對本書的讚譽

Corey Ball 未辜負書名所賦予的承諾，書中內容誠如其名所言論，包含基本原理定義、常見 API 弱點，以及最佳攻擊手法的背後理論，並鼓勵讀者認真探索駭客的思維模式。本書從工具介紹和偵察行動切入，再全面性探討 API 模糊測試及挖掘複雜的存取控制漏洞，並透過實作練習提高學習效果。融合完整詳細的實作、提示和攻擊技巧，以及活生生的案例，本書堪稱是一場完整的 Web API 駭客研討會。

—— Erez Yalon
Checkmarx 安全研究副總裁暨 OWASP API 安全專案負責人

Corey Ball 以生動的方式導覽 API 的生命週期，不僅激勵讀者去瞭解更多資訊，更讓讀者渴望在合法的目標上嘗試學到的新知識。從基本概念到應用範例，再到工具介紹及詳細示範，本書應有盡有，這是一本關於 API 漏洞攻擊的重要礦脈，對於任何想要認真研究對手、評估對手或儘早執行安全性測試（DevSecOp）的人，應該要將本書放在隨手可得的案頭上。

—— Chris Roberts
Ethopass 戰略顧問暨國際副資訊安全長

對於打算從事滲透測試的人，本書能夠提供極大幫助，尤其作者介紹了許多相當不錯的安全檢測工具，可以處理現今 Web App 裡常見的 API 弱點。經驗老到的資安人員也能從本書獲得新的啟發，特別是許多實用的自動化技巧和繞過保護機制的手法，絕對能夠讓你在滲透測試競賽中脫穎而出。

—— Vickie Li
《Bug Bounty Bootcamp》作者

本書將為讀者開啟通往 Wep API 安全殿堂的大門，很多人對這個主題還有些陌生，作者藉由真實案例強調存取控制問題的重要性，實作練習課程可讓讀者一窺 API 安全性的來龍去脈、如何獲取豐厚的賞金回報，也能幫助各行各業提高整體 API 的安全性。

—— Inon Shkedy
Traceable AI 資安研究員暨 OWASP API 安全專案負責人

儘管網際網路充斥著各種有關網路安全的資訊，但有效執行 Web API 滲透測試的真知灼見卻如麟毛鳳角，本書恰可完全滿足這些需求，它不僅適合初級的網路安全從業人員，即使經驗豐富的資安專家亦能有所受益。

—— Cristi Vlad
網路安全分析師和滲透測試員

序

想像一下，若要匯款給朋友，你需要開啟一支應用程式（App），還要經過多次滑鼠點擊；或者，想要監測每日走路的步數、運動資料和營養資訊，可能需要動用三支不同App；又或者，想要比較各家航空公司的票價，需要手動逐一拜訪它們的訂票網站。

此情此景應該不陌生吧！因為不久前才歷經這些生活模式。但是 API 翻轉了這種運作模式，它們就像是一種接著劑，讓跨公司協作得以實現，並改變企業建構和維運 App 的方式，事實上，API 已遍及各領域，在 Akamai 公司 2018 年 10 月的一份報告中提到「所有 Web 流量中，呼叫 API 就佔了 83%」，佔比之高著實讓人震驚！

但就像網際網路上的多數情形，只要有什麼好用的東西，就會引起網路犯罪份子覬覦。對犯罪份子而言，API 絕對是有利可圖的肥水，最主要是這類服務具有兩種理想特性：(1) 豐富的機敏資訊來源；(2) 為數眾多的安全漏洞。

想像 API 在一般應用程式架構裡所扮演的角色，當經由行動 App 查看銀行餘額時，後端會有一組 API 負責請求該項資訊，再將請求結果回傳給 App；同樣地，向銀行申請融資時，會有一支 API 讓銀行請求你的徵信紀錄。API 位於使用者和後端機敏系統之間的關鍵位置，如果網路犯罪分子可以掌控 API 層，便可直接存取極具價值的資訊。

雖然 Web API 已達到前所未有的普及程度，但安全性卻沒有隨著精進。最近與一家百年歷史的能源公司之資安長聊天，發現他們整個組織都有使用 API，實在令人驚訝！然而，他也提到「只要深入瞭解，就會發現這些 API 所具有的權限已超乎所需。」

沒什麼好大驚小怪的！在不斷增加服務功能、向使用者推送新版本及持續修復錯蟲的重大壓力下，開發人員必須日以繼夜地重複著建構和提交的循環，而不是固定間隔幾個月發行一次新版本，因此，並沒有足夠時間讓他們思考每項變更對安全性所造成的衝擊，未能及時發現的漏洞就悄悄地溜進正式環境裡。

不幸地，鬆散的 API 安全實作往往造成難以預料的後果。以美國郵政署（USPS）為例，該機構發行一支名為 Informed Visibility（訊息能見度）的 API，可讓機構和用戶追蹤包裹的遞送進度，此 API 要求使用者提供身分驗證，以便透過 API 存取相關資訊，然而，在通過身分驗證後，該使用者卻可以查找其他人的帳戶資訊，因此造成 6000 萬筆左右的用戶資訊被暴露。

Peloton 這家健身公司的應用程式（甚至其設備）也搭配 API 來提供服務，但由於其中一支 API 無須通過身分驗證即可向 Peloton 伺服器發送請求並得到回應，請求者可透過此 API 查找任何 Peloton 設備（約 400 萬台）的帳戶資訊，甚至存取用戶的機敏資料，就連 Peloton 的知名用戶 -- 美國總統拜登 -- 也因此項漏洞而洩露個資。

第三個例子是電子支付公司 Venmo 透過 API 為其應用程式提供服務功能及連接到金融機構，其中一項行銷功能是透過 API 提供最近的匿名交易情形，雖然使用者界面已將機敏資訊隱匿掉了，但直接呼叫 API 時會回傳所有交易細節，別有用心的使用者藉由此 API，共搜括了約 2 億筆交易資料。

此類事件已司空見慣，以致 Gartner 這家分析公司預測，到 2022 年，API 疏失將成為「最常見的攻擊向量」，IBM 也提到三分之二的雲端資料外洩是因 API 設置不當所造成的。這些事件突顯需要新手法來保護 API 的安全，過去的應用程式安全解決方案只關注最常見的攻擊類型和漏洞。例如透過自動掃描程式搜索通用漏洞披露（CVE）資料庫，藉以查找 IT 系統中的缺陷，再由 *Web* 應用程式防火牆（WAF）即時監控網路流量，阻擋針對已知缺陷的惡意請求。這些工具非常適合檢測傳統威脅，卻無法解決 API 面臨的安全挑戰。

問題在於 API 漏洞並非通用性漏洞，不僅每個 API 有其獨特性，往往也和傳統應用程式的功能有所差異。USPS 的資料外洩並非安全設置

不良，而是程式邏輯上的缺失，亦即，程式邏輯的疏失而留下缺陷，讓通過身分驗證的合法使用者可以存取其他用戶的資料，這類缺陷屬於不當的物件授權，是程式邏輯未有效控制授權用戶的存取內容所導致的結果。

簡言之，對於每支 API，這些獨特的邏輯缺陷就是一項零時差（zeroday）漏洞。為了瞭解這些威脅所涵蓋的範圍，很需要一本教育滲透測試人員和 API 漏洞賞金獵人的書籍，而本書正扮演這個角色，此外，隨著資安需求往系統開發流程「左移」，API 安全不再侷限於機構資安部門的範疇，本書可以作為現今任何工程團隊的安全指南，協助他們在功能測試和單元測試時，也同步進行安全測試。

良好的 API 安全測試計畫應該是持續而全面的，一年才進行一兩次的安全測試是無法跟上版本更新步伐，相反地，安全測試應該成為開發生命週期的一部分，每個版本在投入正式環境之前都應通過適當審查，並且涵蓋 API 的整個足跡。想要找出 API 漏洞就需要新的手法、工具和技術，本書正可為現今世界提供重大貢獻。

Dan Barahona

APIsec 公司資安戰略長

於加州舊金山市

引言

依研究機構估計，因調用應用程式介面（API）而產生的 Web 流量已佔總流量 80% 以上，儘管 API 被大量採用，但 Web App 駭客卻未必對它進行充分測試，這些重要的業務資產可能滿布災難性的弱點。

本書將告訴你為何 API 是一種極佳的攻擊向量，歸根究柢，它們的目標就是要將資訊公開給其他應用程式。想危害或取得某機構的機敏資料，或許不需要應用高超技巧來穿透網路防火牆的保護、規避高階防毒軟體的偵測或使出零時差攻擊，只要簡單地向正確的端點發出 API 請求就可以完成任務。

本書將為讀者介紹 Web API，展示如何檢測它們的諸多弱點，主要是測試 REST API 的安全性，這是 Web App 中最常見的 API 格式，此外，也會說明如何攻擊 GraphQL API。

首先為讀者引介應用在 API 上的駭客工具和技術，接著說明如何探測及利用 API 的漏洞，當讀者具備這些能力，就可以回報所發現的漏洞，並幫助受駭者避免再次受到入侵。

本書亮點

國際商業資訊主要來源之一的《經濟學人》，曾在 2017 年發表「世界上最有價值的資源不再是石油，而是資料」（The world's most valuable resource is no longer oil, but data.）的觀點，而 API 正是讓這些珍貴寶物在眨眼間流向世界各地的數位渠道。

簡言之，API 是一種能夠在不同應用程式之間進行通訊的技術，假如要讓 Python 程式和 Java 程式的功能互動，情況將會變得棘手，若透過 API，開發人員便可設計出引用其他 App 的特殊功能之模組化應用程式，不必自己開發地圖服務、支付功能、機器學習演算法或身分驗證程序。

因此，現今許多 Web App 迫不及待地採用 API 技術，新穎技術通常會在網路安全問題出現之前廣為流通，這些 API 也因此大大地擴展了應用程式的攻擊表面，由於 API 的防禦能力很差，駭客可以利用它們作為竊取資料的捷徑，還有許多 API 缺乏防禦攻擊向量的安全措施，儼然成為死星（Death Star）的細小散熱孔（死星的致命點），讓企業走向毀滅的厄運。

基於上述原因，Gartner 多年前就預測，到了 2022 年 API 將成為主要的攻擊向量。身為白帽駭客，我們必須穿上直排輪、背上阿齊姆火箭（Acme rocket），以飛快的速度趕上技術創新的腳步，以便保護 API 的安全。透過攻擊 API、回報發現的漏洞，讓企業及早知曉面臨的風險，以盡一己之力來阻止網路犯罪。

編排方式

攻擊 API 並不如讀者所想地那麼具有挑戰性，一旦瞭解它們的運作方式，只要針對問題點發送正確的 HTTP 請求，就可以達成破解的目的，儘管如此，一般用來獵捕錯蟲和執行 Web App 滲透測試的工具及技術並無法完全適用在 API 攻擊上，例如，對 API 執行常見漏洞掃描，很難得到預期的有用結果。筆者對有缺失的 API 執行這類掃描，經常收到不正確的診斷結果，如果 API 沒有經過正確檢測，機構會得到一種安全假像，反而讓他們處於被入侵的風險之中。

本書內容採循序漸進方式編排，後面的內容都是奠基在前面內容之上：

PART I 關於 WEB API 的安全性：向讀者介紹 Web App 和支持它們的 API 之必要基本知識，你將學到 REST API（本書的重點項目）及日益流行的 GraphQL API，還會說明與 API 相關的常見漏洞，這些漏洞是讀者日後查找的重點。

PART II 建置測試 API 的實驗環境：指導讀者建構自己的 API 駭客攻擊系統及如何使用相關工具，包括 Burp Suite、Postman 和其他工具，還會建置幾套有漏洞的實驗環境，作為本書實作練習的攻擊目標。

PART III 攻擊 API：談論駭客攻擊 API 的方法，引導讀者完成常見的 API 攻擊。最有趣的事就從這裡開始，讀者將透過公開來源情資技巧找出潛在的 API，然後分析這些 API 的攻擊表面，最後深入研究攻擊它們的各種技巧，例如注入攻擊，讀者也會學到如何進行 API 的逆向工程、繞過其身分驗證管制及對各種安全問題執行模糊測試。

PART IV 真實的 API 入侵事件：最後一部分將為讀者展示有哪些因 API 弱點而造成的資料外洩事件，以及價值多少漏洞賞金，讓讀者瞭解駭客如何將本書介紹的技術應用在現實世界裡。還藉用一個簡單的 GraphQL API 攻擊範例，告訴讀者如何將本書前面介紹的技術修改成適用於 GraphQL 格式。

實作練習：PART II 和 PART III 的各章都有一項實作練習，可供讀者自我練習本書介紹的技巧。當然，讀者也可以使用其他非書中介紹的工具來完成這些實作，筆者非常鼓勵你利用這些實作練習來熟悉本書介紹的技術，然後試著使用自己的方法來完全相同任務。

本書是為任何想從事 Web API 駭侵攻擊及想要增加另一項戰技的滲透測試人員和漏洞賞金獵人而寫。筆者精心安排本書內容，讓初學者能夠在 PART I 學習到必要的 Web App 及 API 相關知識，在 PART II 建立實作練習環境，然後從 PART III 開始進行入侵活動。

攻擊 API 餐廳

進入主題之前，筆者先舉個比喻。假設應用程式是一家餐廳，API 的說明文件就像是菜單，會呈現你可以點餐的內容，而 API 本身就像服

務生，是顧客（你）和大廚（伺服器）之間的聯繫媒介，你可以根據菜單向服務生點餐（提出請求），服務生會為你送上餐點（得到回應）。

很重要的一點，顧客不需知道廚師如何烹調佳餚，API 使用者亦毋需知道後端應用程式如何運作，只要能夠按照正確指令發出請求，就應收到該有的回應，因此，開發人員可以設計他的應用程式來實現想要的功能。

然而身為一名 API 駭客，你將探索此餐廳的每個部分，瞭解餐廳的運作模式，甚至嘗試繞過它的「保全人員」，或者出示偷來的符記（token）來通過身分驗證。此外，還會分析菜單，尋找誘使 API 向你提供原本無權存取的資料之方法，可能是誘使服務生將他們所擁有的一切都交給你，甚至說服 API 擁有者將整間餐廳的鑰匙交給你。

本書藉由下列主題指導讀者以全面性的手法來攻擊 API：

- 瞭解 Web 應用程式的工作原理以及 Web API 的結構。
- 從駭客的角度看待主要的 API 漏洞。
- 學習最有效的 API 入侵工具。
- 透過被動式和主動式偵察找出 API 的蹤跡、已暴露的機敏資訊，並分析 API 的功能。
- 與 API 互動，並對它們執行模糊測試（fuzzing）。
- 對找到的 API 執行各種攻擊，以便利用所找到的漏洞。

整本書都是以駭客思維模式來壓榨 API 的功能和特性，越能以對手的角度來思考，就越能替 API 擁有者找出 API 弱點，甚至能夠防止下一波大規模的 API 資料外洩事件。

翻譯風格說明

資訊領域中，許多英文專有名詞翻譯成中文時，在意義上容易混淆，有些術語的中文譯詞相當混亂，例如 interface 有翻成「介面」或「界面」，為清楚傳達翻譯的意涵，特將本書有關術語之翻譯方式酌作如下說明，若與讀者的習慣用法不同，尚請體諒：

0

為滲透測試做好事前準備

API 安全檢測與一般滲透測試模式不太一樣，也和 Web 應用程式（Web App）滲透測試略有不同，由於許多機構的 API 攻擊表面相當複雜及龐大，因此 API 滲透測試服務有其獨特性。本章將討論在攻擊之前，應該放到測試項目和專案文件裡的 API 相關要求，本章內容可協助讀者評估參與專案所需的活動範圍，確保可為目標 API 規劃完整的測試內容，並避免可能招致的麻煩。

API 滲透測試需要一個明確的範圍，亦即被允許測試的目標和功能，確保客戶和測試人員都正確理解所要完成的工作，需要確定 API 安全測試範圍的主要因子有：使用的方法、測試的規模、針對的功能、測試活動的限制、測試報告的要求事項，以及是否需要提供複測服務。

取得授權

在攻擊 API 之前，最重要的是確認收到一份簽署完成的委託契約，裡頭包含雙方所議定的測試範圍，並授權你在特定時間內可攻擊的客戶資源。

對於 API 滲透測試，該契約可以是*已簽署的工作說明書*（SOW）之形式，明列已被認可的標的，確保你和客戶對所提供的測試服務有一致的理解，包括對 API 測試的方法、工具、範圍、排除項目及執行測試的時間所達成之協議。

要仔細檢查，簽署契約的人須足以代表客戶承諾測試授權，並確認待測試資產標的為該客戶所擁有，否則，必須將這份協議作廢，重新與真正的擁有者議定內容。務必考量客戶託管其 API 的位置，以及他們真的有資格授權對託管 API 的軟體和硬體進行測試。

有些機構會明文界定過於嚴苛的範圍，如果有機會擴大測試範圍，筆者建議以溫和語氣向客戶說明犯罪分子是不受限制的。真正的犯罪分子才不考慮被攻擊者是否會消耗大量 IT 資源，也不在意發動攻擊的時間點，更不受目標限制，只要伺服器保有機敏資料，就算是子網域也是下手的目標。簡單地說，犯罪分子不受協議裡的範圍和時間所限制，盡力讓客戶相信減少協議限制所帶來的好處，並和他們一起將協議的細節記錄下來。

與客戶面對面說明即將發生的事情，並精準地記錄在契約、注意事項或以電子郵件提醒。只要你恪守此份徵求服務的書面協議，應該就可合法且合乎道德地執行專案，然而，進一步諮詢律師或貴公司的法務部門以降低風險，這絕對是值得的。

為 API 測試進行威脅塑模

*威脅塑模*是用來描繪 API 提供者所面臨威脅的程序，若根據相關威脅對 API 滲透測試進行塑模，就能適當地選擇攻擊的工具和技術，最佳的 API 測試是可以契合 API 提供者所面臨的真正威脅。

*威脅參與者*是指攻擊 API 的人，包括無意間發現 API 弱點且幾乎不瞭解應用程式的一般公眾、使用該應用程式的客戶、惡意業務夥伴或對應用程式有相當瞭解的內部人員，為了執行具最大價值的 API 安全測試，最好能描繪出可能的駭客及使用的技術。

測試方法應該以威脅參與者的角度出發，這樣才能獲得有關目標的資訊。如果威脅參與者對 API 一無所知，他們就會研究該應用程式，確認可用的攻擊方法；若是惡意的業務夥伴或來自內部威脅，他們可能已經相當瞭解應用程式，不必進行多餘的偵察活動，為了區別彼此的差異，滲透測試有三種基本方式：黑箱、灰箱和白箱。

黑箱測試是機會主義駭客的威脅模型，他可能偶然間發現目標機構或其 API。參與真正的黑箱 API 測試時，客戶是不會向測試人員透露有關攻擊表面的任何資訊，只能從已簽署的工作說明書裡的公司名稱下手，因此，測試工作將涉及公開來源情資（OSINT）的運用，可以結合搜尋引擎、社群媒體、公開財務紀錄和 DNS 資訊等研究，盡可能瞭解組織的網域範圍，找出目標的攻擊表面。第 6 章會更詳細介紹這種方式的工具和技術。一旦完成 OSINT 研究，應該可彙編出目標的 IP 位址、URL 和 API 端點清單，再將這些清單呈給客戶審查，客戶完成目標清單審查後，應該會給予測試授權。

灰箱測試的協議範圍更加明確，可供我們合理地分配花費在偵察上的時間，而將主要時間投入主動測試。執行灰箱測試時，讀者將扮演見多識廣、消息靈通的駭客，客戶會提供一些特定資訊，例如目標範圍及 API 說明文件，甚至具基本權限的使用者帳戶，也可能同意你直接繞過某些網路邊界的安全管控機制。

漏洞賞金計畫通常介於黑箱測試和灰箱測試之間，漏洞賞金計畫是機構同意駭客測試其 Web App 漏洞的一種要約形式，一旦駭客成功找出並通報主機上的漏洞，將可獲得機構提供的賞金。漏洞賞金計畫並非完美的「黑箱」，因為機構會為賞金獵人提供允許測試的標的、不屬賞金計畫範圍的標的、獎勵的漏洞類型及可使用的攻擊型式，除了這些條件外，剩下限制賞金獵人行動的就是自己所擁有的資源了，因此他們懂得如何分配運用於偵察與其他技術上的時間。讀者若對於如何贏得漏洞賞金有興趣，筆者強烈推薦閱讀 Vickie Li 撰寫的《Bug Bounty Bootcamp》（*https://nostarch.com/bug-bounty-bootcamp*）

對於白箱測試，客戶會盡可能提供待測環境的內部運作資訊。除了為灰箱測試提供的資訊外，還可能包括應用程式的源碼、設計資訊、應用程式所用的軟體開發工具套件（SDK）、……等。白箱測試是以內部駭客的角度進行威脅塑模，內部駭客很清楚機構內部運作方式，並且能夠取得系統源碼。在白箱約定中為你提供的資訊越多，對目標的測試就越徹底。

客戶應該依據威脅模型和威脅情資決定簽署白箱、黑箱或灰箱測試的委託協議，測試人員應該透過威脅塑模和客戶一起分析機構最可能面臨的駭客。假設正與一家不涉及政治的小型企業洽談測試協議，它既沒有參與重要公司的供應鏈，也不提供重要的基礎服務，在這種情況下，若設定該公司是面對資源充足的國家級駭客之進階持續性威脅（APT），這就有違常理。對一家無舉足輕重的小企業動用 APT 技術，簡直是用大砲打小鳥、殺雞用牛刀，為了提供客戶最大價值的服務，應該使用威脅塑模闡述最能符合現實的威脅，以前述例子而言，攻擊行為最可能來自偶然撞見漏洞的機會主義者，這類人頂多具普普的駭客技巧，只會使用現成的漏洞攻擊手法打擊無意間找到的網站漏洞。要模擬機會主義者的攻擊，採用黑箱測試應該較為恰當。

為客戶建立威脅模型的最有效方法是與他們一起調查討論，這項行動必須找出客戶遭受攻擊的範圍、經濟上的意義、涉及政治的程度、是否參與任何供應鏈、是否提供重要基礎服務、是否有引誘潛在駭客的其他因素，讀者可以發展自己的調查方式或從現有的專業資源，例如 MITRE ATT&CK（*https://attack.mitre.org*）或 OWASP（*https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html*）下手。

不同的測試方法會影響調查的力氣，由於黑箱測試人員只知道極少關於測試範圍的資訊，其餘需要界定的項目就是灰箱和白箱測試裡所提示的資訊。

該測試哪些 API 功能

確定 API 安全測試協議範圍的主要目標之一是找出須完成某一部分的工作量，也就是說，必須找出有多少應測試的 API 端點、方法、版本、功能、身分驗證、授權機制及身份權限的種類，可以透過與客戶訪談、檢視相關的 API 說明文件及所存取的 API 集合來確認測試規模。得到所需資訊後，便能估計出需要多少小時才可有效完成客戶的 API 測試。

測試 API 的身分驗證功能

確認客戶打算如何測試使用者有無通過身分驗證的方式，或許他們想要你測試不同的 API 使用者和角色，以便判斷在不同授權層級中是否存在漏洞，也可能希望你測試身分驗證和使用者授權的程序是否妥當。一講到 API 弱點，許多有害的漏洞都是在身分驗證和授權的程序

中發現的，在進行黑箱測試時需要弄清楚目標的身分驗證程序，並想辦法通過身分驗證。

Web 應用程式防火牆

參與白箱測試時，可能會想知道客戶是否啟用 *Web 應用程式防火牆*（WAF），它是 Web App 和 API 的常用防禦機制，可以管制存取 API 的網路流量，若正確部署 WAF，測試時會發現單純的弱點掃描將無法碰觸到 API，WAF 有效地限制預期之外的請求，並阻止 API 安全測試，真正發揮效力的 WAF 會檢測請求的頻率或無效的請求，並封鎖來自測試設備的流量。

對於灰箱和白箱測試的委託，客戶可能會提供 WAF 資訊，如此便能事先建立一些測試決策。機構是否該降低防護強度以提高測試效果，此乃見人見智，多層次網路防禦（縱深防禦）是保護機構安全的重要關鍵，換句話說，誰也不該將所有雞蛋都放進 WAF 籃子裡，只要時間充裕，有決心的駭客可以逐漸拼湊出 WAF 的防護邊界，找出繞過它的方法，或藉由零時差漏洞讓它失去防護能力。

理想情況下，客戶可能允許你直接攻擊 IP 位址而繞過 WAF，或者調低邊界防護的安全等級，方便你測試開放的 API 之安全機制，如前所述，要制訂這樣的計畫和決策必須依靠威脅塑模。對 API 的最佳測試是能夠契合 API 提供者所面臨的威脅。為了獲得最大價值的 API 安全測試，須找出可能的駭客及其使用的駭侵技術，否則，你會發現是在測試客戶的 WAF 之防護能力，而不是 API 自己的安全控制能力。

測試行動 APP

許多機構因提供行動 APP，而放大攻擊表面，更甚者，行動 APP 通常透過 API 與伺服器交換資料，我們可以藉由人工檢視程式碼、自動源碼分析和動態掃描來檢驗這些 API。*人工檢視程式碼*必須先取得行動 APP 的源碼，才能搜尋裡頭的漏洞；*自動源碼分析*與人工檢視程式碼類似，只是改用自動化工具協助搜尋漏洞和駭客關注的部件；*動態分析*則是對運行中的應用系統進行測試，動態分析包括攔截行動 APP 端對 API 的請求和伺服器對此請求的回應結果，藉此嘗試找出可被利用的弱點。

檢視 API 說明文件

*API 說明文件*是指介紹 API 用法的手冊，包括對身分驗證的要求、使用者角色、使用範例和 API 端點資訊。良好文件是讓 API 能成功應用

於商業系統的要件，如果沒有完善的 API 說明文件，企業就必須透過教育訓練來指導使用 API 的開發人員，因此，可斷定機構一定留有待測目標的 API 說明文件。

然而，文件內容可能充斥著不夠精準或過時的資訊，也可能在文件中洩露機敏資訊。身為 API 駭客，一定要蒐索目標的 API 說明文件內容，從中取得優勢。對於灰箱和白箱測試，活動範圍應該包含檢視 API 說明文件，透過檢視文件裡所暴露的弱點（包括程式邏輯缺陷），可用來改善目標 API 的安全性。

測試請求速率限制能力

速率限制是指 API 使用者在特定時間間隔裡可發出的最大請求次數，它是受 API 提供者的 Web 伺服器、防火牆或 WAF 所控制，對 API 提供者而言，這有兩個重要目的，一個是讓 API 具有貨幣價值，另一個是防止過度消耗資源。由於速率限制是機構藉由 API 獲利的重要因素，在訂定 API 測試協議時，應該將速率限制測試納入執行範圍。

例如，某家公司可能只允許免費的 API 用戶在一小時內請求一次，一旦發出 API 請求，於一小時內再提出請求就不會得到該有的服務；如果 API 用戶向該公司支付費用，在相同的時間間隔裡則可發出上百次請求。如果沒有適當的管制措施，非付費的 API 使用者可能會想方設法繞過收費機制而盡可能地獲取服務。

速率限制測試與阻斷服務（DoS）測試不同，DoS 測試的目標是要找出可破壞服務而讓正常用戶無法使用系統和應用程式的攻擊。DoS 測試是評估機構的運算資源之復原能力，速率限制測試則是找出繞過單位時間內限制請求次數的方法，嘗試繞過速率限制不見得會導致服務中斷。相反地，繞過速率限制可能有助於執行其他攻擊，也可說明公司利用 API 謀利的技術存在缺陷。

一般而言，機構會在 API 說明文件敘明該 API 的請求限制，內容可能像：

> **你可以在 X 時間內提出 Y 個請求，若超出此限制，將會收到 Z 的回應訊息。**

以 Twitter 為例，通過身分驗證後會根據你的授權來限制請求，第一級是每 15 分鐘可以發出 15 個請求，再上一級是每 15 分鐘可以發出 180 個請求，如果超出請求限制就會收到 HTTP 420 的錯誤狀態碼，如圖 0-1 所示。

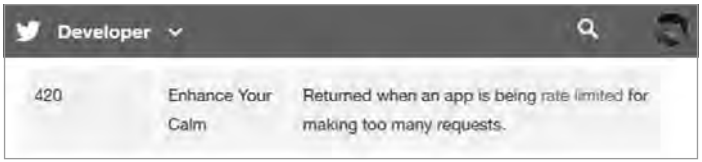


圖 0-1：來自 https://developer.twitter.com/en/docs 的 Twitter HTTP 狀態碼

如果沒有足夠的安全控制來限制對 API 的存取，API 提供者將因使用者欺騙系統而蒙受獲利損失，也因耗用額外的主機資源而增加營運成本，還可能容易受到 DoS 攻擊。

限制和排除條款

除非滲透測試授權文件另有規定，否則，應該假設不允許執行 DoS 和 DDoS 攻擊。以筆者經驗，業者鮮少授權執行此類攻擊，若業者授權執行 DoS 測試，必須明確地記錄於正式文件中。此外，除了某些完全模仿敵手的委託外，滲透測試和社交工程通常是分開執行，儘管如此，執行滲透測試時，仍請確認可否使用社交工程手法（例如網路釣魚、語音釣魚和簡訊釣魚）。

一般漏洞賞金計畫並不接受社交工程、DoS 或 DDoS 攻擊或直接攻擊使用者及存取客戶資料，對於可攻擊使用者的情況下，通常會建議建立多組帳戶，當出現可攻擊的機會時，請拿你自己的帳戶當作目標。

此外，有些賞金計畫或客戶會特別指明已知的問題，對於所發現的某些 API 問題可能被視為安全缺陷，也可能是故意保留的便利功能。例如，「忘記密碼」功能或許會顯示一則訊息，讓終端使用者知道他們輸入的電子郵件或帳號是不正確的，這個功能可讓駭客利用暴力猜解找出有效的帳號或電子郵件，該機構已決定接受此風險，並且不希望你再去測試這項問題。

密切注意契約裡的任何排除或限制條款，當進行 API 測試時，該專案可能允許測試指定的 API 之特定部分，但不同意測試該 API 的某些路徑。例如，銀行業務 API 的提供者可能與第三方共享資源，故無法授權進行這些測試，因此會在委託文件上載明可以攻擊 */api/accounts* 端點，但不能攻擊 */api/shared/accounts*；或者，待測目標是整合你無權攻擊的第三方身分驗證機制，你必須仔細審視工作範圍，以便執行合法授權的測試。

測試雲端 API 的注意事項

現今許多 Web App 是託管在雲端，攻擊這些 Web App，實際是攻擊雲端服務商（亞馬遜、谷歌或微軟）的實體伺服器，每家雲端服務商都有自己的一套滲透測試條款和服務，讀者必須清楚這些條款和服務內容。來到 2021 年，雲端服務商對滲透測試人員愈來愈友善，要求提交授權申請的項目比以前少得多了，儘管如此，某些託管於雲端的 Web App 和 API 還是需要事先取得滲透測試授權，例如機構使用的 Salesforce API。

在攻擊之前，務必清楚待測目標的雲端服務商之最新規定，以下是幾家較有名氣的雲端服務商之政策。

亞馬遜網路服務（AWS）：AWS 對於滲透測試的立場有很大改善，在撰寫本文時，除了 DNS 轄區遍歷、DoS 或 DDoS 攻擊（或模擬攻擊）、針對端口、協定和請求等泛洪攻擊外，AWS 允許其客戶執行各種安全測試，若打算測試前述排外項目，必須以電子郵件向 AWS 申請測試許可，在申請例外許可時，請確認包含測試日期、涉及的帳戶和資產、你的聯絡電話，並說明打算申請的攻擊細節。

谷歌雲端平台（GCP）：谷歌表示在滲透測試前，不需向它請求許可或發送通知，但要求你務必遵守其使用規章（AUP）和服務條款（TOS），不得逾越授權範圍。AUP 和 TOS 禁止非法行為、網路釣魚、垃圾郵件、散播惡意或具破壞性的檔案（如病毒、蠕蟲和木馬）及試圖中斷 GCP 服務。

微軟 Azure：微軟採取對駭客友善的方式，不需在測試前通知該公司，它在「滲透測試運作規則」網頁詳細介紹允許哪些滲透測試行為（詳見 *https://www.microsoft.com/zh-tw/msrc/pentest-rules-of-engagement*）。

到目前為止，雲端服務商對滲透測試活動採取較友善的立場，只要瞭解服務商最新聲明條款，測試有權攻擊的目標，並避免造成對方服務中斷，就應該可以合法地執行滲透測試。

測試 DoS 防禦能力

前面提到 DoS 攻擊通常是不被允許的，請與客戶商談，瞭解他們對特定項目的安全觀點。客戶若想要測試其基礎設施的性能和可靠度，應該將 DOS 測試視為一項可選服務，不然就問問客戶，看他們願意測試到什麼程度。

DoS 攻擊對 API 的安全性具有重大威脅，有意或無意的 DoS 攻擊將破壞目標機構的服務能力，使得 API 或 Web App 無法被存取，像這種非預期的業務中斷，通常促使機構祭出法律手段，因此要謹慎小心，只能對授權的項目執行測試！

最終，客戶是否願將 DoS 作為測試範圍的一部分，將視機構的安全觀點或為了達成特定目的而願意承受的風險而定，瞭解機構的安全觀點可以方便我們制定測試策略，機構擁有先進技術且對安全防護能力自視甚高，那麼對風險測試的程度也會較大，對於這種狀況，在訂定協議內容時，請涵蓋每項功能及執行可能的所有漏洞利用；對立的另一邊是極度趨避風險的機構，這些機構討論協議內容時就像行走蛋殼之上，協議的範圍裡會載明許多細項：明列出可攻擊的機器，在執行某些攻擊法之前必須先徵得同意等等。

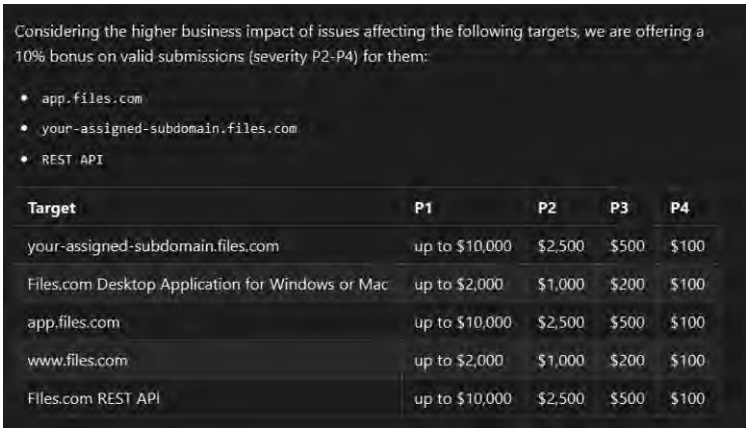
測試報告及複測服務

對客戶而言，你所提交的報告才是測試的真正價值，它會呈現測試 API 安全機制的成果，這份報告應詳細說明測試期間所發現的漏洞，並提供如何執行補救措施，以提高此 API 的安全性。

在商議專案範圍時，最後還要確認 API 提供者是否願意在修補後執行複測，客戶收到測試報告後應嘗試修復 API 漏洞。對於之前發現的漏洞進行複測，可驗證是否有效完成修補，複測作業可以只針對之前發現的弱點，也可以是全面重新測試，檢查套用至此 API 的變更是否引入新的弱點。

關於漏洞賞金

如果想提高駭侵攻擊的專業程度，最佳途徑是讓自己成為漏洞賞金獵人，BugCrowd 和 HackerOne 等機構提供媒合平台，任何人都可以輕鬆建立帳戶並開始狩獵。除此之外，許多機構也經營自己的漏洞賞金計畫，包括谷歌、微軟、蘋果電腦、推特和 GitHub，這些計畫都提供豐厚的 API 漏洞賞金，某些還有額外獎勵，例如，託管在 BugCrowd 的 Files.com 漏洞賞金計畫就專門設置 API 賞金，如圖 0-2 所示。



Considering the higher business impact of issues affecting the following targets, we are offering a 10% bonus on valid submissions (severity P2-P4) for them:

- app.files.com
- your-assigned-subdomain.files.com
- REST API

Target	P1	P2	P3	P4
your-assigned-subdomain.files.com	up to $10,000	$2,500	$500	$100
Files.com Desktop Application for Windows or Mac	up to $2,000	$1,000	$200	$100
app.files.com	up to $10,000	$2,500	$500	$100
www.files.com	up to $2,000	$1,000	$200	$100
FIles.com REST API	up to $10,000	$2,500	$500	$100

圖 0-2：BugCrowd 上的 Files.com 漏洞賞金計畫，是眾多 API 獎勵計畫之一

若對漏洞賞金計畫有興趣，該注意兩份規範：漏洞賞金提供者的服務條款和賞金計畫的範圍。違反任一項規範，不僅可能被漏洞賞金提供者所禁止，還可能招致法律制裁。賞金提供者的服務條款有一些重要資訊，包括如何賺取賞金、報告內容以及賞金提供者、測試人員、研究人員和駭客等與目標之間的關係。

有關賞金範圍的說明文件會提供目標 API、內容描述、獎勵額度、參與規則、報告內容的規格和其他限制條件，對於 API 漏洞賞金計畫，範圍內容通常含有 API 說明文件或該文件的鏈結。表 0-1 是參與漏洞賞金狩獵之前應瞭解的重要注意事項。

表 0-1：漏洞賞金的測試注意事項

測試目標	允許測試及可賺取賞金的 URL，特別注意臚列的子網域，因為有些可能不在測試範圍之內。
漏洞揭露條款	關於能否對外發表所發現的漏洞之規定。
除外項目	不在測試範圍，亦不提供賞金的 URL。
測試限制	規定機構會提供賞金的漏洞類型，通常會要求賞金獵人提供充分的證據，證明可從現實世界成功攻擊此漏洞。
法律規範	適用於機構、客戶和資料中心所在地理位置的法律和規定。

初接觸漏洞賞金的讀者可到 BugCrowd 學院看看，它有一份由 Sadako 提供，專門介紹 API 安全測試的影片和網頁（*https://www.bugcrowd.com/resources/webinars/api-security-testing-for-hackers*），也可閱讀 Vickie Li 撰寫的《Bug Bounty Bootcamp》（No Starch Press 於 2021 年出版），此乃幫助讀者踏上漏洞賞金之路的最佳資源，甚至有一章專門介紹 API 攻擊測試！

在花時間和精力處理各種漏洞之前，最好瞭解每種漏洞可能的報酬（如果有），像筆者就曾見過賞金獵人主張有效攻擊速率限制，但賞金提供者卻認定它是濫發請求。查看過去已公開的提交內容，看看該機構是否過於狡辯或不願為看似有效的提交支付賞金，此外，也要注意成功獲取賞金的提交，漏洞獵人提供哪種類型的證據，用什麼方式撰寫漏洞報告才容易讓金主接受此漏洞提交。

小結

本章檢視了 API 安全測試範圍的組成元素，建立 API 測試的協議範圍將有助瞭解擬採用的測試方法及參與規模，還可清楚哪些標的可以測試，哪些不能測試，以及執行時要使用的工具和技術，如果已經明確訂出測試方向，而你也能在這些規範下進行測試作業，那麼 API 安全測試協議便即將完成。

下一章將為讀者介紹 Web App 的功能，這是瞭解 Web API 工作原理的必要知識，若讀者已具備 Web App 基礎知識，可以跳到第 2 章，那一章將解析 API 所用的技術。