15個用於數位身分識別的程式碼安全實踐

任何應用程式或服務的程式碼安全都很重要，特別在處理個人資料的系統中，這一點更加至關重要。

文／Susan Morrow　譯／曹乙帆

當軟體分析公司CAST分析了1,380個軟體應用程式時，他們在程式碼中發現高達130萬個軟體漏洞。

任何理解資安長（CSO）一職重要性的人都會知道，軟體瑕疵會為網路罪犯打開長驅直入的大門。

「身分識別管理」可以說是所有技術學科中風險最高的項目。身分盜用更一直困擾著我們。Javelin Research多年來一直發表關於身分盜用的研究報告，他們的研究顯示該安全威脅持續困擾著身分識別管理產業。諮詢顧問公司Javelin在其發表的2018年身分詐欺報告裡強調，2017年的統計數字達到歷史新高。

在身分識別管理中，我們常常會談到「身分武裝化」（Weaponizing Identity）一詞。這意味著要強化橫跨無線基地台（AP）的系統，以及從中與服務介接的使用者。然而，武裝化程序必須分層進行，其中一層是在程式碼層級上。

安全的身分管理程式碼

數位身分識別平台可能非常複雜，因為它們通常必須依賴外部資料源並與第三方API整合。消費者的身分識別與存取管理（Identity and Access Management，IAM）可能更複雜。它們可能需要擴展功能以便上傳、儲存和共享文件和圖像。許多身分識別服務還包含或完全基於行動裝置App。僅僅依靠跨身分識別生態系統元件之通訊協定的固有安全性是不夠的。底層程式碼必須在不會對該生態系統的功能造成限制的情況下儘可能地安全。

以下是開發身分識別平台時可採行的最佳程式碼安全實踐：

1.使用良好的資源

就從擁有許多程式碼安全資源的開放網站應用程式安全計畫（Open Web Application Security Project, OWASP）開始。OWASP實際上是程式碼安全的首選資源。他們的程式碼安全「快速參考指南」是一個很好的起點，並且可以在開發過程中用來當作雙重檢查工具使用。請充份利用他們的資源吧！

2.使用防禦性程式設計技術

這有助於避免會有被駭客用來發動漏洞攻擊的程式瑕疵。一個例子是相等性比較（Equivalence Comparisons）：把常數放在首位。如果你不小心把相等運算子（Equality Operator）寫成等號，這樣就會在編譯或執行階段時觸發錯誤。（例如圖1）

3.淨化資料

圖1

```
// poor practice:
if ($result == 'SUCCESS') {
// better - if "==" mistyped as "=" get runtime or compile error
 if ('SUCCESS' == $result) {
```

最佳程式碼安全實踐

1. 使用良好的資源。
2. 使用防禦性程式設計技術。
3. 淨化資料。
4. 篩檢。
5. 過濾器。
6. 防止未經驗證的程式碼執行。
7. 韌性才是王道
8. 使用開放原始碼時要小心。
9. 錯誤回應。
10. 使用稽核日誌。
11. 數位簽章。
12. 用於安全通訊的加密令牌（Token）。
13. 行動 App。
14. 檢查漏洞。
15. 最重要的是－－跟上新興漏洞的腳步。

數位身分識別，尤其是消費者的系統，通常會呼叫外部資料源。所有來自外部源或由使用者提供的資料都不可信賴。對於Web客戶端來說，這包括了從查詢字串或雜湊參數、cookie、本地端儲存等獲得的資料；對於伺服器端應用程式而言，這包括藉由POST、GET、cookie等提供的資料。原生應用程式通常會讀取組態檔，這些檔案可能會被故意篡改。

在所有情況下，第一道防線是「淨化」（Sanitization）：檢查是否只包含了允許的字元/格式。這包括確保檢查了最大資料欄位長度，以避免緩衝區溢位攻擊。

另一個對數位身分識別平台安全檢查很重要的領域是圖像上傳。隨著資料儲存越來越多地儲存和共享可識別文件的照片，圖像在IAM和客戶身分識別與存取管理（CIAM）服務中變得越來越重要。使用者上傳像是圖像等檔案時會特別危險。這些都必須嚴格檢查，以確保它們確實是圖像，並且沒有隱藏可執行內容。

4.篩檢

文件和其他檔案儲存是數位身分識別生態系統裡的主要部分。所有放在儲存媒體中的資料都必須篩選檢查出是否存在可執行內容。這適用於任何底層資料庫技術。例如，雖然非SQL資料庫不容易受到資料隱碼（SQL Injection）攻擊，但它們確實也有自己的漏洞。

5.過濾器

同樣的，永遠不要直接接受外部命令字串。始終經由過濾器檢查命令在該情境中是否有效且適用。

6.防止未經驗證的程式碼執行

避免使用eval()類型函數，因為這些函數允許執行未經驗證的程式碼。這包括使用了能讓eval()隱藏的函數，例如JavaScript的SetTimer()。

7.韌性才是王道

開發消費者導向的數位身分識別服務通常意味著需要涵蓋廣泛的人群。使用者需要考量廣泛的值。嘗試在為應用程式進行編寫時能考量到韌性（Resilience），如果外部資料值超出預期範圍或預期值，則應用程式不會崩潰或允許不明程式碼注入。良好的單元或功能測試通常可以檢測到這些問題。

8.使用開放原始碼時要小心

允許在構建中使用最新版本的開源程式碼會導致產品中包含惡意軟體。使用開源軟體套件時，請透過在套件管理器中設置特定版本號來防止使用未驗證的版本。構建過程應包括對所有外部檔案的雜湊檢查。

9.錯誤回應

錯誤回應是數位身分識別服務的重要組成部分，因為它們會告知使用者問題並提高可用性。然而，惡意實體可以使用它們來計算系統的行為。務必確保這些資訊不會洩漏給攻擊者。主從式錯誤回應應該只包含最少的資訊以通知使用者，而不致於幫助攻擊者。

10.使用稽核日誌

如果確實發生了攻擊，使用者會想知道是什麼攻擊，以及如何發生的。然而，請更理智地面對日誌紀錄 - 用無意義的稽核資料填寫日誌可能會適得其反。

11.數位簽章

在可能的情況下，使用數位簽章來驗證資料的完整性。雜湊訊息驗證碼（Hash-based Message Authentication Code, HMAC）的計算成本低廉，在這方面極具作用價值。同樣的，如果資料很敏感，包括了個人資料，那麼就應該在傳輸和儲存期間進行加密。對於任何通過公共或私有網路傳輸的內容，務必始終採用TLS加密協定。

12.用於安全通訊的加密令牌（Token）

對於程序間通訊（Inter-Process Communication），請考慮使用加密令牌來代替密碼身分驗證。

13.行動App

行動裝置App越來越多地被用作數位身分證明和交易的管道。假設裝置可以越獄。那麼請仔細檢查正在處理的資料並評估所涉及的可能風險。考慮在可能的情況下結合越獄檢測碼。

14.檢查漏洞

每當完成新的升級或新發布版本，都應該執行這個動作。有許多專業公司可提供這樣的服務。像是CheckMarx或CAST等公司就提供獨立的原始碼分析工具（Source Code Analyzer）服務。這類解決方案是任何程式碼安全實踐的重要組成部分，儘管它們不應被視之為程式碼手工檢查的替代品，而應看作額外一層的檢查。

15.最重要的是——與時俱進地跟上新興漏洞的腳步

如同常見漏洞披露（CVE）安全通報細節所證明的那樣，重大的安全瑕疵總是會一直出現。了解新威脅並採取行動以確保程式碼中不存在這些威脅是非常重要的。

任何應用程式或服務的程式碼安全都很重要，特別在處理個人資料的系統中，這一點更加至關重要。只要使用者設計或開發數位身分識別系統，就會發現安全性和可用性總像處在天平的兩端而難以平衡。然而，透過良好的程式碼安全實踐，就可以管理網路攻擊威脅，同時確保消費者適用的身分識別系統的複雜功能是可以達成的。