



Mobile AppSec Verification

Version 1.1
(Chinese Translation)

Project leaders: Sven Schleier and Jeroen Willemsen

Creative Commons (CC) Attribution Share-Alike

Free version at <http://www.owasp.org>



Table of Contents

Changelog	1.1
前言	1.2
卷首	1.3
使用 MASVS	1.4
評估與認證	1.5

安全要求

V1: 架構、設計和威脅模型分析準則	2.1
V2: 資料存儲和隱私要求	2.2
V3: 加密要求	2.3
V4: 身份驗證和對談管理準則	2.4
V5: 網路通訊規範	2.5
V6: 平台互動要求	2.6
V7: 程式碼品質和與建立設定要求	2.7
V8: 彈性要求	2.8

附錄

附錄 A - 詞彙表	3.1
附錄 B - 參考資料	3.2

Changelog

This document is automatically generated at Tue Jan 01 2019 21:19:17 GMT+0100 (Midden-Europese standaardtijd)

WIP - 1.1.2 Sponsorship and internationalization

The following changes are part of release 1.1.2:

- Added thank you note for buyers of the e-book.
- Added missing authentication link & updated broken authentication link in V4.
- Fixed swap of 4.7 and 4.8 in english.
- First international release!
 - Fixes in Spanish translation. Translation is now in sync with English (1.1.2).
 - Fixes in Russian translation. Translation is now in sync with English (1.1.2).
 - Added first release of Chinese (ZHTW) French, German, and Japanese!
- Simplified document for ease of translation.
- Added instructions for automated releases.

1.1.0 14 July 2018:

The following changes are part of release 1.1:

- Requirement 2.6 "The clipboard is deactivated on text fields that may contain sensitive data." was removed.
- Requirement 2.2 "No sensitive data should be stored outside of the app container or system credential storage facilities." was added.
- Requirement 2.1 was reworded to "System credential storage facilities are used appropriately to store sensitive data, such as PII, user credentials or cryptographic keys.".

1.0 12 January 2018:

The following changes are part of release 1.0:

- Delete 8.9 as the same as 8.12
- Made 4.6 more generic
- Minor fixes (typos etc.)

Foreword

By Bernhard Mueller, OWASP Mobile Project. Technological revolutions can happen quickly. Less than a decade ago, smartphones were clunky devices with little keyboards - expensive playthings for tech-savvy business users. Today, smartphones are an essential part of our lives. We've come to rely on them for information, navigation and communication, and they are ubiquitous both in business and in our social lives.

Every new technology introduces new security risks, and keeping up with those changes is one of the main challenges the security industry faces. The defensive side is always a few steps behind. For example, the default reflex for many was to apply old ways of doing things: Smartphones are like small computers, and mobile apps are just like classic software, so surely the security requirements are similar? But it doesn't work like that. Smartphone operating systems are different from Desktop operating systems, and mobile apps are different from web apps. For example, the classical method of signature-based virus scanning doesn't make sense in modern mobile OS environments: Not only is it incompatible with the mobile app distribution model, it's also technically impossible due to sandboxing restrictions. Also, some vulnerability classes, such as buffer overflows and XSS issues, are less relevant in the context of run-of-the-mill mobile apps than in, say, Desktop apps and web applications (exceptions apply).

Over time, our industry has gotten a better grip on the mobile threat landscape. As it turns out, mobile security is all about data protection: Apps store our personal information, pictures, recordings, notes, account data, business information, location and much more. They act as clients that connect us to services we use on a daily basis, and as communications hubs that processes each and every message we exchange with others. Compromise a person's smartphone and you get unfiltered access to that person's life. When we consider that mobile devices are more readily lost or stolen and mobile malware is on the rise, the need for data protection becomes even more apparent.

A security standard for mobile apps must therefore focus on how mobile apps handle, store and protect sensitive information. Even though modern mobile operating systems like iOS and Android offer good APIs for secure data storage and communication, those have to be implemented and used correctly in order to be effective. Data storage, inter-app communication, proper usage of cryptographic APIs and secure network communication are only some of the aspects that require careful consideration.

An important question in need of industry consensus is how far exactly one should go in protecting the confidentiality and integrity of data. For example, most of us would agree that a mobile app should verify the server certificate in a TLS exchange. But what about SSL certificate pinning? Does not doing it result in a vulnerability? Should this be a requirement if an app handles sensitive data, or is it maybe even counter-productive? Do we need to encrypt data stored in SQLite databases, even though the OS sandboxes the app? What is appropriate for one app might be unrealistic for another. The MASVS is an attempt to standardize these requirements using verification levels that fit different threat scenarios.

Furthermore, the appearance of root malware and remote administration tools has created awareness of the fact that mobile operating systems themselves have exploitable flaws, so containerization strategies are increasingly used to afford additional protection to sensitive data and prevent client-side tampering. This is where things get complicated. Hardware-backed security features and OS-level containerization solutions, such as Android for Work and Samsung Knox, do exist, but they aren't consistently available across different devices. As a band aid, it is possible to implement software-based protection measures - but unfortunately, there are no standards or testing processes for verifying these kinds of protections.

As a result, mobile app security testing reports are all over the place: For example, some testers report a lack of obfuscation or root detection in an Android app as "security flaw". On the other hand, measures like string encryption, debugger detection or control flow obfuscation aren't considered mandatory. However, this binary way of looking at things doesn't make sense because resiliency is not a binary proposition: It depends on the particular client-side threats one aims to defend against. Software protections are not useless, but they can ultimately be bypassed, so they must never be used as a replacement for security controls.

The overall goal of the MASVS is to offer a baseline for mobile application security (MASVS- L1), while also allowing for the inclusion of defense-in-depth measures (MASVS-L2) and protections against client-side threats (MASVS-R). The MASVS is meant to achieve the following:

- Provide requirements for software architects and developers seeking to develop secure mobile applications;
- Offer an industry standard that can be tested against in mobile app security reviews;
- Clarify the role of software protection mechanisms in mobile security and provide requirements to verify their effectiveness;
- Provide specific recommendations as to what level of security is recommended for different use-cases.

We are aware that 100% industry consensus is impossible to achieve. Nevertheless, we hope that the MASVS is useful in providing guidance throughout all phases of mobile app development and testing. As an open source standard, the MASVS will evolve over time, and we welcome any contributions and suggestions.



OWASP

Mobile Security Project

Mobile Application Security Verification Standard

關於本標準

歡迎參閱 Mobile Application Security Verification Standard (MASVS) 1.1 版。MASVS 是一個社群的成果，目的在於建立一個安全需求的架構給人們去設計、開發和測試 iOS 跟 Android 的安全行動應用程式。

MASVS 是結合社群的努力和業界意見反饋來訂定的標準。我們期望這份標準能夠隨著時間演進不斷更新，也十分歡迎來自社群的意見反饋。透過 OWASP Mobile Project 的 Slack channel 與我們連絡是聯絡我們的最佳方式，以下是我們的 Slack channel 連結: https://owasp.slack.com/messages/project-mobile_omtg/details/

可以從以下的網址創建帳號：<http://owasp.herokuapp.com/>。

版權和許可



Copyright © 2018 The OWASP Foundation. 本文檔在 Creative Commons

Attribution ShareAlike 3.0 協議許可下發布。對於任何二次使用或發布，你必須向其他人說明清楚這項成果的版權。

致謝

項目負責人	主要作者	貢獻者和校稿
Sven Schleier & Jeroen Willemsen	Bernhard Mueller	Alexander Antukh, Mesheryakov Aleksey, Bachevsky Artem, Jeroen Beckers, Vladislav Chelnokov, Ben Cheney, Peter Chi, Lex Chien, Stephen Corbiaux, Manuel Delgado, Ratchenko Denis, Ryan Dewhurst, Tereshin Dmitry, Christian Dong, Oprya Egor, Ben Gardiner, Rocco Gränitz, Henry Hu, Sjoerd Langkemper, Vinícius Henrique Marangoni, Martin Marsicano, Roberto Martelloni, Gall Maxim, Rio Okada, Abhinav Sejpal, Stefaan Seys, Yogesh Sharmra, Prabhant Singh, Nikhil Soni, Anant Shrivastava, Francesco Stillavato, Romuald SZKUDLAREK, Abdessamad Temmar, Koki Takeyama, Chelnokov Vladislav, Leo Wang

本文檔最初是 Jim Manico 撰寫的 OWASP Application Security Verification Standard 的一個分支。

贊助者

雖然 MASVS 和 MSTG 都是由社群自願創建和維護的，但有時仍需要一些外部的協助。因此，我們感謝我們的贊助者提供資金來聘請技術編輯。請注意，他們的贊助不會以任何方式影響 MASVS 或 MSTG 的內容。贊助方案可在[OWASP Project Wiki](#)找到。

特別感謝



接下來，我們要感謝OWASP灣區分會的贊助。最後，我們要感謝所有從Leanpub購買這本書並以這種方式贊助我們的人。

行動應用資訊安全驗證標準(Mobile Application Security Verification Standard)

MASVS可以被用來確立一個行動應用Application的資訊安全程度。本標準(MASVS)的建立是遵循下列三個主要目標：

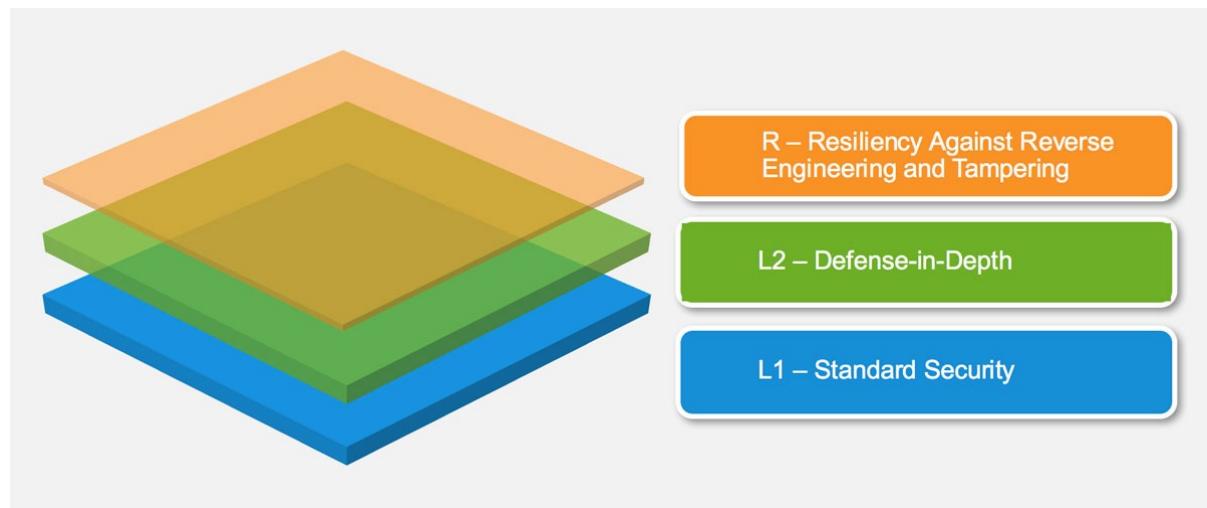
- 作為業界應用的資安標準 - 提供一個資訊安全標準，讓開發者和使用者們可以比較現有的行動應用之資訊安全程度。
- 作為軟體開發的指導綱領 - 提供一個行動應用軟體開發流程中各個步驟的指引，以便行動應用的開發與測試。
- 作為採購檢驗的驗證參考 - 提供一個行動應用資訊安全的驗證基準。

行動應用的資訊安全模型(Mobile AppSec Model)

MASVS定義了兩個嚴格的資訊安全驗證等級(L1和L2)，還有一系列的抗逆向工程防禦韌性準則(MASVS-R)，抗逆向工程準則會根據各種應用的不同威脅模型有相對應的調整。MASVS-L1和MASVS-L2包含通用的資訊安全準則，MASVS-L1是建議所有的行動應用都應該遵循的要求，而MASVS-L2則是針對處理高機敏資訊的行動應用。與此同時，MASVS-R則提供了需要防範用戶端威脅(client-side threat)時，可以使用的額外防護控管方法。

落實MASVS-L1可以確保行動應用的安全，使開發流程符合資訊安全的最佳實踐方法，並且確保行動應用可以防範常見的資訊安全弱點。MASVS-L2則是增加了更多的防禦縱深(defense-in-depth)，例如憑證綁定(SSL pinning)。因此，MASVS-L2可以使行動應用能夠更好的防範複雜的攻擊行為 - 假設行動裝置作業系統的資訊安全管控完好，並且用戶端使用者並不被視為潛在的惡意用戶。落實全部或部分MASVS-R的軟體防護要求，可以幫助防止當用戶端使用者是惡意使用者或行動裝置作業系統被入侵時的特定用戶端威脅(client-side threats)。

請注意MASVS-R和OWASP Mobile Testing Guide(OWASP行動應用測試指南)中提到的軟體防護管控方法，無法確保不會被繞過，並且永遠不應該被視為資訊安全管控的替代方法。相反的，這些軟體防護管控方法旨在增加已落實MASVS L1或L2的行動應用，對特定威脅的額外防護。



文檔結構

MASVS的第一個部分先闡述行動應用的資訊安全模型和可用的安全驗證等級，接著提供如何使用本標準的建議。安全準則的細節和對應的安全驗證等級，則是在第二部分。本標準中的準則，根據技術的目的/範疇，被分類成8個類別(從V1到V8)。準則類別與條目的命名方法，如下所示(MASVS和MSTG都使用相同的命名邏輯)：

- 準則類別: MASVS-Vx, 例如: MASVS-V2: 資訊存儲與隱私
- 準則條目: MASVS-Vx.y, 例如: MASVS-V2.2: "機敏資訊不應該被寫進應用程式紀錄檔(application logs)"

安全驗證等級細節

MASVS-L1: 標準安全等級

達到MASVS-L1的行動應用需要符合資訊安全的最佳實踐方法(best practices)。它包含了程式碼品質、處理機敏資訊和與行動環境交互的基本安全準則。另外，MASVS-L1也要求必須要有驗證資訊安全控管的測試流程。MASVS-L1(標準安全等級)適用於所有的行動應用程式。

MASVS-L2: 防禦縱深(加強安全等級)

MASVS-L2引入了高於標準安全等級的進階安全控管方法。要落實MASVS-L2，首先必須要有一個威脅模型分析，並且把資訊安全視為行動應用的不可分割部分，加入到應用的架構和設計之中。加強安全等級適用於處理機敏資訊的應用程式，例如：行動銀行。

MASVS-R: 抗逆向工程和竄改的韌性

符合MASVS-R的應用程式必須應用最新資訊安全技術，並且有足夠的韌性來抵抗明確定義的特定用戶端攻擊，例如資訊竄改，程序修改或逆向工程來提取機敏的資訊或程式碼。這樣的應用程式需要使用硬體安全模塊或者是被驗證過的高安全性軟體防護技術。MASVS-R適用於處理高機敏資訊的應用程式，並且可以被用來當作一種保護智慧財產的方法或防止竄改應用程式的方法。

使用建議

根據風險評估和應用程式的安全等級需求，應用程式可以被驗證為符合MASVS-L1或L2。L1適用於所有行動應用，而L2通常建議需要處理機敏資訊或功能的行動應用遵循。MASVS-R(或其一部分)可以被用於驗證行動應用抵抗特定威脅的韌性，例如：重封裝或提取機敏資訊。另外，也可以被用於更嚴謹的資訊安全驗證。

總而言之，可行的驗證類別組合，如下所示：

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

不同的驗證類別組合反映出不同的資訊安全等級和韌性。這樣的組合方式是為了保留一些彈性，例如：手機遊戲程式因為高可用性的需求，可能不須加入MASVS-L2的安全控管機制，如：雙因子驗證，但相對的手機遊戲程式對防止竄改的安全控管，通常會有很高的需求。

如何選擇安全驗證等級和準則

落實MASVS-L2的安全準則可以增進資訊安全，但是也會同時增加開發成本以及可能降低使用者體驗。一般來說，決定是否採用L2來開發應用，取決於其風險與成本考量(也就是說，當因為應用程式的保密性或完整性被破壞導致的潛在損失比增加額外安全控管機制的成本高時，會考慮採用L2)。因此，風險評估應該被視為是應用MASVS之前的第一步。

範例

MASVS-L1

- 適用於所有行動應用程式。MASVS-L1列舉了可以不過分影響開發成本和使用者體驗的資訊安全最佳實踐方法。任何不適用於更嚴格安全等級的行動應用程式，應該採用MASVS-L1的準則。

MASVS-L2

- **醫療保健產業:** 儲存個人可識別資訊(personally identifiable information)的行動應用程式，建議採用MASVS-L2，因為個人可識別資訊是機敏資訊，可能被用於冒用身份，詐騙取財或其他各種欺騙行為。對於美國的醫療保健行業，需要遵循相關的資訊保密法規 - 健康保險便利和責任法案(HIPAA) 中的Privacy, Security, Breach Notification Rules and Patient Safety Rule。
- **金融產業:** 可以存取高機敏資訊(例如: 信用卡號碼，個人資訊或轉帳功能)的行動應用程式，建議採用MASVS-L2。這些行動應用程式必須確保有額外的資訊安全控管機制來防止詐騙。金融相關應用程式須確保遵循支付卡產業資料安全標準(PCI DSS)，金融服務法現代化法案(Gramm Leech Bliley Act)以及薩班斯-奧克斯利法案 (SOX)。

MASVS L1+R

- 以智慧財產的保護為商業目的之一的行動應用程式，建議採用MASVS L1+R。MASVS-R中列舉的安全控管機制可以被用於增加攻擊者獲取原始程式碼，竄改或破壞程式的難度。
- **遊戲產業:** 對於防止竄改程式和作弊，遊戲應用程式有強烈的需求。遊戲作弊行為對線上遊戲來說，是一個很重要的議題，因為遊戲作弊者的存在，會引起玩家群體的不滿，並且最終導致整個遊戲的失敗。MASVS-R提供了基礎的防篡改機制來幫助減小遊戲作弊者出現的可能性。

MASVS L2+R

- **金融產業:** 允許使用者做轉帳操作的行動銀行應用程式，有程式碼注入和在被入侵的裝置上執行的風險。在這種情況下，MASVS-R的安全控管機制可以被用來防止篡改和提高惡意程式開發者的進入門檻。
- 所有被設計需要儲存機敏資料在行動裝置和支持許多不同種類行動裝置與作業系統版本的行動應用程式，建議採用MASVS L2+R。在此情況下，安全控管機制可以被用於提高防禦縱深的手段，進而提高攻擊者提取機敏資訊的難度。

評估與認證

OWASP 對 MASVS 認證與信任標誌聲明

OWASP是一個中立無廠商偏好的非營利組織。OWASP本身並不會認證/授權任何廠商、認證機構或者軟體。

任何這樣的保證聲明、信任標誌或認證，都不會是OWASP官方正式審查、註冊或認證的！因此，任何組織都必須十分謹慎地看待，且不應輕易相信任何宣稱其擁有MASVS認證或信任標誌的第三方機構。

這段聲明並不是阻止任何組織去提供相關的認證服務，但任何組織不該宣稱擁有OWASP官方的認證。

行動應用程式認證指南

推薦使用行動應用資訊安全驗證標準(MASVS)來驗證行動應用程式的方法是採用“開書(open book)”的方式。也就是說測試人員會被給予存取關鍵資源(如：誰是應用程式的開發者和架構師、專案文件、原始程式碼，以及對端點<伺服器、應用程式>存取的合法授權帳號<至少每個不同權限各有一個帳號>)的權限。

請注意MASVS只包含了對應用程式用戶端(client-side)、應用程式與其遠端端點的網路通訊，以及一些對使用者驗證和會話(session)管理的通用基本準則。MASVS不包含對與應用程式相關連的遠端服務(如：Web服務)的個別準則，安全地只使用部分對使用者驗證和會話(session)管理的通用準則的方法。然而，MASVS V1要求遠端服務必須被包括在整體威脅模型，並且以適用的標準來驗證遠端服務，例如OWASP ASVS。

提供認證的機構必須要在所有的報告中，闡明以下幾點：認證的範圍(尤其如果有重要程式模塊不在認證範圍中)、認證結果總結(包含通過和失敗的測試項目)並清楚的解釋如何改進未通過的項目。業界的標準做法也同時要求保持詳細的工作記錄、螢幕截屏或影片、可以重複有效地執行弱點入侵的腳本和測試的電子紀錄(如：擷取下來的代理伺服器記錄檔和相關的筆記(如：待清理清單))。只是簡單的執行工具和報告發現的問題是不足夠的，因為這樣無法提供足夠的證據說明所有的認證相關的問題/準則都已經被完整地測試。為了防止爭議，認證機構必須提供充足的支持證據來證明所有需驗證的項目都已經確實地被測試過。

OWASP 移動安全檢測操作指南 (MSTG)

OWASP MSTG是測試行動應用程式安全的指導手冊。它描述了驗證被列在MASVS中相關準則的技術流程。MSTG提供了一個測試方案的列表，每一個測試方案都可以連結到一個MASVS的準則。雖然MASVS的準則都是概念性的和通用性的，MSTG提供了對不同行動作業系統的相對應深層次的建議以及測試流程。

自動化資訊安全檢測工具的使用

在可能的情況下，為了增加效率，程式碼掃描工具和黑箱測試工具的使用是被鼓勵的。然而，要完成MASVS的驗證是不可以只單獨依賴自動化工具的，因為每個行動應用程式都是不同的，而且了解應用程式的整體架構、商業邏輯和應用特定技術和框架帶來的缺點是驗證應用程式安全的必要條件。

其他用途

詳解安全架構開發指南

常見的行動應用資訊安全驗證標準(MASVS)使用方式是把MASVS當成是安全架構師的參考資料。目前兩個主要的安全架構框架：SABSA或TOGAF，都缺少了很多完成行動應用程式安全架構檢查所必需的資訊。MASVS透過提供這些缺失的必需資訊來幫助安全架構師在常見的行動應用程式議題上，選擇更好的安全控管機制。

取代現成的安全程式設計檢查表(Secure Coding Checklists)

許多組織都可以透過套用MASVS而受益，藉由選擇兩個安全等級之一或以MASVS為基準(fork)，發展出因應各個應用程式的風險等級需求的行業特殊準則。我們鼓勵這樣以MASVS為基準的再發展(fork)，只要可以維持其可追溯性。如此一來，當一個應用程式已經通過了準則4.1，也就意味著隨著標準演進，此應用程式通過了再發展版本的準則4.1。

基本安全性檢測與方式

好的行動應用程式安全檢測方法應該包含所有MASVS列舉的準則。OWASP 移動安全檢測操作指南(MSTG)描述了黑箱測試和白箱測試對每一個需檢測準則的測試方案。

自動化單元與和測試指南

MASVS被設計為擁有高度可測試性，除了架構上的需求以外，自動化單元、整合測試和驗收測試基於MASVS準則，可以被整合到程式持續開發生命周期(continuous development lifecycle)中。這不僅增強了開發人員的資安意識，也增進了應用程式的整體品質，更減少了在程式發布前階段，資訊安全測試的工作量(因為有提前做了整體自動化測試，可以減少後續發現重複或明顯問題的機會)。

安全性開發培訓課程

MASVS也可以被用來定義安全行動應用程式的特性。很多“安全開發”課程，其實只是道德駭客課程，再加上一點點程式開發的小提示。這樣的課程並不能很好的幫助開發人員。因此，安全開發課程可以使用MASVS，並且主要專注在MASVS中列舉的預先防護的資安控管機制，而不是其他MASVS的資安議題(例如：十大程式安全問題)。

V1: 架構、設計和威脅模型分析準則

控制目標

在一個完善的社會，整個開發階段都會考慮到安全性。然而實際上，安全性常常只是 SDLC 後期的考慮因素。除了技術控制之外，MASVS 需要制定流程，以確保在規劃行動應用程式的結構時明確考慮到安全性的問題，並確認所有元件的功能性和安全性是清楚的。由於大多數行動應用程式是作為遠程服務的客戶端，因此必須確保妥善的安全標準也運用在這些服務 - 只是獨立測試行動應用程式是不夠的。

“V1”這個類別列出了與應用程式的架構和設計有關的要求。因此，這是唯一沒有對應到 OWASP Mobile Testing Guide 裡的技術測資的類別。為了涵蓋威脅模型分析，安全SDLC，密鑰管理等主題，MASVS 的用戶應參考相對應的 OWASP項目 和/或 其他標準，例如下面連結所提到的。

安全驗證準則

以下列出了 MASVS-L1 和 MASVS-L2 的要求。

#	描述	L1	L2
1.1	所有應用程式組成元件都已歸類並且已知是必需的。	✓	✓
1.2	安全控制永遠不會僅僅在客戶端強制執行，而是在相對應的遠程端點上也必須強制執行。	✓	✓
1.3	行動應用程式的高級架構和所有連接的遠程服務已經被定義，並且該架構已解決了安全性的問題。	✓	✓
1.4	在行動應用程式環境中被認為敏感的資料已被明確歸類。	✓	✓
1.5	所有應用程式的組成都根據它們提供的業務功能 和/或 安全功能進行定義。	✓	
1.6	行動應用程式和相關遠程服務的威脅模型已經制定，以識別潛在的威脅和對策。	✓	
1.7	所有安全控制都有一個集中的管理。	✓	
1.8	如何管理加密金鑰（如果有）有明確方針，而且加密金鑰生命週期需強制實施。理想情況下，遵循 NIST SP 800-57 等鑰匙管理標準。	✓	
1.9	存在強制行動應用程式更新的機制。	✓	
1.10	安全性在軟體開發生命週期的所有階段中仔細納入考量。	✓	

參考

更多資訊請參閱：

- OWASP Mobile Top 10: M10 - Extraneous Functionality: https://www.owasp.org/index.php/Mobile_Top_10_2016-M10-Extraneous_Functionality
- OWASP Security Architecture cheat sheet:
https://www.owasp.org/index.php/Application_Security_Architecture_Cheat_Sheet
- OWASP Thread modelling: https://www.owasp.org/index.php/Application_Threat_Modeling
- OWASP Secure SDLC Cheat Sheet: https://www.owasp.org/index.php/Secure_SDLC_Cheat_Sheet
- Microsoft SDL: <https://www.microsoft.com/en-us/sdl/>
- NIST SP 800-57: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf

V2: 資料存儲和隱私要求

管理目標

保護機敏資料（如用戶憑據和個人訊息）是行動安全的重點。首先，如果操作系統機制諸如 IPC 使用不當，這將是敏感的資料被暴露無意中在同一設備或其他應用程式上。此外，資料可能會無意中從雲端存儲，備份，鍵盤緩存中洩漏。此外，與其他類型的設備相比，行動設備很容易丟失或被盜。因此，第三方可以更容易地訪問它們。在這種情況下，可以實施額外的保護使獲取機敏資料更加困難。

注意：由於 MASVS 是以應用程式開發為中心的，因此它不包括像 MDM 那樣的解決方案和策略。我們建議您通過在企業環境中使用策略來進一步增強數據安全性。

機敏資料的定義

MASVS 規範中的涉及敏感數據用戶憑證和在特定情況中被視為機敏的任何其他數據，例如：

- 可能被犯罪集團濫用的個人識別信息 (PII)：社會安全號碼，信用卡號碼，銀行帳號，醫療信息；
- 機敏資料導致財務費用發生損壞，合約訊息，保密協議，管理訊息等；
- 任何必須受被法律保護或出於原因兒受到保護的資料。

安全驗證要求

通過遵循簡單的規則與定義可以防止絕大多數資料洩露問題，本章中列出的大多管理目標對於所有驗證級別都是必需的。

#	描述	L1	L2
2.1	系統的憑證存儲功能適用於存儲機敏資料，如個人識別訊息，用戶憑證續席，加密密鑰等。	✓	✓
2.2	機敏資料不存儲在應用程式的服務器或系統以外的憑證存儲功能。	✓	✓
2.3	機敏資料未寫入應用程式日誌。	✓	✓
2.4	除非對架構有必要，否則不與第三方共享機敏資料。	✓	✓
2.5	對於明文輸入機敏資料未加密處理，禁用鍵盤緩存。	✓	✓
2.6	機敏資料不通過 IPC 機制公開。	✓	✓
2.7	密碼和定義參數等機敏資料不會通過用戶界面公開。	✓	✓
2.8	備份的檔案中不包含在移動操作系統生成任何的機敏資料。		✓
2.9	後台操作系統中可刪除機敏資料。		✓
2.10	應用程式不會在記憶體中保存超過必要的機敏資料，並且在使用後會明確清除記憶體內容。		✓
2.11	應用程式採用了最低訪問安全策略，會要求用戶設置密碼等。		✓
2.12	應用程序已主動告知如何處理用戶的個人識別信息，並且主動告知用戶的使用應用程式時要遵循的要點。		✓

參考

OWASP 行動安全檢測指南列出相關要求，並且相關章節中有詳細說明。

- For Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05d-Testing-Data-Storage.md>
- For iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06d-Testing-Data-Storage.md>

更多相關信息，另請參閱：

- OWASP Mobile Top 10: M2 - Insecure Data Storage: https://www.owasp.org/index.php/Mobile_Top_10_2016-M2-Insecure_Data_Storage
- CWE: <https://cwe.mitre.org/data/definitions/922.html>

V3: 加密要求

管理目標

加密是保護行動設備上存儲的資料的重要因素。特別是如果不遵守標準規則開發，事情可能出現嚴重錯誤。本章中的管理目的是檢測經過驗證的應用程序是否根據行業最佳實踐使用加密，例如：

- 使用經過驗證的加密方式；
- 正確選擇和設置加密明文；
- 在需要隨機地方使用皆有使用隨機數產生器。

安全檢測要求

#	描述	L1	L2
3.1	應用程式不可使用 hardcoded 作為加密方法。	✓	✓
3.2	應用程式中使用驗證過的加密演算法的加密明文。	✓	✓
3.3	基於業界最佳實踐的方式，使應用程式適用於特定的加密明文。	✓	✓
3.4	應用程式不使用基於安全目的而被廣泛棄用的加密協議和算法。	✓	✓
3.5	加密金鑰不重複使用。	✓	✓
3.6	使用強密碼長度足夠，排列隨機算法產生加密數值。	✓	✓

參考

OWASP 行動安全檢測指南列出相關要求，並且相關章節中有詳細說明。

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05e-Testing-Cryptography.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06e-Testing-Cryptography.md>

更多相關信息，另請參閱：

- OWASP Mobile Top 10: M5 - Insufficient Cryptography: https://www.owasp.org/index.php/Mobile_Top_10_2016-M5-Insufficient_Cryptography
- CWE: <https://cwe.mitre.org/data/definitions/310.html>

V4: 身份驗證和對談管理準則

控制目標

在多數情況下，登錄遠程服務的用戶是整個行動應用程式架構整體的一部分。即使大多數邏輯發生在端點，MASVS定義了有關如何管理用戶帳戶和對談的一些基本要求。

安全驗證準則

#	描述	L1	L2
4.1	如果應用程式向用戶提供對遠程服務的訪問權限，則用戶名/密碼驗證等一些形式的身份驗證會在遠程端點執行。	✓	✓
4.2	如果使用有狀態的對談管理，則遠程端點使用隨機生成的交談識別碼來驗證客戶端請求，而不發送用戶憑據。	✓	✓
4.3	如果使用無狀態token-based驗證方式，則服務器會提供使用安全算法簽名的token。	✓	✓
4.4	當用戶登出時，遠程端點會終止現有對談。	✓	✓
4.5	存在密碼方針，並在遠程端點強制實行。	✓	✓
4.6	遠程端點實踐了一個可以防止多次提交憑據的機制。	✓	✓
4.7	在事先定義的不活動時段和用來訪問的token過期後，對談在遠程端點將無效。	✓	✓
4.8	生物辨識認證（如果有的話）不受事件限制（即是像簡單地返回“真”或“假”的API）。相反，它是基於解鎖鑰匙串/密鑰庫。		✓
4.9	遠程端點存在第二個驗證要素，並且始終強制執行2FA準則。		✓
4.10	敏感的交易需要進一步認證。		✓
4.11	應用程式通過其帳戶通知用戶所有登錄活動。用戶可以查看用於訪問帳戶的裝置列表，以及阻擋特定裝置。		✓

參考

OWASP Mobile Security Testing Guide 提供了有關驗證本章節中列出的準則的詳細使用說明。

- In general - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04e-Testing-Authentication-and-Session-Management.md>
- For Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05f-Testing-Local-Authentication.md>
- For iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06f-Testing-Local-Authentication.md>

更多資訊請參閱：

- OWASP Mobile Top 10: M4 - Insecure Authentication: https://www.owasp.org/index.php/Mobile_Top_10_2016-M4-Insecure_Authentication
- OWASP Mobile Top 10: M6 - Insecure Authorization: https://www.owasp.org/index.php/Mobile_Top_10_2016-M6-Insecure_Authorization
- CWE: <https://cwe.mitre.org/data/definitions/287.html>

V5: 網路通訊規範

管理目標

本節中列出的管理目標是確保行動應用程式和遠程服務之間交換的訊息的機密性和完整性。確保行動應用程式必須使用 TLS 協議為網絡傳輸作為安全的加密通道。L2 列出了其他加強防禦措施，例如 SSL 固定。

安全驗證要求

#	描述	L1	L2
5.1	APP 網路傳輸資料均使用 TLS 協議作為加密協定.	✓	✓
5.2	TLS 協定使用，必須符合目前業界規範，如已通報不安全的協定標準，須立即修正。	✓	✓
5.3	在與遠端建立資料傳輸通道時，須先驗證 X.509 證書，且只能接受有簽名受信任的 CA 憑證。	✓	✓
5.4	應用程式中公鑰和數位憑證來確保系統資訊安全，並不能隨意與不明遠端建立連線，提供受信任 CA 憑證.		✓
5.5	應用程式的相關重要操作，不可只依賴單一方式驗證（如電子郵件或者 SMS）		✓
5.6	應用程式需隨時更新連接方式以及資料庫安全性		✓

參考

OWASP Mobile Security Testing Guide 提供了有關驗證本章節中列出的準則的詳細使用說明。

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05g-Testing-Network-Communication.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06g-Testing-Network-Communication.md>

更多相關信息，另請參閱：

- OWASP Mobile Top 10: M3 - Insecure Communication: https://www.owasp.org/index.php/Mobile_Top_10_2016-M3-Insecure_Communication
- CWE: <https://cwe.mitre.org/data/definitions/319.html>
- CWE: <https://cwe.mitre.org/data/definitions/295.html>

V6: 平台互動規範

控制目標

這個群組中的控制確保應用程式以安全的方式使用平台API和標準元件。此外，控制還涵蓋應用程式（IPC）之間的通訊。

安全驗證要求

#	描述	L1	L2
6.1	應用程式僅請求必要的最小權限集。	✓	✓
6.2	來自外部來源和用戶的所有輸入都經過驗證，並在必要時進行消毒。這包括通過UI，IPC機制像是意圖、自定義URL和網路來源接收的資料。	✓	✓
6.3	除非適當保護這些機制，否則應用程式不會通過自定義URL結構輸出機密功能。	✓	✓
6.4	除非適當保護這些機制，否則應用程序不會通過IPC設備輸出機密功能。	✓	✓
6.5	除非明確地要求，否則在WebView中禁止使用JavaScript。	✓	✓
6.6	WebViews被設置為僅允許所需最少量協議處理程序（理想情況下，僅支援https）。潛在危險的處理程序，例如file、tel和app-id則不能使用。	✓	✓
6.7	如果應用程式的native方法被暴露給WebView，請驗證WebView是否僅提供應用程式套件中包含的JavaScript。	✓	✓
6.8	如果有任何物件反序列化，要使用安全序列化API執行。	✓	✓

參考

OWASP Mobile Security Testing Guide 提供了有關驗證本章節中列出的準則的詳細使用說明。

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05h-Testing-Platform-Interaction.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06h-Testing-Platform-Interaction.md>

更多資訊請參閱：

- OWASP Mobile Top 10: M1 - Improper Platform Usage: https://www.owasp.org/index.php/Mobile_Top_10_2016-M1-Improper_Platform_Usage
- CWE: <https://cwe.mitre.org/data/definitions/20.html>
- CWE: <https://cwe.mitre.org/data/definitions/749.html>

V7: 程式碼品質與建立設定要求

控管目的

此控管目的是確保實作編碼時將遵照應用程式開發基本安全性，且基本(免費)安全性功能是藉由啟用編譯器所提供之。

安全性驗證要求

#	說明	L1	L2
7.1	應用程式經由有效的憑證所提供之簽章，其私鑰是受到適當保護的	✓	✓
7.2	應用程式已在發佈模式建置，其設定適用於發佈版本(不可除錯的)	✓	✓
7.3	除錯標記已從原生二元碼中移除	✓	✓
7.4	除錯程式碼已移除，應用程式將不紀錄詳細錯誤或偵錯訊息	✓	✓
7.5	所有行動應用程式使用的第三方套件，如函式庫及框架，皆經識別及已知漏洞查驗	✓	✓
7.6	應用程式進行例外處理	✓	✓
7.7	預設無法存取安全性控管的錯誤處理邏輯	✓	✓
7.8	在非託管程式碼，記憶體將被安全的被分配、釋出及使用	✓	✓
7.9	由Toolchain所提供的免費安全性功能將啟用，如簡化位元組碼(byte-code)、堆疊(Stack)保護、PIE支援、自動參考計數(Reference Counting)	✓	✓

參考資料

OWASP 行動應用安全性測試指南針對此章節所列出的認證要，提供詳細的說明

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05i-Testing-Code-Quality-and-Build-Settings.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06i-Testing-Code-Quality-and-Build-Settings.md>

更多資訊，請參考：

- OWASP Mobile Top 10: M7 - Poor Code Quality: https://www.owasp.org/index.php/Mobile_Top_10_2016-M7-Poor_Code_Quality
- CWE: <https://cwe.mitre.org/data/definitions/119.html>
- CWE: <https://cwe.mitre.org/data/definitions/89.html>
- CWE: <https://cwe.mitre.org/data/definitions/388.html>
- CWE: <https://cwe.mitre.org/data/definitions/489.html>

V8: 彈性準則

控制目標

本節涵蓋對於存取機敏資料的功能的行動應用程序建議的縱身防禦措施。缺少任何這些控件不會導致漏洞 - 相反的，它們打算增加逆向工程及特定的客戶端攻擊對行動應用程式的彈性

基於對未經授權篡改應用程式和/或代碼逆向工程所導致的風險的評估，應根據需要應用本節中的控件。我們建議諮詢OWASP文件"逆向工程技術風險和未經授權的代碼修改逆向工程和代碼修改預防"(參考下面)列出的業務風險以及相關的技術威脅。

對下面的列表中的任何控件是有效的，應用程式必須滿足至少所有的MASVS-L1(換句話說，實體安全控件一定是到位)，以及V8內所有低編號的要求，例如，下面的混淆控制列表"妨礙理解"一定是跟"應用程式隔離"結合，"妨礙動態分析和竄改"，和"設備綁定"

注意：軟體保護絕不是使用在一個安全控制元件替換，MASVR-R中列出的控件旨在增加特定威脅，對滿足MASVS安全要求的應用程式，提供額外的保護控制

以下注意事項採用：

1. 威脅模型必須明確概述客戶端抵擋的威脅。此外，必須具體說明該方案旨在提供的保護等級。例如，一個明確的目標可以強制使搜尋特定惡意軟體的作者對應用程式投入大量的手動逆向工程作業來實作。
2. 威脅模型必須是敏感的。例如，如果攻擊者可以簡單地將整個白箱code lift，那麼將加密金鑰隱藏在白箱中就沒有意義。
3. 保護的有效性應始終由具有測試所使用的特定類型防篡改和混淆經驗的專家進行驗證（另請參閱 Mobile Security Testing Guide 中的“逆向工程”和“評估軟體保護”章節）。

阻礙動態分析和篡改

#	描述	R
8.1	應用程式通過提醒用戶或終止應用程式來檢測並回應超級用戶權限或越獄設備的存在。	✓
8.2	應用程式防止和回應偵錯 和/或 檢測的偵錯器附加上去。必須涵蓋所有可用的偵錯協議。	✓
8.3	應用程式檢測和回應自己的沙箱中可執行的檔案和重要資料遭竄改。	✓
8.4	應用程式檢測和回應設備上廣泛使用的逆向工程工具和架構的存在。	✓
8.5	應用程式檢測並回應在模擬器中運行。	✓
8.6	應用程式檢測並回應篡改其自己的記憶體空間中的程式碼和資料。	✓
8.7	應用程式在各個防禦類別（8.1到8.6）中實施多種機制。請注意，彈性隨著所用機制的數量，原創性的多樣性而變化。	✓
8.8	檢測機制觸發不同類型的回應，包括延遲和暗中回應。	✓
8.9	混淆應用於程序化防禦，這反過來通過動態分析阻止去除混淆。	✓

裝置綁定

#	描述	R
8.10	應用程式使用從裝置特有的多個特性衍生的設備指紋實作“裝置綁定”功能。	✓

防止洩漏

#	描述	R
8.11	屬於應用程式的所有可執行的檔案和函式庫都在文件級別上加密 和/或 可執行的檔案內的重要程式碼和資料片段被加密或打包。瑣碎的靜態分析不會顯示重要的程式碼或資料。	✓
8.12	考慮到當前發布的研究，如果混淆的目標是保護機密的計算，則使用混淆方案，該方案既適用於特定任務又良好的適用於手動和自動去混淆辦法。必須通過手動測試來驗證混淆方案的有效性。請注意，比起混淆，盡可能使用基於硬體的隔離功能。	✓

參考

OWASP Mobile Security Testing Guide 提供了有關驗證本章節中列出的準則的詳細使用說明。

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06j-Testing-Resiliency-Against-Reverse-Engineering.md>

更多資訊請參閱：

- OWASP Mobile Top 10: M8 - Code Tampering: https://www.owasp.org/index.php/Mobile_Top_10_2016-M8-Code_Tampering
- OWASP Mobile Top 10: M9 - Reverse Engineering: https://www.owasp.org/index.php/Mobile_Top_10_2016-M9-Reverse_Engineering
- WASP Reverse Engineering Threats - https://www.owasp.org/index.php/Technical_Risks_of_Reverse_Engineering_and_Unauthorized_Code_Modification

on

- OWASP Reverse Engineering and Code Modification Prevention -
https://www.owasp.org/index.php/OWASP_Reverse_Engineering_and_Code_Modification_Prevention_Project

附錄 A：詞彙表

雙因素身份驗證(Two-factor Authentication, 2FA)

在帳號登入時增加第二層的身份驗證。

位址空間組態隨機載入(Address Space Layout Randomization, ASLR)

一種能讓利用記憶體損壞漏洞的攻擊更加困難的技術。

應用程式安全(Application Security)

應用層安全性是針對 OSI 模型應用層元件的分析，而非針對作業系統或是網路連線。

應用程式安全驗證(Application Security Verification)

對應用程式利用 OWASP MASVS 進行技術評鑑。

應用程式安全驗證報告(Application Security Verification Report)

一份針對特定應用程式，由檢驗者製作，紀錄所有結果與相關分析的報告。

身份驗證(Authentication)

對應用程式使用者鑒別身分的驗證。

自動化驗證(Automated Verification)

採用自動化工具(包括動態分析工具、靜態分析工具或兩者皆採用)以利用弱點特徵來發掘問題。

黑箱測試(Black box testing)

一種軟體測試方法，能在未知內部結構與運作情形的狀況下檢驗應用程式功能。

元件(Component)

一個獨立的程式碼單元，具有與其他元件通訊的相關磁碟和網路介面。

跨站腳本(Cross-Site Scripting, XSS)

網路應用程式中常見的安全漏洞，允許將客戶端的腳本注入內容。

密碼模組(Cryptographic module)

實作加密演算法和/或生成加密金鑰的硬體，軟體 和/或 韌體。

共通弱點條目(CWE)

CWE 是社群開發的常見軟體安全漏洞列表。它是一種通用語言，是軟體安全工具的衡量標準，也是弱點識別，緩解和預防工作的基準。

動態應用程式安全測試(DAST)

動態應用程式安全性測試（DAST）技術旨在檢測標示處於運行狀態的應用程式中的安全漏洞。

設計驗證(Design Verification)

應用程式安全架構的技術評估。

動態驗證(Dynamic Verification)

運用自動化工具來在應用程式執行期間使用漏洞簽名尋找問題。

全局唯一識別元(Globally Unique Identifier, GUID)

用作軟體識別的唯一標識。

超文本傳輸協定(Hyper Text Transfer Protocol, HTTP)

用於分佈式、協作式和超媒體資訊系統的應用層協定。它是全球資訊網的資料通訊的基礎。

硬編碼密鑰(Hardcoded keys)

儲存在裝置本身中的加密金鑰。

行程間通訊(IPC)

行程間通訊（Inter-Process Communication），在IPC，處理程序彼此通訊並與內核通訊以協調其活動

輸入驗證(Input Validation)

不受信任用戶輸入的標準化和驗證。

JAVA位元組碼(JAVA Bytecode)

JAVA位元組碼是Java虛擬機器執行的一種指令格式。每個位元組碼由一個或有些時候兩個位元組組成，用來表示指令(opcode)，以及用於傳遞參數的零個或多個位元組。

惡意程式碼(Malicious Code)

程式碼在應用程式所有者不知情的情況下添加入應用程式，它繞過了應用程式預期的安全方針。與病毒或蠕蟲等惡意軟體不同！

惡意軟體(Malware)

在應用程式使用者及管理員不知情的情況下，在運行期間添加入應用程式的可執行程式碼。

開放網路應用程式安全計畫(Open Web Application Security Project, OWASP)

開放網路應用程式安全計畫 (OWASP) 是一個全球性的免費開放社群，致力於提高應用程式軟體的安全性。我們的任務是使應用程式安全性“可見”，以便人們和機構能夠就應用程式安全風險做出明智的決策。參閱：

<http://www.owasp.org/>

個人可識別資訊(Personally Identifiable Information, PII)

是可以單獨使用或與其他資訊一起使用的資訊，用於辨識、聯絡、定位單一人士，或在上下文中識別個人。

地址無關可執行文件(PIE)

Position-independent executable (PIE) 是指可在主存儲器中任意位置正確地運行，而不受其絕對地址影響的一種機器碼。

公開金鑰基礎建設(PKI)

PKI是一種將公開金鑰與實體的相應身份鏈結的協定。鏈結是通過數位憑證認證機構 (CA) 註冊和發布憑證的過程建立。

靜態應用程式安全測試(SAST)

Static application security testing (SAST) 是一組技術，旨在分析應用程式原始碼，位元組碼和二進制程式碼，用於程式碼編寫和指示安全漏洞的設計條件。SAST解決方案在非運行狀態下“由內而外”分析應用程式。

系統發展生命週期(SDL

軟體開發生命週期。

安全架構(Security Architecture)

應用程式設計的抽象概念，用於辨識和描述安全控制的使用位置和方式，還可以辨識和描述用戶和應用程式資料的位置和敏感性。

安全配置(Security Configuration)

影響安全控制使用方式的應用程式運行時配置。

安全控制措施(Security Control)

功能或元件執行安全檢查（例如，訪問控制檢查）或被呼叫時所導致的安全結果（例如，產生審查記錄）。

SQL資料隱碼攻擊(SQL Injection, SQLi)

用於攻擊資料驅動的應用程式的程式碼注入技術，將惡意SQL語句插入 entry point。

單一簽入身分驗證(SSO Authentication)

Single Sign On(SSO)發生在當用戶登錄到一個客戶端然後自動登錄到其他客戶端時，無論用戶使用何種平台，技術或域。例如，當你在google登入時，您自動登入了YouTube，文檔和郵件服務。

威脅模型分析(Threat Modeling)

一種由開發日益完善的安全架構，以識別威脅媒介，安全區域，安全控制以及重要的技術和業務資產所組成的技術。

傳輸層安全性協定(Transport Layer Security, TLS)

提供覆蓋網際網路的通訊安全性的加密協定。

統一資源識別符/統一資源定位符(URI/URL fragments)

統一資源識別符是用於標識名稱或網路資源的字元串。統一資源定位符常常用作資源的引用。

用戶驗收測試(User acceptance testing, UAT)

傳統上，測試環境的行為類似於在上線之前執行所有軟體測試的生產環境。

驗證員(Verifier)

正在根據OWASP ASVS要求審核應用程式的人員或團隊。

白名單(Whitelist)

允許資料或操作的列表，例如允許執行輸入驗證的字元列表。

X.509憑證(X.509 Certificate)

X.509憑證是一種數位憑證，它使用廣泛接受的國際X.509公開金鑰基礎建設（PKI）標準來驗證公開金鑰是否屬於憑證中包含的用戶，電腦或服務。

附錄B：參考資料

以下專案內容將幫助使用者/適用者更了解這個標準

- OWASP 行動應用安全性專案 - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- OWASP 行動應用安全測試指南 https://www.owasp.org/index.php/OWASP_Mobile_Security_Testing_Guide
- OWASP 行動應用 Top 10 風險 https://www.owasp.org/index.php/Projects/OWASPMobile_Security_Project-Top_Ten_Mobile_Risks
- OWASP 逆向工程及程式碼編修防護
https://www.owasp.org/index.php/OWASP_Reverse_Engineering_and_Code_Modification_Prevention_Project
- 相同的，以下網站將幫助使用者/適用者更了解這個標準
 - MITRE 常見弱點列舉 - <http://cwe.mitre.org/>
 - PCI 安全性標準委員會 - [<https://www.pcisecuritystandards.org>] (<https://www.pcisecuritystandards.org>)
 - PCI 資料安全性標準(DSS)要求和安全評估程序 第三版-
https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf