An Empirical Study of Cryptographic Misuse in Android Applications

R07944058
網媒碩一 陳鵬宇

1 Summary

加密一直是資訊安全很重要的議題，如何選用安全的加密演算法？使得加密後的密文不容易被回推至原來的明文，及選擇使用怎麼樣的模式？舉凡 ECB、CBC、PBE 等等，是否符合 IND-CPA 的安全標準？攻擊者無法透過窮舉明文，並觀察明文與密文之間的關係來找出「正確的」明文。

本篇論文運用一個輕量化的工具：CRYPTOLINT，去分析了上架在安卓系統各式各樣的應用程式，是否有誤用密碼機制的現象。透過 CRYPTOLINT，我們得出了有將近 88% 的應用程式都違反了一定數量的安全規則。

2 Strength(s)

作者詳細的解釋了密碼學中，應有的安全規則，例如為什麼 ECB 模式不適合用來加密，因為一樣的明文，總是會生出相同的密文，這樣欠缺隨機性很容易受到 CPA 攻擊。即便我們使用了 CBC$，看似更安全了，但若 IV 不是隨機的，仍然能夠透過課堂上介紹的 CBC Padding Oracle Attack 去攻擊，透過每次猜測最後一個 byte，慢慢由後往前推，我們可得出當解開一完整 block 所需要的時間複雜度為 $O(256 \times \#\text{ of bytes})$。

論文中提到的 **Rule 1** 到 **Rule 4** 和 **Rule 6** 都圍繞在一個中心思想——「增加隨機性」，ECB 的隨機性很差，攻擊者可輕易地利用如 $M_1 = 0^{2n}$ 和 $M_0 = 0^n1^n$，的明文送給加密方，再藉由觀察回傳回來的銘文來攻破此系統。而 **Rule 5** 則是「增加計算複雜度」，Apple 透過增加迭代次數，成功降低惡意人士攻擊成功機率，並且在使用者端不造成能查覺的延遲。

最後，作者由 Google Play 下載分析了 145,095 應用程式，並給出了每個應用所違反漏洞是哪些，能讓讀者清楚的知道原來有這麼多應用都是不安全的，也給出了建議的改善方式，並希望未來開源，每個開發者都能在上架自己應用前，先來 CRYPTOLINK 測試。

3 Weakness(s)

若 IV 是可預測的話仍然不夠安全，可預測的 IV 可能受到 CPA 攻擊。我們假設以下情境：Eve 是一個公司的資料庫管理員 (DBA)，今天 Eve 很想看到某個使用他們公司服務的人的個資，但每個欄位都是加密的，因此即使 Eve 是 DBA，他也只能訪問密文。

在 CBC 中，IV 首先與明文 (P_1) 進行 XOR ($\oplus$) :$C_1 = E_k(IV \oplus P_1)$，由於 Eve 是 DBA，所以他也可以使用公司的服務，為自己選取明文。如果 Eve 可以預先預測將應用於他自己 (IV_{eve}) 和 Alice (IV_{alice}) 的 IV，則他可以選則自己的明文如下：$P_{eve} = IV_{eve} \oplus IV_{alice} \oplus$ FALSE。之後經過 block cipher encryption 時，將會加密如下：$C_{eve} = E_k(IV_{eve} \oplus P_{eve}) = E_k(IV_{eve} \oplus (IV_{eve} \oplus IV_{alice} \oplus \text{FALSE})) = E_k(IV_{alice} \oplus \text{FALSE})$，即：$C_{eve} = E_k(IV_{alice} \oplus \text{FALSE})$。現在，Eve 可以比較 C_{eve} 和 C_{alice}。如果他們不同，他就知道 Alice 輸入的資料是 TRUE。

4 Reflection

看完這篇論文，讓我想起了第一週閱讀的"The Developer is the Enemy"，不是每一名開發者都是資安大師，在資安背景知識有限情況下，提供開發者一套安全驗證工具或許是個「暫時的」解決方案。CRYPTOLINK 的實驗結果，揭露了原來有這麼多的關發者都沒有在資安做好把關，即便有想要改善了，都還是存在一定程度的漏洞，像是論文中提到一個例子，本來用 ECB，後來改用 CBC 後還是用固定的 IV，到了最後決定隨機生成 IV 了，該開發者還是使用 regular random number generator 而不是更為建議的 SecureRandom API。