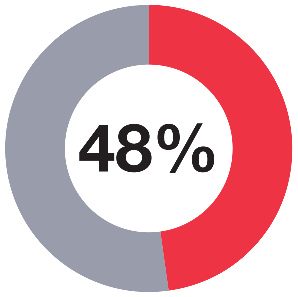
A Manager's Guide to the OWASP Mobile Security Project

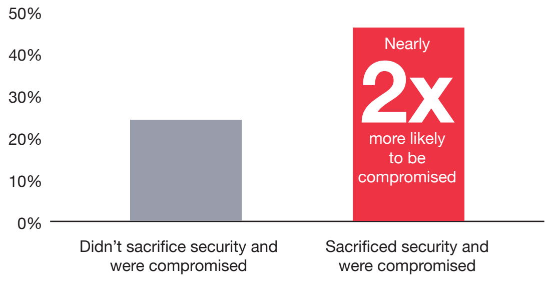
Building & Executing Risk-Based Security Policy for Mobile Apps

NowSecure™

Introduction



The Verizon *Mobile Security Index 2019* found half (48 percent) of respondents sacrificed security to “get the job done.”[1]



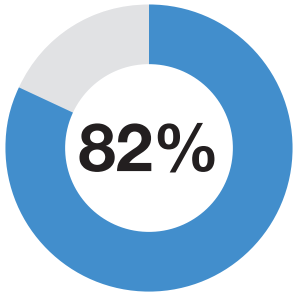
Those who sacrificed security were nearly 2x as likely to suffer a compromise.[1]

The world around us has gone mobile. U.S. adults will spend an average of 3 hours and 34 minutes per day on mobile devices this year, surpassing time spent watching TV.[2] Over 80% of that time will be spent in mobile apps versus a mobile browser. In fact, the demand for mobile apps continues to soar with Apple® App Store® and Google Play™ downloads growing more than 20% from 2016 to 2018.[3]

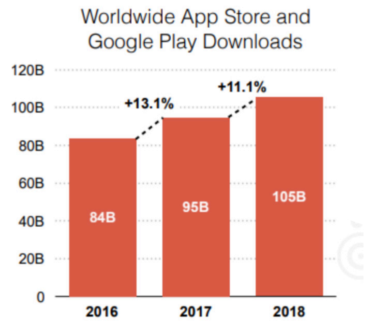
It's no surprise that security teams are finding it difficult to ensure security and privacy. Mobile app downloads grew from zero to 105 billion in just a decade, and developers outnumber security staff 100:1.[4] Large enterprise businesses today develop or maintain over 60 mobile apps yet the standardization of mobile app security and privacy testing is still in its infancy.

NowSecure (formerly viaForensics) has been in the business of mobile app security since Apple and Google debuted their app stores. Through software and services, we have blazed new frontiers with our customers during the last decade, building mobile app security programs from the ground up and modernizing security

“The Verizon *Mobile Security Index 2019* found half (48 percent) of respondents sacrificed security to “get the job done,” up from 32 percent last year...What's more, the index found those who sacrificed security were nearly twice as likely to suffer a compromise — 46 percent versus 24 percent. And the majority (62 percent) of those affected described the event as “major.”[1]



Over 80% of mobile device time is spent on apps vs the mobile browser.[2]



Apple® App Store® and Google Play™ downloads grew by over 20% between 2016 and 2018.[3]

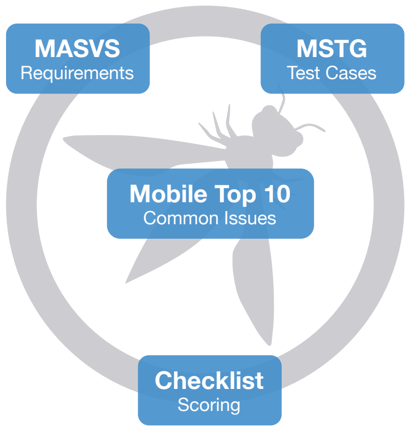
so teams can balance the demands of mobile-first strategies. Over the years, customers have sought our help addressing these frequently asked questions:

- Where should I start when building a new mobile appsec program?
- How do we consistently secure our mobile app portfolio?
- How do we efficiently scale and prioritize our resources?
- Do all apps require the same level of review?
- How do I best prioritize developer remediation?

With most mobile appsec programs still in their early days and the volume of apps to protect ever multiplying, organizations need an effective path forward. Fortunately, a growing library of resources helps security managers better understand mobile app security and privacy risks and provides a common language and framework to align cross-functional teams.

One such community providing mobile security standards is the Open Web Application Security Project (OWASP). As an active OWASP supporter, we wrote this guide to help security managers standardize mobile app security testing to boost efficiency and effectiveness while reducing risk.

OWASP Goes Mobile



OWASP has established itself as a highly respected industry standard for web application security. As the popularity of mobile apps grew dramatically, it became apparent that the risks and attack surface for mobile fundamentally differ from web. This mandated a different approach for mobile app security testing.

In 2011, OWASP introduced the Mobile Top 10 to designate the 10 most common areas of security issues within mobile apps. While updating the Mobile Top 10 in 2016, OWASP launched the Mobile Security Project for which professionals around the globe contribute to the Mobile Security Testing Guide (MSTG), the Mobile App Security Verification Standards (MASVS) and the Mobile App Security Testing Checklist.

These OWASP resources work together to provide foundational building blocks for a common language to use cross-functionally across an organization navigating mobile security. MASVS teaches architects, business owners, and development leaders how to build security requirements into the mobile app by design. The MSTG helps security analysts learn how to test for each area in the Mobile Top 10. The checklist provides security analysts a template to reference line items from MASVS requirements and MSTG testing procedures as they pen test mobile apps.

But not all apps impact the business equally and security teams do not have infinite resources. There must be a way to assess and rank risk among mobile apps across a given portfolio. We typically recommend MASVS as a starting point because it translates a threat model for mobile app security into security requirements to standardize the level of testing necessary to manage risk.

OWASP Top 10 Comparison

Mobile	Web
1. Improper Platform Usage	1. Injection
2. Insecure Data Storage	2. Broken Authentication
3. Insecure Communication	3. Sensitive Data Exposure
4. Insecure Authentication	4. XML External Entities
5. Insufficient Cryptography	5. Broken Access Control
6. Insecure Authorization	6. Security Misconfiguration
7. Client Code Quality	7. Cross Site Scripting
8. Code Tampering	8. Insecure Deserialization
9. Reverse Engineering	9. Using Vulnerable Components
10. Extraneous Functionality	10. Insufficient Logging/Monitoring

OWASP Mobile Security Resources At-a-Glance

What is OWASP?

This open community is vendor-neutral and allows security professionals to contribute unbiased, practical, cost-effective information about application security. OWASP has become the organization that individuals, corporations, universities, government agencies, and other organizations worldwide look to for standards in web and mobile app security.

OWASP Mobile Top 10

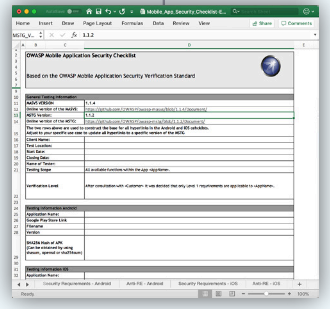
Description: Last updated in 2016, this lists the 10 most common areas of risk in mobile apps.
How to use it: The Mobile Top 10 provides a foundation to educate developers, QA teams, and security professionals on areas of common security gaps and lends itself to discussing secure mobile app development best practices.
>>> Click here for the most current version of the OWASP Mobile Top 10.

OWASP Mobile App Security Verification Standards (MASVS)

Description: This resource outlines the different verification requirements for basic mobile app security, defense-in-depth app security and reverse engineering resilience.
How to use it: MASVS aids mobile app owners, architects and developers in building security by design, as well as ensuring consistent testing coverage by security professionals. This guide maps to the OWASP Mobile Top 10, OWASP Mobile App Security Testing Guide (MSTG) and CWE.
>>> Click here for the most current version of the OWASP MASVS.

OWASP Mobile Security Testing Guide (MSTG)

Description: This is an ever-growing manual for security analysts to test various areas of the OWASP Mobile Top 10 on iOS and Android apps.
How to use it: MSTG is a reference guide for security analysts of all levels of experience to promote well-rounded mobile app security testing.
>>> Click here for the most current version of the OWASP MSTG.

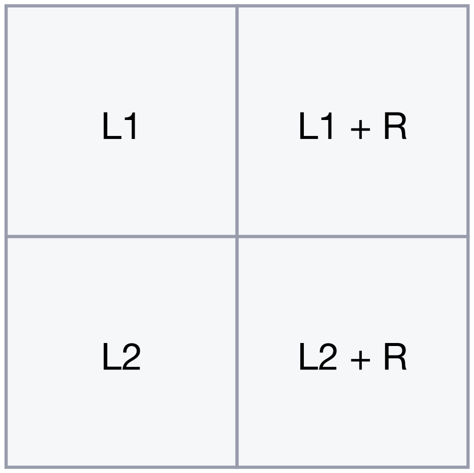


OWASP Mobile Security Testing Checklist

Description: This spreadsheet outlines items to pass or fail for MASVS requirements and references sections of the MSTG to assist with testing.
How to use it: Security analysts can use the checklist as a template for management reports when performing mobile apps assessments.
>>> Click here for the most current version of the OWASP Mobile App Security Testing Checklist.

NowSecure proudly sponsors the OWASP Mobile Security Project.

Mobile App Security Threat Modeling Using OWASP MASVS



Above: The four different levels of security verification using OWASP MASVS

Ensuring all mobile app stakeholders understand the app risks can help everyone make decisions about vulnerabilities. Organizations should apply one of four different levels of security verification per mobile app depending on the characteristics of the app as shown in the diagram to the left.

L1 – Standard Security Verification

L1 describes the most basic level of security verification any mobile app should undergo. This might be a basic calendar application, a voice memo app or an event app with high-level event information. These apps do not handle intellectual property, critical transactions or user information that is considered high risk.

L2 – Defense-in-Depth Verification

L2 requires L1 tests and additional defense-in-depth controls, meaning the mobile app must be resilient to more sophisticated attacks. Mobile apps that reside in this quadrant tend to interact with highly sensitive data such as credit card or healthcare information that can be used for fraud.

L1 + R – Standard Security + Reverse Engineering Resiliency

R stands for the reverse engineering resiliency requirements. These are critical to preserving the integrity of mobile app functionality and protecting confidential information such as intellectual property. A gaming app, for example, requires L1 testing with reverse engineering resiliency. Mobile app game developers cannot afford to allow users to tamper with the app and cheat the system. Anti-tampering requirements protect the integrity of game play.

L2 + R – Defense-in-Depth + Reverse Engineering Resiliency

Online banking apps provide a prime example of apps needing L2 verification with reverse engineering resiliency. Essentially, if the app handles sensitive data, performs high value transactions, needs to comply with regulatory regimes and/or must prevent leakage or tampering of personally identifiable information (PII), the app likely needs L2 + R level of verification.

MASVS L1 *Standard Security* • This is the baseline • No compliance or regulatory needs • No intellectual property (IP) or highly sensitive data handled • No high value transactions performed • Simple apps	**MASVS L1 + R** *Standard Security + High RE Resilience* • Prioritize IP protection • Prevent malicious modification or tampering
MASVS L2 *Defense-in-Depth* • Regulated industry data • Compliance consideration • Apps that perform simple tasks, but handle highly sensitive data	**MASVS L2 + R** *Defense-in-Depth + High RE Resilience* • Apps that perform complex activities between users and handle highly sensitive data • Compliance and IP protection are key • Prevent malware-based attacks

In order to understand which quadrant of verification a mobile app requires, consider the purpose of the app and how it will be used. Organizations need to zero in on three key areas:

- **Purpose:** What does the app do?
- **Data:** What data does this app handle?
- **Perspective:** Protecting the organization is important, as is complying with regulatory standards, but let's not forget that user demand and experience drives mobile app success.

To help make these different levels of verification concrete, let's step through the domains V1-V8 of MASVS and tie security vulnerabilities and business impact stories to each domain. While domain V1 focuses more on the design and architecture of the app and not specific test cases, domains V2-V8 require static analysis (SAST) and dynamic analysis (DAST) to thoroughly exercise the app and confirm it meets the security requirements listed in MASVS.

V1: Architecture, Design and Threat Modeling Requirements

This domain focuses on the architecture and design of the mobile app. Architects, development leaders and mobile app owners should take these requirements into consideration when designing and building mobile apps.

Look at the MASVS requirement below and consider the ramifications in the real-world example that follows.

MASVS Requirement in Question
Security Requirement 1.5
All app components are defined in terms of the business functions and/or security functions they provide.

Relevant Verification Levels:
L2, L2+R

L1	L1 + R
L2	L2 + R

Sample Context
A digital wallet that lets friends share payments skyrockets to popularity for its ease of use. This app handles user personally identifiable information (PII), banking information, has high visibility and high impact to customers, and the code functionality of the app should be tamperproof. Therefore, it requires L2+R type of security verification.

The Attack Scenario
A young man walks into a bar and strikes up a conversation with a young lady. Things are going well so he asks for her number and hands his phone over for her to enter it. But instead of entering her number, she transfers $2,500 to herself using his digital wallet. Because the app doesn't enable session management by default, he was unable to proactively prevent the attack. She hands the phone back to him, says goodbye, and walks away $2,500 richer.

The Business Impact
While con artists and swindlers have been tricking lovers for centuries into giving away sums of money, this digital wallet is just another play in their bag of tricks. The digital wallet company loses millions of dollars from scams, fraudulent payments, and ultimately customers who will never use its mobile app again.

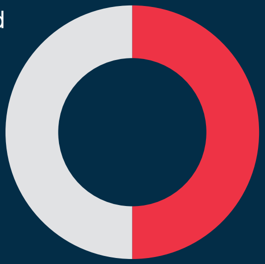
OWASP MASVS Reference
V1: Architecture, Design and Threat Modeling Requirements

OWASP MSTG Reference
N/A - Because this domain does not have specific test cases, there is no reference to the testing guide.

OWASP Mobile Top 10 (2016)
M10 - Extraneous Functionality

V2: Data Storage and Privacy Requirements

50% of mobile apps tested by NowSecure software store data insecurely. Frequently discovered unencrypted PII include username, account number, phone number, IMEI, GPS location, Wi-Fi/ MAC address and more in system files and local files on-device.



As privacy concerns grow, organizations must protect user data more proactively in even the most basic mobile apps. Data classified as “sensitive” may vary across organizations. Sensitive data may include location data, username and passwords, personally identifiable information (PII) and documents with confidential intellectual property (IP).

The example below maps a finding to the V2 domain of MASVS. Links to MSTG are referenced in the sidebar.

Sample Finding
Sensitive Data (PII) Found in Public Device Storage

Brief Description of Finding
The application saves sensitive data to a public app folder on the device that’s accessible by other apps.

MASVS Requirement in Question
Security Verification Requirement 2.2
No sensitive data should be stored outside of the app container or system credential storage facilities.

Relevant Verification Levels:
All

L1	L1 + R
L2	L2 + R

Sample Context
A healthcare provider invests $800,000 to build a mobile app for users to track vital protected health information (PHI) to inform them when to take their medicine and control updates to their Internet of Things (IoT) medical devices. It believes this initiative is mandatory for the best user experience. This app handles user personally identifiable information (PII), has high visibility and high impact to customers, and the code functionality of the app should be tamper-proof. Therefore, it requires L2+R type of security verification.

The Attack Scenario
An attacker discovers that this app stores data and firmware updates in a public app folder on the device. This is considered unprotected storage and can be accessed by other applications. Firmware updates can be modified and wreck the user experience, or worse, seriously compromise the user health.

The Business Impact
Because this IoT device has become woven into the lifestyle of its users, disruption in usage can range from mildly annoying to life threatening. The risks include litigation and tarnished brand reputation. User adoption is critical to revenue stream. Privacy gaps and firmware malfunctions would significantly impede IoT adoption and company growth.

OWASP MASVS Reference
V2: Data Storage and Privacy Requirements

OWASP MSTG Reference
Data Storage on Android
Data Storage on iOS

OWASP Mobile Top 10 (2016)
M2 - Insecure Data Storage

V3: Cryptography Requirements

Cryptography is essential to protecting sensitive data. The requirements in this domain relate to correct implementation of cryptography.

Sample Finding
Hardcoded values used for crypto, weak algorithm for crypto.

Brief Description of Finding
The application was found to be using hardcoded values to generate cryptographic data. It was also discovered that the app relies on an out-of-date cryptography method, AES-ECB, to generate cryptographic data.

MASVS Requirement in Question
Security Verification Requirement 3.2
The app uses proven implementations of cryptographic primitives.

Relevant Verification Levels:
All

L1	L1 + R
L2	L2 + R

Sample Context
A secure messaging app handles highly confidential intellectual property (IP), has high visibility and high impact to users, and the code functionality of the app should be tamper-proof. It requires L2+R type of security verification from both the owner and user perspective.

The Attack Scenario
The app attempts to prevent attackers from obtaining their customers' messages in every attack scenario, so it implements client-side cryptography to protect said data. Unfortunately, a weak algorithm was employed and the key used to generate encrypted data was exposed within the app.

The Business Impact
A secure messaging app that turns out to be leaky would seriously harm the brand reputation and revenue stream of the app producer. Additionally, the company would probably need to cover the damages incurred to its customer that lost IP. The customer would also suffer a loss of brand reputation and first-to-market advantage, among other negative consequences.

OWASP MASVS Reference
V3: Cryptography Requirements

OWASP MSTG Reference
Crypto on Android
Crypto on iOS

OWASP Mobile Top 10 (2016)
M5 - Insufficient Cryptography

V4: Authentication and Session Management Requirements

When users are granted access to a remote service, the endpoint must confirm the users' identity and privileges for the remote service. Additionally, the remote endpoint should employ rate limiting to protect against brute-force attacks.

Sample Finding
Denial of services due to lack of rate limiting

Brief Description of Finding
The application does not implement rate limiting during registration of physical hardware, leading to a denial-of-service attack.

MASVS Requirement in Question
Security Verification Requirement 4.6
The remote endpoint implements a mechanism to protect against the submission of credentials an excessive number of times.

Relevant Verification Levels:
All

L1	L1 + R
L2	L2 + R

Sample Context
A children's toy manufacturer invests heavily to revamp a retro favorite for the modern age. The popularity of a new TV show, comic and movie have driven sales of the new IoT toy through the roof — it is the hottest toy of the holiday season. This app requires L2 security verification because it requires user information for authentication and must comply with the Children's Online Privacy Protection Act (COPPA).

The Attack Scenario
Toy owners use an app to interact with it, but they first must create an account and register the serial number. This registration process connects the toy to the owner's account. An attacker realizes the serial numbers are not only short and sequential, but also discovers that the endpoint lacks rate limiting. He attempts a brute-force attack against the endpoint and registers more than a million serial numbers yet to be claimed.

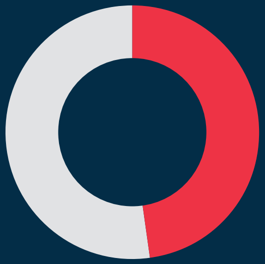
The Business Impact
Thousands of eager new toy owners attempt to create their accounts on the mobile app only to discover their toy has already been registered to another account. Consequently, these children cannot interact with their toy as was marketed. Parents return the defunct toy and take to social media to air complaints about poor customer support and functionality. The toy manufacturer's stock price plummets and its brand reputation suffers from such a high-profile failure.

OWASP MASVS Reference
V4: Authentication and Session Management Requirements

OWASP MSTG Reference
Local Authentication on Android
Local Authentication on iOS

OWASP Mobile Top 10 (2016)
M4 - Insecure Authentication
M6 - Insecure Authorization

V5: Network Communication Requirements



48% of mobile apps tested by NowSecure software have insecure network communications. The most frequent causes are improper hostname verification and improper certificate validation, which allow attackers to expose users to malicious endpoints, data theft and remote phishing attacks. In many instances, these issues lie in third-party code and only present when an app is dynamically analyzed through automation or manual testing.

This domain focuses on ensuring the confidentiality and integrity of mobile app data transmitted over the network. Developers should use encrypted channels for communication using the TLS protocol and in some cases, certificate pinning, to safeguard data in transit.

Sample Finding
Hostname verification improperly implemented

Brief Description of Finding
The application validates the certificate authority (CA), but not the hostname.

MASVS Requirement in Question
Security Verification Requirement 5.3
The app verifies the X.509 certificate of the remote endpoint when the secure channel is established. Only certificates signed by a trusted CA are accepted.

Relevant Verification Levels:
All

L1	L1 + R
L2	L2 + R

Sample Context
A large retail company invests over $1 million to build a mobile app. Based on an analysis of its customer base, the company projects the initiative will lead to a 10% increase in annual sales by driving more customer engagement and transactions. Because this app handles user payment information, the network functionalities in the app should be protected by certificate pinning. This app requires L2 type of security verification from the user perspective and L2 + R from the owner perspective.

The Attack Scenario
In a rush to meet deadlines and jumpstart customer adoption, the mobile app gets pushed to the app store with no security testing. An attacker sees that the app not only failed to implement certificate pinning, but the hostname verification was not properly implemented, and takes advantage of it to launch a phishing attack. Not even a Mobile Device Management system or Virtual Private Network can prevent this attack.

The Business Impact
This phishing attack tricks customers into sending money abroad and creates a social media storm that damages brand reputation and customer loyalty. Adoption of the mobile app stalls, customer transactions dip, stock prices drop, and by the end of the quarter, overall company earnings dip 3%. This doesn't even take into account the ensuing lawsuit and fines.

OWASP MASVS Reference
V5: Network Communication Requirements

OWASP MSTG Reference
Android Network APIs
iOS Network APIs

OWASP Mobile Top 10 (2016)
M3 - Insecure Communications

V6: Environmental Interaction Requirements

These controls ensure that the app uses platform APIs and standard components in a secure manner. They also address communication between apps.

Sample Finding
Native objects exposed through webviews

Brief Description of Finding
The application exposes Java objects through the use of a webview leading to remote code execution through a Javascript interface.

MASVS Requirement in Question
Security Verification Requirement 6.7
If native methods of the app are exposed to a WebView, verify that the WebView only renders JavaScript contained within the app package.

Relevant Verification Levels:
All

L1	L1 + R
L2	L2 + R

Sample Context
A weather app uses Webviews to generate ads within the user interface (UI) and drive revenue for the app developer. This app doesn't handle any sensitive data other than location data and it doesn't need to be tamper proof, so the app requires L1 type of security verification.

The Attack Scenario
This app not only uses `addJavascriptInterface`, it also uses HTTP requests to render UI content within the app from javascript and HTML. An attacker executes a man-in-the-middle (MiTM) attack to inject malicious Javascript into the app, phishing users for personal information and trolling users with unwanted images. The native bridge created by the `addJavascriptInterface` allows the attacker to generate Toast notifications in the users device, and the use of HTTP loaded by a webview allows him to render any content within the app he desires.

The Business Impact
Jarring inappropriate images would abruptly interrupt usage of the app and drive users to uninstall the mobile app. In addition, the Javascript interfaces would allow an external entity to interact with the native code within the app. In some cases this could be as simple as showing toast notification in the UI, and at worst, it could be used to send requests to other apps on the device. This would impede the app developer's revenue stream because there are plenty of other weather apps on the market.

OWASP MASVS Reference
V6: Environmental Interaction Requirements

OWASP MSTG Reference
Android Platform APIs
iOS Platform APIs

OWASP Mobile Top 10 (2016)
M1 - Improper Platform Usage

V7: Code Quality and Build Setting Requirements

Over 32% of mobile apps tested by NowSecure software have flaws related to poor code quality. In many instances, these issues lie in third-party libraries. Proper implementation of the latest versions of code libraries is critical to code quality. Common web issues like SQLInjection and XSS occur in mobile apps as well.

The purpose of this control is to ensure developers follow basic security coding practices in developing the app and take advantage of security features built into the compiler.

Sample Finding
AFNetworking vulnerability detected

Brief Description of Finding
The application was found to be using an outdated version of the AFNetworking library. This vulnerability was patched as of version 2.5.2. However, if an older version is used, it allows all the SSL traffic to be intercepted and decrypted in a standard MiTM environment.

MASVS Requirement in Question
Security Verification Requirement 7.5
All third-party components used by the mobile app such as libraries and frameworks are identified and checked for known vulnerabilities.

Relevant Verification Levels:
All

L1	L1 + R
L2	L2 + R

Sample Context
A developer of a dating app uses code from an old code base containing an outdated version of AFNetworking framework. The code is included in a new feature update that allows the user to display his full profile info, including address and phone number from the app UI. Because of a short deadline, the code is pushed to production without security testing. This app does not have regulatory requirements, so it requires a basic level of security verification (L1).

The Attack Scenario
Similar to network communication issues in domain V5, an attacker can exploit this issue via a MiTM attack. The core difference from the perspective of the developer is he has done everything correctly, but used a vulnerable version of the AFNetwork framework.

The Business Impact
Without any security testing, the app was deployed and the vulnerability was launched into the wild. A security researcher posts a Twitter tirade about how the app should never be used for privacy concerns. The uninstall rate soars and the company's stock plummets.

OWASP MASVS Reference
V7: Code Quality and Build Setting Requirements

OWASP MSTG Reference
Android Code Quality & Build Settings
iOS Code Quality & Build Settings

OWASP Mobile Top 10 (2016)
M7 - Poor Code Quality

V8: Resilience Requirements

NowSecure Stat
Over 33% of mobile apps tested by NowSecure software are exposed to reverse engineering or code tampering. iOS has strong DRM features built-in to improve resilience, but Android app developers need to build more resilience features due to the Java foundation.

Defense-in-depth tactics seek to increase resilience against unauthorized tampering, reverse engineering and specific client-side attacks. Resiliency is required for apps that process or grant access to sensitive data and functionality.

Sample Finding
Lack of anti-tamper techniques

Brief Description of Finding
The app does not implement anti-tamper techniques to prevent malicious modification of the code.

MASVS Requirement in Question
Security Verification Requirement 8.6
The app detects and responds to tampered code and data in its own memory space.

Relevant Verification Levels:
L1+R, L2+R

L1	L1 + R
L2	L2 + R

Sample Context
A retail app implements a new coupon process to increase holiday sales, allowing customers to show their "single use" coupons in-app while shopping at the store or online. The convenience is a huge success with an initial spike in app downloads and increased sales.

The Attack Scenario
The app implements good cryptography practices, certificate pinning, and other security measures, including expensive industry-standard app protection technology. While these measures help protect the user's security and privacy, there is still a major issue within the app's logic.

Unfortunately, the coupons are generated on the client side without tamper protection, allowing an attacker with a penchant for new electronics to reverse engineer the app and generate coupons of their own decided criteria on the fly. Not only are these coupons absurdly good deals, but also are valid in the eyes of the payment system.

The Business Impact
The attacker shares how to use the exploit, leading to massive deals on the most expensive items in the store. Unfortunately, the retail company's slow remediation process doesn't close this gap quickly enough, infringing on holiday sales.

OWASP MASVS Reference
V8: Resilience Requirements

OWASP MSTG Reference
Android Reverse Engineering Resiliency
iOS Reverse Engineering Resiliency

OWASP Mobile Top 10 (2016)
M8 - Code Tampering
M9 - Reverse Engineering

Standardize. Scale. Share.

Mobile app risk continually evolves as mobile platforms, operating systems and development environments advance.

When you scale mobile app security testing, use this as a reference guide to frame the challenge of securing an ever-growing mobile app portfolio with finite resources. Start standardizing mobile app security testing by triaging your current mobile app portfolio. Then build a risk-based security policy for mobile apps to scale as needed.

No single analyst, security expert, developer or architect can solve mobile app security problems alone. It takes a village to deliver secure mobile apps. And it takes great communication and collaboration to speed the production of secure mobile apps.

Share this guide with your colleagues. Establish a common language and framework with them. Create processes that enable mobile app stakeholders to make swift decisions without compromising the security posture of your organization. And most importantly, give back to the community. Global security professionals contribute to OWASP projects and continue to improve the library of resources for mobile app security. We're all in this together to save the world from unsafe mobile apps.

Sources

1. Verizon study: Companies are becoming increasingly more mobile, but investment in mobile security is stuck in park
2. eMarketer Report: US Time Spent with Mobile 2019
3. Business of Apps: App Download and Usage Statistics (2019)
4. 2018 DevSecOps Community Survey

NowSecure Resources

About NowSecure

NowSecure proudly sponsors the OWASP Mobile Security Project. Only NowSecure delivers fully automated mobile app security testing software with speed, accuracy and efficiency for Agile and DevOps initiatives. Through static, dynamic, behavioral and interactive mobile app security testing on real Android and iOS devices, NowSecure identifies the broadest array of security threats, compliance gaps and privacy risks. NowSecure customers can choose automated software on-premises or in the cloud, expert professional penetration testing and managed services, or a combination of all as needed. Visit **www.nowsecure.com** for more information.