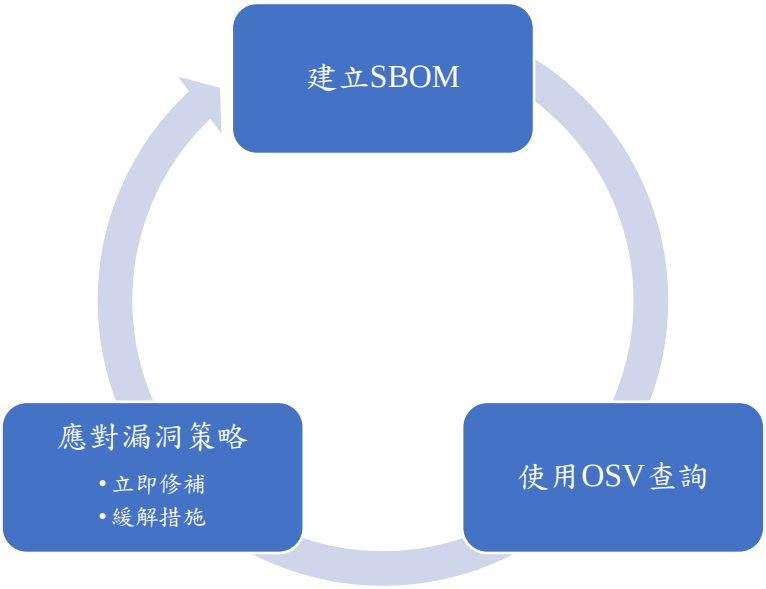
# SBOM 開源工具使用說明

SBOM (Software Bill of Materials): SBOM 提供軟體中包含的所有函式庫與組件的清單，還會列出版本號、原始碼來源和關聯資訊。SBOM 的目的是讓使用者或開發者瞭解軟體的組成，這樣在發生安全事件或需要更新時，能夠快速識別和應對。

OSV (Open Source Vulnerability): 是一種公開的軟體漏洞資料庫。透過產生的 SBOM 資訊搭配 OSV 漏洞資料庫，開發人員可以快速查詢其軟體組件是否存在已知的安全問題。

以下是使用 SBOM 與 OSV 的基本流程：

1. 建立 SBOM: 於系統維運時，定期利用 SBOM 工具建立一份 SBOM 文件，將所有的組件、函式庫和依賴性記錄下來。
2. 使用 OSV 查詢: 定期將產生的 SBOM 資訊，利用 OSV 資料庫查詢，確認組件是否有已知的安全漏洞。一旦發現，可以迅速決策，例如進行更新或替換該組件。
3. 應對漏洞策略: 如果在 OSV 中發現了任何漏洞，應該立即進行修補或緩解措施。這可能包括更新軟體組件、應用修補程式，或在使用中考慮其他安全措施。



此操作文件使用的 SBOM 工具有以下兩種：

- Microsoft sbom-tool
- CycloneDX Generator

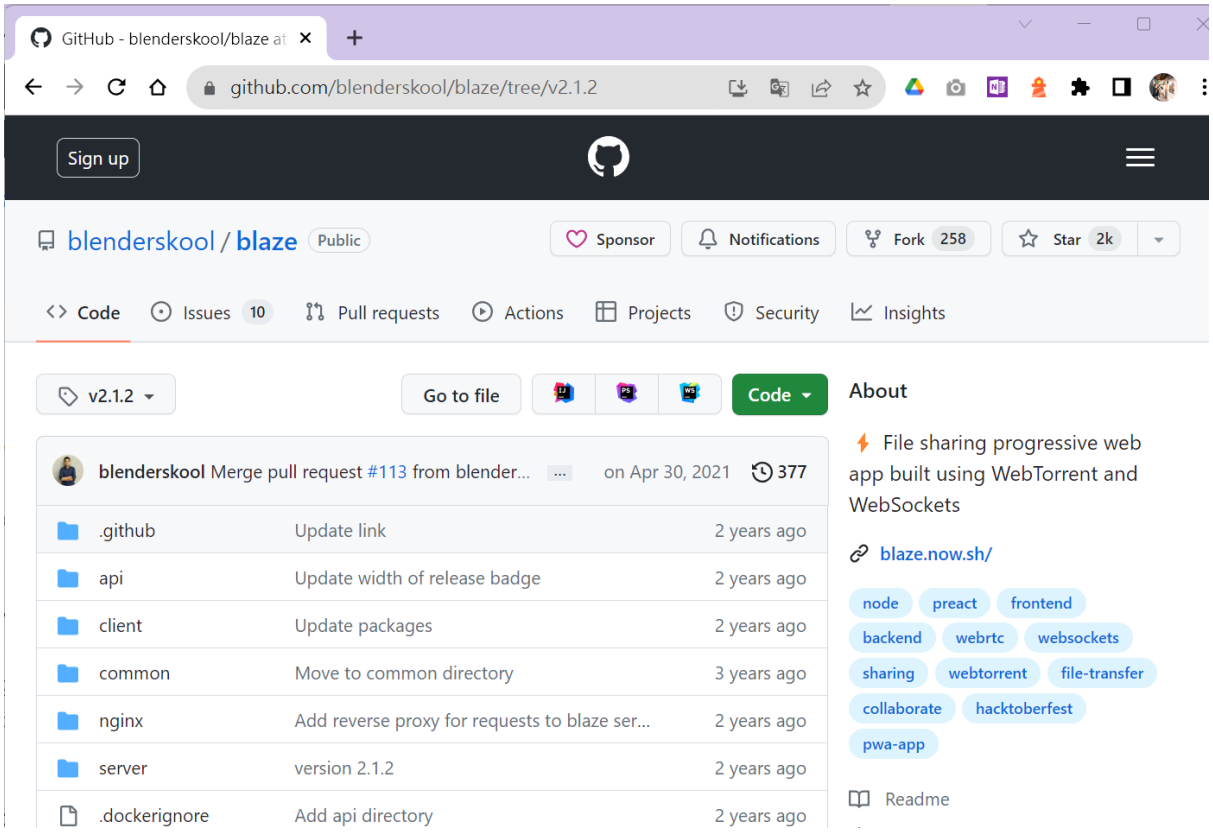
並搭配 Google osv-scanner 掃描產生的 SBOM 檔案找出元件漏洞，進行修補或緩解措施。

依照相容性測試建議 Windows 環境可以使用 Microsoft sbom-tool，Linux 環境可以使用 Microsoft sbom-tool 與 CycloneDX Generator

本操作文件將使用 Windows 10 與 Ubuntu Desktop 22.04 LTS 進行以下測試

掃描 Github 開源專案 blaze 版本 2.1.2

網址：https://github.com/blenderskool/blaze/tree/v2.1.2



# Microsoft sbom-tool 開源工具說明

Microsoft sbom-tool 開源專案，工具支援目前主流的 Package Managers，依照 Package Managers 與語言關係對應整理如下表：

| **語言** | **支援的 Package Managers** |
|---|---|
| go | GoMod |
| Rust | Cargo |
| .NET | NuGet |
| Java | Maven、Gradle |
| Node.js | NPM 、Yarn |
| Python | PIP 、Poety |
| Ruby | Gems |
| Objective-C<br>Swift | CocoaPods |

資料彙整：https://github.com/microsoft/component-detection

● Windows10 環境操作步驟

步驟一、安裝 winget

藉由微軟商店安裝 winget

相關說明：https://learn.microsoft.com/zh-tw/windows/package-manager/winget/

安裝方法：微軟商店安裝”應用程式安裝程式”詳見圖 1

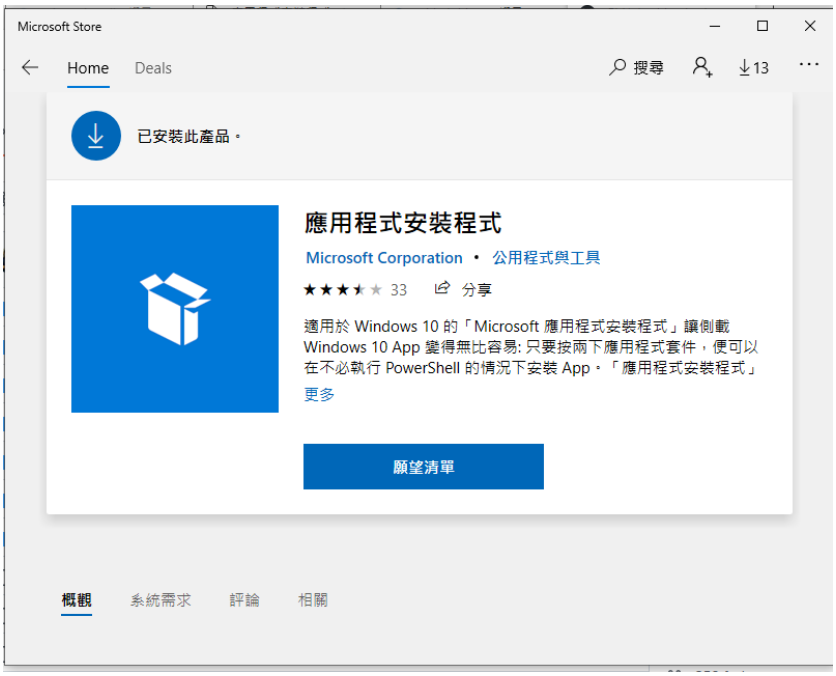



圖 1 微軟商店安裝 winget

步驟二、安裝 Microsoft sbom-tool 工具

開啟 Windows PowerShell 並輸入安裝指令

**(安裝完成後請登出使用者再登入，以完成環境變數設定)**

➢ winget install Microsoft.SbomTool

輸出畫面詳見圖 2

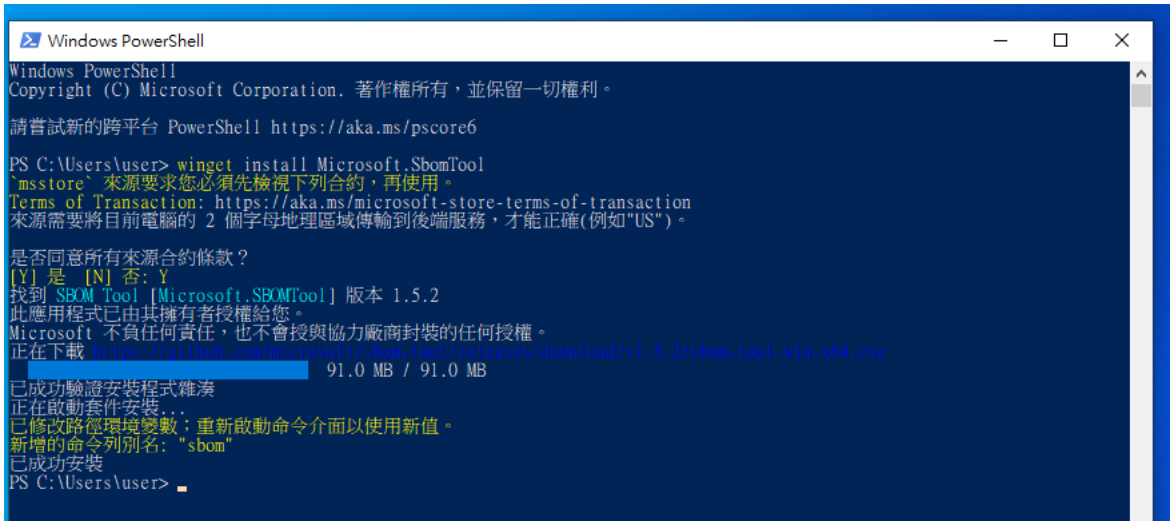



圖 2 安裝 Microsoft sbom-tool 工具指令結果畫面

步驟三、執行 Microsoft sbom-tool 掃描

3.1 將掃描原始檔資料夾”blaze-2.1.2”放置到”本機\文件”內

3.2 在”本機\文件”內建立資料夾”blaze-sbom”放置工具產生的 SBOM 檔案

完成後詳見圖 3

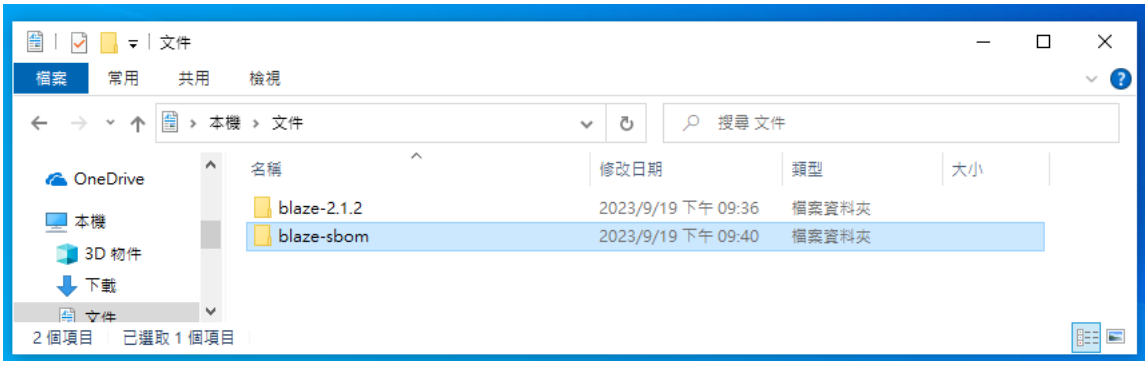



圖 3 本機\文件內的畫面

3.3 開啟 Windows PowerShell 輸入指令移動至”本機\文件”內

- cd C:\Users\user\Documents\

**(此處使用者名稱會隨著不同電腦而變化，請依照本機使用者路徑替換)**

3.4 開啟 Windows PowerShell 輸入指令執行 sbom-tool-win-x64.exe

- sbom-tool-win-x64.exe generate -b "./blaze-sbom" -bc "./blaze-2.1.2" -pn "blaze" -pv "v1" -ps "nics" -D "true"

參數說明：

-b "產生 SBOM 檔案放置目錄位置"

-bc "進行 SBOM 掃描的原始程式碼目錄位置"

-pn "SBOM 產生專案名稱"

-pv "版本號"

-ps "SBOM 產生單位"

-D "設置 true 將刪除目錄內已產生過的 SBOM 檔案"

完成後輸出畫面詳見圖 4

```
Windows PowerShell
PS C:\Users\user> cd C:\Users\user\Documents\
PS C:\Users\user\Documents> sbom-tool-win-x64.exe generate -b "./blaze-sbom" -bc "./blaze-2.1.2" -pn "blaze" -pv "v1" -ps "nics" -D "true"
##[information]Log file: "C:\\Users\\user\\AppData\\Local\\Temp\\GovCompDisc_Log_20230919214435674_4656.log"
##[information]Run correlation id: f7dd380a-66c1-409f-b742-13bf692f459a
##[information]Finding components...
##[information]
##[information]
##[information]|Component Detector Id        |Detection Time              |# Components Found          |# Explicitly Referenced     |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|CocoaPods                   |0.063 seconds               |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Go                          |0.062 seconds               |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Gradle                      |0.063 seconds               |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Ivy (Beta)                  |0.34 seconds                |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Linux                       |0.21 seconds                |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|MvnCli                      |0.8 seconds                 |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Npm                         |0.33 seconds                |4                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|NpmLockfile3 (Beta)         |0.38 seconds                |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|NpmWithRoots                |0.8 seconds                 |1346                        |29                          |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|NuGet                       |0.39 seconds                |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|NuGetPackagesConfig         |0.41 seconds                |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|NuGetProjectCentric         |0.41 seconds                |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Pip                         |0.97 seconds                |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Pnpm                        |0.8 seconds                 |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Poetry (Beta)               |0.8 seconds                 |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Ruby                        |0.8 seconds                 |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|RustCrateDetector           |0.8 seconds                 |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|SPDX22SBOM                  |0.057 seconds               |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Vcpkg (Beta)                |0.8 seconds                 |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Yarn                        |0.06 seconds                |0                           |0                           |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]|Total                       |0.98 seconds                |1350                        |29                          |
##[information]|____________________________|____________________________|____________________________|____________________________|
##[information]
##[information]Detection time: 0.9835563 seconds.
##[information]Scan Manifest file: "C:\\Users\\user\\AppData\\Local\\Temp\\ScanManifest_20230919214435539.json"
PS C:\Users\user\Documents>
```

圖 4 執行 sbom-tool-win-x64 指令結果畫面

步驟四、查看 SBOM 檔案

4.1 開啟”本機\文件\blaze-sbom_manifest\spdx_2.2\” 目錄，找到 manifest.spdx.json 檔案詳見圖 5

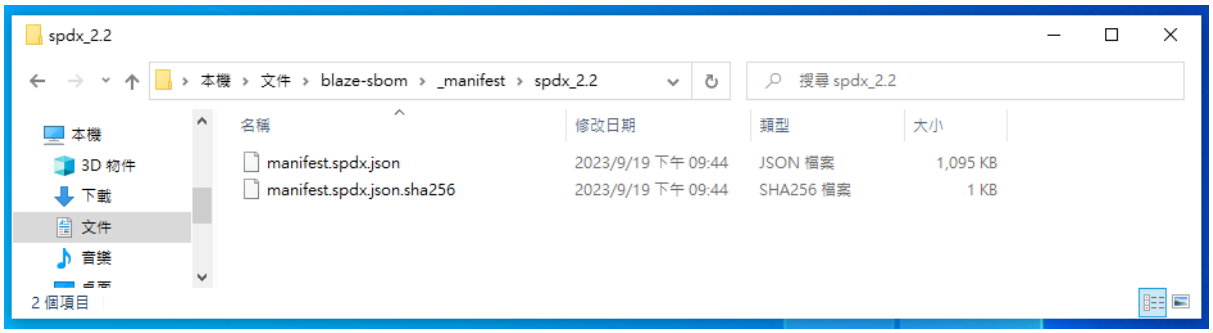



圖 5 manifest.spdx.json 檔案位置

4.2 開啟 manifest.spdx.json 檔案後，由 packages 查看元件版本詳見圖 6

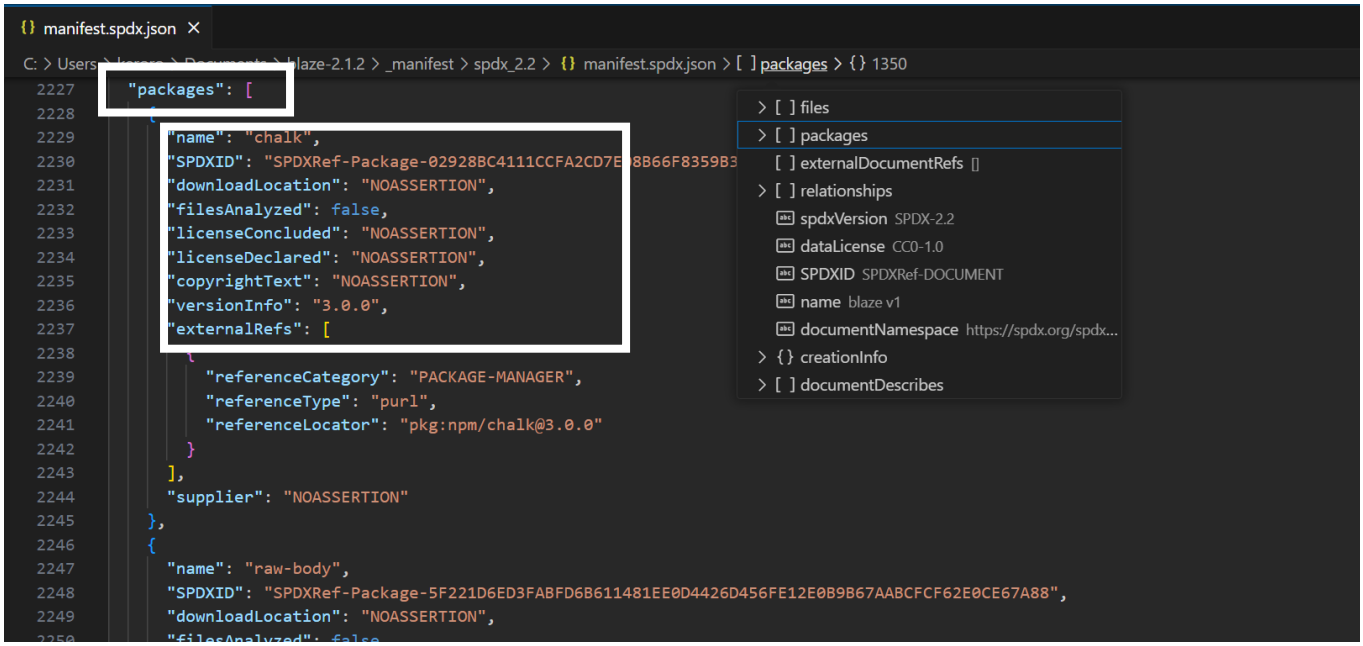



圖 6 manifest.spdx.json 查看元件版本

4.3 開啟 manifest.spdx.json 檔案後，由 relationships 查看元件關係詳見圖 7

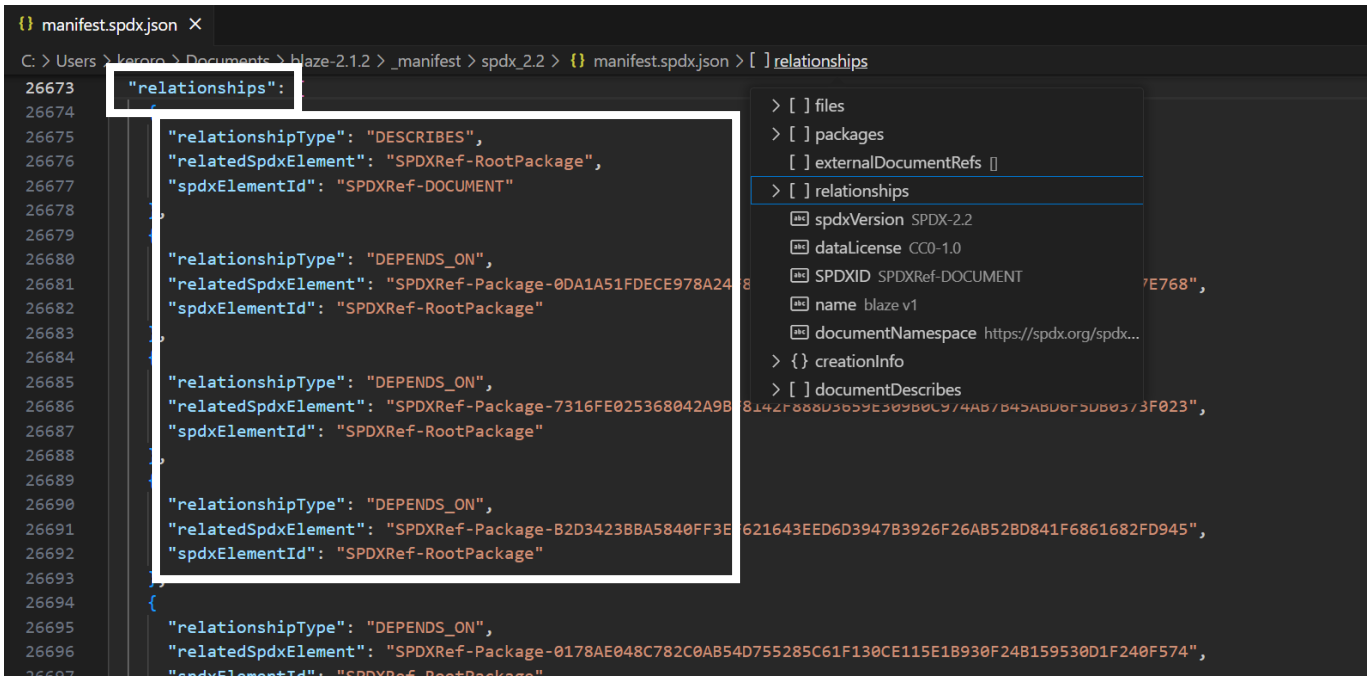



圖 7 manifest.spdx.json 查看元件關係

步驟五、安裝 scoop

開啟 Windows PowerShell 輸入安裝指令

- Set-ExecutionPolicy RemoteSigned -scope CurrentUser
- iwr -useb get.scoop.sh | iex

完成後輸出畫面詳見圖 8

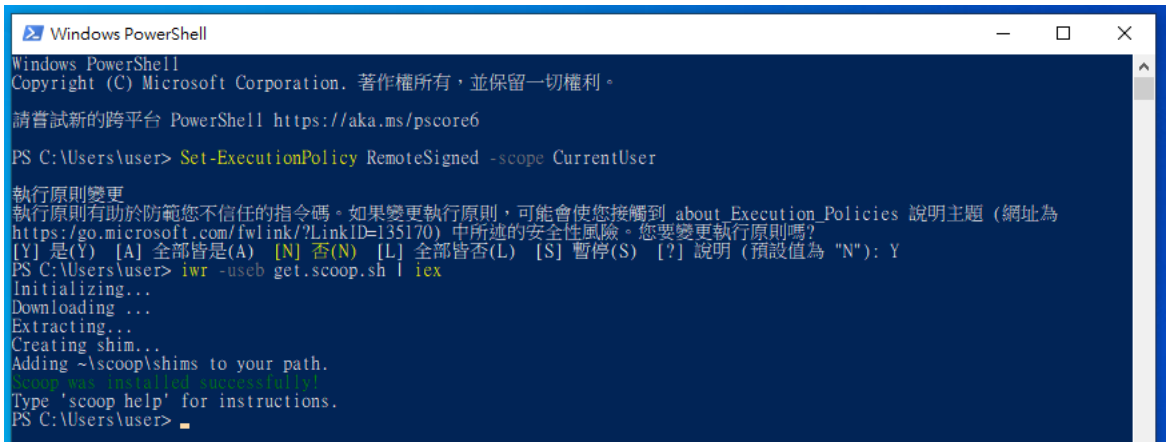



圖 8 安裝 scoop 指令結果畫面

步驟六、安裝 osv-scanner 工具進行

開啟 Windows PowerShell 輸入安裝指令

- scoop install osv-scanner

完成後輸出畫面詳見圖 9

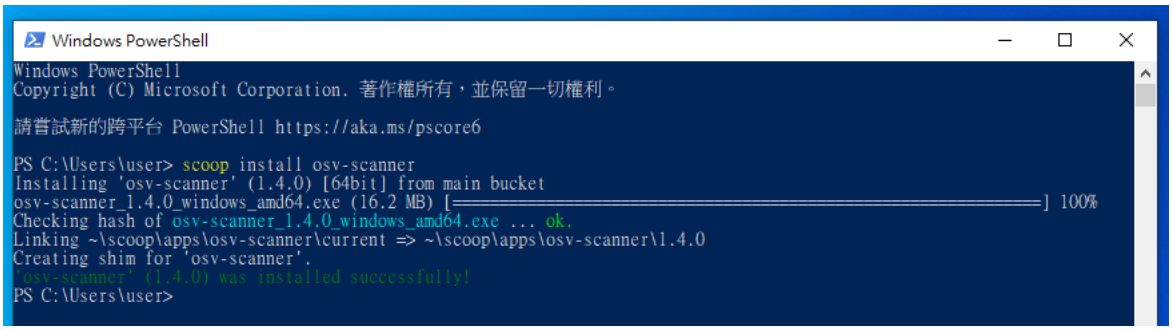



圖 9 安裝 osv-scanner 指令結果畫面

步驟七、執行 osv-scanner 工具掃描產出 json 檔，進行後續應對漏洞策略

7.1 開啟 Windows PowerShell 輸入指令移動至”本機\文件\blaze-sbom_manifest\spdx_2.2\” 目錄

➢ cd C:\Users\user\Documents\blaze-sbom_manifest\spdx_2.2

**(此處使用者名稱會隨著不同電腦而變化，請依照本機使用者路徑替換)**

7.2 開啟 Windows PowerShell 輸入指令執行 osv-scanner

➢ osv-scanner.exe --sbom="./manifest.spdx.json" --format json > "./file.json"

參數說明：

--sbom "SBOM 檔案完整位置"

--format json > "json 檔案輸出位置"

完成後輸出畫面詳見圖 10


圖 10 執行 osv-scanner 指令結果畫面

7.3 開啟”本機\文件\blaze-sbom_manifest\spdx_2.2\” 目錄，找到產生 file.json 檔案詳見圖 11

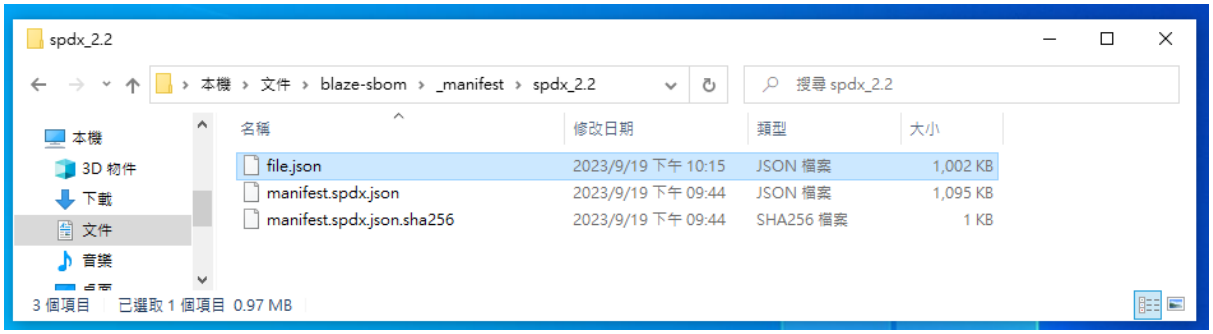



圖 11 file.json 檔案位置

步驟八、查看 file.json 檔案

開啟 file.json 檔案，文件詳細說明各元件弱點版本與 CVE 編號詳見圖 12，開始進行後續應對漏洞策略規劃

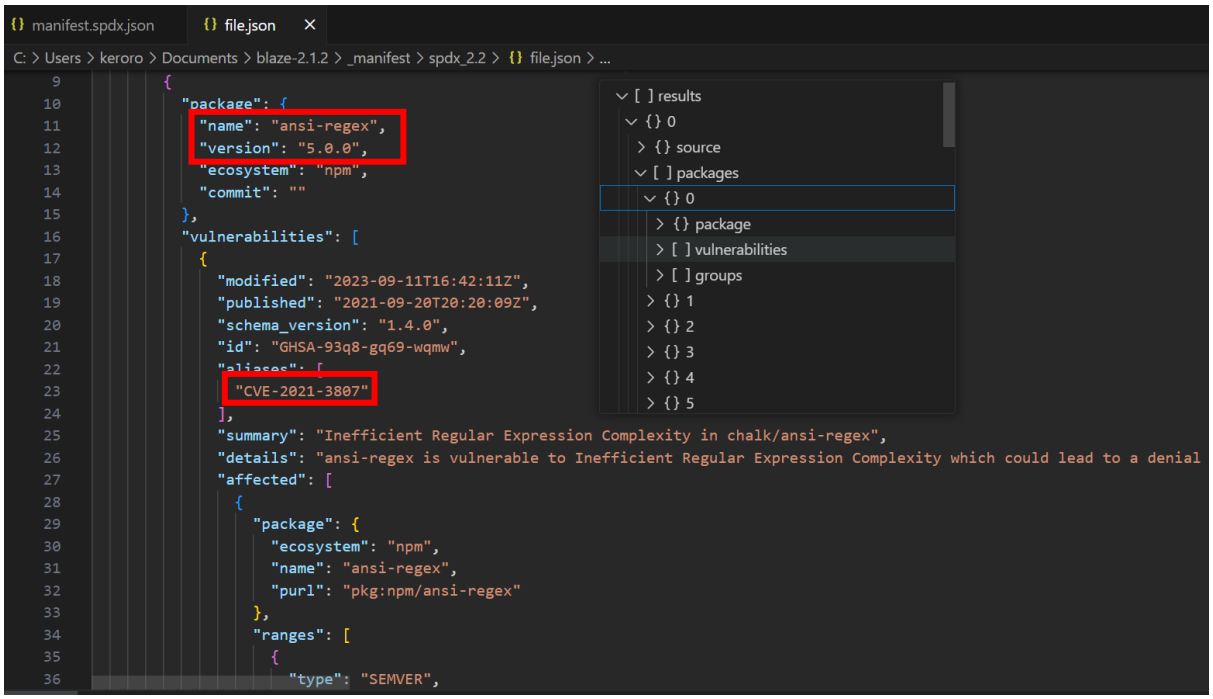



圖 12 file.json 檔案查看元件弱點版本與 CVE 編號

- Ubuntu Desktop 22.04 LTS 環境操作步驟

步驟一、安裝 curl

開啟 terminal 輸入安裝指令

➢ snap install curl

完成後輸出畫面詳見圖 13

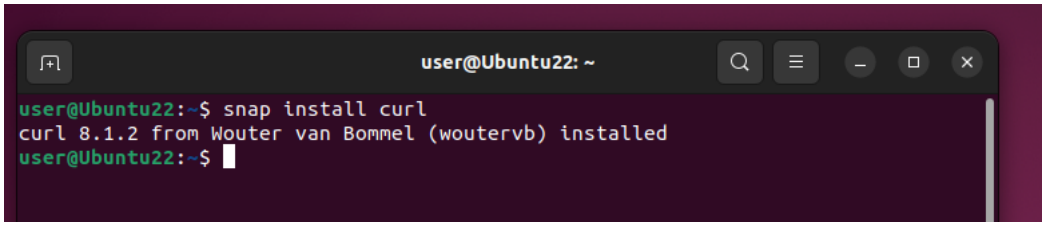



圖 13 安裝 curl 指令結果畫面

步驟二、下載 sbom-tool

2.1 開啟 terminal 輸入指令下載 sbom-tool

➢ curl -Lo sbom-tool https://github.com/microsoft/sbom-tool/releases/latest/download/sbom-tool-linux-x64

2.2 開啟 terminal 輸入指令設定 sbom-tool 權限

➢ chmod +x sbom-tool

2.3 開啟 terminal 輸入指令確認 sbom-tool 是否設定完成

➢ ls

完成後輸出畫面詳見圖 14

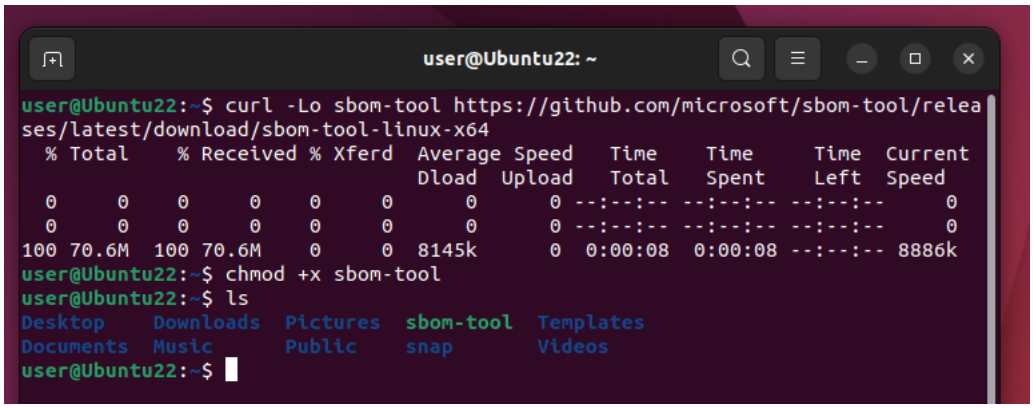



圖 14 下載 sbom-tool 並設定權限結果畫面

步驟三、執行 Microsoft sbom-tool 掃描

3.1 將下載的原始檔檔案按右鍵選擇”Extract to…”進行解壓縮 zip，詳見圖 15

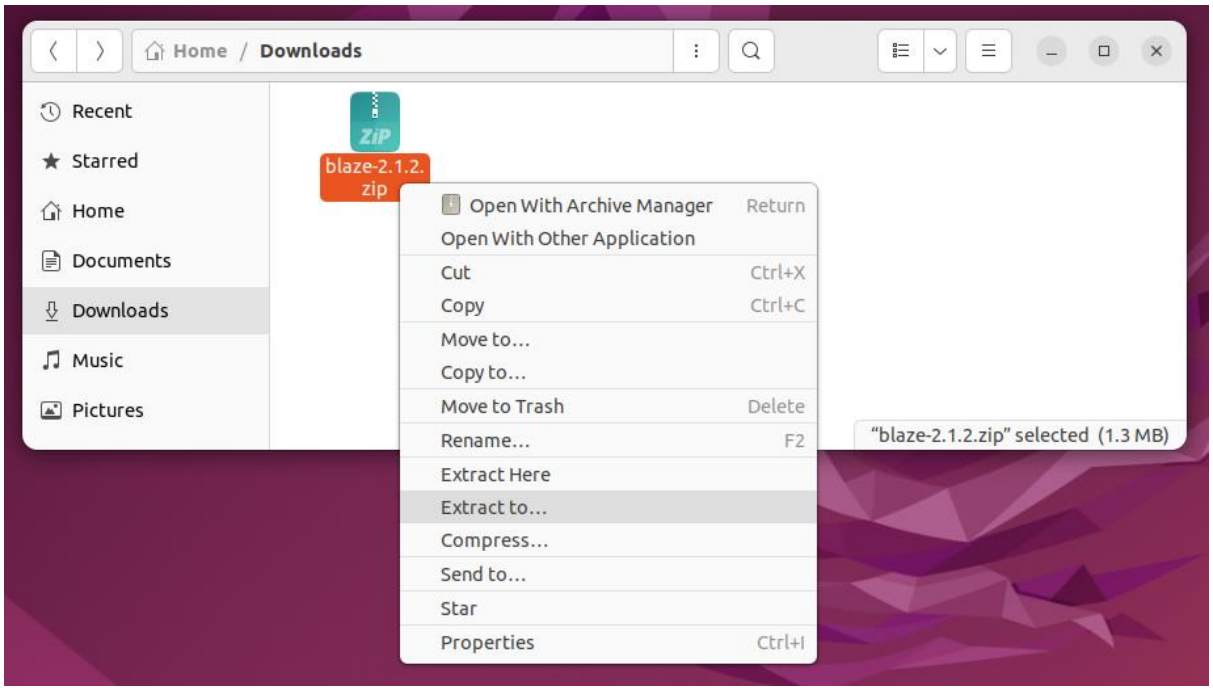



圖 15 對壓縮檔進行解壓縮操作(一)

3.2 選擇 Home 並按下 Select，詳見圖 16

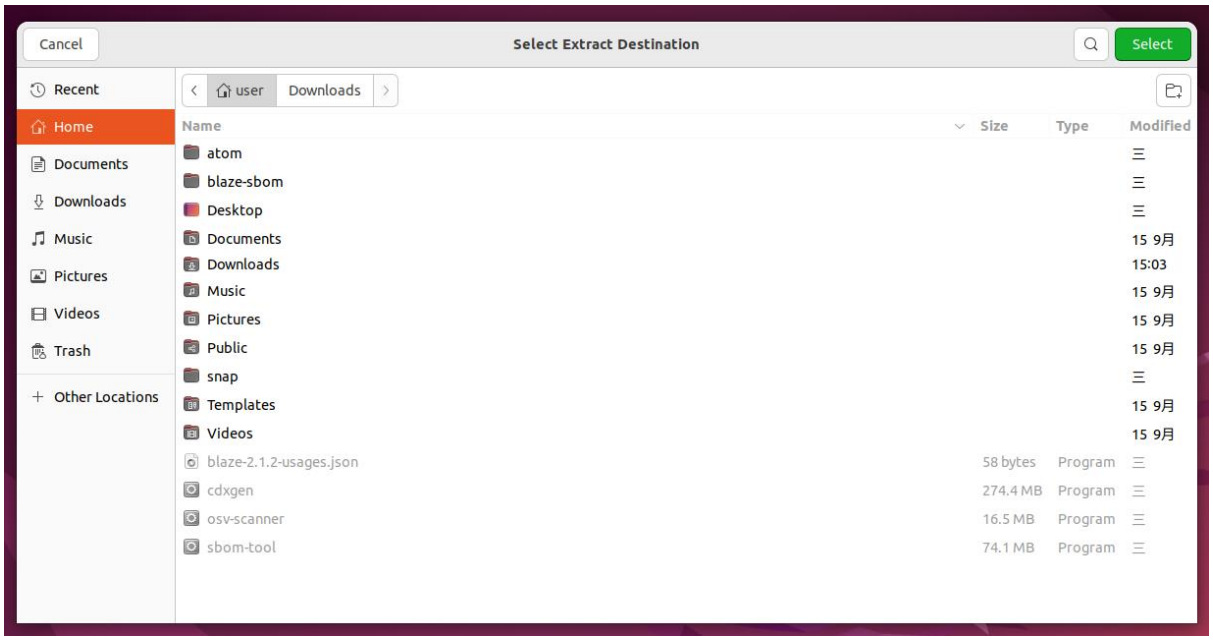



圖 16 對壓縮檔進行解壓縮操作(二)

3.3 於 Home 建立一個目錄”blaze-sbom”放置 SBOM 檔案

完成後詳見圖 17

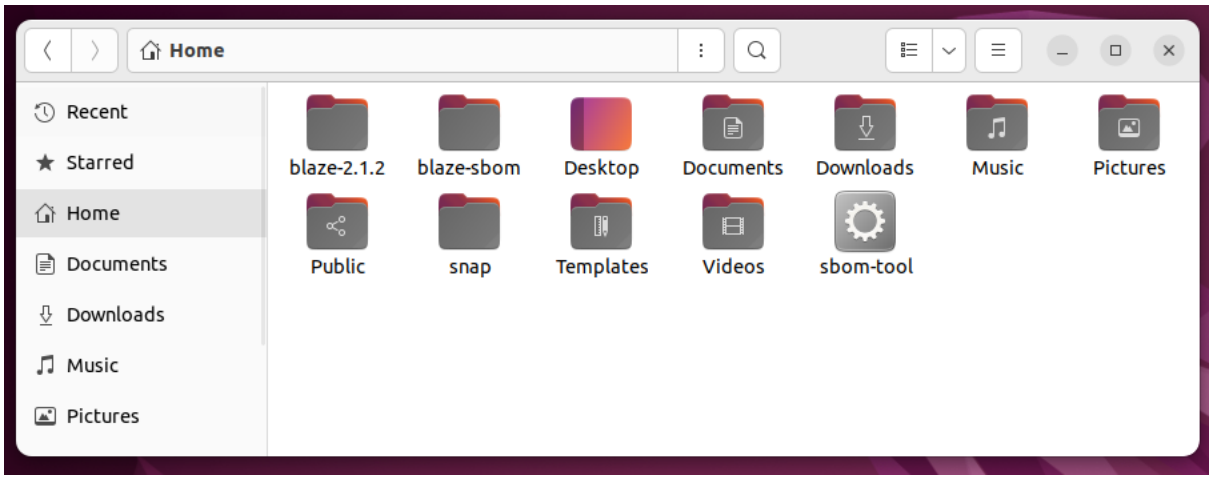



圖 17 放置原始碼與建立放置 SBOM 檔案的目錄

3.4 開啟 terminal 輸入執行 sbom-tool

➢ ./sbom-tool generate -b "./blaze-sbom" -bc "./blaze-2.1.2" -pn "blaze" -pv "v1" -ps "nics" -D "true"

參數說明：

-b "產生 SBOM 檔案放置的目錄位置"

-bc "進行 SBOM 掃描的原始程式碼目錄位置"

-pn "SBOM 產生專案名稱"

-pv "版本號"

-ps "SBOM 產生單位"

-D "設置 true 將刪除目錄內已產生過的 SBOM 檔案"

完成後輸出畫面詳見圖 18 與圖 19

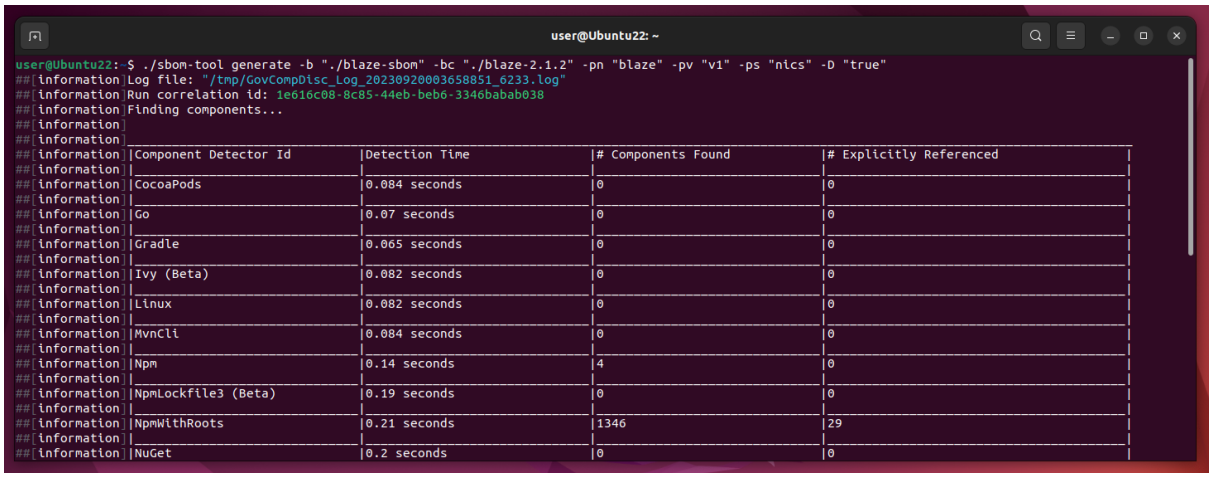



圖 18 執行 sbom-tool 指令結果畫面(上)

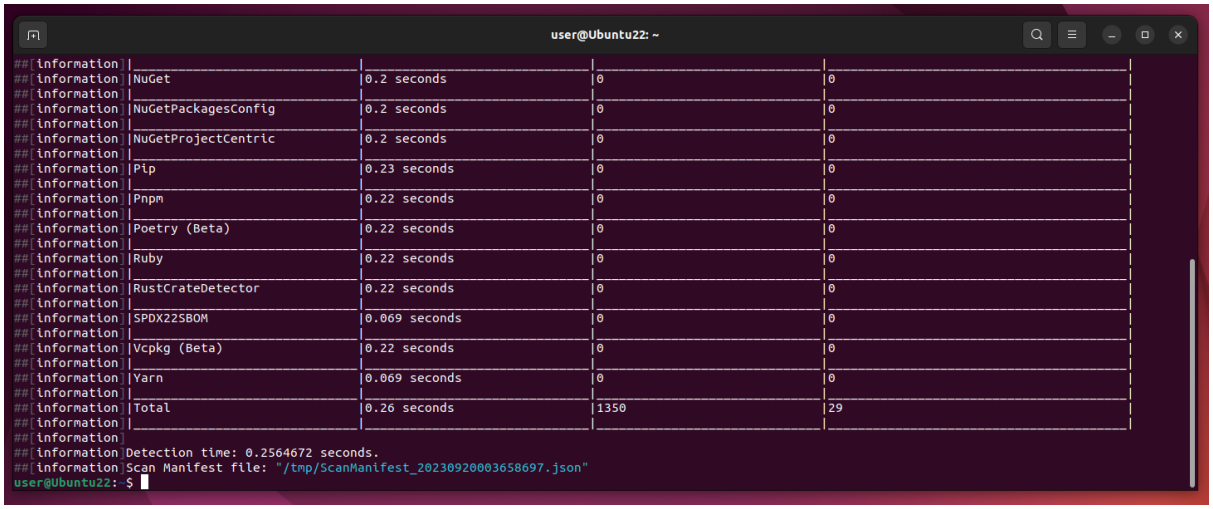



圖 19 執行 sbom-tool 指令結果畫面(下)

步驟四、查看 SBOM 檔案

4.1 開啟”/home/blaze-sbom/_manifest/spdx_2.2”目錄，找到 manifest.spdx.json 檔案詳見圖 20

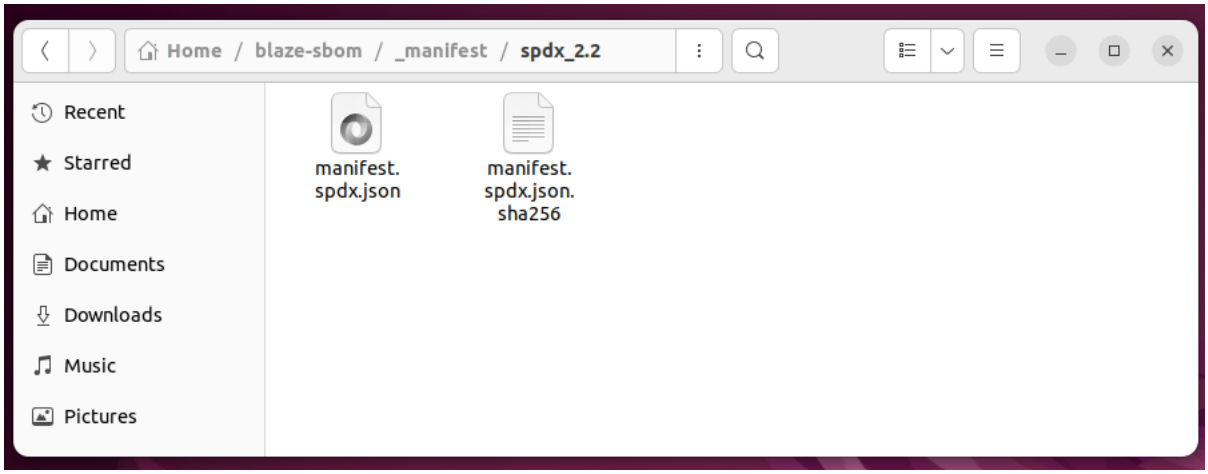
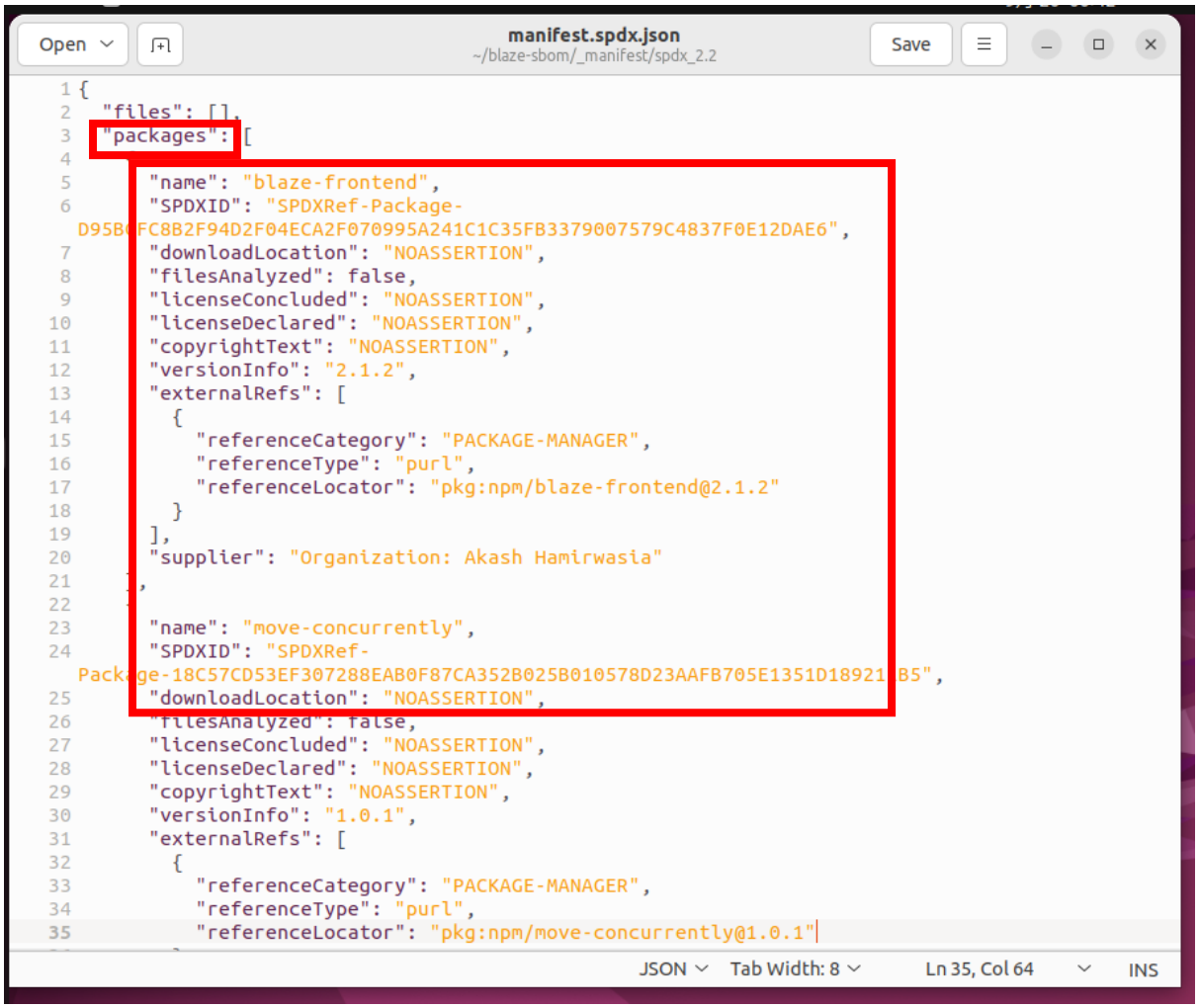



圖 20 manifest.spdx.json 檔案位置

4.2 開啟 manifest.spdx.json 檔案後，由 packages 查看元件版本詳見圖 21





```
{
  "files": [],
  "packages": [
    {
      "name": "blaze-frontend",
      "SPDXID": "SPDXRef-Package-D95B0FC8B2F94D2F04ECA2F070995A241C1C35FB3379007579C4837F0E12DAE6",
      "downloadLocation": "NOASSERTION",
      "filesAnalyzed": false,
      "licenseConcluded": "NOASSERTION",
      "licenseDeclared": "NOASSERTION",
      "copyrightText": "NOASSERTION",
      "versionInfo": "2.1.2",
      "externalRefs": [
        {
          "referenceCategory": "PACKAGE-MANAGER",
          "referenceType": "purl",
          "referenceLocator": "pkg:npm/blaze-frontend@2.1.2"
        }
      ],
      "supplier": "Organization: Akash Hamirwasia"
    },
    {
      "name": "move-concurrently",
      "SPDXID": "SPDXRef-Package-18C57CD53EF307288EAB0F87CA352B025B010578D23AAFB705E1351D18921AB5",
      "downloadLocation": "NOASSERTION",
      "filesAnalyzed": false,
      "licenseConcluded": "NOASSERTION",
      "licenseDeclared": "NOASSERTION",
      "copyrightText": "NOASSERTION",
      "versionInfo": "1.0.1",
      "externalRefs": [
        {
          "referenceCategory": "PACKAGE-MANAGER",
          "referenceType": "purl",
          "referenceLocator": "pkg:npm/move-concurrently@1.0.1"
```


圖 21 manifest.spdx.json 查看元件版本

4.3 開啟 manifest.spdx.json 檔案後，由 relationships 查看元件關係詳見圖 22

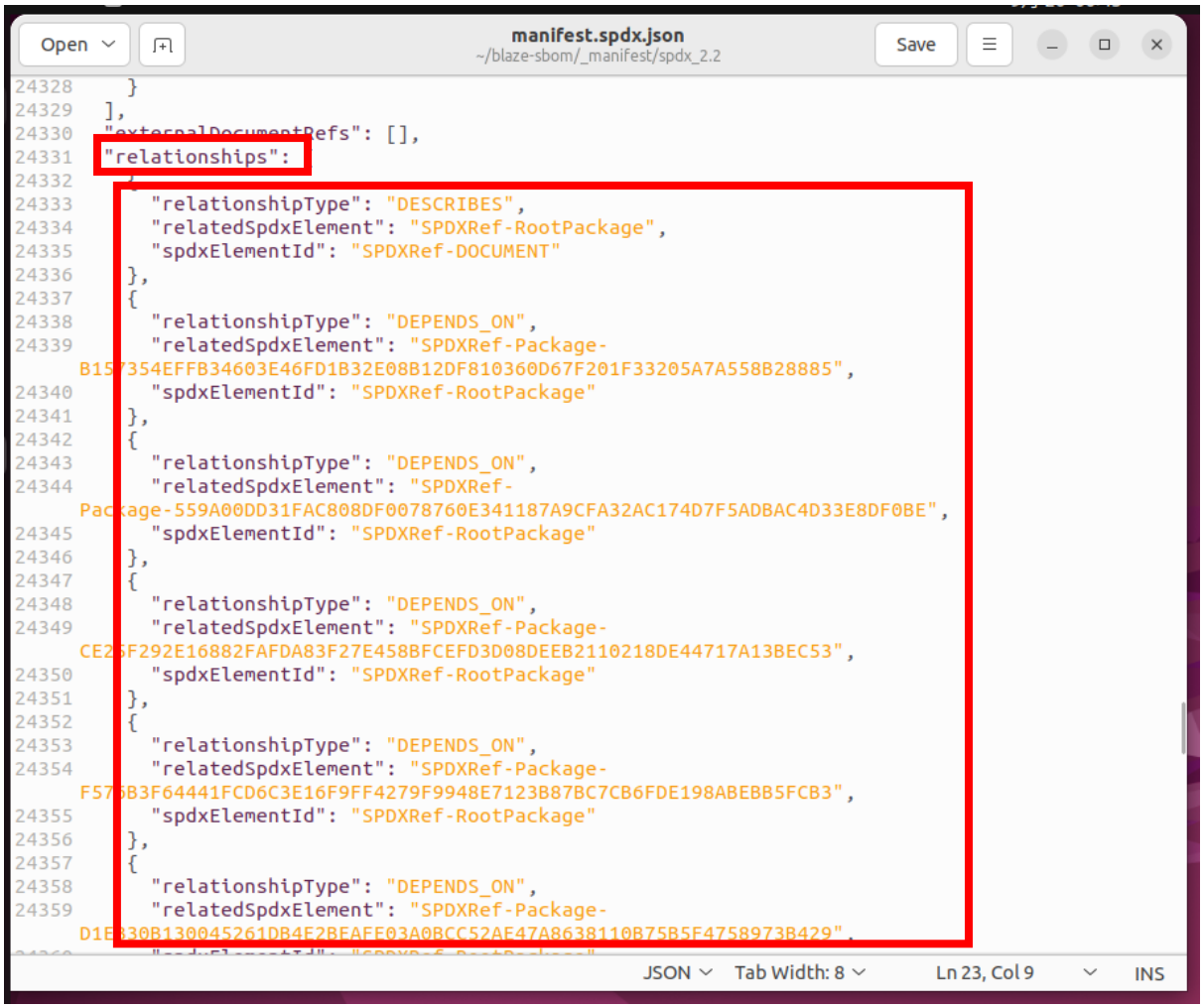



圖 22 manifest.spdx.json 查看元件關係

步驟五、下載 osv-scanner 工具

5.1 下載 osv-scanner 工具

https://github.com/google/osv-scanner/releases/latest/

以 v1.4.0 版為例

假設環境為 Linux 64 位元一般環境，找尋 linux-amd64 選擇：

https://github.com/google/osv-scanner/releases/download/v1.4.0/osv-scanner_1.4.0_linux_amd64

5.2 開啟 terminal 輸入指令下載 osv-scanner 工具

- curl -Lo osv-scanner https://github.com/google/osv-scanner/releases/download/v1.4.0/osv-scanner_1.4.0_linux_amd64

5.3 開啟 terminal 輸入指令設定 osv-scanner 工具權限

- chmod +x osv-scanner

完成後輸出畫面詳見圖 23

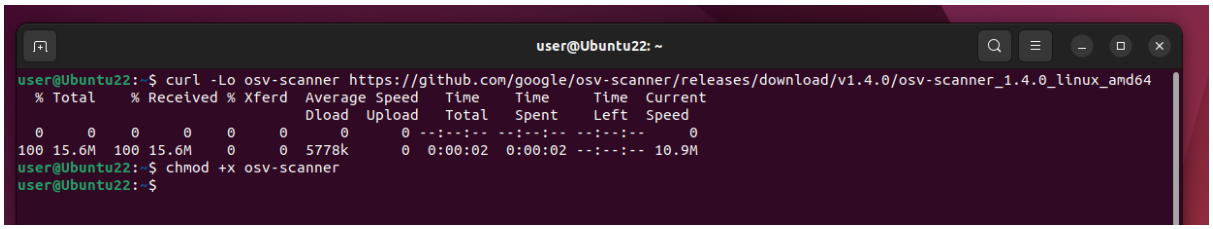



圖 23 下載 osv-scanner 工具與設定權限的指令結果畫面

步驟六、執行 osv-scanner 工具掃描產出 json 檔，進行後續應對漏洞策略

6.1 開啟 terminal 輸入指令執行 osv-scanner 工具

- ./osv-scanner --sbom="./blaze-sbom/_manifest/spdx_2.2/manifest.spdx.json" --format json > "./blaze-sbom/_manifest/spdx_2.2/file.json"

參數說明：

--sbom "SBOM 檔案位置"

--format json > "json 檔案輸出位置"

完成後輸出畫面詳見圖 24

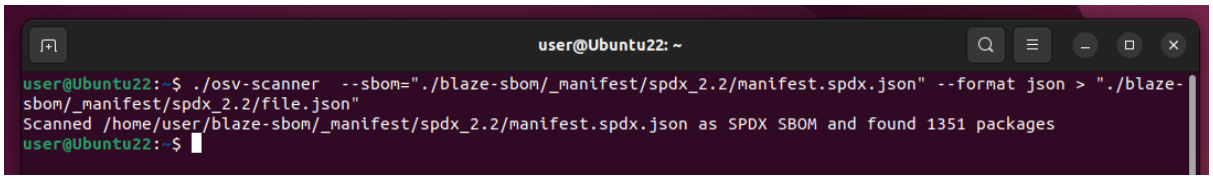



圖 24 執行 osv-scanner 工具指令結果畫面

6.2 開啟”/home/blaze-sbom/_manifest/spdx_2.2” 目錄，找到 file.json 檔案詳見圖 25

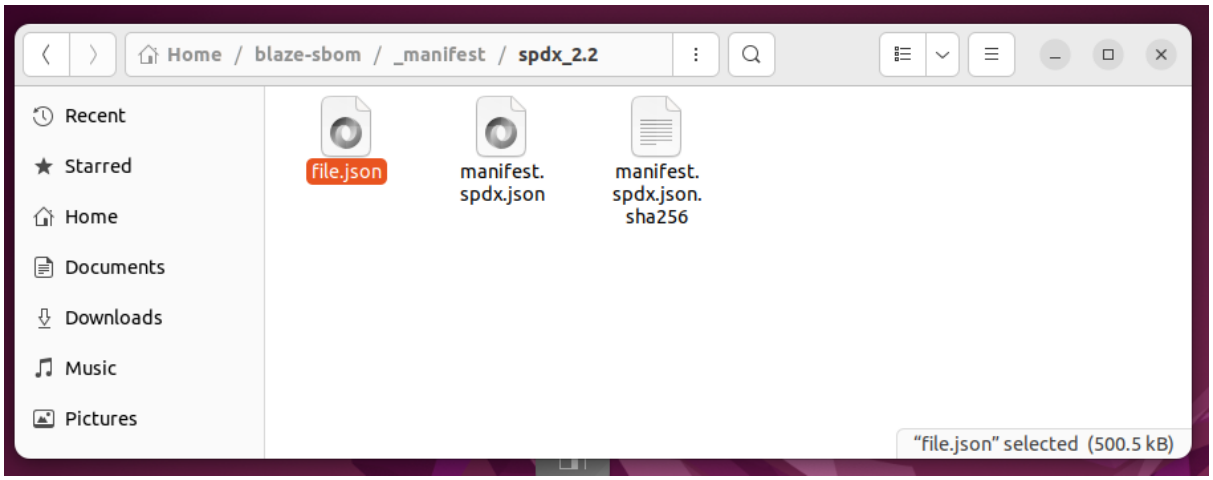



圖 25 file.json 檔案位置

步驟七、查看 file.json 檔案

開啟 file.json 檔案，文件詳細說明各元件弱點版本與 CVE 編號詳見圖 26，開始進行後續應對漏洞策略規劃

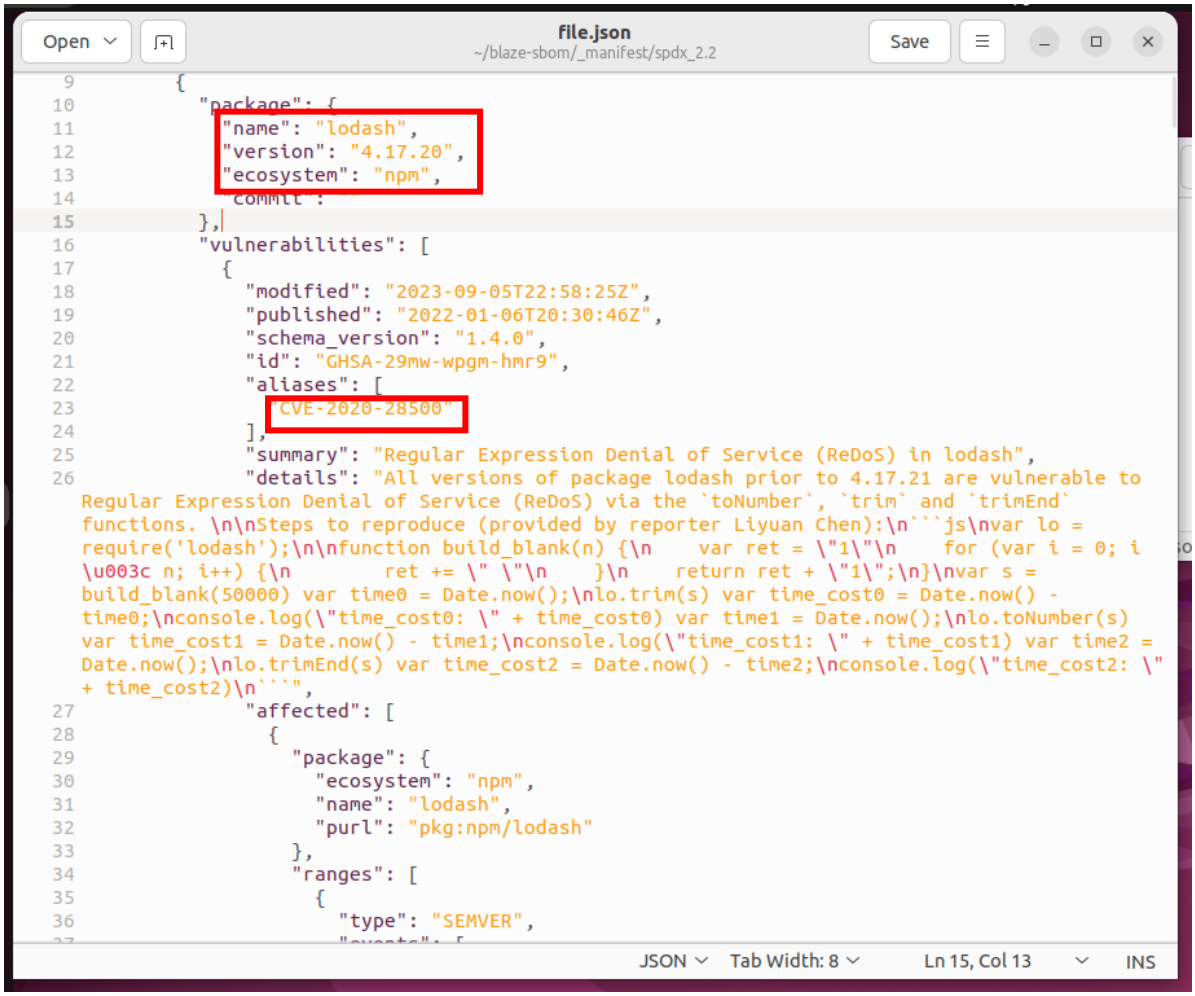



圖 26 file.json 檔案查看元件弱點版本與 CVE 編號

# CycloneDX Generator 開源工具說明

CycloneDX Generator 開源工具，工具支援目前主流的 Package Managers，依照 Package Managers 與語言關係對應整理如下表：

| 語言 | 支援 Package Managers |
|---|---|
| node.js | npm-shrinkwrap.json, package-lock.json, pnpm-lock.yaml, yarn.lock, rush.js, bower.json, .min.js |
| java | maven (pom.xml), gradle (build.gradle, .kts), scala (sbt), bazel |
| php | composer.lock |
| python | pyproject.toml, setup.py, requirements.txt, Pipfile.lock, poetry.lock, pdm.lock, bdist_wheel, .whl, .egg-info |
| go | binary, go.mod, go.sum, Gopkg.lock |
| ruby | Gemfile.lock, gemspec |
| rust | binary, Cargo.toml, Cargo.lock |
| .Net | .csproj, packages.config, project.assets.json, packages.lock.json, .nupkg |
| dart | pubspec.lock, pubspec.yaml |
| haskell | cabal.project.freeze |
| elixir | mix.lock |
| c/c++ | conan.lock, conanfile.txt |
| clojure | Clojure CLI (deps.edn), Leiningen (project.clj) |
| swift | Package.resolved, Package.swift (swiftpm) |

資料彙整：https://github.com/CycloneDX/cdxgen

- Ubuntu Desktop 22.04 LTS 環境操作步驟

步驟一、安裝 curl

開啟 terminal 輸入安裝指令

➢ snap install curl

完成後輸出畫面詳見圖 27

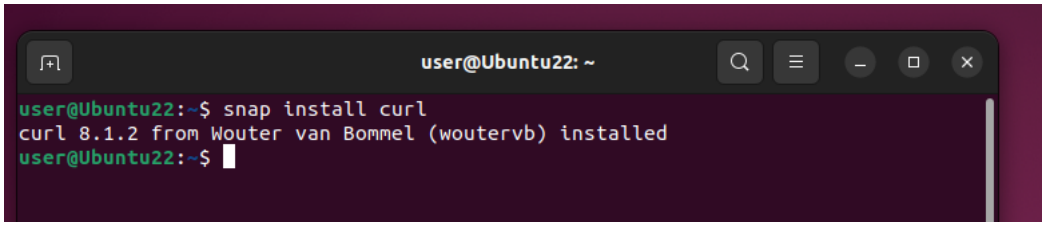



圖 27 安裝 curl 指令結果畫面

步驟二、下載 cdxgen

2.1 開啟 terminal 輸入指令下載 cdxgen

➢ curl -Lo cdxgen https://github.com/CycloneDX/cdxgen/releases/latest/download/cdxgen

2.2 開啟 terminal 輸入指令設定 cdxgen 權限

➢ chmod +x cdxgen

完成後輸出畫面詳見圖 28

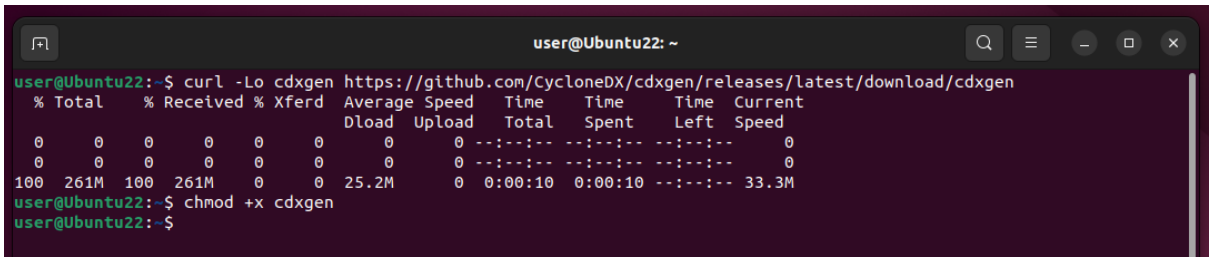



圖 28 下載 cdxgen 並設定權限結果畫面

步驟三、執行 cdxgen 掃描

3.1 將下載的原始檔檔案按右鍵選擇”Extract to…”進行解壓縮 zip，詳見圖 29

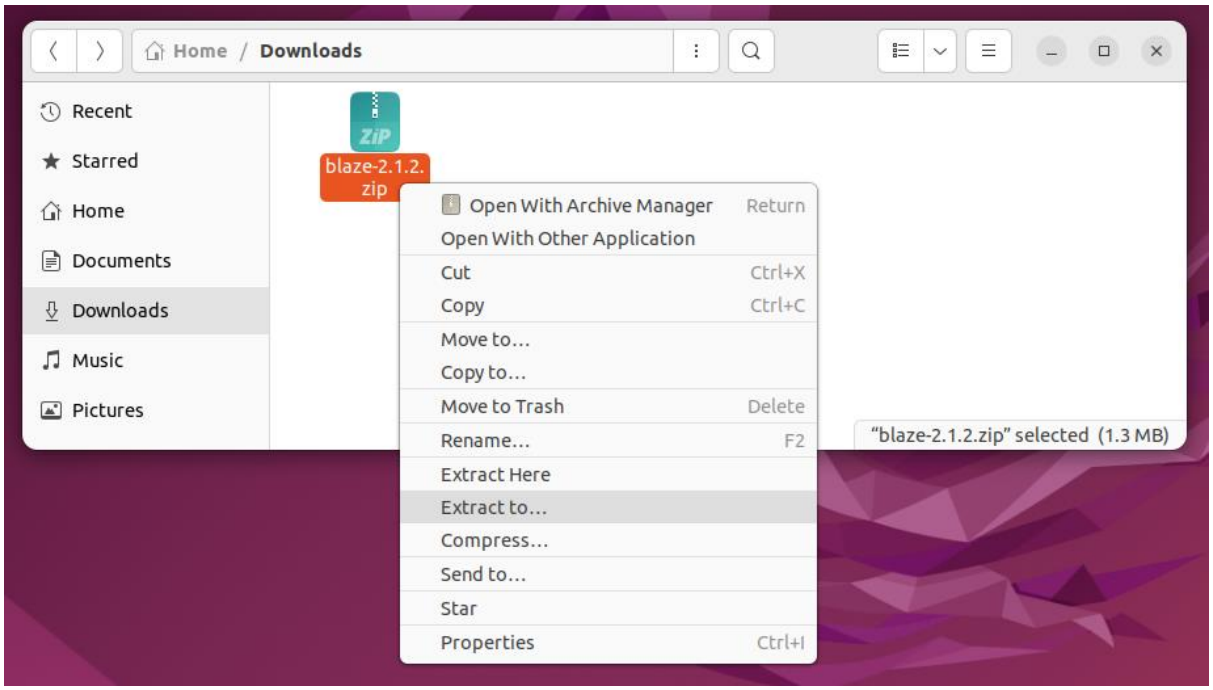



圖 29 對壓縮檔進行解壓縮操作(一)

3.2 選擇 Home 並按下 Select，詳見圖 30

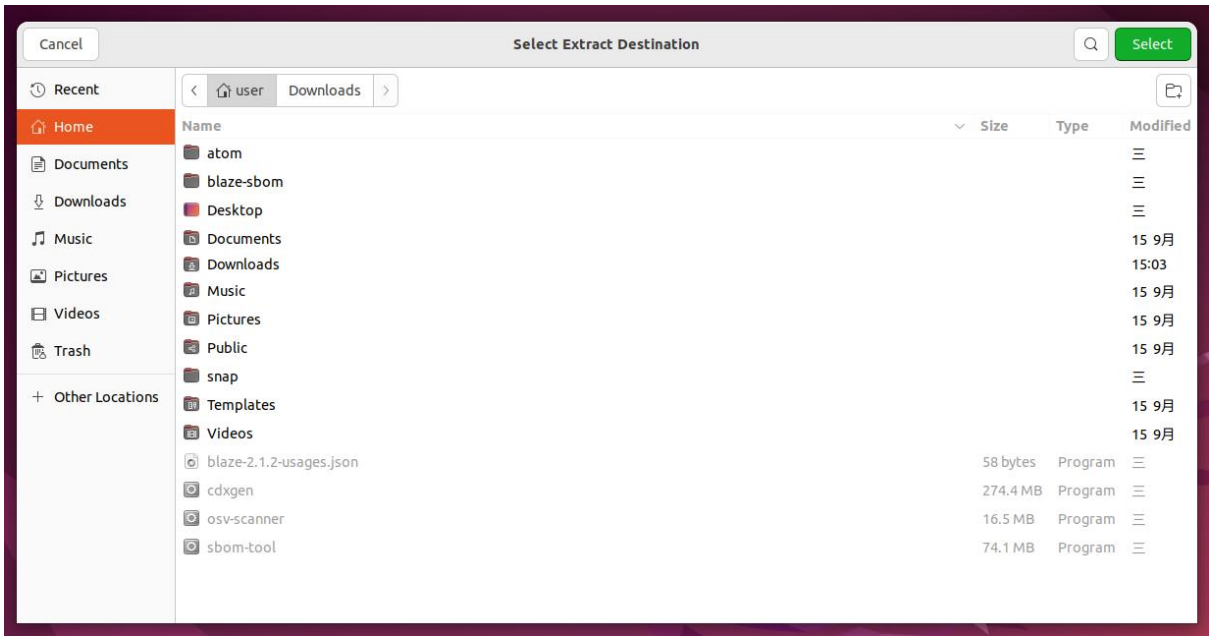



圖 30 對壓縮檔進行解壓縮操作(二)

3.3 於 Home 建立一個目錄”blaze-sbom”放置 SBOM 檔案

完成後詳見圖 31

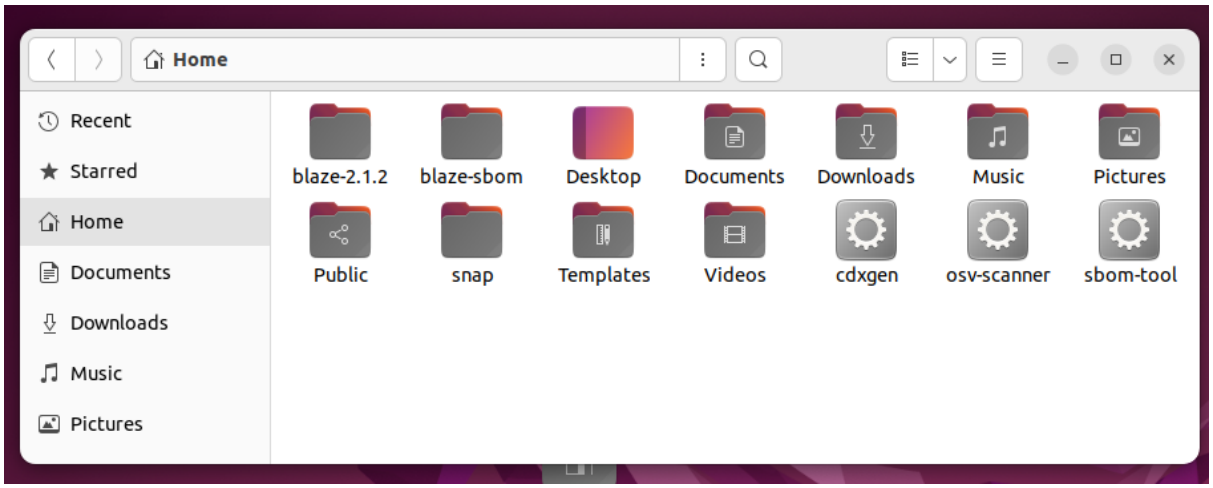



圖 31 放置原始碼與建立放置 SBOM 檔案的目錄

3.2 開啟 terminal 輸入指令執行 cdxgen

➢ ./cdxgen -r “./blaze-2.1.2” -o “./blaze-sbom/sbom.json” --spec-version 1.4

參數說明：

-r "進行 SBOM 掃描的原始程式碼目錄位置"

-o "產生 SBOM 檔案放置位置"

--spec-version 1.4 (目前 1.4 版本支援度較完整)

完成後輸出畫面詳見圖 32

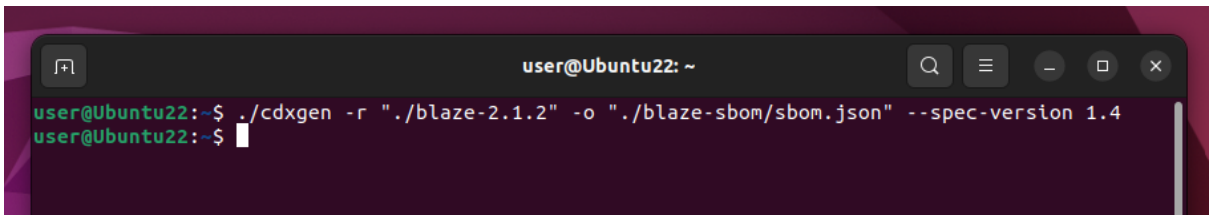



圖 32 執行 cdxgen 指令結果畫面

步驟四、查看 SBOM 檔案

4.1 ”/home/blaze-sbom” 目錄，找到 sbom.json 檔案詳見圖 33

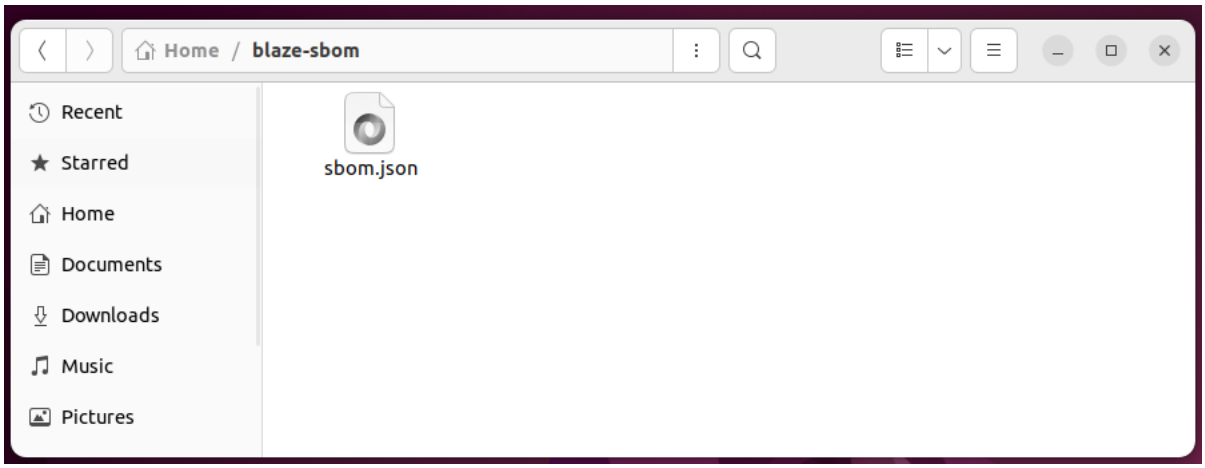
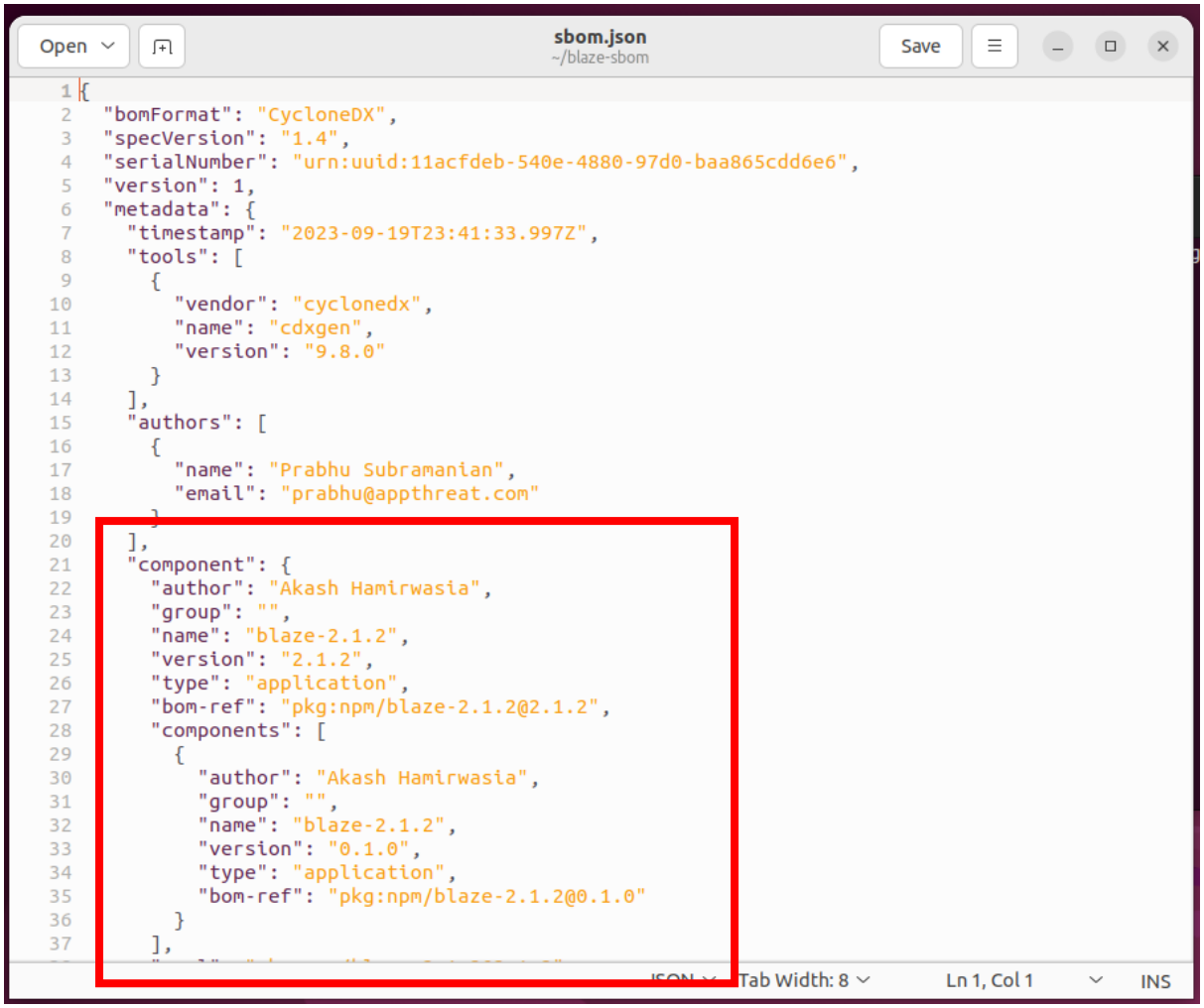



圖 33 sbom.json 檔案位置

4.2 開啟 sbom.json 檔案後，由 component 查看元件版本詳見圖 34





```
{
  "bomFormat": "CycloneDX",
  "specVersion": "1.4",
  "serialNumber": "urn:uuid:11acfdeb-540e-4880-97d0-baa865cdd6e6",
  "version": 1,
  "metadata": {
    "timestamp": "2023-09-19T23:41:33.997Z",
    "tools": [
      {
        "vendor": "cyclonedx",
        "name": "cdxgen",
        "version": "9.8.0"
      }
    ],
    "authors": [
      {
        "name": "Prabhu Subramanian",
        "email": "prabhu@appthreat.com"
      }
    ],
    "component": {
      "author": "Akash Hamirwasia",
      "group": "",
      "name": "blaze-2.1.2",
      "version": "2.1.2",
      "type": "application",
      "bom-ref": "pkg:npm/blaze-2.1.2@2.1.2",
      "components": [
        {
          "author": "Akash Hamirwasia",
          "group": "",
          "name": "blaze-2.1.2",
          "version": "0.1.0",
          "type": "application",
          "bom-ref": "pkg:npm/blaze-2.1.2@0.1.0"
        }
      ],
```


圖 34 sbom.json 查看元件版本

4.3 開啟 sbom.json 檔案後，由 ref 查看查看元件關係詳見圖 35


圖 35 sbom.json 查看元件關係

步驟五、下載 osv-scanner 工具

5.1 下載 osv-scanner 工具

https://github.com/google/osv-scanner/releases/latest/

以 v1.4.0 版為例

假設環境為 Linux 64 位元一般環境，找尋 linux-amd64 選擇：

https://github.com/google/osv-scanner/releases/download/v1.4.0/osv-scanner_1.4.0_linux_amd64

5.2 開啟 terminal 輸入指令下載 osv-scanner 工具

- curl -Lo osv-scanner https://github.com/google/osv-scanner/releases/download/v1.4.0/osv-scanner_1.4.0_linux_amd64

5.3 開啟 terminal 輸入指令設定 osv-scanner 工具權限

- chmod +x osv-scanner

完成後輸出畫面詳見圖 36

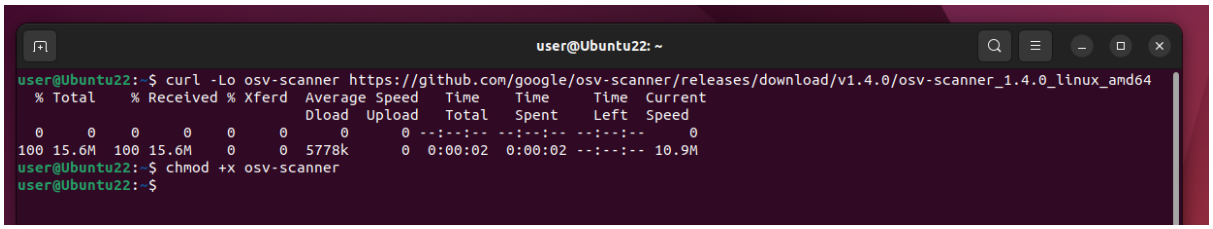



圖 36 下載 osv-scanner 工具與設定權限的指令結果畫面

步驟六、執行 osv-scanner 工具掃描產出 json 檔，進行後續應對漏洞策略

6.1 開啟 terminal 輸入指令執行 osv-scanner 工具

- ./osv-scanner --sbom="./blaze-sbom/sbom.json" --format json > "./blaze-sbom/file.json"

參數說明：

--sbom "SBOM 檔案位置"

--format json > "json 檔案輸出位置"

完成後輸出畫面詳見圖 37

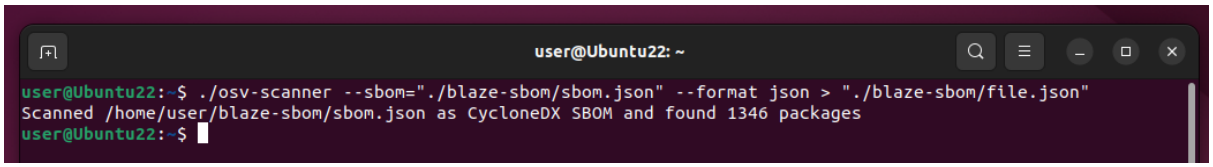



圖 37 執行 osv-scanner 指令結果畫面

6.2 開啟”/home/blaze-sbom”目錄，找到 file.json 檔案詳見圖 38

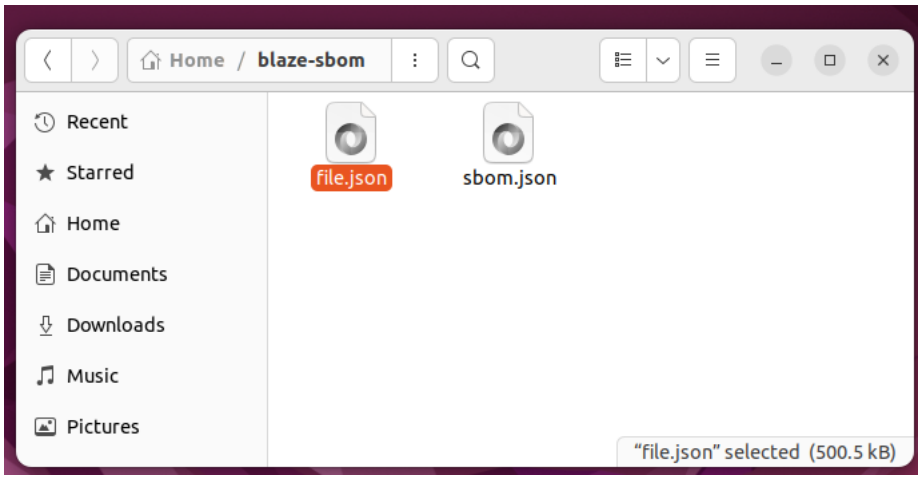



圖 38 file.json 檔案位置

步驟七、查看 file.json 檔案

開啟 file.json 檔案，文件詳細說明各元件弱點版本與 CVE 編號詳見圖 39，開始進行後續應對漏洞策略規劃

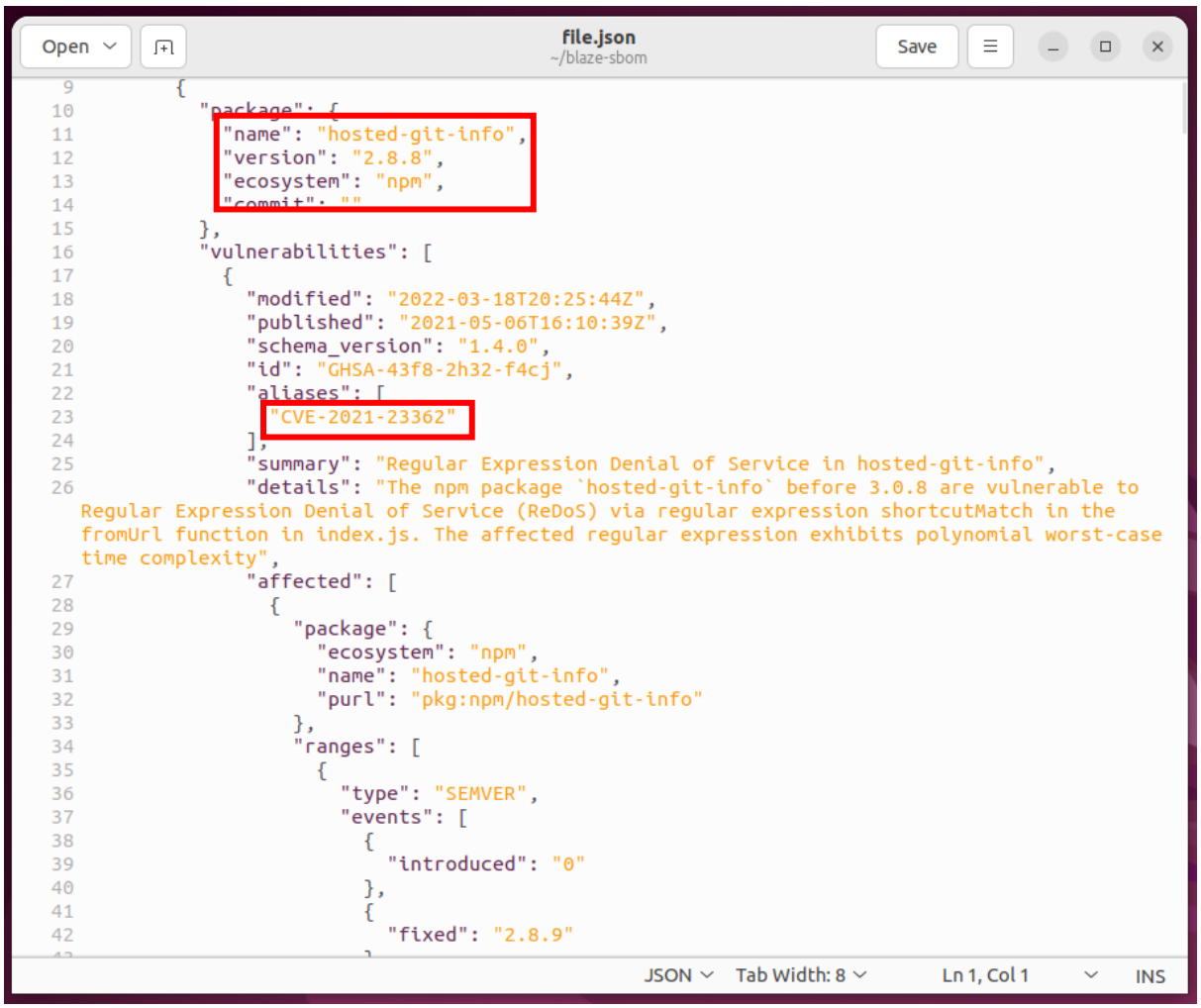



圖 39 file.json 檔案查看元件弱點版本與 CVE 編號