CYBERARK
CONJUR

白皮書

技術概述

CyberArk.com

Conjur 是用於為開發及維運團隊及其整個基礎架構提供目錄服務、授權及稽核的原生支援雲端的平台。

Gartner 預計，到 2016 年，25% 的全球 2000 大企業將已制定主流 DevOps 實施策略，意味著資安團隊需要立即開始轉型。

Conjur 100% API 涵蓋率、可擴展、可輕鬆部署的高可用性架構，相比於複雜的自己編寫程式碼 (homegrown script) 及組態設定管理工具，可協助節省時間，降低與建構授權管理系統的相關成本及複雜性。Conjur 有時也被稱為「雲端 Active Directory」，Conjur 在虛擬機器或容器中運行，可與一系列身份及存取管理 (IAM) 解決方案協作，共同應對存取及授權挑戰：機器對機器間的權限、機敏系統部署及存取權限，以及滿足合規要求所需的稽核。

為什麼選擇 Conjur?

創新的腳步相當迅速。系統是即時上架、下架或更版；Conjur 可協助企業應對此急速變革的時代、適應並降低資安風險，同時滿足合規要求，而不會減緩其持續整合工作流程，避免將祕密、金鑰、憑證及身份認證資料保存在儲存庫及硬碟中，同時確保其安全。

由於在建構時考慮到系統管理者、DevOps 專家及雲端架構師的需求，Conjur 可以原生方式支援一系列可能的應用場景。企業不內部自建解決方案，而選擇 Conjur 的一些最常見的原因包括：

- **合規性：**透過現有報表系統 (如 SIEM 平台、SumoLogic、Splunk)，以可稽核的方式落實組織及監管政策與法規
- **風險管理：**利用 Conjur 的政策及監管平台，減少機敏性資料 (帳密、SSL/SSH 金鑰及憑證、祕密等) 可被攻擊的面向
- **實現最佳 DevOps：**在上游開發及維運工作中整合資安與控制措施，確保所有線上系統的一致性
- **存取情報：**在整個基礎架構 (裸機、私有雲、公有雲) 中統一控制身份 (人類及機器) 及權限有助於防止故障發生，而無需依賴舊型系統來強制執行原則與監管

常見應用場景

雖然 Conjur 可用於一系列應用程式，但其中三種應用程式最為常見，包括：

SSH 存取管理

Conjur 提供強大、便捷、基於標準的 SSH 存取功能，可用於存取雲端伺服器及 VM，並為 SSH 登入提供身份認證及授權功能。

- **透過公開金鑰進行身份認證。**公開金鑰不需要複製或以其它方式分發給每台伺服器或 VM。Conjur 會在登入時為 SSH 動態分配公開金鑰。
- **SSH 連線透過身份認證後即獲得授權。**授權是獨立於身份認證的步驟，在此登入時，系統將決定認證已認證是否可登入存取及其存取等級(透過群組成員資格分配)

身份認證及授權是即時的，且授予及撤銷存取權限的操作會立即生效。

接受 SSH 存取的伺服器應將 Conjur 作為身份認證及授權機制提供者，並且必須在 Conjur 中設置「主機」身份識別。

要透過 SSH 存取伺服器，使用者需要：

1. 具有相關 Conjur 使用者身份識別 (擁有已註冊的個人公開金鑰)。請注意，不得提前在目標環境中建立任何系統使用者
2. 使用與已註冊公開金鑰相匹配的個人 SSH 秘鑰
3. 被授予適當的特權等級 (通常透過群組成員資格分配) 可存取目標主機

根據該成員群組在 Conjur 權限 '於成功登入後，使用者在存取目標系統時可能具有或不具有特權 (「sudo」)。

目標系統上發生的所有授權事件將自動發送給 Conjur 並成為稽核紀錄的一部分。

有關詳細資訊，請瀏覽：
http://developer.conjur.net/tutorials/ssh。

2016 年技術支出增加情況——前五名

- 46% - 資安技術
- 42% - 雲端運算
- 38% - 業務分析
- 36% - 儲存
- 35% - 無線 / 行動

Secrets 管理

自動化情景，特別是持續整合及部署，通常不可避免地需要處理 secrets (加密金鑰、API 金鑰、密碼等)。傳統的處理方法是將 secrets 保存在程式碼儲存庫、組態設定管理工具或伺服器檔案系統上的檔案中。

這些做法會引起兩個主要資安風險：

- 透過存取伺服器檔案系統或伺服器檔案系統快照，即可輕鬆侵入祕密資料
- 很難反查誰存取了祕密資料，實作精細權限控制往往也非常繁瑣

Conjur 提供集中處理祕密資料的解決方案，附有完整的稽核紀錄，可協助解決這些問題。每項 secret 都儲存在 Conjur 伺服器上受到存取控制與加密的容器中。執行這些操作的典型工作流程如下：

1. 有權管理特定祕密儲存庫的 Conjur 使用者將「secrets」儲存到合適的容器中 —— Conjur 支援各種類型的容器 (從二進位到 ASCII 文字)
2. 有權提取特定 secret 以進行自動化處理的 Conjur 主機使用「conjur env」工具從這些容器中獲取相關值，然後再透過環境變數及 / 或暫存檔案將其提供給呼叫的程式碼，避免使用永久檔案系統儲存及執行日誌以最大幅度降低外洩風險
3. 如果需要，可以允許 Conjur 使用者提取特定 secret，這時就可以使用 Conjur CLI 工具集操作。

4. （選項）有權更新 secret 的 Conjur 主機可以執行自訂程式碼來執行輪換

對 secrets 操作的所有記錄將保留在稽核紀錄中；有關詳細資訊，請瀏覽：http://developer.conjur.net/tutorials/secrets

服務對服務之間的授權

在服務導向架構中，通常必須限制某些服務之間彼此溝通，為滿足合規要求，可能需要能夠稽核服務間呼叫成功及失敗的結果。

Conjur 的基於角色的存取控制（RBAC）功能可滿足這兩方面的需求：

1. 任何需要呼叫其它服務的服務實例都應有 Conjur 主機身份識別

2. 權限模型（很可能在外部建立）應接受審查，以確保權限模型能說明哪些主機 / 層級可以與其它主機 / 層級進行通訊。

3. 對外呼叫其他服務時，應透過身份認證代理機制（可以為其新增 Conjur 身份認證標頭）

4. 應將受保護的服務對應到授權代理器 (proxy)，後者能夠識別傳入請求中的身份認證權杖，並透過 Conjur 伺服器執行權限檢查，以確定是否允許該請求

如需執行概念驗證，可以使用以下 nginx 外掛程式：

https://github.com/conjurdemos/forward-proxy-nginx
https://github.com/conjurdemos/service-to-service-nginx-lua

架構及部署

Conjur 所採用的設計及架構旨在實現以下兩個主要目標：1）提供易於使用、有據可查、可擴展的授權系統，及 2）便於以無干擾的方式整合到企業工作流程中。這種方法在實作時具有諸多優勢：

- **易於整合：**Conjur 內建實現最佳 DevOps 運作的使用者經驗（由 CLI 驅動），100% 透過 API 存取，支援一系列語言，並且可部署為 Linux 虛擬設備或部署到裸機系統中

- **不受供應商限制：**Conjur 相容於各種主要的組態設定管理及 DevOps 工具（Puppet、Chef、Salt、Docker 等）

- **HA 設計：**Conjur 採用分散式組態設定架構與全面的故障復原 / 備援功能，可原生支援高可用性及容錯

- **相依性最低：**Conjur 配備齊全，不需要外部資料庫或伺服器

- **安全：**依照業界針對傳輸中及靜止中資料的最佳實踐，所有通訊全面加密；此外，Conjur 還進行嚴格的第三方安全性驗測分析。

架構概述

Conjur 伺服器

基本 Conjur 安裝只需一台伺服器，在本機 Postgres 資料庫中運行。

Conjur 會公佈 RESTful API/HTTPS（用於身份認證、權限管理、權限檢查、操作 secrets 及存取稽核紀錄）及支援傳統 LDAPS，這二者均提供目錄功能（如 bind 及搜尋），讓第三方系統可輕鬆透過設定 LDAP 來利用 Conjur。

加密

Conjur 中保存的所有靜止的機敏用戶端資料均已加密，如果沒有主金鑰（單獨儲存在 DB 之外）將無法存取這些資料。與用戶端的所有通訊都採用 SSL。

附註：可以使用自訂憑證或在安裝伺服器時自動產生的自簽憑證。

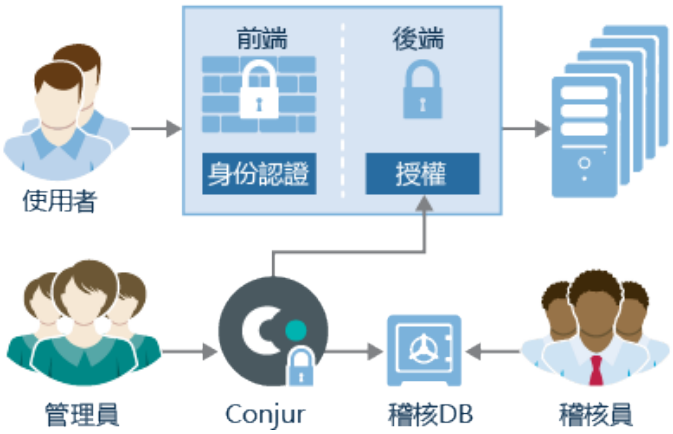
身份認證主要處理使用者身份識別：這個人是誰？他/她的身份是否真實？

授權則回答一組不同的問題：應允許此使用者（或系統授權，可以是服務間以及使用者與服務間的權限）存取哪些內容？

身份認證

Conjur 不僅支援人類身份識別（「使用者」），而且支援虛擬機器、處理程序 (process) 及作業 (job) 的身份識別（在所有文件中，非人類身份識別稱為「主機 (host)」）。

幾乎所有操作（「登入」及「讀取存取」使用者公開金鑰例外）都需要有效的 Conjur 特定身份認證權杖，這是有時效



- 45% 的駭客針對特權帳號憑證發動攻擊，以存取大量關鍵資料
- 33% 的駭客設法竊取終端使用者憑證及帳密

性、經過加密簽名的 Conjur 身份識別證據。有關詳細資訊，請瀏覽：http://developer.conjur.net/reference/services/authentication。

這種身份認證機制不應被視為通用身份認證協議的替代方案，而是可以並建議以程式設計方式將外部身份認證系統對應到 Conjur 的內部身份認證機制中，具體方式取決於現有身份認證基礎架構而異。

稽核

權限或 secret 管理以及權限檢查等操作將自動記錄到稽核軌跡中，這些軌跡可用易於讀取的格式查看，也可以用在存取情報收集上，匯出為 JSON 串流。

管理者還可以新增自訂記錄到稽核軌跡中，例如在其他基礎架構中產生的事件。

授權

Conjur 本身沒有「超級使用者」的概念，也沒有預先定義權限階層結構。管理者必須新增並套用每項規則。

根據指派的角色，每個使用者或主機都具有特定權限，以及授予給該角色的權限。

Conjur 中預先定義實體只有「admin」（初期用來新增真正管理者身份的使用者帳號）及一個空的「金鑰管理者」(key-managers) 群組（負責管理 Conjur 使用者的公開金鑰目錄）。

管理者將設定符合企業需求的權限模型：新增使用者及主機身份，為使用者劃分群組，為主機劃分層級，在它們之間授予權限，並使用下面介紹的用戶端工具來完成此任務。

使用者目錄可以從零開始建立，也可以從外部來源（如 LDAP）匯入。

雖然可建構簡單的權限模型涵蓋群組、層級及資源，然後在它們之間授予權限，但也可以透過載入以 Ruby 為基礎的 DSL 所撰寫的 Conjur「原則」，一次性佈建更加複雜的模型。

有關詳細資訊及按分驟教程，請瀏覽：http://developer.conjur.net/tutorials/authorization。

高可用性設計

Conjur 架構可以從一台獨立伺服器擴展為高可用性架構，其中包括：

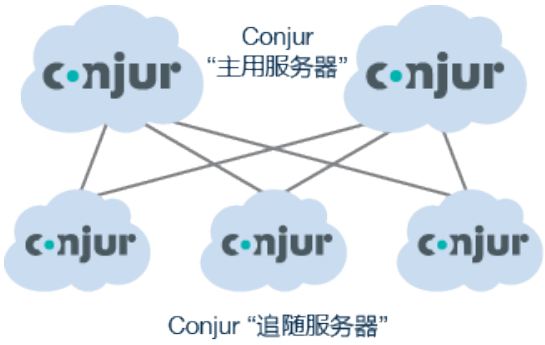
1. 主要 (master) 伺服器：處理目錄及稽核記錄，管理所有權限更新操作，並保管用於簽署追蹤者 (follower) 伺服器憑證的 SSL 根憑證

2. 備用 (Standy) 伺服器：為主要伺服器的唯讀副本，如果主要伺服器出現故障，備用伺服器將替代主要伺服器

3. 追蹤者 (follower) 伺服器：可以執行所有唯讀功能，如權限檢查、分配公開金鑰以及提供 secret 存取。追蹤者伺服器會透過頻外通訊將流水帳傳遞給主要伺服器

主要 - 追蹤者架構可達到：

- **全球分佈：**主要伺服器及追蹤者伺服器可以分散到可用性區域、各個地區及雲端。
- **低延時:**在地理區位上，將追蹤者節點安置在伺服器附近，就可以在需要時提供低延時的 Conjur 連線
- **高可用性：**任何追蹤者伺服器或主要伺服器都可以呼叫 Conjur 服務。因此，如果主要伺服器或追蹤者伺服器失聯，任何其它主要伺服器或追蹤者伺服器都可以接管其責任。故障復原可以手動執行，也可以透過 IaaS 的運行狀況檢查及自動擴展功能自動執行。
- **雲端友善的網路架構：**Conjur 存取只需 443 (https) 及 636 (LDAPS) 兩個網路埠即可：透過將追蹤者伺服器分佈到所需位置，就不需要設定複雜的網路安全及路由



組態設定以便伺服器聯繫 Conjur；追蹤者伺服器應放置到所需位置（如私有子網路中），並連線到主要伺服器以便進行即時複製

一般是透過 AWS CloudFormation 範本來佈建 HA 元件，不過，也可以在 AWS 以外的平台上運用自訂程式碼進行佈建。

在佈建追蹤者伺服器的過程中，一台新追蹤者伺服器是用所提供的 SSH 金鑰作為參數來連線到主要伺服器。建立連線後，這台新追蹤者伺服器將從主要伺服器下載初始化資料，完成對 Conjur 服務的設定。在此組態設定過程中，將為追蹤者伺服器產生 SSL 憑證並由主要伺服器簽名，然後安裝在追蹤者伺服器中，其中包括用戶端 VM 與 Conjur 通訊所使用的特定環境的主機名稱。

SSL 根憑證僅存在於主要伺服器及備用伺服器中，不存在於追蹤者伺服器中。因此，無法將追蹤者伺服器提升為主要伺服器。

有關詳細資訊，請瀏覽：http://developer.conjur.net/reference/architecture/ha.html。

了解详情
www.conjur.net
blog.conjur.net

资源中心
info.conjur.net/resources

演示
info.conjur.net/demos