Table of Contents

# 



# 應用安全驗證標準 4.0.3

## 終版

2021年10月

# 卷首語

## 關於本標準

應用安全驗證標準，是一份要求和測試應用安全的清單，可供架構師、開發人員、測試人員、安全專家、工具供應商和消費者參考，用於定義、構建、測試和驗證安全的應用。

## 版權和許可

Version 4.0.3, 2021年10月



Copyright © 2008-2021 The OWASP Foundation. 本文件根據 [Creative Commons Attribution ShareAlike 3.0 license](https://creativecommons.org/licenses/by-sa/3.0/) 釋出。 對於任何重用或分發，你必須向他人明確說明本作品的許可條款。

## 專案負責人

|  |  |  |
| --- | --- | --- |
| Andrew van der Stock | Daniel Cuthbert | Jim Manico |
| Josh C Grossman | Elar Lang |  |

## 主要貢獻者

|  |  |  |
| --- | --- | --- |
| Abhay Bhargav | Benedikt Bauer | Osama Elnaggar |
| Ralph Andalis | Ron Perris | Sjoerd Langkemper |
| Tonimir Kisasondi |  |  |

## 其他貢獻者和審查者

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Aaron Guzman | Alina Vasiljeva | Andreas Kurtz | Anthony Weems | Barbara Schachner |
| Christian Heinrich | Christopher Loessl | Clément Notin | Dan Cornell | Daniël Geerts |
| David Clarke | David Johansson | David Quisenberry | Elie Saad | Erlend Oftedal |
| Fatih Ersinadim | Filip van Laenen | Geoff Baskwill | Glenn ten Cate | Grant Ongers |
| hello7s | Isaac Lewis | Jacob Salassi | James Sulinski | Jason Axley |
| Jason Morrow | Javier Dominguez | Jet Anderson | jeurgen | Jim Newman |
| Jonathan Schnittger | Joseph Kerby | Kelby Ludwig | Lars Haulin | Lewis Ardern |
| Liam Smit | lyz-code | Marc Aubry | Marco Schnüriger | Mark Burnett |
| Philippe De Ryck | Ravi Balla | Rick Mitchell | Riotaro Okada | Robin Wood |
| Rogan Dawes | Ryan Goltry | Sajjad Pourali | Serg Belkommen | Siim Puustusmaa |
| Ståle Pettersen | Stuart Gunter | Tal Argoni | Tim Hemel | Tomasz Wrobel |
| Vincent De Schutter | Mike Jang |  |  |  |

如果上面的 4.0.3 致謝列表中缺少某些內容，請在GitHub上記錄一個工單，以便在未來的更新中得到確認。

從2008年的ASVS 1.0，到2016年的3.0，這份應用安全驗證標準，一直都站在前人的肩膀上。 直到今天，ASVS中的大部分結構和驗證項，都是最初由Mike Boberski、Jeff Williams和Dave Wichers編寫的，但還有許多貢獻者。 感謝那些以前參與過的人。若需檢視所有早期版本貢獻者的名單，請查閱每個以前的版本。

# 序言

歡迎來到應用安全驗證標準（Application Security Verification Standard，以下簡稱ASVS）4.0版。 ASVS是一項社群驅動的工作，旨在建立一個安全要求和控制的框架，在設計、開發和測試現代網路應用程式和網路服務時，定義所需要的功能和非功能性的安全控制措施。

4.0.3版是對4.0版的第三個小補丁，旨在修正拼寫錯誤，使需求更清晰，而不做實質性的改變（如加強或增加需求）。 然而，在我們認為合適的地方，一些要求可能被稍微削弱，一些完全多餘的要求被刪除（但沒有重新編號）。

ASVS v4.0是過去十年來社群努力和行業反饋的結晶。 我們試圖讓 ASVS 更容易地適應安全軟體開發生命週期中的各種用例場景。

我們知道，對於任何 Web 應用程式標準（包括 ASVS）的內容，很可能永遠不會達成 100% 的一致。 在某種程度上，風險分析總是主觀的，這對試圖以一刀切的標準來概括造成了一種挑戰。 然而，我們希望這個最新的版本，是朝著正確方向邁出的一步，並增強了這個關鍵行業標準中的概念。

## 4.0版的新內容

這個版本中最重要的變化，是採用了 NIST 800-63-3 數字身份指南，引入了現代的、基於證據的高階認證控制。 雖然我們預計在與高階認證標準接軌方面會有一些阻力（這個廣受好評的應用安全標準是基於證據的），但我們認為標準保持一致是非常必要的。

資訊保安標準應儘量減少獨特要求的數量，以便遵守的組織不必決定衝突或不相容的控制措施。 在認證和會話管理方面，《OWASP Top 10 2017》和現在的《OWASP 應用安全驗證標準》，已與NIST 800-63保持一致。我們鼓勵其他標準制定機構與我們、NIST和其他機構合作，制定一套普遍接受的應用安全控制措施，最大限度地提高安全性和減少合規成本。

ASVS 4.0從頭到尾都進行了重新編號。新的編號方案使我們能夠彌補長期消失的章節的空白，並允許我們分割較長的章節，以儘量減少開發人員或團隊必須遵守的控制項。例如，如果應用程式不使用JWT，那麼會話管理中有關 JWT 的整個部分都不適用。

4.0版本中的新內容，是對 常見弱點列舉（Common Weakness Enumeration，以下簡稱CWE）的全面對映，這是我們在過去十年中最普遍的功能要求之一。 CWE 對映，允許供應商和使用漏洞管理軟體的人，將其他工具和先前 ASVS 版本的內容，同 4.0 及之後的版本進行匹配。 為了給 CWE 條目騰出空間，我們不得不取消“起始時間”列，由於我們完全重新編號，因此這一列意義不大。 並非ASVS中的每個專案都有相關的CWE，由於CWE有大量的重複，我們嘗試使用最常用而非最接近的匹配項。 驗證控制，並不總是可以對應到等效的CWE。 我們樂意與 CWE 社群和資訊保安領域就縮小這一差距進行更廣泛的討論。

我們致力於全面滿足並超越《OWASP Top 10 2017》和《OWASP Proactive Controls 2018》的要求。由於《OWASP Top 10 2017》是避免疏忽的最低要求，我們特意將Top 10中的所有要求（除“日誌記錄”外）列為1級控制（Level 1），讓OWASP Top 10的採用者，更容易將實際的安全標準落地。

我們著手確保ASVS 4.0 1級（Level 1），是PCI DSS 3.2.1第6.5節的超集，用於應用程式設計、編碼、測試、程式碼安全審計和滲透測試。這需要覆蓋 V5 中的緩衝區溢位和不安全記憶體操作，以及 V14 中與不安全記憶體相關的編譯標誌，以及現有行業領先的應用程式和 Web 服務驗證要求。

W我們已經完成了ASVS的轉變，從單一的伺服器端控制，到為所有現代應用和API提供安全控制。 在server-less API、移動、雲、容器、CI/CD和DevSecOps、聯邦等等的時代，我們不能繼續忽視現代應用架構。 現代應用的設計，與2009年最初的ASVS釋出時的設計大不相同。ASVS必須始終著眼於未來，這樣我們才能為我們的主要受眾——開發者提供合理的建議。 我們已經刪除了那些“假設應用程式在單個組織擁有的系統上執行”的要求。

由於ASVS 4.0的規模，以及我們希望成為所有其他應用安全驗證標準的基線，我們已經淘汰了“移動”章節，轉而支援移動應用程式安全驗證標準（MASVS）。 物聯網附錄將出現在未來OWASP IoT專案的物聯網ASVS中。我們在附錄C中包含了物聯網ASVS的早期預覽。 感謝OWASP Mobile團隊和OWASP IoT專案團隊對ASVS的支援，期待未來與他們合作提供補充標準。

最後，我們刪除和淘汰了影響較小的控制項。 隨著時間的推移，ASVS開始成為一套全面的控制措施，但並非所有控制項在開發安全軟體方面都是平等的。 消除低影響專案的努力可能會更進一步。 在 ASVS 的未來版本中，通用弱點評分系統（Common Weakness Scoring System，CWSS），將有助於進一步優先考慮那些真正重要的控制和應該停用的控制。

從4.0版本開始，ASVS將專注於成為領先的Web應用程式和服務標準，涵蓋傳統和現代應用架構，以及敏捷的安全實踐和DevSecOps文化。

# 使用ASVS

ASVS有兩個主要目標：

* 幫助組織開發和維護安全的應用程式。
* 允許安全服務廠商、安全工具供應商和消費者調整他們的要求和產品。

## 應用安全驗證級別

應用程式安全驗證標準（ASVS）定義了三個安全驗證級別，每個級別的深度都在增加。

* ASVS Level 1 適用於低保證級別，可通過滲透測試驗證。
* ASVS Level 2 適用於包含敏感資料的應用程式（需要保護），是大多數應用程式的推薦級別。
* ASVS Level 3 適用於最關鍵的應用程式：執行高價值交易、包含敏感醫療資料的應用程式，或任何需要最高級別信任的應用程式。

每個ASVS級別都包含一個安全要求的列表。其中的每一項，都可以對應到開發人員必須在軟體中建立的特定安全特性和功能。



圖1 - OWASP應用安全驗證標準級別（4.0版）

Level 1 是唯一可以由人類進行滲透測試的級別。 所有其他的等級都需要接觸文件、原始碼、配置以及開發人員。 然而，即使L1允許通過“黑盒”（無文件和無原始碼）測試來驗證，但它並不是有效的保證活動，應儘量避免單獨採用。 惡意攻擊者有大量的時間，但大多數滲透測試在幾周內就會結束。 防禦者需要在合理的時間內，建立安全控制，發現、解決和保護所有的弱點，檢測和應對惡意攻擊者。惡意攻擊者的時間基本上是無限的，只需要一處漏洞，或者一個缺少檢測的弱點就可以成功。 黑盒測試，通常在開發結束時快速執行，或者根本不執行，完全無法應對這種不對稱性。

在過去的 30 多年裡，黑盒測試一次又一次被證明會遺漏關鍵的安全問題，直接導致了越來越多的大規模漏洞。 我們強烈鼓勵使用廣泛的安全保證和驗證，包括用 Level 1 的原始碼主導的（混合）滲透測試來取代滲透測試，在整個開發過程中可以充分接觸到開發人員和文件。 金融監管機構不會容忍“無法接觸賬本、交易樣本或執行控制的人員”的外部財務審計。 行業和政府必須要求軟體工程領域具有相同的透明度標準。

我們強烈鼓勵在開發過程中使用安全工具。 構建管道可以持續使用 DAST 和 SAST 工具，來查詢容易發現的、不應該出現的安全問題。

在沒有人工協助的情況下，ASVS中超過半數的驗證項無法被自動化工具和線上掃描完成。 如果需要對每次構建進行全面的自動化測試，則使用自定義單元和整合測試的組合，以及由構建發起的線上掃描。 業務邏輯缺陷和訪問控制測試只能使用人工協助。 這些內容應該變成單元和整合測試。

## 如何使用這個標準

使用應用程式安全驗證標準（ASVS）的最佳方法之一，是將其作為一個藍圖，建立一個針對您的應用程式、平臺或組織的安全編碼檢查表。 建議您根據不同的專案場景，針對其中最重要的安全要求，在定製的ASVS中增加關注。

### Level 1 - 第一步，自動化，或全景圖

如果一個應用程式能夠充分防禦 OWASP Top10 或其他類似檢查表中提到的那些容易發現的安全漏洞，那麼它就達到了ASVS Level 1。

Level 1，是所有應用程式應爭取的最低限度。 作為多個階段工作的第一步，或是當應用程式不儲存或處理敏感資料，因此不需要L2或L3的更嚴格的控制時，它也是有用的。 Level 1控制可以通過工具自動檢查，也可以在不訪問原始碼的情況下簡單地手動檢查。我們認為Level 1是所有應用程式的最低要求。

對應用程式的威脅，往往來自那些使用簡單、省力的技術來識別“簡單漏洞”的攻擊者。 這與堅定的攻擊者形成對比，後者將花費集中的精力專門針對應用程式。 如果您的應用程式處理的資料具有很高的價值，您很少會止步於Level 1。

### Level 2 - 大多數應用程式

如果一個應用程式能夠充分抵禦當今與軟體相關的大多數風險，那麼它就達到了ASVS Level 2（級別或標準）。

Level 2 確保安全控制在應用程式中得到有效的落實。 Level 2 通常適用於處理重要B2B事務（B2B，Business-to-Business）的應用程式， 包括處理醫療保健資訊、實施關鍵業務、實現敏感功能、或處理敏感資產的應用程式，或完整性是保護其業務的關鍵方面的行業，例如打擊遊戲外掛和作弊的遊戲行業。

對 Level 2 應用程式的威脅，通常是熟練的和有動機的攻擊者，他們專注於特定的目標，使用工具和技術有效地發現和利用應用程式中的風險。

### Level 3 - 高價值、高保證或高安全性

ASVS Level 3，是 ASVS 內的最高級別驗證。 這個級別通常保留給需要大量安全驗證的應用，例如軍事、健康、安全和關鍵基礎設施等領域的應用。

對於執行關鍵功能的應用程式，組織可能需要 ASVS Level 3，在這種情況下，故障可能會嚴重影響組織的運作，甚至影響其生存。下面提供了有關 ASVS 3Level 3應用的示例指導。 如果一個應用程式能夠充分防禦高階應用程式的安全漏洞，並顯示出良好的安全設計原則，那麼它就達到了ASVS Level 3（或高階）。

與其他級別相比，ASVS Level 3 的應用程式需要對架構、編碼和測試進行更深入的分析。 安全的應用程式以有意義的方式模組化的（以促進彈性，可擴充套件性，以及最重要的安全層），每個由網路連線和/或物理例項分開的模組，負責自己縱深防禦的安全職責，這需要適當的記錄。 職責包括確保機密性（例如加密）、完整性（例如事務、輸入驗證）、可用性（例如優雅地處理負載）、身份驗證（包括系統之間）、授權和審計（日誌記錄）的控制。

## 在實踐中應用ASVS

不同的威脅有不同的動機。某些行業有獨特的資訊和技術資產，以及特定領域的監管要求。

我們強烈建議，各組織根據其業務性質，深入研究其獨特的風險特徵，並根據該風險和業務要求，確定適當的ASVS級別。

## 如何引用ASVS要求

每個需求都有一個識別符號，格式為 <章>.<節>.<要求> ，每個元素都是一個數字，例如：1.11.3 。

* <chapter> 值對應於需求出現的章，例如：所有 1.#.# 的需求都來自Architecture這一章。
* <section> 值對應於該章中需求出現的小節，例如：所有 1.11.# 需求都在 Architecture 章的 Business Logic Architecture 部分。
* <requirement>值對應該章的具體要求，例如：本標準的4.0.3版本中的 1.11.3 是：

驗證所有高價值業務邏輯流（包括身份驗證、會話管理和訪問控制）都是執行緒安全的，並且可以防止“檢查時間和使用時間不一致”導致的條件競爭問題。

識別符號可能會在標準版本之間發生變化，因此在其他文件、報告或工具最好使用以下格式： v<version>-<chapter>.<section>.<requirement> ，其中：“version”是 ASVS 版本標籤。 例如：v4.0.3-1.11.3 將被理解為特指版本 4.0.3 中“架構”這一章中“業務邏輯架構”這一節的第 3 項要求。 （可概括為v<version>-<requirement\_identifier>。）

注意：版本部分前面的 v 要小寫。

如果使用不包括 v<version> 的識別符號，那麼它們應該被認為是指最新的應用安全驗證標準內容。顯然，隨著標準的增長和變化，這將導致問題，這就是為什麼作者或開發者應該將版本包括進去。

ASVS 需求列表，以 CSV、JSON 和其他可能對參考或程式設計有用的格式提供。

# 評估和認證

## OWASP對ASVS認證和信任標誌的立場

OWASP作為一個與供應商無關的非營利性組織，目前不認證任何供應商、驗證人員或軟體。

所有這類保證宣告、信任標誌或認證，均未經 OWASP 正式審查、註冊或認證，因此依賴此類觀點的組織，需要謹慎對待任何第三方的信任或聲稱ASVS認證的信任標誌。

這並不影響組織提供此類保證服務，只要他們不要求官方的 OWASP 認證。

## 認證組織指南

應用程式安全驗證標準，可以用作應用程式的公開驗證，包括對關鍵資源的開放和自由訪問（如架構師和開發人員、專案文件、原始碼），對測試系統的認證訪問（包括對每個角色的一個或多個帳戶的訪問），特別是L2和L3驗證。

從歷史上看，滲透測試和安全程式碼審查都包含“異常”問題——即只有未通過的測試項才會出現在最終報告中。 認證組織必須在任何報告中包括驗證的範圍（特別是某個關鍵元件不在範圍內時，如SSO身份驗證）、驗證結果的摘要，包括通過的和未通過的測試，並清楚地說明如何解決未通過的測試。

某些驗證要求可能不適用於被測試的應用程式。例如，如果你向客戶提供無狀態的服務層API而沒有客戶端實現，那麼“V3-會話管理”中的許多要求就不能直接使用。 在這種情況下，認證機構仍可聲稱完全符合 ASVS 的要求，但必須在報告中明確說明被排除的驗證要求不適用的原因。

保留詳細的工作底稿、螢幕截圖或視訊、可靠地重複利用一個問題的指令碼，以及測試的電子記錄，如攔截代理日誌和相關的筆記（如清理清單），被認為是標準的行業慣例，哪怕是對於最可疑的開發人員來說，它們也能作為調查結果的證明。 僅僅跑一個工具並報告故障是不夠的，這根本不能提供充分的證據，證明所有認證級別的問題都經過了徹底的測試。 在有爭議的情況下，應該有足夠的證據，來證明每一個經過驗證的需求確實被測試過。

### 測試方法

認證機構可自由選擇適當的測試方法，但應在報告中註明。

根據所測試的應用程式和驗證需求，可以使用不同的測試方法來獲得相似的結果置信度。 例如，要驗證應用程式輸入驗證機制的有效性，可以通過手動滲透測試或通過原始碼來分析。

#### 自動化安全測試工具的作用

鼓勵使用自動化滲透測試工具以提供儘可能多的覆蓋範圍。

僅使用自動滲透測試工具，是不可能完全完成ASVS驗證的。雖然L1中的絕大多數需求可以使用自動化測試來執行，但總體上，絕大多數需求並不適合自動化滲透測試。

請注意，隨著應用安全行業的成熟，自動化和手動測試之間的界限已經變得模糊。 自動化工具通常由專家手動調整，而手動測試人員通常會利用各種自動化工具。

#### 滲透測試的作用

在 4.0 版本中，我們決定讓 L1 完全可滲透測試，而無需訪問原始碼、文件或開發人員。 OWASP Top 10 2017 A10 要求的兩個日誌記錄專案，將需要訪談、螢幕截圖或其他證據，就像它們在 OWASP Top 10 2017 中的一樣。 然而，在無法獲得必要資訊的情況下進行測試，並不是一種理想的安全驗證方式，因為它不僅錯過了審查來源、識別威脅和缺失控制的可能性，還會錯過在更短的時間內進行更徹底測試的可能。

在可能的情況下，執行L2或L3評估時，需要訪問開發人員、文件、程式碼，以及訪問具有非生產資料的測試應用程式。 在這些級別進行的滲透測試，需要這種級別的訪問，我們稱之為 “混合審查” 或 “混合滲透測試”。

## ASVS的其他用途

除了用於評估應用程式的安全性外，我們還確定了ASVS的許多其他潛在用途。

### 作為詳細的安全架構指南

應用程式安全驗證標準的更常見用途之一，是作為安全架構師的資源。 Sherwood應用業務安全架（Sherwood Applied Business Security Architecture，SABSA）缺少大量的資訊，而這些資訊是完成一次徹底的應用安全架構審查所必需的。 ASVS可以用來填補這些空白，讓安全架構師為常見問題選擇更好的控制措施，如資料保護模式和輸入驗證策略。

### 作為現有安全編碼Checklists的替代品

許多組織可以從採用ASVS中受益，通過選擇三個級別中的一個，或通過fork ASVS，在特定領域改變每個應用風險級別的要求。 我們鼓勵這種fork，只要保持可追溯性，因此，如果一個應用程式已經通過了標準版本中的“要求4.1”，那麼也就通過了fork版本中的這個要求。

### 作為自動化單元和整合測試的指南

ASVS的設計是高度可測試的，唯一的例外是架構和惡意程式碼要求。 通過構建單元和整合測試，對相關的濫用情況進行fuzz測試，應用程式幾乎可以在每次構建中進行自我驗證。 例如，可以為登入控制器製作額外的測試，測試常見的預設使用者名稱引數、帳戶列舉、暴力破解、LDAP注入、SQL 注入以及 XSS。 同樣地，對密碼引數的測試，應該包括常用密碼、密碼長度、空位元組注入、移除引數、XSS等。

### 用於安全開發培訓

ASVS 還可用於定義安全軟體的特徵。 許多“安全編碼”課程只是帶有少量編碼技巧的道德黑客課程。 這不一定能幫助開發人員編寫更安全的程式碼。 相反，安全開發課程可以使用 ASVS，重點關注 ASVS 中的主動控制，而不是前 10 項不該做的負面事情。

### 作為敏捷應用安全的驅動程式

在敏捷開發過程中，為了獲得安全的產品，ASVS可以作為框架來定義團隊需要實施的特定任務。 一種可能的方法是：從 Level 1 開始，根據指定級別的 ASVS 要求，驗證特定應用程式或系統，查詢缺少哪些專案，並在待辦事項中提出特定工單/任務。 這有助於對具體任務進行優先排序（梳理），並使安全在敏捷開發中可見。 這也可用於確定組織中審計和審查任務的優先；其中，特定的 ASVS 要求，可以作為團隊成員審查、重構或審計的驅動因素，並可以記錄到最終的待辦清單中。

### 作為指導安全軟體採購的框架

ASVS 是一個很好的框架，可以幫助確保安全軟體的採購或定製開發服務的採購。 買方可以簡單地設定一個要求，即他們希望採購的軟體必須按照 ASVS 的 Level x 來開發，並要求賣方證明該軟體滿足ASVS的x級。

# V1 架構、設計和威脅建模

## 控制目標

在許多組織中，安全架構幾乎已成為一門失傳的藝術。 在DevSecOps時代，企業架構師的日子已經過去。應用安全領域必須迎頭趕上，採用敏捷安全原則，同時將領先的安全架構原則重新介紹給軟體從業者。 A架構不是一種實施，而是一種思考問題的方式，它可能有許多不同的答案，而沒有一個單一的“正確”答案。 很多時候，安全被視為不靈活的，要求開發人員以特定方式修復程式碼，而開發人員可能知道解決問題的更好方法。 對於架構來說，沒有單一的、簡單的解決方案，嘗試尋找這種方案，是對軟體工程領域的一種損害。

一個Web應用程式的具體實現，很可能在其生命週期中不斷被修改，但整體架構可能很少改變，而是緩慢發展。 安全架構也是一樣的，身份驗證——我們今天需要、明天需要、五年後也需要。 如果我們今天做出合理的決定，選擇和複用符合架構的解決方案，那麼就可以節省大量的精力、時間和金錢。 例如，十年前，多因素認證很少被實施。

如果開發人員已經在單一的“安全標識提供程式模型”上有所投入（例如SAML聯邦認證），身份提供者可以更新以納入新的要求（例如NIST 800-63標準），同時不改變原始應用程式的介面。 如果許多應用程式共享相同的安全架構和元件，那麼它們都將同時從這次升級中受益。 然而，SAML 並不總是最好或最合適的身份驗證解決方案——隨著需求的變化，可能需要替換為其他解決方案。 像這樣的更改要麼很複雜，成本高到需要完全重寫，要麼在沒有安全架構的情況下完全不可能。

在本章中，ASVS涵蓋了任何良好安全架構的主要方面：可用性、保密性、完整性、不可抵賴性和隱私。 這些安全原則中的每一條，都必須適用並內置於所有應用程式中。 “左移”至關重要，從安全編碼Checklists、指導和培訓、編碼和測試、構建、部署、配置和操作開始，到後續的獨立測試，確保所有安全控制存在且功能正常。 這最後一步，曾經是我們作為一個行業所做的一切，但當開發人員每天數十次或數百次地推送程式碼時，就已經不夠了。 應用安全專業人員必須跟上敏捷技術的步伐，這意味著要適應開發人員的工具，學習編碼，並與開發人員一起工作，而不是在其他人離開的幾個月後再批評專案。

## V1.1 安全軟體開發生命週期

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.1.1** | 驗證使用安全的軟體開發生命週期，在開發的各個階段解決安全問題。 ([C1](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ |  |
| **1.1.2** | 驗證在每次設計變更或sprint計劃中使用威脅建模，以識別威脅、計劃對策、促進適當的風險響應，並指導安全測試。 |  | ✓ | ✓ | 1053 |
| **1.1.3** | 驗證所有使用者資訊和功能是否包含功能安全約束，例如 “作為一個使用者，我應該能夠檢視和編輯我的個人資料。我不應該能夠檢視或編輯其他人的資料” |  | ✓ | ✓ | 1110 |
| **1.1.4** | 驗證應用程式所有的信任邊界、元件和重要資料流的文件，判斷其合理性。 |  | ✓ | ✓ | 1059 |
| **1.1.5** | 驗證應用程式的高階架構及遠端連線服務涉及的定義和安全分析。 ([C1](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 1059 |
| **1.1.6** | 驗證集中、簡單（設計）、安全、經過審查、和可重複使用的安全控制措施的實施情況，以避免重複、缺失、無效或不安全的控制措施。 ([C10](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 637 |
| **1.1.7** | 向所有開發人員和測試人員，驗證安全編碼Checklist、安全需求、指南或策略的可用性。 |  | ✓ | ✓ | 637 |

## V1.2 認證架構

在設計身份驗證時，如果攻擊者可以通過撥打客服電話，回答常見的問題來重置帳戶，那麼是否具有強大硬體支援的多因素身份驗證（MFA）並不重要。 在證明身份時，所有的認證途徑必須具有相同的強度。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.2.1** | 驗證應用程式所有的元件、服務和伺服器，是否使用了唯一或特殊的低許可權作業系統帳戶。 ([C3](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 250 |
| **1.2.2** | 驗證應用元件之間（包括 API、中介軟體和資料層）的通訊是否經過驗證。元件只具有最低的必要許可權。 ([C3](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 306 |
| **1.2.3** | 驗證應用程式是否使用已知安全的單一認證機制，可以擴充套件到強身份驗證，並有足夠的日誌記錄和監控，來檢測帳戶濫用或違規行為。 |  | ✓ | ✓ | 306 |
| **1.2.4** | 驗證所有的認證途徑和身份管理 API ，都實現了一致的認證安全控制強度， 以便收斂應用程式的風險。 |  | ✓ | ✓ | 306 |

## V1.3 會話管理架構

這是未來架構需求的佔位符。

## V1.4 訪問控制架構

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.4.1** | 驗證受信任的實施點（如訪問控制閘道器、伺服器和Serverless函式）是否實施了訪問控制。切勿在客戶端實施訪問控制。 |  | ✓ | ✓ | 602 |
| **1.4.2** | [已刪除，不可操作] |  |  |  |  |
| **1.4.3** | [已刪除，與 4.1.3 重複] |  |  |  |  |
| **1.4.4** | 驗證應用程式使用單一的、經過嚴格審查的訪問控制機制，來訪問受保護的資料和資源。 所有請求都必須通過這個單一機制，以避免複製、貼上或不安全的替代路徑。 ([C7](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 284 |
| **1.4.5** | 驗證是否使用基於屬性/特徵的訪問控制，即程式碼應檢查使用者對某一特徵/資料項的授權，而不僅僅是他們的角色。 許可權仍應依照不同角色進行分配。 ([C7](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 275 |

## V1.5 輸入和輸出架構

在4.0中，我們已經不再把 “伺服器端” 作為一個表示信任邊界的術語。 信任邊界仍然令人擔憂——在不受信任的瀏覽器或客戶端裝置上做決定，很容易被繞過的。 然而，在今天的主流架構部署中，信任的執行點已經發生了巨大的變化。 因此，在 ASVS 中使用 “受信任的服務層” 這一術語時，我們描述的都是受信任的執行點，無論其位置如何，如微服務、Serverless API、伺服器端、具有安全啟動的客戶端裝置上的受信任的API、合作伙伴或外部API等等。

這裡的“不受信任的客戶端”一詞，是指呈現表示層的客戶端技術，通常稱為“前端”技術。 這裡的術語“序列化”不僅表示通過網路傳送資料（如一個數組的值或獲取 JSON 結構），還指傳遞可以包含邏輯的複雜物件。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.5.1** | 驗證輸入和輸出要求，明確規定如何根據型別、內容以及適用的法律、法規和其他政策規定，來操作和處理資料。 |  | ✓ | ✓ | 1029 |
| **1.5.2** | 驗證在與不受信任的客戶進行通訊時，不使用序列化。 如果無法做到這一點，請確保執行足夠的完整性控制（如果傳送敏感資料，可能還要進行加密），以防止反序列化攻擊，包括物件注入。 |  | ✓ | ✓ | 502 |
| **1.5.3** | 驗證輸入驗證是否在可信的服務層上執行。 ([C5](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 602 |
| **1.5.4** | 驗證輸出編碼是否發生在其預期的直譯器附近（或由直譯器進行）。 ([C4](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 116 |

## V1.6 加密架構

應用程式需要設計強大的加密架構，以根據其分類保護資料資產。加密所有東西，是浪費；不加密任何東西，是法律上的疏忽。 在架構或頂層設計、設計衝刺（Design Sprint）或架構高峰期，通常需要取得一種平衡。 一邊設計加密技術，一邊進行開發迭代，這樣的安全實施，其成本不可避免地要比一開始就做簡單的構建要高得多。

架構要求是整個程式碼庫的內在要求，因此很難進行單元或整合測試。架構需求需要在整個編碼階段的編碼標準中加以考慮，並應在安全架構、程式碼審查或覆盤會議中加以審查。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.6.1** | 驗證是否有明確的加密金鑰管理政策，以及加密金鑰的生命週期是否遵循金鑰管理標準，如NIST SP 800-57。 |  | ✓ | ✓ | 320 |
| **1.6.2** | 驗證密碼服務的消費者是否通過使用金鑰庫或基於API的替代方案，來保護金鑰材料和其他機密。 |  | ✓ | ✓ | 320 |
| **1.6.3** | 驗證所有的金鑰和密碼是否可替換的，並且是重新加密敏感資料的明確定義流程的一部分。 |  | ✓ | ✓ | 320 |
| **1.6.4** | 驗證架構是否將客戶端機密（例如對稱金鑰、密碼或 API 令牌）視為不安全的，並且從不使用它們來保護或訪問敏感資料。 |  | ✓ | ✓ | 320 |

## V1.7 錯誤、日誌和審計架構

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.7.1** | 驗證整個系統是否使用了通用的日誌記錄格式和方法。 ([C9](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 1009 |
| **1.7.2** | 驗證日誌是否安全地傳輸到遠端系統，以便進行分析、檢測、報警和升級。 ([C9](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ |  |

## V1.8 資料保護和隱私架構

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.8.1** | 驗證所有敏感資料都已識別並歸入保護級別。 |  | ✓ | ✓ |  |
| **1.8.2** | 驗證所有保護級別都具有一套相關的保護要求，如加密要求、完整性要求、保留、隱私和其他機密性要求，並在架構中應用這些要求。 |  | ✓ | ✓ |  |

## V1.9 通訊架構

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.9.1** | 驗證應用程式對元件之間的通訊進行加密，特別是當這些元件處於不同的容器、系統、站點或雲提供商時。 ([C3](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 319 |
| **1.9.2** | 驗證應用元件是否驗證了通訊連結中每一方的真實性，以防止中間人攻擊。例如，應用程式元件應校驗TLS證書鏈。 |  | ✓ | ✓ | 295 |

## V1.10 惡意軟體架構

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.10.1** | 驗證是否使用了原始碼控制系統，以及有程式確保簽入時附帶問題或變更單。原始碼控制系統應該具有訪問控制和可識別的使用者，以追溯任何的更改。 |  | ✓ | ✓ | 284 |

## V1.11 業務邏輯架構

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.11.1** | 驗證所有應用元件在其提供的業務或安全功能方面的定義和文件。 |  | ✓ | ✓ | 1059 |
| **1.11.2** | 驗證所有高價值的業務邏輯流，包括認證、會話管理和訪問控制，不共享不同步的狀態。 |  | ✓ | ✓ | 362 |
| **1.11.3** | 驗證所有高價值的業務邏輯流，包括身份驗證、會話管理和訪問控制都是執行緒安全的，並能抵抗檢查時間和使用時間不同步時的條件競爭。 |  |  | ✓ | 367 |

## V1.12 安全上傳架構

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.12.1** | [已刪除，與 12.4.1 重複] |  |  |  |  |
| **1.12.2** | 驗證使用者上傳的檔案——如果需要顯示或從應用中下載，是通過二進位制流下載，或從無關的域（如雲檔案儲存桶）提供。實施合適的內容安全策略（CSP），以減少來自上傳檔案的XSS向量或其他攻擊的風險。 |  | ✓ | ✓ | 646 |

## V1.13 API架構

這是未來架構需求的佔位符。

## V1.14 配置架構

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **1.14.1** | 通過明確的安全控制、防火牆規則、API 閘道器、反向代理、基於雲的安全組或類似機制，驗證不同信任級別的元件的隔離情況。 |  | ✓ | ✓ | 923 |
| **1.14.2** | 驗證二進位制簽名、可信連線和經過驗證的介面，以將二進位制檔案部署到遠端裝置。 |  | ✓ | ✓ | 494 |
| **1.14.3** | 驗證構建管道是否對過期或不安全的元件發出警告並採取適當的行動。 |  | ✓ | ✓ | 1104 |
| **1.14.4** | 驗證構建管道是否包含自動構建和驗證應用安全部署的構建步驟，特別是當應用基礎設施是軟體定義時，例如雲環境構建指令碼。 |  | ✓ | ✓ |  |
| **1.14.5** | 驗證應用程式部署是否在網路級別進行了充分的沙盒化、容器化或隔離，以延遲和阻止攻擊者攻擊其他應用程式，尤其是當攻擊者執行敏感或危險操作時（如反序列化）。 ([C5](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 265 |
| **1.14.6** | 驗證應用程式未使用不受支援、不安全或不推薦的客戶端技術，如NSAPI外掛、Flash、Shockwave、ActiveX、Silverlight、NACL或客戶端Java applets。 |  | ✓ | ✓ | 477 |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Threat Modeling Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html)
* [OWASP Attack Surface Analysis Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Attack_Surface_Analysis_Cheat_Sheet.html)
* [OWASP Threat modeling](https://owasp.org/www-community/Application_Threat_Modeling)
* [OWASP Software Assurance Maturity Model Project](https://owasp.org/www-project-samm/)
* [Microsoft SDL](https://www.microsoft.com/en-us/sdl/)
* [NIST SP 800-57](https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final)

# V2 認證

## 控制目標

認證是建立或確認某人（或某物）的真實性，並且個人或裝置的宣告是正確的，可防止假冒，並防止恢復或攔截密碼。

當ASVS首次釋出時，使用者名稱+密碼是最常見的認證形式（除高安全系統以外）。 多因素身份驗證（MFA）在安全界被普遍接受，但在其他地方很少需要。 隨著密碼洩露次數的增加，認為使用者名稱在某種程度上是保密的，而密碼則是未知的這種想法使得許多安全控制無法成立。 例如，NIST 800-63 將使用者名稱和基於知識的身份驗證（KBA）視為公共資訊，將 SMS 和電子郵件通知視為[“受限”的認證型別](https://pages.nist.gov/800-63-FAQ/#q-b1)，而密碼是預先洩露的。 這一現實使基於知識的認證器、簡訊和電子郵件恢復、密碼歷史、複雜性和輪換控制變得毫無用處。 這些控制措施不總那麼有用，經常迫使使用者每隔幾個月就想出一些弱的密碼，但是隨著50多億使用者名稱和密碼洩露事件的公佈，現在是時候繼續前進了。

在ASVS的所有章節中，認證和會話管理章節的變化最大。採用有效的、以證據為基礎的領先實踐，對許多人來說將是挑戰，這完全沒問題。 現在我們必須開始向未來的後密碼時代過渡。

## NIST 800-63 - 現代的、基於證據的認證標準

[NIST 800-63b](https://pages.nist.gov/800-63-3/sp800-63b.html) 是一種現代的、基於證據的標準，代表了可用的最佳建議，無論其適用性如何。該標準對世界各地的所有組織都有幫助，但與美國機構和與美國機構打交道的機構尤其相關。

NIST 800-63的術語一開始可能有點令人困惑，特別是你只習慣於使用者名稱+密碼認證的話。 現代認證的進步是必要的，所以我們必須引入將來會變得司空見慣的術語，但我們確實理解：在行業落實這些新術語之前，理解這些新術語的困難。 我們在本章末尾提供了一個詞彙表，以提供幫助。 我們重新表述了許多要求，以滿足要求的意圖，而不只是拘泥於文字。 例如，當NIST在本標準中使用“記憶祕密”時（memorized secret），ASV使用術語“密碼”（password）。

ASVS V2 身份驗證、V3 會話管理以及在較小程度上的V4 訪問控制，已被調整為符合 NIST 800-63b 控制項的一個子集，主要圍繞常見的威脅和經常被利用的認證弱點。如果需要完全遵守NIST 800-63，請參考NIST 800-63。

### 選擇合適的 NIST AAL 級別

應用程式安全驗證標準（ASVS），已嘗試將 ASVS L1 對應到 NIST AAL1 要求，將 L2 對應到 AAL2，將 L3 對應到 AAL3。 然而，ASVS Level 1作為“基本”的控制，不一定是驗證應用或API的正確AAL級別。 例如，如果該應用是 L3 應用或有 AAL3 的監管要求，則應在V2和V3會話管理章節選擇 L3。 應根據NIST 800-63b指南選擇符合NIST標準的認證保證級別（AAL），如[NIST 800-63b第6.2節](https://pages.nist.gov/800-63-3/sp800-63-3.html#AAL_CYOA) 中的 *Selecting AAL* 。

## 圖例

應用程式總是可以超過當前級別的要求，特別是如果現代認證是在應用程式的路線圖上。以前，ASVS要求強制MFA。NIST不要求強制MFA。因此，我們在本章中使用了一個可選的指定，以表明ASVS鼓勵但不要求控制的地方。本標準自始至終使用了以下圖示：

|  |  |
| --- | --- |
| 標記 | 說明 |
|  | 不要求 |
| o | 建議，但不要求 |
| ✓ | 要求 |

## V2.1 密碼安全

在NIST 800-63中，密碼被稱為“記憶的祕密”（Memorized Secrets），包括密碼、PIN、解鎖圖案、選擇正確的小貓或其他影象元素以及密碼短語。 它們通常被認為是“您知道的東西”，並且通常用作單因素身份認證工具。 繼續使用單因素認證有很大的風險，包括網際網路上披露的數十億有效使用者名稱和密碼、預設或弱密碼、彩虹表和最常見密碼的有序字典。

應用程式應強烈鼓勵使用者註冊多因素認證，並應允許使用者重新使用他們已經擁有的令牌，如FIDO或U2F令牌，或連結到提供多因素認證的憑證服務提供商。

憑據服務提供商（CSP）為使用者提供聯合身份。 使用者通常會擁有多個CSP的多個身份，例如使用Azure AD、Okta、Ping identity或Google的企業身份，或使用Facebook、Twitter、Google或微信的普通使用者，這只是一些常見的可能。 這份清單並不是對這些公司或服務的認可，而只是鼓勵開發者考慮使用者有許多既定身份的現實。 組織應該考慮與現有的使用者身份整合，根據CSP的身份證明強度的風險狀況，組織應考慮與現有使用者身份整合。 例如，政府機構不太可能接受社交媒體身份作為敏感系統的登入名，因為很容易偽造或丟棄身份，而移動遊戲公司可能需要與主要社交媒體平臺整合，以擴大他們的活躍玩家群。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.1.1** | 驗證使用者設定的密碼長度至少為 12 個字元（多個空格合併後）。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| **2.1.2** | 驗證是否允許64個字元以上的密碼，並拒絕超過128個字元的密碼。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| **2.1.3** | 驗證不進行密碼截斷。然而，連續的多個空格可以被單個空格代替。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| **2.1.4** | 驗證密碼中是否允許使用任何可列印的Unicode字元，包括語言中立字元，例如空格和表情符號。 | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| **2.1.5** | 驗證使用者可以更改其密碼。 | ✓ | ✓ | ✓ | 620 | 5.1.1.2 |
| **2.1.6** | 驗證密碼更改功能是否需要使用者的當前密碼和新密碼。 | ✓ | ✓ | ✓ | 620 | 5.1.1.2 |
| **2.1.7** | 驗證在賬戶註冊、登入和密碼更改過程中提交的密碼，是否出現在被洩露過的密碼中，這些密碼可以是本地的（如符合系統密碼策略的前1000個或10000個最常見的密碼），也可以使用外部API。 如果使用API，應使用零知識證明或其他機制，以確保純文字密碼不被髮送或用於驗證密碼的違反狀態。 如果密碼被洩露，應用程式必須要求使用者設定一個新的未被洩露的密碼。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| **2.1.8** | 驗證是否提供了密碼強度表，以幫助使用者設定更強的密碼。 | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| **2.1.9** | 驗證是否有限制允許的字元型別的密碼組成規則。對大寫或小寫、數字或特殊字元不應有任何要求。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| **2.1.10** | 驗證沒有定期更換憑證或密碼歷史的要求。 | ✓ | ✓ | ✓ | 263 | 5.1.1.2 |
| **2.1.11** | 驗證是否允許 “貼上” 功能、瀏覽器密碼輔助工具和外部密碼管理器。 | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |
| **2.1.12** | 驗證使用者可以選擇臨時檢視整個遮蔽的密碼，或者在沒有內建功能的平臺上臨時檢視密碼的最後輸入的字元。 | ✓ | ✓ | ✓ | 521 | 5.1.1.2 |

注意：允許使用者檢視密碼或臨時檢視最後一個字元的目的，是為了提高憑證輸入的可用性，尤其是在使用更長的密碼、口令和密碼管理器時。 包含該要求的另一個原因，是為了防止測試報告不必要地要求組織重寫內建平臺密碼欄位的行為，從而保持這種現代使用者友好的安全體驗。

## V2.2 通用身份驗證器的安全性

身份驗證器的敏捷性，對於面向未來的應用程式至關重要。 重構應用程式驗證器以允許使用者根據偏好新增額外的驗證器，並允許以有序的方式停用已棄用或不安全的驗證器。

NIST 將電子郵件和 SMS 視為 [“受限”的身份驗證器型別](https://pages.nist.gov/800-63-FAQ/#q-b1)，它們很可能在未來的某個時候從NIST 800-63以及ASVS中刪除。 應用程式應計劃一個不需要使用電子郵件或簡訊的路線圖。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.2.1** | 驗證反自動化控制的措施能夠有效地緩解被洩露的憑證測試、暴力破解和賬戶鎖定攻擊。 這些控制措施包括阻止最常見的洩露密碼、軟鎖定、速率限制、驗證碼、每次嘗試後逐漸增加的間隔時間、IP地址限制，或基於風險的限制，例如位置、裝置上的首次登入、最近解鎖賬戶的嘗試等類似情況。 驗證單個帳戶每小時的失敗嘗試次數不超過 100 次。 | ✓ | ✓ | ✓ | 307 | 5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2 |
| **2.2.2** | 驗證弱身份驗證器（例如 SMS 和電子郵件）的使用，僅限於二次驗證和批准交易，而不是作為更安全的認證方法的替代。 驗證是否在弱方法之前提供了更強的方法，使用者是否意識到風險，或者是否採取了適當的措施來限制帳戶洩露的風險。 | ✓ | ✓ | ✓ | 304 | 5.2.10 |
| **2.2.3** | 驗證在更新認證資訊（如憑證重置、電子郵件或地址變更、從未知或風險地點登入）後向使用者傳送安全通知。 最好使用推送通知——而不是簡訊或電子郵件，但在沒有推送通知的情況下，只要通知中沒有披露敏感資訊，簡訊或電子郵件也是可以接受的。 | ✓ | ✓ | ✓ | 620 |  |
| **2.2.4** | 驗證對網路釣魚的抗冒充性，如使用多因素認證、有意圖的加密裝置（如有推送認證的連線金鑰），或在更高的AAL級別，客戶端證書。 |  |  | ✓ | 308 | 5.2.5 |
| **2.2.5** | 驗證當憑證服務提供者（CSP）和驗證認證的應用程式分開時，兩個端點之間有相互認證的TLS（mTLS）。 |  |  | ✓ | 319 | 5.2.6 |
| **2.2.6** | 驗證抗重放性，是否通過強制使用一次性密碼（OTP）裝置、加密認證器或查詢程式碼。 |  |  | ✓ | 308 | 5.2.8 |
| **2.2.7** | 通過要求輸入OTP令牌或使用者發起的動作（如按下FIDO硬體鑰匙的按鈕）來驗證認證意圖。 |  |  | ✓ | 308 | 5.2.9 |

## V2.3 身份驗證器生命週期

認證器是密碼、軟令牌、硬體令牌和生物識別裝置。認證器的生命週期對應用程式的安全至關重要——如果任何人都可以在沒有身份證明的情況下自行註冊一個賬戶，那麼對身份斷言的信任就會很低。 對於像Reddit這樣的社交媒體網站，這是完全可以的。 對於銀行系統來說，更加註重憑證和裝置的註冊和發放（對應用程式的安全至關重要）。

注意：密碼不要有最長的使用壽命，也不要進行密碼輪換。 應檢查密碼是否已洩露，而不是定期更換。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.3.1** | 驗證系統生成的初始密碼或啟用碼應該是安全隨機生成的，應該至少有6個字元的長度，可以包含字母和數字，並在短時間內過期。這些初始祕密不得被允許成為長期密碼。 | ✓ | ✓ | ✓ | 330 | 5.1.1.2 / A.3 |
| **2.3.2** | 驗證是否支援註冊和使用使用者提供的認證裝置，如U2F或FIDO令牌。 |  | ✓ | ✓ | 308 | 6.1.3 |
| **2.3.3** | 驗證更新指令的傳送時間是否足夠，以更新有時間限制的認證器。 |  | ✓ | ✓ | 287 | 6.1.4 |

## V2.4 憑證儲存

架構師和開發人員在構建或重構程式碼時，應遵守本節。本節內容只能使用原始碼審查或通過安全單元或整合測試來完全驗證。滲透測試不能識別這裡面的任何問題。

經批准的單向金鑰推導功能列表詳見NIST 800-63 B第5.1.1.2節，以及 [BSI Kryptographische Verfahren: Empfehlungen und Schlussellängen (2018)](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile)。除了上面這些選擇，還可以根據最新的國家或地區選擇演算法和金鑰長度標準。

此部分無法進行滲透測試，因此控制措施未標記為 L1。但是，如果證書被盜，這部分對證書的安全至關重要，因此如果為架構或編碼指南或原始碼審查清單Fork ASVS，請在您的私有版本中將這些控制措施放回 L1。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.4.1** | 驗證密碼是以一種可以抵抗離線攻擊的形式儲存的。密碼應使用認可的單向金鑰推導或密碼雜湊函式進行加鹽和雜湊。金鑰推導和密碼雜湊函式，在生成密碼雜湊時，將密碼、鹽和計算成本作為輸入。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 916 | 5.1.1.2 |
| **2.4.2** | 驗證鹽的長度至少為32位，並且是任意選擇的，以減少儲存的雜湊值之間的碰撞。對於每個憑證，應儲存唯一的鹽值和由此產生的雜湊值。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 916 | 5.1.1.2 |
| **2.4.3** | 驗證如果使用 PBKDF2，迭代次數應在驗證伺服器效能允許的範圍內，一般至少為100,000次迭代。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 916 | 5.1.1.2 |
| **2.4.4** | 驗證如果使用 bcrypt，工作係數應在驗證伺服器效能允許的範圍內儘量大，最小為10。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 916 | 5.1.1.2 |
| **2.4.5** | 驗證是否執行了金鑰派生函式的額外迭代，使用的是隻有驗證者知道的祕密鹽值。使用經批准的隨機位生成器 [SP 800-90Ar1] 生成鹽值，並至少提供 SP 800-131A 最新修訂版中規定的最低安全強度。祕密鹽值應與雜湊密碼分開儲存（例如，在像硬體安全模組這樣的專用裝置中）。 |  | ✓ | ✓ | 916 | 5.1.1.2 |

在提到美國標準時，可以根據需要使用地區或當地標準來代替或補充美國標準。

## V2.5 憑證恢復

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.5.1** | 驗證系統生成的初始啟用或恢復密碼，不會以明文形式傳送給使用者。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 640 | 5.1.1.2 |
| **2.5.2** | 驗證密碼提示或基於知識的身份驗證（所謂的“密碼保護問題”）不存在。 | ✓ | ✓ | ✓ | 640 | 5.1.1.2 |
| **2.5.3** | 驗證密碼憑據恢復不會以任何方式洩露當前密碼。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 640 | 5.1.1.2 |
| **2.5.4** | 驗證共享或預設帳戶不存在（例如“root”、“admin”或“sa”）. | ✓ | ✓ | ✓ | 16 | 5.1.1.2 / A.3 |
| **2.5.5** | 驗證如果更改或替換了身份驗證因素，則使用者會收到此事件的通知。 | ✓ | ✓ | ✓ | 304 | 6.1.2.3 |
| **2.5.6** | 驗證忘記密碼以及其他恢復路徑，使用了安全的恢復機制，例如基於時間的OTP（TOTP）或其他軟令牌、移動推送或其他離線恢復機制。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 640 | 5.1.1.2 |
| **2.5.7** | 驗證如果OTP或多因素身份驗證因素丟失，身份證明的執行水平與註冊時相同。 |  | ✓ | ✓ | 308 | 6.1.2.3 |

## V2.6 查詢密碼認證

查詢密碼（譯者注：Look-up secrets，類似於密碼保護卡），是預先生成的祕密程式碼列表，類似於交易授權號碼（TAN）、社交媒體恢復程式碼，或包含一組隨機值的網格。 這些被安全地分配給使用者。這些 查詢程式碼只使用一次，一旦全部使用，查詢的祕密列表就會被丟棄。這種型別的認證器被認為是 “你擁有的東西”。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.6.1** | 驗證查詢密文只能使用一次。 |  | ✓ | ✓ | 308 | 5.1.2.2 |
| **2.6.2** | 驗證查詢祕密有足夠的隨機性（112位熵），如果少於112位熵，則用唯一的隨機32位鹽進行加鹽，並用認可的單向雜湊進行雜湊。 |  | ✓ | ✓ | 330 | 5.1.2.2 |
| **2.6.3** | 驗證查詢祕密能夠抵抗離線攻擊，例如可預測的值。 |  | ✓ | ✓ | 310 | 5.1.2.2 |

## V2.7 帶外驗證器

在過去，常見的帶外驗證器是包含密碼重置連結的電子郵件或簡訊。攻擊者利用這種薄弱的機制來重置他們尚未控制的賬戶，例如接管一個人的電子郵件賬戶並重新使用任何發現的重置連結。 處理帶外驗證有更好的方法。

安全的帶外驗證器是可以通過安全的二級通道與驗證器進行通訊的物理裝置。 例子包括向移動裝置推送通知。這種型別的驗證器被認為是 “你擁有的東西”。 當用戶希望進行認證時，驗證應用程式通過與認證器的連線，直接或間接地通過第三方服務向帶外認證器傳送一個訊息。 該訊息包含一個認證程式碼（通常是一個隨機的六位數或一個模式化的批准對話方塊）。 驗證應用程式等待通過主通道接收驗證碼，並將接收到的值的雜湊值與原始驗證碼的雜湊值進行比較。如果它們匹配，則帶外驗證器可以認為使用者已通過身份驗證。

ASVS假定只有少數開發者會開發新的帶外認證器，如推送通知，因此以下ASVS控制措施適用於驗證器，如認證API、應用程式和單點登入實現。 如果開發一個新的帶外認證器，請參考 NIST 800-63B § 5.1.3.1。

不安全的帶外認證器，如電子郵件和VOIP是不允許的。 PSTN和SMS認證目前受到NIST的 “限制”，應該被棄用，以支援推送通知或類似的方式。 如果你需要使用電話或簡訊的帶外認證，請參見 NIST 800-63B § 5.1.3.3。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.7.1** | 驗證預設情況下不提供簡訊或PSTN等帶外的明文認證器，並首先提供推送通知等更強的替代方案。 | ✓ | ✓ | ✓ | 287 | 5.1.3.2 |
| **2.7.2** | 驗證帶外驗證器在10分鐘後將帶外驗證請求、程式碼或令牌過期。 | ✓ | ✓ | ✓ | 287 | 5.1.3.2 |
| **2.7.3** | 驗證帶外驗證器身份驗證請求、程式碼或令牌僅可使用一次，並且僅可用於原始身份驗證請求。 | ✓ | ✓ | ✓ | 287 | 5.1.3.2 |
| **2.7.4** | 驗證帶外驗證器和驗證器是否通過安全的獨立通道進行通訊。 | ✓ | ✓ | ✓ | 523 | 5.1.3.2 |
| **2.7.5** | 驗證帶外驗證器只保留認證程式碼的雜湊版本。 |  | ✓ | ✓ | 256 | 5.1.3.2 |
| **2.7.6** | 驗證初始驗證碼是否由安全隨機數生成器生成，包含至少 20 位熵（通常為 6 位數字隨機數即可）。 |  | ✓ | ✓ | 310 | 5.1.3.2 |

## V2.8 一次性驗證器

單因素一次性密碼（OTP），是顯示持續變化的偽隨機一次性挑戰的物理或軟令牌。 這些裝置使網路釣魚（冒充）變得困難，但並非不可能。 這種型別的身份驗證器被認為是“你擁有的東西”。 多因素令牌，類似於單因素 OTP，但需要輸入有效的 PIN 碼、生物識別解鎖、USB 插入、NFC 配對或一些附加值（例如交易簽名計算器）才能建立最終 OTP。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.8.1** | 驗證基於時間的OTP在過期前有確定的使用壽命 | ✓ | ✓ | ✓ | 613 | 5.1.4.2 / 5.1.5.2 |
| **2.8.2** | 驗證用於驗證提交的OTP的對稱金鑰是否被高度保護，例如使用硬體安全模組或基於安全作業系統的金鑰儲存。 |  | ✓ | ✓ | 320 | 5.1.4.2 / 5.1.5.2 |
| **2.8.3** | 驗證OTP的生成、播種和驗證是否使用了經過批准的加密演算法。 |  | ✓ | ✓ | 326 | 5.1.4.2 / 5.1.5.2 |
| **2.8.4** | 驗證基於時間的OTP在有效期內只能使用一次。 |  | ✓ | ✓ | 287 | 5.1.4.2 / 5.1.5.2 |
| **2.8.5** | 驗證如果基於時間的多因素OTP令牌在有效期內被重複使用，將被記錄並拒絕，同時向裝置持有者傳送安全通知。 |  | ✓ | ✓ | 287 | 5.1.5.2 |
| **2.8.6** | 驗證物理單因素OTP生成器在被盜或其他損失的情況下可以被撤銷。確保撤銷在登入會話中立即生效，無論在何處。 |  | ✓ | ✓ | 613 | 5.2.1 |
| **2.8.7** | 驗證生物特徵身份驗證器僅限於用作次要因素，與“你擁有的東西”和“你知道的東西”一起使用。 |  | o | ✓ | 308 | 5.2.3 |

## V2.9 密碼驗證器

加密安全金鑰是智慧卡或FIDO金鑰，使用者必須將加密裝置插入或配對到計算機上才能完成身份驗證。驗證者立即向加密裝置或軟體傳送挑戰隨機數，裝置或軟體根據安全儲存的加密金鑰計算出響應。

對單因素密碼裝置和軟體的要求，和多因素密碼裝置和軟體的要求是一樣的，都是證明密碼認證者擁有認證因素。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.9.1** | 驗證用於驗證的加密金鑰是否安全儲存並防止洩露，例如使用可信平臺模組（TPM）或硬體安全模組（HSM），或可以使用這種安全儲存的作業系統服務。 |  | ✓ | ✓ | 320 | 5.1.7.2 |
| **2.9.2** | 驗證質詢隨機數的長度至少為 64 位，並且在統計學上是唯一的，或在加密裝置的生命週期內是唯一的。 |  | ✓ | ✓ | 330 | 5.1.7.2 |
| **2.9.3** | 驗證在生成、播種和驗證中使用經批准的加密演算法。 |  | ✓ | ✓ | 327 | 5.1.7.2 |

## V2.10 服務認證

此部分不可滲透測試，因此沒有任何 L1 要求。但是，如果用於架構、編碼或安全程式碼審查，請假設軟體（就像 Java 金鑰庫一樣）是 L1 的最低要求。 在任何情況下都不允許明文儲存祕密。

注：

* HSM，硬體安全模組（Hardware Security Module）
* OS assisted，作業系統協助

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **2.10.1** | 驗證服務內機密不依賴於不變的憑據，例如密碼、API 金鑰或具有特權訪問許可權的共享帳戶。 |  | OS assisted | HSM | 287 | 5.1.1.1 |
| **2.10.2** | 驗證如果服務身份驗證需要密碼，則使用的服務帳戶不是預設憑據（例如，root/root 或 admin/admin 是安裝過程中某些服務的預設設定）。 |  | OS assisted | HSM | 255 | 5.1.1.1 |
| **2.10.3** | 驗證儲存的密碼是否有足夠的保護，以防止離線恢復攻擊，包括本地系統訪問。 |  | OS assisted | HSM | 522 | 5.1.1.1 |
| **2.10.4** | 驗證密碼、與資料庫和第三方系統的整合、種子和內部機密以及 API 金鑰都得到安全管理，不包含在原始碼中或儲存在原始碼儲存庫中。 這種儲存應能抵禦離線攻擊。建議使用安全的軟體金鑰儲存（L1）、硬體 TPM 或 HSM（L3）來儲存密碼。 |  | OS assisted | HSM | 798 |  |

## 美國機構的其他要求

美國機構對 NIST 800-63 有強制性要求。 一直以來，應用安全驗證標準都是對幾乎100%的應用採取80%的控制措施，而不是最後20%的高階控制或那些有限的適用性。 因此，ASVS 是 NIST 800-63 的一個嚴格的子集，特別是對於 IAL1/2 和 AAL1/2 分類，但不夠全面，特別是關於 IAL3/AAL3 分類。

我們強烈敦促美國政府機構全面審查和實施 NIST 800-63。

## 術語表

|  |  |
| --- | --- |
| 術語 | 含義 |
| CSP | 憑證服務提供者也稱為身份提供者 |
| Authenticator | 驗證密碼、令牌、MFA、聯合斷言等的程式碼。 |
| Verifier | “通過使用認證協議，驗證申請人對一個或兩個認證器的擁有和控制，來驗證申請人的身份的實體。 為此，驗證者可能還需要驗證將認證器與使用者的識別符號聯絡起來的憑證，並檢查其狀態” |
| OTP | 一次性密碼 |
| SFA | 單因素身份驗證，例如“您知道的東西”（記憶的祕密、密碼、密碼短語、pin），“您的特徵“（生物特徵識別、指紋、面部掃描），或“您擁有的東西“（OTP令牌、智慧卡等加密裝置）， |
| MFA | 多因素認證，包括兩個或多個單因素 |

## 參考文獻

有關更多資訊，請參閱：

* [NIST 800-63 - Digital Identity Guidelines](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf)
* [NIST 800-63 A - Enrollment and Identity Proofing](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63a.pdf)
* [NIST 800-63 B - Authentication and Lifecycle Management](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf)
* [NIST 800-63 C - Federation and Assertions](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63c.pdf)
* [NIST 800-63 FAQ](https://pages.nist.gov/800-63-FAQ/)
* [OWASP Testing Guide 4.0: Testing for Authentication](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/04-Authentication_Testing/README.html)
* [OWASP Cheat Sheet - Password storage](https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html)
* [OWASP Cheat Sheet - Forgot password](https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html)
* [OWASP Cheat Sheet - Choosing and using security questions](https://cheatsheetseries.owasp.org/cheatsheets/Choosing_and_Using_Security_Questions_Cheat_Sheet.html)

# V3 會話管理

## 控制目標

任何基於Web的應用程式或有狀態的API的核心組成部分之一，是它控制和維護與之互動的使用者或裝置的狀態的機制。會話管理將無狀態協議變為有狀態協議，這對於區分不同的使用者或裝置至關重要。

確保經過驗證的應用程式滿足以下高階會話管理需求:

* 會話對每個人都是唯一的，不能被猜測或共享。
* 會話在不再需要時失效，在不活動期間超時。

如前所述，這些要求已被調整為符合 NIST 800-63b 控制的一個子集，重點關注常見的威脅和經常被利用的認證弱點。 以前的驗證要求已被淘汰、去重，或者在大多數情況下已調整為與強制性 [NIST 800-63b](https://pages.nist.gov/800-63-3/sp800-63b.html) 要求的意圖保持高度一致。

## 安全驗證要求

## V3.1 基本會話管理安全

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **3.1.1** | 驗證應用不會在URL引數中顯示會話令牌。 | ✓ | ✓ | ✓ | 598 |  |

## V3.2 會話繫結

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **3.2.1** | 驗證應用程式在使用者身份驗證時，生成新的會話令牌。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 384 | 7.1 |
| **3.2.2** | 驗證會話令牌具有至少 64 位的熵。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 331 | 7.1 |
| **3.2.3** | 驗證應用程式僅使用安全方法在瀏覽器中儲存會話令牌，例如適當的 cookie保護（參見第 3.4 節）或 HTML 5 會話儲存。 | ✓ | ✓ | ✓ | 539 | 7.1 |
| **3.2.4** | 驗證會話令牌是使用批准的加密演算法生成的。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 331 | 7.1 |

會話管理必須使用 TLS 或其他安全傳輸通道。這在“通訊安全”一章中有介紹。

## V3.3 會話終止

會話超時已經與NIST 800-63保持一致，它允許比傳統安全標準所允許的更長的會話超時。組織應查看下錶，如果應用的風險需要更長的超時，NIST值應是會話空閒超時的上限。

在此上下文中，L1 是 IAL1/AAL1，L2 是 IAL2/AAL3，L3 是 IAL3/AAL3。對於 IAL2/AAL2 和 IAL3/AAL3，空閒超時時間越短，表示登出或重新認證恢復會話的時間越短。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **3.3.1** | 驗證登出和到期是否會使會話令牌無效，以便後退按鈕或下游依賴方不會恢復經身份驗證過的會話。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 613 | 7.1 |
| **3.3.2** | 如果認證器允許使用者保持登入狀態，請驗證在活躍使用或空閒一段時間過後，定期進行重新認證。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | 30天 | 12小時 或 30分鐘不活動，可選2FA | 12小時 或 15 分鐘不活動，使用2FA | 613 | 7.2 |
| **3.3.3** | 驗證應用程式是否提供了在成功更改密碼（包括通過密碼重置/恢復）後終止所有其他活動會話的選項，並且這在應用程式、聯合登入（如果存在）和任何依賴方中都是有效的。 |  | ✓ | ✓ | 613 |  |
| **3.3.4** | 驗證使用者能夠檢視並（在重新輸入登入憑證後）登出當前的所有活動會話和裝置。 |  | ✓ | ✓ | 613 | 7.1 |

## V3.4 基於 Cookie 的會話管理

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **3.4.1** | 驗證基於 cookie 的會話令牌是否設定了'Secure'屬性。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 614 | 7.1.1 |
| **3.4.2** | 驗證基於 cookie 的會話令牌是否設定了'HttpOnly'屬性。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 1004 | 7.1.1 |
| **3.4.3** | 驗證基於cookie的會話令牌是否使用了'SameSite'屬性，以限制跨站點請求偽造攻擊（CSRF）的風險。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 16 | 7.1.1 |
| **3.4.4** | 驗證基於cookie的會話令牌是否使用'\_\_Host-'字首，這樣cookie只會被髮送到最初設定cookie的主機。 | ✓ | ✓ | ✓ | 16 | 7.1.1 |
| **3.4.5** | 驗證如果應用程式在一個域名下發布，而其他應用程式設定或使用會話cookie（這可能會洩露會話cookie），則在基於cookie的會話令牌中設定路徑屬性（Path），儘可能使用最精確的路徑。 ([C6](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 16 | 7.1.1 |

## V3.5 基於令牌的會話管理

基於令牌的會話管理包括JWT、OAuth、SAML和API金鑰。其中，API金鑰公認較弱，不應該在新程式碼中使用。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **3.5.1** | 驗證該應用允許使用者撤銷與連結應用建立信任關係的OAuth令牌。 |  | ✓ | ✓ | 290 | 7.1.2 |
| **3.5.2** | 驗證應用程式使用會話令牌，而不是靜態API密碼或金鑰，舊的實現除外。 |  | ✓ | ✓ | 798 |  |
| **3.5.3** | 驗證無狀態會話令牌是否使用數字簽名、加密等對策，來防止篡改、封裝、重放、空密碼和金鑰替換等攻擊。 |  | ✓ | ✓ | 345 |  |

## V3.6 聯合重認證

本節與那些編寫依賴方（RP）或憑證服務提供商（CSP）程式碼的人員有關。如果依賴於實現這些功能的程式碼，請確保正確處理這些問題。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **3.6.1** | 驗證依賴方（RP）是否指定了憑證服務提供商（CSP）的最長身份驗證時間，並且如果使用者在該期間內未使用會話，CSP是否會重新驗證使用者。 |  |  | ✓ | 613 | 7.2.1 |
| **3.6.2** | 驗證憑證服務提供商（CSP）通知依賴方（RP）最後一次認證事件，以便 RP 確定他們是否需要重新認證使用者。 |  |  | ✓ | 613 | 7.2.1 |

## V3.7 針對會話管理漏洞的防禦措施

已知的一些會話管理漏洞，與會話的使用者體驗（UX）相關。以前，根據 ISO 27002 要求，ASVS 需要阻止多個併發會話。 現在，阻止併發會話已不再合適，不僅因為現代使用者有許多裝置，或者應用程式是沒有瀏覽器會話的 API，還因為在大多數這些實現中，最後一個身份驗證者獲勝，這通常是攻擊者。本小節提供了使用程式碼阻止、延遲和檢測會話管理攻擊的主要指導。

### 半開放攻擊的描述

在2018年初，一些金融機構遭到了攻擊者所謂的“半開放攻擊”（half-open attacks）。這個術語在行業中一直存在。攻擊者以不同的專有程式碼庫攻擊了多家機構，實際上，在同一個機構中似乎也有不同的程式碼庫。這種“半開放攻擊”利用了許多現有認證、會話管理和訪問控制系統中常見的設計模式缺陷。

攻擊者通過試圖鎖定、重置或恢復一個憑證來開始半開放攻擊。流行的會話管理設計模式，在未認證、半認證（密碼重置、忘記使用者名稱）和完全認證的程式碼之間，重用使用者配置檔案會話物件/模型。 這種設計模式會填充一個有效的會話物件或令牌，其中包含受害者的個人資料，包括密碼雜湊值和角色。如果訪問控制檢查在控制器或路由器沒有正確驗證使用者是完全登入，攻擊者將能夠冒充使用者。攻擊手段可能包括：將使用者的密碼更改為已知值、更新電子郵件地址以執行有效的密碼重置、禁用多因素身份驗證或註冊新的MFA裝置、暴露或更改API金鑰等。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE | [NIST §](https://pages.nist.gov/800-63-3/sp800-63b.html) |
| **3.7.1** | 在允許任何敏感交易或帳戶修改之前，驗證應用程式確保完整、有效的登入會話，或要求重新驗證（二次驗證）。 | ✓ | ✓ | ✓ | 306 |  |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Testing Guide 4.0: Session Management Testing](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/06-Session_Management_Testing/README.html)
* [OWASP Session Management Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html)
* [Set-Cookie \_\_Host- prefix details](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie#Directives)

# V4 訪問控制

## 控制目標

授權是一個概念，即只允許那些被允許使用資源的人訪問資源。確保經過驗證的應用程式滿足以下高階要求：

* 訪問資源的人員持有有效憑據才能這樣做。
* 使用者與一組明確定義的角色和許可權相關聯。
* 角色和許可權元資料受到保護，不會被重放或篡改。

## 安全驗證要求

## V4.1 通用訪問控制設計

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **4.1.1** | 驗證應用程式是否在受信任的服務層上執行訪問控制規則，尤其是在有客戶端訪問控制並且可能被繞過的情況下。 | ✓ | ✓ | ✓ | 602 |
| **4.1.2** | 驗證訪問控制所使用的所有使用者和資料屬性以及策略資訊，不能被終端使用者操縱，除非得到特別授權。 | ✓ | ✓ | ✓ | 639 |
| **4.1.3** | 驗證是否存在最小許可權原則——使用者應該只能訪問他們擁有特定授權的功能、資料檔案、URL、控制器、服務和其他資源。這意味著防止欺騙或特權提升。 ([C7](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 285 |
| **4.1.4** | [已刪除，與 4.1.3 重複] |  |  |  |  |
| **4.1.5** | 驗證訪問控制安全，在發生異常時是否失效。 ([C10](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 285 |

## V4.2 操作級訪問控制

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **4.2.1** | 驗證敏感資料和API的保護，防止針對建立、讀取、更新和刪除記錄的不安全直接物件引用（IDOR）攻擊，如建立或更新別人的記錄，檢視每個人的記錄或刪除所有記錄。 | ✓ | ✓ | ✓ | 639 |
| **4.2.2** | 驗證應用程式或框架是否實施了強大的反 CSRF 機制來保護經過身份驗證的功能，以及有效的反自動化或反 CSRF 保護無需身份驗證的功能。 | ✓ | ✓ | ✓ | 352 |

## V4.3 其他訪問控制注意事項

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **4.3.1** | 驗證管理介面使用適當的多因素認證，防止未經授權的使用。 | ✓ | ✓ | ✓ | 419 |
| **4.3.2** | 驗證目錄瀏覽被禁用，除非特意需要。此外，應用程式不應允許披露檔案或目錄元資料，例如Thumbs.db、.DS\_Store、.git或.svn資料夾。 | ✓ | ✓ | ✓ | 548 |
| **4.3.3** | 驗證應用程式對低價值的系統有額外的授權（如升級或自適應認證），對高價值的應用程式進行職責分離，以根據應用程式和過去的欺詐風險執行反欺詐控制。 |  | ✓ | ✓ | 732 |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Testing Guide 4.0: Authorization](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/05-Authorization_Testing/README.html)
* [OWASP Cheat Sheet: Access Control](https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html)
* [OWASP CSRF Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)
* [OWASP REST Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html)

# V5 驗證、過濾和編碼

## 控制目標

最常見的 Web 應用程式安全漏洞，是在沒有任何輸出編碼的情況下，直接使用來自客戶端或環境的輸入（缺乏正確的驗證）。 這一弱點導致了 Web 應用程式中幾乎所有的重大漏洞，例如跨站點指令碼（XSS）、SQL 注入、直譯器注入、語言環境/Unicode 攻擊、檔案系統攻擊和緩衝區溢位。

確保經過驗證的應用程式滿足以下高階要求：

* 輸入驗證和輸出編碼架構有一條約定的管道來防止注入攻擊。
* 輸入資料是強型別的，經過驗證，範圍或長度檢查，或者在最壞的情況下，經過消毒或過濾。
* 輸出結果根據資料的上下文進行編碼或轉義，儘可能地接近直譯器。

隨著現代網路應用架構的發展，輸出編碼比以往任何時候都更重要。在某些情況下很難提供健壯的輸入驗證，因此使用更安全的API，如引數化查詢、自動轉義的模板框架或精心選擇的輸出編碼，對應用程式的安全性至關重要。

## V5.1 輸入驗證

正確實施的輸入驗證控制，使用白名單列表和強資料型別，可以消除 90% 以上的所有注入攻擊。長度和範圍檢查可以進一步減少這種情況。在應用程式架構、設計衝刺（Design Sprint）、編碼以及單元和整合測試期間，需要構建安全輸入驗證。 儘管其中許多專案在滲透測試中找不到，但不實施它們的結果通常可以在 V5.3 - 輸出編碼和注入預防要求 中找到。建議開發人員和安全程式碼審查人員將本小節作為基礎（正如所有專案都需要滿足 L1 那樣），以防止注入。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **5.1.1** | 驗證應用程式是否有HTTP引數汙染攻擊的防禦措施，特別是當應用程式框架沒有區分請求引數的來源（GET、POST、cookies、請求頭或環境變數）。 | ✓ | ✓ | ✓ | 235 |
| **5.1.2** | 驗證框架是否能防止批量引數分配攻擊，或者應用程式是否有對策來防止不安全的引數分配，如將欄位標記為私有等型別。 ([C5](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 915 |
| **5.1.3** | 驗證所有輸入（HTML 表單欄位、REST 請求、URL 引數、HTTP 請求頭、cookies、批處理檔案、RSS 源等）都使用“白名單”（允許列表）。 ([C5](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 20 |
| **5.1.4** | 驗證結構化資料是強型別的，並根據定義的模式進行驗證，包括允許的字元、長度和模式（如信用卡號碼、電子郵件地址、電話號碼，或驗證兩個相關欄位是否合理，如檢查郊區和郵政編碼是否匹配）。 ([C5](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 20 |
| **5.1.5** | 驗證URL重定向和轉發的目標地址都在白名單中，或者在重定向到可能不受信任的內容時顯示警告。 | ✓ | ✓ | ✓ | 601 |

## V5.2 過濾和沙盒化

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **5.2.1** | 驗證所有來自“所見即所得”編輯器或類似的不受信任的HTML輸入，都已經通過HTML過濾庫或框架功能，進行了適當的淨化。 ([C5](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 116 |
| **5.2.2** | 驗證非結構化資料是否經過消毒處理，以執行安全措施，如允許的字符集和長度。 | ✓ | ✓ | ✓ | 138 |
| **5.2.3** | 驗證應用程式在傳遞給郵件系統之前，對使用者的輸入進行過濾，以防止SMTP或IMAP注入。 | ✓ | ✓ | ✓ | 147 |
| **5.2.4** | 驗證應用程式是否避免使用eval()或其他動態程式碼執行功能。在沒有其他選擇的情況下，任何被包含的使用者輸入必須在執行前進行過濾或沙箱處理。 | ✓ | ✓ | ✓ | 95 |
| **5.2.5** | 驗證應用程式是否對相關的使用者輸入進行過濾或沙箱處理，來防止模板注入攻擊。 | ✓ | ✓ | ✓ | 94 |
| **5.2.6** | 驗證應用程式是否通過驗證或淨化不受信任的資料或HTTP檔案元資料（如檔名和URL輸入欄位），並使用協議、域、路徑和埠的白名單，來防止SSRF攻擊。 | ✓ | ✓ | ✓ | 918 |
| **5.2.7** | 驗證應用程式是否過濾、禁用或沙盒處理了使用者提供的可擴充套件向量圖（SVG）指令碼內容，特別是與內聯指令碼產生的XSS有關的內容，以及外部物件。 | ✓ | ✓ | ✓ | 159 |
| **5.2.8** | 驗證應用程式是否對使用者提供的模板語言內容（指令碼或表示式，如Markdown、CSS或XSL樣式表、BBCode或類似內容）進行過濾、禁用或沙盒處理。 | ✓ | ✓ | ✓ | 94 |

## V5.3 輸出編碼和預防注入

靠近或鄰近當前直譯器的輸出編碼，對應用程式的安全至關重要。通常情況下，輸出編碼並不持久化，而是用於在適當的輸出環境中使輸出安全，以便立即使用。不進行輸出編碼，將最終形成一個不安全、可注入的應用程式。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **5.3.1** | 驗證輸出編碼是否與所需的直譯器和環境相關。例如，根據HTML值、HTML屬性、JavaScript、URL引數、HTTP頭、SMTP等上下文的要求，使用專門的編碼器，特別是來自不可信任的輸入（如帶有Unicode或單引號的名字，如ねこ或O'Hara）。 ([C4](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 116 |
| **5.3.2** | 驗證輸出編碼是否保留了使用者選擇的字符集和地域，從而使任何Unicode字元點都能得到有效和安全的處理。 ([C4](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 176 |
| **5.3.3** | 驗證上下文感知，最好是自動——或者最差也是手動——轉義輸出，以防止反射、儲存或基於DOM的XSS。 ([C4](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 79 |
| **5.3.4** | 驗證資料選擇或資料庫查詢（如 SQL、HQL、ORM、NoSQL）是否使用引數化查詢、ORM、實體框架，或以其他方式防止資料庫注入攻擊。 ([C3](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 89 |
| **5.3.5** | 驗證在沒有引數化或更安全機制的情況下，使用特定上下文的輸出編碼來防止注入攻擊，例如使用SQL轉義來防止SQL注入。 ([C3, C4](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 89 |
| **5.3.6** | 驗證應用程式是否可以防止JSON注入攻擊、JSON eval攻擊和JavaScript表示式評估。 ([C4](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 830 |
| **5.3.7** | 驗證應用程式可以防止LDAP注入漏洞，或者已經實施了特定的安全控制來防止LDAP注入。 ([C4](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 90 |
| **5.3.8** | 驗證應用程式是否能防止作業系統命令注入，以及作業系統呼叫是否使用引數化的作業系統查詢或使用上下文命令列輸出編碼。 ([C4](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 78 |
| **5.3.9** | 驗證應用程式是否能防止本地檔案包含（LFI）或遠端檔案包含（RFI）攻擊。 | ✓ | ✓ | ✓ | 829 |
| **5.3.10** | 驗證應用程式是否能防止XPath注入或XML注入攻擊。 ([C4](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 643 |

注意：使用引數化查詢或轉義SQL並不總是足夠的；表和列名、ORDER BY等不能被轉義。若在這些欄位中轉義使用者提供的資料，會導致查詢失敗或SQL注入。

注意：SVG格式在幾乎所有情況下都明確允許ECMA指令碼，所以可能無法完全阻止所有的SVG XSS向量。如果需要上傳SVG，我們強烈建議將這些上傳的檔案作為text/plain提供，或者使用一個單獨的“使用者提供內容”域，以防止成功的XSS接管應用程式。

## V5.4 記憶體、字串和非託管程式碼

以下要求僅在應用程式使用系統語言或非託管程式碼時適用。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **5.4.1** | 驗證應用程式是否使用記憶體安全字串、更安全的記憶體複製和指標運算，以檢測或防止堆疊、緩衝區或堆溢位。 |  | ✓ | ✓ | 120 |
| **5.4.2** | 驗證格式化字串不接受潛在的有害輸入，並且是常量。 |  | ✓ | ✓ | 134 |
| **5.4.3** | 驗證運用了符號、範圍和輸入驗證技術來防止整數溢位。 |  | ✓ | ✓ | 190 |

## V5.5 預防反序列化

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **5.5.1** | 驗證序列化物件是否使用完整性檢查或加密，以防止惡意物件的建立或資料篡改。 ([C5](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 502 |
| **5.5.2** | 驗證應用程式正確限制 XML 解析器，使其只使用最嚴格的配置，並確保禁用不安全的功能，如解析外部實體，以防止 XML 外部實體注入（XXE）攻擊。 | ✓ | ✓ | ✓ | 611 |
| **5.5.3** | 驗證自定義程式碼和第三方庫（如JSON、XML和YAML解析器）禁止或限制不受信任資料的反序列化。 | ✓ | ✓ | ✓ | 502 |
| **5.5.4** | 驗證在瀏覽器或基於 JavaScript 的後端解析 JSON 時，使用 JSON.parse 來解析 JSON 文件。不使用 eval() 來解析 JSON。 | ✓ | ✓ | ✓ | 95 |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Testing Guide 4.0: Input Validation Testing](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/07-Input_Validation_Testing/README.html)
* [OWASP Cheat Sheet: Input Validation](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)
* [OWASP Testing Guide 4.0: Testing for HTTP Parameter Pollution](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/07-Input_Validation_Testing/04-Testing_for_HTTP_Parameter_Pollution.html)
* [OWASP LDAP Injection Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.html)
* [OWASP Testing Guide 4.0: Client Side Testing](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client_Side_Testing/)
* [OWASP Cross Site Scripting Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)
* [OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html)
* [OWASP Java Encoding Project](https://owasp.org/owasp-java-encoder/)
* [OWASP Mass Assignment Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Mass_Assignment_Cheat_Sheet.html)
* [DOMPurify - Client-side HTML Sanitization Library](https://github.com/cure53/DOMPurify)
* [XML External Entity (XXE) Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html)

有關自動轉義的更多資訊，請參閱：

* [Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems](https://googleonlinesecurity.blogspot.com/2009/03/reducing-xss-by-way-of-automatic.html)
* [AngularJS Strict Contextual Escaping](https://docs.angularjs.org/api/ng/service/$sce)
* [AngularJS ngBind](https://docs.angularjs.org/api/ng/directive/ngBind)
* [Angular Sanitization](https://angular.io/guide/security#sanitization-and-security-contexts)
* [Angular Security](https://angular.io/guide/security)
* [ReactJS Escaping](https://reactjs.org/docs/introducing-jsx.html#jsx-prevents-injection-attacks)
* [Improperly Controlled Modification of Dynamically-Determined Object Attributes](https://cwe.mitre.org/data/definitions/915.html)

有關反序列化的更多資訊，請參閱：

* [OWASP Deserialization Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html)
* [OWASP Deserialization of Untrusted Data Guide](https://owasp.org/www-community/vulnerabilities/Deserialization_of_untrusted_data)

# V6 儲存密碼學

## 控制目標

確保經過驗證的應用程式滿足以下高階要求：

* 所有加密模組以安全的方式失效，並且正確處理錯誤。
* 使用合適的隨機數發生器。
* 金鑰訪問被安全地管理。

## V6.1 資料分類

最重要的資產是由應用程式處理、儲存或傳輸的資料。始終執行隱私影響評估，對任何儲存資料的資料保護需求進行正確分類。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **6.1.1** | 驗證受監管的私人資料在靜止狀態下是否被加密儲存，如個人身份資訊（PII）、敏感個人資訊或經評估可能受制於歐盟GDPR的資料。 |  | ✓ | ✓ | 311 |
| **6.1.2** | 驗證受監管的健康資料在靜止狀態下是否被加密儲存，如醫療記錄、醫療裝置詳情或去匿名化的研究記錄。 |  | ✓ | ✓ | 311 |
| **6.1.3** | 驗證受監管的金融資料在靜止狀態下是否被加密儲存，如金融賬戶、違約或信用記錄、稅務記錄、工資記錄、受益人或去匿名化的市場或研究記錄。 |  | ✓ | ✓ | 311 |

## V6.2 演算法

密碼學的最新進展意味著以前安全的演算法和金鑰長度不再安全或足以保護資料。因此，應該可以改變演算法。

雖然這一部分不容易進行滲透測試，但開發人員應該把這一整節視為強制性的，即使在大多數專案中 L1 都沒有要求。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **6.2.1** | 驗證所有的加密模組即使在故障時也是安全的，並且處理錯誤的方式不會使Padding Oracle攻擊得逞。 | ✓ | ✓ | ✓ | 310 |
| **6.2.2** | 驗證使用業界認可或政府批准的加密演算法、模式和庫，而不是自定義編碼的加密技術。 ([C8](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 327 |
| **6.2.3** | 驗證加密初始化向量、密碼配置和分組模式是否使用最新建議進行安全配置。 |  | ✓ | ✓ | 326 |
| **6.2.4** | 驗證隨機數、加密或雜湊演算法、金鑰長度、輪次、密碼或模式，可以在任何時候重新配置、升級或交換，以防止密碼中斷。 ([C8](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 326 |
| **6.2.5** | 驗證不使用已知不安全的分組模式（如ECB等）、填充模式（如PKCS#1 v1.5等）、小塊大小的密碼（如Triple-DES、Blowfish等）和弱雜湊演算法（如MD5、SHA1等），除非需要向後相容。 |  | ✓ | ✓ | 326 |
| **6.2.6** | 驗證隨機數、初始化向量和其他一次性使用的數字，不得與特定的加密金鑰使用超過一次。生成方法必須適合所使用的演算法。 |  | ✓ | ✓ | 326 |
| **6.2.7** | 驗證加密資料是否通過簽名、認證的密碼模式或 HMAC 進行身份驗證，以確保密文不會被未經授權的一方更改。 |  |  | ✓ | 326 |
| **6.2.8** | 驗證所有的密碼操作都是恆定時間的，在比較、計算或返回中沒有“短路”操作，以避免資訊洩漏。 |  |  | ✓ | 385 |

## V6.3 隨機值

真正的偽隨機數生成（PRNG）很難實現。通常，如果過度使用，系統內良好的熵源不但很快耗盡，而且隨機性較小的源會導致可預測的金鑰和祕密。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **6.3.1** | 驗證所有的隨機數、隨機檔名、隨機GUID和隨機字串，都是使用加密模組認可的加密安全隨機數生成器生成的，而這些隨機值旨在不被攻擊者猜測。 |  | ✓ | ✓ | 338 |
| **6.3.2** | 驗證是否使用 GUID v4 演算法和加密安全偽隨機數生成器（CSPRNG）建立了隨機 GUID。使用其他偽隨機數生成器建立的 GUID 可能是可預測的。 |  | ✓ | ✓ | 338 |
| **6.3.3** | 驗證應用程式即使在處於高負載下時也使用適當的熵建立隨機數，或者應用程式在這種情況下優雅地降級。 |  |  | ✓ | 338 |

## V6.4 金鑰管理

雖然這一部分不容易進行滲透測試，但開發人員應將整個部分視為強制性的，即使大多數專案中都缺少 L1 的要求。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **6.4.1** | 驗證祕密管理解決方案，如鑰匙庫，用於安全地建立、儲存、控制對祕密的訪問和銷燬。 ([C8](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 798 |
| **6.4.2** | 驗證金鑰材料是否未暴露給應用程式，而是使用一個隔離的安全模組（如保險庫）進行加密操作。 ([C8](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 320 |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Testing Guide 4.0: Testing for weak Cryptography](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/README.html)
* [OWASP Cheat Sheet: Cryptographic Storage](https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html)
* [FIPS 140-2](https://csrc.nist.gov/publications/detail/fips/140/2/final)

# V7 錯誤處理和日誌記錄

## 控制目標

錯誤處理和記錄的主要目標，是為使用者、管理員和事件響應團隊提供有用的資訊。目標不是建立大量日誌，而是建立高質量的日誌，資訊多於丟棄的噪聲。

高質量的日誌往往包含敏感資料，必須按照當地的資料隱私法或指令進行保護。這應該包括：

* 除非特別要求，否則不收集或記錄敏感資訊。
* 確保所有記錄的資訊被安全地處理，並根據其資料分類進行保護。
* 確保日誌不會被永久儲存，而是有一個儘可能短的生命週期。

如果日誌包含私人或敏感資料（各國對這些資料的定義各不相同），則日誌將成為應用程式所持有的最敏感資訊之一，因此對攻擊者本身來說非常有吸引力。

同樣重要的是，要確保應用程式安全地故障，而且錯誤不會洩露不必要的資訊。

## V7.1 日誌內容

記錄敏感資訊是很危險的——日誌本身就成了機密，這意味著它們需要被加密，成為保留政策的物件，並且必須在安全審計中被披露。確保只有必要的資訊被儲存在日誌中，當然沒有支付、憑證（包括會話令牌）、敏感或個人身份資訊。

V7.1 涵蓋了 OWASP Top 10 2017:A10. 由於 2017:A10 和本節不可通過滲透測試來驗證，重要的是：

* 開發人員要確保完全遵守本節，就像所有專案都被標記為L1一樣。
* 滲透測試人員通過訪談，螢幕截圖或斷言來驗證 V7.1 中所有專案的完全合規性。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **7.1.1** | 驗證應用程式不記錄憑證或支付細節。會話令牌應該只以不可逆的雜湊形式儲存在日誌中。 ([C9, C10](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 532 |
| **7.1.2** | 驗證應用程式不會記錄當地隱私法或相關安全政策規定的其他敏感資料。 ([C9](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 532 |
| **7.1.3** | 驗證應用程式是否記錄安全相關事件，例如成功和失敗的認證事件、訪問控制失敗、反序列化失敗和輸入驗證失敗的事件。 ([C5, C7](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 778 |
| **7.1.4** | 驗證每個日誌事件都包含必要的資訊，以便在事件發生後詳細調查時間軸。 ([C9](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 778 |

## V7.2 日誌處理

及時的日誌記錄對於事件的審計、研判和升級是至關重要的。確保應用程式的日誌是清晰的，可以很容易地在本地監測和分析，或將日誌傳送到遠端監控系統。

V7.2 涵蓋了 OWASP Top 10 2017:A10. 由於 2017:A10 和本節不可通過滲透測試來驗證，重要的是：

* 開發人員要確保完全遵守本節，就像所有專案都被標記為L1一樣。
* 滲透測試人員通過訪談，螢幕截圖或斷言來驗證 V7.2 中所有專案的完全合規性。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **7.2.1** | 驗證所有的認證決策都被記錄下來，不儲存敏感的會話令牌或密碼。這應該包括安全調查所需的具有相關元資料的請求。 |  | ✓ | ✓ | 778 |
| **7.2.2** | 驗證是否可以記錄所有訪問控制決策並記錄所有失敗的決策。這應包括安全調查所需的具有相關元資料的請求。 |  | ✓ | ✓ | 285 |

## V7.3 日誌保護

可以輕易修改或刪除的日誌，對於調查和起訴毫無用處。日誌的披露可能會暴露有關應用程式或其包含的資料的內部細節。在保護日誌免遭未經授權的披露、修改或刪除時，必須小心謹慎。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **7.3.1** | 驗證所有日誌元件是否對資料進行了適當的編碼，以防止日誌注入。 ([C9](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 117 |
| **7.3.2** | [已刪除，與 7.3.1 重複] |  |  |  |  |
| **7.3.3** | 驗證安全日誌是否受到保護，防止未授權的訪問或修改。 ([C9](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 200 |
| **7.3.4** | 驗證時間源是否同步到正確的時間和時區。如果系統是全球性的，強烈考慮只用UTC來記錄，以協助事件後的取證分析。 ([C9](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ |  |

注意：日誌編碼（7.3.1）很難使用自動化工具和滲透測試進行測試和審查，但架構師、開發人員和原始碼審查人員應將其視為 L1 要求。

## V7.4 錯誤處理

錯誤處理的目的，是允許應用程式提供與安全有關的事件，用於監控、研判和升級。目的不是為了建立日誌。當記錄安全相關的事件時，要確保日誌有其目的，並且可以被SIEM或分析軟體識別。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **7.4.1** | 驗證在發生意外或安全敏感錯誤時，是否顯示通用資訊，可能帶有支援人員可以用於調查的唯一ID。 ([C10](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 210 |
| **7.4.2** | 驗證整個程式碼庫是否使用了異常處理（或類似功能），以說明預期和非預期的錯誤情況。 ([C10](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 544 |
| **7.4.3** | 驗證是否定義了“最後手段”的錯誤處理程式，以捕獲所有未處理的異常。 ([C10](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 431 |

注意：某些語言，例如 Swift 和 Go ——以及通過常見的設計實踐——許多函式式語言，不支援異常或最後的事件處理程式。在這種情況下，架構師和開發人員應該使用一種模式、語言或框架友好的方式，來確保應用程式能夠安全地處理異常、意外或安全相關的事件。

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Testing Guide 4.0 content: Testing for Error Handling](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/08-Testing_for_Error_Handling/README.html)
* [OWASP Authentication Cheat Sheet section about error messages](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html#authentication-and-error-messages)

# V8 資料保護

## Control Objective

健全的資料保護有三個關鍵因素。機密性、完整性和可用性（CIA）。這個標準假定資料保護是在一個可信的系統上執行的，比如伺服器，它已經被加固並有足夠的保護措施。

應用程式必須假設所有的使用者裝置都以某種方式受到損害。如果應用程式在不安全的裝置上傳輸或儲存敏感資訊，如共享電腦、手機和平板電腦，應用程式有責任確保儲存在這些裝置上的資料是加密的，不能輕易地被非法獲取、改變或披露。

確保經過驗證的應用程式滿足以下高水平的資料保護要求：

* 機密性。資料應受到保護，在傳輸過程中和儲存時都不會被未經授權的觀察或披露。
* 完整性。應保護資料不被未經授權的攻擊者惡意建立、更改或刪除。
* 可用性。資料應按要求提供給授權的使用者。

## V8.1 通用資料保護

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **8.1.1** | 驗證應用程式保護敏感資料不被快取在負載均衡和應用程式快取等伺服器元件中。 |  | ✓ | ✓ | 524 |
| **8.1.2** | 驗證在伺服器上所儲存敏感資料的所有快取或臨時副本是否受到保護（防止未經授權的訪問），或在被授權使用者訪問後清除/失效。 |  | ✓ | ✓ | 524 |
| **8.1.3** | 驗證應用程式儘量減少請求中的引數數量，如隱藏欄位、Ajax 變數、cookies 和請求頭。 |  | ✓ | ✓ | 233 |
| **8.1.4** | 驗證應用程式能夠檢測並提醒異常的請求數量，例如按IP、使用者、每小時或每天的總數，或其它對應用程式有意義的指標。 |  | ✓ | ✓ | 770 |
| **8.1.5** | 驗證是否對重要資料進行了定期備份，是否對資料進行了測試恢復。 |  |  | ✓ | 19 |
| **8.1.6** | 驗證備份的安全儲存，防止資料被盜或損壞。 |  |  | ✓ | 19 |

## V8.2 客戶端資料保護

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **8.2.1** | 驗證應用程式設定足夠的“禁止快取”頭，以便敏感資料不會在現代瀏覽器中被快取。 | ✓ | ✓ | ✓ | 525 |
| **8.2.2** | 驗證儲存在瀏覽器儲存（例如 localStorage、sessionStorage、IndexedDB 或 cookie）中的資料不包含敏感資料。 | ✓ | ✓ | ✓ | 922 |
| **8.2.3** | 在客戶端或會話終止後，驗證經過身份驗證的資料已從客戶端儲存（例如瀏覽器 DOM）中清除。 | ✓ | ✓ | ✓ | 922 |

## V8.3 敏感私有資料

本小節有助於保護敏感資料免遭未經授權的建立、讀取、更新或刪除，尤其是批量資料。

遵守本節意味著遵守 V4 訪問控制，尤其是 V4.2。例如，為了防止個人敏感資訊的未授權的更新或洩露，需要遵守 V4.2.1。請遵守本節和 V4 以獲得全面覆蓋。

注意：隱私法規和法律，例如澳大利亞隱私原則 APP-11 或 GDPR，直接影響應用程式必須如何處理敏感個人資訊的儲存、使用和傳輸的實施。範圍從嚴厲的處罰到簡單的建議。請查閱您當地的法律法規，並根據需要諮詢專業的隱私專家或律師。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **8.3.1** | 驗證敏感資料是在HTTP訊息正文或請求頭中被髮送到伺服器，以及HTTP請求方法的查詢字串引數都不包含敏感資料。 | ✓ | ✓ | ✓ | 319 |
| **8.3.2** | 驗證使用者是否有途徑按需刪除或匯出自己的資料。 | ✓ | ✓ | ✓ | 212 |
| **8.3.3** | 驗證向用戶提供了關於收集和使用其個人資訊的明確語言，並且在以任何方式使用這些資料之前，使用者已勾選了同意。 | ✓ | ✓ | ✓ | 285 |
| **8.3.4** | 驗證應用程式建立和處理的所有敏感資料是否已被識別，並確保已制定瞭如何處理敏感資料的策略。 ([C8](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 200 |
| **8.3.5** | 如果資料是根據相關資料保護指令收集的或（應用）要求記錄訪問日誌，驗證訪問敏感資料是否被審計（不記錄敏感資料本身）。 |  | ✓ | ✓ | 532 |
| **8.3.6** | 為了減少記憶體轉儲攻擊，一旦不再需要記憶體中的敏感資訊，請檢查該敏感資訊是否會被覆蓋（使用0或隨機數）。 |  | ✓ | ✓ | 226 |
| **8.3.7** | 驗證需要加密的敏感資訊或私有資訊是否使用經過批准的機密性和完整性演算法加密。 ([C8](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 327 |
| **8.3.8** | 驗證敏感的個人資訊是否符合資料保留分類，以便自動、按計劃或根據情況需要刪除舊資料或過時資料。 |  | ✓ | ✓ | 285 |

在考慮資料保護時，首要的考慮應該圍繞批量提取、修改或過度使用。例如，許多社交媒體系統只允許使用者每天新增100個新好友，但這些請求來自哪個系統並不重要。銀行平臺可能希望阻止每小時超過5筆的、轉移超過1000歐元的外部交易。每個系統的要求可能非常不同，所以決定 “異常” 必須考慮威脅模型和商業風險。重要的標準是檢測、遏制，或者最好是阻止這種異常批量行為的能力。

## 參考文獻

有關更多資訊，請參閱：

* [Consider using Security Headers website to check security and anti-caching headers](https://securityheaders.io)
* [OWASP Secure Headers project](https://owasp.org/www-project-secure-headers/)
* [OWASP Privacy Risks Project](https://owasp.org/www-project-top-10-privacy-risks/)
* [OWASP User Privacy Protection Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html)
* [European Union General Data Protection Regulation (GDPR) overview](https://edps.europa.eu/data-protection_en)
* [European Union Data Protection Supervisor - Internet Privacy Engineering Network](https://edps.europa.eu/data-protection/ipen-internet-privacy-engineering-network_en)

# V9 通訊

## 控制目標

確保經過驗證的應用程式滿足以下高階要求：

* 要求 TLS 或強加密，與內容的敏感性無關。
* 遵循最新指南，包括：
  + 配置建議
  + 首選演算法和密碼
* 避免使用弱的或即將被廢棄的演算法和密碼，除非是最後的手段。
* 禁用已廢棄或已知不安全的演算法和密碼。

在這些要求範圍內：

* 瞭解業界對安全TLS配置的建議，因為它經常變化（往往是由於現有演算法和密碼的災難性破壞）。
* 使用最新版本的TLS配置審查工具，來配置首選順序和演算法選擇。
* 定期檢查你的配置，以確保安全通訊始終存在並有效。

## V9.1 客戶端通訊安全

確保所有客戶端訊息都通過加密網路傳送，使用TLS 1.2或更高版本。 使用最新的工具定期檢查客戶端配置。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **9.1.1** | 驗證所有客戶端連線都使用了TLS，並且不會降級到不安全或未加密的通訊。 ([C8](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 319 |
| **9.1.2** | 使用最新的TLS測試工具，驗證是否只啟用了強密碼套件，並將最強的密碼套件設定為首選。 | ✓ | ✓ | ✓ | 326 |
| **9.1.3** | 驗證只啟用最新推薦版本的TLS協議，如TLS 1.2和TLS 1.3。最新版本的TLS協議應該是首選項。 | ✓ | ✓ | ✓ | 326 |

## V9.2 伺服器通訊安全

伺服器通訊不僅僅是 HTTP。與其他系統的安全連線，例如監控系統、管理工具、遠端訪問和 ssh、中介軟體、資料庫、大型機、合作伙伴或外部源系統——必須到位。所有這些都必須加密，以防止“外面安全，裡面被輕易截獲”。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **9.2.1** | 驗證與伺服器的連線是否使用受信任的TLS證書。在使用內部生成或自簽名證書的情況下，必須將伺服器配置為只信任特定的內部CA和特定的自簽證書。所有其他的都應該被拒絕。 |  | ✓ | ✓ | 295 |
| **9.2.2** | 確認所有入站和出站連線都使用了 TLS 等加密通訊，包括管理埠、監控、身份驗證、API 或 Web 服務呼叫、資料庫、雲、serverless、大型機、外部和合作夥伴的連線。伺服器不得回退到不安全或未加密的協議。 |  | ✓ | ✓ | 319 |
| **9.2.3** | 驗證所有外部系統中與敏感資訊/功能相關的加密連線，均已通過身份驗證。 |  | ✓ | ✓ | 287 |
| **9.2.4** | 驗證是否啟用並配置了正確的證書吊銷，例如線上證書狀態協議（OCSP）Stapling。 |  | ✓ | ✓ | 299 |
| **9.2.5** | 驗證是否記錄了後端TLS連線失敗（的事件）。 |  |  | ✓ | 544 |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP – TLS Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html)
* [OWASP - Pinning Guide](https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning)
* 關於 “TLS的批准模式“ 的說明:
  + 在過去，ASVS提到了美國標準FIPS 140-2，但作為一個全球標準美國標準的應用可能充滿困難、矛盾或混亂。
  + 實現第9.1節的更好方法是審查指南，如 [Mozilla's Server Side TLS](https://wiki.mozilla.org/Security/Server_Side_TLS) or [generate known good configurations](https://mozilla.github.io/server-side-tls/ssl-config-generator/)，並使用已知最新的TLS評估工具來獲得所需的安全等級。

# V10 惡意程式碼

## 控制目標

確保程式碼滿足以下高階要求：

* 惡意活動得到安全和適當的處理，不會影響應用程式的其餘部分。
* 沒有定時炸彈或其他基於時間的攻擊。
* 不會向惡意或未經授權的目的地“打電話回家”。
* 沒有後門、復活節彩蛋、Salami攻擊、rootkit 或攻擊者可以控制的未授權程式碼。

發現惡意程式碼是否定的證明，這是不可能被充分驗證的。應盡最大努力，確保程式碼沒有固有的惡意程式碼或不需要的功能。

## V10.1 程式碼完整性

對惡意程式碼的最佳防禦是“信任，但要驗證”。在許多司法管轄區，將未經授權或惡意的程式碼片段引入程式碼，通常是刑事犯罪。策略和過程應明確對惡意程式碼的制裁。

首席開發人員應該定期檢查程式碼簽入，特別是那些可能去訪問時間、I/O或網路功能的程式碼簽入。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **10.1.1** | 驗證是否使用了程式碼分析工具，可以檢測潛在的惡意程式碼，如時間函式、不安全的檔案操作和網路連線。 |  |  | ✓ | 749 |

## V10.2 惡意程式碼搜尋

惡意程式碼極為罕見，難以檢測。手動逐行審查程式碼可以幫助尋找邏輯炸彈，但即使是最有經驗的程式碼審查員也很難找到惡意程式碼，哪怕事先知道它們的存在。

如果不能完全接觸到原始碼，包括第三方庫，就不可能遵守本節的規定。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **10.2.1** | 驗證應用程式的原始碼和第三方庫不包含未經授權的回連或資料收集功能。如果存在這樣的功能，在收集任何資料之前，要獲得使用者的操作許可。 |  | ✓ | ✓ | 359 |
| **10.2.2** | 驗證應用程式不會對隱私相關的功能或感測器（例如聯絡人、攝像頭、麥克風或位置）要求不必要或過度的許可權。 |  | ✓ | ✓ | 272 |
| **10.2.3** | 驗證應用程式的原始碼和第三方庫不包含後門，如硬編碼或額外的未記錄的賬戶或金鑰、程式碼混淆、未記錄的二進位制blobs、rootkits或反除錯、不安全的除錯特性，或其他過時、不安全或隱藏的功能，一旦被發現可能會被惡意使用。 |  |  | ✓ | 507 |
| **10.2.4** | 通過搜尋日期和時間相關函式，來驗證應用程式原始碼和第三方庫不包含時間炸彈。 |  |  | ✓ | 511 |
| **10.2.5** | 驗證應用程式原始碼和第三方庫不包含惡意程式碼，例如salami攻擊、邏輯繞過或邏輯炸彈。 |  |  | ✓ | 511 |
| **10.2.6** | 驗證應用程式的原始碼和第三方庫不包含復活節彩蛋或任何其他潛在的冗餘功能。 |  |  | ✓ | 507 |

## V10.3 應用程式完整性

應用程式被部署後，惡意程式碼仍然可以被插入。應用程式需要保護自己免受常見的攻擊，例如執行來自不受信任來源的未簽名程式碼或子域接管。

本節內容的實現，很可能是操作性和持續性的。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **10.3.1** | 驗證如果應用程式具有客戶端或伺服器自動更新功能，則應通過安全通道獲得更新，並進行數字簽名。更新程式碼必須在安裝或執行更新之前驗證更新的數字簽名。 | ✓ | ✓ | ✓ | 16 |
| **10.3.2** | 驗證應用程式是否採用了完整性保護，如程式碼簽名或子資源完整性。應用程式不得從不受信任的來源載入或執行程式碼，例如從不可信任的來源或網際網路載入模組、外掛、程式碼或庫。 | ✓ | ✓ | ✓ | 353 |
| **10.3.3** | 如果應用程式依賴 DNS 條目或 DNS 子域，例如過期的域名、過時的 DNS 指標或 CNAME、公共原始碼庫中過期的專案或臨時的雲API介面、serverless功能或儲存桶（*autogen-bucket-id*.cloud.example.com）或類似情況，則驗證該應用程式是否具有防止子域接管的措施。保護措施可以包括確保定期檢查應用程式使用的DNS名稱是否過期或改變。 | ✓ | ✓ | ✓ | 350 |

## 參考文獻

有關更多資訊，請參閱：

* [Hostile Subdomain Takeover, Detectify Labs](https://labs.detectify.com/2014/10/21/hostile-subdomain-takeover-using-herokugithubdesk-more/)
* [Hijacking of abandoned subdomains part 2, Detectify Labs](https://labs.detectify.com/2014/12/08/hijacking-of-abandoned-subdomains-part-2/)

# V11 業務邏輯

## 控制目標

確保經過驗證的應用程式滿足以下高階要求：

* 業務邏輯流程是序列的，按順序處理的，並且不能被繞過。
* 業務邏輯包括檢測和防止自動化攻擊，如連續的小額資金轉移，或一次新增上百萬個好友等。
* 高價值的業務邏輯流已經考慮了濫用情況和惡意行為者，並有防止欺騙、篡改、資訊披露和特權提升攻擊的保護措施。

## V11.1 業務邏輯安全

業務邏輯安全對每個應用程式來說都是非常獨特的，因此沒有通用的檢查表。業務邏輯安全必須設計成能夠抵禦可能的外部威脅——它不能使用 Web 應用防火牆或安全通訊來新增。我們建議在設計衝刺（Design Sprint）期間使用威脅建模，例如使用 OWASP Cornucopia 或類似工具。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **11.1.1** | 驗證應用程式僅按序列順序處理同一使用者的業務邏輯流，不會跳過步驟。 | ✓ | ✓ | ✓ | 841 |
| **11.1.2** | 驗證應用程式將只處理業務邏輯流，所有步驟都在現實的人工時間內處理，即事務不會提交得太快。 | ✓ | ✓ | ✓ | 799 |
| **11.1.3** | 驗證應用程式是否對特定的業務操作或交易有適當的限制，並在每個使用者的基礎上正確執行。 | ✓ | ✓ | ✓ | 770 |
| **11.1.4** | 驗證應用程式具有反自動化的控制手段，以防止過度呼叫，如大量資料洩露、業務邏輯請求、檔案上傳或拒絕服務攻擊。 | ✓ | ✓ | ✓ | 770 |
| **11.1.5** | 驗證應用程式是否具有業務邏輯限制或驗證，以防止可能的業務風險或威脅（使用威脅建模或類似方法識別）。 | ✓ | ✓ | ✓ | 841 |
| **11.1.6** | 驗證應用程式是否存在TOCTOU（Time Of Check to Time Of Use）問題 或敏感操作的其他條件競爭問題。 |  | ✓ | ✓ | 367 |
| **11.1.7** | 驗證應用程式是否從業務邏輯角度監控異常事件或活動。例如，嘗試執行無序的操作或普通使用者永遠不會嘗試的操作。 ([C9](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 754 |
| **11.1.8** | 驗證應用程式在檢測到自動化攻擊或異常活動時，具有可配置的警報。 |  | ✓ | ✓ | 390 |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Web Security Testing Guide 4.1: Business Logic Testing](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/10-Business_Logic_Testing/README.html)
* 反自動化可以通過多種方式實現，包括使用 [OWASP AppSensor](https://github.com/jtmelton/appsensor) 和 [OWASP Automated Threats to Web Applications](https://owasp.org/www-project-automated-threats-to-web-applications/)
* [OWASP AppSensor](https://github.com/jtmelton/appsensor) 也可以幫助進行攻擊檢測和響應。
* [OWASP Cornucopia](https://owasp.org/www-project-cornucopia/)

# V12 檔案和資源

## 控制目標

確保經過驗證的應用程式滿足以下高階要求：

* 不受信任的檔案資料應以安全的方式進行相應處理。
* 從不可信任的來源獲得的不可信任的檔案資料被儲存在Web目錄之外，並具有有限的許可權。

## V12.1 檔案上傳

儘管zip炸彈很容易使用滲透測試技術進行測試，但它們被認為是L2及以上級別，以鼓勵設計和開發時考慮仔細的人工測試，並避免對拒絕服務場景進行自動化或不熟練的手動滲透測試。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **12.1.1** | 確認應用程式不會接受可能會填滿儲存空間或導致拒絕服務的大檔案。 | ✓ | ✓ | ✓ | 400 |
| **12.1.2** | 驗證應用程式在解壓縮檔案前，根據允許的最大解壓縮尺寸和最大檔案數檢查壓縮檔案（如zip，gz，docx，odt）。 |  | ✓ | ✓ | 409 |
| **12.1.3** | 驗證檔案大小配額和每個使用者的最大檔案數是否被強制執行，以確保單個使用者不能用過多的檔案或過大的檔案填滿儲存。 |  | ✓ | ✓ | 770 |

## V12.2 檔案完整性

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **12.2.1** | 驗證從不可信任的來源獲得的檔案，根據檔案的內容，驗證其是否為預期型別。 |  | ✓ | ✓ | 434 |

## V12.3 檔案執行

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **12.3.1** | 驗證系統或框架檔案系統不直接使用使用者提交的檔名元資料，並且使用 URL API 來防止路徑遍歷。 | ✓ | ✓ | ✓ | 22 |
| **12.3.2** | 驗證使用者提交的檔名元資料是否經過驗證或忽略，以防止通過本地檔案包含（LFI） 洩露、建立、更新或刪除本地檔案。 | ✓ | ✓ | ✓ | 73 |
| **12.3.3** | 驗證使用者提交的檔名元資料是否經過驗證或忽略，以防止通過遠端檔案包含（Remote File Inclusion，RFI）或伺服器端請求偽造攻擊（server - Server Side Request Forgery，SSRF）洩露或執行遠端檔案。 | ✓ | ✓ | ✓ | 98 |
| **12.3.4** | 驗證應用程式通過驗證或忽略使用者提交的JSON、JSONP或URL引數中的檔名來防止反射檔案下載（RFD），響應的Content-Type頭應該設定為 text/plain，而Content-Disposition頭應該有一個固定的檔名。 | ✓ | ✓ | ✓ | 641 |
| **12.3.5** | 驗證未受信任的檔案元資料不直接用於系統API或庫，以防止作業系統命令注入。 | ✓ | ✓ | ✓ | 78 |
| **12.3.6** | 驗證應用程式不包含或執行不可信任來源的功能，如未經驗證的內容分發網路、JavaScript 庫、node npm 庫或伺服器端 DLL。 |  | ✓ | ✓ | 829 |

## V12.4 檔案儲存

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **12.4.1** | 驗證從不受信任的來源獲得的檔案是否儲存在 Web 根目錄之外，並具有有限的許可權。 | ✓ | ✓ | ✓ | 552 |
| **12.4.2** | 驗證從不受信任的來源獲得的檔案是否已被防病毒掃描程式掃描，以防止上傳和提供已知的惡意內容。 | ✓ | ✓ | ✓ | 509 |

## V12.5 檔案下載

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **12.5.1** | 驗證網路層是否被配置為只提供具有特定副檔名的檔案，以防止意外資訊和原始碼洩漏。例如，除非有需要，應阻止提供備份檔案（如.bak）、臨時工作檔案（如.swp）、壓縮檔案（.zip、.tar.gz等）以及其他編輯人員常用的副檔名。 | ✓ | ✓ | ✓ | 552 |
| **12.5.2** | 驗證對上傳檔案的直接請求永遠不會作為 HTML/JavaScript 內容執行。 | ✓ | ✓ | ✓ | 434 |

## V12.6 SSRF保護

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | CWE |
| **12.6.1** | 驗證 Web 或應用程式伺服器是否配置了資源或系統的白名單列表，伺服器可以向其傳送請求或載入資料/檔案。 | ✓ | ✓ | ✓ | 918 |

## 參考文獻

有關更多資訊，請參閱：

* [File Extension Handling for Sensitive Information](https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload)
* [Reflective file download by Oren Hafif](https://www.trustwave.com/Resources/SpiderLabs-Blog/Reflected-File-Download---A-New-Web-Attack-Vector/)
* [OWASP Third Party JavaScript Management Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.html)

# V13 API和Web Service

## 控制目標

確保經驗證使用可信服務層API（通常使用JSON或XML或GraphQL）的應用程式具有:

* 對所有網路服務進行充分的認證、會話管理和授權。
* 對所有從較低信任級別傳入較高信任級別的輸入引數進行驗證。
* 有效地對所有API型別進行安全控制，包括雲和Serverless API。

請將本章與其他章節結合起來閱讀；我們不再重複說明認證或API會話管理問題。

## V13.1 通用Web Service安全

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **13.1.1** | 驗證所有應用程式元件使用相同的編碼和解析器，以避免利用不同的URI或檔案解析特性的攻擊（這些解析特性可能被用於SSRF和RFI攻擊）。 | ✓ | ✓ | ✓ | 116 |
| **13.1.2** | [已刪除，與 4.3.1 重複] |  |  |  |  |
| **13.1.3** | 驗證 API URL不公開敏感資訊，例如 API 金鑰、會話令牌等。 | ✓ | ✓ | ✓ | 598 |
| **13.1.4** | 驗證授權決策是同時在URI（由程式性或宣告性的控制器或路由安全執行）和資源層面（由基於模型的許可權執行）做出的。 |  | ✓ | ✓ | 285 |
| **13.1.5** | 驗證包含意外或缺少內容型別的請求是否通過適當的響應頭拒絕（HTTP 響應狀態 406 Unacceptable 或 415 Unsupported Media Type）。 |  | ✓ | ✓ | 434 |

## V13.2 RESTful Web Service

JSON模式驗證還處於標準化的草案階段（見參考文獻）。當考慮使用JSON模式驗證（這是RESTful web services的最佳實踐）時，可以考慮將這些額外的資料驗證策略與JSON模式驗證結合使用:

* 對JSON物件進行解析驗證，例如是否有缺失或多餘的元素。
* 使用標準的輸入驗證方法對JSON物件的值進行驗證，如資料型別、資料格式、長度等。
* 以及正式的JSON模式驗證。

一旦JSON模式驗證標準被正式確定，ASVS將更新這方面的建議。仔細監測正在使用的任何JSON模式驗證庫，因為它們需要定期更新，直到標準正式化並且從參考實現中消除錯誤。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **13.2.1** | 驗證啟用的RESTful HTTP方法對使用者或操作來說是有效的選擇，例如防止普通使用者在受保護的API或資源上使用DELETE或PUT。 | ✓ | ✓ | ✓ | 650 |
| **13.2.2** | 驗證 JSON 模式驗證是否到位，並在接受輸入之前進行驗證。 | ✓ | ✓ | ✓ | 20 |
| **13.2.3** | 通過使用以下至少一項或多項來驗證使用 cookie 的 RESTful Web services是否受到跨站點請求偽造（CSRF）的保護：雙重提交 cookie 模式、CSRF 隨機數或 Origin 請求頭檢查。 | ✓ | ✓ | ✓ | 352 |
| **13.2.4** | [已刪除，與 11.1.4 重複] |  |  |  |  |
| **13.2.5** | 驗證REST服務明確檢查傳入的Content-Type是否為預期型別，如application/xml或application/json。 |  | ✓ | ✓ | 436 |
| **13.2.6** | 驗證訊息頭和有效載荷是可信的，在傳輸過程中沒有被修改。在許多情況下，要求對傳輸進行強加密（僅TLS）可能就足夠了，因為它同時提供保密性和完整性保護。每條資訊的數字簽名可以在傳輸保護的基礎上，為高安全性的應用提供額外的保證，但也帶來了額外的複雜性和風險，需要權衡利弊。 |  | ✓ | ✓ | 345 |

## V13.3 SOAP Web Service

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **13.3.1** | 驗證是否進行了 XSD 模式驗證以確保 XML 文件格式正確，然後在對該資料進行任何處理之前驗證每個輸入欄位。 | ✓ | ✓ | ✓ | 20 |
| **13.3.2** | 驗證訊息負載是否使用 WS-Security 進行簽名，以確保客戶端和service之間的可靠傳輸。 |  | ✓ | ✓ | 345 |

注意：由於針對 DTD 存在 XXE 攻擊問題，因此不應使用 DTD 驗證，並且根據 V14 配置 中規定的要求，禁用框架 DTD 評估。

## V13.4 GraphQL

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **13.4.1** | 驗證是否使用查詢白名單或深度和數量限制的組合，來防止昂貴的巢狀查詢，導致對 GraphQL 或資料層表示式的拒絕服務（DoS）。對於更高階的場景，應該使用查詢成本分析。 |  | ✓ | ✓ | 770 |
| **13.4.2** | 驗證 GraphQL 或其他資料層的授權邏輯應在業務邏輯層，而不是 GraphQL 層實現。 |  | ✓ | ✓ | 285 |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Serverless Top 10](https://github.com/OWASP/Serverless-Top-10-Project/raw/master/OWASP-Top-10-Serverless-Interpretation-en.pdf)
* [OWASP Serverless Project](https://owasp.org/www-project-serverless-top-10/)
* [OWASP Testing Guide 4.0: Configuration and Deployment Management Testing](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/README.html)
* [OWASP Cross-Site Request Forgery cheat sheet](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)
* [OWASP XML External Entity Prevention Cheat Sheet - General Guidance](https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html#general-guidance)
* [JSON Web Tokens (and Signing)](https://jwt.io/)
* [REST Security Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html)
* [JSON Schema](https://json-schema.org/specification.html)
* [XML DTD Entity Attacks](https://www.vsecurity.com/download/publications/XMLDTDEntityAttacks.pdf)
* [Orange Tsai - A new era of SSRF Exploiting URL Parser In Trending Programming Languages](https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf)

# V14 配置

## 控制目標

確保經過驗證的應用程式具有：

* 一個安全的、可重複的、可自動化的構建環境。
* 加固第三方庫、依賴和配置管理，使應用不包括過時的或不安全的元件。

應用程式開箱即用配置應該是安全的，可以放在網際網路上，這意味著安全的開箱配置。

## V14.1 構建和部署

構建管道是可重複安全性的基礎——每次發現不安全的東西時，都可以在原始碼、構建或部署指令碼中解決它，並自動進行測試。我們強烈鼓勵使用自動化的構建管道來執行安全和依賴檢查，這些檢查會警告或破壞構建，以防止已知的安全問題被部署到生產環境中。不定期執行的手動步驟會直接導致可避免的安全錯誤。

隨著行業向DevSecOps模式的轉變，必須確保部署和配置的持續可用性和完整性，以實現“已知良好”的狀態。在過去，如果一個系統被入侵，需要幾天到幾個月的時間來證明沒有進一步的入侵發生。今天，隨著軟體定義的基礎設施、零停機時間的快速A/B部署和自動化容器構建的出現，自動和持續地構建、加固和部署一個“已知良好”的替代品來替代任何被入侵的系統，是有可能的。

如果傳統模式仍然存在，那麼就必須採取手動步驟來加固和備份該配置，以便及時用高完整性的、未受損害的系統快速替換受損害的系統。

遵守本節的規定要求一個自動化的構建系統，以及對構建和部署指令碼的訪問。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **14.1.1** | 驗證應用程式的構建和部署過程是以安全和可重複的方式進行的，如 CI / CD 自動化、自動配置管理和自動部署指令碼。 |  | ✓ | ✓ |  |
| **14.1.2** | 驗證編譯器標誌的配置是否配置為啟用所有可用的緩衝區溢位保護和警告，包括堆疊隨機化、資料執行保護，並在發現不安全的指標、記憶體、格式字串、整數或字串操作時中斷構建。 |  | ✓ | ✓ | 120 |
| **14.1.3** | 驗證伺服器配置是否按照應用程式伺服器和所使用框架的建議進行了加固。 |  | ✓ | ✓ | 16 |
| **14.1.4** | 驗證應用程式、配置和所有依賴項是否可以使用自動部署指令碼重新部署、在合理的時間內根據記錄和測試的執行手冊構建，或者及時從備份中恢復。 |  | ✓ | ✓ |  |
| **14.1.5** | 驗證授權管理員可以驗證所有安全相關配置的完整性，以發現篡改行為。 |  |  | ✓ |  |

## V14.2 依賴

依賴管理，對於任何型別應用程式的安全執行都至關重要。未能及時更新過時的或不安全的依賴，是迄今為止最大和最昂貴攻擊的根本原因。

注意：在 L1 ，14.2.1 的合規性與客戶端和其他庫、元件的觀察或檢測有關，而不是更準確的構建時靜態程式碼分析或依賴分析。這些更準確的技術可根據需要在訪談中發現。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **14.2.1** | 驗證所有元件都是最新的，最好是在構建或編譯時使用依賴檢查工具。 ([C2](https://owasp.org/www-project-proactive-controls/#div-numbering)) | ✓ | ✓ | ✓ | 1026 |
| **14.2.2** | 驗證所有不需要的功能、文件、示例應用程式和配置均已被刪除。 | ✓ | ✓ | ✓ | 1002 |
| **14.2.3** | 應用資產，例如JavaScript庫、CSS或網頁字型，如果被託管在外部的內容分發網路（CDN）或供應商，則驗證使用子資源完整性（SRI）來驗證該資產的完整性。 | ✓ | ✓ | ✓ | 829 |
| **14.2.4** | 驗證第三方元件來自預先定義的、可信的和持續維護的資源庫。 ([C2](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 829 |
| **14.2.5** | 驗證是否維護了正在使用中的所有第三方庫的軟體材料清單（SBOM）。 ([C2](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ |  |
| **14.2.6** | 驗證通過沙盒或封裝第三方庫來減少攻擊面，只將必需的行為暴露在應用程式中。 ([C2](https://owasp.org/www-project-proactive-controls/#div-numbering)) |  | ✓ | ✓ | 265 |

## V14.3 意外安全洩露

應加強生產配置以防止常見攻擊，例如除錯控制檯，提高跨站點指令碼（XSS）和遠端檔案包含（RFI）攻擊的門檻，並消除瑣碎的資訊發現“漏洞”，這是許多滲透測試報告中不受歡迎的標誌。 其中許多問題很少被評為重大風險，但它們可跟其他漏洞聯絡在一起。如果這些問題在預設情況下不存在，那就提高了大多數攻擊的門檻。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **14.3.1** | [已刪除，與 7.4.1 重複] |  |  |  |  |
| **14.3.2** | 驗證Web或應用伺服器和應用框架的除錯模式在生產中是否被禁用，以消除除錯功能、開發人員控制檯和非預期的安全披露。 | ✓ | ✓ | ✓ | 497 |
| **14.3.3** | 驗證HTTP標頭或HTTP響應的任何部分不暴露系統元件的詳細版本資訊。 | ✓ | ✓ | ✓ | 200 |

## V14.4 HTTP 安全標頭

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **14.4.1** | 驗證每個HTTP響應都包含一個 Content-Type 頭。如果內容型別是 text/\* 、 /+xml 和 application/xml ，還要指定一個安全的字符集（如UTF-8，ISO-8859-1）。內容必須與提供的Content-Type頭相匹配。 | ✓ | ✓ | ✓ | 173 |
| **14.4.2** | 驗證所有 API 響應是否包含 Content-Disposition: attachment; filename="api.json" 標頭（或內容型別的其他適當檔名）。 | ✓ | ✓ | ✓ | 116 |
| **14.4.3** | 驗證內容安全策略（CSP）響應標頭是否到位，有助於減輕對 HTML、DOM、JSON 和 JavaScript 注入漏洞等 XSS 攻擊的影響。 | ✓ | ✓ | ✓ | 1021 |
| **14.4.4** | 驗證所有響應是否包含 X-Content-Type-Options: nosniff 標頭。 | ✓ | ✓ | ✓ | 116 |
| **14.4.5** | 驗證所有響應和所有子域中是否包含 Strict-Transport-Security 標頭，例如 Strict-Transport-Security: max-age=15724800; includeSubdomains。 | ✓ | ✓ | ✓ | 523 |
| **14.4.6** | 驗證是否包含合適的 Referrer-Policy 標頭，以避免通過 Referer 標頭將 URL 中的敏感資訊暴露給不受信任的各方。 | ✓ | ✓ | ✓ | 116 |
| **14.4.7** | 驗證網路應用程式的內容在預設情況下不能被嵌入第三方網站，只有在必要時，才允許使用合適的Content-Security-Policy: frame-ancestors和X-Frame-Options響應頭嵌入確切的資源。 | ✓ | ✓ | ✓ | 1021 |

## V14.5 HTTP 請求頭驗證

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 描述 | L1 | L2 | L3 | CWE |
| **14.5.1** | 驗證應用伺服器只接受應用/API使用的HTTP方法，包括預檢請求的OPTIONS，並對使應用上下文無效的請求進行記錄/警告。 | ✓ | ✓ | ✓ | 749 |
| **14.5.2** | 驗證提供的 Origin 標頭是否不用於身份驗證或訪問控制決策，因為 Origin 標頭很容易被攻擊者更改。 | ✓ | ✓ | ✓ | 346 |
| **14.5.3** | 驗證跨域資源共享（CORS）的 Access-Control-Allow-Origin 標頭是否使用受信任域和子域的嚴格白名單匹配。並且不支援'null'源。 | ✓ | ✓ | ✓ | 346 |
| **14.5.4** | 驗證由受信任的代理或 SSO 裝置新增的 HTTP 標頭（例如bearer令牌）是否已通過應用程式的身份驗證。 |  | ✓ | ✓ | 306 |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Web Security Testing Guide 4.1: Testing for HTTP Verb Tampering](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/07-Input_Validation_Testing/03-Testing_for_HTTP_Verb_Tampering.html)
* 將 Content-Disposition 新增到 API 響應，有助於防止許多基於客戶端和伺服器之間的MIME型別誤解的攻擊，並且“filename”選項特別有助於防止 [Reflected File Download attacks.](https://www.blackhat.com/docs/eu-14/materials/eu-14-Hafif-Reflected-File-Download-A-New-Web-Attack-Vector.pdf)
* [Content Security Policy Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html)
* [Exploiting CORS misconfiguration for BitCoins and Bounties](https://portswigger.net/blog/exploiting-cors-misconfigurations-for-bitcoins-and-bounties)
* [OWASP Web Security Testing Guide 4.1: Configuration and Deployment Management Testing](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/README.html)
* [Sandboxing third party components](https://cheatsheetseries.owasp.org/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.html#sandboxing-content)

# 附錄A：詞彙表

* **地址空間佈局隨機化**（ASLR, Address Space Layout Randomization）- 一種技術，使記憶體損壞的漏洞利用更加困難。
* **允許列表** - 允許的資料或操作的列表，例如輸入驗證時允許執行的字元列表。
* **應用程式安全** - 應用程式級安全性側重於分析構成開放系統互連參考模型（OSI 模型）應用層的元件，而不是側重於例如底層作業系統或連線網路。
* **應用安全驗證** – 根據OWASP ASVS對一個應用進行技術評估。
* **應用安全驗證報告** - 記錄驗證者對某一特定應用的總體結果和支援性分析的報告。
* **認證** - 對應用使用者所聲稱身份的驗證。
* **自動化驗證** – 使用漏洞簽名來發現問題的自動化工具（動態分析工具、靜態分析工具或兩者兼有）。
* **黑盒測試** - 它是一種軟體測試方法，在不窺視其內部結構或工作原理的情況下檢查應用程式的功能。
* **元件** - 一個獨立的程式碼單元，有相關的磁碟和網路介面，與其他元件進行通訊。
* **跨站指令碼**（XSS, Cross-Site Scripting）- 通常在網路應用中發現的一種安全漏洞，允許在內容中注入客戶端指令碼。
* **加密模組** - 實現加密演算法或生成加密金鑰的硬體、軟體或韌體。
* **常見弱點列舉**（CWE, Common Weakness Enumeration）- 一個社群開發的常見軟體安全弱點列表。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。
* **設計驗證** - 對應用程式的安全架構進行技術評估。
* **動態應用安全測試**（DAST, Dynamic Application Security Testing）- 技術旨在檢測應用程式在執行狀態下的安全漏洞。
* **動態驗證** - 使用自動化工具，使用漏洞簽名，在應用程式的執行過程中發現問題。
* **快速線上身份認證**（FIDO, Fast IDentity Online）- 一組認證標準，允許使用各種不同的身份驗證方法，包括生物識別、可信平臺模組（TPM）、USB 安全令牌等。
* **全球唯一識別符號**（GUID, Globally Unique Identifier）– 在軟體中作為識別符號使用的唯一參考號。
* **超文字傳輸​​協議**（HTTPS）– 分散式、協作式、超媒體資訊系統的應用協議。它是全球資訊網資料通訊的基礎。
* **硬編碼金鑰** – 儲存在檔案系統中的加金鑰匙，無論是程式碼、註釋還是檔案。
* **硬體安全模組**（HSM, Hardware Security Module）- 硬體元件，能夠以受保護的方式儲存加密金鑰和其他密碼。
* **Hibernate查詢語言**（HQL）- 一種查詢語言，在外觀上類似於Hibernate ORM庫使用的SQL。
* **輸入驗證** - 對未受信任的使用者輸入的規範化和驗證。
* **惡意程式碼** - 在應用程式所有者不知情的情況下，在開發過程中將程式碼引入到應用程式中，從而規避了應用程式的預期安全策略。這與病毒或蠕蟲等惡意軟體不同！
* **惡意軟體** - 在應用程式使用者或管理員不知情的情況下，在執行期間被引入到應用程式的可執行程式碼。
* **OWASP**（OWASP，Open Web Application Security Project）– 開放網路應用安全專案（OWASP）是一個全球自由開放的社群，致力於提高應用軟體的安全性。我們的使命是使應用安全“可見”，以便人們和組織能夠對應用安全風險做出明智的決定。見： See: <https://www.owasp.org/>
* **一次性密碼**（OTP）- 唯一生成的密碼，可在單一場合中使用。
* **物件關係對映**（ORM）- 一種系統，用於允許使用應用相容的物件模型，在應用中引用和查詢基於關係/表的資料庫。
* **PBKDF2**（PBKDF2，Password-Based Key Derivation Function 2）- 一種特殊的單向演算法，用於從輸入文字（如密碼）和額外的隨機鹽值中建立一個強大的加密金鑰，因此，如果產生的值被儲存（而不是原始密碼），則可用於使密碼更難被離線破解。
* **個人可識別資訊**（PII）- 是指可單獨使用或與其他資訊一起使用的資訊，可用於識別、聯絡或定位一個人，或用於識別一個人的背景。
* **與位置無關的可執行檔案**（PIE）- 放置在主儲存器某處的機器程式碼體，無論其絕對地址如何，都能正確執行。
* **公鑰基礎設施**（PKI）- 將公鑰與實體的各自身份結合起來的一種安排。繫結是通過在證書機構（CA）註冊和頒發證書的過程建立的。
* **公共交換電話網**（PSTN）- 傳統的電話網路，包括固定電話和行動電話。
* **依賴方**（RP，Relying Party）- 指依賴使用者對單獨的認證提供者進行認證的應用程式。該應用程式依賴於該身份驗證提供者提供的某種令牌或一組簽名斷言，來相信使用者就是他們所說的那個人。
* **靜態應用安全測試**（SAST）- 一套分析應用原始碼、位元組碼和二進位制檔案的技術，用於瞭解表明存在安全漏洞的編碼和設計場景。SAST解決方案在非執行狀態下從“內部”分析一個應用程式。
* **軟體開發生命週期**（SDLC）- 軟體從最初的需求到部署和維護的一步步發展過程。
* **安全架構** - 應用程式設計的抽象，確定和描述安全控制的位置和方式，同時也確定和描述使用者和應用程式資料的位置和敏感性。
* **安全配置** - 應用程式的執行時配置，影響安全控制的使用方式。
* **安全控制** - 執行安全檢查（如訪問控制檢查）或在呼叫時產生安全效果（如生成審計記錄）的功能或元件。
* **伺服器端請求偽造**（SSRF）- 濫用伺服器上的功能，通過更改在伺服器上執行的程式碼會讀取或提交資料的URL，來讀取或更新內部資源的攻擊。
* **單點登入驗證**（SSO）- 這發生在使用者登入到一個應用程式，然後就自動登入到其他應用程式，而無需重新認證。例如，當你已登入到Google時，在訪問其他谷歌服務，如YouTube、谷歌文件和Gmail時，你將自動登入。
* **SQL注入**（SQLi）- 一種程式碼注入技術，用於攻擊資料驅動的應用程式，其中惡意的SQL語句被插入到一個入口點。
* **SVG** - 可擴充套件向量圖形
* **基於時間的OTP** - 一種生成OTP的方法，將當前的時間作為生成密碼的演算法的一部分。
* **威脅建模** - 一種技術，包括開發越來越精細的安全架構，以確定威脅代理、安全域、安全控制以及重要的技術和商業資產。
* **傳輸層安全**（TLS）- 通過網路連線提供通訊安全的加密協議。
* **信任平臺模組**（TPM，Trusted Platform Module）- 一種HSM，通常連線到較大的硬體元件，如主機板，並作為該系統的“信任根”。
* **雙因素認證**（2FA）- 這為賬戶登入增加了第二層認證。
* **通用第二因素**（U2F）- 由FIDO建立的標準之一，專門用於允許USB或NFC安全金鑰作為第二認證因素使用。
* **URI/URL/URL分片** – 統一資源識別符號是用於標識web資源名稱或web資源的字串。統一資源定位符通常用作對資源的引用。
* **驗證者** – 根據OWASP ASVS要求稽核應用程式的人員或團隊。
* **所見即所得**（WYSIWYG，What You See Is What You Get）- 一種富文字的內容編輯器，顯示內容在渲染時的實際效果，而不是顯示用於管理渲染的編碼。
* **X.509證書** - X.509證書是一種數字證書，它使用廣泛接受的國際X.509公鑰基礎設施（PKI）標準，來驗證公鑰是否屬於證書中包含的使用者、計算機或服務身份。
* **XML外部實體**（XXE，XML eXternal Entity）- 一種XML實體，可以通過宣告的系統標識訪問本地或遠端內容。這可能會導致各種注入攻擊。

# 附錄B：參考文獻

以下OWASP專案最可能對本標準的使用者/採用者有用：

## OWASP核心專案

1. OWASP Top 10專案：<https://owasp.org/www-project-top-ten/>
2. OWASP網路安全測試指南：<https://owasp.org/www-project-web-security-testing-guide/>
3. OWASP 主動控制：<https://owasp.org/www-project-proactive-controls/>
4. OWASP安全知識框架：<https://owasp.org/www-project-security-knowledge-framework/>
5. OWASP軟體保障成熟度模型（SAMM）：<https://owasp.org/www-project-samm/>

## OWASP Cheat Sheet系列專案

[該專案](https://owasp.org/www-project-cheat-sheets/) 有許多與 ASVS 中的不同主題相關的備忘單。

可以在此處找到到 ASVS 的對映： <https://cheatsheetseries.owasp.org/cheatsheets/IndexASVS.html>

## 移動安全相關專案

1. OWASP 移動安全專案：<https://owasp.org/www-project-mobile-security/>
2. OWASP Mobile Top 10 風險：<https://owasp.org/www-project-mobile-top-10/>
3. OWASP移動安全測試指南和移動應用安全驗證標準：<https://owasp.org/www-project-mobile-security-testing-guide/>

## OWASP物聯網相關專案

1. OWASP物聯網專案：<https://owasp.org/www-project-internet-of-things/>

## OWASP Serverless專案

1. OWASP Serverless專案：<https://owasp.org/www-project-serverless-top-10/>

## 其他

同樣，以下網站最有可能對本標準的使用者/採用者有用

1. SecLists Github: <https://github.com/danielmiessler/SecLists>
2. MITRE 常見弱點列舉: <https://cwe.mitre.org/>
3. PCI 安全標準委員會: <https://www.pcisecuritystandards.org>
4. PCI 資料安全標準（DSS）v3.2.1 要求和安全評估程式：<https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf>
5. PCI 軟體安全框架 - 安全軟體要求和評估程式： <https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf>
6. PCI 安全軟體生命週期（Secure SLC）要求和評估程式： <https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf>

# 附錄C: 物聯網驗證要求

本章原本是在main分支中，但考慮到OWASP IoT團隊已完成的工作，所以在該主題上維護兩個不同的執行緒沒有意義。對於 4.0 版本，我們將其移到附錄中，並敦促所有需要此功能的人使用主要的 [OWASP IoT 專案](https://owasp.org/www-project-internet-of-things/)

## 控制目標

嵌入式/IoT裝置應該滿足：

* 通過在受信任的環境中實施安全控制，在裝置內擁有與伺服器中相同級別的安全控制。
* 儲存在裝置上的敏感資料，應使用硬體支援的儲存（如安全元件）以安全的方式完成。
* 從裝置傳輸的所有敏感資料，都應利用傳輸層安全。

## 安全驗證要求

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 說明 | L1 | L2 | L3 | 起始時間 |
| **C.1** | 驗證應用層除錯介面，如USB、UART和其他序列變體，是否被禁用或受到複雜密碼的保護。 | ✓ | ✓ | ✓ | 4.0 |
| **C.2** | 驗證加密金鑰和證書對於每個單獨的裝置都是唯一的。 | ✓ | ✓ | ✓ | 4.0 |
| **C.3** | 驗證嵌入式/IoT 作業系統（如果適用）是否啟用了記憶體保護控制（如 ASLR 和 DEP）。 | ✓ | ✓ | ✓ | 4.0 |
| **C.4** | 驗證是否禁用了 JTAG 或 SWD 等片上除錯介面，或者是否啟用並正確配置了可用的保護機制。 | ✓ | ✓ | ✓ | 4.0 |
| **C.5** | 驗證是否已實施並啟用受信任的執行（如果在裝置 SoC 或 CPU 上可用）。 | ✓ | ✓ | ✓ | 4.0 |
| **C.6** | 驗證敏感資料、私鑰和證書是否安全儲存在Secure Element、TPM、TEE (Trusted Execution Environment)中，或使用強加密保護。 | ✓ | ✓ | ✓ | 4.0 |
| **C.7** | 驗證韌體應用程式使用傳輸層安全，保護傳輸中的資料。 | ✓ | ✓ | ✓ | 4.0 |
| **C.8** | 驗證韌體應用程式驗證與伺服器連線的數字簽名。 | ✓ | ✓ | ✓ | 4.0 |
| **C.9** | 驗證無線通訊鑑權。 | ✓ | ✓ | ✓ | 4.0 |
| **C.10** | 驗證無線通訊是否通過加密通道傳送。 | ✓ | ✓ | ✓ | 4.0 |
| **C.11** | 驗證任何被禁止的C函式，都被替換成適當的安全函式。 | ✓ | ✓ | ✓ | 4.0 |
| **C.12** | 驗證每個韌體都有一個軟體材料清單，其中包括第三方元件、版本和已公佈的漏洞。 | ✓ | ✓ | ✓ | 4.0 |
| **C.13** | 驗證所有程式碼，包括第三方二進位制檔案、庫、框架都經過審查，以防止硬編碼憑據（後門）。 | ✓ | ✓ | ✓ | 4.0 |
| **C.14** | 通過呼叫shell命令封裝器、指令碼或安全控制，來防止作業系統命令注入，驗證應用程式和韌體元件不受作業系統命令注入的影響。 | ✓ | ✓ | ✓ | 4.0 |
| **C.15** | 驗證韌體應用程式將數字簽名固定到可信伺服器。 |  | ✓ | ✓ | 4.0 |
| **C.16** | 驗證是否存在防篡改或篡改檢測功能。 |  | ✓ | ✓ | 4.0 |
| **C.17** | 驗證是否啟用了晶片製造商提供的任何可用的智慧財產權保護技術。 |  | ✓ | ✓ | 4.0 |
| **C.18** | 驗證安全控制是否到位，以阻止韌體逆向工程（例如，刪除冗長的除錯符號）。 |  | ✓ | ✓ | 4.0 |
| **C.19** | 驗證裝置在載入前校驗啟動映象的簽名。 |  | ✓ | ✓ | 4.0 |
| **C.20** | 驗證韌體更新過程不會受到“檢查時間與使用時間”攻擊（譯者注：time-of-check vs time-of-use attacks）。 |  | ✓ | ✓ | 4.0 |
| **C.21** | 驗證裝置在安裝前使用程式碼簽名並校驗韌體升級檔案。 |  | ✓ | ✓ | 4.0 |
| **C.22** | 驗證裝置不能被降級到有效韌體的舊版本（防回滾）。 |  | ✓ | ✓ | 4.0 |
| **C.23** | 驗證嵌入式裝置使用了密碼學安全的偽隨機數生成器（例如，使用晶片提供的隨機數生成器）。 |  | ✓ | ✓ | 4.0 |
| **C.24** | 驗證韌體能夠按照預定的時間表，執行自動韌體更新。 |  | ✓ | ✓ | 4.0 |
| **C.25** | 驗證裝置在檢測到篡改或收到無效資訊時，能擦除韌體和敏感資料。 |  |  | ✓ | 4.0 |
| **C.26** | 驗證只使用了支援禁用除錯介面（如JTAG、SWD）的微控制器。 |  |  | ✓ | 4.0 |
| **C.27** | 驗證只使用了提供實質性保護的微控制器，以防止“去封裝”（譯者注：de-capping, decapsulation）和側通道攻擊。 |  |  | ✓ | 4.0 |
| **C.28** | 驗證敏感導線不暴露在印刷電路板的外層。 |  |  | ✓ | 4.0 |
| **C.29** | 驗證晶片間的通訊是加密的（如主機板到子板的通訊）。 |  |  | ✓ | 4.0 |
| **C.30** | 驗證裝置使用程式碼簽名並在執行前驗證程式碼。 |  |  | ✓ | 4.0 |
| **C.31** | 驗證儲存在記憶體中的敏感資訊一旦不再需要，就立即用零值覆蓋。 |  |  | ✓ | 4.0 |
| **C.32** | 驗證韌體應用程式利用核心容器在應用程式之間進行隔離。 |  |  | ✓ | 4.0 |
| **C.33** | 驗證安全編譯器標誌，例如 -fPIE, -fstack-protector-all, -Wl,-z,noexecstack, -Wl,-z,noexecheap 已配置到韌體構建中。 |  |  | ✓ | 4.0 |
| **C.34** | 驗證微型控制器是否配置了程式碼保護（如果適用）。 |  |  | ✓ | 4.0 |

## 參考文獻

有關更多資訊，請參閱：

* [OWASP Internet of Things Top 10](https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf)
* [OWASP Embedded Application Security Project](https://owasp.org/www-project-embedded-application-security/)
* [OWASP Internet of Things Project](https://owasp.org/www-project-internet-of-things/)
* [Trudy TCP Proxy Tool](https://github.com/praetorian-inc/trudy)