HCLSoftware

HCL AppScan Source

源碼檢測

Fast, Accurate, Agile

Security Testing

Kevin Chia (CEH / ISO 27001 LA)
Secure DevOps Technical Advisor
Greater China / Japan / ANZ

HCL AppScan

具有 360° 可視性的完整 AppSec 平臺

靈活的體驗

- 開發人員
- DevOps人員
- 資安人員

生態系統

- IDE
- CI/CD
- DTS
- SCM
- Robust API

部屬方式

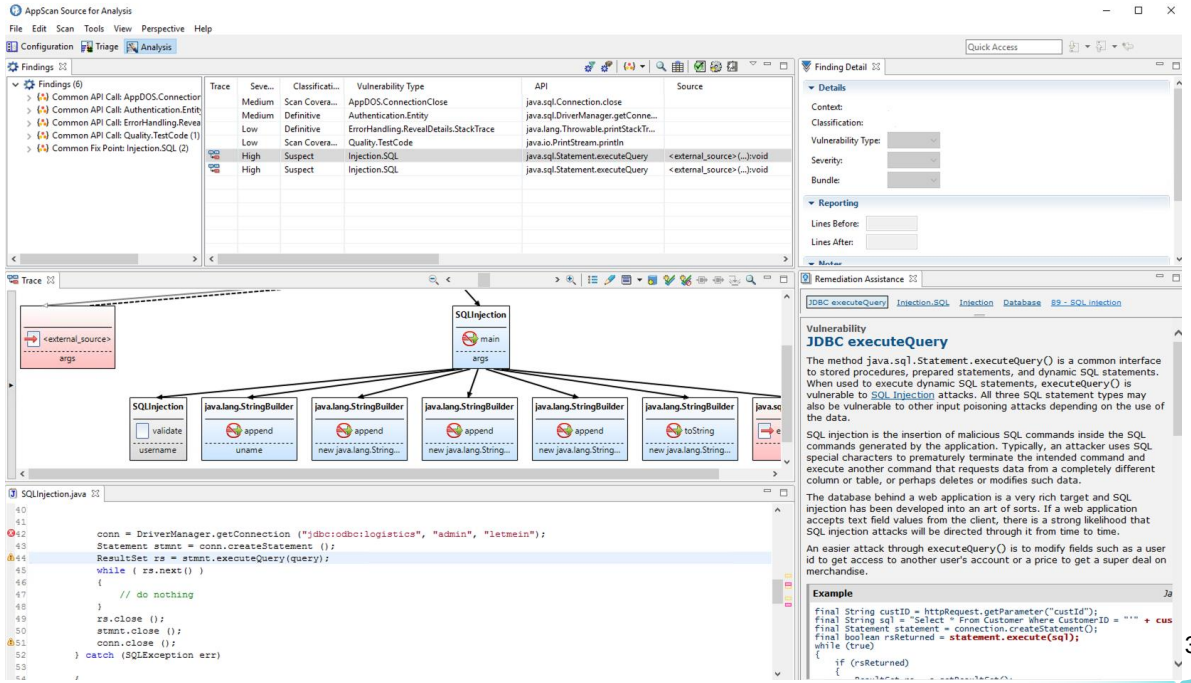
- 雲端 (SaaS)
- 地端
- 混合Hybrid
- MSP

服務

- **AppScan for You**
  個人化諮詢服務
- **Advanced Technical Services**
  工作坊、培訓和顧問服務
- **AppScan On Demand**
  全面託管服務

AppScan Source 亮點

- **源代碼安全測試工具 (SAST)**又名 “白盒檢測”
- **可與AppScan Enterprise整合**，為全方位解決方案
- **CI/CD整合，Restful API調用**, Jenkins, Github, Azure Devops
- **多樣產業標準報告:OWASP Top 10 2021, Mobile Top 10, CWE 等**
- **地端解決方案**, 可與Enterprise 混合部屬
- **自動化弱點評估**，ICA/IFA – 人工智慧判斷
- **中文**修補建議及技術文件
- **不限掃描數量**



HCL AppScan Source 支援語言

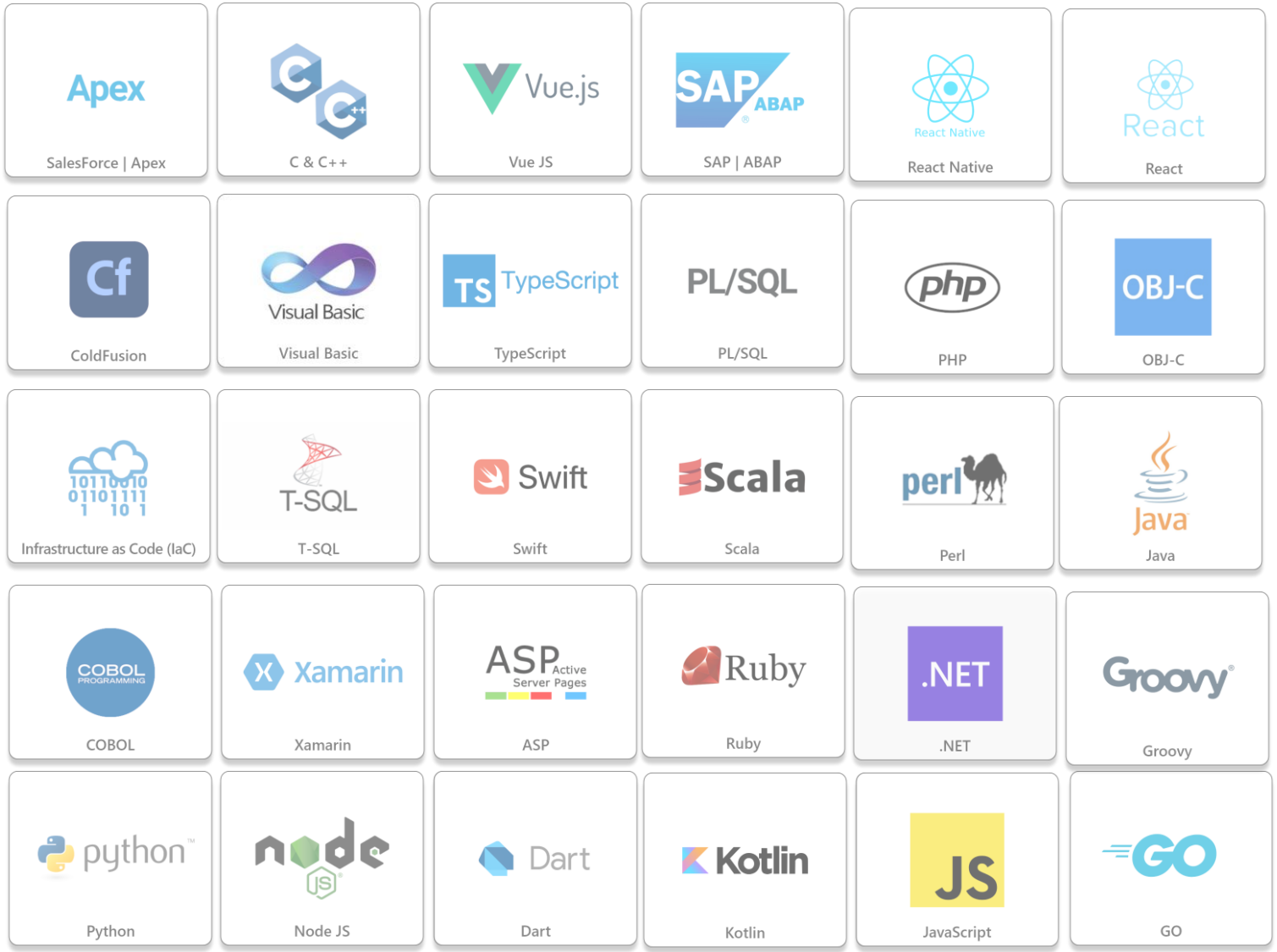
SAST 的大量語言支援 30+

大型規則庫：

- 廣泛的框架和庫支援
- 基於標準的規則
- CERT、NIST、OWASP、CVE/CWE

分析技術：

- 數據流分析
- 控制流分析
- 過程間分析
- 多執行緒掃描
- 基於模式的語義分析
- 人工智慧 機器學習分類



AppScan V10 – 測試項目 CWE Top 25

排名	CWE	漏洞	說明
1	CWE-787	超出範圍寫入	當向陣列中寫入資料時，寫入的資料超過了陣列預留空間的範圍。攻擊者可以利用這種漏洞來執行任意代碼，甚至完全接管受影響的系統。
2	CWE-79	不當摧毀產生網頁期間的輸入（跨網站 Scripting）	當應用程式將使用者輸入的資料顯示在Web頁面上時，沒有正確地過濾或轉義用戶輸入，使攻擊者能夠注入惡意腳本並欺騙用戶執行操作，例如竊取憑據或竊取敏感性資料。
3	CWE-125	超出範圍讀取	當程式讀取超過陣列預留空間的範圍時，會發生此類型的漏洞。攻擊者可以利用此漏洞洩露敏感性資料或執行拒絕服務攻擊。
4	CWE-20	輸入驗證不適當	當應用程式未正確驗證使用者輸入時，攻擊者可以利用此漏洞執行惡意操作，如注入惡意程式碼或繞過身份驗證。
5	CWE-78	不當摧毀作業系統指令中使用的特殊元素（作業系統指令注入）	當應用程式未正確過濾或驗證用戶輸入時，攻擊者可以通過注入惡意命令來執行未經授權的操作，如執行任意命令或訪問敏感性資料。
6	CWE-89	不當摧毀 SQL 指令中使用的特殊元素（SQL 注入）	注入是一種常見的Web安全性漏洞，攻擊者可以利用該漏洞執行惡意SQL查詢並獲取未授權的存取權限。
7	CWE-416	可用之後使用	在使用後釋放物件或記憶體後，攻擊者可以利用該漏洞執行任意代碼或崩潰應用程式。
8	CWE-22	不當限制受限目錄的路徑名稱（路徑遍訪）	路徑遍歷漏洞允許攻擊者通過將檔路徑指向外部目錄來訪問系統檔和目錄，從而執行惡意操作。
9	CWE-352	偽造跨網站要求 (CSRF)	CSRF攻擊利用用戶當前登錄的身份驗證，向應用程式發送未經授權的請求。攻擊者可以執行未經授權的操作，如更改密碼或執行交易。
10	CWE-434	未限定上傳危險類型的檔案	攻擊者可以利用未對上傳的檔進行正確的驗證和過濾的漏洞，上傳包含惡意程式碼的檔，從而執行惡意操作。

包括但不限於

AppScan V10 – 測試項目 CWE Top 25

排名	CWE	漏洞	說明
11	CWE-306	遺漏鑒別重要功能	如果應用程式缺少對重要功能的身份驗證，攻擊者可以利用該漏洞訪問未授權的功能並執行惡意操作。
12	CWE-190	整數溢位元或折返	整數溢出和折返漏洞允許攻擊者通過利用算數運算的錯誤，從而導致應用程式執行不正確的操作或崩潰。
13	CWE-502	不受信任資料的解除序列化	反序列化漏洞是指攻擊者利用未正確驗證和過濾的反序列化資料，從而導致應用程式執行不安全的操作。
14	CWE-287	不當鑒別	如果應用程式缺少對使用者的身份驗證，攻擊者可以利用該漏洞訪問敏感資訊或執行未經授權的操作。
15	CWE-476	空值指標解除參照	空指標解除引用漏洞可以導致應用程式崩潰或執行未定義的操作，攻擊者可以利用該漏洞執行任意代碼。
16	CWE-798	使用寫在程式中的認證	通過向緩衝區邊界之外寫入資料，攻擊者可能會更改重要的程式狀態或執行惡意程式碼。
17	CWE-119	在記憶體緩衝區範圍內不當限制作業	通過Web應用程式接受的惡意輸入，攻擊者可以在目標使用者的流覽器上執行腳本代碼。
18	CWE-862	遺漏授權	當程式讀取的資料超出了記憶體緩衝區的邊界時，攻擊者可能會訪問敏感資訊、更改程式列為或執行代碼。
19	CWE-276	預設許可權不正確	未正確驗證用戶輸入的資料類型、格式、長度、範圍或語義，導致攻擊者能夠提交惡意輸入來執行不受控制的操作。
20	CWE-200	將機密資訊洩露給未獲授權的動作者	在命令中使用未過濾或未適當過濾的用戶輸入時，攻擊者可以在目標系統上執行任意命令。

包括但不限於

AppScan V10 – 測試項目 CWE Top 25

排名	CWE	漏洞	說明
21	CWE-522	未充分保護的認證	當應用程式使用未適當過濾或轉義的用戶輸入來構造SQL查詢時，攻擊者可以通過注入惡意程式碼來執行任意SQL查詢。
22	CWE-732	重要資源的許可權指派不正確	當程式在使用已釋放的記憶體時，攻擊者可以更改程式狀態、引發崩潰或執行惡意程式碼。
23	CWE-611	不適當限制 XML 外部實體參照	當程式未正確限制路徑名時，攻擊者可以訪問和修改受保護的檔或目錄，或利用其他漏洞來執行任意代碼。
24	CWE-918	偽造伺服器端要求 (SSRF)	攻擊者通過欺騙使用者在應用程式中執行未經授權的操作，以利用用戶已登錄的憑據。
25	CWE-77	不當中性化指令中使用的特殊元素（「指令注入」）	當應用程式未正確驗證上傳檔的類型或內容時，攻擊者可以上傳包含惡意程式碼的檔，從而更改應用程式的行為或執行惡意程式碼。

15, 16, 20, 73, 74, 77, 79, 88, 89, 90, 91, 95, 98
105, 109, 112, 113, 116, 117, 120, 129, 130, 131, 134, 185, 190
201, 209, 242, 250, 257, 264, 266, 267, 285, 287, 288, 295
310, 311, 312, 319, 327, 331, 335, 345, 352, 359, 367, 382, 388, 390, 398
400, 404, 407, 425, 434, 447, 470, 472, 477, 489, 497
506, 507, 511, 517, 520, 521, 522, 523, 524, 525, 532, 538, 543, 544, 546, 547, 565, 569, 586
601, 613, 615, 624, 643, 645

包括但不限於

AppScan V10 – 測試項目 OWASP API Top 10

排名	OWASP	說明
1	失效的物件級授權 Broken object level authorization	在請求中操縱對象識別符以獲得對敏感性資料的未經授權的存取。
2	失效的使用者驗證 Broken user authentication	如果驗證實作不正確，攻擊者可能能夠冒充 API 使用者，並存取機密資料。
3	過度的資料暴露 Excessive data exposure	許多 API 在暴露資料方面犯錯，並依靠 API 使用者正確篩選資料。這可能允許未經授權的人檢視資料。
4	缺乏資源和限速 Lack of resources & rate limiting	預設情況下，許多 API 不限制它們在給定時間可以接收的請求的數量或大小。這使他們容易受到阻斷服務 (DoS) 攻擊。
5	失效的功能級授權 Broken function level authorization	這種風險與授權有關。API 使用者可能被授權做太多事情，導致資料暴露。
6	批量指派 Mass assignment	API 自動將使用者輸入套用至多個屬性。例如，攻擊者可以利用此漏洞將自己變更為管理員，同時更新其使用者設定檔的其他一些無害屬性。
7	安全設定錯誤 Security misconfiguration	這涵蓋了設定 API 時的各種錯誤，包括錯誤設定的 HTTP 標頭、不必要的 HTTP 方法，以及 OWASP 所說的「包含敏感性資訊的詳細錯誤訊息」。
8	插入 Injection	在資料隱碼攻擊中，攻擊者向 API 傳送專門的命令，誘騙其洩露資料或執行其他一些意外動作。瞭解 SQL 資料隱碼攻擊。
9	資產管理不當 Improper assets	當沒有追蹤當前的生產 API 和已棄用的 API 時，就會發生這種情況，從而導致影子 API。API 很容易受到這種風險的影響，因為它們往往提供太多端點。
10	記錄和監控不足 Insufficient logging & monitoring	OWASP 指出，研究表明通常需要 200 多天才能偵測到洩露。詳細的事件記錄和嚴密監控可以使 API 開發人員更早地偵測和阻止洩露。

包括但不限於

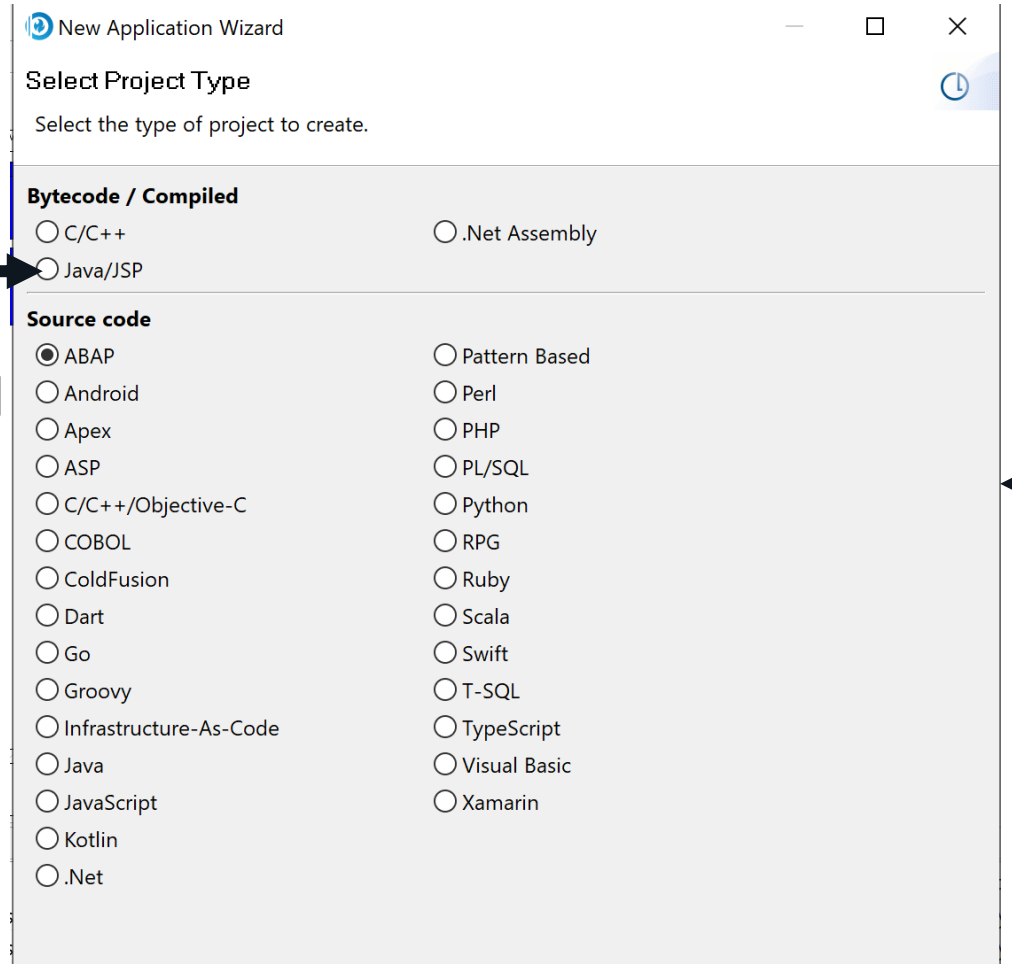
位元組碼/編譯 vs 原始程式碼掃描

AppScan Source 兩者都支持!

位元組碼/已編譯

- 高精准度
- 自訂原則降低FNs
- 可識別消毒劑或驗證器以降低誤判
- 從接收器到源的資料跟蹤

When
- 執行時間更長的自動化 (CI/CD)
- 重要應用程式
- 人工代碼審查

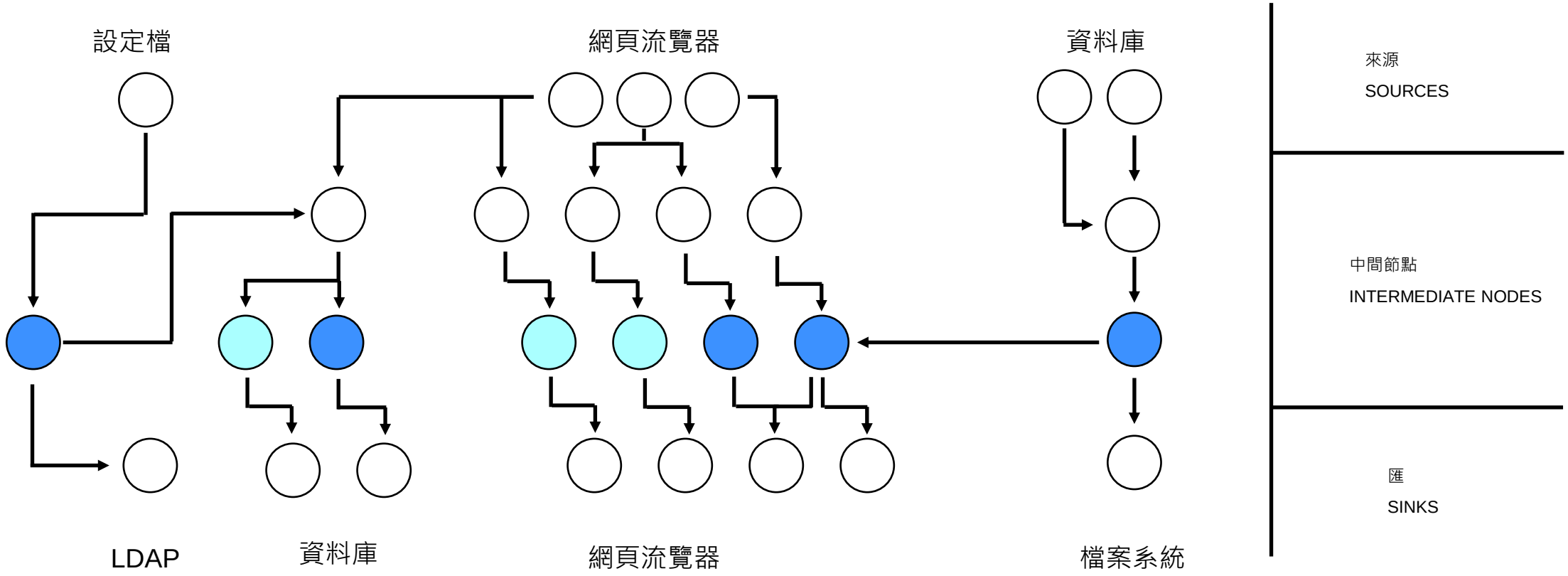


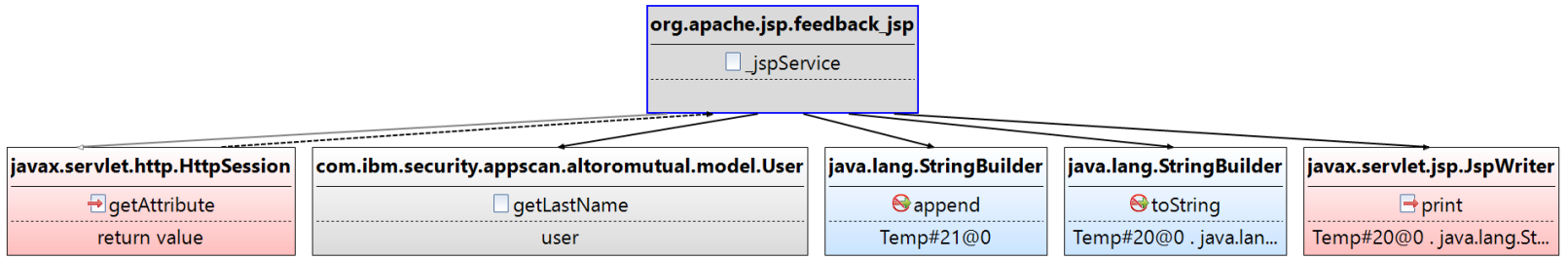
原始程式碼

- 掃描所需時間短
- 修補建議對應到漏洞
- 可直接掃描原始程式碼
- 可掃部分原始程式碼

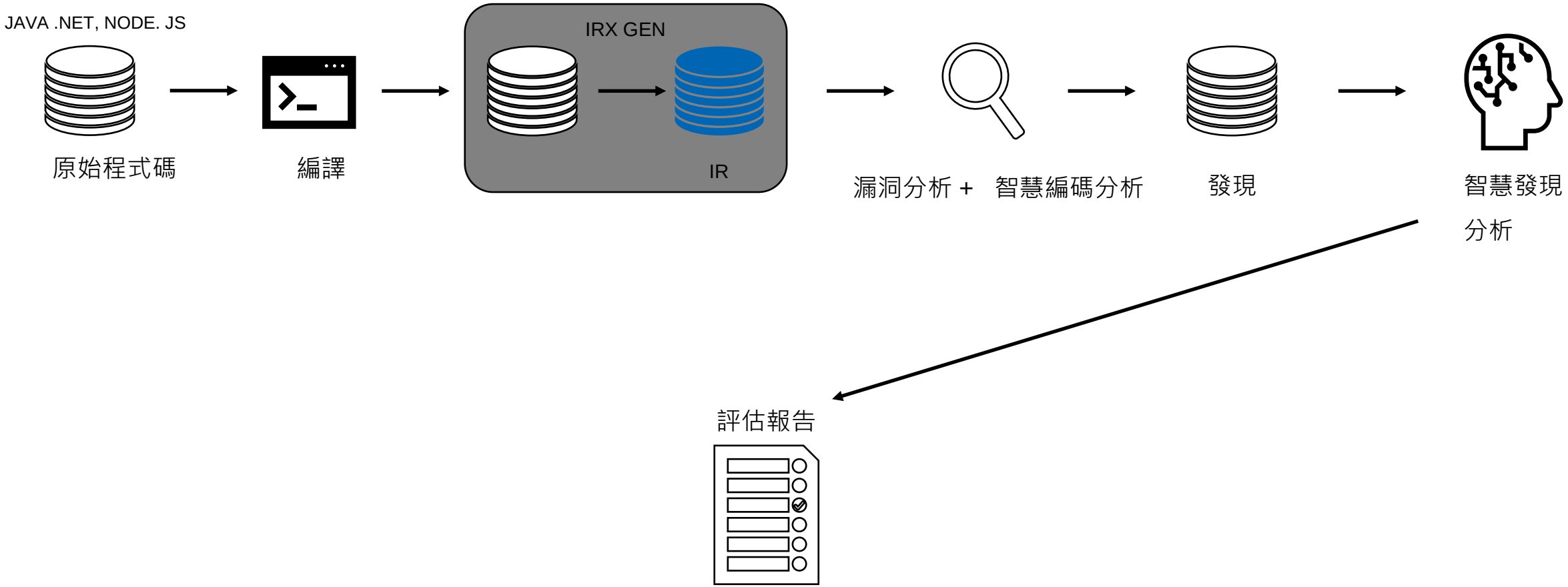
When
- 非常快的 CI/CD
- 非關鍵應用程式
- 當無法編譯時

靜態分析的工作原理：資料流程分析 (TAINT ANALYSIS)

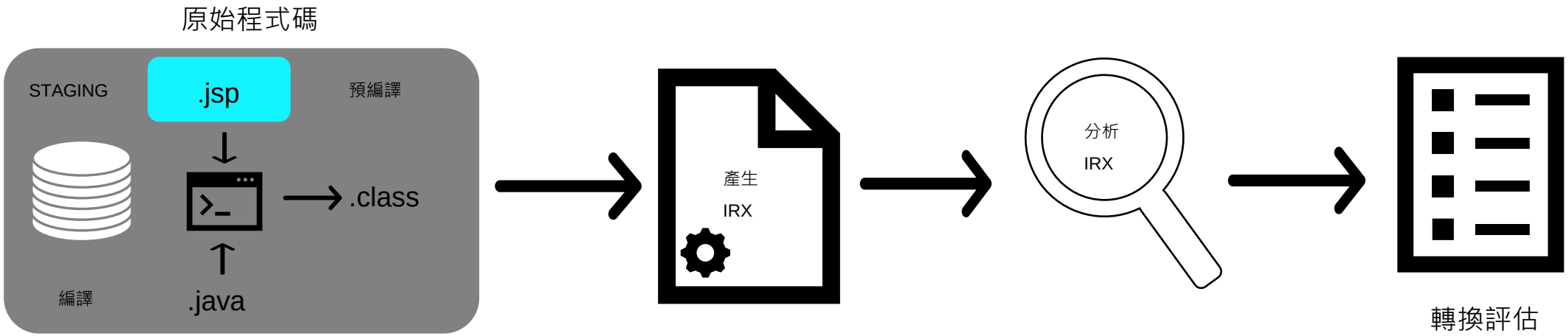




AppScan Source - 靜態分析掃描過程



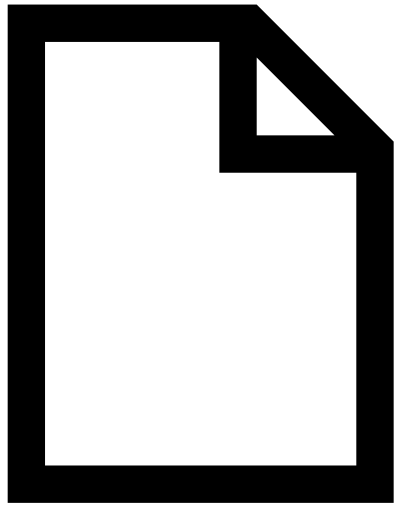
示例掃描：JAVA 掃描



INTERMEDIATE REPRESENTATION OF APPLICATION (IRX文件）

IRX 包含以下內容：

1. 關於應用程式的中繼資料(Metadata)
2. 對於編譯語言 - 應用程式中資料流程的表示
3. 對於解釋型語言 - devops 掃描器非資料流程



靜態分析性能：並行掃描

測試應用	預計速度提升
altoromutual.war	73%
Authorize.net.jar	74%
FtpServer.jar	68%
mockito.jar	13%
SQLConnection.jar	67%
webgoat-container-7.1.wa.war	85%

INFRASTRUCTURE AS CODE (IaC) — SCAN DOCKER

IaC — 僅限 SAST：

- 環境應該是什麼樣子的代碼表示

  （開發人員喜歡開發盡可能接近正式環境）

- Docker、Kubernetes (K8s)

- Dockerfiles, docker-compose (yaml), K8s Pods (yaml), sh / bat

- 我們不掃描容器，我們掃描配置！

Rule	Severity
Exposed Docker Daemon Socket	High
User Not Specified	High
New Priviledges Not Blocked	High
Running Container As Privileged User	High
Default Security Profile Disabled	High
Mounted Volumes Not Read Only	Low
No CPU Limit Set	Medium
No Memory Limit Set	Medium
Kernel Capabilities Not Configured	Medium
Restart On Failure Not Limited	Medium
No Ulimit Set	Medium
Root File System Not Set To Read Only	Medium

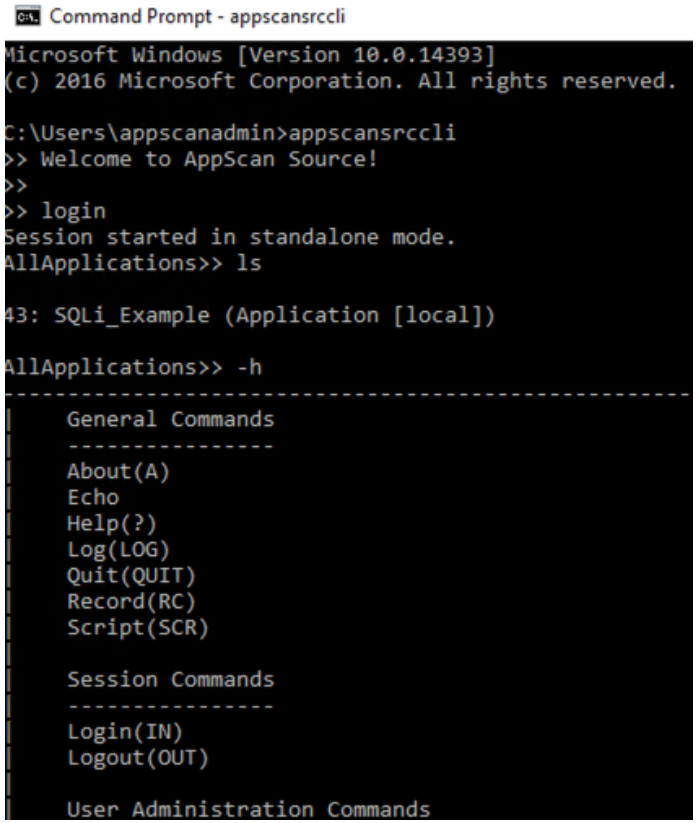
HCL AppScan Source 用於自動化

AppScan Source用於自動化:

- 在構建環境中或在持續整合期間提供 AppScan Source 掃描

產品特點：

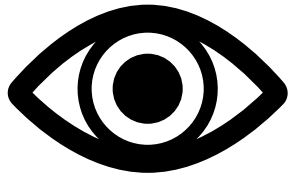
- 排程掃描請求
- 掃描應用程式
- 保存評估
- 生成調查結果報告
- 將結果發佈到 AppScan Enterprise
- 與 ANT、Make 和 Maven 整合
- 與其他解決方案的CLI 命令
- 用戶管理



適合構建工程師

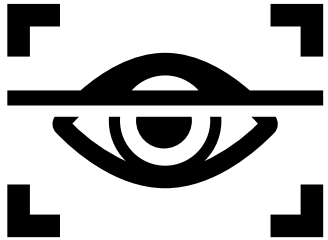
APPSCAN SOURCE 人工智慧能力

Intelligent Code Analytics (ICA):



- 通過為應用程式在跟蹤分析期間使用的任何框架生成安全規則來擴展分析範圍並消除誤報

Intelligent Findings Analytics (IFA):



- 通過提供對應用程式安全測試結果的全自動審查，可將誤報率降低多達 99%，並消除冗長的人工審查流程

簡化的修復組建議和增量：

- 提供修復建議，幫助開發團隊通過單個代碼修復解決多個漏洞

INTELLIGENT FINDINGS ANALYTICS (IFA)

真實世界的結果：

- 達到或超過人類專家
- 以秒為單位返回結果，而不是幾小時或幾天
- 誤報率平均減少 99%
- 直接整合回開發工作流程
- 在代碼中的一個位置平均修復 5-50 多個問題

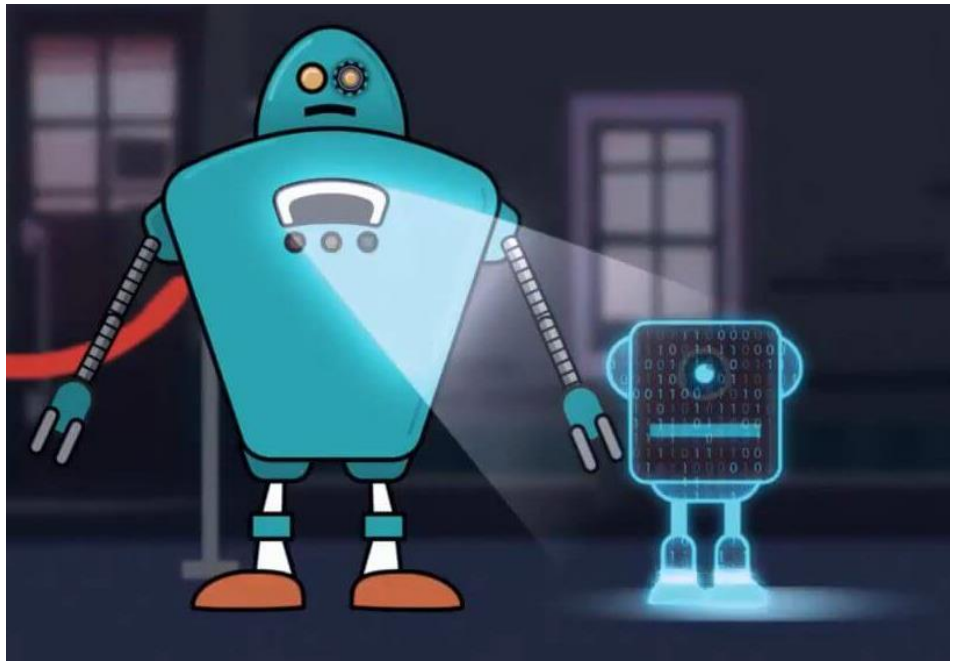
實際應用示例	掃描結果	漏洞	修復建議
應用程式 1	55,132	14,050	60
應用程式 2	12,480	1,057	35
應用程式 3	247,350	1,271	103

*PATENTED TECHNOLOGY

INTELLIGENT CODE ANALYTICS (ICA)

真實世界的結果：

標記類型	正確分類	不正確的分類	百分比正確
TAINT PROPEGATOR	12,871	193	98.5%
SOURCE	30,675	724	97.7%
SINK	31,335	64	99.8%



真實世界的結果
由 ICA 識別的 Java 8 300K
（8 分鐘內分類）

真實世界的結果
客戶發現與使用 SpringBot 和 Play 框架相關的漏洞

30 個客戶應用程式的基準（ICA 在 2/3 中發現漏洞）

AppScan – 報告

完整繁體中文報告、修補建議

提供不同合規報告

- CWE Top 25
- DISA STIF V4R10 V5R1
- OWASP API/Mobile/Web Top 10
- PCI DSS V3.2
- Software Security Profile
- 可以將結果發佈到 AppScan Enterprise
- 可以將結果發佈到 AppScan On Cloud

跨網站 Scripting

問題概觀 / Java / JSP JspWriter

JSP JspWriter

API
javax.servlet.jsp.JspWriter.println(java.lang.String):void javax.servlet.jsp.JspWriter.print(java.lang.String):void

原因

javax.servlet.jsp.JspWriter 類別會將 HTML 頁面寫入至使用者瀏覽器。此類別一般不會直接用於一般 Java 檔案，主要用於 JSP 伺服器上從 JSP 頁面前置編譯的 Java 檔案。

這些 JspWriter API 會將字串寫入至 HTML 頁面。如果字串是靜態字串或來自信任的來源，則這不是問題。不過，如果字串來自使用者輸入或其他不受信任的來源，則攻擊者可以將 JavaScript 這類惡意 Script 引進輸出 HTML 頁面。檢視時，這些 Script 將由啟用 Script 的 Web 瀏覽器進行解譯，並與檢視使用者的安全環境定義搭配執行以與原始網站通訊，進而導致 XSS 攻擊。

程式碼範例

HCL AppScan

hcl-software.com/AppScan