台灣數位安全聯盟
Taiwan Cyber Security Alliance

從駭客視角看網站安全
OWASP Top 10 實務解析

蔡一郎

蔡一郎 Steven Tsai

以策略與專業，推動資安產業發展與人才培育，
共築**更安全的數位未來**。

> 以策略與專業，
> 推動資安產業發展與
> 人才培育，
> 共築更安全的
> 數位未來。
>
> Steven Tsai

學歷

- 国立成功大學 電腦與通信工程研究所 博士候選人
- 國立成功大學 電機工程研究所 碩士
- 国立成功大學 EMBA 碩士班

現任

- 來毅數位科技股份有限公司 全球資安長 暨台灣區總經理
- 博暉顧問股份有限公司 創辦人
- 台灣數位安全聯盟 榮譽理事長
- 台灣網際空間與安全策略發展協會 理事長
- 台灣資安大聯盟 副會長
- 多個國際資安組織與專業社群領導職務 (InfoSec Taiwan / OWASP / HoneyNet / Cloud Security Alliance / CSCIS...等)
- 政府部會資安稽核委員
- 自由作家：資訊圖書著作 37 本 技術專欄文章 100+ 篇
- 部落格：https://blog.yilang.org

曾任

- 微智安聯股份有限公司 創辦人兼執行長
- 財團法人國家實驗研究院國家高速網路與計算中心 研究員
- 台灣數位安全聯盟 理事長
- 中華民國資料保護協會 監事
- 中華民國資訊安全學會 常務監事
- 數位經濟暨產業發展協會 理事
- 台灣資訊安全聯合發展協會 監事
- 中華民國人壽保險商業同業公會 資安顧問 / 資安工作小組委員

專業證照

RHCE · CCNA · CCAI · CEH · CHFI · ACIA · ITIL Foundation · ISO 27001 LA
ISO 20000 LAC · BS10012 · LAC · ISO 17065 · ISO 42001 LA · CSA STAR Auditing
CCSK · CMMC Essential · CSM · CSPO

課程大綱

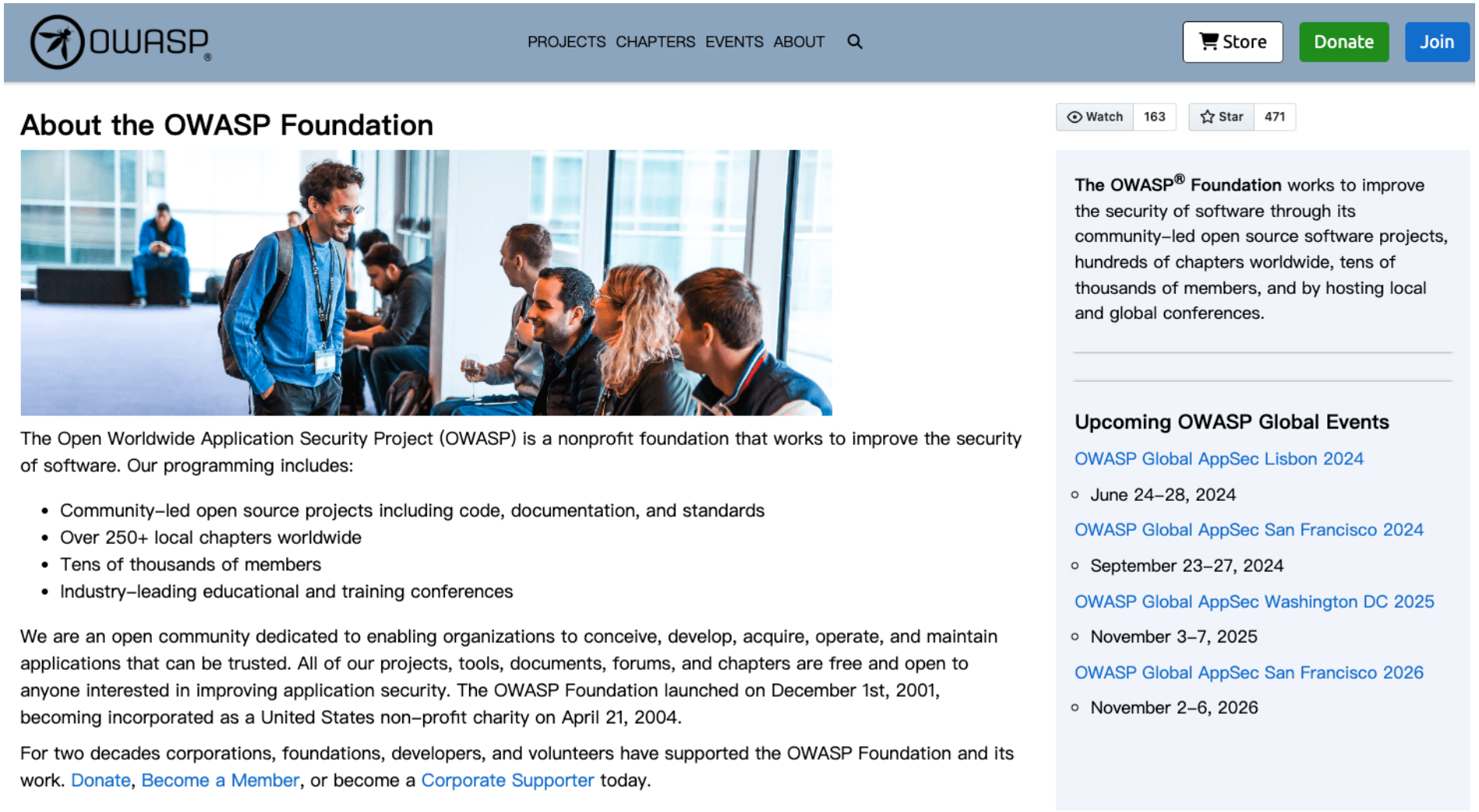
- OWASP Top 10 風險解析
- 常見攻擊案例分享
- 安全開發與防護思維

TWCSA 台灣數位安全聯盟
Taiwan Cyber Security Alliance

TWCSA
台灣數位安全聯盟
Taiwan Cyber Security Alliance

OWASP To 10 : 2025

OWASP 以應用程式安全出發的國際資安組織



https://owasp.org/

認識應用程式安全

- 應用程式安全(AppSec)，包括向開發團隊引入安全軟體開發生命週期的所有任務
- 最終目標是改進安全實踐，並通過它來發現、修復並防止應用程式中的安全問題
- 它涵蓋了從需求分析、設計、實施、驗證和維護的整個應用程序生命週期
- 網站應用程式安全，專門處理網站、 Web 應用程式和Web 服務的安全性
- 多種自動化工具可用於識別應用程式中的漏洞。用於識別應用程式漏洞的常用工具類別包括：
  - 靜態應用程式安全測試（SAST）在應用程式開發過程中分析安全漏洞的原始碼
  - 動態應用程式安全測試（DAST，通常稱為漏洞掃描器）通過抓取和分析網站自動檢測漏洞
  - 交互式應用程式安全測試（IAST）使用軟體工具從內部評估應用程式

https://zh.wikipedia.org/zh-tw/%E6%87%89%E7%94%A8%E7%A8%8B%E5%BC%8F%E5%AE%89%E5%85%A8

靜態應用程式安全測試（SAST）

- 資料流分析 (Data Flow Analysis)
  - 追蹤資料從輸入點（來源，如使用者輸入）到敏感執行點（匯點，如資料庫查詢）的傳遞路徑，確認資料是否經過正確的過濾或轉義，以防範 A05 注入攻擊。
- 控制流分析 (Control Flow Analysis)
  - 檢查程式碼執行的邏輯順序與路徑，識別是否存在邏輯缺陷、無窮迴圈或無法到達的程式碼（Dead Code），這與 A06 不安全設計 息息相關。
- 語法與語義分析 (Syntax & Semantic Analysis)
  - 將程式碼轉換為抽象語法樹（AST）或中間表示（IR），藉此識別違反安全編碼規範的語法結構，例如使用已廢棄且不安全的 API。
- 配置與環境檢查 (Configuration Analysis)
  - 分析專案中的設定檔（如 XML, YAML, JSON），找出如硬編碼金鑰、預設密碼或不安全的權限設定，直接對應 A02 安全配置錯誤。
- 特徵匹配與規則庫 (Pattern Matching & Ruleset)
  - 利用內建的安全規則庫（如 OWASP Top 10 規則），比對程式碼中是否存在已知的弱點模式，例如使用了 MD5 等過時的弱加密演算法（A04 加密失效）。
- 結構化分析 (Structural Analysis)
  - 評估程式碼整體的層級架構，確認是否符合安全框架的要求，避免跨層級的非法呼叫或不當的例外處理邏輯（A10 特殊狀態處理不當）。

動態應用程式安全測試（DAST）

- 執行環境模擬 (Runtime Analysis)
  - 在程式運作時進行測試，能發現僅在執行期才會出現的問題，例如 A02 安全配置錯誤 或伺服器環境（如伺服器標頭、SSL/TLS 設定）的弱點。
- 攻擊載荷注入 (Payload Injection)
  - 模擬攻擊者向輸入欄位、URL 參數或標頭發送惡意字串，藉此驗證系統是否會產生 A05 注入攻擊 或 A10 特殊狀態處理不當 的連鎖反應。
- 爬蟲與自動發現 (Spidering & Crawling)
  - 自動遍歷應用程式的所有頁面、表單與 API 節點，繪製出完整的攻擊面，確保沒有遺漏任何隱藏的介面或 A01 權限控制失效 的路徑。
- 回應分析 (Response Analysis)
  - 分析伺服器回傳的 HTTP 狀態碼與內容。如果偵測到詳細的報錯資訊，則對應 A10 資訊洩漏；若發現 Session ID 未更新，則涉及 A07 驗證失效。
- 語言無關性測試 (Language Agnostic)
  - 因為 DAST 是從外部透過 HTTP/HTTPS 協議進行通訊，它不關心後端是用 Java、Python 還是 PHP 寫的，這讓它能跨技術棧評估整體安全性。
- 驗證防禦機制有效性 (Defensive Control Verification)
  - 直接測試 WAF（網頁應用程式防火牆）或入侵偵測系統是否能有效攔截攻擊，確認 A09 安全監控 是否發揮作用。

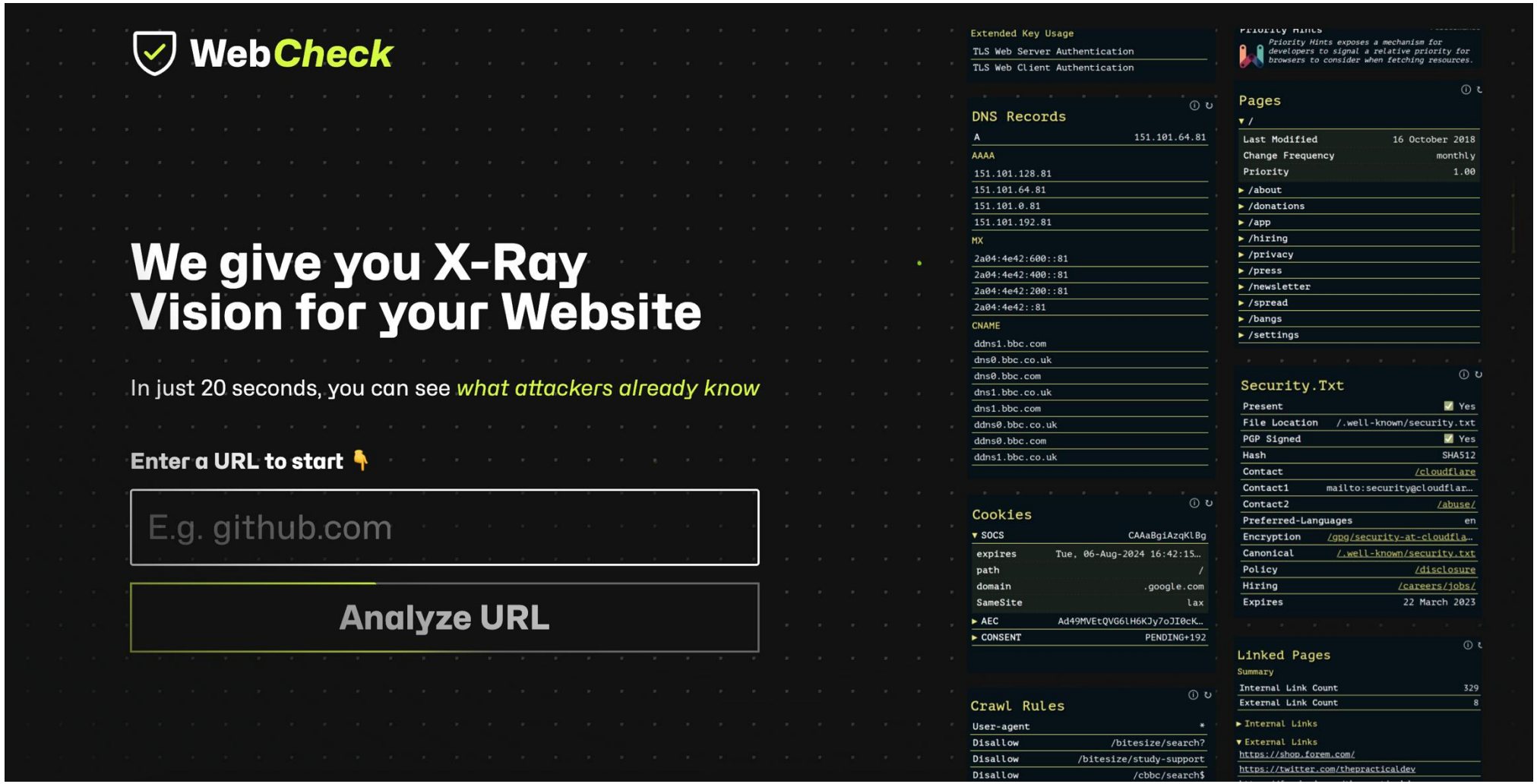
交互式應用程式安全測試（IAST）

- 內部監控與插樁 (Instrumentation)
  - 透過在執行環境（如 JVM, .NET Runtime）植入代理程式，即時監控程式碼執行過程，能精確定位漏洞在原始碼中的具體行數。
- 即時資料流追蹤 (Real-time Data Flow Tracking)
  - 監控不安全資料（來源）如何進入系統並傳遞至敏感函數（匯點），有效識別 A05 注入攻擊 與 A01 權限控制失效。
- 分析運行時配置 (Runtime Configuration Analysis)
  - 檢查應用程式在執行時的實際環境設定，例如是否開啟了不安全的框架功能或弱加密套件，對應 A02 安全配置錯誤。
- API 與第三方元件監控 (Library & API Analysis)
  - 監控應用程式與外部 Library 的交互，識別是否存在 A03 軟體供應鏈失效 或調用了具備已知漏洞的舊版元件。
- 低誤報率驗證 (Low False Positive Verification)
  - 由於 IAST 同時掌握了「攻擊請求」與「內部執行路徑」，它能確認攻擊是否真的成功觸發漏洞，顯著降低了 SAST 常見的誤報問題。
- 與 QA 測試流程整合 (Test Integration)
  - IAST 不需要像 DAST 那樣掃描整個網站，它可以在功能測試（QA）或自動化測試進行時，「順便」完成安全檢查，不增加額外的測試負擔。

應用程式安全測試技術對照表

特性	SAST	DAST	IAST
測試性質	白箱 (有原始碼)	黑箱 (無原始碼)	灰箱 (運行時監控)
掃描對象	原始碼、二進制檔	運行中的 Web 服務	執行環境 (JVM/.NET)
導入階段	開發初期 (Coding)	測試/預發布 (Staging)	測試階段 (QA/Automation)
漏洞定位	精確 (可指到程式碼行數)	模糊 (僅知哪個 URL)	精確 (程式碼行與調用鏈)
誤報率	較高 (易產生過多雜訊)	低 (測到通常就是真的)	極低 (結合運行時驗證)
主要發現	SQL 注入、硬編碼、弱加密	伺服器配置、跨站腳本	複雜資料流、第三方套件漏洞
執行成本	隨開發進行，修復成本低	需環境就緒，修復成本高	需安裝 Agent，與測試整合
對應 OWASP	A04, A05, A10	A02, A05, A07	A01, A03, A05, A08

檢查一下自己的網站



https://web-check.xyz/

OWASP 主要專案介紹

- OWASP Top 10 Risk
  - 前十大風險，做為資安意識認知的訓練用途。
- OWASP ASVS (Application Security Verification Standard)
  - 檢測標準，提供在進行滲透測試或驗收時的檢核表。
- OWASP WSTG (Web Security Testing Guide)
  - 測試指南， 提供對於 Web 應用程式和 Web 服務安全的測試實踐。
- OWASP SAMM (Software Assurance Maturity Mode)
  - 軟體保障成熟度模型，提供基於風險的軟體安全框架，協助組織評估、制定和實施軟體安全策略。

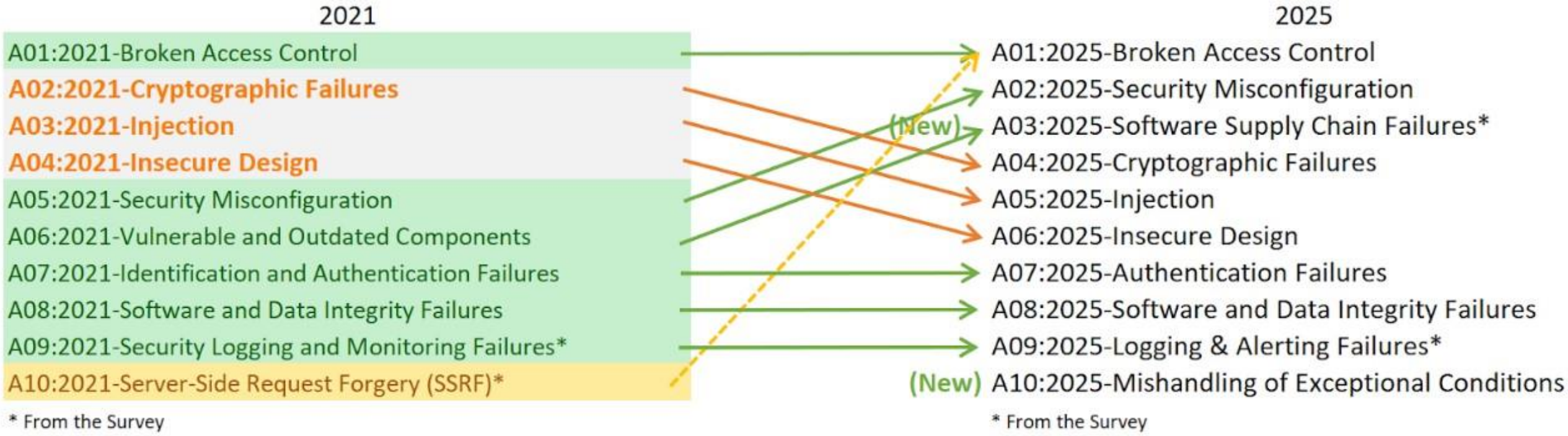
應用情境

- 產品、平台驗收 (OWASP ASVS)
  - Level 1：使用自動化工具進行基本掃描，對象多數為低風險的系統。
  - Level 2：產業常用的標準等級，涵蓋多數的控制項，適用處理個資的系統。
  - Level 3：高強度的驗證要求，多數適用於關鍵基礎設施 (政府、金融、醫療等)

- 軟體開發流程改善 (OWASP SAMM)
  - 治理 (Governance)：需要制定策略與量測指標
  - 設計 (Design)：進行軟體的威脅建模
  - 實作 (Implementation)：建立安全程式開發規範
  - 驗證 (Verification)：進行程式碼檢核與驗測
  - 維運 (Operations)：需要有漏洞管理與事件應變流程

OWASP Top 10 - 2025

- 根據 OWASP Top 10（2025）的最新調整，Web 應用程式安全風險的結構已出現明顯轉變。
- 本次更新不僅新增兩個關鍵風險類別，分別是「軟體供應鏈缺失」與「特殊情況處理不當」，同時也將過往獨立列項的伺服器端請求偽造（SSRF）風險，重新歸納至「存取控制漏洞」之中，反映出攻擊手法與防禦視角的成熟演進。



OWASP Top 10 - 2025

- 「軟體供應鏈缺失」首次躍升至前三大風險，清楚揭示現代應用系統對開源套件、第三方函式庫、雲端服務與外部 API 的高度依賴，已成為攻擊者最具成本效益的入侵路徑，一旦供應鏈任一環節遭到污染，其影響往往呈現橫向擴散與規模化風險，對企業營運與信任基礎構成重大衝擊。
- 「加密機制失效」與「注入攻擊」雖仍名列風險清單，但整體排名下滑，顯示多數組織在這些傳統漏洞類型上，已逐步建立起基本防護能力與安全開發共識。然而，這並不代表風險消失，而是防禦門檻的提升，使攻擊者轉向更隱蔽、跨系統的攻擊面。
- 新納入的「特殊情況處理不當」類別，聚焦於系統在非預期狀態下的行為風險，涵蓋錯誤處理不足、異常流程設計缺陷、邏輯失效與資源耗盡等問題，快速成為下一波 Web 應用攻擊的重要切入點。

OWASP Top 10 : 2025

- A01 存取控制失效
- A02 安全配置錯誤
- A03 軟體供應鏈失效
- A04 加密機制失效
- A05 注入攻擊
- A06 不安全設計
- A07 身分驗證失敗
- A08 軟體及資料完整性失效
- A09 安全日誌與告警失效
- A10 特殊情況處理不當

注意：
Top 10 不是驗收標準！

因應 Top 10 異動的注意事項

- 強化存取管理：落實最小權限原則並定期審核，杜絕非法授權。
- 掌握供應鏈安全：導入 SBOM (軟體物料清單)，完整追蹤並監控第三方元件風險。
- 主動防禦檢測：定期執行弱掃與滲透測試，搶先修補外部曝險點。
- 強化身份認證：強制執行 MFA (多因子驗證) 並設有登入失敗阻斷機制。
- 建立應變韌性：組織 PSIRT 團隊，建構完善的事件響應與異常處理流程。
- 持續教育訓練：定期更新資安職能培訓，確保團隊掌握最新威脅情資。

OWASP TOP10

TWCSA 台灣數位安全聯盟 Taiwan Cyber Security Alliance

A01 存取控制失效 (Broken Access Control)

- 定義
  - 存取控制失效是指應用程式未能正確限制使用者對機敏資料或功能的存取權限。這通常源於後端驗證機制不足，導致攻擊者能繞過安全限制，執行原本未獲授權的操作。例如，一般使用者可能直接修改 URL 存取管理後台，或在未經授權下查看、刪除他人的帳戶資料。此類漏洞嚴重威脅資料的機密性與完整性。
- 常見問題
  - IDOR (不安全的物件直接引用)：攻擊者透過修改參數（如 user_id=123 改為 456），在不需額外權限檢查的情況下存取他人的私人檔案或記錄。
  - 權限提升 (Privilege Escalation)：普通使用者透過漏洞獲得管理員權限（垂直提權），或是在相同權限等級下切換身份存取其他使用者資源（水平提權）。
  - 強制瀏覽與繞過檢查：攻擊者直接輸入隱藏頁面的 URL 或透過修改 HTTP 標頭（如 JWT token）來規避權限驗證流程，進而存取受限資源。
- 防禦重點
  - 實施最小權限原則 (Least Privilege)：預設情況下應拒絕所有存取請求 (Deny by Default)，僅授予使用者完成任務所必需的最基本權限。
  - 集中化且伺服器端驗證：所有權限檢查必須在受信任的伺服器端執行，而非依賴前端 UI 隱藏按鈕。建議建立統一的權限管控組件供全系統重複呼叫。
  - 落實日誌記錄與限制：對所有存取控制失效的嘗試進行詳細日誌記錄與監控，並針對 API 介面實施流量限制 (Rate Limiting)，以防範自動化工具的大規模攻擊。

A01 存取控制失效 (Broken Access Control)


```
{
 "timestamp": "2025-05-20T14:30:15.123Z",
 "event_type": "ACCESS_CONTROL_FAILURE",
 "severity": "CRITICAL",
 "actor": {
  "user_id": "user_8892",
  "username": "johndoe_test",
  "ip_address": "203.0.113.42",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)..."
 },
 "request": {
  "method": "GET",
  "url": "https://api.myapp.com",
  "request_id": "req-af23-9901"
 },
 "result": {
  "status_code": 403,
  "reason": "Insecure Direct Object Reference (IDOR) detected",
  "details": "User tried to access an order belonging to another organization."
 },
 "context": {
  "resource_owner_id": "admin_root",
  "action": "READ_ORDER_DETAIL"
 }
}
```


- event_type (事件類型)：明確標記為「權限控制失效」，方便後端 SIEM 系統直接分類。
- actor (行為者)：記錄了是「誰」從「哪裡」發動請求，包括 User ID 與來源 IP。
- status_code: 403：這是標準的「禁止存取」狀態碼，是 A01 監控的最重要指標。
- details (詳細原因)：指出這是一個 IDOR (不安全的物件引用) 嘗試，因為使用者 user_8892 想要看 admin_root 的訂單。
- request_id：這是關聯式 ID，可以讓工程師在追蹤分散式系統的 Log 時，串接起整個操作流程。

A02 安全配置錯誤（Security Misconfiguration）

- 定義
  - 此風險發生於系統、應用程式或雲端環境未進行正確的安全設定。開發或維運人員常為了方便而保留預設密碼、開啟非必要的服務與連接埠，或在正式環境中遺留詳細的報錯訊息。由於現代軟體高度依賴框架與雲端基礎設施，任何細微的配置疏漏都可能成為攻擊者入侵、獲取系統資訊或篡改資料的捷徑。
- 常見問題
  - 預設配置未更改：系統安裝後仍沿用廠家預設的帳號密碼（如 admin/admin），或保留示範用的範例頁面與過時的測試帳戶。
  - 過度暴露錯誤資訊：系統出錯時直接在前端顯示詳細的堆疊追蹤（Stack Traces）或環境變數，讓攻擊者能藉此掌握系統架構、版本與邏輯漏洞。
  - 雲端儲存權限大開：雲端空間（如 S3 Bucket）或資料庫設定錯誤，導致權限設定為「公開存取」，使得機敏檔案或備份資料直接暴露於網際網路上。
- 防禦重點
  - 自動化增加配置：建立自動化腳本或映像檔，移除所有不必要的元件、服務與範例內容，並確保各環境（開發、測試、正式）的配置一致且安全。
  - 落實最小化原則：僅開啟業務運作必需的連接埠與功能，並關閉詳細的錯誤回傳機制，改以通用的錯誤訊息取代，並將詳細記錄存於伺服器後端。
  - 持續性配置審核：定期執行自動化掃描，檢查各層級（App、DB、Network、Cloud）的設定是否符合安全標準，及時修復因更新或人為操作產生的配置漂移。

A02 安全配置錯誤（Security Misconfiguration）


```
{
  "timestamp": "2025-06-12T09:15:30.456Z",
  "event_type": "SECURITY_CONFIG_EXPOSURE",
  "severity": "HIGH",
  "actor": {
    "username_attempted": "admin",
    "ip_address": "45.122.33.107",
    "location": "Unknown/Proxy",
    "user_agent": "python-requests/2.28.1"
  },
  "request": {
    "method": "POST",
    "url": "https://internal.company.com",
    "protocol": "HTTP/1.1"
  },
  "result": {
    "status_code": 401,
    "reason": "Default Credentials Usage Attempt",
    "details": "Access denied for default system account 'admin'. This account should be disabled or renamed."
  },
  "context": {
    "environment": "PRODUCTION",
    "component": "Admin_Management_Interface",
    "config_vulnerability": "Unchanged_Default_Password"
  }
}
```


- username_attempted: "admin"：記錄了攻擊者嘗試使用的帳號。在安全的系統中，「admin」或「root」等預設名稱應該被停用或更改，出現此紀錄代表系統可能仍保有預設配置。
- user_agent: "python-requests"：顯示這是一個自動化腳本而非正常瀏覽器，這是掃描器在尋找未更改預設設定的典型徵兆。
- config_vulnerability：標註了具體的配置弱點（未更改的預設密碼），這能讓維運人員快速鎖定需要強化的設定項。
- environment: "PRODUCTION"：區分環境非常重要，因為有些除錯配置在測試環境（Staging）是允許的，但在正式環境（Production）則是嚴重的漏洞。

A03 軟體供應鏈失效（Software Supply Chain Failures）

- 定義
  - 此風險關注軟體從開發到部署過程中，所有第三方元件、庫（Library）及供應鏈環節的安全。當企業使用含有惡意程式碼或已知漏洞的第三方插件、依賴套件，或開發流程（CI/CD）遭篡改時，攻擊者便能「借刀殺人」，透過受信任的管道滲透進企業內部。這類攻擊隱蔽性高且影響範圍極廣，是現代軟體開發面臨的重大挑戰。
- 常見問題
  - 依賴混淆攻擊 (Dependency Confusion)：攻擊者在公開倉庫上傳與企業內部私有套件同名的惡意套件，誘使開發環境自動抓取錯誤的「更新版」惡意程式碼。
  - 不安全的 CI/CD 流程：建置環境或自動化部署工具權限管理不當，導致攻擊者能在程式碼打包發布前，偷偷植入後門（Backdoor）。
  - 第三方套件投毒：駭客接管了維護頻率低的開源專案，並在合法更新中加入惡意程式，使所有引用該套件的企業被迫感染。
- 防禦重點
  - 導入軟體物料清單 (SBOM)：完整盤點系統中使用的所有第三方元件清單與版本，確保能快速對應最新漏洞資訊，並進行合規性檢查。
  - 建立私有鏡像倉庫與驗證機制：強制使用內部的私有套件庫，並實施數位簽章（Digital Signing）驗證，確保程式碼在傳輸與建置過程中未被竄改。
  - 強化流水線完整性監控：對開發工具（IDE）、版本控制系統及 CI/CD 伺服器實施嚴格的存取控制與異動監控，確保軟體供應鏈的每個環節皆可追蹤且受信任。

A03 軟體供應鏈失效（Software Supply Chain Failures）


```
{
 "timestamp": "2025-07-15T10:20:45.789Z",
 "event_type": "SUPPLY_CHAIN_INTEGRITY_FAILURE",
 "severity": "CRITICAL",
 "pipeline": {
  "stage": "DEPENDENCY_FETCH",
  "job_id": "build-worker-772",
  "runner_ip": "10.0.5.122"
 },
 "dependency": {
  "package_manager": "npm",
  "package_name": "useful-utility-lib",
  "version": "2.4.1",
  "source_url": "https://registry.npmjs.org"
 },
 "result": {
  "status": "SIGNATURE_VERIFICATION_FAILED",
  "reason": "Digital signature does not match the public key from the trusted provider.",
  "expected_hash":
"sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "actual_hash": "sha256:8f43a2c5a1a1f1a1f1a1f1a1f1a1f1a1f1a1f1a1f1a1f1a1f1a1f1a1f1a1f1a1f1a1f1a1"
 },
 "action_taken": "BUILD_ABORTED",
 "context": {
  "vulnerability_scan_status": "FAILED",
  "sbom_update": "PENDING"
 }
}
```


- event_type: "SUPPLY_CHAIN_INTEGRITY_FAILURE"：明確定義為供應鏈完整性失效，這與單純的程式碼錯誤（Bug）不同。
- SIGNATURE_VERIFICATION_FAILED：這是 A03 最關鍵的監控點。雖然下載了正確名稱的套件，但內容（Hash 值）與預期不符，代表套件已被掉包。
- action_taken: “BUILD_ABORTED”：展示了安全的設計，一旦發現供應鏈異常，應立即中斷建置，防止惡意程式碼進入正式環境。
- package_name & version：精確記錄受影響的元件資訊，方便資安人員追蹤該套件是否也被其他專案引用。

A04 加密機制失效（Cryptographic Failures）

- 定義
  - 此風險涉及敏感資料在儲存（Rest）或傳輸（Transit）過程中未受到足夠的保護。當系統使用過時的加密演算法、弱金鑰，或根本未對機敏資訊（如密碼、個資、信用卡號）進行加密時，攻擊者一旦攔截或竊取資料庫，即可輕易讀取原始內容。這類失效通常是導致大規模資料外洩與違反個資保護法（如 GDPR）的核心原因。
- 常見問題
  - 傳輸過程明文傳輸：系統仍在使用不安全的 HTTP、FTP 或 SMTP 協議，或者 TLS 設定版本過舊（如 SSLv3 或 TLS 1.0），導致攻擊者可透過中間人攻擊（MITM）竊聽資料。
  - 使用弱加密演算法：使用已被破解的雜湊函數（如 MD5 或 SHA-1）儲存密碼，或使用對稱式弱加密（如 DES），使得攻擊者能輕易透過暴力破解或彩虹表還原資料。
  - 金鑰管理不當：將加密金鑰直接寫死（Hardcoded）在程式碼中、存放在公開的 Git 儲存庫，或未定期更換金鑰，導致加密保護形同虛設。
- 防禦重點
  - 全面強制加密傳輸：禁用所有明文傳輸協議，全面導入 TLS 1.2（或更高版本）並配置 HSTS (HTTP 嚴格傳輸安全)，確保瀏覽器強制使用安全連線。
  - 選用強健的演算法與鹽值：密碼儲存應使用具備工作因子（Work Factor）的演算法如 Argon2 或 bcrypt，並結合隨機「鹽值 (Salt)」，增加破解難度。
  - 安全金鑰管理：使用專業的金鑰管理系統 (KMS) 或硬體安全模組 (HSM) 來存放金鑰，並嚴格限制存取權限與執行定期輪替機制。

A04 加密機制失效（Cryptographic Failures）


```
{
  "timestamp": "2025-08-05T16:45:10.888Z",
  "event_type": "CRYPTOGRAPHIC_PROTOCOL_FAILURE",
  "severity": "MEDIUM",
  "connection": {
    "source_ip": "198.51.100.12",
    "destination_port": 443,
    "request_url": "https://payments.company.com"
  },
  "crypto_details": {
    "protocol_attempted": "TLSv1.0",
    "cipher_suite": "TLS_RSA_WITH_AES_128_CBC_SHA",
    "reason": "DEPRECATED_PROTOCOL_REJECTED"
  },
  "result": {
    "status": "CONNECTION_TERMINATED",
    "error_message": "The server requires TLS 1.2 or higher for secure data
transmission."
  },
  "context": {
    "application_layer": "Payment_Gateway",
    "compliance_standard": "PCI-DSS_v4.0_Requirement"
  }
}
```


- protocol_attempted: "TLSv1.0"：TLS 1.0 已被證實存在嚴重漏洞。記錄下這個資訊，可以幫助團隊判斷是否還有舊型客戶端（如過時的瀏覽器或舊系統）正在嘗試進行不安全連線。
- DEPRECATED_PROTOCOL_REJECTED：明確標註是因為「協議過時」而被拒絕。這符合 A04 防禦重點中提到的「禁用不安全協議」。
- compliance_standard: "PCI-DSS"：將技術錯誤與合規標準（如支付卡產業安全標準）連結，這對於處理金融或個資的開發者來說非常有說服力。
- CONNECTION_TERMINATED：展示了「失敗即安全」的原則與其降級加密（Downgrade Attack）讓攻擊者竊聽，不如直接斷開連線。

A05 注入攻擊（Injection）

- 定義
  - 注入攻擊發生於應用程式將不可信的外部資料，直接作為指令或查詢語句的一部分傳送給解釋器（Interpreter）執行。當程式未對輸入進行嚴格過濾或轉譯時，攻擊者可藉此操控資料庫查詢（SQL）、執行作業系統指令（OS Command）或竄改輕量級目錄存取協議（LDAP）。這類攻擊通常能直接繞過身份驗證，進而讀取、修改甚至刪除系統核心資料。
- 常見問題
  - SQL 注入 (SQL Injection)：攻擊者在登入欄位或 URL 參數輸入特定語法（如 ' OR 1=1 -- ），誘使資料庫回傳所有使用者帳密或刪除整張資料表。
  - 跨網站指令碼 (XSS)：這被歸類為注入攻擊的一種（資料注入瀏覽器執行），攻擊者將惡意腳本植入網頁，藉此竊取其他使用者的 Cookie 或 Session。
  - 指令注入 (Command Injection)：應用程式呼叫系統殼層（Shell）處理檔案或網路請求時，若未過濾特殊字元，攻擊者可藉此下達 rm -rf 等惡意指令控制伺服器。
- 防禦重點
  - 使用參數化查詢 (Parameterized Queries)：這是防禦 SQL 注入的最有效手段。透過預編譯語句（Prepared Statements）確保資料與指令分離，使輸入內容僅被視為純文字而非指令。
  - 實施嚴格的白名單過濾：對所有使用者輸入進行驗證，僅允許符合預期格式（如數字、日期、特定字元）的資料通過，拒絕任何含有特殊控制字元的請求。
  - 採用安全 API 與跳脫處理 (Escaping)：盡量使用內建具備防禦機制的框架或庫，並在資料輸出至 HTML、JavaScript 或 SQL 語法前執行適當的編碼轉義，消除資料的指令特性。

A05 注入攻擊（Injection）


```
{
 "timestamp": "2025-09-10T22:05:40.321Z",
 "event_type": "INJECTION_ATTEMPT_DETECTED",
 "severity": "CRITICAL",
 "attack_vector": {
  "type": "SQL_INJECTION",
  "parameter": "user_id",
  "payload": "101' OR '1'='1' --",
  "location": "QUERY_STRING"
 },
 "actor": {
  "ip_address": "185.220.101.44",
  "user_agent": "sqlmap/1.6.7#stable (https://sqlmap.org)",
  "request_id": "req-9988-inject"
 },
 "request": {
  "method": "GET",
  "url": "https://shop.company.com",
  "headers": {
   "host": "shop.company.com",
   "accept": "*/*"
  }
 },
 "result": {
  "status": "BLOCKED",
  "reason": "MALICIOUS_PATTERN_MATCH",
  "filter_rule": "SQL_TAUTOLOGY_CHECK"
 },
 "context": {
  "module": "USER_PROFILE_SERVICE",
  "database_engine": "POSTGRESQL"
 }
}
```


- payload: "101' OR '1'='1' --"：記錄下攻擊者輸入的原貌。這是一個典型的「恆真式」攻擊，意圖繞過權限檢查獲取所有資料。
- user_agent: "sqlmap"：這是非常重要的識別。sqlmap 是著名的自動化 SQL 注入工具，出現在 Log 代表這不是誤擊，而是有計畫的漏洞掃描。
- type: "SQL_INJECTION"：明確分類。注入攻擊也可能發生在 OS 指令或 LDAP，精確分類有助於應變人員決定修補哪個組件。
- status: "BLOCKED"：展示了防護措施（如參數化查詢或 WAF）發揮作用，攻擊指令僅停留在 Log 中，並未進入資料庫執行。

A06 不安全設計（Insecure Design）

- 定義
  - 不安全設計關注的是「設計缺陷」而非「實作漏洞」。即使程式碼寫得再完美，如果最初的架構邏輯就有問題，系統依然是不安全的。這強調在開發生命週期（SDLC）的最早期，就必須導入威脅建模、安全設計模式與參考架構。這類問題無法透過事後的工具掃描或修補程式碼來解決，通常需要更動整體的業務邏輯或系統流程。
- 常見問題
  - 密碼重設邏輯缺陷：例如重設密碼的驗證碼過於簡單（如 4 位純數字）且未限制嘗試次數，或將驗證碼回傳至前端供檢查，導致攻擊者可輕易暴力破解。
  - 電商邏輯漏洞：在結帳流程中，系統僅在前端計算總金額，未在後端重新核對商品單價，導致使用者可透過攔截封包修改金額，以 1 元購買高價商品。
  - 缺乏防自動化機制：設計功能時未考量 API 濫用，導致查詢介面可被爬蟲程式大規模抓取個資，或登入頁面被自動化工具進行帳號填充（Credential Stuffing）。
- 防禦重點
  - 導入威脅建模 (Threat Modeling)：在設計階段模擬攻擊者的思維，識別系統可能的弱點（如：資料流向、信任邊界），並在開發前制定相應的防護策略。
  - 使用安全的設計模式：參考已驗證的安全架構（如：MVC 安全控制項），並落實「失敗即安全」(Fail Securely) 原則，確保系統在異常狀態下會自動切換至限制最高的狀態。
  - 建立開發安全指南：為開發團隊提供標準的安全設計範本，包含統一的身份驗證、授權、日誌與資料加密流程，減少因個人判斷差異導致的邏輯缺失。

A06 不安全設計（Insecure Design）

- 以「忘記密碼」為例

設計階段	錯誤設計 (Insecure Design)	正確設計 (Secure Design)
帳號確認	顯示「該帳號不存在」，讓攻擊者可用來枚舉 (Enumeration) 有效的使用者清單。	無論帳號是否存在，統一顯示「若帳號存在，系統已發送重設郵件」。
驗證媒介	使用「安全提問」（如：你出生在那個城市？），這類資訊極易透過社交工程取得。	透過信箱或簡訊發送一次性、高強度且具時效性的隨機連結/驗證碼。
驗證機制	驗證碼僅 4 位數字且無次數限制，攻擊者可在幾分鐘內暴力破解。	驗證碼具備複雜度，且限制錯誤嘗試次數（例如 5 次後鎖定），並導入圖形驗證。
傳輸邏輯	為了方便，將「正確的驗證碼」藏在網頁原始碼 (HTML) 或 API 回傳值中供前端檢查。	驗證碼僅存於伺服器端 (Server-side)，前端僅負責傳送使用者輸入的結果。
重設後狀態	重設密碼後，原本在其他裝置登入的 Session 依然有效。	密碼修改成功後，強制登出 (Global Logout) 該帳號在所有裝置的連線。

A06 不安全設計（Insecure Design）


```
{
  "timestamp": "2025-10-12T11:45:22.910Z",
  "event_type": "BUSINESS_LOGIC_INTEGRITY_VIOLATION",
  "severity": "HIGH",
  "actor": {
    "user_id": "cust_5543",
    "ip_address": "114.34.88.12",
    "session_id": "sess_v99e2r1"
  },
  "request": {
    "method": "POST",
    "url": "https://shop.myapp.com",
    "payload": {
      "item_id": "PRO_999",
      "submitted_price": 1.0,
      "currency": "TWD"
    }
  },
  "result": {
    "status": "REJECTED",
    "reason": "PRICE_MISMATCH_DETECTED",
    "details": "Client submitted price (1.0) does not match system master price (29900.0) for
item PRO_999."
  },
  "context": {
    "vulnerability_type": "Insecure_Business_Process",
    "check_point": "Server_Side_Price_Validation"
  }
}
```


- BUSINESS_LOGIC_INTEGRITY_VIOLATION：這不是程式當掉，而是使用者的行為違反了商業規則。這類日誌對於抓出試圖鑽漏洞的「進階使用者」非常有效。
- submitted_price vs system_master_price：記錄下「用戶給的」跟「系統規定的」差異。這證明了系統在後端有進行二次核對，而非盲目相信前端傳來的數據（這就是安全設計的展現）。
- REJECTED：雖然交易格式正確，但因為邏輯不符而被拒絕，防止了企業損失。
- vulnerability_type: "Insecure_Business_Process"：標記這是一個流程設計上的威脅，提醒開發者檢查是否還有其他結帳路徑漏掉了後端驗證。

A07 身分驗證失效（Authentication Failures）

- 定義
  - 此風險發生於應用程式未能正確確認使用者的身份。如果驗證機制存在漏洞，攻擊者可以輕易偽裝成合法使用者，接管個人帳戶甚至獲得管理員權限。這類問題通常涉及密碼保護不足、工作階段（Session）管理不當，或缺乏多重因素驗證等防禦手段，是身份盜用與帳號填充攻擊（Credential Stuffing）的主因。
- 常見問題
  - 帳號填充與暴力破解：應用程式未限制登入嘗試次數，導致攻擊者可以使用自動化工具，拿著從其他網站外洩的帳號密碼清單不斷嘗試登入。
  - 不安全的密碼政策：允許使用者設定過於簡單的密碼（如 123456），或未強制執行複雜度要求，且不具備對抗常見密碼字典的檢查機制。
  - 工作階段管理漏洞：登入後產生的 Session ID 固定不變（固定攻擊）、未在登出後立即失效，或是將敏感的驗證資訊暴露在 URL 中。
- 防禦重點
  - 強制執行多因子驗證 (MFA)：這是防範驗證失效的最有效手段。即使密碼外洩，攻擊者若無第二層驗證碼（如手機 App、硬體金鑰）也無法進入系統。
  - 實施登入失敗阻斷機制：針對連續失敗的登入嘗試執行速率限制 (Rate Limiting) 或暫時鎖定帳號，並在前端加入圖形驗證碼（CAPTCHA）以防止機器人攻擊。
  - 強化工作階段安全性：在登入成功後核發新的隨機 Session ID，並確保所有 Cookie 具備 HttpOnly（防止腳本竊取）與 Secure（僅限加密傳輸）屬性。

A07 身分驗證失效（Authentication Failures）

- MFA 驗證類型安全性比較表

驗證類型	運作方式	優點	缺點/風險	安全等級
簡訊 (SMS) / 語音	系統發送 6 位數代碼至手機。	普及度最高，使用者不需要額外安裝 App。	易遭受 SIM 卡劫持 (SIM Swapping) 或攔截簡訊攻擊；收訊不良時無法使用。	低 (Basic)
OTP App (如 Google Authenticator)	每個固定週期產生一個動態驗證碼。	無需網路即可產生代碼；抗簡訊截獲。	手機遺失需重新設定；仍可能被「釣魚網站」騙取手動輸入。	中 (Standard)
推播通知 (Push Notification)	App 彈出視窗點擊「允許」。	體驗最快，無需手動輸入。	可能導致 MFA 疲勞攻擊 (攻擊者瘋狂發送通知直到使用者不耐煩按錯)。	中 (Standard)
FIDO2 / 硬體金鑰 (如 Yubikey)	插入實體金鑰並觸碰。	物理隔離，能徹底防範釣魚網站與中間人攻擊。	需額外購買硬體；若金鑰遺失需有備援方案。	高 (High)
Hybrid 混合型態	整合軟體與硬體特性。	複雜的驗證機制，實現零信任身分驗證框架	整合複雜度較高，須考量中控平台與風險管理方案。	最高 (Advanced High)

A07 身分驗證失效（Authentication Failures）


```
{
 "timestamp": "2025-11-20T23:10:05.123Z",
 "event_type": "AUTHENTICATION_FAILURE_BRUTE_FORCE",
 "severity": "HIGH",
 "actor": {
  "username_attempted": "manager_alice",
  "ip_address": "192.0.2.1",
  "location": "Unknown/Proxy",
  "user_agent": "Mozilla/5.0 (compatible; CustomBruteForcer/1.0)"
 },
 "request": {
  "method": "POST",
  "url": "https://auth.company.com",
  "mfa_status": "NOT_REACHED"
 },
 "result": {
  "status": "ACCOUNT_LOCKED",
  "reason": "EXCEEDED_MAX_LOGIN_ATTEMPTS",
  "details": "User 'manager_alice' failed login 10 times in 2 minutes. Account locked for 30
minutes."
 },
 "context": {
  "failure_count_in_window": 10,
  "lockout_duration_seconds": 1800,
  "auth_method": "PASSWORD_ONLY"
 }
}
```


- AUTHENTICATION_FAILURE_BRUTE_FORCE：明確標註為暴力破解嘗試，有別於一般使用者的偶發忘記密碼。
- mfa_status: "NOT_REACHED"：顯示攻擊者連第一關「密碼」都沒過，所以還沒進入 MFA 驗證階段。
- ACCOUNT_LOCKED & EXCEEDED_MAX_LOGIN_ATTEMPTS：這是 A07 最重要的防禦行動-「自動化鎖定」。日誌詳細記錄了觸發條件（2 分鐘內失敗 10 次）。
- lockout_duration_seconds：記錄鎖定時長。這對於後台管理員判斷是否需要介入解除鎖定或聯繫使用者非常有用。

A08 軟體與資料完整性失效（Software and Data Integrity Failures）

- 定義
  - 此風險聚焦於程式碼與資料在傳輸、儲存或更新過程中，缺乏有效的完整性驗證。當系統信任來自不可信來源的軟體更新、資料流或反序列化物件，且未確認其是否遭到竄改時，攻擊者便能植入惡意程式碼。這類失效不僅發生在應用程式本身，也常見於 CI/CD 流水線中，是導致大規模供應鏈入侵與勒索軟體植入的關鍵路徑。
- 常見問題
  - 不安全的軟體更新：應用程式在下載更新檔或套件時，未檢查數位簽章（Digital Signature），導致攻擊者可透過中間人攻擊（MITM）替換為含後門的偽造版本。
  - 不安全的反序列化 (Insecure Deserialization)：系統直接接收並轉換來自外部的物件資料，攻擊者可藉由精心構造的惡意物件，在反序列化過程中執行遠端指令（RCE）。
  - CI/CD 流水線配置不當：自動化建置流程中缺乏權限管控，使得攻擊者能未經審核地修改程式碼或腳本，將惡意邏輯直接打包進正式環境的產品中。
- 防禦重點
  - 採用數位簽章與驗證機制：確保所有軟體套件、更新檔及機敏資料流都經過加密簽章。在安裝或執行前，系統必須強制驗證來源的真實性與內容的完整性。
  - 嚴格限制反序列化資料來源：避免直接反序列化來自不可信客戶端的資料。若必須使用，應採用限制較多的資料格式（如純文字 JSON），或在反序列化前進行嚴格的類型檢查。
  - 強化流水線審核流程：在 CI/CD 流程中加入程式碼異動審核（Code Review）與自動化掃描，並確保建置環境與發布管道具備物理或邏輯上的隔離。

A03 關注的是「來源」是否可靠，而 A08 關注的是「過程」中是否被動了手腳。

A08 軟體與資料完整性失效（Software and Data Integrity Failures）


```
{
  "timestamp": "2025-12-01T14:22:33.999Z",
  "event_type": "DATA_INTEGRITY_VIOLATION",
  "severity": "CRITICAL",
  "actor": {
    "ip_address": "45.33.22.11",
    "user_id": "anonymous_guest",
    "session_id": "sess_integrity_9901"
  },
  "request": {
    "method": "POST",
    "url": "https://api.myapp.com",
    "content_type": "application/x-java-serialized-object"
  },
  "result": {
    "status": "REJECTED",
    "reason": "UNTRUSTED_DESERIALIZATION_ATTEMPT",
    "details": "Blocked attempt to deserialize class
'org.apache.commons.collections.functors.InvokerTransformer' which is not in the allow-list."
  },
  "context": {
    "vulnerability_type": "Insecure_Deserialization",
    "payload_hash": "sha256:7d8e9f...",
    "action_taken": "OBJECT_DISCARDED"
  }
}
```


- DATA_INTEGRITY_VIOLATION：明確標註為資料完整性受損，這代表傳入的資料「長得不對」或「含有危險結構」。
- UNTRUSTED_DESERIALIZATION_ATTEMPT：這是 A08 最核心的防禦點。攻擊者試圖讓系統解析一個不被允許的類別（如 InvokerTransformer），這通常是為了發動遠端指令執行（RCE）。
- OBJECT_DISCARDED：展示了「拒絕不完整或不可信資料」的防禦動作，防止惡意物件被載入記憶體。
- payload_hash：記錄惡意載荷的特徵值，方便資安團隊在不同伺服器間比對是否遭受同樣的自動化攻擊。

A09 安全日誌與告警失效（Security Logging and Alerting Failures）

- 定義
  - 此風險是指應用程式未能記錄關鍵的安全事件，或紀錄不全、監控不足，導致攻擊行為無法被及時偵測與回應。如果缺乏有效的日誌紀錄（Logging）與即時監控（Monitoring），攻擊者可以在系統中長期潛伏（Dwell Time）而不被發現。根據統計，許多企業在遭受入侵後，往往需要數月時間才能從日誌中察覺異狀，這大大增加了資料外洩的損害程度。
- 常見問題
  - 關鍵事件未記錄：登入失敗、權限變更、高風險交易或伺服器端錯誤（500 Error）等重要安全事件完全沒有紀錄，導致事後無法進行鑑識分析。
  - 日誌內容無意義或過多雜訊：日誌僅記錄「存取錯誤」，卻未包含時間戳記、使用者識別碼或來源 IP；或是產出過多無關資訊，淹沒了真實的攻擊警告。
  - 監控與告警機制缺失：雖然有日誌，但沒有自動化工具即時掃描異常（如短時間內大量登入失敗），導致攻擊發生時管理員毫無察覺。
- 防禦重點
  - 建立完整的稽核追蹤 (Audit Trail)：確保所有與身份驗證、存取控制及伺服器端驗證相關的事件均被詳細紀錄，包含：誰、在何時、做了什麼、結果為何。
  - 確保日誌的完整性與不可篡改性：將日誌即時傳送至獨立的、唯讀的中心化日誌伺服器（如 ELK 或 SIEM 系統），防止攻擊者在入侵後刪除本地日誌以湮滅證據。
  - 導入自動化告警機制：設定門檻值（Threshold），針對異常行為（如偵測到 SQL 注入嘗試或暴力破解）即時發出告警，並建立標準的事件應變流程 (IRP)。

A09 安全日誌與告警失效（Security Logging and Alerting Failures）


```
{
  "timestamp": "2025-12-15T03:45:12.001Z",
  "event_type": "LOG_INTEGRITY_TAMPER_ATTEMPT",
  "severity": "CRITICAL",
  "actor": {
    "user_id": "local_admin_01",
    "ip_address": "127.0.0.1",
    "session_id": "direct_ssh_access",
    "process_id": 4502
  },
  "operation": {
    "action": "DELETE_LOG_FILE",
    "target_file": "/var/log/myapp/audit_2025_12.log",
    "tool_used": "/usr/bin/rm"
  },
  "result": {
    "status": "DENIED",
    "reason": "FILE_IMMUTABILITY_POLICY_ACTIVE",
    "details": "Attempt to delete audit log was blocked by file system immutable flag (chattr
+i)."
  },
  "context": {
    "vulnerability_type": "Log_Tampering_Attempt",
    "monitoring_system": "SIEM_AGENT_01",
    "action_taken": "SECURITY_ALERT_SENT"
  }
}
```


- LOG_INTEGRITY_TAMPER_ATTEMPT：明確記錄這是一次對日誌完整性的挑釁。如果 A09 失敗，這條日誌可能根本不會產生或被刪除。
- FILE_IMMUTABILITY_POLICY_ACTIVE：展示了高階防禦，將日誌設為「不可變 (Immutable)」，即使是管理者也無法輕易刪除，這是對抗內部威脅或已取得權限攻擊者的關鍵。
- SECURITY_ALERT_SENT：強調了監控的即時性。當有人動到日誌時，系統必須立刻發出告警，而不是等隔天人工檢查。
- action: "DELETE_LOG_FILE"：記錄下攻擊者的手法，這對於事故調查 (Forensics) 至關重要。

A10 特殊狀態處理不當（Mishandling of Exceptional Conditions）

- 定義
  - 此風險是指應用程式在遇到錯誤（Errors）、例外（Exceptions）或資源耗盡等特殊狀態時，未能以安全的方式處理。當程式碼缺乏完善的例外處理機制，系統可能會洩漏敏感的技術資訊（如資料庫結構、環境變數），或者直接崩潰導致服務中斷（DoS）。更嚴重的狀況是，系統在出錯後自動切換至「不安全模式」，讓攻擊者能繞過原本的防禦檢查。
- 常見問題
  - 詳細錯誤訊息外洩：在生產環境中顯示詳細的堆疊追蹤（Stack Traces）或報錯頁面，內容包含資料庫欄位名稱、程式庫版本或內部網路路徑。
  - 失敗後開啟權限 (Fail-Open)：安全檢查流程在遇到異常（如記憶體不足、超時）時，預設改為「通過」而非「攔截」，導致攻擊者可藉由製造壓力來繞過驗證。
  - 資源耗盡未處理：未對輸入長度、檔案大小或請求頻率設定上限，導致單一異常請求即可耗盡伺服器 CPU 或記憶體，造成系統癱瘓。
- 防禦重點
  - 落實「失敗即安全」(Fail-Safe)：確保所有安全驗證邏輯在遇到任何未預期的異常時，預設狀態均為「拒絕存取」。
  - 統一錯誤處理機制：建立全局的例外處理器（Global Exception Handler），對使用者僅顯示模糊、友好的通用訊息（如「系統忙碌中」），並將詳細技術內容記錄在後端日誌。
  - 資源配額與輸入限制：針對 API 請求與資料處理實施嚴格的超時（Timeout）限制與大小管制，防止因處理超大型或格式錯誤的資料而導致系統資源枯竭。

A10 特殊狀態處理不當（Mishandling of Exceptional Conditions）


```
{
 "timestamp": "2025-12-25T18:30:45.002Z",
 "event_type": "EXCEPTION_HANDLED_SAFELY",
 "severity": "WARNING",
 "actor": {
  "user_id": "cust_4432",
  "ip_address": "210.61.12.34",
  "request_id": "req-xyz-7788"
 },
 "exception_details": {
  "type": "DatabaseConnectionTimeout",
  "stack_trace_id": "st-99201-abc",
  "internal_error_code": "DB_ERR_009",
  "actual_message": "Connection limit reached for postgres_cluster_01; Pool size: 100"
 },
 "result": {
  "status_to_user": 503,
  "message_to_user": "Service temporarily unavailable. Please try again later.",
  "is_technical_info_leaked": false
 },
 "context": {
  "vulnerability_type": "Information_Exposure_Prevention",
  "module": "PAYMENT_PROCESSOR",
  "action_taken": "LOG_AND_GENERIC_RESPONSE"
 }
}
```


- EXCEPTION_HANDLED_SAFELY：明確表示這是一個被捕獲並妥善處理的例外，而非造成系統崩潰或洩漏資訊。
- message_to_user vs actual_message：這是 A10 的防禦核心。給使用者的訊息非常模糊（服務忙碌），而真正的技術細節（資料庫連線池滿了）只留在後端日誌。
- is_technical_info_leaked: false：這是一個安全指標，用來確認此錯誤處理流程符合資安規範。
- stack_trace_id：日誌中不直接印出整串程式碼堆疊（Stack Trace），而是提供一個 ID，讓開發者去專用的錯誤追蹤系統查詢，避免 Log 檔案過於臃腫且具備層次化管理。

OWASP Top 10:2025 風險與防禦對照表

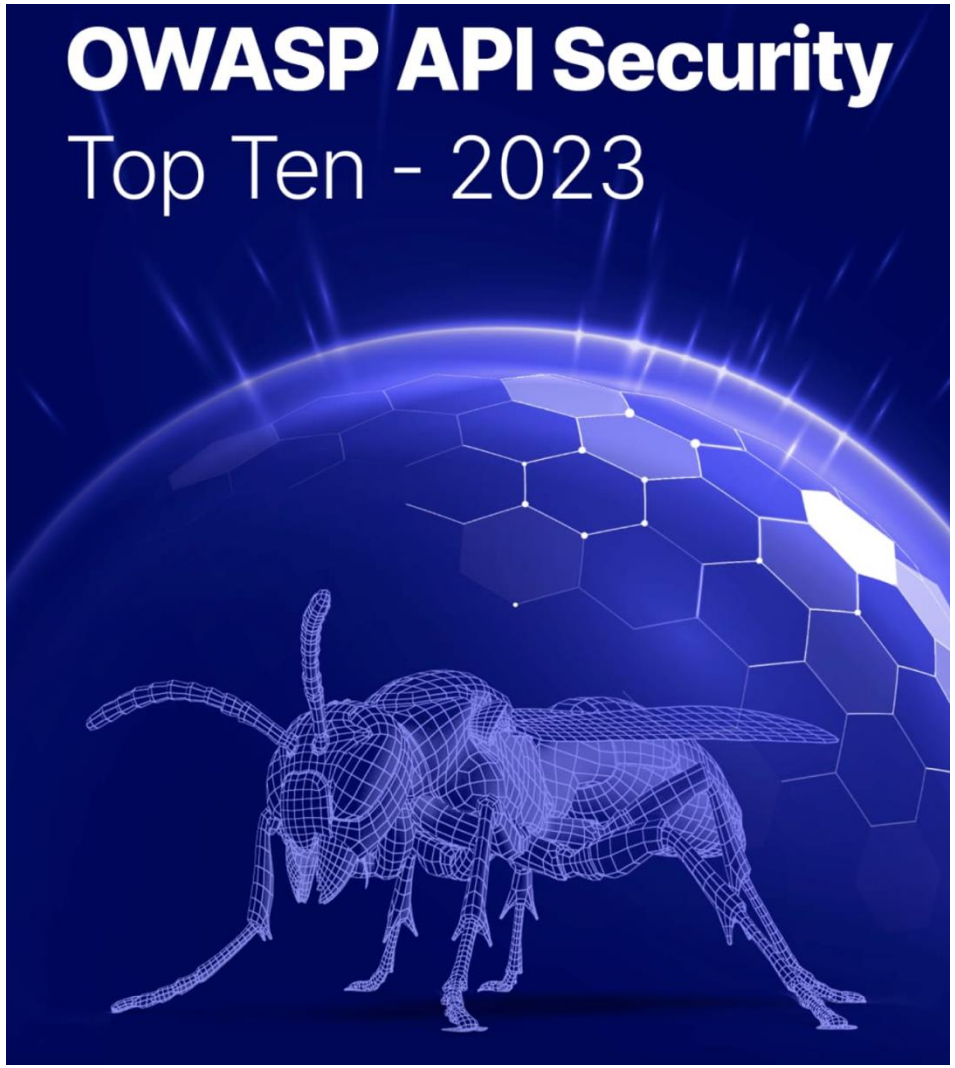
Top	名稱	核心概念	防禦重點
A01	權限控制失效	使用者存取了不該看的資料或功能	最小權限、伺服器端驗證
A02	安全配置錯誤	系統、雲端或框架設定不當留後門	自動化配置、預設拒絕
A03	軟體供應鏈失效	第三方套件或開發流程被加料	SBOM、數位簽章、套件稽核
A04	加密碼制失效	敏感資料沒加密或加密太弱被破解	強演算法、HSTS、金鑰管理
A05	注入攻擊	惡意指令混入正常資料被系統執	參數化查詢、輸入白名單
A06	不安全設計	程式碼沒 Bug 但業務邏輯有漏洞	威脅建模、安全設計模式
A07	身分驗證失敗	帳號被盜、暴力破解或 Session 被搶	多因子驗證 (MFA)、登入限速
A08	軟體及資料完整性失效	資料或更新檔在傳輸過程中被篡改	完整性檢查、反序列化防護
A09	安全日誌與告警失效	發生攻擊卻沒紀錄，或是沒人發現	中心化日誌、自動化告警
A10	特殊狀態處理不當	系統出錯時洩漏機密或防禦失效	失敗即安全、統一錯誤處理

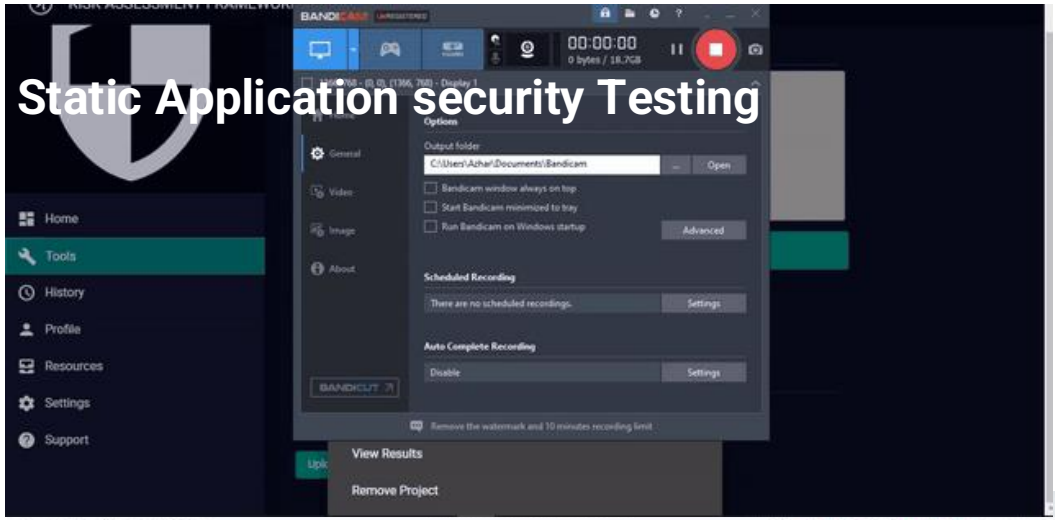
OWASP API Top 10 : 2023

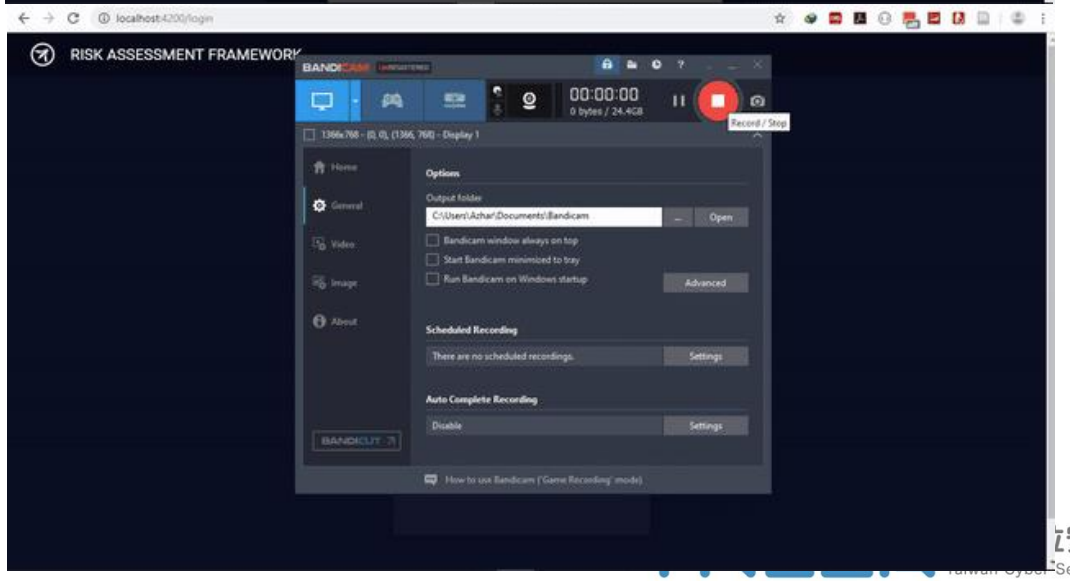
- API 01 物件級授權被破壞
- API 02 認證失效
- API 03 破碎物件屬性級授權
- API 04 資源消耗不受限制
- API 05 功能等級授權被破壞
- API 06 伺服器端請求偽造
- API 07 安全配置錯誤
- API 08 缺乏針對自動化威脅的保護
- API 09 資產管理不當
- API 10 API 的不安全使用

https://owasp.org/API-Security/

API Security Top 10 - 2023



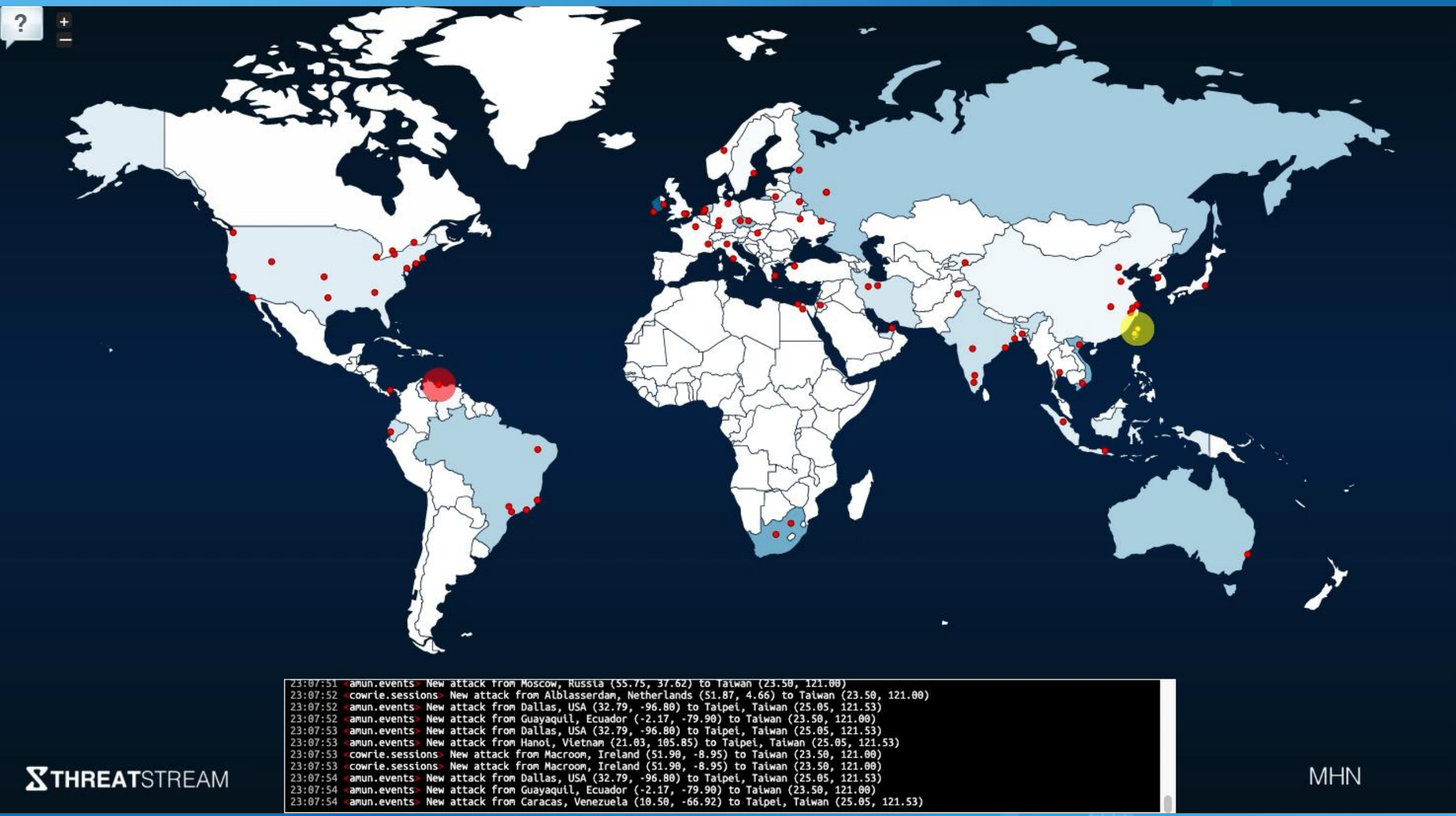




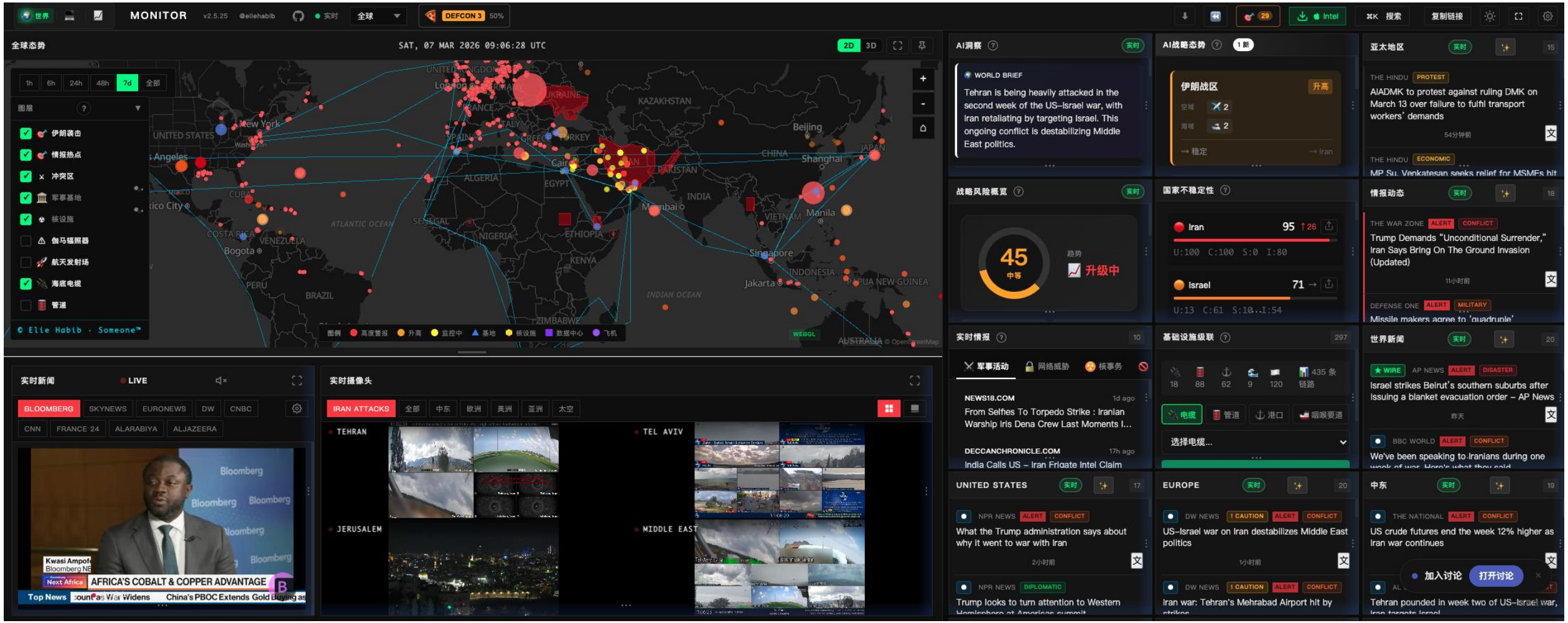
https://owasp.org/API-Security/

TWCSA
台灣數位安全聯盟
Taiwan Cyber Security Alliance

常見攻擊案例分享


?
+
−
23:07:51 <amun.events> New attack from Moscow, Russia (55.75, 37.62) to Taiwan (23.50, 121.00)
23:07:52 <cowrie.sessions> New attack from Alblasserdam, Netherlands (51.87, 4.66) to Taiwan (23.50, 121.00)
23:07:52 <amun.events> New attack from Dallas, USA (32.79, -96.80) to Taipei, Taiwan (25.05, 121.53)
23:07:52 <amun.events> New attack from Guayaquil, Ecuador (-2.17, -79.90) to Taiwan (23.50, 121.00)
23:07:53 <amun.events> New attack from Dallas, USA (32.79, -96.80) to Taipei, Taiwan (25.05, 121.53)
23:07:53 <amun.events> New attack from Hanoi, Vietnam (21.03, 105.85) to Taipei, Taiwan (25.05, 121.53)
23:07:53 <cowrie.sessions> New attack from Macroom, Ireland (51.90, -8.95) to Taiwan (23.50, 121.00)
23:07:53 <cowrie.sessions> New attack from Macroom, Ireland (51.90, -8.95) to Taiwan (23.50, 121.00)
23:07:54 <amun.events> New attack from Dallas, USA (32.79, -96.80) to Taipei, Taiwan (25.05, 121.53)
23:07:54 <amun.events> New attack from Guayaquil, Ecuador (-2.17, -79.90) to Taiwan (23.50, 121.00)
23:07:54 <amun.events> New attack from Caracas, Venezuela (10.50, -66.92) to Taipei, Taiwan (25.05, 121.53)
THREATSTREAM
MHN

從 World Monitor 看資訊應用



https://www.worldmonitor.app/

物聯網裝置安全

- 一位名為Stackoverflowin的網路駭客入侵15萬台網路印表機，並遠端作業，使其印表機印出了文字與ASCII編碼構成的機器人，受害印表機廠商遍布多數大廠牌，如Canon、Epson、HP、Lexmark、Brother等
- 網路印表機能夠存取機密資料等，若被侵入傷害並不小於電腦資料庫
- 使用者多使用預設密碼，且大部分網路印表機使用外網IP，有極大機會被駭客入侵，並成為DDoS網路攻擊時的殭屍裝置

你以為印表機很安全嗎？全臺46所學校印表機遭駭客入侵！

POSTED ON 2017 年 03 月 10 日 BY 165反詐騙

Like 104 Share G+

2017年資安風暴一波未平一波又起，上個月二月初臺灣證券市場才遭逢第一件大型駭客攻擊，二月底多所學校的網路印表機均收到自動列印署名為Emerson Rodrigues之恐嚇信件，要求校方指定時間前支付3個比特幣（約新臺幣10萬元），若未準時付款就會於3月1日發動網路攻擊以癱瘓學校網路。

延伸閱讀>>臺灣證券市場遭逢第一件大型駭客攻擊！刑事局已介入偵辦

經過教育部調查後，上週五至本周一為止，全臺已有多達46所學校收到駭客威脅信，目前以大專院校為大宗。仍有部分縣市尚未回報，估計受駭客入侵學校數量恐怕不只如此。根據行政院國家資通安全會報公告，該類印表機因曝露於網際網路且未設置相關登入帳號密碼或使用預設密碼，導致有心人士可於外部直接存取網路印表機。

https://blog.trendmicro.com.tw/?p=48217

真實的案例：事務機上的資料...

SHARP
MX-4071

操作手冊下載 網站地圖

繁體中文

登入

狀態 | 位址目錄 | 原稿操作 | 用戶控制 | 系統設定

- 文書管理
- 主資料夾
  - 快速歸檔資料夾
  - 自訂資料夾
  - 搜尋
  - 歸檔批次列印
- 發送列印工作

國立[redacted]學臨時工/工讀生/部分工時人員 □本人勞、健保、勞退金 □眷屬健保 加保申請表

被保險人資料

姓名	黃[redacted]	任職單位	註冊組	☎聯絡人、校內分機
身分證字號（外籍人士請統一證號）	B1[redacted]	職稱	□臨時工 □兼任助理 ☒教學助理 □工讀生	手機： 單位聯絡人林 校內分機：50[redacted]
出生年月日	87[redacted]	月支酬勞	5632 元	元
本次聘期	自112年8月1日 至 112年12月31日	勞、健保公提經費來源		會計編號：R[redacted]

身分別註記（具右列身分請勾選）
□持有身心障礙手冊（另請檢附身心障礙手冊影本）
□與本國籍人士結婚之外國人（另請檢附戶口名簿影本）
□持永久居留證之外國人

勞保投保資格調查（依所勾選項目對應右表應參加之保險）

調查項目	請擇一勾選	應參加之保險：●可 （×不可）		
		項目	普通事故保險	職業災害保險	就業保險
年滿65歲	□是 □否	未滿65歲，未領養老改老年給付。	●	●	●
		未滿65歲，已領公保養老給付。	●	●	×
已領公/勞保養老/老年給付	□是 □公保養老給付 □勞保老年給付 □否	未滿65歲，已領勞保老年給付。	×	●	×
		年滿65歲，65歲前曾參加勞保，未領養老改老年給付。	●	●	×
		年滿65歲，65歲前曾參加勞保，已領公保養老給付。	●	●	×
曾參加勞工保險	□是 □否	年滿65歲，65歲前曾參加勞保，已領勞保老年給付。	×	●	×
		年滿65歲，從未參加勞保，未領養老改老年給付。	×	●	×
		年滿65歲，從未參加勞保，已領公保養老年給付。	×	●	×

勞工退休金（自提比率）：□不提撥 □自願提繳，提繳率___%(≤6%)

健保（本勾選限可不填入）：□不納入本校（短期性工作不超過3個月，或每週工作時數未達12時） □轉入本校(加保日：□勞期起日□依所附健保轉出表)

健保依附投保眷屬資料

姓名	身分證字號（外籍人士請填統一證號）	出生年月日	稱謂	眷屬申請加保日
		年 月 日		年 月 日
		年 月 日		年 月 日
		年 月 日		年 月 日
		年 月 日		年 月 日

注意事

一、應檢附文件（請依下列項目逐項檢視）
1.臨時工檢附臨時工申請書；教學助理檢附教學助理加保清冊；工讀生檢附工讀生申請書。
2.身分證（居留證）影本請黏貼於後【外籍人士另檢附工作許可函、護照（含入境戳章）影本】
3.眷屬加保應附戶口名簿影本(不同戶籍，請各附一份)；年滿20歲以上子女加保者，請附學生證影本或無工作能力證明；退伍後1年內加保者，請附退伍證影本。
4.上開「身分別註記」欄有勾選者，應另附相關證明文件。
二、健保不得重複加保，於本校加保者（含眷屬），請務必於原加保單位完成健保轉出。
三、因勞保無法追溯加保，為確保被保險人權益，應於到職日前填送本表及應備文件送人事室專案人力組辦理加保，如於到職

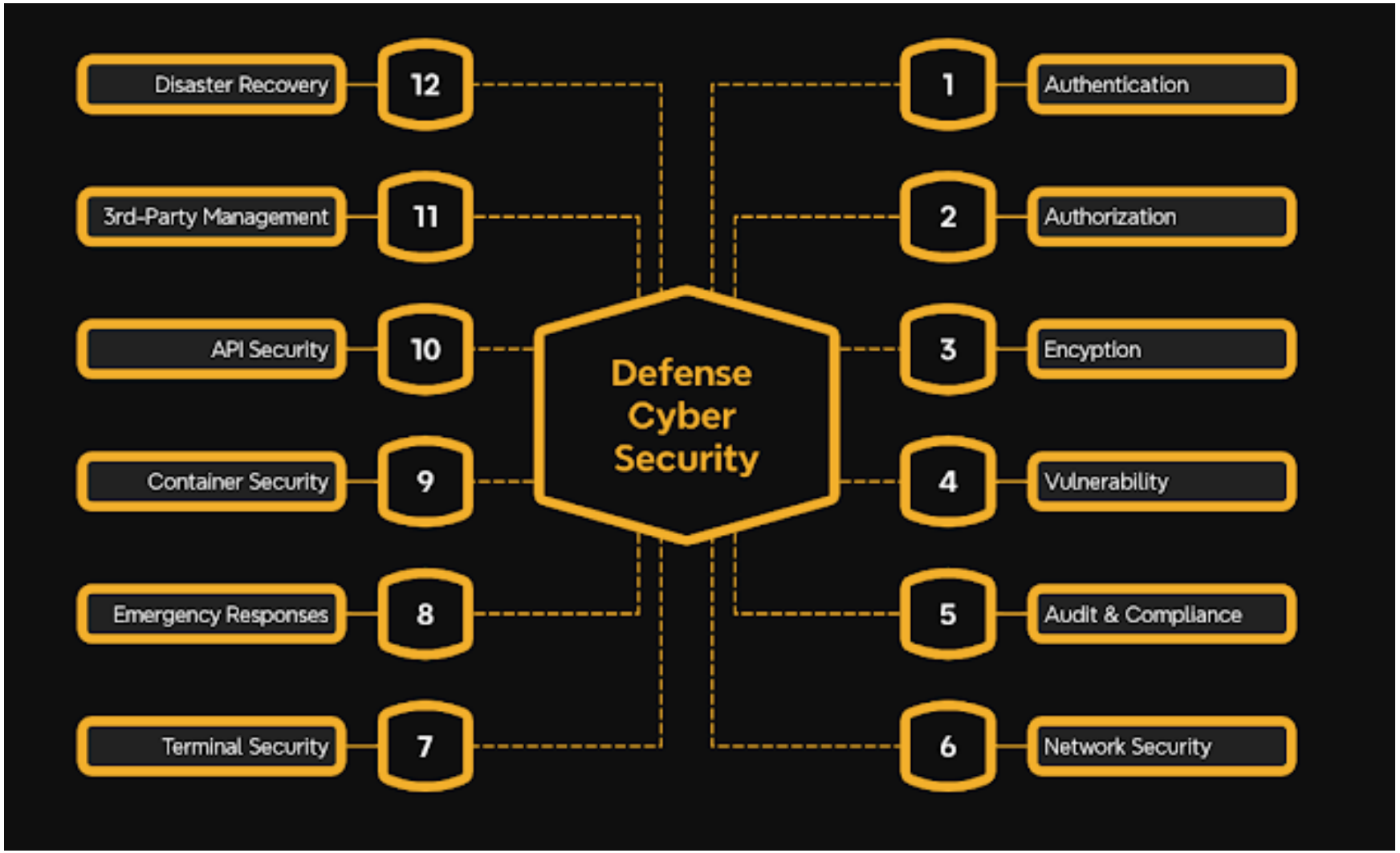
PAGE TOP

真實的案例：攝影機的安全...

校園中潛在的資安風險

- 資訊的竊取為首要目的
- 進而影響到系統的安全
- 教職員生都可能成為受駭的目標

資訊安全的十二個面向



- 目前應該要思考的十二個面向，如何進行組織所需要的資訊安全防禦，
- 而這些不同的面向仍然會因為組織的屬性、使用的平台以及商業模式的不同，而有不同的權重比例
- 從 ESG 角度思考資安管理已成未來的趨勢

https://blog.yilang.org/2024/10/blog-post.html

資訊安全從 CDM 做起

資安防禦框架 OWASP CDM	識別 Identify	保護 Protect	偵測 Detect	回應 Respond	復原 Recover
設備 Devices	組態管理、弱點掃描	端點裝置防護 (AV/HIPS)	使用者及設備行為分析 UEBA	資安事件管理與應變 SIEM 資安協同自動化回應 SOAR	異地備援 資料備份
應用程式 Applications	靜態分析、動態分析、軟體資產管理	應用服務防護 (WAF)			
網路 Networks	設備與組態管理	網路安全防護 (FW/IPS/VPN)			
資料 Data	資料審計	雲端應用資料保護 (Cloud/DLP)			
使用者 Users	多因子驗證	資安教育 (Training)			
相依層級 Degree of Dependency	技術 (Technology)				人員 (People)
	程序 (Process)				

TWCSA
台灣數位安全聯盟
Taiwan Cyber Security Alliance

安全開發與防護思維

從兩本「軟體測試實務」的撰寫談起...

我們都知道
應用程式安全很重要！

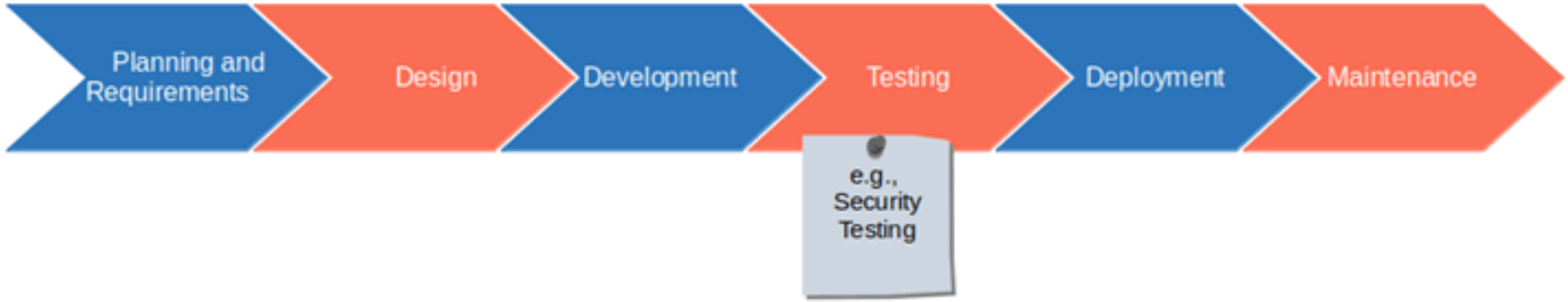
軟體測試是確保軟體品質與安全的重要手段。然而，現代軟體市場激烈競爭與開發迭代速度加快，對軟體測試的效率與效果提出了更高的要求。軟體測試領域知識與主題繁多，且由於測試品質常仰賴施測者的實務經驗，較缺乏測試經驗的人難以入門。因此，本書設計的精神即為幫助讀者「參考業界成功經驗，快速實踐軟體測試」，期望透過本書有效地分享業界寶貴的軟體測試成功經驗，加速國內專業軟體測試人才的培育，提高軟體品質和安全水準。

~主編 李信杰

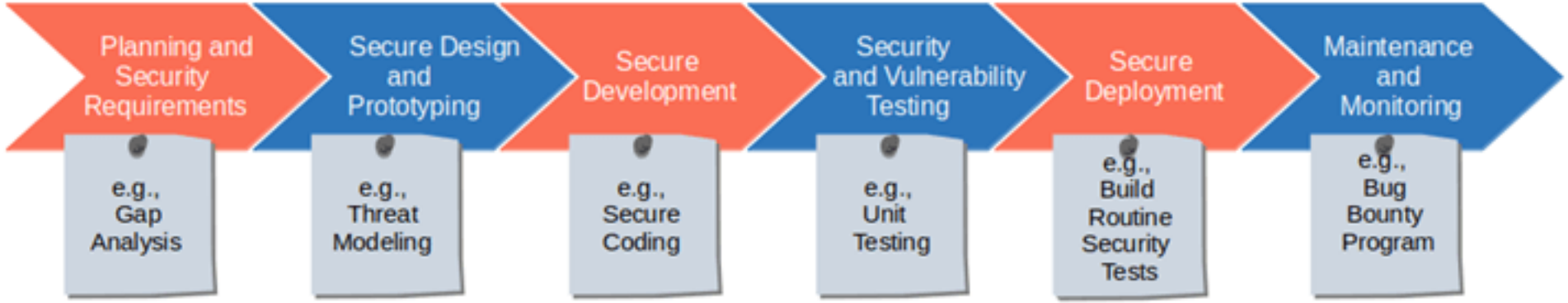
https://sites.google.com/view/swtesting

軟體開發 vs 安全軟體開發

Software Development Life Cycle (SDLC) Process



Secure Software Development Life Cycle (SSDLC) Process



常見用來降低資安風險的策略

- 落地「預設安全」的開發文化 (Security by Design)
  - 在程式撰寫初期就導入「失敗即安全 (Fail-Safe)」邏輯。
  - 具體做法：所有 API 與網頁預設必須經過身份驗證才能存取（Deny by Default）；當系統出錯時，只回傳通用錯誤代碼，嚴禁洩露技術細節（Stack Trace）。
- 建立軟體物料清單 (SBOM) 監控
  - 現代開發 80% 依賴第三方套件，供應鏈已成為最大破口。
  - 具體做法：使用工具（如 Dependency-Check 或 Snyk）自動盤點所有 Library 版本，並在 CI/CD 流水線加入自動化掃描，一旦發現高風險漏洞（CVE）或套件數位簽章不符，立即阻斷部署。
- 強制推行多因子驗證 (MFA) 與最小權限
  - 密碼已不可信，身份竊取是入侵的最快路徑。
  - 具體做法：對外服務、管理後台與 VPN 必須強制開啟 MFA。針對內部員工，嚴格執行「最小權限原則」，每季定期審核權限清單，移除離職或調職人員的存取權。
- 提升日誌的可視性與主動告警
  - 有紀錄不代表安全，能「即時發現」才是關鍵。
  - 具體做法：將分散的 JSON 日誌匯整至中心化平台（如 ELK 或 Splunk）。設定行為閾值告警（例如：同一 IP 在 1 分鐘內出現 50 次 403 錯誤），讓資安團隊能在攻擊發生的「當下」就介入，而非事後才看 Log 寫報告。
- 持續性的安全評測與演練
  - 環境是變動的，昨天的安全配置不代表今天也安全。
  - 具體做法：不要只做一年一次的滲透測試。建議針對核心業務執行週期性弱點掃描，並針對員工進行社交工程演練（防釣魚），因為「人」往往是技術防禦中最脆弱的一環。

通用弱點與漏洞編號

- Common Vulnerabilities & Exposures
  - 訂定一個唯一的名稱
  - 提供一個標準的描述
  - 使評估報告更容易被理解與解讀

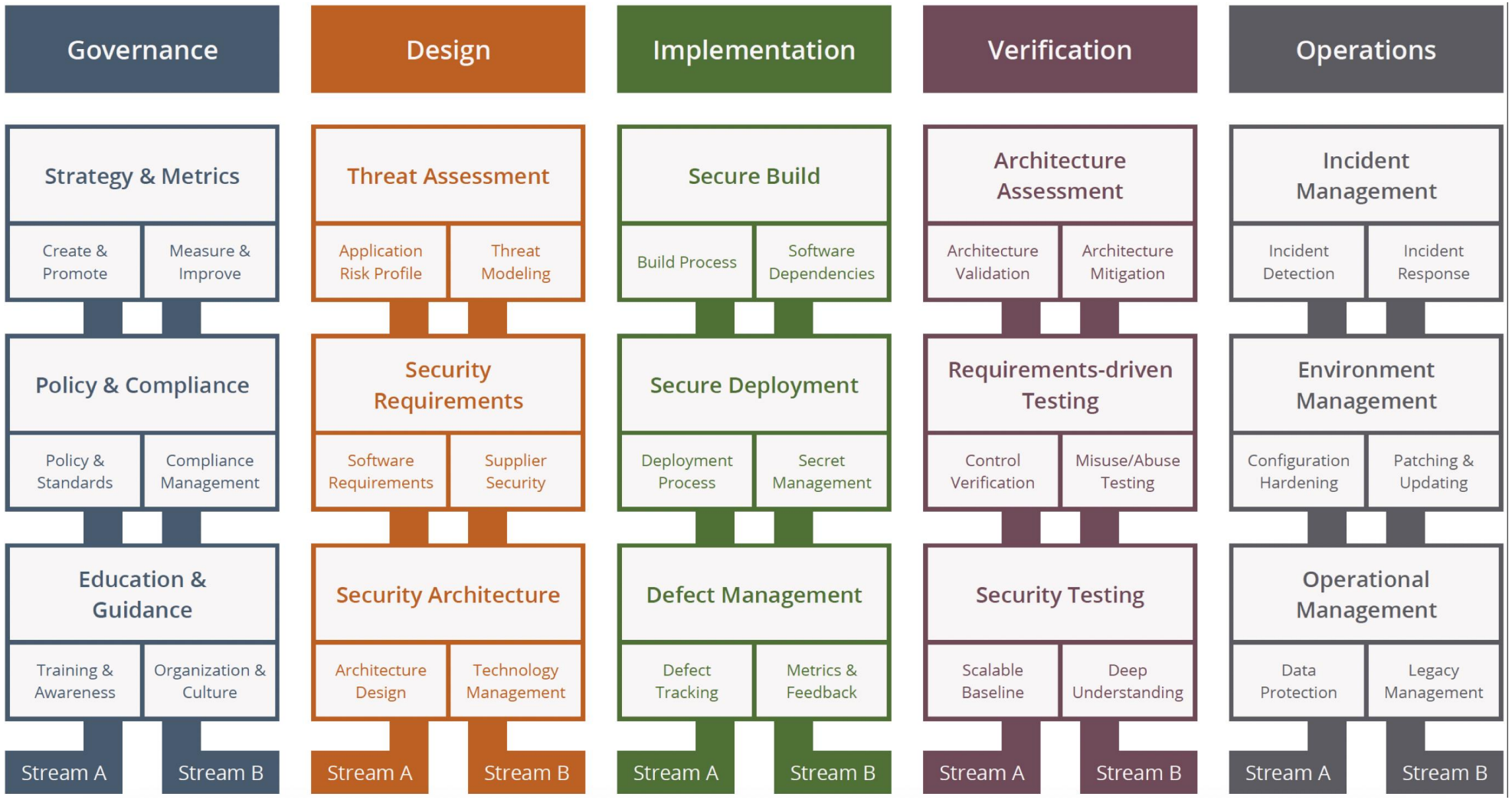
- CVE編號格式 (CVE-XXXX-XXXX)
  - 第一組數字表示年度
  - 第二組數字表示該年度被發現的序號

弱點掃描與滲透測試的差異

- 滲透測試
  - 以駭客的角度來進行各種攻擊測試
  - 可能發現潛在的漏洞

- 弱點掃瞄
  - 針對已知的弱點進行檢測
  - 可藉由自動化的工具來大幅減少檢測的時間
  - 誤判率較高

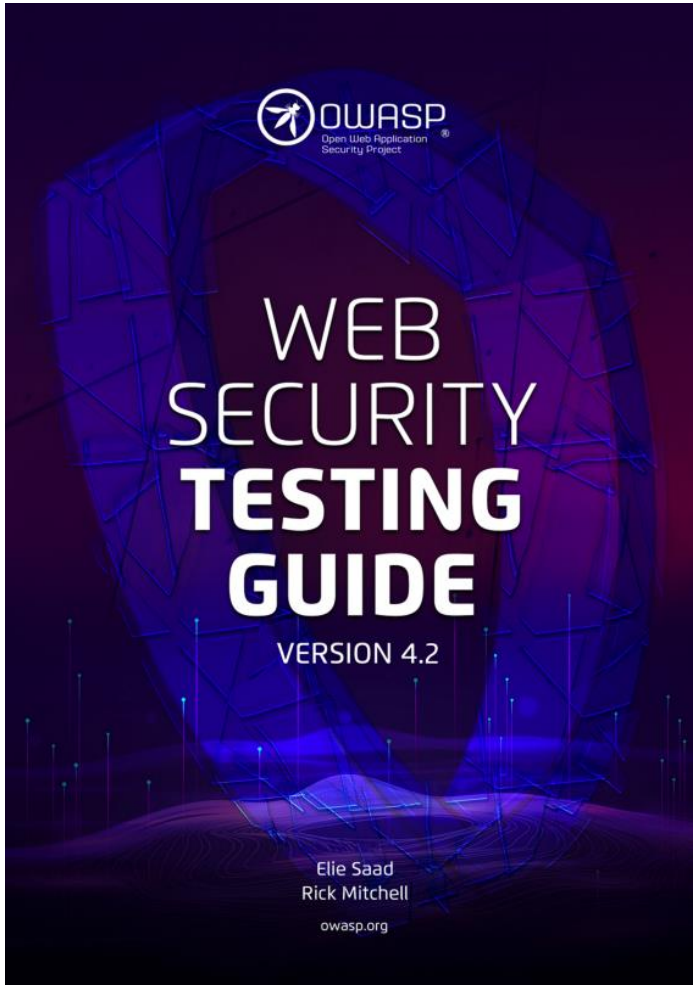
OWASP SAMM



OWASP Software Assurance Maturity Model (軟體保證成熟度模型)

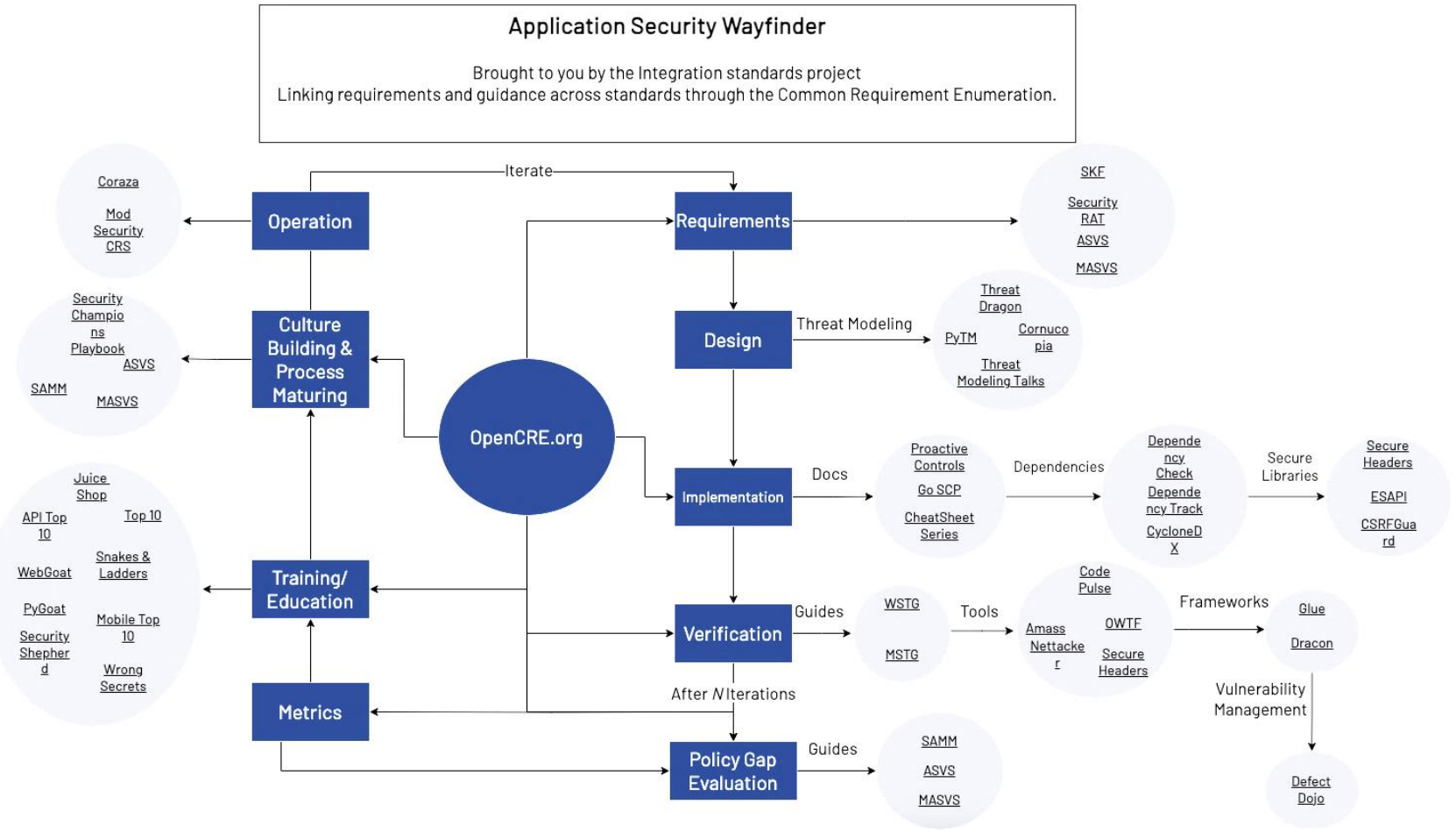
主要面臨的資安威脅

- COVID-19 全球疫情帶來企業營運模式轉變
- 資訊科技發展快速，大量使用雲端應用服務平台
- 資安認知不足，層出不窮的資安事件
- 資訊系統弱點，造成營運資料外洩
- 數位轉型未審慎思考資安風險
- 典型資安防禦機制，無法因應企業轉型需求



https://github.com/OWASP/wstg/releases/download/v4.2/wstg-v4.2.pdf

OWASP Wayfinder

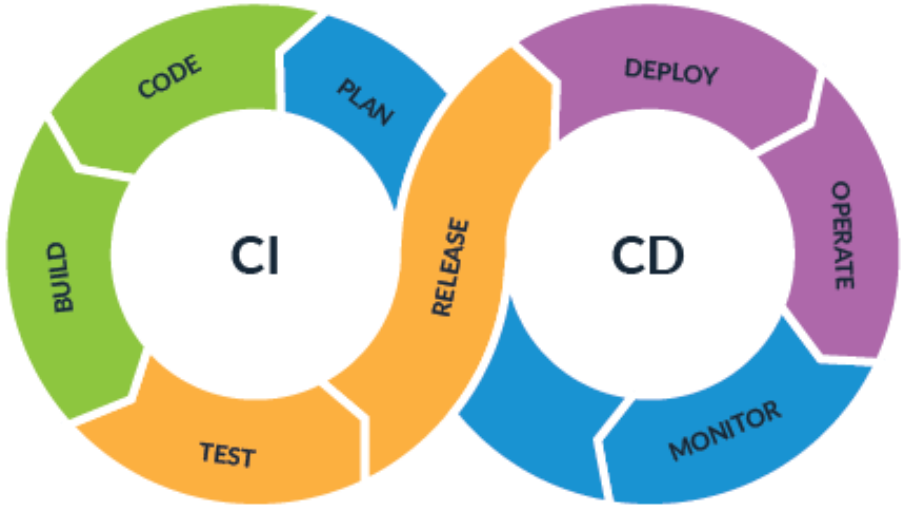


- **Top 10**
  - OWASP Top 10 2021
  - Web Application Security Risks
- **API Top 10**
  - OWASP API Security Top 10 2023
  - OWASP Risk Assessment Framework
- **ASVS**
  - OWASP Application Security Verification Standard
- **WSTG**
  - OWASP Web Security Testing Guide
- **SAMM**
  - OWASP Software Assurance Maturity Model
- **ZAP**
  - OWASP Zed Attack Proxy

https://owasp.org/projects/

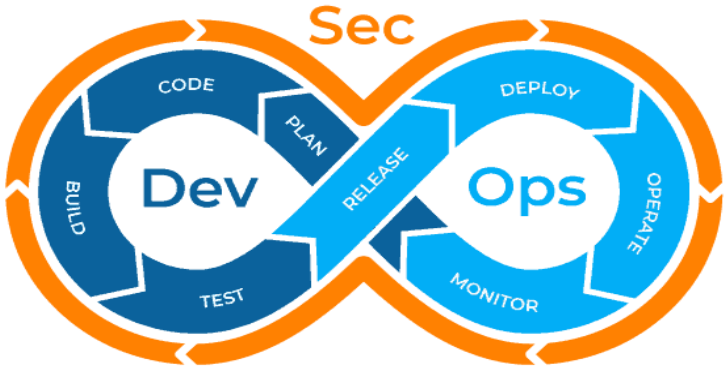
從 CI/CI 思考應用程式的構建、測試與部署

- CI/CD或CICD通常指的是持續集成和持續交付或持續部署的組合實踐
- 透過自動化的程序，在開發和運營團隊之間架起了橋梁
- 現代 DevOps 實踐涉及軟體應用程式在整個開發生命周期內的持續開發、持續測試、持續集成、持續部署和持續監控
- 構成了現代DevOps業務的主軸



https://zh.wikipedia.org/zh-tw/CI/CD

思考未來的挑戰- DevSecOps

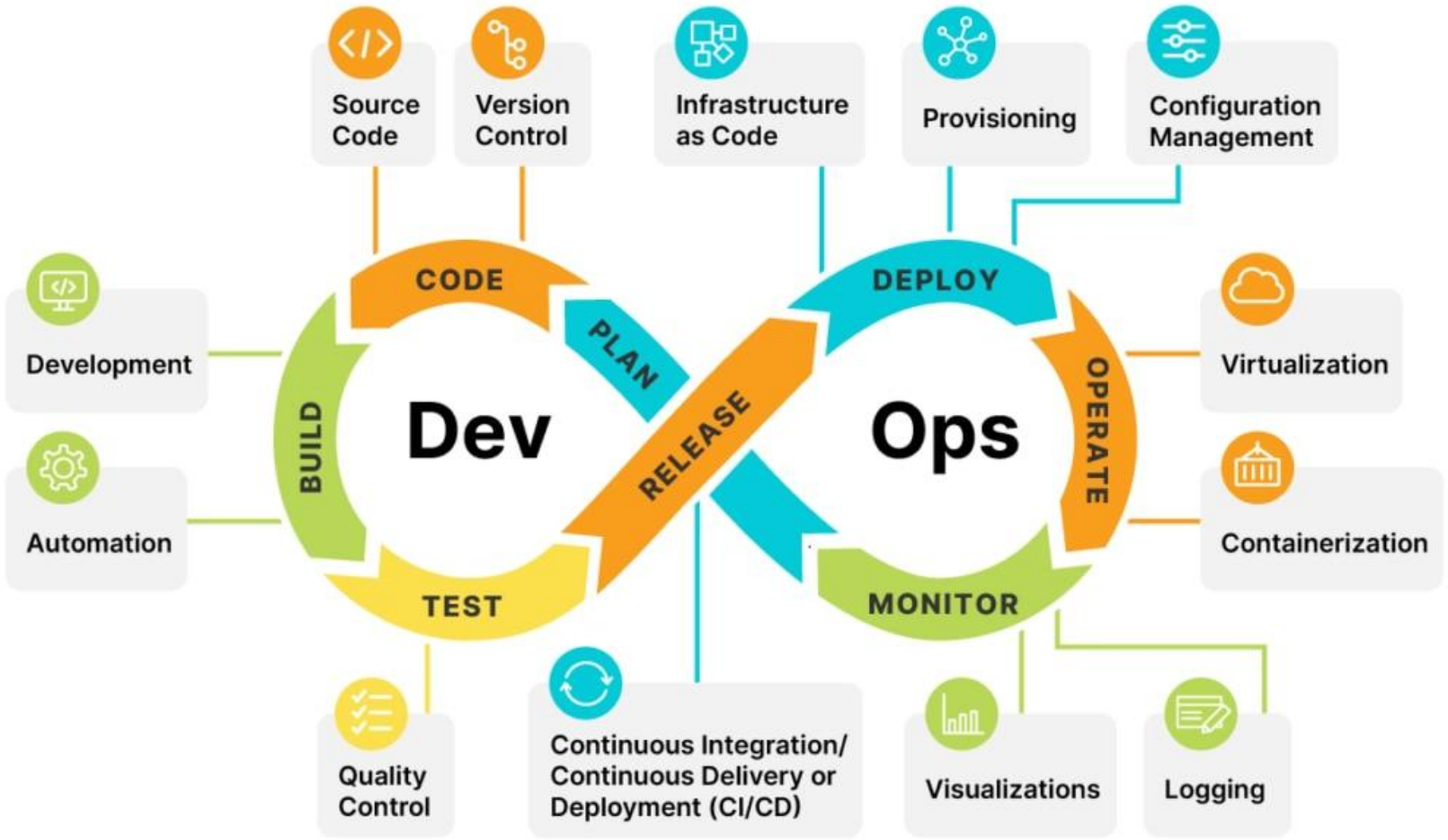


開發 + 維運 + 資安

- 以「**資安**」拉近「**開發**」與「**維運**」的邊界
- 提供「**開發**」與「**維運**」需要的「**資安**」工具與方法
- 建立以「**資安**」為基礎的「**開發**」與「**維運**」的作業流程

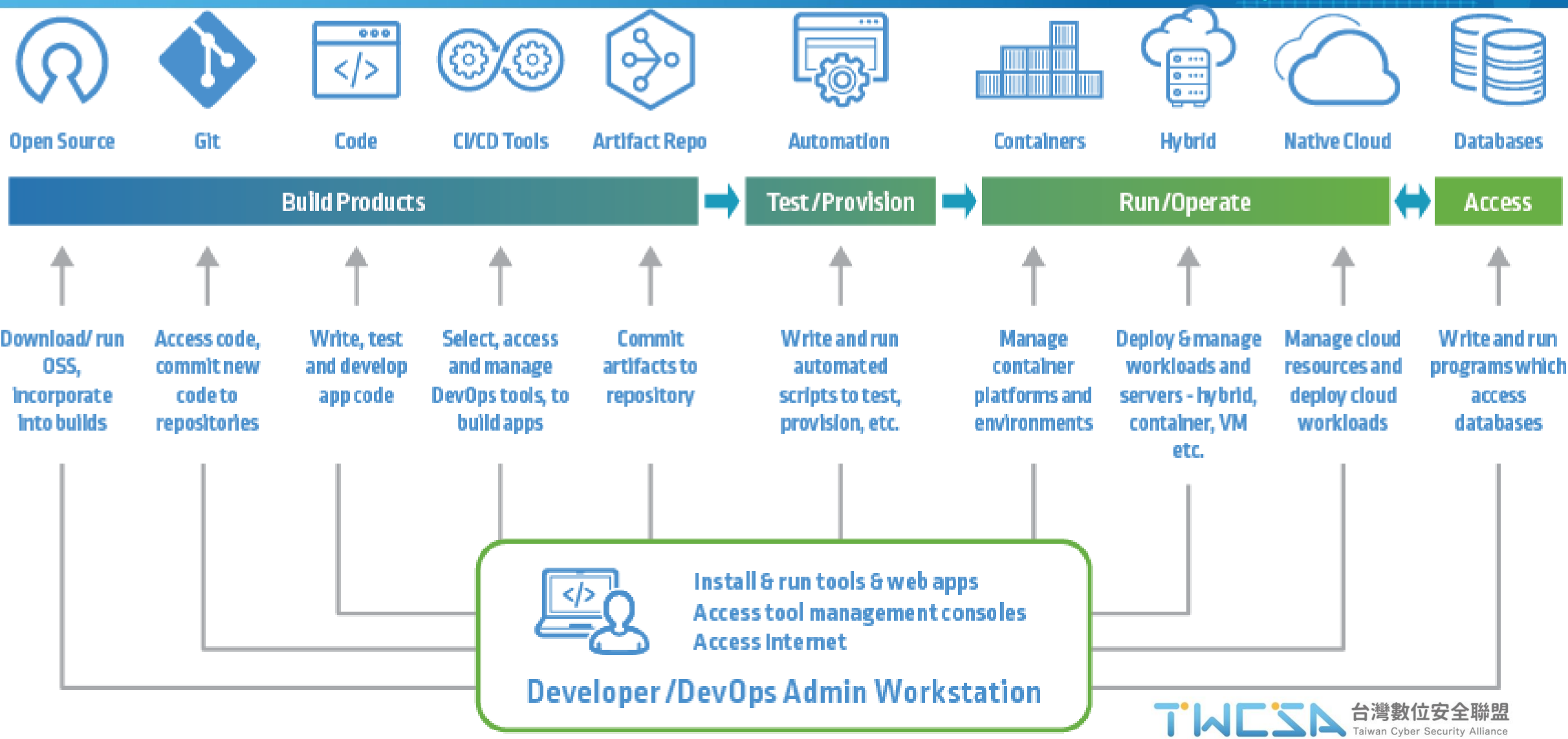
	準備階段	設計階段	建構階段	部署階段	執行階段
	資產、威脅模型、風險評估、量化	評估現有資安架構、存取控制、機敏資訊管理方法等	靜態掃描、程式碼安全檢測	環境互動測試、第三方函式庫測試、動態測試、安全配置	監控、自我保護、威脅偵測與應變、整合至維運管理作業
工具與方法	• 風險評鑑 • 採用國際慣用之評估模型 • 現行資安架構可能的風險	• 選擇適合的程式開發與設計工具 • 確定機敏資訊的保護機制 • 盤合網路與系統的存取控制機制	• 程式碼安全檢測 (Checkmarx等) • 開源程式碼安全檢測 (Mend.io等)	• 應用服務平台弱點檢測 (Burp Suite、OWASP ZAP等) • 系統安全弱點檢測(Kali、Nessus	• 全天候資安維運中心 (SOC) • 資安智能監控平台(IBM Qradar等) • 應用程式安全弱點與威脅監控

瞭解 DevOps



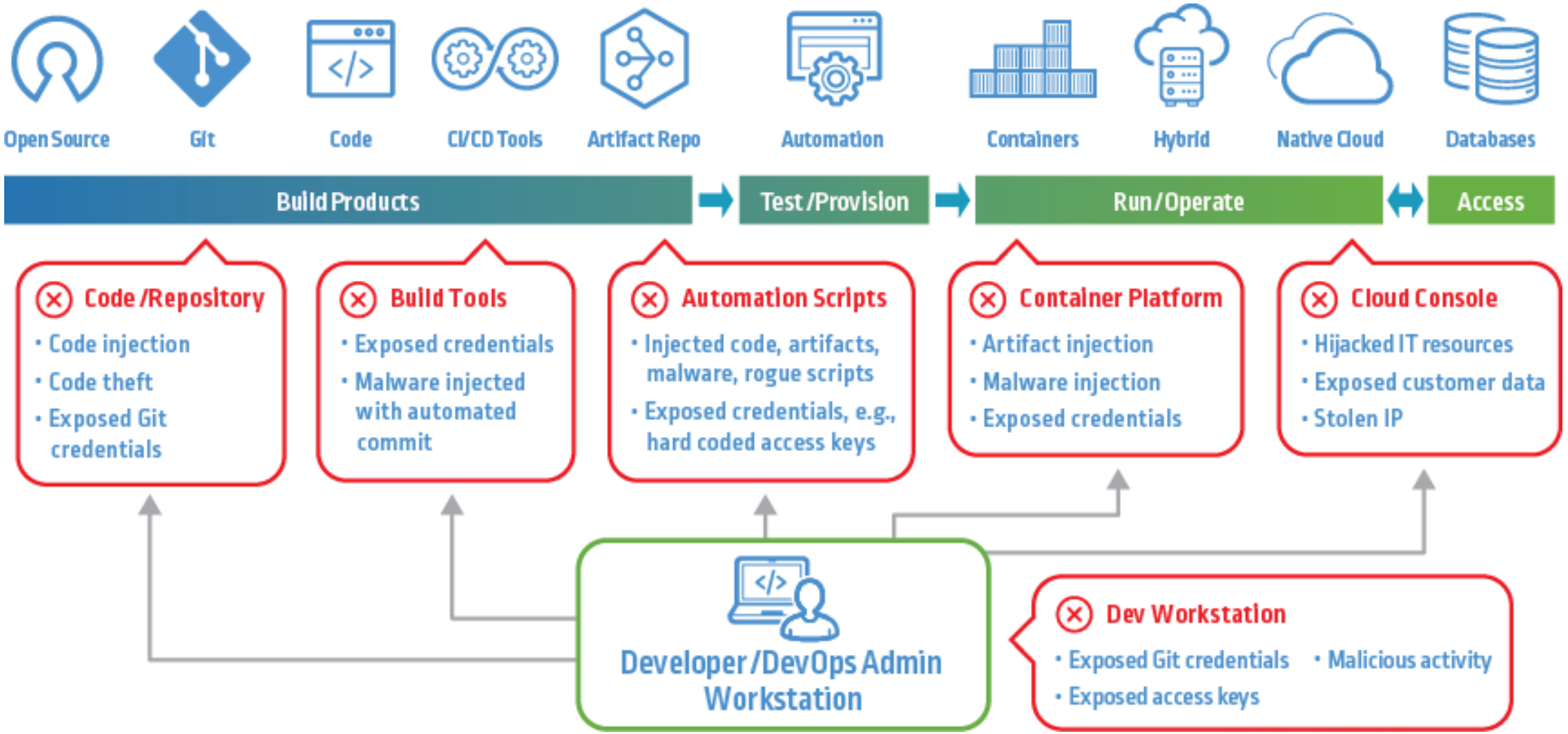
https://orangematter.solarwinds.com/2022/03/21/what-is-devops/

CI/CD Pipeline



https://www.cyberark.com/zh-hant/what-is/ci-cd-pipeline/

Securing the CI/CD Pipeline



https://www.cyberark.com/zh-hant/what-is/ci-cd-pipeline/

Data Science ON GOOGLE CLOUD

#GCPSketchnote
@PVERGADIA THECLOUDGIRL.DEV
01.10.2022

DATA ENGINEERING

Data Ingestion & Discovery
- DATA FUSION
- PUB/SUB
- DATAFLOW
- DATASTREAM
- STORAGE TRANSFER SERVICE
- BIGQUERY DATA TRANSFER SERVICE

Data Preprocessing
- DATAFLOW
- DATAPROC
- DATAPREP
- VERTEX AI DATA LABELING

Data Storage
- CLOUD STORAGE
- BIGQUERY
- CLOUD SQL
- CLOUD SPANNER
- BIGTABLE
- DATAPLEX

Data Cataloging
- DATA CATALOG
- VERTEX ML METADATA

DATA ANALYSIS

Data Exploration
- VERTEX AI WORKBENCH
- LOOKER
- DATA STUDIO
- BIGQUERY

Data Preprocessing
- VERTEX AI WORKBENCH
- DATAFLOW
- DATAPROC
- DATAPREP
- BIGQUERY

Data Insights
- VERTEX AI WORKBENCH
- BIGQUERY
- LOOKER
- DATA STUDIO

MODEL DEVELOPMENT

Feature Engineering
- VERTEX AI WORKBENCH
- SQL WITH BIGQUERY
- SPARK ON GOOGLE CLOUD

Model Training
PRODUCTS
- VERTEX AI TRAINING
- AUTOML
- SPARK ON GOOGLE CLOUD
- BIGQUERY ML

HARDWARE ACCELERATORS
- GPUS
- TPUS

HYPERPARAMETER OPTIMIZATION
- VERTEX AI VIZIER

Model Evaluation
- VERTEX AI TENSORBOARD

Feature Store
- VERTEX AI FEATURE STORE

Model Registry
- VERTEX ML METADATA

ML ENGINEERING

Model Serving
REAL-TIME INFERENCE
- VERTEX AI PREDICTION
- VERTEX ML EDGE MANAGER
- DATAFLOW
- VERTEX AI MATCHING ENGINE

BATCH INFERENCE
- VERTEX AI PREDICTION
- BIGQUERY ML

Model Deployment
(e.g. edge, microservice)
- VERTEX AI
- TFLITE (EDGE TPU)
- DOCKER CONTAINER
- CORE ML
- TENSORFLOW.JS

Model Monitoring
- VERTEX AI MODEL MONITORING

INSIGHTS ACTIVATION

Influence business decisions
(e.g. reports, dashboards, alerting)

REAL-TIME INFERENCE
- LOOKER
- DATA STUDIO

Influence consumer decisions

Serve other applications/services
- CLOUD FUNCTIONS
- CLOUD RUN
- APIGEE API MANAGEMENT

ORCHESTRATION
(E.G. DATA PIPELINES, MLOPS, SCHEDULING, CI/CD)

DATA PIPELINES
- CLOUD COMPOSER (MANAGED AIRFLOW)
- CLOUD SCHEDULER

GENERAL ORCHESTRATION TOOLS
- CLOUD FUNCTIONS
- CLOUD RUN

ML PIPELINES
- VERTEX AI PIPELINES
- TENSORFLOW EXTENDED
- CLOUD COMPOSER

CI/CD
- CLOUD BUILD

Dataflow

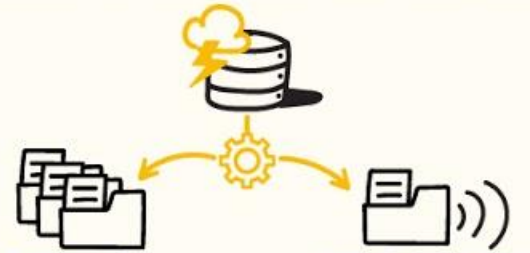
#GCPSketchnote

@PVERGADIA THECLOUDGIRL.DEV

6.28.2021

What is Dataflow?

SERVERLESS, DATA PROCESSING SERVICE FOR BOTH STREAMING AND BATCH DATA



CREATE DATA PROCESSING PIPELINE

Apache Beam (SDK)

DEPLOY & EXECUTE

Dataflow

AUTOMATE
Operations & Management

Data Pipeline → Dataflow Job

How does it WORK?

DATA PROCESSING PIPELINE

STEP 1
Read Data Source

Files, BigQuery, Bigtable, Pub/Sub, Custom Source

STEP 2
Transform Data

PCollection 1 → PCollection 2 → PCollection 3

STEP 3
Write Data Sink

Files, BigQuery, Bigtable, Pub/Sub, Custom Source

Dataflow

WORKER VIRTUAL MACHINES EXECUTE THE DATA PROCESSING

Worker Virtual Machines

Autoscale up or down dynamically as needed

Dataflow Streaming Engine

Separate Compute | Separate Storage

BATCH PIPELINES SCALE SEAMLESSLY, WITHOUT ANY TUNING REQUIRED.

Dataflow → FEATURE Service-based Dataflow Shuffle → Data Grouping & Joining → Scale to 100s of TBs

HOW TO USE Dataflow

1 DATAFLOW SQL VIA BIGQUERY

- Use SQL from BigQuery web UI
- Read from pub/sub, cloud storage or BigQuery
- Write to BigQuery

2 DATAFLOW TEMPLATES

- Share pipeline with teams
- Easy & repeatable Pre built templates

3 AI PLATFORM NOTEBOOKS

- Use latest data science and machine learning frameworks

SECURITY for Dataflow Pipelines

EXACTLY-ONCE PROCESSING

0 Avoid risk of data loss

1 Turn off public IPs

2 VPC controls

3 Customer managed encryption key (CMEK)

Dataflow PRICING

DATAFLOW
Per second billing

DATAFLOW JOB
Batch or Streaming
(FlexRS) flexible resource scheduling = low cost

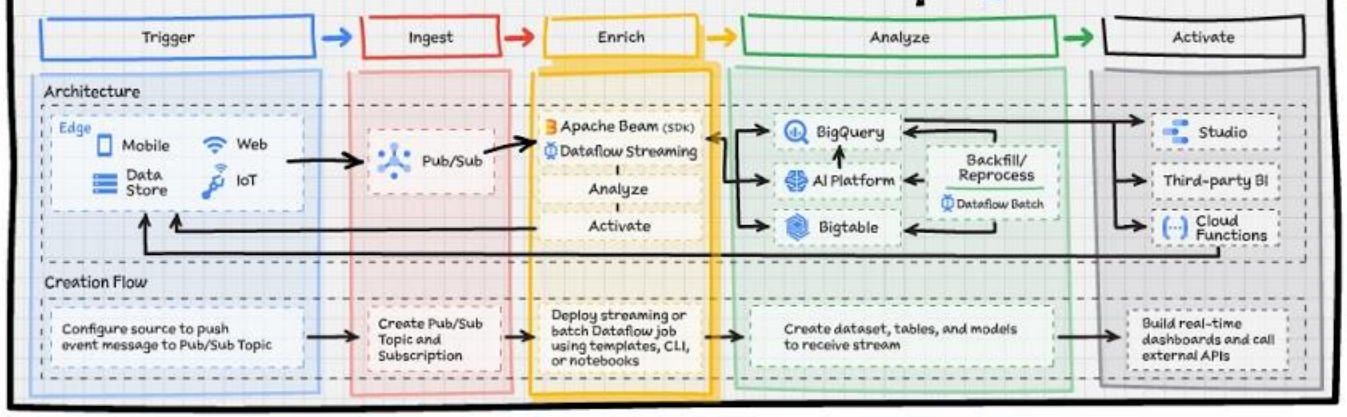
AUTOSCALING
Worker(s)
Better price to performance

CUSTOMIZABLE
vCPU + Memory + Storage
Each billed on per second basis

Dataflow Use case example

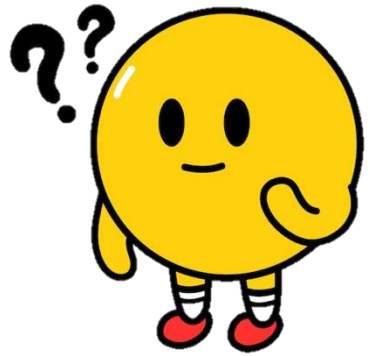
Streaming analytics

Real-time AI



面對問題！

- 每天億級筆數的原始資料，如何有效處理？
- 雲端平台是否有適合的工具？
- 對於資訊生命週期的評估？
- 多少時間內必須取得有價值的資訊？
- 如何讓「資料」變成「資訊」加值成為「情資」？
- GPT的時代，如何整合與融入？



台灣數位安全聯盟
Taiwan Cyber Security Alliance

Q&A