網頁常見 10 大風險整理 – OWASP–2021

撰文｜ OWASP 官方網站、叡揚資訊 資安事業處 整理提供

OWASP(Open Web Application Security Project) 是一個致力於提升軟體安全的非營利組織，利用社群、全球數百的分會、數萬成員的力量，對於不同主題的，累積了各種對於軟體安全研究的專案，其中莫過於大家所熟知的 OWASP Top 10，網路常見十大風險，依過往慣例每 3~4 年會公告一次，最新的新版為 2021 年底所發佈的版本。

在這個版本中，我們可以觀察到有三個新類別出現，四個類別將範圍進行了調整，當然有些類別進行順序的改變。針對 OWASP 十大網路應用系統安全弱點說明如下：

第一名
2021–Broken Access Control
權限控制失效

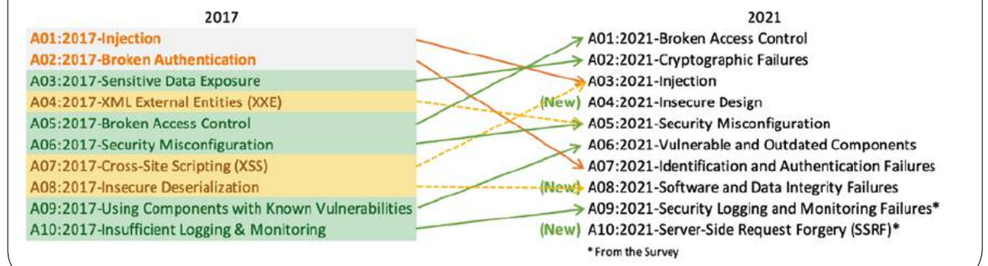
由第 5 名前進到第 1 名。

建議處理檔案或敏感性資料時，需嚴格進行監控及身份權限驗證，使用者無法擁有預期權限之外的操作，降低駭客利用漏洞存取未經驗證或授權的功能，進而察看敏感資料、修改數據和提升存取權限等。

可能的危險情境為：

- 通過修改 URL、內部應用程式狀態或 HTML 頁面，或僅使用自定義 API 攻擊工具來繞過存取控制檢查。
- 容許提權為其他用戶，允許查看或編輯其他人的帳戶。
- 控制失效通常會導致未經授權的資訊洩露、修改或損壞所有資料，或執行超出用戶權限的業務功能。
- 常見的存取控制弱點包括：
  1. 允許查看或編輯其他人的帳戶 / 記錄。
  2. 未登入即成為用戶，或一般使用者身份提權為管理者。
  3. 缺少存取控制的 API– 得以進行 POST、PUT 和 DELETE 操作。
  4. CORS(Cross–Origin Resource Sharing 跨來源資源共用) 錯誤配置允許未經授權的 API 存取。

▶OWASP Top 10 從 2017 到 2021 的版本變化



第二名
2021– Cryptographic Failures
加密機制失效

由第 3 名前進到第 2 名。

其實應用系統中往往隱含了許多敏感資訊，無論是財務資料、醫療資料、個人資料甚至於系統運作時的設定 / 交易資料等等，若系統的設計考量未從保護敏感性資料出發，即有可能讓攻擊者透過竊取或修改未加密數據，實施詐騙、身份盜取或其他犯罪行為。未加密的敏感資料容易受到破壞，因此，使用者需要對敏感資料加密，包括傳輸過程中的資料、儲存的資料及瀏覽器的交換資料。建議檢視應用系統是否含有以下問題：

- 系統是否以明碼傳輸？像是 HTTP/SMTP/FTP 等等協定。
- 是否有任何老舊或脆弱的加密演算法被預設使用或存在於較舊的程式碼？
- 是否有任何預設的加密金鑰被使用、脆弱的加密金鑰被重複使用，是否有適當的金鑰管理或金鑰輪換？

第三名
2021–Injection 注入式攻擊

下滑到第三名。結合 Injections & XSS 攻擊。此次調查過程中發現被測試的應用程式中高達 94% 都有測到某種類別注入式攻擊的問題。

無論何種應用程式，Injection 類型的弱點是發容易被忽略的，尤其系統中有滿多情境都是需要利用「外部輸入值」作為組合指令的一部份，而若於取得前未經相關的處理 / 驗證，可能會使攻擊者可以插入其他語句來對系統進行操作；Injection 最具代表性的弱點莫過於 SQL Injection，但其實 Injection 的泛指了各式指令執行，如 OS 指令、LDAP 指令、XML 指令、物件關係對映 (ORM)……等。

除了 Injection 在此弱點中還有另一大弱點 – 跨站腳本攻擊 (Cross–site Scripting, XSS)，相信此弱點對於網路應用系統的負責人員來說是非常耳熟的，此弱點也是上一版的第 7 名。XSS 簡單的說是將「外部輸入值」作為系統畫面呈現的資料來源之一，而若未進行資料驗證或編碼時，就有可能引發。

第四名
2021– Insecure Design 不安全設計

全新類別。

在開發系統時一定聽過軟體開發生命週期 (Software Development Life Cycle, SDLC)，然爾隨著時代的演進在這幾年間另一個新穎名稱出現了：安全軟體開發生命週期 (Secure Software Development Life Cycle, SSDLC)，顧名思意即是在原始的週期中加上安全的考量，此類別就此而生了，主要為了提倡「Shift–Left」的概念，流程通常是由左而右，向左移也就是將部份事情提早開始，像是安全的需求、設計、威脅建模等等的設計規畫作業都因以安全為出發點。不再像過去功能至上的角度來進行系統建置。

不安全設計是一個涵括多種弱點的代稱，主要需要注意的為：「缺乏或無效的控制設計」、「沒有有效的控制措施」、「未定義安全邊界」等，這些不安全的設計多半都是不小心忽略而造成的，讓所有參與專案的成員都具備安全系統設計、開發的認知就非常重要了。

第五名
2021– Security Misconfiguration 未使用安全設定

由第 6 名前進到第 5 名。

應用系統的組態設定有許多應注意的小細節，往往可決定整個應用系統的安全等級，像是啟用了不安全的設定值如：錯誤的 HTTP 標頭配置、除錯屬性開啟，導致提供過多敏感資訊的詳細錯誤、未使用加密通道、未使用有效憑證等等。而系統的管理者不僅需要對所有的操作系統、框架、函式庫和應用程序進行安全的設定，同時也需注意修補和升級。

- 常見的組態設定的問題包括：
  1. 不必要的功能啟用或是安裝。
  2. 使用預設帳號與密碼。
  3. 因錯誤處理而暴露出的堆疊追蹤，或是向使用者，暴露出過多的錯誤警告資訊。

第六名
2021– Vulnerable and Outdated Components 危險或過舊的元件

由第 9 名前進到第 6 名。

你的系統不是你的系統，我想這句話應該讓人覺得困惑，但現實的情境即是如此，多數的應用系統已不再是從頭到尾由自己的團隊撰寫完成，而是利用許多的開源軟體或是元件再加上自行撰寫程式「組裝」而成。也因如此對於系統的管理還需要注意函式庫、框架及套件管理工具，並細至弱點通報、版本的更新，尤其像 2021 年底的 Log4J 事件，需預先設想的是若未再發生熱門的元件發生問題時，對於盤點、測試、升級作業……等機制應對如下：

- 建立軟體物料清單 (SBOM)，至少每季分析、解決、記錄第三方套件的問題。
- 完全掌控第三方套件中已知的安全問題，建立可使用清單，有漏洞的要修正，有新的漏洞資訊要知道，EOS 的元件的要升級，不使用的元件要拿掉。
- 第三方元件檢測與建構流程合併，當上版有碰到弱點的元件時，要阻止上線。

第七名
2021–Identification and Authentication Failures 認證及驗證機制失效

由第 2 名前降至第 7 名。

應用系統是如何對操作者進行身份識別的呢？通常都離不開 Session，因此對於 Session 的管理 / 驗證 / 識別對於身份的部份就很重要了，常見的問題如下：

- 忘記密碼的流程，如不安全的 " 個人相關問答。
- 將密碼使用明碼、加密或較脆弱雜湊法的方式儲存。
- 密碼字典檔攻擊或其他自動化攻擊。
- 允許預設、脆弱、常見的密碼，像是 "Password1" 或 "admin/admin"。
- 於 URL 中洩漏 Session
- 成功登入後沒有更換 Session。
- 沒有正確的註銷 Session。

第八名
2021–Software and Data Integrity Failures 軟體及資料完整性失效

全新類別。

營運環境中的應用系統隨著時間不定期的改版，要如何確保程式碼、系統架構之完整性呢？而系統的運作是否依賴來自於不受信任來源，外掛程式、函式庫或模組。

甚至有未經驗證之自動更新功能。這個問題就需要回歸源頭確認於 SDLC 流程之中，是否有下述相關的配套措施：

- 程式碼入版控的機制。
- 函式庫及從屬套件，例如 npm 或 Maven，是從受信任的函式庫取得。
- 建立持續整合與部署 (CI/CD) 完善的機制。

第九名
2021– 資安記錄及監控失效

由第 10 名前進到第 9 名。

記錄與監控不足會讓攻擊者能夠進一步的攻擊系統、竄改資料、存取資料或是刪除資料。而且大多數的研究報告指出，當系統被攻擊後，受害系統要花超過 200 天以上才會發現資料外洩，且通常是透過第三方檢測工具發現的，而不是透過內部流程監控；建議應注意以下常忽略的小細節：

- 確保日誌格式符合一般日誌管理系統常用格式。
- 確保日誌經正確編碼以防止遭受注入攻擊或日誌 / 監控系統遭受攻擊。
- DevSecOps 團隊應建立有效地監控及告警機制以利偵測可疑活動並快速應變。

第十名
2021– 伺服端請求偽造

全新類別。

這次被抽出來獨立的品項，雖然發生的機率相對來說偏低，但一但發生後衝繫危害程度很高，例舉來說網頁應用程式存取遠端資源時，未驗證由使用者提供的網址，此時有機會發生偽造伺服端請求。即便有防火牆、VPN 或其他網路 ACL 保護的情況下，攻擊者仍得以強迫網頁應用程式發送一個經過捏造的請求給一個非預期的目的端。

現今的網頁應用程式已相當普及，存取網址是非常常見的，而雲端服務的方便性也登高偽造請求的可能性。從以下二種面向來進行防禦：

- From Network layer (從網路層著手)
  1. 區分網路區塊，彼此獨立不互連
  2. 對於外部預設全部拒絕，封鎖全部來自外部之網路流量
- From Application layer: (從應用層)
  1. 過濾並驗證所有外來的輸入
  2. 以白名單方式驗證
  3. 不傳送原始回應
  4. 停用 HTTP 重新導向