應用系統安全系列：在 DevOps 中達成安全性

The AppSec How-to: Achieving Security in DevOps

資料來源: Checkmarx

翻譯整理: 叡揚資訊 資安事業處

在持續部署 (CD，Continuous Deployment) 環境中，每 5 分鐘就需要發布一項新功能、擴充項目或錯誤修復持續部署，您應該如何整合安全性呢？

在步調快速的環境之中，大家已不想用需要冗長環境設定、調整及學習的傳統資安檢測工具。然而，只靠開發人員的安全程式設計（Secure coding）仍不足夠。

安全程式設計需要新方法，其中，資安工具可成為開發環境的一部分 – 並減少其他不必要的負擔。透過與開發團隊協同作業、瞭解其需要與需求，您即可在數分鐘內為安全部署做好準備。

DevOps 整體來說是什麼？

DevOps 是一種持續部署流程，它會在很短的時間，經常部署小功能及錯誤修正。做為一種嶄新的開發方法，DevOps 並不僅限於年輕的新創公司。像是 Facebook、Netflix、Etsy、LinkedIn 及 Twitter 等無數的大企業均已採用 DevOps。嚴格遵守 DevOps 模型的 Amazon，據說在 1 小時內即有超過 1,000 個部署。[1]

傳統 vs 破壞：DevOps 環境中的網路應用程式

傳統的 Web 應用程式安全檢測，是否能融入破壞性的 DevOps 環境中呢？讓我們來審視一下常見的 Web 應用程式資安檢測工具：

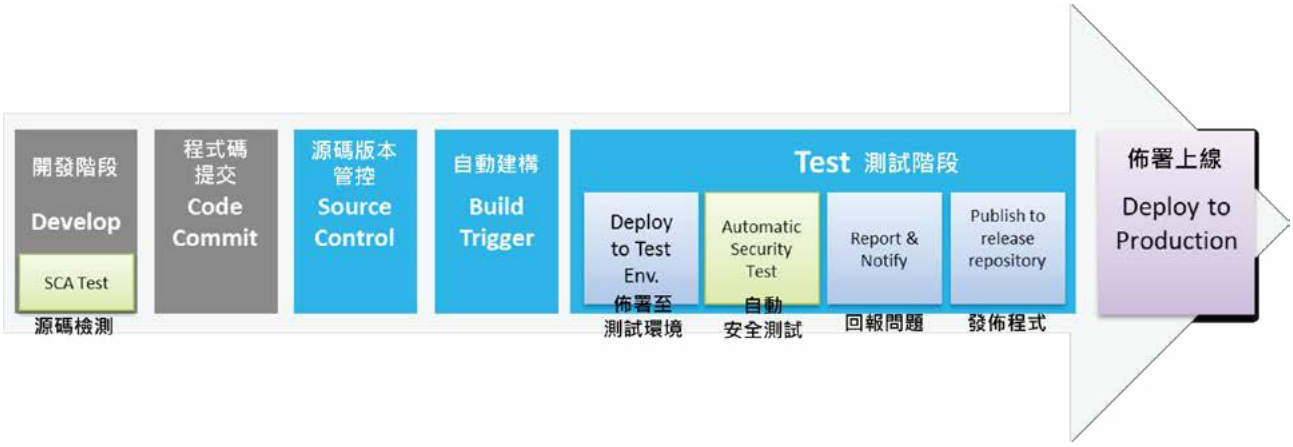
- 滲透測試。 一種最有價值的安全性測試方法，但有個與生俱來的問題：費時。滲透測試不論是在內部執行或由協力廠商執行，都需要耗費數天的時間來測試應用程式，然後再花些時間才能得出調查結果。在最後呈現調查結果時，還需要花時間分析結果、集合受影響的開發小組，然後排定工作的優先順序。經歷為期三周的評估循環，包含兩天的後續追蹤分析，及兩週的時間將問題併入開發流程內進行修正後，大型專案通常會出現一份 300 頁的調查結果報告。

1 http://www.slideshare.net/AmazonWebServices/advanced-topics-session-1-continuous-deploymentpracticesonaws

- 網頁應用程式防火牆 (WAF) 。WAF 需要調整與學習其所要保護的應用程式。就變動不太大的應用程式而言，需要數小時到數天的時間進行 WAF 的設定。但當應用程式不斷變動時，會發生什麼事情呢？在此情況時，WAF 需要不斷的更改設定，對於動態程序來說不是一個有效的解決之道。
- 程式碼分析。此方法就只是因太慢而壞了名聲。無論是設定時間、執行時間或分析時間 – 任何要花上數秒以上時間的方法，都無法真正整合在 DevOps 之中。

需求：新的安全軟體開發生命週期 (SDLC) 方法

解決之道為，從開發流程一開始就加入安全性。從安全性的觀點考量專案，並使安全性成為 SDLC 中的默認流程。



下述步驟有助於您達成此目標。

步驟 1：安全計劃

請研究在開發與部署過程中，您將會遇到那些技術與流程。藉以考量其安全性面向：

1. 技術的安全性

A. 鑑定不安全的元件及架構。例如，某些組織會分析整個程式碼底層，以找出所有不安全的模式、架構及函式庫。

B. 選擇內建安全模式的程式語言。例如，PHP 在新版本發佈前，都會公告舊版本中將要廢除的不安全模式。同樣地，幾乎所有架構都有安全性缺口，也會針對這些缺口提供所需的修正。

2. 程式碼開發的安全性

A. 找出程式碼裡有安全性疑慮的部分。程式碼並非都需一樣的安全性。例如，您測試函式庫的安全性絕對不如密碼更換機制、使用者驗證機制或信用卡處理機制同樣重要。

B. 對有安全性敏感度高的程式碼必需注意更多的安全性措施。當程式有被標記敏感性，且這些模組套用變更時，這些模組需觸發程式碼審查、特殊測試，以及這些模組專用的個別掃描。

3. 功能的安全性

A. 預先考慮安全性問題並做好規劃。您最後終將遇到它。若未事先做好此事的準備，則會在之後因產品變更、附加元件以及已架構好的程式碼修改時，讓您蒙受損失。在程式設計階段，需訂出程式安全標準。在測試流程中，也需將相關的驗證方式規劃好

步驟 2：凝聚開發人員的向心力，縮短開發與資安團隊的距離

DevOps 將開發人員視為流程的核心。而且，開發人員需保持高品質且標準的程式碼產出。資安團隊如何傳達安全程式碼的嚴重性與重要性？

有許多公司已經發現了下列的建議，有助於縮短安全性與開發人員之間的落差：

1. 使開發人員具有安全性意識。

- **在每個開發團隊中建立「Security Champion(資安種子專家)」機制**。與團隊分享有關最新的威脅概況及駭客動機的資安文章。共同參加當地的 OWASP 訓練。
- **讓安全性訓練更有價值**。指導開發人員有效地解讀漏洞描述、傳達程式碼中漏洞的風險，以及探討正確的漏洞解決策略。透過安全開發演練，向開發人員提出其常見、經常發生的程式碼撰寫問題。
- **分享攻擊的經過**。向開發人員講述安全性與駭客的真實現況。出示駭客嘗試入侵的紀錄，以展現之前在安全性開發演練中使用的程式碼如何阻止攻擊。

2. 設置線上協同作業平台。例如，在任何分享與協同作業平台（如 Jive 或 Confluence）上，張貼安全性問題並呈現解決或防止問題的方法。甚至進一步地建立分享專門分享安全性問題的協同作業平台。

3. 提供暢通諮詢管道。當開發人員提問時，能即時從旁提出協助。例如，與開發人員合作修復並避免較不為人知的程式碼瑕疵。

步驟 3：讓開發人員整裝待發。

提供開發人員合適的工具，協助他們避免並減輕安全性漏洞。

1. 安全架構

安全架構是您保護基底中的程式碼安全的內建工具。目前有相當多樣的安全架構可供選擇，例如：

SpringSecurity、JAAS、Apache、Shiro、Java SE、Symfony2。而且，對於輸入驗證、驗證及工作階段管理，Ruby on Rails 備有非常廣泛的安全性解決方案。針對各種不同的程式語言，OWASP 也提供了開放原始碼安全性架構 (ESAPI)。

2. 使用原始碼分析工具，獲得提交前階段的安全性意見回饋

執行來源程式碼分析工具似乎與本文的前言相互牴觸，在前言中，此方法因為耗費過多時間而不受青睞。如之前所述，在需要每幾分鐘就交付的 DevOps 環境下，是無法忍受因安全性掃描而出現的任何延遲。但因為開發環境更迭，所以會調整不同的掃描工具，以利於開發團隊能快速做出因應之道。開發人員該如何善用這些新的掃描功能呢？

A. 掃描少部分的程式碼。

僅針對上一次掃描與當下掃描之間有變動之處，進行掃描。依此方式，掃描工具可掃描少部分的程式碼部分，而不須花數小時的時間設定與掃描專案。

B. 從開發環境內存取工具。

由開發人員負責在其選擇的 IDE 環境內測試其本身的程式碼。此亦應包括測試程式碼的安全性。開發人員亦可透過程式碼審查或使用 SCA 工具進行此動作。唯有在開發人員對於其程式碼安全無虞有信心時，才可將程式碼提交到原始碼儲存庫。

步驟 4：將流程自動化

DevOps 的骨幹是自動化。資安亦應如此。資安應先融入標準自動化持續部署流程之中。第二步則是使用應用程式安全性測試工具 – 無論是靜態或動態 – 此類工具能夠在非常短的時間內產生結果。

1. 在您的建置流程 (Jenkins、Bamboo, TeamCity, etc.) 中整合不同的應用程式安全性工具如靜態應用程式安全性測試 (SAST) 與動態應用程式安全性測試 (DAST) 等。

於提交程式碼後，建置流程中 – 一般是透過例如 Jenkins 或 Bamboo 等工具 – 應觸發動態與靜態測試工具進行掃描。靜態源碼掃描工具會執行全面的掃描，以涵蓋由多位開發人員同時提交的情況。動態測試工具則是一種自我學習環境，其會監控針對一般測試工具撰寫的正向測試 (Positive Test)。此工具也會執行負向測試 (Negative Test) 的輸入，以驗證正向測試未抓到的輸入。

2. 中斷未達標準的流程。

我們瞭解當您最初聽到此概念時，很可能會嗤之以鼻。但正如未通過開發階段的高優先順序 Bug 一樣，資安也應享有同樣的重要性。

步驟 5：聰明地使用舊工具

不要馬上丟棄舊工具。請保留舊工具，但以不同的方式運用：

1. 滲透測試

下令定期進行滲透測試（例如每六個月），以確保您的系統屬於軍規等級。在此階段，若不存在系統漏洞，則測試的益處不大，但這些動作可讓您的系統獲得雙重保障。

此外，請您的客戶對您的系統執行滲透測試。第一個原因是，由於某些客戶需要稽核協力廠商系統才能滿足標準，因此，這可能是必要條件。再者，雲端環境關係是立基於提供者與客戶之間的信任感。請客戶對您的系統執行滲透測試，將可提升此信任程度。當安全性整合至您的系統時，您即可確保不會有任何漏洞被發現。

2. Web 應用程式防火牆 (WAF)

將 WAF 作為使 Web 應用程式部分更穩定的解決之道。維護 WAF，每隔一段時間就進行維護調校作業，以確保 WAF 仍防護著不太常變更的主要功能。

3. 源碼檢測

對於安全性敏感度高的程式碼，執行源碼檢測 (source code review)。例如，對身份驗證與信用卡處理的程式碼進行程式式碼審查確保安全性。

DevOps 正在發生，RIGHT NOW，最後一點建議

資安可以且應當是持續部署流程中的一環。但一開始應小規模進行，以免被壓得喘不過氣來，並使得流程難以實行。請從較容易存取、較不關鍵的功能開始實行，並在每次開發之間建置安全性流程。最後，您即可因安全性強化功能而達成小小的成功，這點可從漏洞意見回饋越來越少而得證。請向管理階層回報這些成果，並獲得管理階層支持，以將安全性整合到開發生命週期的每一環節之中。