應用系統安全白皮書：選擇靜態分析工具 (SAST) 的六大評估項目

文章來源：廠商提供　翻譯整理：叡揚資訊 資訊安全事業處

該如何挑選擇靜態分析工具 (SAST, static application security testing) 是從安全專家、稽核人員、開發人員都會面臨的問題。選擇一個好的工具需要考量不同的層面，以下為挑選工具時的需求：

1. **支援語言**：確認 SAST 工具支援所使用的語言。
2. **存取原始碼及 binary code**。部份 SAST 工具只會檢測原始碼，但有些會連同 binary 一同處處理。相較於原始碼檢測，binary 檢測需連同建置環境一併提供，以便進行檢測。
3. **部署**。確認 SAST 工具作業方式：於單位內部或是採用雲端架構。
4. **對於組織內部程式碼安全訂定責任歸屬**。定義在組織中如何管控程式碼的安全性。如有個專門的團隊 (審核團隊、資訊安全團隊) 專職負責組織內的安全事務。另一方面也有組織認為每個開發團隊都需要專人負責安全。這些管理政策將影響 SAST 工具的部署架構、授權與工具的用法。

階段一：安裝

簡易安裝，應包含以下步驟：

1. **資源**。工具安裝為自動安裝還是手動安裝。若手動安裝，需考量安裝過程中所需的技能以及安裝花費的工時。
2. **彈性**。若為用戶端軟體，需安裝在每個終端的設備中，並要求每位開發團隊暫時無法作業，以便進行安裝。另一種集中管理的安裝只需進行一次。
3. **授權**。某些授權的架構是需於各終端設備皆需自己的授權。某些的授權是集中管理的，可減少需要多個授權的管理。

階段二：設定

需考量以下 2 個因素：

1. **困難度及複雜度**
   - **單純**。檢測工作需越簡單越好。檢測原始碼工具的操作不應加重使用者的工作量。
   - **語言的擴展性**。增加一個新的語言對於檢測來說不需再修改設定即可支援新的語言。
2. **檢測時間**。無論任何的工具，皆需花費時間
   在這邊需考量工具的特性或不同的檢測方式，讓花費的時間有機會加速。如對於開發團隊來說在大型專案中會需要只先針對專案中部份程式檢測。

階段三：檢測能力

檢測功能包含：

1. **語言支援範圍**。工具應不只包含現有的語言，應考量未來新語言的擴充。像是行動裝置或是發展中的語言 (Android, Objective C, Ruby on Rails)
2. **Framework 的支援範圍**。支援 Framework 讓 SAST 工具更能識別原始碼的弱點以及減少因為不認得 Framework 造成的誤判。
3. **多個檢測**。同時運行數個掃描或是支援多工處理。
4. **弱點涵蓋率**。SAST 工具應包含以下幾種類型的弱點：
   - **技術安全弱點。偵測已定義常見弱點如 OWASP TOP 10, Sans, CWE。由於每種工具的**
   - **分類方式不同。因此以符合規範來做為弱點涵蓋率比較。**
   - **商業邏輯缺陷。包含應用系統中的驗證方式或是後門。**
   - **最佳撰寫實務。包含錯誤處理與資源競爭。**
5. **結果的準確性**。為確保結果的精準性，工具應將檢測結果與已知驗證案例比對。通常會使用 OWASP WebGoat 專案做為驗證案例。然而，真正困難的是針對一個未知的應用系統來進行檢測，並能主動調整測試環境以防止發生以下情況：
   - True Positives (TPs) **的數量**：事實為真，猜測也為真，則為正確猜對
   - False Positives (FPs) **的數量**：事實為假，猜測為真，則為誤報。現今沒有一個工具能輸出完全無 FP 的檢測，但仍會以最少數量的 FP 為目標。
6. **客製化能力**。能適應不同程式 frameworks 與商業邏輯產生檢測結果。每個組織內採用的 framework、資料庫與驗證涵式都不盡相同，工具必須能客製調整檢測規則。同時因調整規則後減少 FPs 的發生。
7. **凝聚檢測結果**。將同一專案的所有檢測結果匯總在一起的能力。

階段四：結果管理

檢測結果需要清楚且能快速解決問題：

1. 結果分析與管理工具 - 結果分析應即時提供使用者相關安全情報與工具來修復問題
   - **弱點流程**。提供弱點的程式碼流程，幫助開發人員了解弱點的流程及其意義。
   - **最佳修復位置**。提供文字或圖像的最佳弱點描述。如精準的說明有相依性的問題，修復某特定位置即能同時處理多個弱點。
   - **標記與過濾能力**。可以依據政策對結果分群，並排列出優先順序。此外也提供篩選的功能輔助結果閱讀。
   - **專案結果追蹤**。工具需能持續追蹤專案中每次檢測的弱點狀態。
   - **檢測比較**。工具應可比較不同的檢測結果，以便監控弱點狀態。
2. 報表 - 應提供多層次的報告
   - **儀表板**。提供執行摘要等概觀資訊。
   - **報表內容**。提供檢測配置資訊與結果。

階段五：與 SDLC 的整合

在軟體開發生命周期 (SDLC) 中進行程式碼邏輯與技術分析

1. SDLC 模型。應包含：
   - **早期檢測**。盡早在開發過程中進行安全漏洞程式修補。工具應能提供無論程式是否已編譯，皆可進行檢測的能力。
   - **支援敏捷式開發與持續發展環境**。敏捷與持續發展 (又名為 DevOps) 要求短時間內需完成掃描並無法忍受任何的延遲 (檢測花費的時間及修復時間)。因此工具需整合至開發環境中以便開發人員可執行檢測。
   - **再次檢測**。專案複掃應不需要求已分析過的檔案。例如可只針對異動部份的程式碼進行檢測。
2. SDLC 整合。SAST 工具應要能無痛併入作業流程中。不僅節省開發人員時間，使安全變成開發流程的一部份。建議整合的部份包含：
   - **開發環境**。SAST 工具要能無縫融入開發環境中如 Visual Studio、Eclipse 或 IntelliJ。
   - **建置管理工具**。如 Bamboo, Jenkins, Maven 及 Ant。
   - **版本控管工具**。如 GIT, SVN, TFS, Mercurial, ClearCase。部份的 SAST 工具不需透過建置管理工具可以直接整合版本控管工具整合。
   - Bug **追蹤系統**。工具應要能將檢測結果匯入 Bug 追蹤系統，可依據弱點的嚴重性及其他工作項目安排修復進度。

階段六：供應商的支援

工具購買後並不是結束。而是一個持續的過程。如同任何的工具，購買後可能會有使用上的問題、最佳的做法或客製化的部份。皆需考量到供應商提供的服務：

- 檢測規則及政策可否依客戶企業調整設定
- 技術人員的支援及教育訓練
- 在導入及使用 SAST 工具的整個期間，應能回覆客戶的詢問
- 組織中軟體開發生命周期與 SAST 工具的整合