Critique3: How Bad It Git? Characterizing Secret Leakage in Public GitHub Repositories

B07902127 資工二 羅義鈞

Summary:

本篇 paper 探討 GitHub 上的 public repositories 究竟洩漏了多少 secret key。藉由檢測 public commit 以及 public snapshot 是否有 private key 或是 distinct API key 的形式，做出一個對於 secret leakage 的 lower bound 估計。主要透過兩個方式來進行 file collection，得到 candidate 後再透過三層 filter 去得到最後可能藏有 sensitive keys 的檔案。同時研究是否有補救辦法，以及經過實驗發現，若是不小心在 public repo 上洩漏的資訊，被偵測到的時間中位數約為 20 秒。而作者同時也測試 TruffleHog 是否能有效幫助 secret key 檢測，發現準確率僅接近 25%。由此可知 GitHub 上面有許多資訊安全的問題仍待解決。

Strength:

不同於過去的研究，作者藉由他們設計的有效方式，搜集了大量的數據並且對其做分析與實驗，這是我認為不容易做到的，也是這篇 paper 相較於其他對於 public repo 是否有 secret leakage 的研究，更加的全面性。不僅將問題用數據清楚的表示清楚，作者同時提出一些補救或是緩和 secret leakage 的方法，像是針對 GitHub 本身的 security alert program 去做延伸，或是將作者提出的技術應用在 monitor 上。

Weakness:

首先，透過作者提出的技術所得到的那些 secret 不能保證是 sensitive 的，這來自本次實驗的限制—作者們基於道德和法律考量, 無法去實際試驗這些得到的 secret。再者, 作者的技術只是概念化的提出, 例如 phase0 的篩選機制，透過去辨別 distinct structure of key sets 就能得到初步篩選，接著是 manually construct “distinct secret regular expression”，這些具體要如何做，是怎樣的演算法設計，我認為說明的不是很清楚，仍對其保有疑慮。還有，validity filter 具有一些 trade off，會刪掉一些其實是有 sensitive secret 的檔案。最後，作者提出的也僅是拖延 secret 被發現的時間，並沒有辦法去阻止 leakage。

Reflection:

1. 我認為這篇 paper 不容易寫深入，若是他將如何得到 secret 的方式寫得過於清楚，會讓更多攻擊者得以用有效率的方式去獲得這些 API keys。因此如果是我，會希望研究方向能轉成該如何去預防或是解決這樣的問題。
2. 雖然作者基於一些 limitation, 沒辦法再更深入去得知使用者的狀況以及這些 keys 的實際危害程度，但是這些數據、技術與研究結果還是能讓大家注意到 GitHub 很大的安全漏洞，且大範圍的分析資料是不容易的，這些仍是我想肯定作者的地方。

Reference: M. Meli, M. R. McNiece, and B. Reaves. "How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories," in NDSS. 2019.