VERSION 1.1

eShard

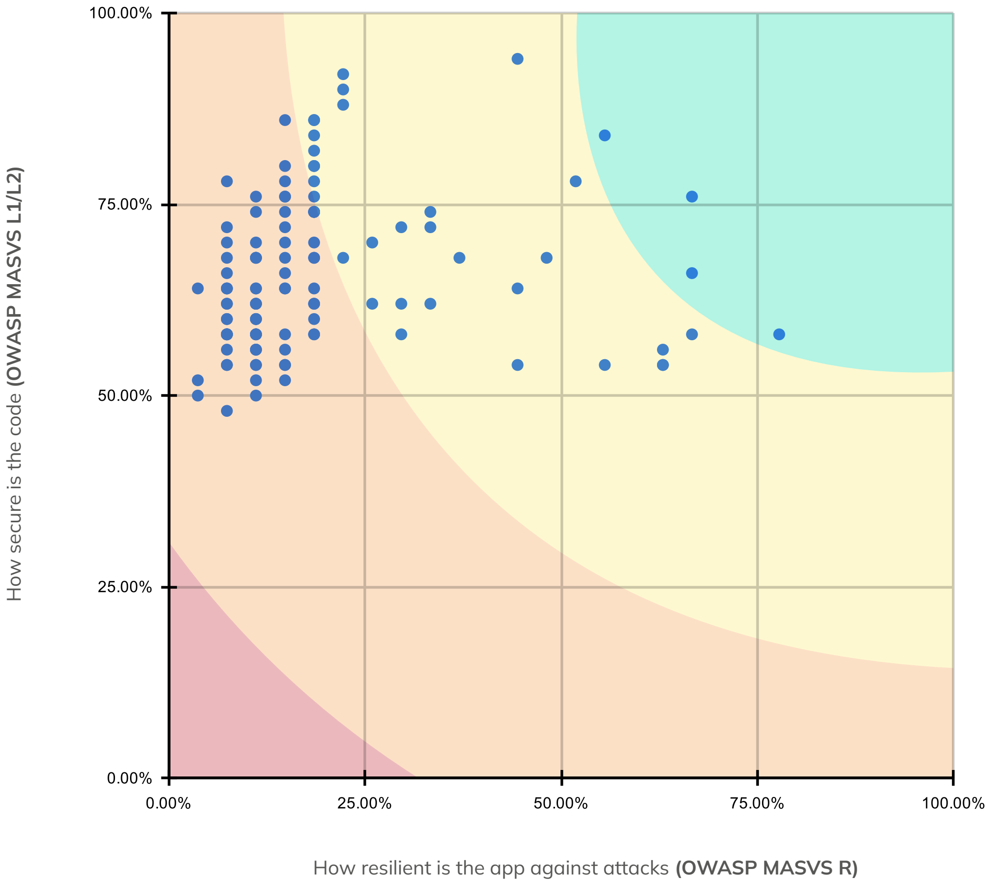
European Mobile Banking Apps Security Benchmark

WHITE PAPER

2022 EDITION

GLOBAL RESULTS OVER 100+ APPS

OWASP MASVS Coverage



In this comprehensive security report on the European Mobile Banking apps, two major trends can be observed:

> The apps have started their security strategy by securing the code first. All apps pass **more than 50%** of the tests in this category

> On the other hand, the protection implementation to prevent reverse engineering is not a global trend at all. Only 15 apps score **higher than 50%** in this test category

We have identified 4 clusters:

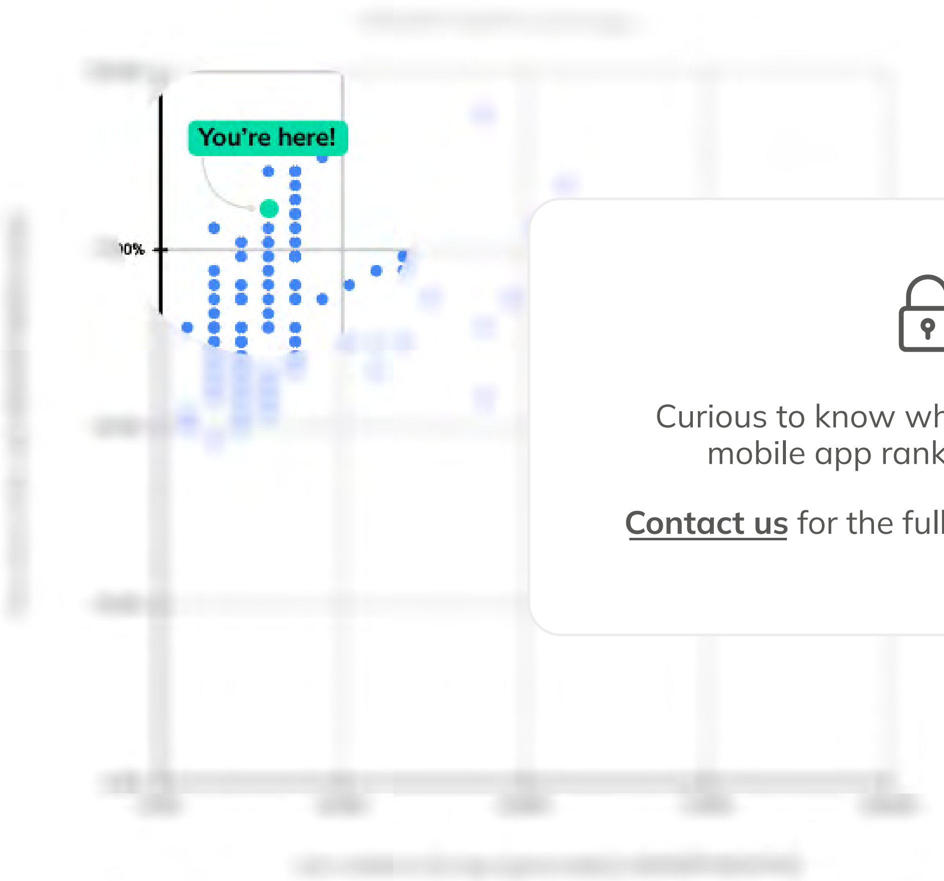
> **The Leaders**: These apps are the industry models, following best practices in the code and implementing hardening protections against attack techniques. Best protections against hacking can be found.

> **The Contenders**: These apps display partial protections, which does not suffice to meet the standards and withstand attack techniques.

> **The Followers**: These apps are far from reaching a good security level. Security strengthening should be focused at the code level and the resiliency layer, where protection barely existed in this cohort.

> **The Non-Mature**: These apps have been developed without any security at the core. Attackers can easily stress the banking service as operated by the bank.

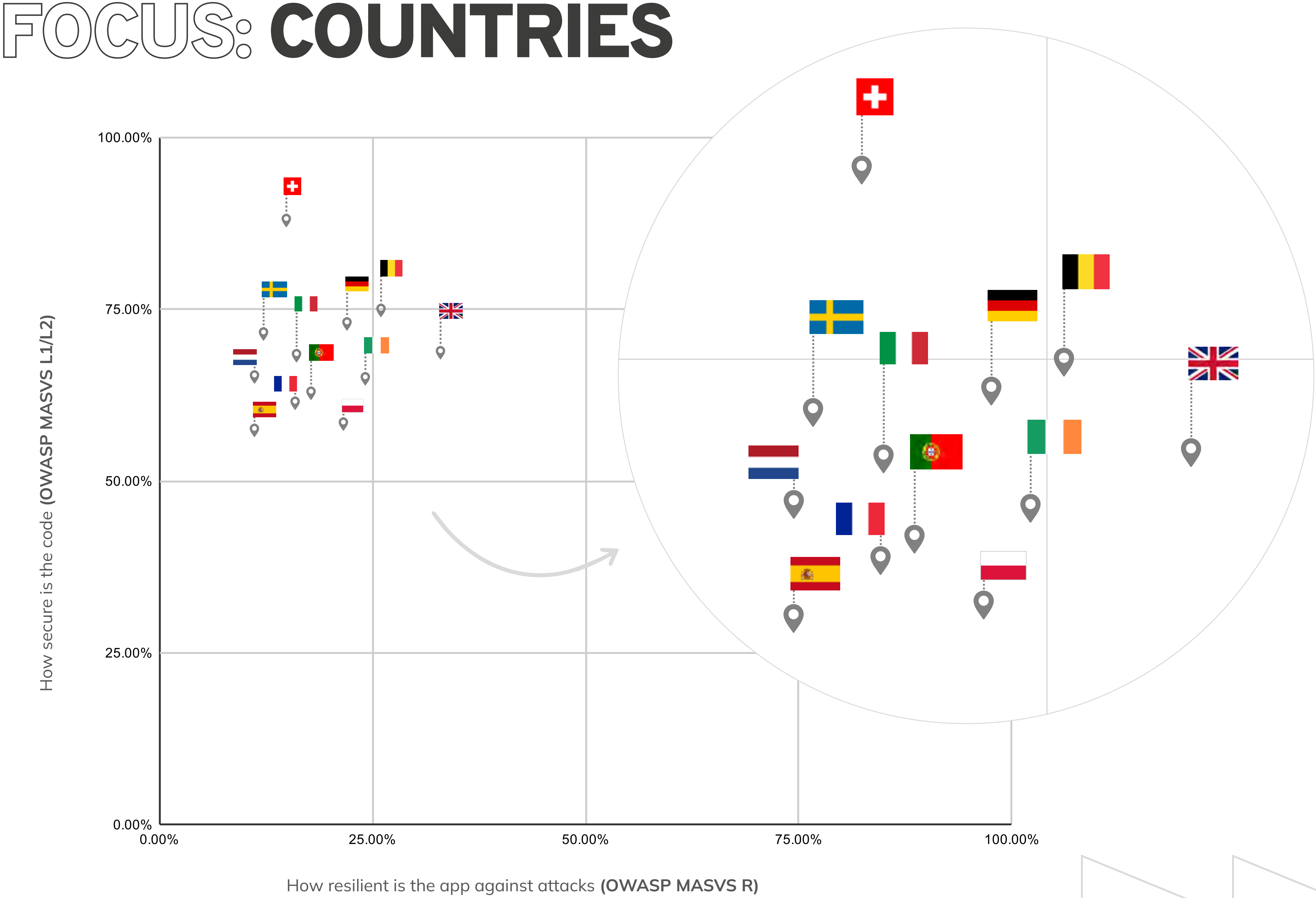
RESULTS: YOUR APP



Curious to know where your banking mobile app rank in this study?

Contact us for the full customised report.

FOCUS: COUNTRIES

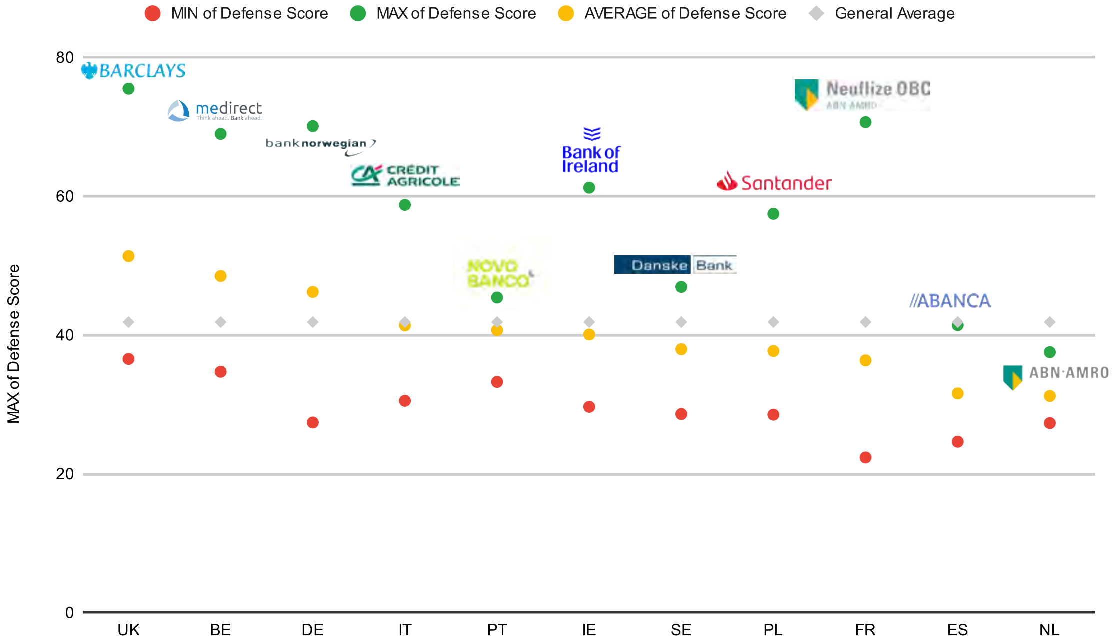


MATURITY GAP BY COUNTRY

We have calculated the Defense Score for each application, which is a synthesis of all static and dynamic tests run on the apps. This score can range from 0 to 100, 100 being the highest rank an app could perform. We have then calculated the General Average (42) as well as a country-based Average.

Thanks to the latter, we can draw some trends which lead to important discrepancies between the top-performing countries and the others.

MAX of Defense Score vs. MIN of Defense Score



POTENTIAL CONSEQUENCES

The European market is globally not mature. Even though some local actors are pretty advanced, the gaps are quite wide in some countries and across the entire continent.

The **UK** is the country where the Average Defense Score is the highest (51) and **the Netherlands** have the lowest (31).

We can also note that the maturity within countries can significantly vary, more specifically in 2 countries: Germany and France.

In **Germany**, we have identified top performers (2 German Banks in the Overall Top 10), but also some very not mature actors with 9 Banks with a Defense Score far behind the local trend, some performing below the 30. The gap in Germany is one of the widest, with a 43-point difference between the best and worst banks.

In **France**, despite performing quite poorly with an Average Defense Score of 36 (8 points below the General Average), we have identified one actor exceeding the 70-point bar. Here again, the difference between the 2 extremes is wide (48 points of difference).

EXECUTIVE SUMMARY

Mobile application security has long been neglected. For many, this was simply not critical, because cybersecurity verifications and monitoring were managed at the back-end level, and an attack targeting the mobile app would be limited in impact.

Such assertion was common in the mobile banking industry. The mobile app was considered as a mirror of the web app... and the traffic on mobile apps remained low.

Time has changed. Mindsets become more mature. Recent regulations (PSD2) in Europe pushed bank organizations to rethink their mobile banking application. The mobile banking application is no longer mimicking the web app, it provides new services. The most notable one is sure the eCommerce transactions validation leveraging the strong customer authentication.

Does that mean that the mobile banking application was put high in the agenda for security officers? Not really, but things are moving.

The testing of **100+** mobile banking applications in Europe shows that the banking ecosystem is migrating to more security - and some banks run the show.

Additionally, securing mobile apps is proven to be critical to protect the integrity and reputation of a business, as well as the data and privacy of end users. Implementing security measures is no longer a practice of displaying thought leadership, nor a nice-to-have value added initiative, but a risk management mindset that should be embedded into the operations of all mobile banking institutions.

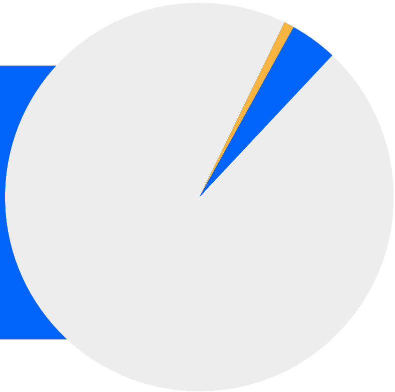
Here are some facts:

10% of the tested apps meet OWASP L2 requirements in cryptography. **90% have weak cryptography.**

4 🏆 out of 120 apps are **leaders**, as they show few vulnerabilities and are somewhat resilient to attacks.

ZERO app meet OWASP L2+R, even though this is a MASVS recommendation.

🇬🇧 UK is the most mature country.



OWASP L2+R is the strictest security verification level (L2) plus a set of resiliency requirements adaptable to an app-specific threat model (R).

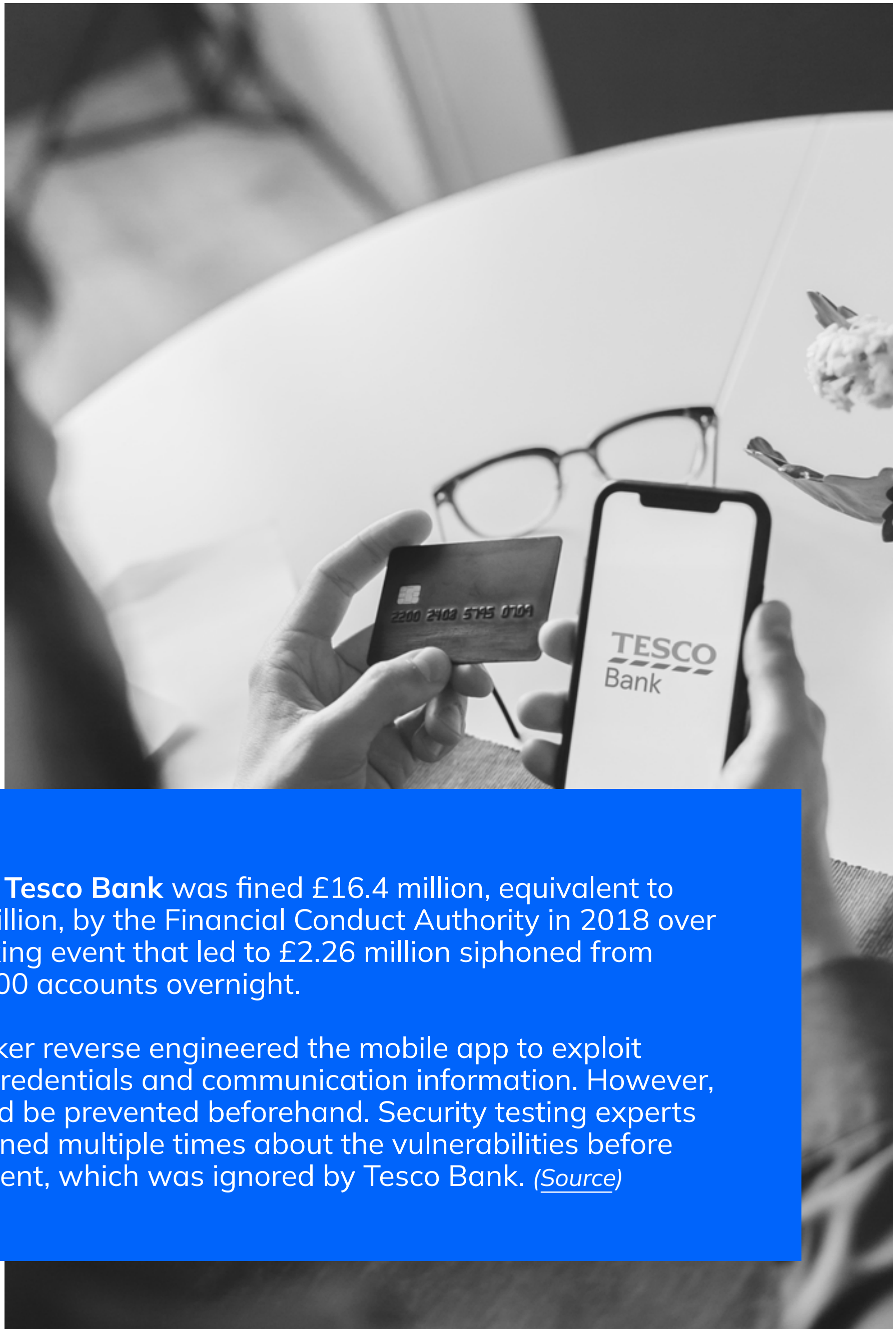
MOBILE BANKING APPLICATIONS

What is at stake?

PSD2 PSD2 was introduced along with the growing prevalence of mobile banking. eCommerce transactions leveraging Strong Customer Authentication, have become the focus.

60%+ Clients going mobile is an ongoing mega trend in banking services. More than 60% of European banks have recognized that mobile banking is a must-have strategic asset.

⚠ Evidence* has shown that banking apps can be an entry point for hackers. Mobile apps can no longer be excluded from the security framework of banks.



*

Britain's **Tesco Bank** was fined £16.4 million, equivalent to €18.4 million, by the Financial Conduct Authority in 2018 over the hacking event that led to £2.26 million siphoned from over 9,000 accounts overnight.

The hacker reverse engineered the mobile app to exploit clients' credentials and communication information. However, this could be prevented beforehand. Security testing experts had warned multiple times about the vulnerabilities before the incident, which was ignored by Tesco Bank. *(Source)*

What could an attacker attempt to do by **targeting a mobile banking app**?

Seek for vulnerabilities in the system

Mobile apps communicate with the back-end system through API. Apps without protection leave hackers easy access to API and back-end.

Change app behaviour, get private data

Man in the middle to change the app behavior or observe data when being used by the client.

Extract secret data or malevolent access

Malevolent access to data or operations belonging to a client (e.g., PIN code)

Get private information

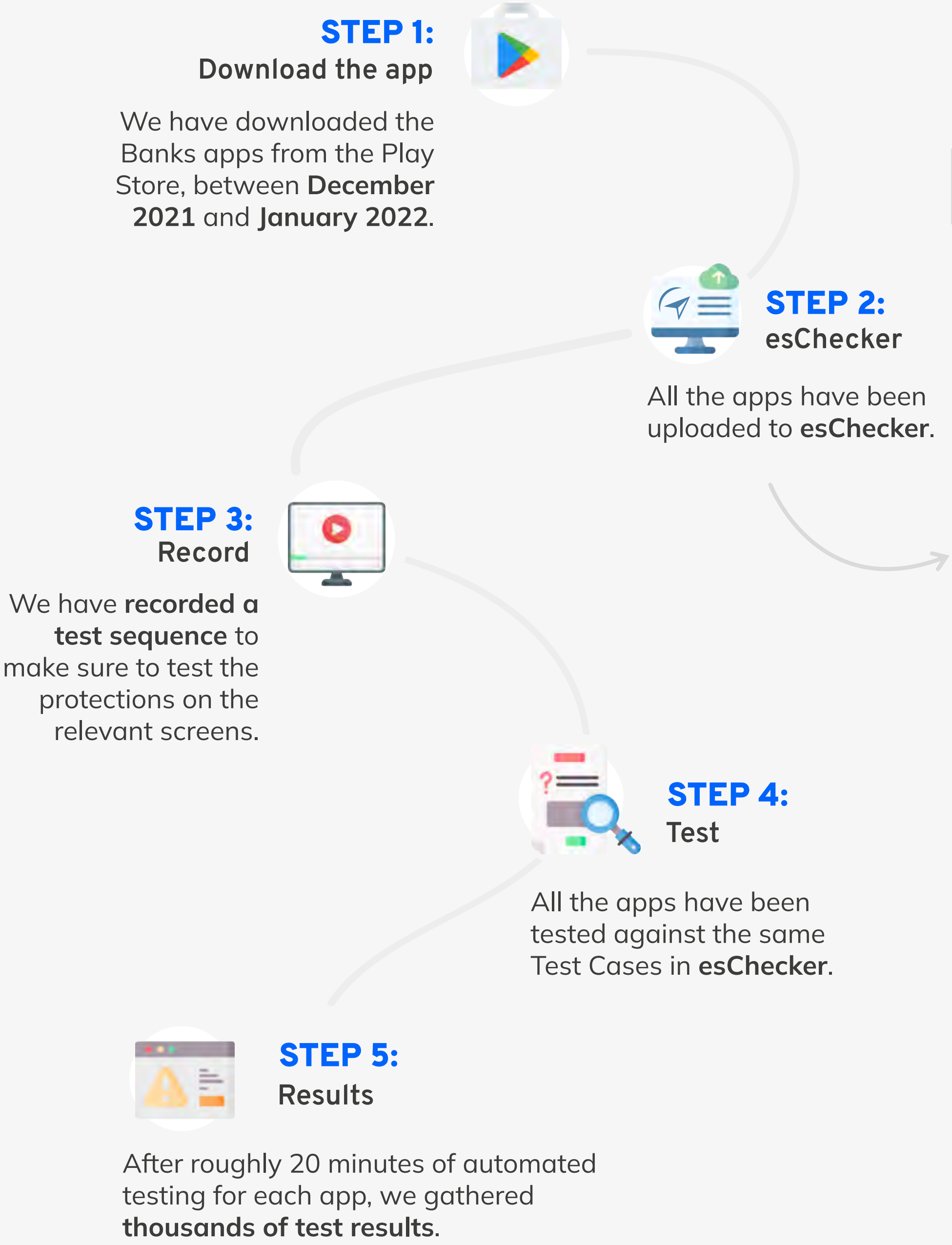
Sensitive and personal data, like bank statements, are the prime target of information theft.

Compromise critical features

Such as 2FA, which may severely impact the security of the transactions and generate fraud.

Reputation damage usually represents a far more impactful consequence that requires heavy resources to repair. Once the client's trust in banks is breached, it could be a definitive loss of customers.

MODUS OPERANDI



STEP 1:
Download the app

We have downloaded the Banks apps from the Play Store, between **December 2021** and **January 2022**.

STEP 2:
esChecker

All the apps have been uploaded to **esChecker**.

STEP 3:
Record

We have **recorded a test sequence** to make sure to test the protections on the relevant screens.

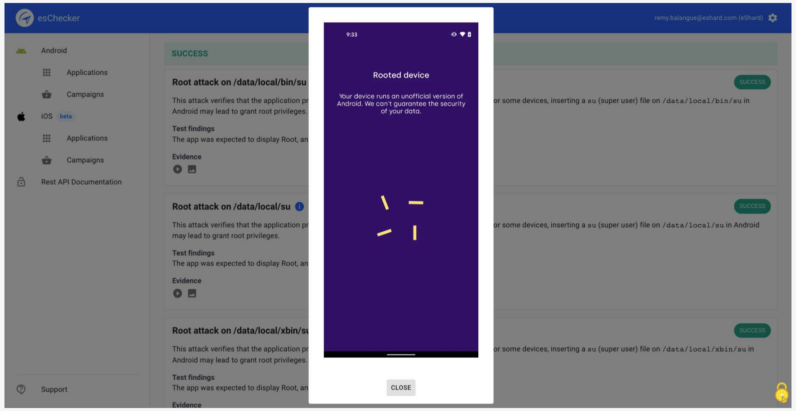
STEP 4:
Test

All the apps have been tested against the same Test Cases in **esChecker**.

STEP 5:
Results

After roughly 20 minutes of automated testing for each app, we gathered **thousands of test results**.

The workflow diagram on the left is to demonstrate how the tests were carried out with our tool, esChecker.



Root attack on /data/local/bin/su

This attack verifies that the application pr … or some devices, inserting a su (super user) file on /data/local/bin/su in Android may lead to grant root privileges.

Test findings
The app was expected to display Root, an

Evidence

Root attack on /data/local/su

This attack verifies that the application pr … or some devices, inserting a su (super user) file on /data/local/su in Android may lead to grant root privileges.

Test findings
The app was expected to display Root, an

Evidence

Root attack on /data/local/xbin/su

This attack verifies that the application pr … or some devices, inserting a su (super user) file on /data/local/xbin/su in Android may lead to grant root privileges.

Test findings
The app was expected to display Root, an

Evidence

Rooted device

Your device runs an unofficial version of Android. We can't guarantee the security of your data.

esChecker MAST (Mobile Application Security Testing) performs automated testing to validate app behaviours facing an intrusion with advanced reverse engineering testings. All tests involved in this report were run on esChecker.

Binary of the latest Android app (apk) were used as input. All tests were automated taking an average time of approximately 30 mns per app. The automated tests leveraged the record and replay feature. The recording sequence ran the application until the login page. When required, a specific success criteria was set for a given app. This was to avoid any false positive. Any security protection implemented after the login page was not part of the scope of the study.

Analysis Methodology

PROTOCOL

All applications have been tested against the same test protocol.

Application Reverse Engineering Protection: 33 tests, to check for:
> Runtime Code Instrumentation Protections (12)
> Runtime Environment Verification (17)
> Static Binary Protections (4)

Secure Coding: 61 tests to check for:
> Application Misconfiguration (51)
> Application Vulnerability (10)

DYNAMIC TEST SEQUENCE

All Dynamic Testing have been done on real Google Pixel 4A with the following user journey:

1. Go to the login screen
2. Enter the login
3. Trigger the attack
4. Enter the password
5. Wait for the app response, and stop the recording
6. Save the recording

DYNAMIC TEST PROTECTIONS

For each dynamic test, we try different methods to see if some sort of protection is triggered:

> Does the application crash?

> Does the application send a warning message?

> Does the application block the usage of the tool that is being used to attack the app?

TEST RESULTS

A test can end up in one of these 3 statuses:

✓ **Success**: in dynamic tests, the application reacts as predefined criteria. In static tests, esChecker succeeds in finding the embedded protection.

✕ **Failure:** the application doesn't respond properly (dynamic testing) or the test hasn't found what it was looking for (static testing)

! **Action Required:** it's impossible to automatically set a Success / Failure state, an analyst conclusion is required.

DISCLAIMER

STATIC TESTS AND OBFUSCATION

When running Static Tests, we decompile the app and search for certain pieces of code in the source code. Sometimes, thanks to a strong obfuscation method, this code can't be found.

However, it doesn't mean that it's not used, it's simply harder or impossible to find automatically.

DYNAMIC TESTS AND RUNTIME PROTECTIONS

The recording sequence encompassed the application execution from the start to the login page. We did not use any specific login accounts. Therefore, the testing did not cover security controls implemented after the login page.

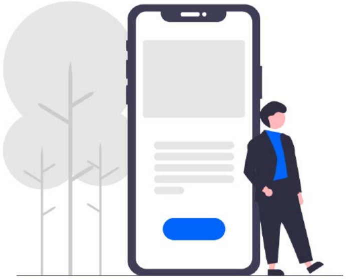
AVOID FALSE POSITIVE

Tune your success criteria against app reaction.

When a threat is detected, the app can trigger several types of reactions.

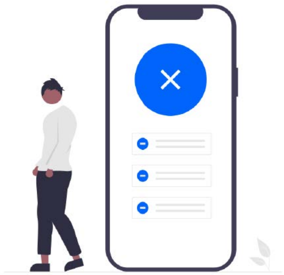
The purpose of these protections is to prevent the potential unsafe actions from being done at the exact time they are detected before the hacker is able to harm the application.

There are several types of protections that are possible, all of them being configurable in esChecker to avoid false positives:



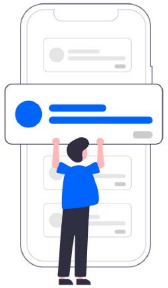
No reaction

The app does nothing and we can use it without being informed of the potential danger.



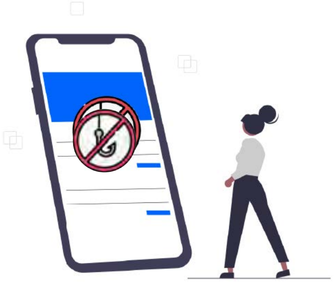
The app crashes

The app crashes to prevent any type of unexpected actions from the phone, or from a 3rd-party malware application.



The app informs the user

The app throws a warning message to the user, making the application unusable.



The app blocks the unsafe 3rd-party tool

The app blocks the 3rd-party tool used to hook the app, allowing the user to keep using the app, without them noticing the attack attempt.

Thanks to its unique "**Attack Video Replay**" feature, **esChecker** returns a video proof of the reaction of the apps when being attacked. It increases your confidence in the test results, no false positive possible.

Threats on mobile apps

When considering security, we have to assume that mobile applications can be executed on different platforms:

- An non-modified mobile device
- A mobile device with high privileges. (In other words, it is rooted or jailbroken)
- A mobile device with the framework of reverse engineering
- A computer with emulation capabilities

The cost of these environments remains affordable. Many analysis frameworks are open source and do not represent an obstacle. In short, without dedicated in-app protections, the mobile application is exposed to attacks. It may result in the research and exploitation of a vulnerability. The mobile application being part of a system, it may be used as an entry point to compromise the system.

What are OWASP Mobile Top 10?

They represent the channels of threats that a mobile application may face.

M1 Improper Platform Usage

M2 Insecure Data Storage

M3 Insecure Communication

M4 Insecure Authentication

M5 Insufficient Cryptography

M6 Insecure Authorization

M7 Client Code Quality

M8 Code Tampering

M9 Reverse Engineering

M10 Extraneous Functionality

OWASP

To help organizations efficiently develop and secure their mobile apps, the OWASP (Open Web Application Security Project®) has put together highly valuable resources:

> The **OWASP Mobile Top 10**: gathers the most critical security risks encountered on mobile applications. This list helps you identify the top priority risks you must be protected against.

> The **Mobile Application Security Verification Standards (MASVS)**: describes 4 levels of verification standards that help you quantify your level of compliance against the OWASP. This score is a good way for you to measure your progress over time and to communicate both internally and to external third parties.

> The **Mobile Security Testing Guide** (MSTG) is a set of test cases to be performed in order to evaluate your MASVS compliance score. Your Security Policy starts from here.

> “The **OWASP Mobile Top 10** was never meant to be a standard. It is a list of risks.
>
> Instead, **MASVS** is intended for reviewers and developers and provides clear requirements and metrics."

Hugues Thiebeauld, CEO of eShard

MASVS

Aiming at covering the **Mobile Top 10**, OWASP has defined a set of 8 requirements for mobile applications. Named Mobile Application Security Verification Standards (**MASVS**), they can be used by developers or experts to make sure that the mobile application does not embed obvious vulnerabilities and integrates a resilience layer.

VERIFICATION STANDARDS

- **V1** Architecture, Design and Threat Modeling Requirements
- **V2** Data Storage and Privacy Requirements
- **V3** Cryptography Requirements
- **V4** Authentication and Session Management
- **V5** Network Communication Requirements
- **V6** Platform Interaction Requirements
- **V7** Code Quality and Build Setting Requirements
- **V8** Resilience Requirements

Because security has a cost, it is necessary to choose the right level according to the mobile application. For financial services, the recommendation is to embrace the **L2** + **R** (Defense-in-depth + Resilience Against Reverse Engineering and Tampering).

In front of each requirement, there is a guideline for the test (MSTG). Some can be automated. This is exactly what was done on esChecker MAST. Our study concluded that:

None of the tested applications are OWASP L2+R compliant.

MSTG

The Mobile Application Security Verification Standard (MASVS) is a standard for mobile app security. Depending on the apps' features and the required level of protection, these standards can be combined as follows:

MASVS-L2+R MASVS-L2 MASVS-L1+R MASVS-L1

R: Resiliency Against Reverse Engineering and Tampering

L2: Defense-in-depth

L1: Standard Security

To help the app testers during the mobile app security testing phase, the OWASP has put together a series of testing cases called the MSTG for Mobile Security Testing Guide, broken down into 8 categories:

MSTG-ARCH	MASVS-L1 & MASVS-L2	Architecture, Design and Threat Modeling Requirements
MSTG-STORAGE	MASVS-L1 & MASVS-L2	Data Storage and Privacy Requirements
MSTG-CRYPTO	MASVS-L1 & MASVS-L2	Cryptography Requirements
MSTG-AUTH	MASVS-L1 & MASVS-L2	Authentication and Session Management Requirements
MSTG-NETWORK	MASVS-L1 & MASVS-L2	Network Communication Requirements
MSTG-PLATFORM	MASVS-L1 & MASVS-L2	Platform Interaction Requirements
MSTG-CODE	MASVS-L2	Code Quality and Build Setting Requirements
MSTG-RESILIENCE	MASVS-R	Resilience Requirements

eShard

MASVS-L1 Standard Security & MASVS-L2 Defense-in-depth

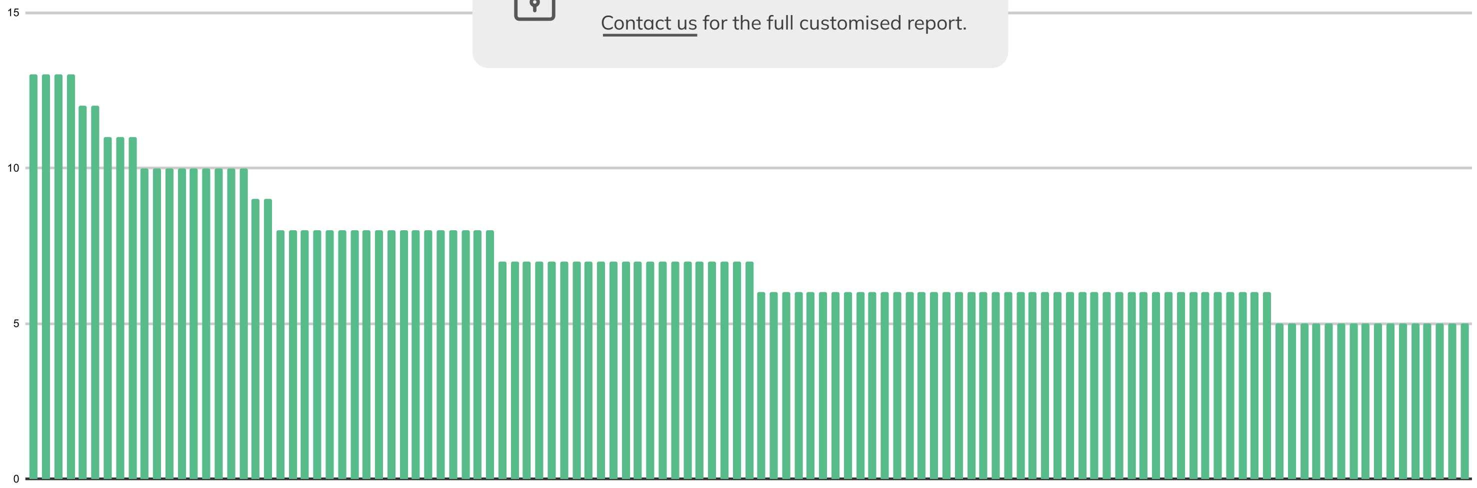
How secure is the code?

V2

MASVS-L1 & L2

STORAGE

These requirements cover the way an application stores sensitive data and how the corresponding accesses are managed.



Curious to know where your banking mobile app rank in this study?

Contact us for the full customised report.

4 mobile banking apps reached the **highest score** of this study. Few violations were however spotted.

6.8 violations reported, in average.

POTENTIAL CONSEQUENCES

When data is stored on the device, the chosen data storage type is very important and its location as well. One must not assume that users or other apps running on the same device will not have access to the device's file system and thus get access and inspect the data. The impacted data can be for instance Personally Identifiable Information (PII), Application Data, Passwords, etc. Failing to securely manage those data can lead to:

> Identity Theft,
> Fraud,
> Reputation Damage,
etc.

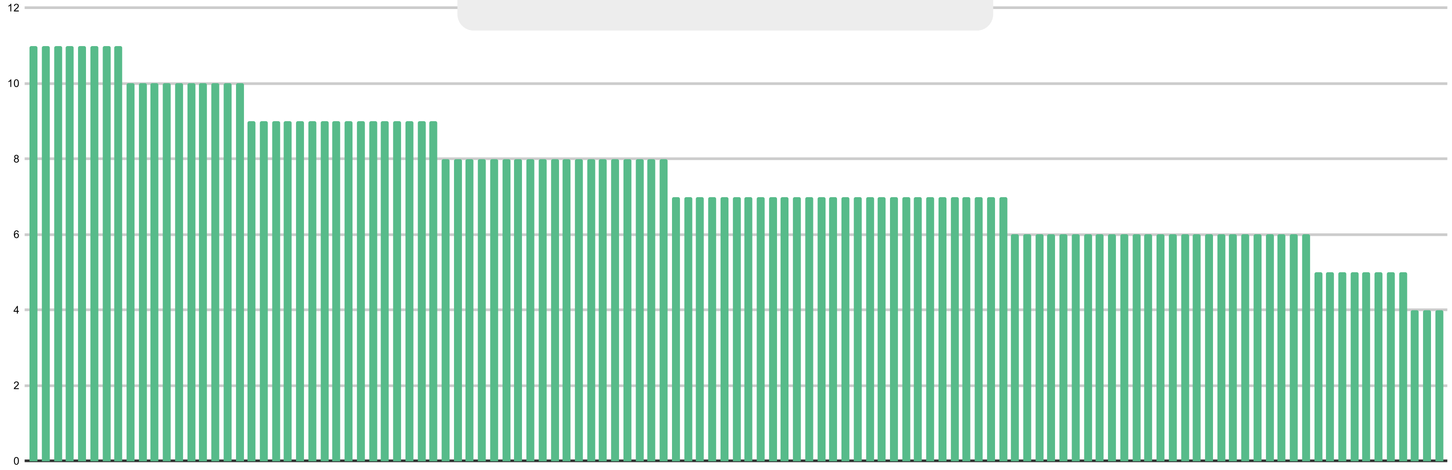
V3

MASVS-L1 & L2
CRYPTO

Cryptography is the heart of mobile application security. It is important to follow the best practices to avoid using a weak algorithm or protocol.

Curious to know where your banking mobile app rank in this study?

Contact us for the full customised report.



8 mobile banking apps reached the **highest score** of this study without any violation.

3.5 violations reported, in average.

POTENTIAL CONSEQUENCES

Cryptography can be used to protect data stored on a mobile device or handled by an application at runtime. When standard conventions and best practices are not followed properly, a user or an application running on the same device can try to exploit a cryptographic weakness that was unintentionally left to get the protected data. This will result in an unauthorized retrieval of potentially sensitive information which can translate into:

> Privacy Violations,
> Information Theft,
> Intellectual Property Theft,
etc.

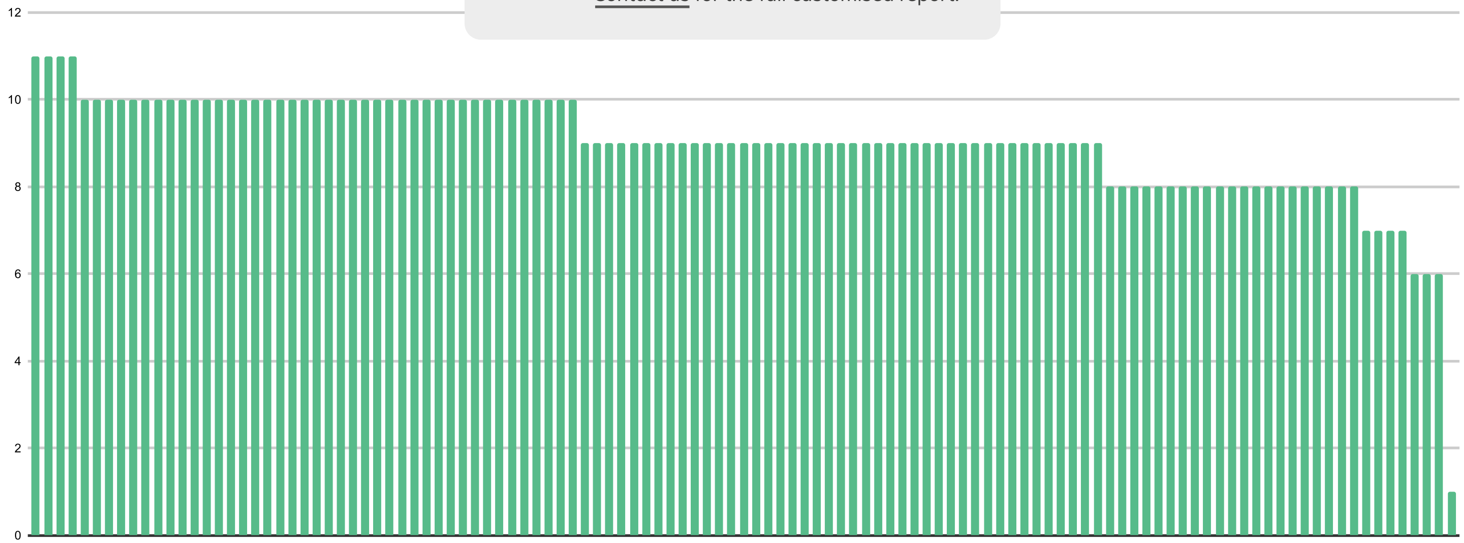
V5

MASVS-L1 & L2
NETWORK

Network requirements cover many topics since any link with the external world may be an open door to a system. It covers topics such as certificate pinning or usage of SSL connections.

Curious to know where your banking mobile app rank in this study?

Contact us for the full customised report.



4 mobile banking apps reached the **highest score** of this study without any violation.

2 violations reported, in average.

POTENTIAL CONSEQUENCES

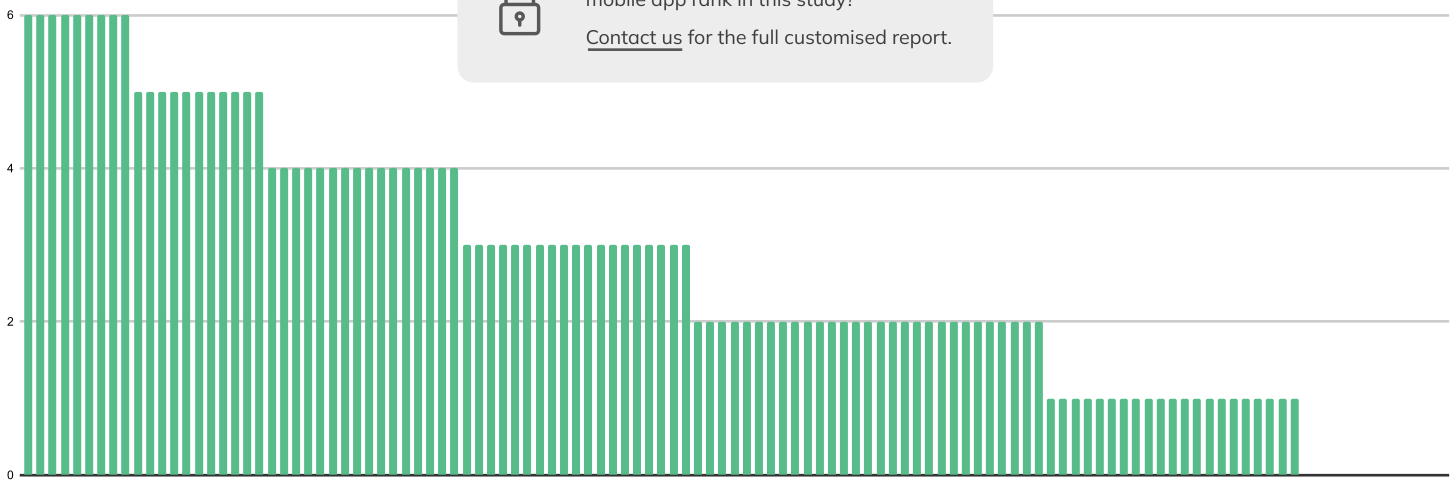
If the information exchanged between the mobile application and its remote service endpoints is not properly protected, its confidentiality and integrity can be compromised by means of various attack vectors, such as for instance, Man-in-The-Middle attacks, Phishing attacks etc. It may result in:

> Identity Theft,
> Fraud,
> Reputational Damage

V6

MASVS-L1 & L2 PLATFORM

The controls in this group ensure that the app uses platform APIs and standard components in a secure manner. Additionally, the controls cover communication between apps (Interprocess Communication).



Curious to know where your banking mobile app rank in this study?

Contact us for the full customised report.

9 mobile banking apps reached the **highest score** of this study without any violation.

3.1 violations reported, in average.

POTENTIAL CONSEQUENCES

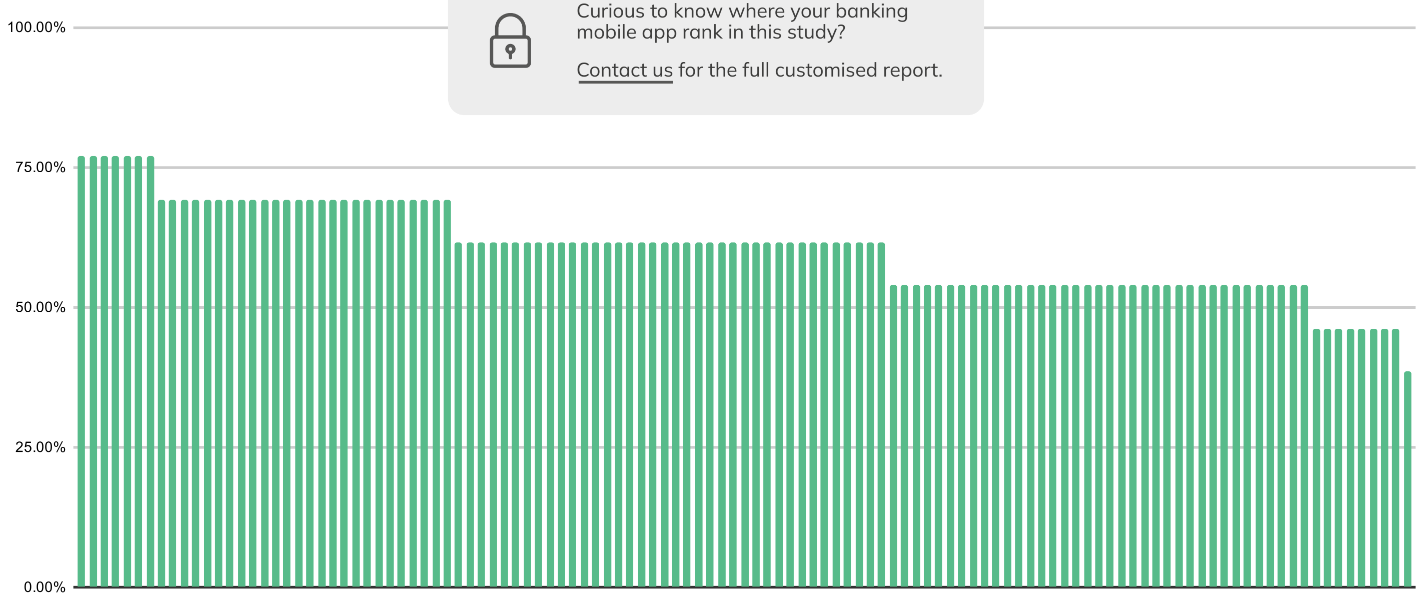
If the app fails in correctly using the features provided by the platform it may result in bugs or introduce weaknesses that might be exploited to implement an attack. This can for instance lead to the violation of development guidelines if those features contradict with the defined best practices.

For example, there are guidelines on how to properly use some features provided by Android. A misunderstanding of how a platform-provided security feature works can also lead to a wrong implementation that can introduce bugs, like for instance setting a wrong flag on an API call.

V7

MASVS-L2 CODE QUALITY & BUILD SETTINGS

Code quality and build settings requirements cover a variety of specifications. It mainly aims at ensuring that basic secure coding practices are followed while developing the application and that "free" security features offered by the compiler are activated (e.g stripping symbols).



Curious to know where your banking mobile app rank in this study?

Contact us for the full customised report.

7 mobile banking apps reached the **highest score** of this study. Few violations were however spotted.

5.2 violations reported, in average.

POTENTIAL CONSEQUENCES

When secure coding best practices are not properly applied, it can introduce security vulnerabilities (e.g arbitrary input injection, buffer overflows, etc.) or greatly help one to reverse engineer the code of the application and thus find potential vulnerabilities. Bad programming practice can then lead to local or remote foreign code execution or denial of service on remote server endpoints. The actual consequences really depend upon the nature of the exploit:

- Information Theft,
- Reputational Damage,
- Intellectual Property Theft

eShard

Resiliency Against Reverse Engineering and Tampering

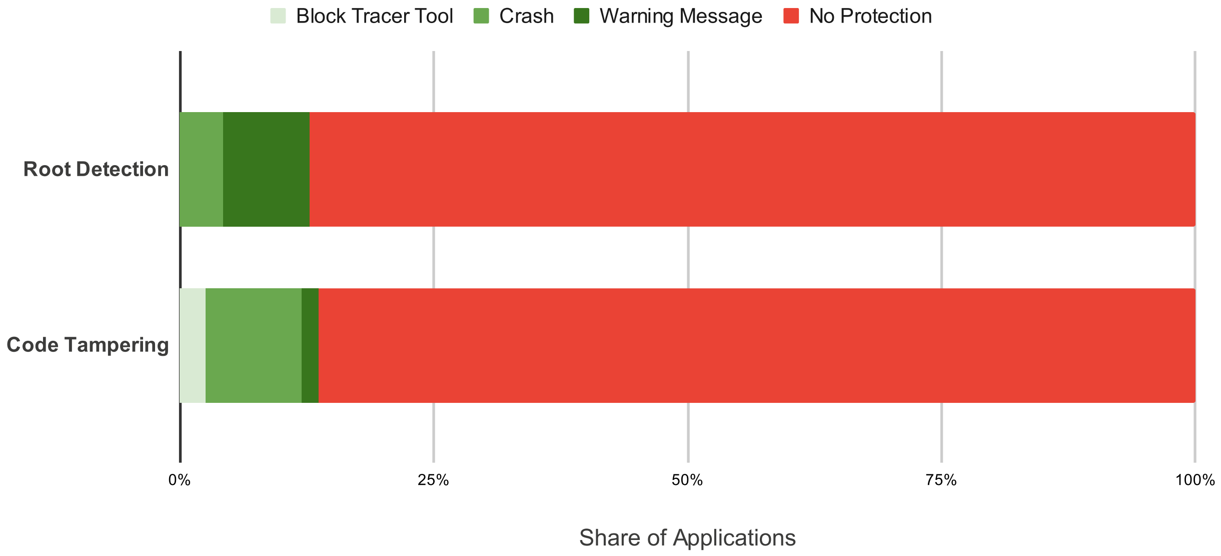
How resilient is the app against attacks?

How do the apps react when triggering their protections?

esChecker provides a unique set of tests leveraging DAST (Dynamic Application Security Testing) technology and real mobile phones. As such, app reactions can be captured genuinely, providing a report that's true to reality. In this study, we have run two types of dynamic tests:

> **Root Detection:** we install the application on a rooted device and we use the app.
> **Code Tampering:** we install the application on a device and try to hook it using an orchestration tool, in an attempt to inject some code for example.

Reactions to Root Detection & Code Tampering



16% of the tested apps are protected and **trigger a protection** mechanism.

75% of these few protected apps, **crash** as a protection against code tampering.

POTENTIAL CONSEQUENCES

Protecting the applications against these types of threats is vital for a highly sensitive market such as Mobile Banking.

We can see that reverse engineering protection within the apps is not the top priority, which echos the knowledge of experts in eShard. At the end market, it's widely observed that the focus is placed on the backend side of the system.

V8

MASVS-R

ROOT DETECTION

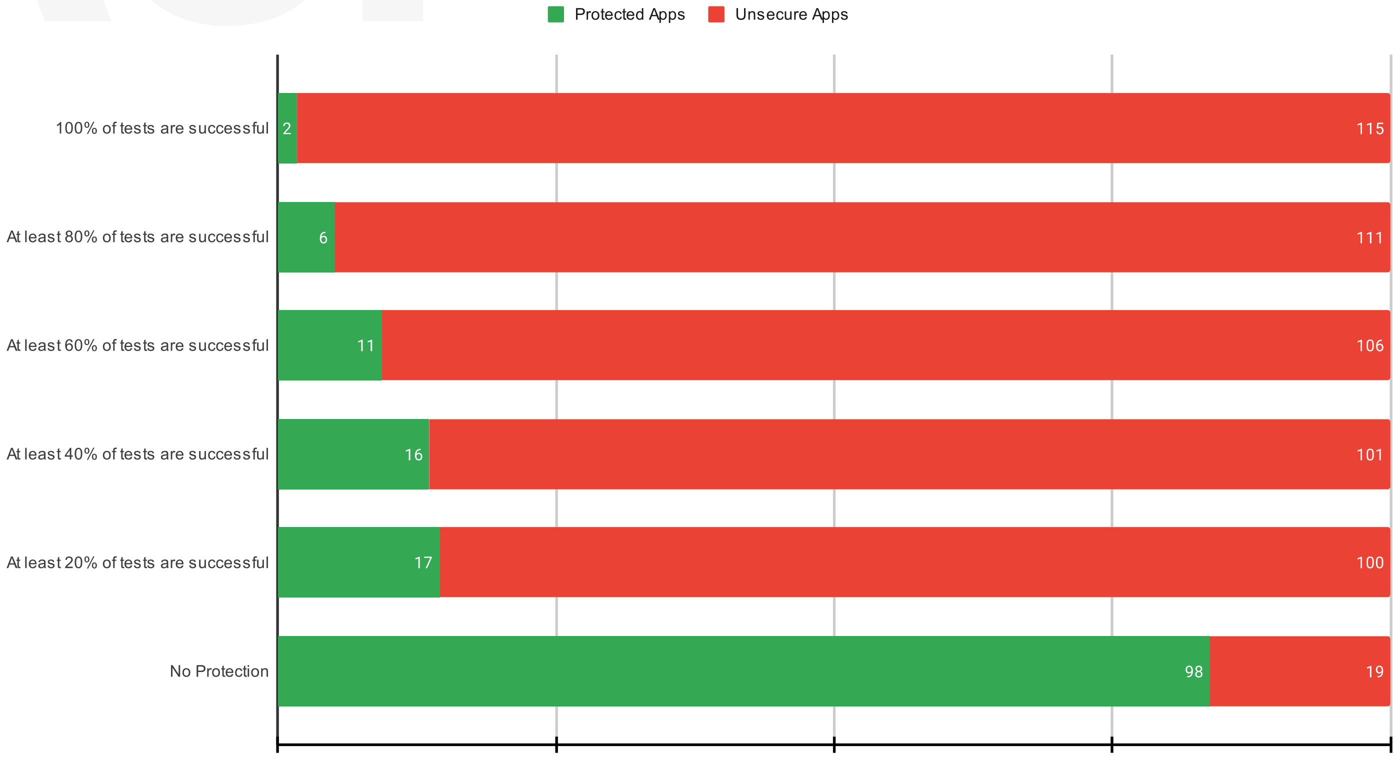
Rooting (Android) or Jailbreaking (iOS) a device allows its owner to take full control of it. Although this operation leads to decent advantages for their users, it removes some strong and native OS protections.

As a result, for applications owners, it's a huge dilemma to choose whether their apps should be allowed to be used on rooted devices.

We have tested the Mobile Banking Apps versus 14 Dynamic Root Detection Tests, to see if they were protected and reacting accordingly.

Curious to know where your banking mobile app rank in this study?

Contact us for the full customised report.



83% of the tested apps have **no root detection** mechanism at all.

ONLY 3 of the tested apps are fully protected.

POTENTIAL CONSEQUENCES

83% of the Top European Mobile Banking Apps don't implement any Root Detection mechanism which means that they do not react against any of the 14 automated tests we launched.

In this scenario, that means that the end-users of these Mobile Banking Apps, when using a rooted phone, may be exposed. If these users have installed a malicious app embarking malware, this app could see all processed and data exchanges between the Banking App and its backend (for instance by looking into the process memory space).

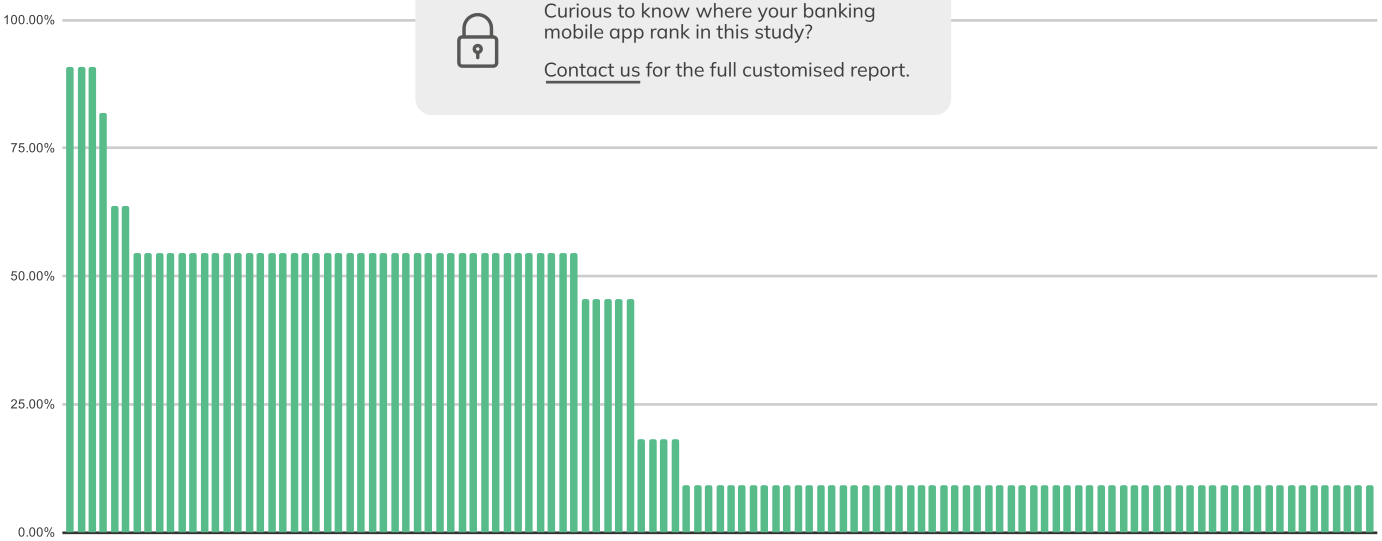
V8

MASVS-R CODE INSTRUMENTATION

Hackers often use tools such as Xposed, Substrate, FRIDA, Debuggers, Code Emulators, etc. to understand the features of the apps and take control.

When developing a mobile application, it's your responsibility to preserve your users' integrity, as well as your business.

The apps need to detect the use of such tools at runtime and react accordingly to prevent any code tampering.



Curious to know where your banking mobile app rank in this study?

Contact us for the full customised report.

ONLY 1 application passes all the Code Instrumentation Tests.

50% of the apps are very exposed, passing less than 10% of the tests.

POTENTIAL CONSEQUENCES

These Code Instrumentation tools are used by hackers to reverse engineer the application in order to catch the network requests, to get the data stored in the app and on the phone and sometimes to alter the data sent and received by the app to the backend system.

This could lead to some poor data quality sent to your servers. Hackers could also bypass some login steps and access some protected areas of the applications.

In our study, we have noticed a long-tail of apps very exposed to these risks. Even if the backend system is highly protected, the application is a window to your fortress accessible to potential hackers. You should make it as hard as possible to break.

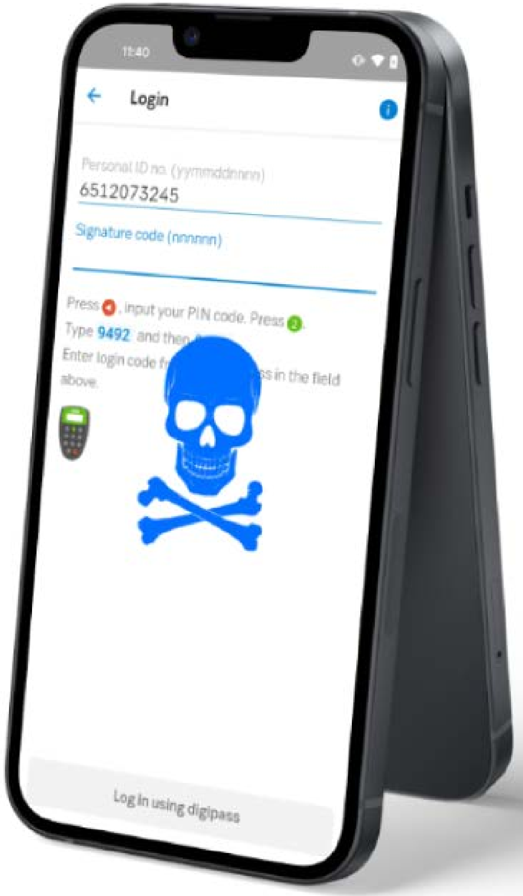
MASVS-R V8

CODE INJECTION EXAMPLE

Thanks to esChecker, we manage to inject some code into the applications when they don't detect the usage of tools such as FRIDA.

In the following example, the tool is able to automatically put an image into the application, on top of the rest of the screen.

Most of the tested applications didn't pass this test.



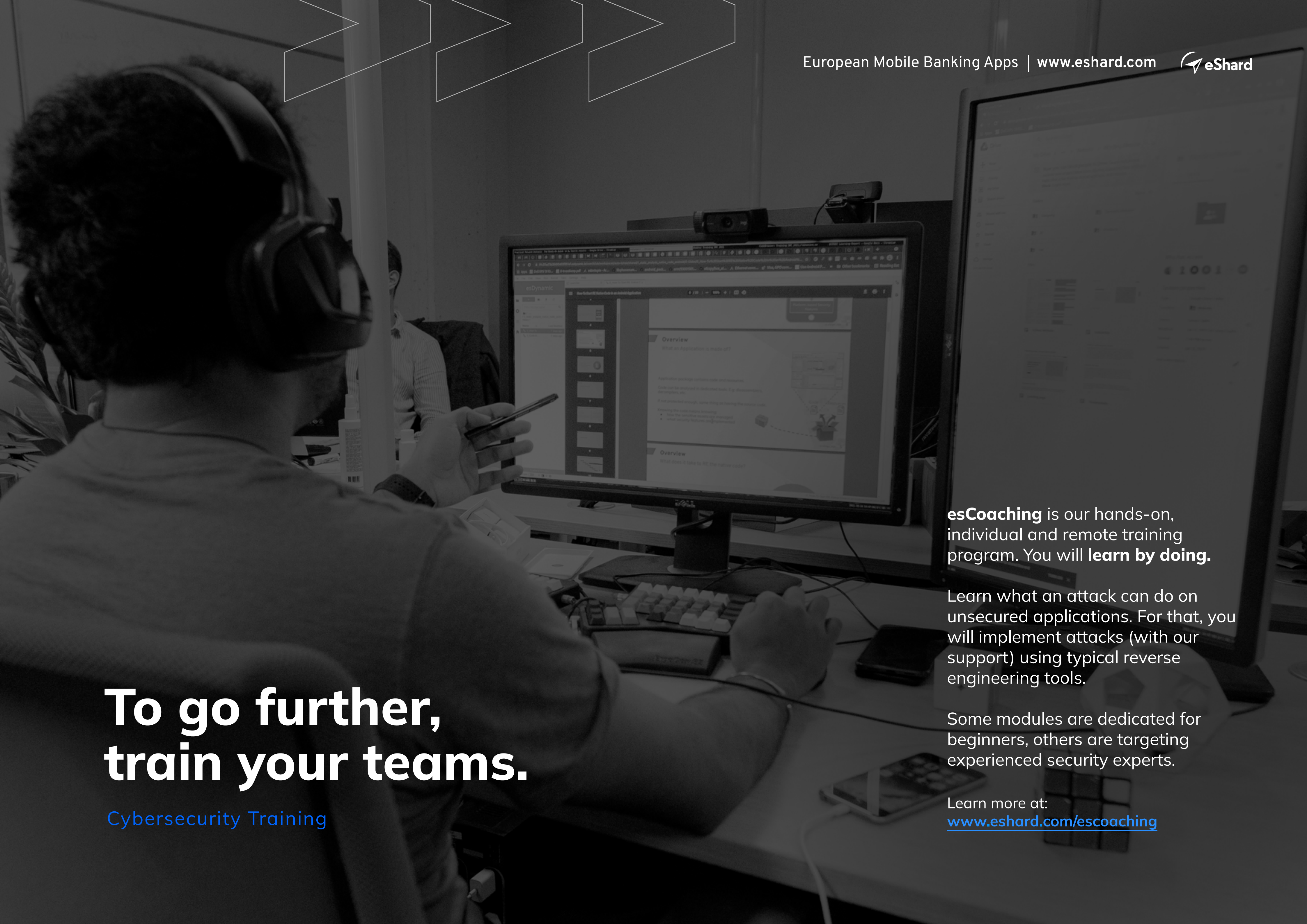
Login

Personal ID no. (yymmddnnnn)
6512073245

Signature code (nnnnnn)

Press ●, input your PIN code. Press ●.
Type 9492 and then
Enter login code f s in the field above.

Log in using digipass



To go further, train your teams.

Cybersecurity Training

esCoaching is our hands-on, individual and remote training program. You will **learn by doing.**

Learn what an attack can do on unsecured applications. For that, you will implement attacks (with our support) using typical reverse engineering tools.

Some modules are dedicated for beginners, others are targeting experienced security experts.

Learn more at:
www.eshard.com/escoaching

About eShard

We believe that security must be implemented in depth and multi layers. For each layer, the threat is specific.
We developed software and services to validate the efficiency of the protections.
It is important that the protections in place are checked at the right level.

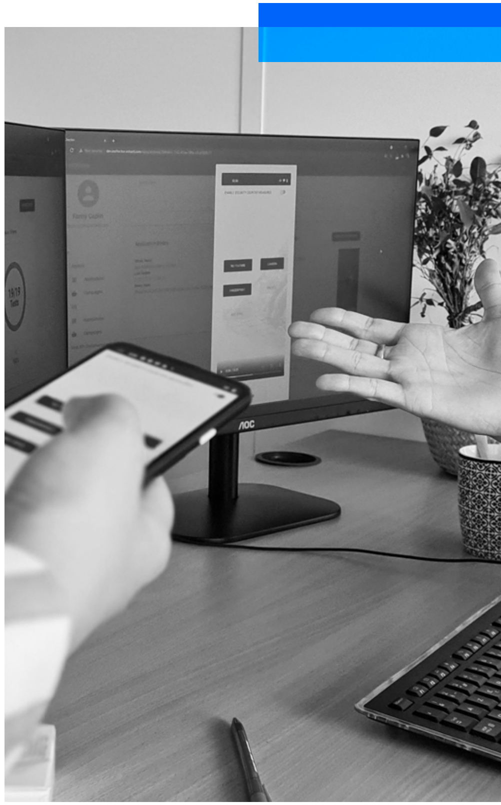
Find out more: www.eshard.com.

About esChecker

esChecker performs automated testing to validate your app behaviour facing an intrusion with advanced reverse-engineering testings. A common belief is to think that only applications handling banking or medical data are to be protected, but in fact, any application handling personal data has to be secured.

As solution providers, you are responsible to do your best to secure personal and sensitive data. This starts by having a qualification process to validate the protections.

Find out more: www.eshard.com/eschecker.



eShard

Get in touch

www.eshard.com

contact@eshard.com

/company/eshard

@eshard

France HQ
Bâtiment GIENAH
11 avenue de Canteranne
33600 Pessac, France

France R&D
7 rue Gaston de Flotte
13012 Marseille, France

Singapore
#04-01 Paya Lebar Quarter
1 Paya Lebar Link
Singapore, 408533

Germany
eShard GmbH
Beethovenallee 21
53173 Bonn