

# ECB 靜態工具之開發與開源專案之檢測

周聖詠, 鄭力維, 唐靖傑, 趙偉恆

**Abstract** —本研究提出一套針對 AES-ECB 模式誤用的偵測與利用工具, 專門應用於分析 Python 生態系中以 pycryptodome 為主的開源專案。該工具設計理念借鏡於 CRYLOGGER 框架, 整合靜態掃描與攻擊驗證機制, 能自動辨識並驗證加密模式誤用的潛在風險。我們的系統在實測中成功找出多個存在潛在風險的專案, 並驗證部分案例可被實際利用。然而, 漏洞偵測的語意理解與攻擊自動化仍有待進一步完善與擴充。

**keywords:** 資訊安全、AES-ECB、靜態分析、Python、密碼學誤用

## 1 INTRODUCTION

隨著資安威脅不斷升級, 加密技術已成為現代資訊系統中不可或缺的一環。從 Web 服務、行動應用到物聯網設備, 對稱式加密演算法(如 AES)廣泛應用於敏感資料的保護。然而, 錯誤的加密模式選擇卻可能導致加密機制形同虛設, 其中最典型的例子便是 ECB(Electronic Codebook)模式的誤用。

AES-ECB 模式因為「相同明文區塊會被加密成相同密文區塊」的特性, 無法隱藏資料結構, 極易遭受密文重組、統計分析, 甚至偽造攻擊。因此, 它早在密碼學實務中被視為不安全模式, 國際間多數安全標準與套件(如 OpenSSL、NIST)皆建議禁止使用。然而, 在實際開發情境中, 這種模式仍常被誤用, 尤其是在 Python 環境下。

在 Python 中, pycryptodome 函式庫為常用的加密套件, 其提供的 AES.new(..., AES.MODE\_ECB) 接口由於使用簡單、範例眾多, 成為許多初學者與開源專案的首選。但這種「預設合法卻缺乏警告」的設計, 讓開發者在無意間將資料以 ECB 模式加密, 造成潛在的資訊洩露風險。

更嚴重的是, 多數開發者並無檢測自身專案加密安全性的意識, 也缺乏相關工具協助辨識此類誤用。市面上既有的靜態分析工具, 如 Bandit, 雖可掃描密碼 API 使用情形, 但多聚焦於高風險函式或硬編碼金鑰, 對於密碼模式選擇與上下文風險辨識能力有限。此外, 這些工具多半缺乏大規模自動爬取與分析 GitHub 專案的能力, 也無法針對已發現的誤用進行攻擊驗證或修復建議。

為解決上述問題, 本研究提出一套針對 Python 生態系統設計的ECB漏洞偵測與驗證工具, 專門針對 pycryptodome 使用情境進行分析。我們的系統具備以下三大特性:

- (1) GitHub 專案爬取與靜態分析模組, 可自動擷取含有 AES-ECB 使用的程式碼, 並進行風險標記;
- (2) 攻擊模擬模組, 能針對疑似漏洞進行密文篡改測試, 驗證前端行為是否受到影響;
- (3) 初步修復建議模組, 協助開發者將 ECB 模式轉換為更安全的 CBC 或 GCM 模式, 提升專案安全性。

本研究不僅關注「是否使用不安全加密模式」, 更進一步分析其是否處於敏感上下文, 是否缺乏完整性驗證, 並實作模擬攻擊驗證可行性。我們希望透過這套工具, 提高開源社群對加密誤用問題的警覺, 並提供一套有效實用的檢測與修復框架, 協助開發者建立更安全的密碼使用習慣。

## 2 PROBLEM DEFINITION

本研究針對 PYTHON 生態系統中廣泛使用的加密套件 PYCRYPTODOME, 深入探討其中 AES-ECB 模式的誤用情形。儘管 ECB 模式早已在加密實務中被認定為不具備足夠安全性, 但其簡便的實作介面與範例仍使得許多開發者在未察覺風險的情況下廣泛採用。本研究旨在系統性地識別這類加密誤用行為, 並評估其是否構成實際安全風險。綜合觀察與初步實驗結果, 我們界定出三項關鍵問題:

## 2.1 AES-ECB 模式在開源專案中仍廣泛存在

儘管 ECB 模式因無法隱藏明文結構、易於密文重組與偽造等問題，在密碼學上早被視為不安全，但於實務應用中仍被大量誤用，尤以 PYTHON 專案中使用 PYCRYPTODOME 函式庫者最為常見。該套件允許使用者透過 `AES.new(key, AES.MODE_ECB)` 簡單建立加密物件，且無任何預設警告或替代建議，使初學者在未具備足夠背景知識下，容易直接套用不安全範例。

此現象反映出開源社群中對於加密模式安全性的認知不足，加上缺乏防呆機制與開發時期警示，導致錯誤的實作習慣延續至大量專案中而未被注意。

## 2.2 欠缺針對開源專案的自動化規模分析能力

目前主流的靜態分析工具雖能針對密碼學 API 使用情形進行掃描，但多半專注於單一專案或單一檔案的檢查，難以針對成千上萬個 GitHub 開源專案進行批次分析與誤用追蹤。此外，現有系統鮮少具備從「程式碼爬取 → 誤用辨識 → 風險分析 → 報告彙整」的完整流程，限制了誤用現象的統計研究與整體風險評估的可行性。

在 `PYCRYPTODOME` 這類輕量函式庫應用廣泛的背景下，若缺乏系統性且規模化的分析工具，將使潛藏其中的加密誤用無法被即時發現或回報，進而形成資訊安全管理上的「盲區」。

### 2.3 缺乏誤用案例的實測驗證機制

單純偵測 API 使用不足以說明安全性，本研究針對選定案例進行密文篡改與資料注入測試，驗證實際攻擊效果，強化偵測可信度。

即使透過靜態分析偵測到 AES-ECB 模式的使用，若無實際驗證其應用場景與密文結構，仍無法斷言該行為是否構成安全漏洞。例如，若加密資料為結構化帳號資訊或缺乏完整性驗證機制，攻擊者可能進行密文竄改或重放攻擊。為此，本研究除分析程式碼外，亦針對特定高風險案例進行實測操作，模擬密文偽造、前端欺騙與訊息重組等攻擊場景，證實 ECB 誤用在特定條件下確實可被利用。

### 3 SYSTEM DESIGN

我們的研究設計並實作了一套自動化ECB加密誤用檢測與分析工具，結合BANDIT等靜態分析工具與自動化爬蟲，能針對網站原始碼、API及公開程式庫進行大規模弱點搜尋。工具設計強調以下特點：

3.1靜態分析:利用BANDIT、SEMGREP等工具,快速檢查常見加密誤用模式。

### 3.2 自動化爬取:整合GitHub爬蟲,自動擷取目標專案進行批次掃描。

3.3弱點回報:針對發現的高風險加密誤用,產生詳細報告,協助後續漏洞利用與修補。

## 4 RESULTS

#### 4.1 成功發現多個實際GITHUB專案存在AES-ECB加密誤用的嚴重漏洞

[illegible]

圖一.經由工具搜尋所得之漏洞專案列表

包含密鑰直接包含於密文中(如以明文資料前16位作為金鑰)、缺乏完整性驗證(無MAC)、前端完全信任解密結果、網站未啟用HSTS等基本安全疏失，導致密文可被偽造與竄改，且傳輸過程易受中間人攻擊。

#### 4.2自動化工具偵測效果良好, 但漏洞利用 (EXPLOIT) 功能仍待加強

以BANDIT等工具自動掃描原始碼與API，能有效偵測加密誤用，但模擬攻擊與自動化利用尚未完全實作，EXPLOIT部分有進一步提升空間。

#### 4.3 攻擊者可利用現有漏洞進行多種攻擊

利用已知漏洞可偽造密文，竄改用戶資料或消費紀錄，進行假公告、假登入等社交工程攻擊。因無HSTS與完整性驗證，系統易遭中間人攻擊與降級攻擊（如SSLSTRIP）。

4.4 網站多個 API 均採用相同不安全加密方式，風險範圍廣泛。一旦攻破一處，即可橫向移動至其他 API，造成更大資安威脅。

#### 4.5 專案已建立自動化漏洞搜尋與檢查流程

結合 BANDIT 掃描與自動爬取 GitHub 等技術，提升大規模檢測效率，能快速發現潛在高風險目標。

#### 4.6 深入個案分析發現複合性弱點

某專案以資料本身前 16 位元作為 ECB 加密金鑰，等同於金鑰明文附於密文中，極易被破解。該系統未啟用 HSTS，導致初次連線易受降級攻擊與 SSL STRIP 攔截。網站回傳的 JSON 資料中嵌有 HTML 字串，若前端未妥善處理，還可能導致 XSS 攻擊。整體而言，該網站加密方式錯誤、金鑰管理薄弱，且缺乏傳輸層與應用層的防護機制，暴露於多種攻擊風險之下。

```
def decrypt_aes_ecb(encrypted_data: str) -> str:
    key = encrypted_data[:16].encode('utf-8')
    encrypted_data = encrypted_data[16:]
    encrypted_data_bytes = base64.b64decode(encrypted_data)

    cipher = AES.new(key, AES.MODE_ECB)

    decrypted_data = unpad(cipher.decrypt(encrypted_data_bytes), AES.block_size)

    return decrypted_data.decode('utf-8')
```

圖二 專案程式碼之漏洞

## 5 EVALUATION

5.1 工具效能：在多個真實網站案例中，工具能自動化發現高危險等級的 ECB 誤用問題，並能針對 API 與前後端互動進行弱點追蹤。

5.2 實務影響：透過自動化流程，能協助資安團隊快速定位問題並提出修正建議，減少人工檢查負擔。

5.3 限制：本研究雖已初步建構自動化密碼誤用偵測與攻擊模擬系統，惟目前仍存在多項限制。首先，EXPLOIT 模組尚不完善，目前僅支援針對 ECB 模式之選擇明文攻擊模擬，尚未涵蓋如 CBC 模式缺 IV、弱金鑰來源等其他常見誤用情境。且模擬過程仍需使用者手動提供明文樣本，未能完全自動化。

其次，本系統僅具備誤用偵測功能，尚無法提供自動修改或修補建議，需由開發者手動查閱與修正。此外，系統目前僅針對 PYTHON 程式碼進行分析，尚未支援 JAVASCRIPT、JAVA、PHP 等常見語言，導致工具應用範圍受限。

在資料來源方面，由於本研究以 GitHub 為主要掃描平台，實務中常出現偵測結果來自未維護之殭屍專案，導致命中率偏低、實用性不佳。

## 6 RELATED WORK

許多研究致力於透過靜態或動態分析偵測密碼學誤用 (CRYPTOGRAPHIC MISUSE)。EGELE 等人 (2013) 針對 11,748 個 ANDROID 應用進行大規模靜態分析，發現其中約 88% 存在至少一項密碼誤用行為，例如使用不安全的 ECB 模式、硬編碼金鑰、或缺乏初始化向量 (IV)。此研究凸顯加密誤用在實務中普遍且嚴重的問題。

PICCOLBONI 等人 (2020) 則提出 CRYLOGGER，為首個可記錄加密 API 實際參數並執行合法性檢測的動態分析工具。與靜態工具 (如 CRYPTO GUARD) 搭配使用時，可顯著提升偵測範圍與精確度，展示靜動分析互補的潛力。

近年也有研究聚焦於靜態分析工具的偵測準確性與可用性。CHEN 等人 (2024) 針對多套主流工具進行實證比較，指出部分工具存在高誤警率與漏報問題，影響實務應用價值。

SCHLICHTIG 等人 (2022) 開發 CAMBENCH，為 JAVA 平台的密碼誤用偵測工具提供涵蓋範圍廣泛的基準資料集，包含真實專案與合成測試個案，便於工具間的客觀比較，亦可作為未來研究的共通測試平台。

在工具面，BANDIT 是針對 PYTHON 開發的輕量級靜態掃描工具，常被整合進 CI/CD 流程中，但對於加密使用情境的語意理解能力有限。SEMGREP 則是一款可擴充的規則導向工具，支援模式比對，但需使用者撰寫匹配邏輯。GITHUB 上亦有如 TABNEIB/ECBACP 的實作，能針對 ECB 模式執行選擇明文攻擊模擬，作為誤用偵測後續驗證的實務工具。此外，PYCRYPTODOME 為常用的 PYTHON 密碼學函式庫，儘管其 API 安全設計良好，但若使用不當 (如硬編碼金鑰、隨機生成不當)，仍可能導致安全風險。

## 7 CONCLUSION

本研究針對 Python 生態系統中 pycryptodome 套件的 AES-ECB 模式誤用問題，提出一套整合 GitHub 專案爬取、靜態掃描與攻擊驗證的自動化分析流程。我們基於 Bandit 靜態分析工具，成功掃描並定位多個實際存在安全風險的專案，並針對部分案例進行密文竄改與偽造實驗，驗證其可利用性。

本研究的主要貢獻包括：

- (1) 建立一套基於 Bandit 的 ECB 加密誤用偵測流程，應用於 GitHub 專案；
- (2) 彙整並驗證多個實際可利用的案例，證明誤用情形在實務中確實具備風險；
- (3) 提供初步分析框架，作為資安教學與未來工具擴充的基礎。

然而，本研究仍有若干限制。首先，雖然本系統結合了 Bandit 等靜態分析工具，具備一定程度的自動化檢測能力，但此類工具無法掌握密碼應用的語意脈絡，例如程式片段是否實際處理登入資訊、金流交易或其他安全敏感操作。此外，目前我們尚未實作自動修復模組，無法針對偵測出的潛在漏洞提供即時修正建議。

由於 GitHub 上部分程式庫為長期未維護的殭屍專案，本研究過程中所獲得的漏洞清單中命中率偏低，降低了實用性與實際影響力。針對此問題，未來可考慮將專案的star數、活躍度或commit 歷史等特徵納入篩選條件，以提高搜尋結果的品質與相關性。

另一方面，本系統目前僅針對 Python 語言進行分析，尚無法涵蓋如 JavaScript、Java、PHP 等常見語言，導致整體涵蓋範圍有限。而動態測試方面，由於各開源專案的建置與執行環境差異甚大，自動化測試流程實施困難，因此尚未納入評估。未來可進一步擴充搜尋範圍與漏洞類型，涵蓋除 ECB 外的更多常見密碼誤用情境，以提升分析的完整性與實用價值。

## 8 FUTURE WORK

本研究雖成功建立初步的偵測與驗證框架，但仍存在若干限制與未竟之處。未來工作可從以下幾個方向持續精進：

(1) 強化 **BANDIT** 的語意辨識能力：目前 BANDIT 難以辨識加密邏輯的語境背景，未來將透過開發 PLUGIN 增強其對加密資料敏感性(如帳號資訊、TOKEN、交易參數等)的判斷力。

(2) 跨語言與跨函式庫支援：目前工具僅支援 PYTHON + PYCRYPTODOME 的情境，未來將擴展至 JAVASCRIPT、JAVA、PHP 等語言，以及如 CRYPTOJS、BOUNCYCASTLE 等常用函式庫。

(3) 自動化修復與建議模組：針對已偵測的 ECB 誤用，加入建議替代方案(如 GCM/CBC)與重構範例，輔助開發者落實修正。

(4) 建立視覺化互動介面與報告功能：未來將提供統一分析儀表板與互動報表格式，利於企業資安部門追蹤並通報風險狀況。

(5) 提升動態測試的自動化能力：透過 SANDBOX 技術或模擬執行環境，提高對誤用案例實測的涵蓋率。

(6) 增加開發模型的通用性：我們的偵測技術不應僅侷限於 ECB 模式。可以擴展為通用型密碼學安全偵測平台，涵蓋所有已知有缺陷的加密模式(如 RC4、DES 等)及弱雜湊函數(如 MD5、SHA1)。

```
Metrics:
Total lines of code: 26
Total lines skipped (fnoscc): 0

blacklist: The pyCrypto library and its module AES are no longer actively maintained and have been deprecated. Consider using pycryptodome library.
Test ID: B413
Severity: HIGH
Confidence: HIGH
CWE: CWE-327
File: /Users/.../Projects/CNS_Final/SaadAhia_FilelessPELoader/SaadAhia_FilelessPELoader/aes.py
Line number: 2
More info: https://bandit.readthedocs.io/en/1.8.3/blacklists/blacklist_imports.html#b413-import-cryptocrypto

1 import sys
2 from Crypto.Cipher import AES
3 from Crypto.Util.Padding import pad

blacklist: The pyCrypto library and its module pad are no longer actively maintained and have been deprecated. Consider using pycryptodome library.
Test ID: B413
Severity: HIGH
Confidence: HIGH
CWE: CWE-327
File: /Users/.../Projects/CNS_Final/SaadAhia_FilelessPELoader/SaadAhia_FilelessPELoader/aes.py
Line number: 3
More info: https://bandit.readthedocs.io/en/1.8.3/blacklists/blacklist_imports.html#b413-import-cryptocrypto

2 from Crypto.Cipher import AES
3 from Crypto.Util.Padding import pad
4 from os import urandom
```

圖三 初步檢查設計使用不安全的 ECB 模式、缺 IV 的 CBC、弱隨機鑰與硬編碼金鑰

(7) 提高偵查結果之有效性：我們的偵測結果時常充斥著各式密碼學教學用專案或是已停止更新多年的殭屍專案。我們可以以各式filter去篩選能夠有效提高我們真正找到正在運行的漏洞，藉此提高工具可用性。

```
Vincent-G-Van/AES-Encryption-Python ★ 26
Two scripts in Python to encrypt/decrypt using the 128 bits AES algorithm, ECB mode with hex "00" as padding for each character. For the encryption, an ascii plaintext file is taken as the input, then an encrypted hex file is outputted. For the decryption, a ciphertext hex file is taken as the input, then a decrypted ascii file is outputted.
```

#### 圖四 以星號排序之搜尋結果

整體而言, 本研究展示了靜態分析與攻擊驗證工具在開源安全檢測中的應用潛力, 未來若能持續深化其語意理解與使用情境適配, 將有望成為資安研究與開發實務中具實用價值的利器。

#### REFERENCES

- [1] [用 Bandit 靜態掃描工具, 掃描 Python 專案中的安全性問題-MyApollo](#)
- [2] [Quickstart | Semgrep](#)
- [3] [tabneib/ecbacp: Simple tool for ECB adaptive chosen plaintext attack.](#)
- [4] PythonBanditDocumentation.  
<https://bandit.readthedocs.io>
- [5] pycryptodome Official Docs  
<https://pycryptodome.readthedocs.io>
- [6] 專案:<https://github.com/leverimmy/THU-Annual-Eat>
- [7] Egele, M., Brumley, D., Fratantonio, Y., & Kruegel, C. (2013). An empirical study of cryptographic misuse in Android applications. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security
- [8] Piccolboni, A., Backes, M., Marino, D., & Bugiel, S. (2020). CryLogger: Detecting Crypto Misuses Dynamically. In Proceedings of the 29th USENIX Security Symposium.
- [9] Chen, C.-Y., Chen, Y., Zhang, Y., et al. (2024). Revisiting the Use and Misuse of Static Analysis Tools for Detecting Cryptographic Misuse. arXiv preprint arXiv:2404.01518.
- [10] Schlichtig, M., Deubzer, M., Hermann, B., & Bodden, E. (2022). CamBench: A Benchmark Suite for API-Misuse Detectors in Java Cryptography. In Proceedings of the ACM/IEEE 45th International Conference on Software Engineering.
- [11] Bandit. (n.d.). Python Bandit Documentation. <https://bandit.readthedocs.io>
- [12] Semgrep. (n.d.). Semgrep Quickstart. <https://semgrep.dev/docs/quickstart/>
- [13] tabneib. (2023). ecbacp: A Simple Tool for ECB Adaptive Chosen Plaintext Attack. GitHub. <https://github.com/tabneib/ecbacp>
- [14] PyCryptodome. (n.d.). PyCryptodome Documentation. <https://pycryptodome.readthedocs.io>