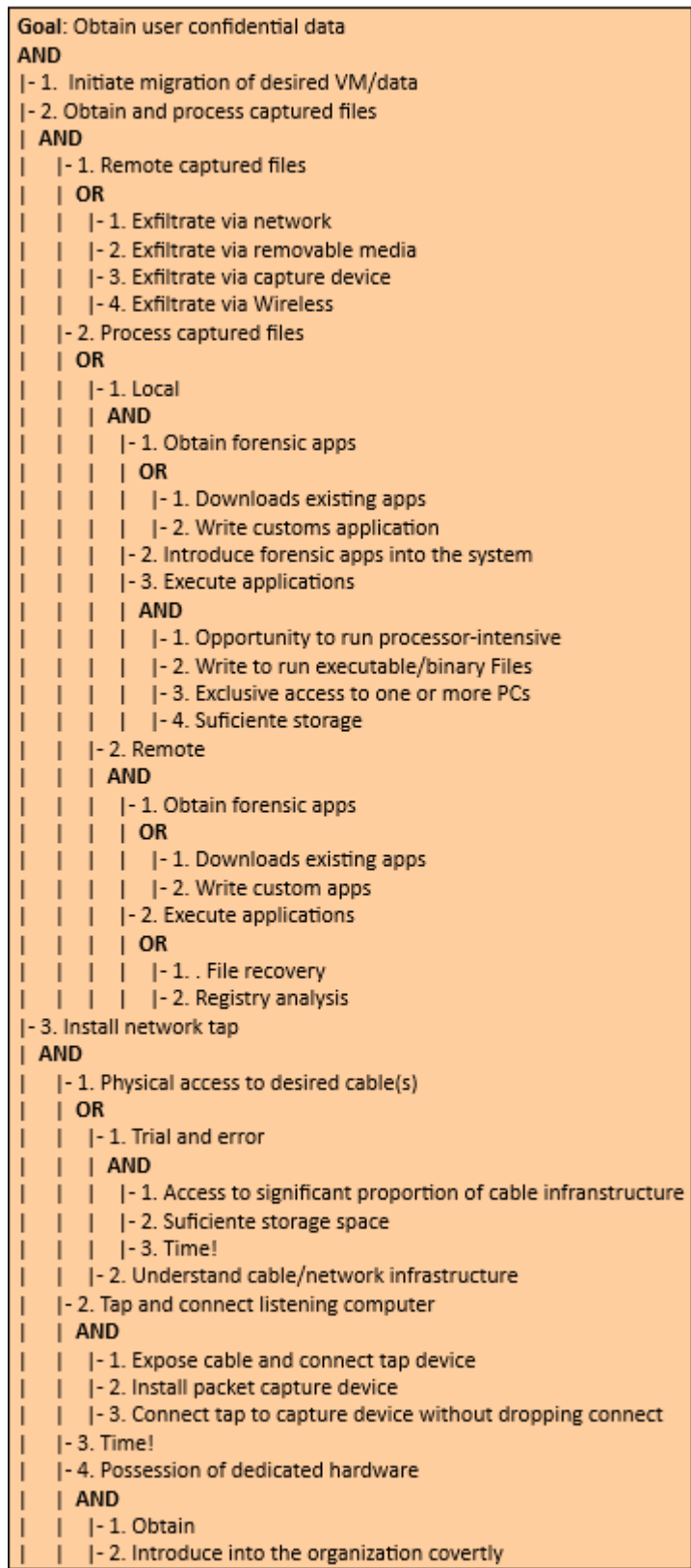
Attacker Powers

- 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.

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.

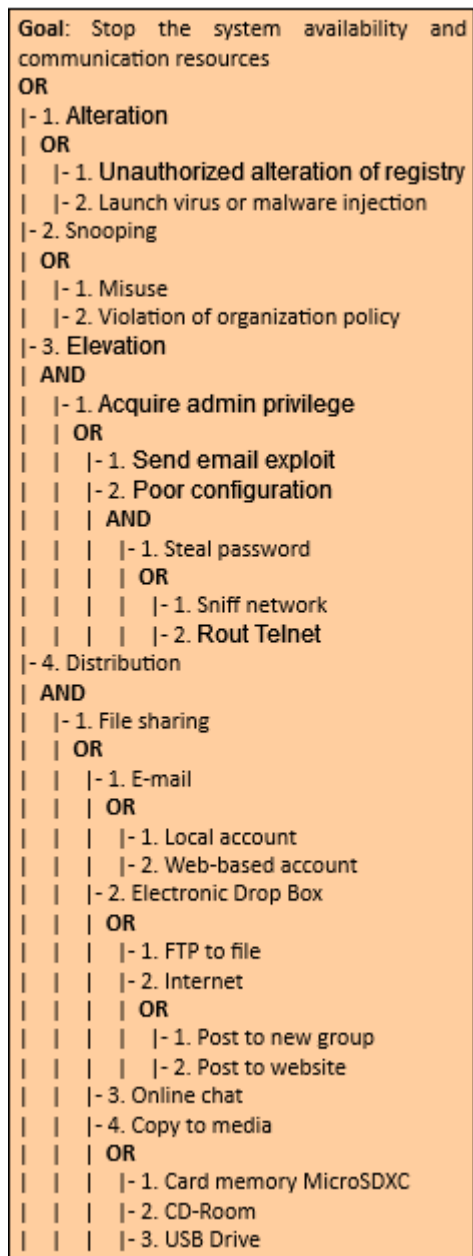
Attacker Powers

- 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.

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.

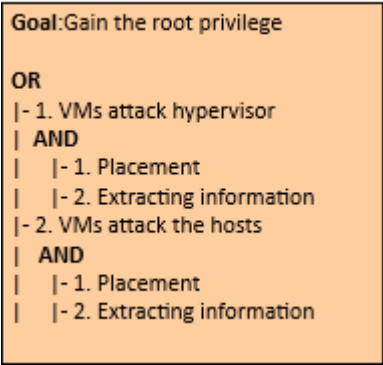
Attacker Powers

- 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



Cross VM Attacks (Side channel attacks)

Side-channel attacks are used to extract cryptographic keys from a victim device or process in a virtualized layer of the cloud ecosystem where a Cross-VM attack exploits the nature of multi-tenancy, which enables that VMs belonging to different customers may co-reside on the same physical machine.

Definition

The side-channel attack takes advantage of low-bandwidth message channels in a system to leak sensitive security information. There is no doubt that this type of attack exists and is real for today's computer systems, including modern smartphones and tablets. Here we highlight the cache-based side-channel attacks that have been used to steal cryptographic information from a single OS. Furthermore, the weak link is in the fact that cryptographic algorithms usually have data-dependent memory access patterns, giving the possibility of being revealed by the observation and statistical analysis of hits / errors from the associated cache. Recent research has shown attackers can build up cross-VM side channels to obtain sensitive information. However, currently these channels are mostly based on shared CPU cache, networks, CPU loads and so on. These attacks are generally categorized into one of three classes:

- Time-driven side-channel attack;
- Trace-driven side-channel attacks;
- Access-driven side-channel attacks.

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.

Cross VM Attacks Diagram

Goal: Gain access over another VM running on the same hypervisor

OR

- | - 1. Time-driven side-channel attack
- | **AND**
 - | | - 1. Exploit hypervisor vulnerability
 - | | - 2. Access another VM
 - | | - 3. Extract confidential information, file and documents
- | - 2. Trace-driven side-channel attack
- | **AND**
 - | | - 1. Exploit hypervisor vulnerability
 - | | - 2. Access another VM
 - | | - 3. Extract confidential information, file and documents
- | - 3. Access-driven side-channel attack
- | **AND**
 - | | - 1. Exploit hypervisor vulnerability
 - | | - 2. Access another VM
 - | | - 3. Extract confidential information, file and documents

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.

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

Goal: To compromise the system or system component

OR

- | - 1. Penetration
- | - 2. Monitoring
- | - 3. Manipulation
- | - 4. Modification
- | - 5. Substitution

Final Security Test Specification and Tools Report

Mobile Platform	Web Application
Application domain type	m-Health
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Factors-based authentication
Has DB	Yes
Type of data storage	SQL
Which DB	MySQL
Type of data stored	Personal Information ; Confidential Data ; Critical Data
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	HTML5 ; PHP ; Javascript
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	Private Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Wi-Fi ; GPS
Data Center Physical Access	Yes

In order to avoid or prevent *SQLi, XSS, Spoofing and CSRF attacks*, the following security tests should be performed.

Test Parameter	Testing Types	Testing Methods	Tools
			Both Android iOS
Authentication and Authorization	Grey Box	Dinamic analysis via Vulnerability Scanner	OWASP WebScarab, OWASP Berretta, Nikto, Wikto, Paros Proxy, Spike Proxy, EOR, Pantera
Access Control	Grey Box	Dinamic Analysis via Penetration Test	NMAP and Kali Linux

In order to avoid or prevent *Botnet, Spoofing and Sniffing attacks*, the following security tests should be performed.

Test Parameter	Testing Types	Testing Methods	Tools
			Both Android iOS
Exploit Database Vulnerabilities	White Box	Manual Dinamic Analysis via Penetration Test	SQLite browser Xcode
Proper SSL usage and Use of encryption	Black Box	Dinamic Analysis via Proxies	WebScarab
Database frangibility scanner	Grey Box	Dinamic Analysis via Vulnerability Scanner	Database Scanner of Internet Security Systems Co. and MetaCortex
Find Bugs	White Box	Static Analysis via Bytecode Scanner	FindBugs, BugScan of LogicLab Co.
	White Box	Static Analysis via source code Analyser	C++Test, RATS, C Code Analyzer(CCA)
	White Box	Static Analysis via Binary code Scanner	BugScan of Logi- cLab Co. and Fx- Cop;BugScam
Input validation of user SID	Grey Box	Manual Dinamic Analysis Checking input fields in GUI	
Runtime manipulation: code injection, patching	Grey Box	Static Analysis via Test Penetration	Cydia Substrate Cypcript

In order to avoid or prevent *Malicious Insider and VM-Migration attacks*, the following security tests should be performed.

Test Parameter	Testing Types	Testing Methods	Tools
			Both Android iOS
Input validation	Grey Box	Static Analysis via Forensic Mobile	Slueth Kit+Autopsy Browser AndroGuard, Drozer, apktool, Amandroid

In order to avoid or prevent *Malware injection and Side-channel Attacks*, the following security tests should be performed.

Test Parameter	Testing Types	Testing Methods	Tools
			Both Android iOS

Debug flag	White Box	Static Analysis via Forensic Mobile	BlackBag Blacklight, Encase forensics	AndroGuard, Drozer, FindBugs, Andriiller	
Content providers	White Box	Static Analysis via Forensic Mobile	Slueth Kit+Autopsy Browser	AndroGuard, Drozer, apktool	
Code quality	White Box	Static Analysis via Byte-code Scanner	FindBugs, BugScan of LogicLab Co.		
	White Box	Static Analysis via source code Analyser	C++Test, RATS, C Code Analyzer(CCA)		
	White Box	Static Analysis via source code Analyser	BugScan of LogicLab Co. and Fx- Cop, BugScam		class-dump-z

In order to avoid or prevent *physical attacks*, the following security tests should be performed.

Test Parameter	Test Approach	Test Method	Both	Tools	
Debug flag, Content providers, Code quality	White Box	Static Analysis via Forensic Mobile		Android AndroGuard, Drozer, FindBugs	iOS
Leak, Breach and data Loss	Black Box	Manual Dinamic Analysis Checking input fields from device and GUI			