Microsoft Azure

Secure DevOps：應用安全原則與實踐

手動安全驗證

模組概述

- 安全測試概述
- 需求與設計驗證
- 代碼審查

Microsoft Azure

安全測試概述

功能測試與安全測試

功能測試和安全測試並不相同

功能測試：驗證應用程式是否可供合法使用者使用

安全測試：驗證應用程式不會被惡意用戶濫用

安全測試技巧

安全測試的單一規則：沒有規則！

"惡意使用者不會這麼做" —— 他們會的！

惡意用戶會試圖繞過應用用戶端元件

惡意使用者將嘗試破壞應用程式的依賴項

Microsoft Azure

需求與設計驗證

安全要求

已發佈的當前要求清單

涵蓋組織的安全與隱私政策以及任何監管要求

在 Microsoft 中叫 *Liquid*

安全漏洞欄（security bug bar）

管理漏洞

我們如何處理開發、測試和生產過程中發現的安全漏洞?

定義 security bug bar（安全漏洞列）

定義了我們如何處理安全漏洞

包括威脅，例如（按嚴重到輕重排列）：

- 特權的提升
- 拒絕服務
- 定向資訊披露
- 欺騙
- 篡改

安全防漏洞條

需要有一致的方法來確定安全漏洞的風險

並非所有安全漏洞都一樣

CVSS 中的一個行業標準 —— 通用漏洞評分系統

Microsoft 也有 MSRC BugBar （並且使用 CVSS！ ）

CVSS

一種描述漏洞風險的方式

由事件回應和安全團隊論壇（FIRST）管理

當前版本是3.1

有三種測量方式：

1. 漏洞固有特性的基本指標
2. 隨著漏洞生命週期變化的特徵的時間指標
3. 依賴於實現或環境的漏洞的環境指標

這可能很複雜

$$Exploitability = 20 \times AccessVector \times AttackComplexity \times Authentication$$

$$Impact = 10.41 \times (1 - (1 - ConfImpact) \times (1 - IntegImpact) \times (1 - AvailImpact))$$

$$f(Impact) = \begin{cases} 0, & \text{if } Impact = 0 \\ 1.176, & \text{otherwise} \end{cases}$$

$$BaseScore = roundTo1Decimal(((0.6 \times Impact) + (0.4 \times Exploitability) - 1.5) \times f(Impact))$$

CVSS —— 一個例子

5. VMware 訪客主機逃逸漏洞（CVE-2012-1516）

CVSS v3.1 Base Score: 9.9

Metric	Value	Comments
Attack Vector	Network	VMX process is bound to the network stack and the attacker can send RPC commands remotely.
Attack Complexity	Low	The only required condition for this attack is for virtual machines to have 4GB of memory. Virtual machines that have less than 4GB of memory are not affected.
Privileges Required	Low	The attacker must have access to the guest virtual machine. This is easy in a tenant environment.
User Interaction	None	The attacker requires no user interaction to successfully exploit the vulnerability. RPC commands can be sent anytime.
Scope	Changed	The vulnerable component is a VMX process that can only be accessed from the guest virtual machine. The impacted component is the host operating system which has separate authorization authority from the guest virtual machine.
Confidentiality	High	Full compromise of the host operating system via remote code execution.
Integrity	High	Full compromise of the host operating system via remote code execution.
Availability	High	Full compromise of the host operating system via remote code execution.

CVSS：3.0/AV：N/AC：L/PR：L/UI：N/S：C/C：H/I：H/A：H

The MSRC BugBar

旨在幫助識別安全漏洞的風險
一套客觀條件有助於保持過程的一致性
根據你的 Policy 自行制定

危急	現在就修
重要	在下次更新前修復
溫和	下次更新修復
低	可能不需要修理

代碼審查

為什麼要手動審查安全代碼？

市場上已經有許多自動化工具

那麼，為什麼要手動審查代碼呢？

自動化工具遠非完美

- 他們通常會發現一些問題
- 它們容易出現誤報
- 在減少誤報的過程中，他們忽略了實際問題

因此，自動化工具應配合手動代碼審查！

手動代碼審查的缺點

手動代碼審查很慢

基於 Microsoft 的測量數據

- 一個優秀的安全代碼審查員每天最多只能做幾千行代碼

沒有什麼能替代由專業人士手動進行的代碼審查

你可以遵循一些模式來加快進度

優先考慮審查哪段代碼

假設有大量代碼需要審查，如何優先確定哪些代碼必須先審查？

最重要的因素是代碼的攻擊面

這些細節可以在你的威脅模型中找到

- 威脅模型包括介面和信任邊界
- 高攻擊面代碼應儘早且最深入地審查
- 匿名且遠端訪問的代碼是最脆弱的

安全代碼審查的黃金法則

大多數安全漏洞都是由不可信的數據引起的

“所有輸入都是有害的，除非被證明不是”

“安全系統是一個只做它應該做的事，別無其他功能的系統。”

面對不受信任的輸入，系統可能會開始做出非預期的作（例如：SQL 注入）

謝謝！