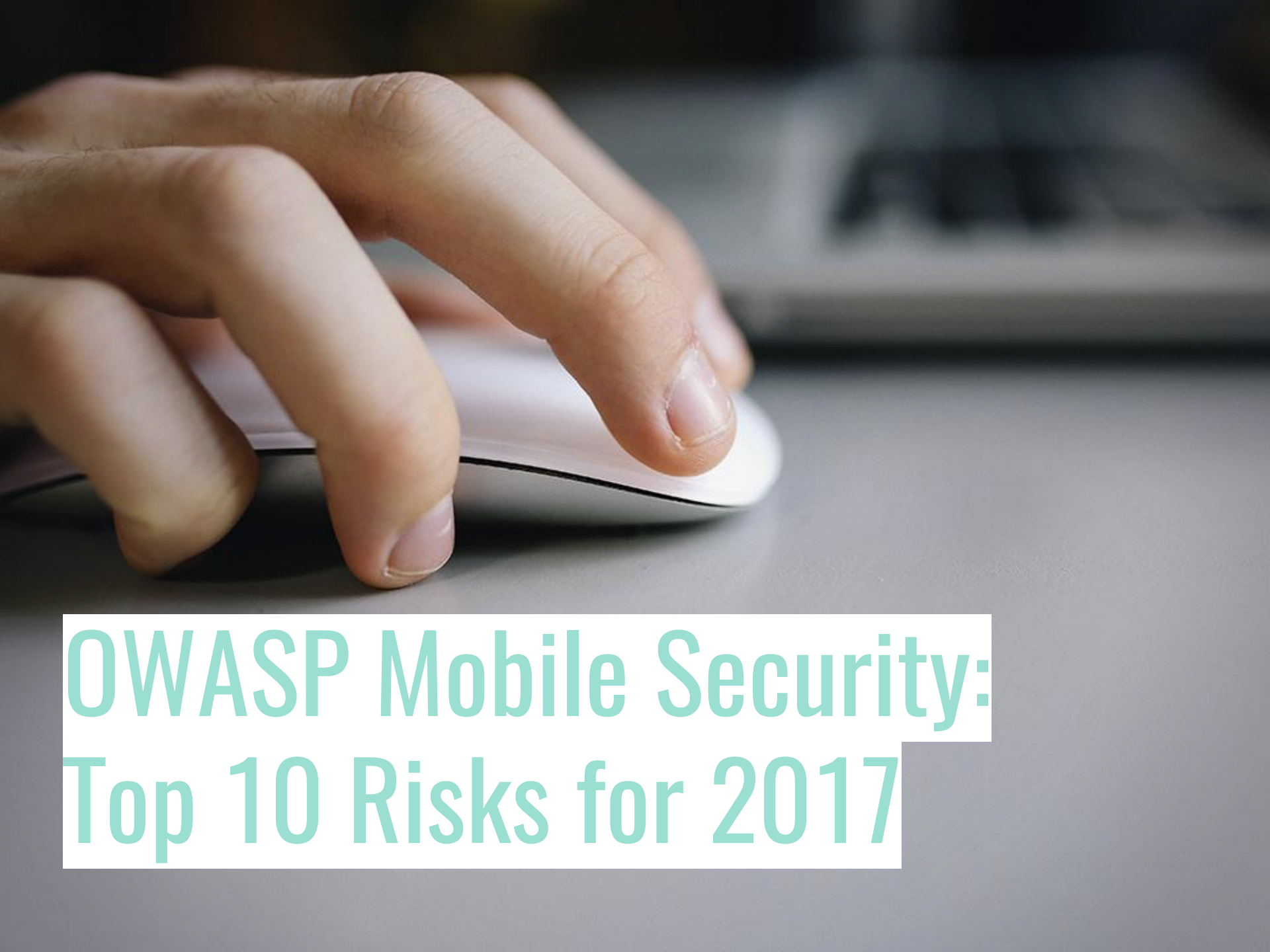


OWASP Mobile Security: Top 10 Risks for 2017

Security in the mobile development field

...is important as never before.

There is a wild range of various threats that we don't even think about.

It's already an established fact that every developer should follow the OWASP research about the most common vulnerabilities and attacks regarding mobile and web products.

OWASP

Open Web Application Security Project

OWASP Mobile Security Project

It gives the mobile developers and security teams all resources needed to create secure mobile products.

Started in 2010, it's already proven worthiness, by building a mobile threat model and classifying mobile OWASP top 10 security risks.

Project is still in progress

Every year there is a new research and improvement to the list of the main threats to keep up with the industry trends and innovations.

Security is like virginity: you're either a virgin or you're not. You either have security or you don't.

-- Lennart Meri

OWASP Mobile Top 10 Risks: buckle up for a long ride

The one and only purpose of this list is to level up the mobile security and make it understandable for any freelancer or development company.

Last obtained results showed that some of the initial threats have been, in fact, almost beaten by talented developers, QA and security engineers.

The technological progress isn't stopping, therefore, in 2016 some new risks were discovered.

Top 10

OWASP Mobile Top 10 Risks

Find the full-text on our blog
https://tecsynt.com/blog/development/owasp-mobile-security-top-10-risks-for-2017

1

Improper Platform Usage

[established in 2016]

Improper Platform Usage

We’re dealing with the misuse of a platform feature or failure to use platform security controls.

Android internal storage, platform permissions, misuse of the TouchID, the iOS Keychain – all of that and more. It can be mobile operating system’s any security control.

What can be done as a preventive action? Be careful with the usage of a Keychain and Android intents.

Insecure Data Storage

2

Insecure Data Storage

This is a combination of insecure data storage and unintended data leakage, which applies to locally stored data along with cloud synced.

The impact:

- Loss of the data confidentiality
- Disclosure of the credentials
- Privacy violations
- Non-compliance

Preventive actions

Identify and store only necessary data on the mobile device, and avoid public storage areas and query strings in sensitive data. Protect your data storage and password credentials by using containers and encrypted APIs.

3

Insecure Communication

Insecure Communication

We are talking about poor handshaking (tokens sent during Wi-Fi connection), incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.

Preventive actions

You have to ensure that all of the valuable data that is leaving the device is encrypted. So, always validate SSL/TLS certificates and implement a secure network transmission of the sensitive data.

Read more in our posts

https://tecsynt.com/blog/development/charles-proxy-tutorial-for-mobile-developers-and-testers
https://tecsynt.com/blog/development/how-bugs-in-the-api-can-tell-you-users-credentials

Insecure Authentication

4

Insecure Authentification

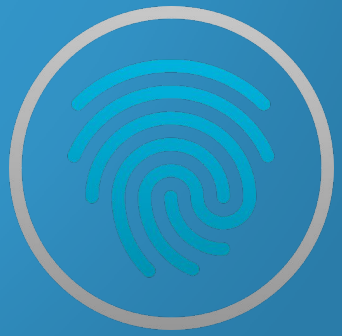
This category represents poor authentication protocols, bad session management and issues with compromised tokens. This means failures to identify the user and to maintain his identity when it is required. All of it leads to a data theft or third-side tampering with it.

Preventive actions

Use a native keychain of the device or your own encrypted database for sensitive data storing. And if the app doesn’t need an offline access then just disable it.

Read more in

“Token Based Authentication: Tips for App Security”
https://tecsynt.com/blog/development/token-based-authentication-how-to-improve-your-app-security



Insecure Authorization

5

Insecure Authorization

The difference between 5th and 4th points is that this category includes server's failures in authorization process like improper identity and permissions enforcing, authorization decisions on the client side and forced browsing. This usually ends in bad communication between the app and a back-end third-party.

Preventive actions

Here it's recommended to ensure proper Server-Side SSL and web server configuration.

Insufficient Cryptography

6

Insufficient Cryptography

The code must apply cryptography to a sensitive data, but this category represents issues when cryptography was attempted but wasn't done properly. The access to a sensitive information can be gained by some adversary due to insufficient data protection if the vulnerabilities impact the process of encryption/decryption or if the algorithm behind encryption/decryption is weak in nature.

Preventive actions

Learn how to use the app platforms' advantages like a native keychain to keep there all the sensitive information.

Client Code Quality

[established in 2016]

7

Client Code Quality

It used to be called "Security Decisions Via Untrusted Inputs", which combined all code-level implementation problems in the mobile client, like buffer overflows, format string vulnerabilities, and other code-related mistakes that allow rewriting mobile device's code. As was proved by OWASP, bad quality of the code can allow hackers to exploit a business logic and bypass security controls implemented on the device.

Preventive actions

Three steps, actually, you should use carefully chosen logic, better not a simple one. Always test third-party libraries and validate client's input.

Code Tampering

[established in 2016]

8

Code Tampering

Talking about binary patching, method hooking and swizzling, local resource modification, and dynamic memory modification. An enemy can directly modify the code, change the contents of memory dynamically or replace the application’s APIs, along with modifying the app's data and resources.

Preventive actions

Mobile developers and security engineers must learn and implement anti-tamper and tamper-detection techniques.

9

Reverse Engineering

[established in 2016]

Reverse Engineering

This means attacks on the final core binary to get a grip on the source code, cryptographic constants & ciphers, libraries, algorithms, back-end servers, etc. There are plenty of tools to do so very easily for even newbie hackers.

Preventive actions

Binary protection means making your code as complex as possible. Always store your code in a secure environment and take care of the correct use of jailbreak detection, and certificate pinning.

10

Extraneous Functionality

[established in 2016]

Extraneous Functionality

Mobile developers often include hidden backdoors in the app's functionality or some internal security controls that are needed during a development process. And when such actions are accidentally released into the bad hands... adversaries will, without a doubt, take a chance to tamper with / compromise an app.

Preventive actions

Be careful with the debugger detection controls and pay attention while you manage debugging logs.

The first and prior responsibility

to ensure the security

of your product to keep your customers safe

Top 10 risks

The key focus areas can be easily identified for 2017:

- Improper sensitive app's data storage & Unintended data leakage
- Weak encryption algorithms & Insufficient cryptography
- Weak encryption algorithms & Insufficient cryptography
- Code Protection from Extraneous tampering
- Lack of Binary protection (Reverse engineering attacks)

TECSYNT

Mobile Development Company

GET IN TOUCH

hello@tecsynt.com
tecsynt.com

THANKS

Kateryna Ganiukova, Marketing Manager
Kateryna Lysak, PM