Insiders Guide to Mobile AppSec with OWASP MASVS

OWASP Meetup

Brian Reed, Chief Mobility Officer
br@nowsecure.com
@reed_on_the_run

NowSecure
12 years in Mobile Security
OWASP Sponsor & Contributor
Mobile AppSec Testing Tools, Training, Pen Testing
Creators of Frida and Radare

Remember when BlackBerry ruled the world?
Now I live on iOS, Droid, Apple Watch, Oura....

NowSecure, Good Technology, BlackBerry, ZeroFOX, BoxTone, and MicroFocus
OWASP Mobile Project Financial Sponsor & Contributor
NowSecure Security Researcher Carlos Holguera (@grepharder) is co-project lead for OWASP Mobile Project

OWASP MSTG Advocate
recognition for years of contributions

OWASP CycloneDX SBOM Contributor
NowSecure Founder Andrew Hoog on the CycloneDX leadership board
NowSecure IoXT Authorized Lab
Certify Mobile-Connected IoT devices

NowSecure ADA Authorized Lab
Independent Security Reviews for Google Play Data Safety
Open Source Community

**FRIDA**

Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers.

**Scriptable**

Inject your own scripts into black box processes. Hook any functions, spy on crypto APIs or trace private application code, no source code needed. Edit, hit save, and instantly see the results. All without compilation steps or program restarts.

**Portable**

Works on Windows, macOS, GitHub, iOS, Android, and QNX. Install the Node.js bindings from npm, grab a Python package from PyPI, or use Frida through its Swift bindings, .NET bindings, Go bindings, or C API. We also have a scalable framework.

**Free**

Frida is and will always be free software (free as in freedom). We want to empower the next generation of developer tools, and help other free software developers achieve interoperability through reverse engineering.

**Battle-tested**

We are proud that NowSecure is using Frida to do fast, deep analysis of mobile apps at scale. Frida has a comprehensive test-suite and has gone through years of rigorous testing across a broad range of use-cases.

© Copyright 2021 NowSecure, Inc. All Rights Reserved. Proprietary information. Do not distribute.
Peloton Responsible Disclosure from NowSecure

NowSecure researcher Austin Emmitt found and disclosed 4 vulnerabilities to Peloton mobile, web & APIs that have now been fixed:

1. Peloton user exposure to account takeover
2. Peloton user exposure to phishing attack
3. Remote access to Peloton users’ private personal info
4. Ability to remotely change device ID and serial number

There is NO evidence that any customers were breached

Read the two Blogs:

https://www.nowsecure.com/blog/2022/02/09/a-zero-click-rce-exploit-for-the-peloton-bike-and-also-every-other-unpatched-android-device/
Mobile Powers the World, But Mobile Risk is Pervasive

69% of all digital traffic & time spent is on mobile vs. web

200bn Mobile App Downloads in 2021

85% of Mobile Apps have security risks (Fail OWASP Mobile Top 10)

70% of Mobile Apps leak personal data to violate GDPR/CCPA

Sources: AppAnnie, March, 2020; Comscore, January 2020
Gartner, Avoid Mobile App Security Pitfalls, Zumerle, 27Jul2020
NowSecure Privacy Benchmark, 2019; NowSecure Security Benchmark 2020
What Mobile Apps Do You Use?

https://www.nowsecure.com/mobile-app-security-news/
Benchmark Trackers to Learn More

https://mobilerisktracker.nowsecure.com

Inside Mobile AppSec
Unique Characteristics of Mobile AppDev & AppSec

WEB VS MOBILE
98% of code behind perimeter with broad layered protection
Substantial code “in the wild”, running on untrusted device & easily reversible

- 2 Mobile OS with varying security capabilities
- 4 Dev Languages, Dozens of Frameworks, Thousands of libraries
- Continuous updates of Mobile OS and Dev tools
- Effective testing requires physical devices, not emulators
- Dynamic & APISec testing are challenging, but can be automated

The OWASP MASVS is here to help!
### OWASP Top 10 Industry Standards

<table>
<thead>
<tr>
<th>Mobile</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Improper Platform Usage</td>
<td>1. Broken Access Control</td>
</tr>
<tr>
<td>2. Insecure Data Storage</td>
<td>2. Cryptographic Failures</td>
</tr>
<tr>
<td>3. Insecure Communication</td>
<td>3. Injection</td>
</tr>
<tr>
<td>4. Insecure Authentication</td>
<td>4. Insecure Design</td>
</tr>
<tr>
<td>5. Insufficient Cryptography</td>
<td>5. Security Misconfiguration</td>
</tr>
<tr>
<td>8. Code Tampering</td>
<td>8. Software &amp; Data Integrity Failures</td>
</tr>
<tr>
<td>10. Extraneous Functionality</td>
<td>10. Server-Side Request Forgery</td>
</tr>
</tbody>
</table>

Mobile Attack Surface

- Leak
- Attack
- Leak
- Attack
- Leak
- Attack
- Leak
- Network & Cloud Services
- Data Center & App Backend

© Copyright 2021 NowSecure, Inc. All Rights Reserved. Proprietary information. Do not distribute.
# What’s Inside the Mobile Attack Surface?

## App Code
- App signing key unprotected
- Buffer overflow
- App Debuggable
- Configuration manipulation
- Missing User-input validation
- Insecure 3rd party libs
- Tampering/repacking possible
- No rooting/jailbreak detection
- No Code Obfuscation

## Data at Rest
- Sensitive Data caching
- Lack of keychain usage
- Sensitive Data in log files
- Sensitive Data in memory
- Sensitive Data in World Writable/Readable Files

## App Architecture
- Lack of Threat Modeling
- Insecure SDLC
- Bad Security Architecture
- Lack of Sensitive Data overview

## Data in Use
- Dynamic runtime injection
- Insecure URL schemes
- UI Data leaks
- Clipboard data leaks
- Unnecessary permissions

## Data in Motion
- Vulnerable to MITM attacks
- Vulnerable to session hijacking
- Improper TLS validation
- Weak App transport security
- Use of insecure protocols
- Insecure Cookies

## API Backends
- Unauthenticated APIs
- Unprotected APIs
- Excessive API Data
- API SQL Injection
- Remote code execution
- Privilege Escalation
- Denial of Service

## API Backends
- Unauthenticated APIs
- Unprotected APIs
- Excessive API Data
- API SQL Injection
- Remote code execution
- Privilege Escalation
- Denial of Service
Reduce the Attack Surface to Protect Sensitive Data

Focus on ensuring secure coding practices across all areas: app code, data storage, network communications and API backend.
Mobile App Security Testing
OWASP Mobile Security Project Resources

**MASVS**
Mobile Application Security Verification Standard

Establish security baseline for mobile apps

Latest Release: 2022

**MSTG**
Mobile Security Testing Guide

Cookbook for mobile app security testing

Latest Release: 2022

**Mobile Security Testing Checklist**

Checklist for mobile app security testing linking the MASVS to the MSTG

Latest Release: 2022
## MASVS Mobile AppSec Model

### MASVS L1
**Standard Security**
- The minimum
- No compliance or regulatory needs
- Simple apps

*Example: Healthcare WebMD App*

### MASVS L2
**Defense-in-Depth**
- Regulated industry data
- Compliance consideration
- Apps that perform simple tasks, but handled highly sensitive data.

*Example: Healthcare Weight Monitoring App*

### MASVS L1 + R
**Standard Security + High RE Resilience**
- Prioritize IP protection
- Prevent malicious modification or tampering

*Example: Medical Formulary App*

### MASVS L2 + R
**Defense-in-Depth + High RE Resilience**
- Apps that perform complex activities between users and handle high sensitive data
- Compliance and IP protection are key
- Preventing Malware based attacks is in your threat model

*Example: Healthcare Drug Delivery App*
Inside the MASVS Levels

<table>
<thead>
<tr>
<th>MASVS L1</th>
<th>MASVS L1 + R</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard Security</strong></td>
<td><strong>Standard Security + High RE Resilience</strong></td>
</tr>
<tr>
<td>- The minimum</td>
<td>- Prioritize IP protection</td>
</tr>
<tr>
<td>- No compliance or regulatory needs</td>
<td>- Prevent malicious modification or tampering</td>
</tr>
<tr>
<td>- Simple apps</td>
<td></td>
</tr>
<tr>
<td><strong>Example: Healthcare WebMD App</strong></td>
<td><strong>Example: Medical Formulary App</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MASVS L2</th>
<th>MASVS L2 + R</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Defense-in-Depth</strong></td>
<td><strong>Defense-in-Depth + High RE Resilience</strong></td>
</tr>
<tr>
<td>- Regulated industry data</td>
<td>- High sensitive operations &amp; data handling</td>
</tr>
<tr>
<td>- Compliance consideration</td>
<td>- Compliance and IP protection are key</td>
</tr>
<tr>
<td>- Apps that perform simple tasks, but handled highly sensitive data.</td>
<td>- Preventing Malware based attacks is in your threat model</td>
</tr>
<tr>
<td><strong>Example: Healthcare Weight Monitoring App</strong></td>
<td><strong>Example: Healthcare Drug Delivery App</strong></td>
</tr>
</tbody>
</table>

**R expects hardening**
- Device Binding
- Obfuscation
- Anti-Tamper
- Not meant to compensate for poor security
OWASP MASVS Addresses the Mobile Attack Surface

**App Code**
- App signing key unprotected
- Buffer overflow
- App Debuggable
- Configuration manipulation
- Missing User-input validation
- Insecure 3rd party libs
- Tampering/repacking possible
- No rooting/jailbreak detection
- No Code Obfuscation

**Data at Rest**
- Sensitive Data caching
- Lack of keychain usage
- Sensitive Data in log files
- Sensitive Data in memory
- Sensitive Data in World Writable/Readable Files

**App Architecture**
- Lack of Threat Modeling
- Insecure SDLC
- Bad Security Architecture
- Lack of Sensitive Data overview

**Data in Use**
- Dynamic runtime injection
- Insecure URL schemes
- UI Data leaks
- Clipboard data leaks
- Unnecessary permissions

**Data in Motion**
- Vulnerable to MITM attacks
- Vulnerable to session hijacking
- Improper TLS validation
- Weak App transport security
- Use of insecure protocols
- Insecure Cookies

**API Backends**
- Unauthenticated APIs
- Unprotected APIs
- Excessive API Data
- API SQL Injection
- Remote code execution
- Privilege Escalation
- Denial of Service
8 Domains of MASVS Requirements

V1: Architecture, Design and Threat Modeling
V2: Data Storage and Privacy
V3: Cryptography
V4: Authentication and Session Management
V5: Network Communication
V6: Environmental Interaction
V7: Code Quality and Build Setting
V8: Resiliency Against Reverse Engineering
Top 5 Areas To Focus
OWASP MASVS
Let’s Use Both the Builder and Breaker POV
1 - Insecure Data Storage & Crypto
50% of Mobile Apps have insecure data storage
# Insecure Data Storage & Crypto

<table>
<thead>
<tr>
<th>Security bug:</th>
<th>Use of the device file system without security controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack vector:</td>
<td>Malware, lost/stolen device, malicious USB charger</td>
</tr>
<tr>
<td>Business impact:</td>
<td>Identity theft, fraud, policy/compliance violation, data loss, reputational risk</td>
</tr>
</tbody>
</table>

**OWASP MASVS Mapping**
- V2: Data Storage & Privacy
- V3: Cryptography

**Resources:**
- OWASP MASVS V2: Insecure Data Storage
- OWASP MASVS V3: Cryptography
- Android: Data and file storage overview
- Apple: File system basics
Insecure Data Storage & Weak Crypto

Best Practices for Secure Coding
- Avoid writing sensitive data to device
- Encrypt sensitive files
- Avoid query strings in sensitive data
- Implement secure data storage
- Use strong current Crypto (e.g. SHA3)
- Use SecureRandom
- Use Key with a length of at least 2048 bits (preferably 4096 bits)

Best Practices for AppSec Testing
- Test for credentials & PII in files, logs, IPC
- Test for data removed when background
- Test Crypto libs & storage
- Confirm req use of device password
- Check for weak crypto & bad practices
2- Insecure Network Communication
Insecure Network Communication

48% of Mobile Apps have insecure data communication
## Insecure Network Communication

### OWASP MASVS Mapping

- V5: Network Communication

### Security bug:

Unprotected network communications (e.g., use of HTTP, lack of TLS validations)

### Attack vector:

Malicious VPN, exploited networks, public Wi-Fi

### Business impact:

Identity theft, fraud, reputational risk

### Resources:

- [OWASP MASVS V5: Network Comms](#)
- [Android: Network security configuration](#)
- [Apple: Preventing insecure network connection](#)
Improperly Coded Network Calls

Best Practices for Secure Coding

- Only generate TLS sessions after a successful trust evaluation and a valid DNS name
- Perform certificate pinning for connections carrying regulated data
- Leverage iOS App Transport Security and Android Network Security Configuration
- Learn about how to prevent man-in-the-middle attacks

Best Practices for AppSec Testing

- Test TLS, Cert Pinning, zip files in transit
- Check for use of ATS & NSC
- Check 3rd party libraries
3- Insecure Authentication or Authorization
Insecure Authentication or Authorization

14% of Mobile Apps have insecure authentication

Username
Password
Login
Insecure Authentication or Authorization

Security bug:
Improper authentication scheme (e.g., weak password acceptance), design flaws in session management or authorization scheme (e.g., flaws in user’s privilege level, authorization permissions provided through the client-side code)

Attack vector:
API endpoints, stolen device

Business impact:
Unauthorized access, theft, and reputational risk

Resources:
- OWASP MASVS V4: Auth & Session Mgmt
- Android: Authenticate Users
- Apple: User Authentication

OWASP MASVS Mapping
- V4: Authentication & Session Mgmt
Insecure Authentication or Authorization

Best Practices for Secure Coding

- Terminate the active session after a given amount of time
- Ensure no app data is visible when session is invalidated
- Discard and clear all memory associated with the user data and encryption
- Run authorization checks for roles and permissions of an authenticated user at the server, not client side

Best Practices for AppSec Testing

- Test session validation
- Test data in memory
4- Insecure Coding Practices
Insecure Coding Practices

47% of Mobile Apps have insecure exploitable extraneous functionality
Insecure Coding Practices

OWASP MASVS Mapping

- V7: Code Quality & Build Setting Requirements

Security bug:
Issue as a result of poor coding practices (e.g., logic flaws in code, vulnerable third-party library, buffer overflows and memory leaks), unnecessary component built into app (e.g., debug features, security controls)

Attack vector:
Malware, phishing, unsuspected user, extraneous func. feature

Business impact:
Data theft, reputational risk, fraud, unauthorized access

Resources:

- OWASP MASVS V7: Code Quality

© Copyright 2021 NowSecure, Inc. All Rights Reserved. Proprietary information. Do not distribute.
Insecure Coding Practices

Best Practices

- Remove Debug symbols & code
- Ensure Secure Coding practices
- Use free security features offered by the toolchain (stack protection, ARC, etc.)
- Keep track of 3rd party dependencies with an SBOM! Scan for well-known vulnerabilities

Best Practices for AppSec Testing

- Test app signed with valid cert
- Test for debug build, hardcoded keys
- Test error conditions, verbose log files
- Check 3rd party libraries
5- Reverse Engineering & Anti-Tampering
Exposure to Reverse Engineering

32% of Mobile Apps have exposure to reverse engineering
Reverse Engineering

OWASP MASVS Mapping

- V8: Resiliency Against Reverse Engineering & Tampering

Resources:

- OWASP MASVS V8: Resiliency
- OWASP Reversing Prevention Project
- Reversing tools: Frida, Radare, 2Frida Repo

<table>
<thead>
<tr>
<th>Security bug:</th>
<th>Unprotected IP and binary enables attackers to reverse engineer process and data to exploit in other ways</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack vector:</td>
<td>Reverse engineering of mobile app binary</td>
</tr>
<tr>
<td>Business impact:</td>
<td>Data theft, IP theft, reputational risk, fraud, unauthorized access</td>
</tr>
</tbody>
</table>
Exposure to Reverse Engineering

Best Practices for Secure Coding
- Use third-party code obfuscation tools, especially for Android apps
- Use Android SafetyNet API to check for Android device tampering
- Implement anti-tampering techniques

Best Practices for AppSec Testing
- Test for reversibility via detect JB/root, debugger, data/file manipulation
- Test String tables & methods
- Check for Android SafetyNet API
Resiliency Against Reverse Engineering & Tampering

Testing Tip

Tamper proofing helps, but only so far...

“Anti tampering doesn’t fix security bugs, or protect security bugs in production code...”
Key Takeaways

Recognize Mobile & Web are different
Get to know the OWASP Mobile Project
Start exploring, leverage the great resources!
Build your skills and toolkit
Threat modeling is your friend
The 8 Requirements help break down the problem
Start with the Big 5 (storage, network, auth, code, RE)
Get involved in the OWASP Mobile Project - Sign Up!
OWASP MASVS Project Updates
OWASP MASVS V2 Refactoring Process Update

https://github.com/OWASP/owasp-masvs/discussions/categories/big-masvs-refactoring
OWASP MASVS Refactoring Process

### Key Areas

- **Code** /App Package
- **Internal**
- **External**
- **KeyStore** or SE/StrongBox
- **logs**
- **Memory**
- **All Data**

### Controls

- **Data at rest**
  - App package
  - Source code / app binaries
  - Internal caches
  - External Caches
  - Libraries
  - Non-key Data in KeyChain (iOS)
  - Explicit User consent
  - Data Encrypt. Keys well-protected? + CRYPTO-3
  - Scoped Storage
  - Encrypted Strong Data Prot. EncryptedFile
  - Encrypted KEK+DEK, KDF, encr. PDF
  - Encrypted if not in KeyStore
  - Auth Data & Crypto Material only in KeyStore
  - Uses SE/StrongBox

- **Data in use**
  - No sensitive data in logs
  - Short time
  - Zeroed
  - Decrypt only before use

- **Data Privacy**
  - Not shared/coll ect
  - Unless declared

### Tests

- **MASVS-STORAGE-1**
- **MASVS-STORAGE-2**
- **MASVS-STORAGE-3**
- **MASVS-STORAGE-4**
- **MASVS-STORAGE-5**
OWASP MASVS V2 Compliance-as-Code

Human + excel/PDF/Word

- Read and interpret manually
- Hard to prove control and test coverage
- Compare providers manually
- Hard to maintain

Automation + yaml/json/xml

- Machine-readable
- Easy to prove control and test coverage
- Compare providers with benchmarking
- Fully traceable

Standard and fully tailored testing

MASVS + proprietary + cross-standards

MASVS provided

Privacy

Automation-friendly

IoT

Health

Community created

L1

L2

R
Join Our OWASP Project Team

Contribute & connect with us!

https://github.com/OWASP/owasp-mstg#connect-with-us
Apple and Google Updates
Apple Privacy And Google Play Data Safety
By July 20th 2022, the Data safety section for all your apps must be approved.

Thanks to Google's App Defense Alliance (ADA), Developers can showcase key privacy and security practices, at a glance.
ADA Mobile App Security Assessments (MASA)

MASA has a published formal set of requirements based on OWASP MASVS and MSTG.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG-STORAGE-1</td>
<td>System credential storage facilities used to store sensitive data.</td>
</tr>
<tr>
<td>MSG-CRYPTO-1</td>
<td>App does not rely on symmetric cryptography with hardcoded keys.</td>
</tr>
<tr>
<td>MSG-AUTH-1</td>
<td>Authentication for remote services.</td>
</tr>
<tr>
<td>MSG-NETWORK-1</td>
<td>Data is encrypted on the network using TLS.</td>
</tr>
<tr>
<td>MSG-PLATFORM-1</td>
<td>App requests the minimum set of permissions.</td>
</tr>
<tr>
<td>MSG-CODE-1</td>
<td>App is signed and provisioned with a valid certificate.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG-STORAGE-2</td>
<td>No sensitive data should be stored outside of the app container.</td>
</tr>
<tr>
<td>MSG-CRYPTO-2</td>
<td>Proven implementations of cryptographic primitives.</td>
</tr>
<tr>
<td>MSG-AUTH-2</td>
<td>Randomly-generated session identifiers.</td>
</tr>
<tr>
<td>MSG-NETWORK-2</td>
<td>The TLS settings are in line with current best practices.</td>
</tr>
<tr>
<td>MSG-PLATFORM-2</td>
<td>Inputs from external sources and the user are validated.</td>
</tr>
<tr>
<td>MSG-CODE-2</td>
<td>App has been built in secure modes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG-STORAGE-3</td>
<td>No sensitive data is written to application logs.</td>
</tr>
<tr>
<td>MSG-CRYPTO-3</td>
<td>App uses cryptographic primitives that are appropriate for the particular use case.</td>
</tr>
<tr>
<td>MSG-AUTH-3</td>
<td>Session token-based authentication are signed.</td>
</tr>
<tr>
<td>MSG-NETWORK-3</td>
<td>The app verifies the X.509 certificate of the remote endpoint.</td>
</tr>
<tr>
<td>MSG-PLATFORM-3</td>
<td>App does not accept sensitive functionality via custom URL schemes.</td>
</tr>
<tr>
<td>MSG-CODE-3</td>
<td>Debugging symbols have been removed from native binaries.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG-STORAGE-4</td>
<td>The keyboard cache is disabled on sensitive data inputs.</td>
</tr>
<tr>
<td>MSG-CRYPTO-4</td>
<td>App does not export cryptographic protocols or algorithms.</td>
</tr>
<tr>
<td>MSG-AUTH-4</td>
<td>Remote endpoint terminates the existing session when the user logs out.</td>
</tr>
<tr>
<td>MSG-PLATFORM-4</td>
<td>App does not export sensitive functionality through IPV4 facilities.</td>
</tr>
<tr>
<td>MSG-CODE-4</td>
<td>Debugging symbols and developer assistance tools have been removed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG-STORAGE-5</td>
<td>No sensitive data is exposed through the user interface.</td>
</tr>
<tr>
<td>MSG-CRYPTO-5</td>
<td>No re-use the same cryptographic key for multiple purposes.</td>
</tr>
<tr>
<td>MSG-AUTH-5</td>
<td>Password policy is enforced at the remote endpoint.</td>
</tr>
<tr>
<td>MSG-PLATFORM-5</td>
<td>Third party components are checked for known vulnerabilities.</td>
</tr>
<tr>
<td>MSG-CODE-5</td>
<td>Security features enforced by the blockchain are achieved.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG-STORAGE-12</td>
<td>Educate the user about the types of personally identifiable information processed.</td>
</tr>
<tr>
<td>MSG-CRYPTO-12</td>
<td>Random values are generated using a sufficient secure random number generator.</td>
</tr>
<tr>
<td>MSG-AUTH-12</td>
<td>Limited force mitigations.</td>
</tr>
<tr>
<td>MSG-PLATFORM-12</td>
<td>Sessions are invalidated at the remote endpoint after a predefined period of inactivity.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG-CODE-12</td>
<td></td>
</tr>
</tbody>
</table>
OWASP CycloneDX for SBOM
What is OWASP CycloneDX?

New Flagship Project at OWASP

A new industry standard for SBOM interoperability

Chaired by Steve Springett & Patrick Dwyer

“The CycloneDX SBOM standard is a result of security experts and industry coming together to create an SBOM standard that delivers the transparency and interoperability necessary to communicate software inventory and the relationships across different systems.”

Cross links with OWASP MASVS Project as well

Link to Dependency Track SBOM tool

https://dependencytrack.org/

https://owasp.org/www-project-cyclonedx/
What is OWASP CycloneDX?

https://owasp.org/www-project-cyclonedx/

Get Free Mobile SBOMS
Resources Resources Resources Resources
Mobile Pen Tester’s Toolkit

Manual & OSS Testing Resources

- MASVS repo
- MSTG repo
- MSTG Hacking Playground
- Frida Dynamic Instrumentation Toolkit
- Radare Portable Reversing Framework
- Burp Suite or ZedAttackProxy
- Jailbroken & Rooted devices

Automated Testing Resources

- Free Mobile SBOMs
- Free Mobile Analysis Report
- Free Online Training Academy
- NowSecure Workstation Toolkit
- NowSecure Platform Automation
  - ✓ 600+ security, privacy and compliance tests
  - ✓ SAST+DAST+IAST+APISec
  - ✓ Automated & Interactive Modes
  - ✓ Embedded remediation
Best Practice Tuning Security Test Coverage & Frequency

- **High Risk**
  - Automated Continuous Testing
  - Frequent Guided & Expert Pen Testing

- **Medium Risk**
  - Automated Continuous Testing
  - Periodic Guided & Expert Pen Testing

- **Low Risk**
  - Periodic Automated Testing
Free Training

Online Courseware
https://academy.nowsecure.com

Full Replays
Checkout Your Own Mobile Apps

Free SBOM

Free Security Report
More Free Resources


OWASP Android CrackeMe r2Comm
NowSecure Full Mobile AppSec Solution Suite

NowSecure Platform
Continuous security testing for mobile DevSecOps

NowSecure Supply Chain
Continuous monitoring of app store mobile risk

NowSecure Workstation
All-in-one mobile pen tester toolkit for productivity

NowSecure Academy
Online courseware and certification for mobile

NowSecure Pen Testing
Expert full scope mobile pen testing services & remediation

NowSecure Mobileverse™
Customer community to onboard, learn & network with peers
Password Exposed and Modifiable Over the Network

Context

Description
Password was intercepted over HTTP traffic.
A remote attacker with access to the local or upstream network as the user could use network monitoring software, such as Wireshark, to observe and modify the data.

Steps To Reproduce
Use a packet interception and analysis tool, such as Wireshark, on your testing network to identify unencrypted network traffic that may contain sensitive information.
NowSecure's test for this finding involves capturing HTTP traffic between an app running on a physical device and servers. The resulting HTTP traffic is examined for the presence of credential information which results in a list of credentials leaked to servers over insecure HTTP communications.

Business Impact
The app is not encrypting sensitive information being sent over the internet. A malicious actor could remotely see and/or modify the sensitive data coming to and from the endpoints listed, potentially affecting many users at once. Depending on the type of data being transmitted insecurely, this vulnerability could lead to exposure of sensitive personal data or intellectual property.

Evidence

74 Results
<table>
<thead>
<tr>
<th>Domain</th>
<th>Host</th>
<th>IP</th>
<th>Port</th>
<th>Organization</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>taobao.com</td>
<td>ptbslog.umeng.com</td>
<td>::ffff:106.11.223.204</td>
<td>443</td>
<td>Zhejiang Taobao Network Co., Ltd</td>
<td>Hangzhou, Zhejiang, CN</td>
</tr>
<tr>
<td>amazon.com</td>
<td>smart.kebijia.com</td>
<td>::ffff:52.1.231.197</td>
<td>80</td>
<td>Amazon Technologies Inc.</td>
<td>Ashburn, Virginia, US</td>
</tr>
<tr>
<td>ubistor.com</td>
<td>easymessage.com, s_jpush.cn, sis_jpush.io</td>
<td>::ffff:103.230.236.25</td>
<td>19000</td>
<td>XIAMEN CenturyNetcomNetwork Services Limited</td>
<td>Xiamen, Fujian, CN</td>
</tr>
<tr>
<td>huawei.com</td>
<td>122.9.121.124</td>
<td>::ffff:122.9.121.124</td>
<td>7000</td>
<td>Huawei Public Cloud Service</td>
<td>Guangzhou, Guangdong, CN</td>
</tr>
<tr>
<td>huawei.com</td>
<td>bjuser.jpush.cn</td>
<td>::ffff:122.9.15.248</td>
<td>443</td>
<td>Huawei Public Cloud Service</td>
<td>Guangzhou, Guangdong, CN</td>
</tr>
<tr>
<td>chinamobilelttd.com</td>
<td>183.232.25.163</td>
<td>::ffff:183.232.25.163</td>
<td>21004</td>
<td>China Mobile Communications Corporation</td>
<td>Guangzhou, Guangdong, CN</td>
</tr>
<tr>
<td>huawei.com</td>
<td>121.36.205.81</td>
<td>::ffff:121.36.205.81</td>
<td>7000</td>
<td>Huawei Public Cloud Service</td>
<td>Guangzhou, Guangdong, CN</td>
</tr>
<tr>
<td>chinamobilelttd.com</td>
<td>183.232.58.113</td>
<td>::ffff:183.232.58.113</td>
<td>21003</td>
<td>China Mobile Communications Corporation</td>
<td>Guangzhou, Guangdong, CN</td>
</tr>
<tr>
<td>huawei.com</td>
<td>121.36.75.206</td>
<td>::ffff:121.36.75.206</td>
<td>7006</td>
<td>Huawei Public Cloud Service</td>
<td>Beijing, Beijing, CN</td>
</tr>
</tbody>
</table>
Outdated nanopb Library Contains Known Security Flaw

Context
The application was found to be using a vulnerable version of the nanopb library. The library does not properly validate information that it processes which can lead to unintended access or potentially malicious code being run. This test specifically checks for versions < 2.30908.0 as cited by CVE-2021-21401.

Business Impact
The app is using a 3rd party library which contains a known, high risk flaw which could expose the application and its users to severe attacks.

Evidence

<table>
<thead>
<tr>
<th>Version</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.30907.0</td>
<td>Payload/pMп COVID-19.app/Frameworks/nanopb.framework/Info.plist</td>
</tr>
</tbody>
</table>
## Package Details

### Versions

<table>
<thead>
<tr>
<th>Version</th>
<th>Build</th>
<th>Origin</th>
<th>Security Score</th>
<th>Assessments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4.0</td>
<td>1.4.0-1031.3.40</td>
<td>App Store</td>
<td>Poor - 48</td>
<td>1</td>
</tr>
<tr>
<td>1.3.0</td>
<td>1.3.0-1029.3.30</td>
<td>App Store</td>
<td>Good - 75</td>
<td>1</td>
</tr>
<tr>
<td>1.2.2</td>
<td>1.2.2-1028.3.22</td>
<td>App Store</td>
<td>Good - 77</td>
<td>1</td>
</tr>
<tr>
<td>1.2.1</td>
<td>1.2.1-1028.3.21</td>
<td>App Store</td>
<td>Partial Results</td>
<td>1</td>
</tr>
<tr>
<td>1.2.0</td>
<td>1.2.0-1027.3.20</td>
<td>App Store</td>
<td>Partial Results</td>
<td>1</td>
</tr>
<tr>
<td>1.1.5</td>
<td>1.1.5-1022.3.00</td>
<td>App Store</td>
<td>Good - 76</td>
<td>1</td>
</tr>
<tr>
<td>1.1.4</td>
<td>1.1.4-1021.2.99</td>
<td>App Store</td>
<td>Good - 76</td>
<td>1</td>
</tr>
<tr>
<td>1.1.3</td>
<td>1.1.3-1020.2.98</td>
<td>App Store</td>
<td>Good - 76</td>
<td>1</td>
</tr>
<tr>
<td>1.1.2</td>
<td>1.1.2-1019.2.96</td>
<td>App Store</td>
<td>Good - 76</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1</td>
<td>1.1.1-1018.2.93</td>
<td>App Store</td>
<td>Good - 77</td>
<td>2</td>
</tr>
</tbody>
</table>
Sample Automated Workflow: Build, Test, Ticket, Repair

- Faster Feedback Loops
- Faster Mean Time to Repair
- Lower Defect Escape Rate
NowSecure GitHub Action for Developer-First Mobile AppSec Testing

NowSecure™

All-in-one SAST+DAST+IAST

Security test any iOS/Android app binary, any language

Supports Kotlin, Java, Swift, ObjectiveC & More

Analyzes all code including 3rd party SDKs & transitive dependencies

Tickets w/ embedded dev guide & sample code to fix fast

NowSecure GitHub Action Resources

Demo Video Link
GitHub Action Link
THANK YOU!

OWASP Meetup

Brian Reed, Chief Mobility Officer
br@nowsecure.com
@reed_on_the_run