Microsoft Azure

Secure DevOps：應用安全原則與實踐

應用安全原則

模組概述

- 設計安全
- 輸入與輸出處理
- 錯誤處理
- 可用安全
- 認證與授權
- 超期元件
- 審計、警告與監控

Microsoft Azure

設計安全

安全始於設計

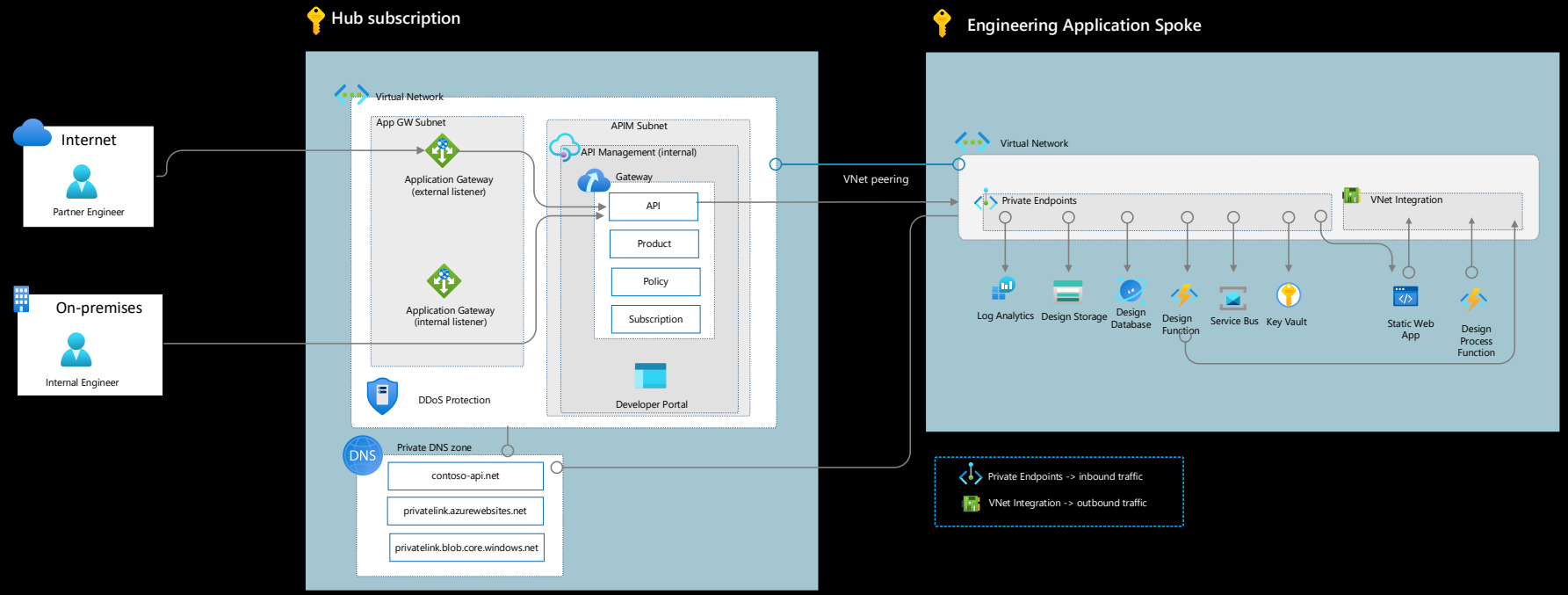
- 縱深防禦 - 補償控制
- 攻擊面縮減
- 預設安全
- 最小特權
- 零信任
- 隱私

縱深防禦

如果一個防禦層被攻破，還有哪些其他防禦層（如果有的話）為應用提供額外保護?

例子：

- 邊界網路
- 身份與網路
- 安全隔離區



攻擊面縮減

攻擊面

應用程式中任何部分，只要是人類或是外部程式可訪問的

攻擊面減少

盡量減少惡意使用者可能發現並試圖利用的暴露攻擊面數量

例子：

- 在生產環境中關閉元資料端點（Swagger）
- 不做額外功能
- 不要暴露异常的细节（FileNotFoundException， IOException）

預設安全

以更安全的配置部署應用

客戶在使用你的應用時能獲得更安全的體驗

降低安全和隱私水準取決於使用者

例子：

- 具有鎖定配置的可重用 IaC 元件
- Microsoft Entra ID 中的 Security defaults settings

最小特權

程式或服務應僅獲得滿足其需求的許可權 —— 不多不少

- 如果一個程式不需要訪問許可權，就不應被授予該許可權
- 可以把它看作 「*need to know*」 規則

如果應用程式被攻破，惡意者可能造成的損害會被控制並相應地降到最低

- 更高的特權是攻擊者所渴望的

零信任 - 分割管理

最大限度減少安全漏洞的影響

提升“應用內”和“應用間”的安全性

建立信任邊界以隔離所有組成部分

- 我的應用運行在哪裡?
- 它從哪裡載入數據?
- 它是如何驗證的?
- 通信管道是如何受到保護的?
- 網路防禦
- 身份防禦

回頭思考

軟體工程就是為使用者的需求創造解決方案

- 使用者渴求功能、特性模組和高性能
- 他們想要安全嗎?

安全的系統開發需要考慮使用者想做什麼以及使用者不應該做什麼，然後為此創建解決方案

- 使用者不應能夠入侵或篡改未分配給他們的資源

補充思考

別重複發明輪子

- 你可能不是安全專家
- Microsoft 等公司提供了許多優秀的安全庫
- 不要自己創造密碼學

避免通過模糊化來保障安全

- 在不受你控制的設備上，沒有任何東西是秘密的（手機、消費類筆記型電腦）
- 不要依賴於隱藏
- 不要指望某件事永遠保密

安全是設計出來的

- 使用威脅建模
- Azure 的藍圖、圖案,.....

這是你的工作，不是網路安全團隊的

- 他們沒有在做你的工作
- 不要依賴手動測試和滲透測試來找出錯誤
- 使用分析工具避免簡單錯誤

Microsoft Azure

輸入和輸出處理

三大安全問題

1 和 2：輸入處理

- 輸入就是「萬惡之源」！（防禦者座右銘）
- 切勿假設輸入有效; 所有輸入均屬可疑，除非有證據證明
- 永遠不要依賴客戶端驗證（信任度極低）; 在伺服器上驗證

Vulnerable statement

```
string sql = "SELECT * FROM users WHERE name = '" + username + "';";
```

Setting the 'username' to

```
a';DROP TABLE users;
```

Renders the following SQL statement

```
SELECT * FROM Users WHERE name = 'a';DROP TABLE users;--
```

3 - 其他所有

- AuthN， AuthZ， 最小特權
- 審計、日誌、SIEM
- 密碼學
- 機密處理
- 錯誤處理
- 過時的元件
- …

哪裡？

永遠不要依賴客戶端驗證！

- JavaScript 的“onSubmit”或“onPostBack”事件可以在用戶端上更改
- 僅將 JavaScript 作為功能驗證，以提升用戶體驗
- 攻擊者會編寫自定義代碼來入侵你的系統

捍衛你的信任界限（保持懷疑）

- 驗證和/或清理所有外部輸入
- 驗證對服務和數據查詢請求的返回值

目的

- 系統不會執行它沒有被設計的任務
- 使用者犯的非惡意錯誤也會被捕捉並報告
- 讓攻擊者更難滲透系統

輸入驗證

只處理你知道並能處理的東西

- 拒絕並記錄其他所有內容

驗證類型、長度、範圍和格式

- 為你的團隊/公司制定戰略
- 如果不可避免，使用正則表達式

安全清單輸入

- 對於每個使用者輸入欄位，都應對輸入內容進行驗證。
- 只接受符合特定標準的數據

從

- 利用操作系統提供的修復措施
- OWASP 10 涵蓋了許多輸入驗證漏洞（A1、A7 等）
- 確保 semantic 一致性（可重用驗證庫）
- 優先使用白名單而非黑名單
- 對於需要更大靈活性的輸入，也可以應用黑名單，將已知的不良輸入模式或字元遮罩掉

輸出處理

進行上下文輸出編碼

所有輸出函數必須在發送數據給使用者之前對上下文進行編碼

- 例如，放置在 URL 上下文中的數據必須與 HTML 頁面中 JavaScript 上下文中的數據進行不同的編碼

輸入與輸出處理（續）

- 使用參數化SQL查詢 CWE - CWE-89：SQL命令中使用的特殊元素的不當中和（4.9）（mitre.org）
- 防止不安全的反序列化 CWE - CWE-502：不可信數據的反序列化（4.9）（mitre.org）
- 使用令牌防止偽造請求 CWE - CWE-352：跨站請求偽造（CSRF）（4.9）（mitre.org）
- 防止伺服器端請求偽造（SSRF） CWE - CWE-918：伺服器端請求偽造（SSRF）（4.9）（mitre.org）
- 為您的應用設置編碼 CWE - CWE-172：編碼錯誤（4.9）（mitre.org）
- 驗證輸入源 CWE - CWE-346：起源驗證錯誤（4.9）（mitre.org）
- X-Frame-Options 或 CSP 請求頭 CAPEC - CAPEC-103：点击劫持（版本 3.8）（mitre.org）
- 驗證上傳檔 CWE - CWE-434：無限制上傳危險類型檔（4.9）（mitre.org）
- 防止標籤頁劫持 CWE - CWE-1022：使用帶有 window.opener Access（4.9）的 Web Link 訪問不受信任目標（4.9）（mitre.org）

Microsoft Azure

錯誤處理

異常

處理所有預期的異常

非預期異常是安全漏洞（總是如此）

- 經常遺漏輸入驗證
- 應用層 DoS 攻擊

記錄異常詳情

- 用於提醒安全團隊
- 不要把機密洩漏到日誌里

錯誤資訊洩露細節！

使用簡單的錯誤資訊

- 顯示簡單的錯誤資訊，不包含太多資訊
- 將詳細資訊寫入紀錄檔
- 特別的：不要顯示 SQL 錯誤資訊

Web 應用使用自訂錯誤頁面

不要將機密和高度保密的數據洩露到日誌中

- 注意在 NVA 上記錄日誌，尤其是查詢參數
- GDPR 陷阱

Microsoft Azure

可用的安全性

安全卻無可用性

可用的安全需要全面審視用戶體驗。 它需要對使用者所面臨的需求有全面的視角，並採取以使用者為中心的策略

當系統的安全功能難以訪問和/或應用時，使用者可能會犯錯，甚至完全放棄保護措施。

可訪問性和易用性

易用 ！= 易破解

可用性對安全軟體的重要性 |Signiant 部落格

Microsoft Azure

認證與授權

無密碼認證

用戶體驗考慮

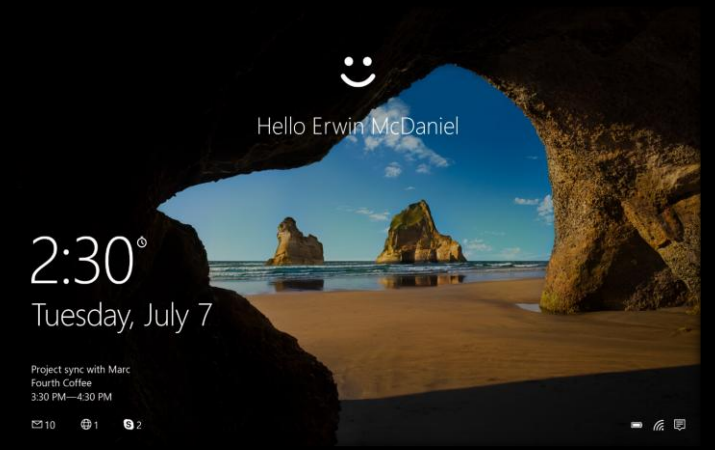
人們覺得密碼難以記住，尤其對某些殘障人士來說更為困難

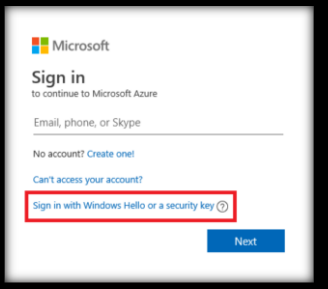
增加密碼長度和複雜度，並強制定期更改密碼，加劇了使用者的挫敗感，降低了用戶體驗。

安全考量

超過60%的駭客攻擊數據洩露使用了包括使用者名和密碼在內的“被盜憑證”。

密碼容易受到針對個人的攻擊（例如，社會工程學、網路釣魚、密碼填充和密碼竊取惡意軟體）。

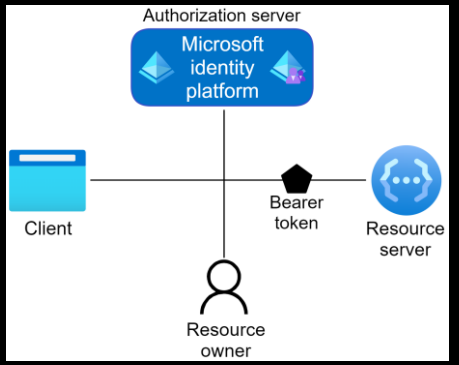




OAuth 2.0 与 OpenID Connect

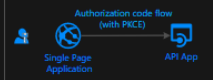
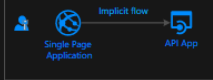
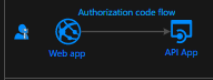
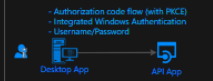
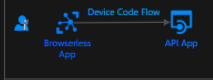
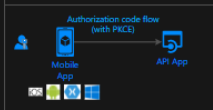
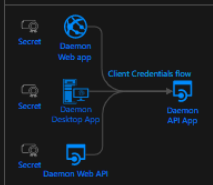
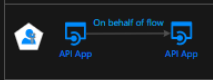
	OAuth2	OIDC
目標	資源伺服器可以決定是否服務請求	用戶端應用知道用戶是誰
機制	訪問令牌	身份令牌
誰會要求的？	客戶	客戶
誰來驗證它？	資源伺服器	客戶
過程	OAuth2 定義的流類型之一	使用OAuth2流型

- 用戶端可能不關心使用者身份（例如移動應用）—— 純 OAuth2.0 協定
- 用戶端無需調用其他 API（資源伺服器），只需知道使用者身份（例如簡單的 Web 應用）—— 純 OIDC 使用 OAuth2 流程
- 用戶端需要既瞭解用戶、調用，又要知道API —— 混合流程



OAuth 2.0 登錄流程

流程名稱	使用者？	瀏覽器？	用戶端類型	典型情景
客戶端憑證	N	N	C	後端服務/守護進程。 使用X509以獲得更強的認證。
授權代碼授予與PKCE合作	和	和	C	網頁應用
	和	和	P	移動應用，SPA
隱式流	和	?	P	不再是合適的認證方式。 改用授權代碼流程。
裝置程式碼流程	和	N	P	無瀏覽器的公共設備（例如恒溫器、電視）
延伸流（OBO）	和	N	C	多層次API。 也可以用來用 SAML 令牌換取 JWT。
資源擁有者障礙者	和	N	P	如果其他方法都失敗，例如無人值守使用需要有人值守使用的服務（如測試、離線郵箱處理）

Scenario	Detailed scenario walk-through	OAuth 2.0 flow and grant	Audience
	Single-page app	Authorization code with PKCE	Work or school accounts, personal accounts, and Azure Active Directory B2C (Azure AD B2C)
	Single-page app	Implicit	Work or school accounts, personal accounts, and Azure Active Directory B2C (Azure AD B2C)
	Web app that signs in users	Authorization code	Work or school accounts, personal accounts, and Azure AD B2C
	Web app that calls web APIs	Authorization code	Work or school accounts, personal accounts, and Azure AD B2C
	Desktop app that calls web APIs	Interactive by using authorization code with PKCE	Work or school accounts, personal accounts, and Azure AD B2C
		Integrated Windows authentication	Work or school accounts
		Resource owner password	Work or school accounts and Azure AD B2C
		Device code	Work or school accounts, personal accounts, but not Azure AD B2C
	Mobile app that calls web APIs	Interactive by using authorization code with PKCE	Work or school accounts, personal accounts, and Azure AD B2C
		Resource owner password	Work or school accounts and Azure AD B2C
	Daemon app that calls web APIs	Client credentials	App-only permissions that have no user and are used only in Azure AD organizations
	Web API that calls web APIs	On-behalf-of	Work or school accounts and personal accounts

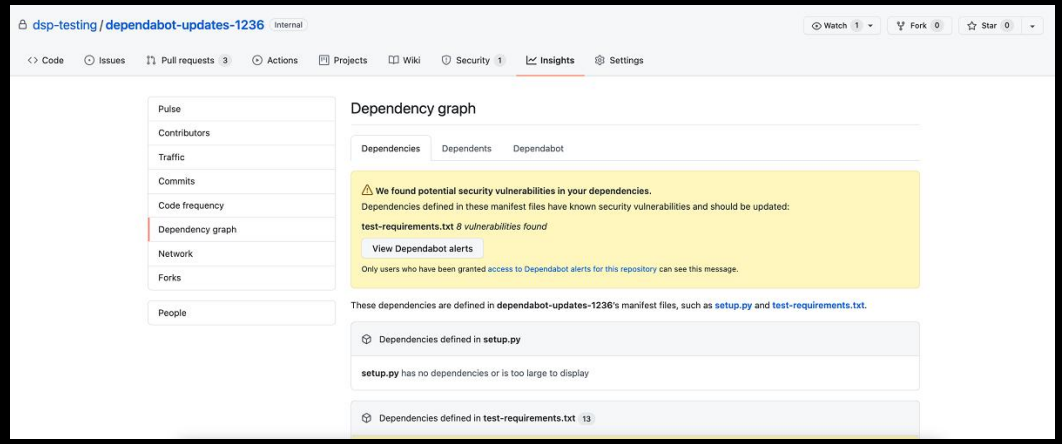
Microsoft Azure

超期的元件

你的元件有多老？

所有軟體都有（安全）漏洞

- 发表于 CVE - https://cve.mitre.org
- 通常已修復
- 許多洩露源於已知漏洞
- 零日漏洞很少見



你的策略是什麼？

- 從不？ 每兩年一次？ 每個衝刺？
- 你經常查看 CVE 嗎?。。。
- 明確你的策略（超過兩個月一次就不好了）

使用工具來管理

- 黑鴨、Nexus、DependaBot、WhiteSource,...

Microsoft Azure

密碼學

問題空間

在傳輸中、記憶體中、靜止時被竊取的數據

- 傳輸中加密
- 靜止加密
- 對秘密和高度機密的數據進行雙重加密

加密依賴於

- 目前安全的演算法（尚未被攻破）
- 強金鑰生成
- 最小鍵長
- 安全存儲和交換金鑰
- 按計劃以及在已知洩露后（緊急更換）輪換密鑰

密碼學

隨機數生成

散列法

- 從大量數據中獲得唯一值
- 快速（尤其是在圖形處理器上），除非被人為減速（反覆運算）
- 單向變換

對稱加密

- 單一共享金鑰
- 快！
- 我們如何共用金鑰

非對稱加密

- 基於極大的質數（2048+位）
- 公鑰/私鑰對——私鑰是任意選擇
- 慢！
- 加密：多對一加密

演算法問題

所有演算法最終都會被攻破

- 計算機速度越來越快
- 量子計算
- 演算法存在數學缺陷（類似於代碼缺陷）

必修

- 加密演算法標準（你們組織需要有）
- SHA-1 自 2005 年起就被認為已被攻破，並在 2020 年出現了實用級攻擊

Microsoft Azure

審計、警告與監控

審計與記錄

有意的安全日誌

保護你的日誌

定期備份日誌

近即時分析日誌

使用 SIEM 和 SOAR 系統

主動日誌

善意

- 認證/授權
- 關鍵應用事件，例如高價值交易請求
- 關鍵資料的讀寫（但不包括資料本身）
- 異常

警告！ 記錄敏感資訊時要小心

- 日誌是另一種輸入來源
- 在日誌前對數據進行清理
- 假設攻擊者能夠訪問這些數據

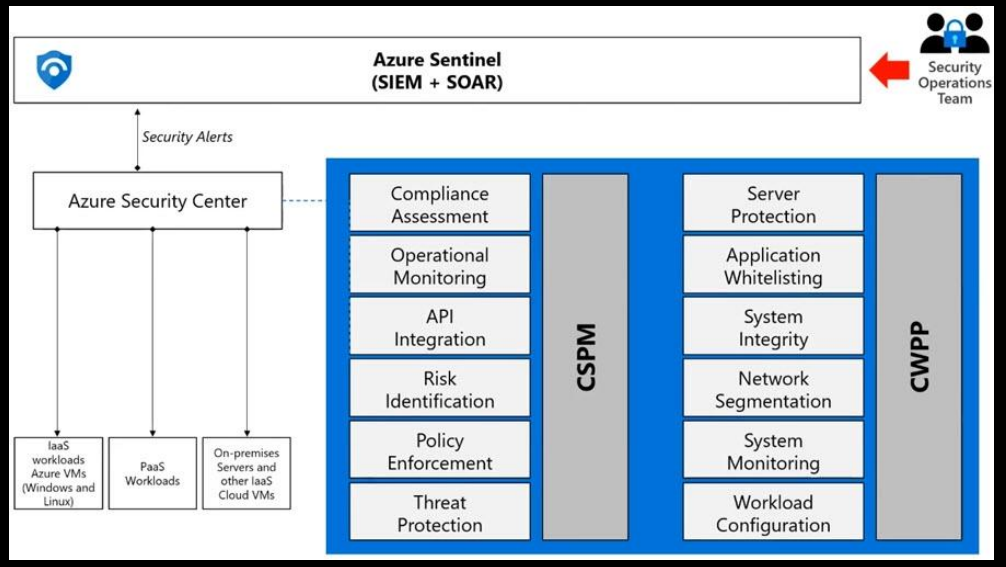
Microsoft Sentinel + Microsoft Defender for Cloud

Microsoft Defender for Cloud

- 雲安全態勢管理（CSPM）
- 雲工作負載保護平臺（CWPP）

Microsoft Sentinel

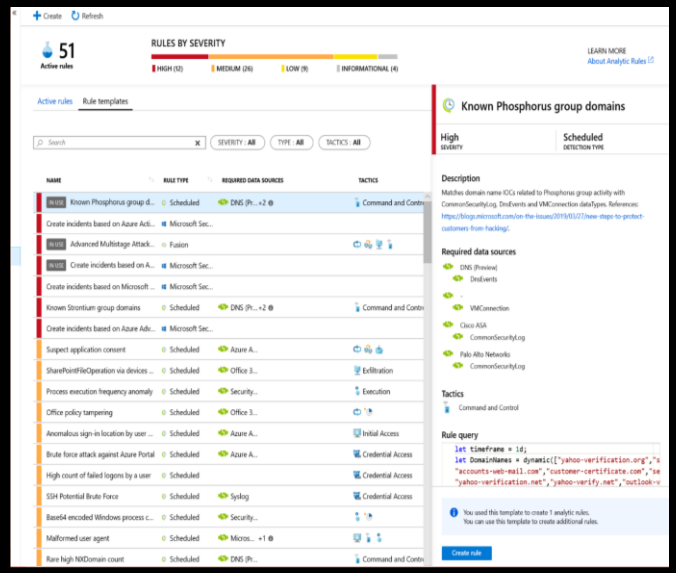
- 安全資訊與事件管理（SIEM）
- 安全編排、自動化與回應（SOAR）



Microsoft Sentinel 簡要介紹

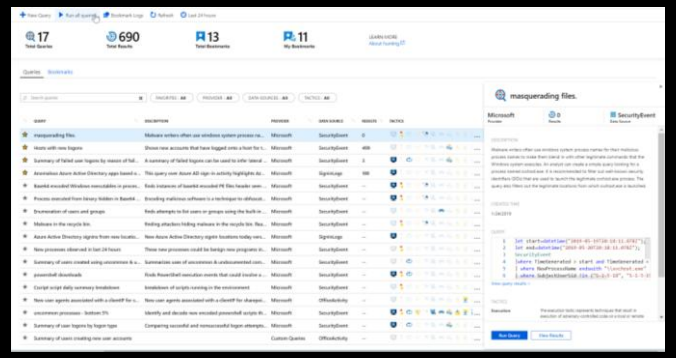
分析

- 從100多個內置分析規則中選擇
- 利用 KQL 自訂查詢並建立您自己的規則
- 將事件與您的威脅情報關聯，現在又與 Microsoft URL 情報相關聯



打獵

- 運行內置的威脅 hunting 查詢 —— 無需事先有查詢經驗
- 使用 KQL 自定義並建立您自己的 hunting 查詢
- 整合 hunting 與調查



Microsoft Sentinel 簡要介紹

自動化

- 構建自動化且可擴展的 playbooks，實現跨工具集成
- 從樣本庫中選擇
- 使用 200+ 內置連接器建立您自己的 playbook。
- 從警報或事件調查中觸發 playbook

事件管理

將事件分配給分析師

開工單（ServiceNow/Jira）

保持事件狀態同步

在 Teams 或 Slack 頻道發帖

豐富化+調查

查找 Geo 以獲取 IP

Trigger Defender ATP 調查

向使用者發送驗證郵件

修復

遮罩IP位址

阻止用戶訪問

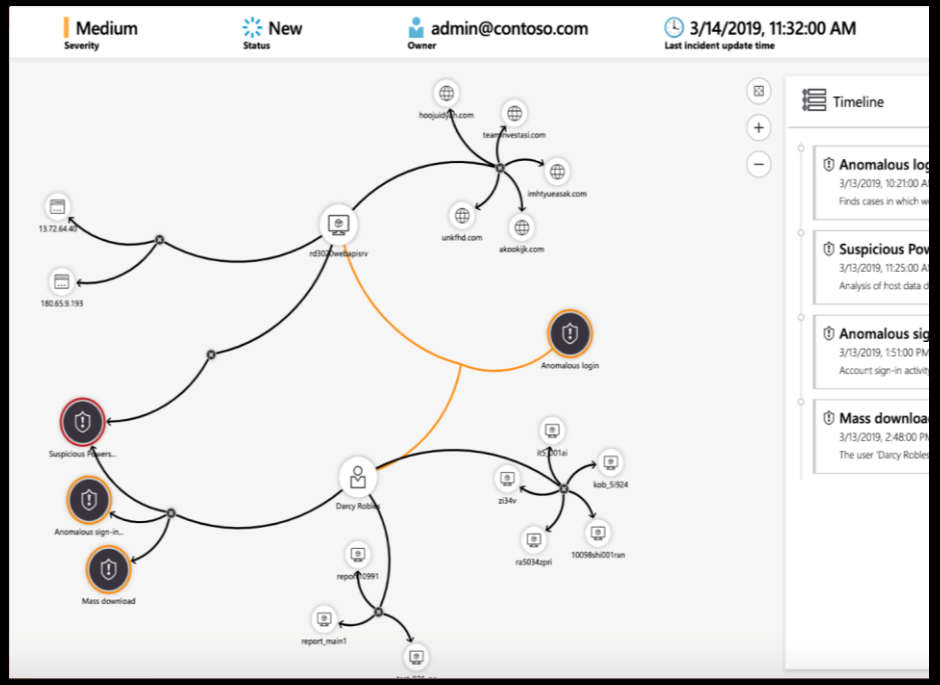
觸發條件訪問

隔離機

Microsoft Sentinel 簡要介紹

探討

- 在相關報警、書籤和實體之間進行關係導航
- 通過探索查詢擴展範圍
- 查看相關報警、事件和書籤的時間線
- 深入瞭解相關實體 —— 使用者、功能變數名稱等



基礎監控

最低

- 基礎設施監控
- APM 風格（例如：Application Insights 或类似格式）

現代監測

- 基於 SRE 概念的監測
  - SLI、SLO、誤差預算
  - https://learn.microsoft.com/en-us/azure/site-reliability-engineering/articles/devops

其他：

Microsoft Defender for Cloud |DevOps 安全性

謝謝！