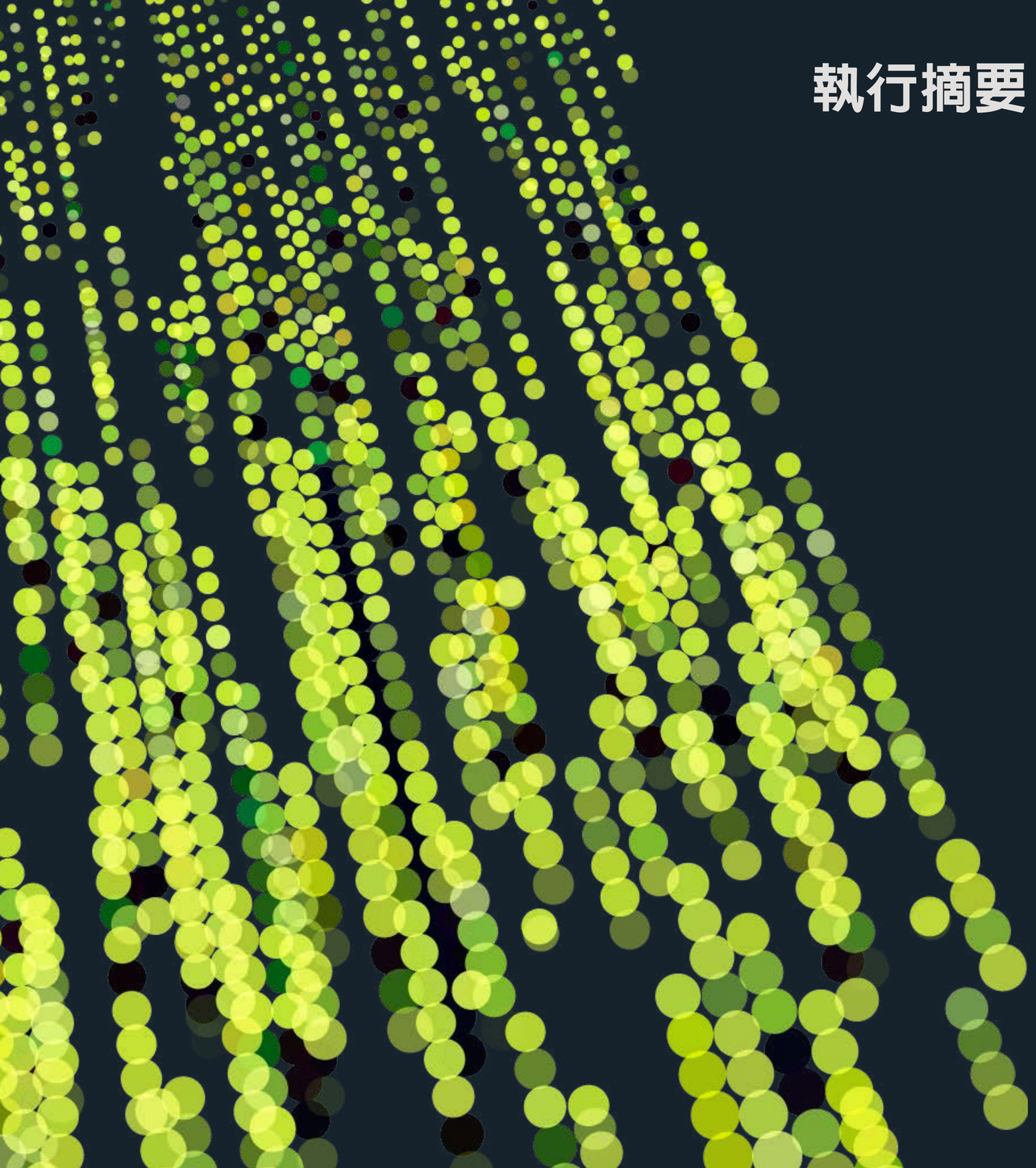
SYNOPSYS®

2024年開放原始碼安全與風險分析報告

協助您確保開放原始碼供應鏈的安全性

目錄



執行摘要

本報告為開放原始碼軟體的創作者與使用者提供建議，協助開放原始碼軟體的良好管理，重點則是專注在確保軟體供應鏈安全的領域。軟體的消費者或提供者都一樣是軟體供應鏈的一部分，都需要保護自己所使用的應用程式，避免來自上游與下游的風險。在接下來的幾頁裡，我們會探討以下主題：

- 根深蒂固的開放原始碼安全問題
- 為什麼開發者需要改善持續更新開放原始碼組件的能力
- 軟體供應鏈管理對於軟體物料清單(SBOM)的需求
- 如何因應AI編碼工具所引發的資訊安全與合規風險

將近十年以來，開放原始碼安全與風險分析報告(OSSRA)報告的主要課題一直都是：*你真的知道你的程式碼裡面裝了什麼東西嗎？*到了2024年，這個問題的重要性更是變得比以往的任何時候都更加重要。隨著開放原始碼的普及以及AI程式碼的崛起，現在已經有越來越多的應用程式都是利用第三方程式碼所構建的。

如果沒有對於程式碼真正的內容有完整的了解，不論是您本人、您的供應商或者是您的終端使用者都不可能對您的軟體可能包含的風險感到安心。確保軟體供應鏈安全的起點就是了解您的程式碼裡究竟包含了哪些開放原始碼組件，並且針對每一個開放原始碼組件去辨識它的授權、程式碼品質以及潛在漏洞。

關於2024年開放原始碼安全與風險分析報告(OSSRA)

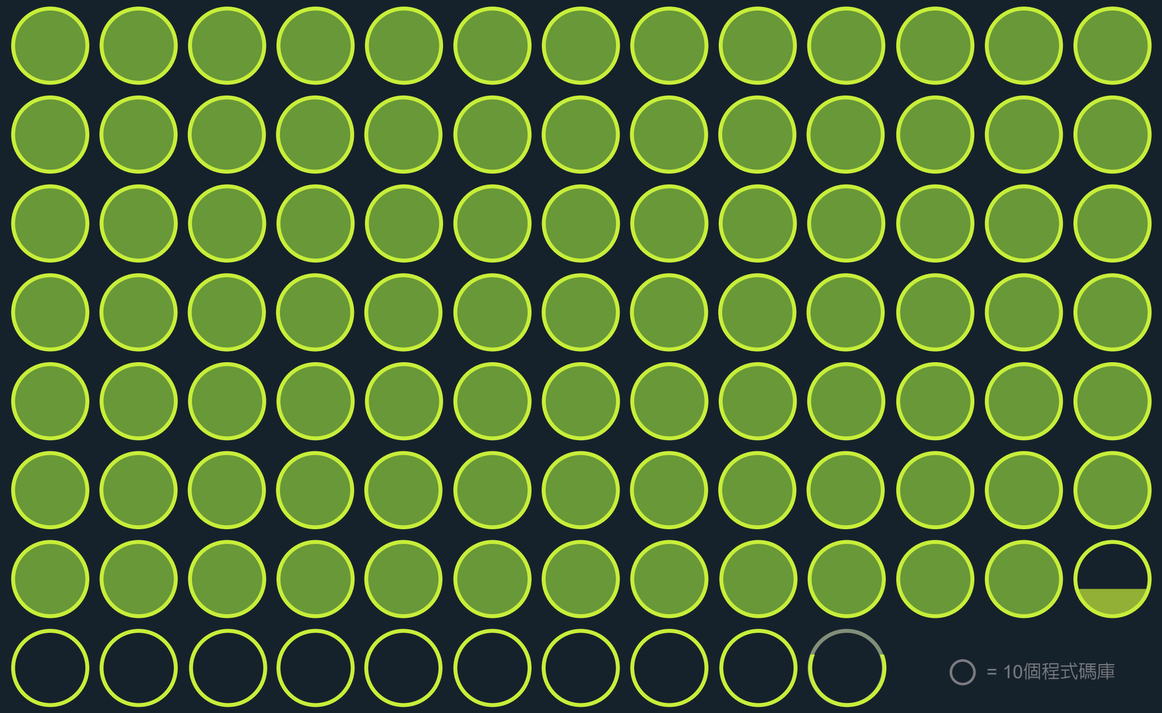
這一份2024年度第九版OSSRA報告針對市售軟體深入探討了開放原始碼的安全性、合規性、授權以及程式碼品質風險的現況。在這份報告裡指摘各種缺失的宗旨在於協助安全、法規、風險以及開發團隊對於開放原始碼安全與授權風險的概況有更好的理解。

這份報告採用來自Synopsys Black Duck®稽核服務團隊在2023年裡針對17個產業領域裡1,067個商業程式碼庫的缺失進行分析的匿名數據結果。這個稽核服務團隊已經累積超過20年的經驗，協助來自全球的安全、開發以及法規團隊加強其安全與授權合規規劃。該團隊每年為我們的客戶數以千計的程式碼庫進行稽核，其主要目的則是在併購(M&A)交易期間辨識來自於軟體的風險。

前述的稽核還提供了一份全面而準確的軟體物料清單，其內容涵蓋了在整個組織應用程式之中所包含的開放原始碼、第三方程式碼、網路服務以及應用程式介面(API)。稽核服務團隊利用來自於Black Duck KnowledgeBase™ 的數據做為基礎，識別潛在的授權合規與安全風險。Black Duck KnowledgeBase™的資料內容由Synopsys網路安全研究中心(CyRC)收集並加以管理，其中包含了來自於超過31,000個程式碼儲存庫中的780多萬個開放原始碼組件的資料。

這一份OSSRA報告強調了軟體之中採用的開放原始碼日益普及，以及如果未能適當進行相關管理所會帶來的潛在危險。開放原始碼是當前企業與消費者所仰賴的所有應用程式的基礎。想要讓軟體安全計劃成功，對於開放原始碼的有效辨識、追蹤以及管理就是不可或缺的——而這同時也是強化軟體供應鏈安全的關鍵要素。

摘要



在2023裡**掃描**的**1,067**個程式碼庫

有**936**個程式碼庫進行過**風險評估**

84%

接受風險評估的程式碼庫中含有**漏洞**

74%

接受風險評估的程式碼庫中含有**高風險漏洞**

在所有程式碼庫中，有

96%

含有開放原始碼

在所有程式碼庫中，有

53%

含有授權衝突

在所有程式碼庫的程式碼中，有

77%

源自於開放原始碼

在所有程式碼庫中，有

31%

含有**無授權**或**自訂授權**的開放原始碼。

10
年

2.8
年

24
個月

12
個月

14%接受風險評估的程式碼庫中含有**存在超過10年的**漏洞

在接受風險評估的程式碼庫中，**漏洞平均存在的年數**為**2.8**年

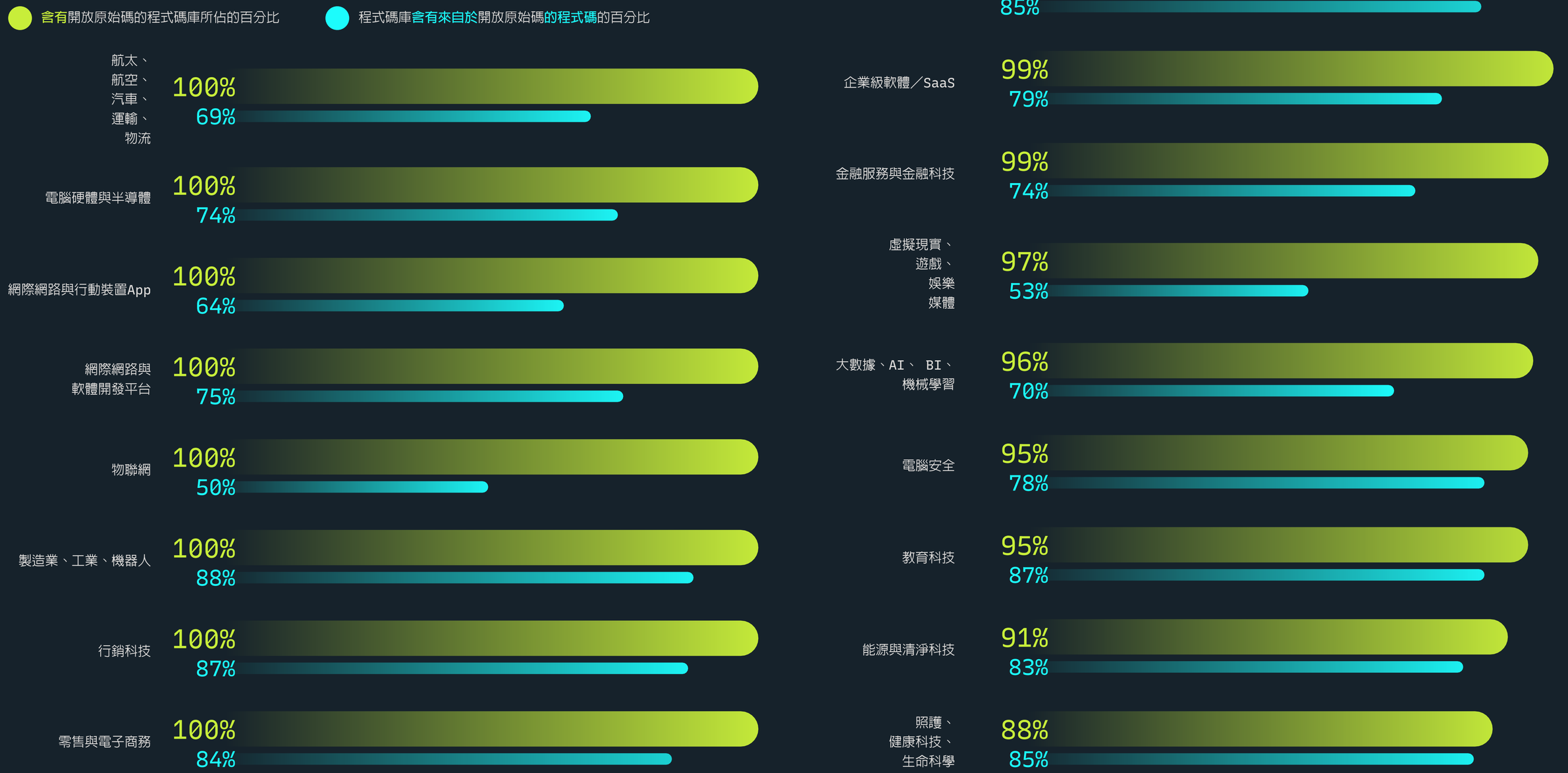
49%接受風險評估的程式碼庫中，含有在**過去24個月裡完全沒有開發活動**的組件

1%接受風險評估的程式碼庫中，含有在程式碼維護人員上一次**更新／修補**之後，**更新／修補**已經至少落後**12個月**的組件

91%

接受風險評估的程式碼庫中，含有**版本至少已經比現行版落後10個版次**的組件

圖1：經過業界掃描的1,067個程式碼庫



開放原始碼的漏洞與安全

稽核相關說明

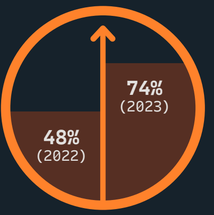
所有Black Duck稽核都會檢查開放原始碼授權的合規性。客戶可以根據自己的判斷選擇是否接受有關漏洞／營運風險評估的稽核。2023年，Black Duck稽核服務團隊進行了1,067次稽核，在其中有88% (936次)的稽核包含了漏洞／營運風險評估。在《2024年開放原始碼安全與風險分析》(OSSRA)報告中「開放原始碼的漏洞與安全」以及「影響開放原始碼風險的操作性因素」這兩個章節的資料是以接受了風險評估的936個程式碼庫為基礎。而「開放原始碼授權」章節資料則來自於所有接受稽核的1,067個程式碼庫。

在Black Duck稽核服務團隊所分析的1,067個程式碼庫中，有96%的程式碼庫包含開放原始碼。在所有經過掃描的原始碼與檔案中，有77%是來自於開放原始碼。

在這一年度的資料中，每個應用程式裡平均含有526個開放原始碼組件——由此可以看出自動化安全測試在實務上十分重要，甚至可以說是絕對必要。在組件數量較少的時候，手動測試或許是可行的，但在組件數量龐大的時候，手動測試則幾乎是不可能實現的，這時候就需要使用像是軟體成分分析(SCA)這樣的自動化解決方案。自動化安全測試與手動測試有所不同，它的執行迅速而一致，讓開發人員得以在開發過程的前期就發現問題，從而避免對交付產品的時程或生產力造成影響。

在接受了風險評估的程式碼庫中，有84%包含至少一個已知的開放原始碼漏洞。其中有74%的程式碼庫含有高風險漏洞，相較於2022年，這個比例有了大幅的增加，2022年的時候則只有發現48%的程式碼庫含有高風險漏洞。所謂的高風險漏洞，指稱的就是那些已遭受積極利用、在漏洞利用上已經有了概念驗證的記錄，或被歸類為遠端程式碼執行漏洞的漏洞。

84% 程式碼庫含有至少**一個開放原始碼漏洞**



過去一年裡，含有高風險漏洞的程式碼庫**成長了54%**

在2022年至2023年之間，高風險漏洞的成長率為54% (26個百分點)，這個現象並沒有一個確定的原因可以解釋。由於經濟衰退造成裁員，因此使得檢測並修補漏洞的資源減少會是一個有可能的解釋。除此以外，我們發現有91%的程式碼庫之中都有一部份的組件版本比最新可用的版本落後10個版本以上。所以我們得到一個簡單的結論——絕大多數開放原始碼的使用者根本沒有更新他們使用的組件。

有49%的程式碼庫含有在過去24個月裡沒有開發活動的組件，有1%的程式碼庫含有在程式碼維護人員上一次更新／修補之後，更新／修補已經至少落後12個月的組件。

「維護人員」一詞泛指那些領導開放原始碼專案，貢獻自己心力的人士，維護人員往往是可以作出最終決斷，決定要建立哪些程式碼、或是要釋出哪些程式碼的決策者，他們可能會自己獨力審查所有的程式碼，並且以自己本身的名義管理一些小專案，而且他們可能會針對整個專案的方向進行最終決策。他們的日常工作可能各有不同，但其中可能會包含：審查所有合併請求(pull request)以及其他人貢獻的開發成果、釋出新版軟體、分類並處理安全性修復(security fix)以及社群管理與調整。

大多數維護者都很勤於工作，努力維護他們所參與的開放原始碼專案，讓專案能夠跟得上時代。事實上，許多公司會特別雇用人力來維護該公司的軟體所仰賴的開放原始碼專案。開放原始碼的使用者也應該要這麼勤於工作。開放原始碼軟體的使用者需要注意他們所使用的版本，建立定期更新的週期，並且在開放原始碼軟體的使用上落實方面落實軟體衛生的概念——只從維護人員與貢獻開發成果的人員生態環境都足夠良好的專案之中取用資源。

常見弱點列舉(CWE)與常見漏洞暴露(CVE)都是很常用的列表工具，可以用來辨識並分類軟體的安全弱點以及漏洞。Black Duck安全建議(BDSA)是專屬於Synopsys的報告，其目的在於為客戶提供比國家漏洞資料庫(NVD)的CVE通知更及時、更詳盡的資訊。BDSA會提供可行的建議以及會對客戶的軟體物料清單(SBOM)之中的項目造成影響的漏洞相關詳盡資訊，藉此協助確保客戶他們能完全瞭解開放原始碼漏洞可能帶來的風險。

在十大漏洞之中，有八個都可以追溯到一個關鍵的CWE，也就是CWE-707，此一漏洞所牽涉的是未能滿足安全要求，而且可能導致跨網站指令碼攻擊以及SQL注入等漏洞的問題。

如圖2所示， CWE-707 是CWE 20、79、80、97以及937的支柱。CWE-707所關注的是未能在從上游組件讀取數據或將數據傳輸到下游組件之前滿足安全要求的問題。未能適切地將來自輸入端的風險加以中和(neutralize)，可能會導致像是跨網站指令碼攻擊(XSS)或SQL注入的漏洞。XSS是一種常見且危險的攻擊手段，因此本報告中特意加以強調的十大漏洞大多數都與它有關。

所謂的跨網站指令碼攻擊，就是攻擊者利用網站的缺陷，送出格式錯誤的惡意程式碼(通常是JavaScript)來進行攻擊。由於來自外界的指令碼沒有經過適當的中和(neutralize)或跳脫(escape)處理，因此攻擊者就可以利用這種漏洞來操控原本在安全上值得信賴的主機來執行惡意的任務。不過，大多數XSS攻擊的最終目標並不是主機本身，而是使用網路應用程式的其他使用者。一旦網站裡被加入了外來的惡意腳本，攻擊者就可以利用惡意腳本來竊取機敏性資訊，比如說工作階段cookie (session cookie)。

XSS不只是名列於我們的前十大漏洞列表之中而已，它同時也在OWASP前十大漏洞列表之中佔有一席之地——在這份報告裡列出了十大最為關鍵的網路應用程式安全風險。在這份列表裡，OWASP將XSS稱作A03:2021 – 注入。XSS漏洞之所以普遍存在，可以說是因為各種組織越來越依賴網路應用程式，用來與客戶互動，也用來讓用戶之間彼此互動。比如說，您可以想想看有多少電子商務公司、銀行、網路服務供應商、保險公司等等是如何利用網路體驗來與客戶以及合作夥伴互動。

再次強調，我們的數據清楚地顯示，開發團隊需要改善持續更新開放原始碼組件的能力，尤其是在使用到像jQuery這樣的流行開放原始碼組件時，使用較舊、較容易遭受攻擊的版本的開放原始碼，後果可能會很嚴重。比如說，我們在稽核裡發現的前十大漏洞之中，排名第二的是BDSA-2020-0686 (CVE-2020-11022)，一個存在於jQuery 1.2版到3.5.0版以前的XSS漏洞。這個漏洞會允許來自於不可信任來源的HTML程式碼傳遞給jQuery的某一個DOM操控方法，結果可能會導致不可信任的程式碼的執行。

這個問題在jQuery 3.5.0版中得到了修復，但正如我們的數據所顯示的，在我們針對安全風險進行掃描的時候，發現有三分之一的程式碼庫仍然在使用容易受到此漏洞危害的版本，這使得攻擊者可以利用惡意數據來滲透一個系統，或者是用來盜取機敏性資料，這可能會導致密碼或信用卡資訊外洩。如同先前所提到的，跨網站指令碼攻擊是應用程式中最常見的弱點之一，攻擊者通常會利用注入攻擊的方式針對瀏覽器裡的各種解譯器(interpreter)，比如HTML或JavaScript的解譯器。

緩解風險：安全地使用jQuery以及其他熱門開放原始碼資源的訣竅

我們在稽核之中發現到的前十大開放原始碼組件全部都是用JavaScript編寫的。我們在稽核之中發現的漏洞也大部分都與JavaScript函式庫相關，其中尤以過舊的jQuery JavaScript函式庫之中的漏洞為甚。

- 使用最新版本的jQuery——如本報告中所述，舊版的jQuery往往會含有安全漏洞。
- 考慮訂閱安全諮詢服務，例如Black Duck安全顧問服務，藉此獲取最新的漏洞相關資訊。在jQuery之中時常會發現新的安全漏洞。
- 使用安全編碼架構來幫助您辨識並避免程式碼中潛在的安全漏洞。
- 使用自動化安全測試——包括靜態分析、軟體成分分析以及動態分析工具，藉此解決整個軟體開發生命周期中的程式碼品質瑕疵以及安全缺陷。

jQuery並不是從本質上就不安全，事實上，它是一個維護良好的開放原始碼函式庫，擁有大量的使用者、開發人員以及維護人員社群。然而，樹大招風就是他的問題。依據我們的稽核結果，jQuery是最有可能存在漏洞的組件——雖然本報告裡所列出的每一個jQuery漏洞都有可用的修補程式。對於jQuery的使用者——事實上，應該說是對於所有開放原始碼的使用者來說，對於舊版軟體的潛在安全風險有所認知，並且採取措施、降低風險是很重要的。

採取行動，避免漏洞進入您的軟體供應鏈

- 建立並維護一份軟體物料清單(SBOM)。在對抗軟體供應鏈攻擊的戰爭裡，對於評估暴露狀況，並且確保您的程式碼維持高品質、合規與安全來說，擁有一份準確的最新SBOM，列出開放原始碼組件清單是非常關鍵的。一份全面的SBOM會列出您的應用程式之中的所有開放原始碼組件，以及這些組件的授權、版本以及修補狀態——這就是對抗供應鏈攻擊的強力防護，當然也能抵禦那些使用惡意套件的攻擊。
- 隨時確保吸收新知。確保您有方法可以獲得有關新發現的惡意套件、惡意軟體以及最新公開的開放原始碼漏洞。尋找新聞來源或定期發布的諮詢資訊，針對那些會影響您的SBOM所包含的開放原始碼組件的問題，尋求可行的建議與詳細資訊。
- 進行程式碼審查。在將下載的軟體納入到您的專案以前，檢查該軟體的程式碼。檢查是否有任何已知的漏洞。為了獲得更深入的瞭解，考量是否也一併執行原始碼的靜態分析，藉此檢查未知的安全弱點。
- 保持主動積極的態度。某個組件目前沒有漏洞，並不表示它明天也仍然會是安全的。如果是有意設計的惡意套件，甚至有可能永遠不會有人發現它是一個「漏洞」。在執行任何組件的程式碼以前，多關注組件的健康狀況以及來源，藉此避免日後的安全問題。
- 使用自動化軟體成分分析(SCA)工具。SCA工具會自動化地進行辨識、管理並且緩解軟體安全問題的流程，讓開發者可以專注於程式碼的編寫。這一類的工具可以評估開放原始碼以及第三方程式碼。

前10大漏洞裡有8個可以追溯到同一個CWE

所謂的「支柱(Pillar)弱點」就是由CWE專案定義的最高等級弱點，是所有相關類別(class)以及變體(variant)的基礎。

圖2：Top 10 CVE/BDSA

BDSA-2021-3651
受影響組件： jQuery
相關CWE： CWE-546

BDSA-2014-0063
受影響組件： jQuery
相關CWE： CWE-79
支柱弱點： CWE-707

BDSA-2020-0686 (CVE-2020-11022)
受影響組件： jQuery
相關CWE： CWE-707

BDSA-2015-0567
受影響組件： jQuery
相關CWE： CWE-937
支柱弱點： CWE-707

BDSA-2020-0964 (CVE-2020-11023)
受影響組件： jQuery
相關CWE： CWE-80
支柱弱點： CWE-707

BDSA-2019-0423 (CVE-2019-8331)
受影響組件： Bootstrap
相關CWE： CWE-97
支柱弱點： CWE-707

BDSA-2019-1138 (CVE-2019-11358)
受影響組件： jQuery
相關CWE： CWE-20
支柱弱點： CWE-707

BDSA-2020-1173 (CVE-2020-7656)
受影響組件： jQuery
相關CWE： CWE-80
支柱弱點： CWE-707

BDSA-2017-2930 (CVE-2015-9251)
受影響組件： jQuery
相關CWE： CWE-79
支柱弱點： CWE-707

BDSA-2022-1987 (CVE-2022-31160)
受影響組件： jQuery UI
相關CWE： CWE-79
支柱弱點： CWE-707

為什麼有些BDSA沒有CVE

公開來源的資訊，像是美國國家漏洞資料庫(NVD)，會是一個了解開放原始碼軟體的公開漏洞資訊的良好起點。然而。任何NVD CVE條目的報告都可能會有延遲。即時性一直是影響NVD安全漏洞資料公開能力的要素。事實上，從漏洞首次被揭露到NVD發布該漏洞資訊之間，經常會有顯著的時間延遲，有部分研究報告指出，從最初的公告到NVD發布之間平均需要一個月的時間。

NVD的另外一個問題是它所提供的漏洞資料經常不太完整。NVD之中所發布的許多CVE並不包括容易遭受攻擊的版本範圍，而且資訊往往太少，無法提供有用資訊。這通常是因為發行方沒有資源來研究該CVE條目。

顯然，想要僅僅依賴NVD來獲得漏洞資訊是不明智的選擇。許多市售軟體成分分析(SCA)解決方案可以提供比NVD更豐富的漏洞資料，除此以外，在有需要的時候，這些市售SCA還能更準確地找到問題，並且更能迅速地協助您修復問題。比如說，如果你有在使用Synopsys的SCA解決方案——Black Duck，您就可以利用BDSA。BDSA是由Synopsys安全研究團隊所發現的開放原始碼漏洞建議。BDSA的建議往往可以為您提供更為迅速的漏洞通知——可能會是在NVD的數天或數週以前。BDSA還能提供更為完整的漏洞資訊，提供超越當前市面上可用的任何其他資訊的安全洞見、技術細節以及升級／修補指引。比如說，在我們2023年的前十大漏洞清單之中，就有三個BDSA條目，在NVD之中沒有與其相關的CVE。

BDSA-2021-3651

依據BDSA的內容，有某些版本的jQuery包含針對遭受劫持的網域的註釋引用。這算是一種安全問題，最安全的做法是移除指向遭受劫持的網域的連結，而這正是jQuery在3.6.1版本中所做的，因為如果使用者在不知道域名狀態下試圖訪問這些連結，，他們仍然有可能會暴露在未知的攻擊之下。雖然通過運行程式碼無法到達這些站點，但提出此一問題仍有其價值，因為開發人員或任何有存取該頁面權限的人都有可能會不小心點擊到惡意網站。

Synopsys CyRC團隊建議，這個BDSA的標籤是「具參考價值(informational)」，在供應商提供的「修復」措施以警告的形式出現在程式碼裡，或者是出現在產品的說明裡，又或是在供應商拒絕接受CVE或對缺失發現提出爭議，並認為被通報的漏洞是其組件中事先預期／有意設計的行為時，就會使用這個標籤。

類別

CWE-546：可疑註釋

CVSS v3.1分數

5.10

BDSA-2014-0063

這是一個比較舊的漏洞，首次通報時間是2014年1月，其內容涉及jQuery可能存在的跨網站指令碼攻擊(XSS)漏洞，其原因則是缺乏對於使用者輸入資訊進行的驗證。這可能會容許攻擊者注入任意的網頁指令碼，並且竊取受害者的工作階段cookie (session cookie)。

依據BDSA的內容，會有一個函數解析HTML字串，形成一個DOM節點的字元陣列。任何利用這個函數傳遞，包含在事件屬性之中的指令都會立刻執行。如果呼叫端在將不受信任的輸入資訊傳遞給該函數以前未能適當地進行淨化，就有可能會造成呼叫端產生遭受XSS攻擊的風險。攻擊者可以利用這個漏洞，製作惡意的HTML給受害者使用。如果受害者處理了這些HTML，則其中所包含的所有網路指令都會在其系統上執行。

這個漏洞在3.0.0-rc1之中得到了緩解。然而，這種緩解措施並不會針對惡意輸入資訊進行淨化，仍然會允許指令碼執行。解析器的預設行為有所改變，如果未指定執行環境或指定為null／undefined，則會建立一個新的文件檔。這會延遲先前解析的HTML的執行，直到該HTML被注入到文件檔之中，這讓其他工具有機會在呼叫函數之後遍歷先前創建的DOM，並移除不安全的內容。

類別

CWE: 79：在網頁生成過程中對於輸入資訊的不當中和(「跨網站指令碼」)

CAPEC-588：以DOM為基礎的XXS

CVSS v3.1分數

8.60

BDSA-2015-0567

這是另外一個比較舊的漏洞，這個漏洞與jQuery相關，使用未修補版本的UglifyJS解析器的jQuery版本容易會因為精心設計的JavaScript檔案而執行任意程式碼，而這最終可能會讓攻擊者執行惡意程式碼。

本漏洞已在1.12.0與2.2.0之中修復。

類別

CWE Category A9：利用已知漏洞的組件

CAPEC-251：本地程式碼包含漏洞(常見攻擊列舉與分類[CAPEC]的成果有提供其公開可用的攻擊模式目錄及其完整架構與分類學資訊)

CVSS v3.1分數

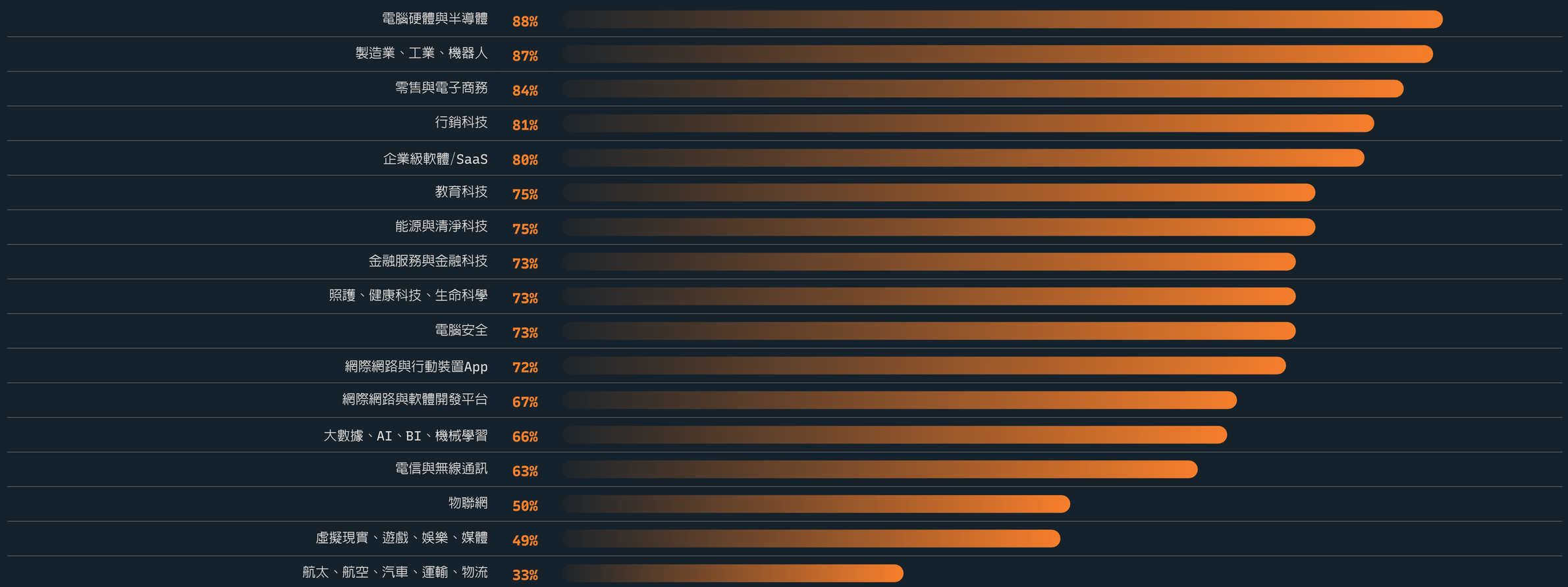
7.9

各大產業的漏洞

在電腦硬體與半導體領域，有88%的程式碼庫裡包含了高風險類別的漏洞(嚴重度評分7分以上)，接下來則是製造業、工業、機器人、以及零售與電子商務，其占比分別為87%與84%。

在各大領域裡都有類似的發現，即使是百分比最低的航太、航空、汽車、運輸、物流領域的狀況也令人擔憂，此一業務領域中有三分之一的程式碼庫包含高風險漏洞。如圖1所示，我們所檢查到的每一個業界程式碼庫裡都有開放原始碼，它們建構了各種不同業界的程式碼庫。如圖3所示，這些程式碼庫裡還含有大量已知的開放原始碼漏洞，而擁有程式碼庫的組織未能進行修補，從而造成了易於遭受攻擊的漏洞。

圖3：含有高風險漏洞的程式碼庫的百分比

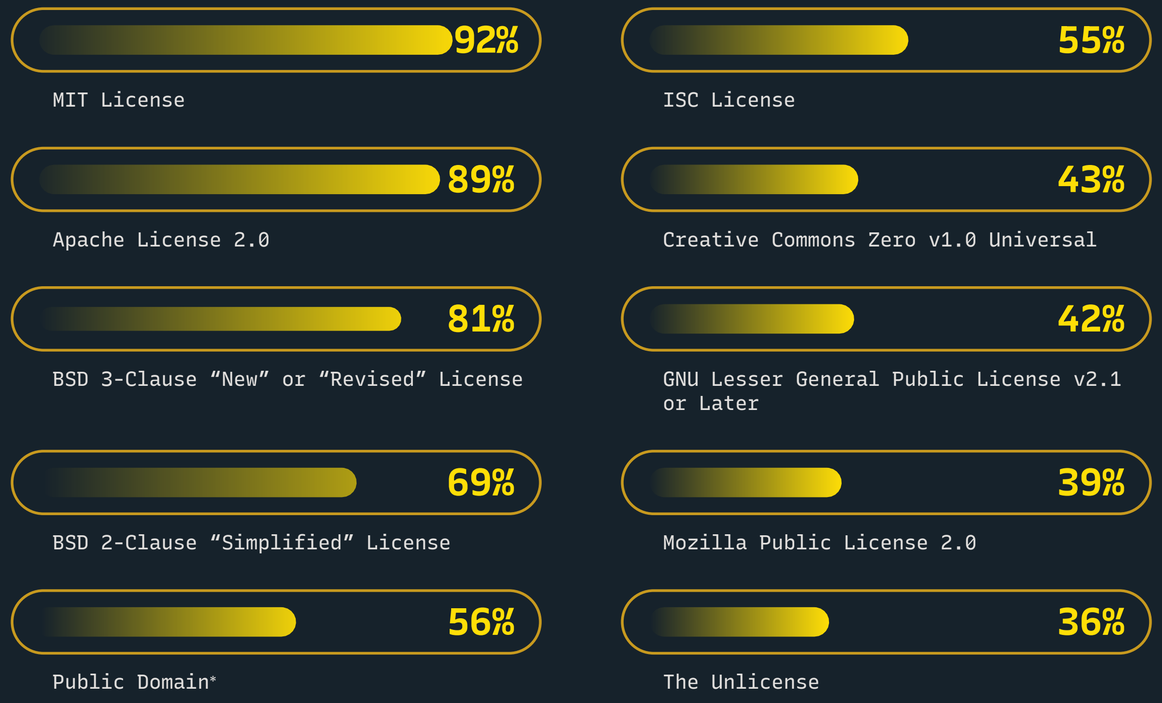


開放原始碼的授權

有效的軟體供應鏈管理需要在授權以及安全方面合規。您正在使用開放原始碼組件以及函式庫來構建軟體，同時也了解這些組件受到開放原始碼授權的管轄，但您了解這些授權的細節嗎？在您的軟體中，即使有任何一個不合規的授權，也可能會導致法律問題、損失可以獲利的智慧財產權、必須為了補救而曠日廢時，並且延遲您的產品上市的時機。

Black Duck稽核服務團隊發現，在2023年接受稽核的程式碼庫之中，有超過一半(53%)裡包含了具有授權衝突的開放原始碼。

圖4：在程式碼庫裡發現的前十大授權百分比



*Component includes a statement that it is in the public domain but does not use a specific public domain license such as the Unlicense or Creative Commons

在2023年接受Black Duck Audit Services稽核的開放原始碼軟體之中，發現有92%含有MIT授權。MIT是一種允許在專有軟體之中重複使用的寬鬆授權，與其他軟體授權之間具有較高的相容性，風險也較低。如果您的軟體裡含有第三方組件，那麼可以肯定您可能也會發現流行的寬鬆授權，例如MIT、Apache、BSD、ISC以及Unlicense。

應該注意的是，所謂「低風險」的說法只是一種指導性的參考，在針對是否使用開放原始碼軟體進行決策的時候，不應該仰賴這種所謂「低風險」的說法。比如說，雖然一般認為授權風險較低的Apache 2軟體可以被納入在GNU通用公衆授權條款3.0 (GPLv3)的專案中，但GPLv3軟體則不能被被納入在Apache專案裡。由於Apache軟體基金會的授權觀念與GPLv3作者對著作權法律的解釋，在後者的狀況下，這兩種授權是不相容的。最安全的策略就是開

發人員應該諮詢其公司的政策與法律團隊，獲得有關授權合規的具體指導。

我們在2023年的稽核裡發現，Creative Commons授權是最常見的授權衝突的原因。光是Creative Commons ShareAlike 3.0 (CC-SA 3.0)這一種授權就在已知授權衝突之中占了17%的成因。

在Black Duck稽核之中，常常會發現「程式碼片段」(時常在原始碼中複製貼上的程式碼)，它們往往來自於熱門博客網站Stack Overflow，該網站會自動針對所有公開可存取的使用者貢獻進行Creative Commons ShareAlike (CC-SA)授權。不幸的是，這種統一授權也適用於在該網站上發布的程式碼片段。我們之所以會說「不幸」，是因為這種授權不是為軟體而設計的，Creative Commons在其常見問題解答中明確表示：「我們不建議對軟體使用Creative Commons授權」。在某些狀況下，我們可以認為CC-SA授權具有類似於GNU公衆授權的「病毒式」效果(也就是從該著作權授權作品所衍生的任何作品也必須在相同的著作權條款下對外授權)，從法律角度來看，這可能會引發問題。

了解授權風險

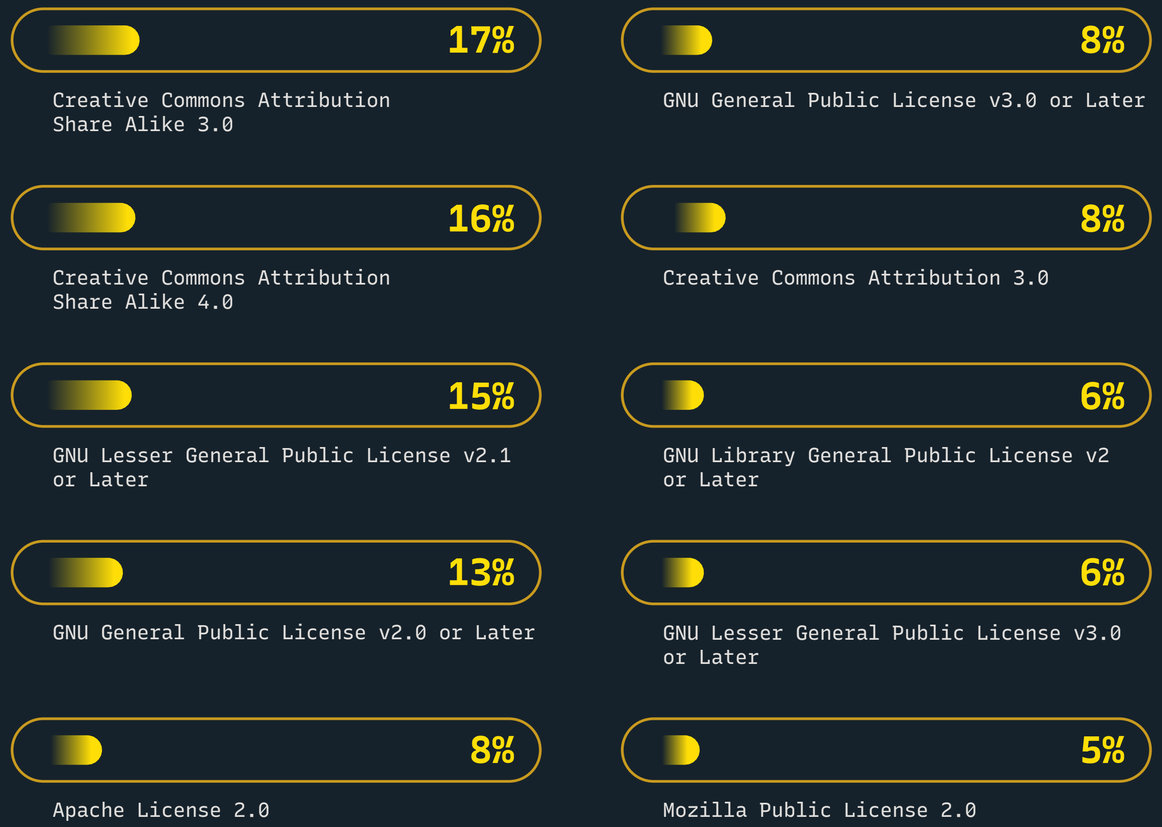
在美國以及其他許多不同的當地法律管轄之下都預設創意作品(包括軟體)受到專有版權的保護。未經創作者／作者以明示的方式給予許可，任何人都不能合法地使用、複製、發布或修改其軟體。

即使是最友善的開放原始碼授權也會包含使用者使用該軟體時所須承擔的義務。當程式碼庫含有開放原始碼，且其授權可能與程式碼庫整體的授權衝突的時候，就會出現潛在的授權風險。GNU通用公衆授權(GPL)是開放原始碼專案中的最常見的著作傳授權。如果依據GPL授權的程式碼出現在商用封閉軟體裡的時候，就可能會出現衝突。

標準開放原始碼授權的變體或其自訂版本可能會對獲得授權的人提出不良要求，並需要針對可能發生的智慧財產權問題或其他影響進行法律評估。JSON授權是自訂授權的主要範例。JSON授權以寬鬆的MIT授權在為基礎，並且增加了「該軟體只能用於善良意圖，非邪惡意圖」的限制。此一聲明的曖昧性造成其含義有待商榷，許多律師都建議不要使用以這種方式進行授權的軟體，尤其是在進行併購的時候。

在2023年接受稽核的程式碼庫之中，有31%的程式碼庫有在使用沒有可以識別的授權的程式碼，或者是自訂授權的程式碼，這與去年的結果幾乎相同。在針對開放原始碼的稽核之中，在網站裡找到沒有明確服務條款或沒有提及軟體條款的程式碼片段的情況並不少見——Stack Overflow則是一個例外。造成開放來源程式碼沒有授權條款的另一個常見原因是開發人員使用了程式碼片段，但卻沒有加上該程式碼片段的相關授權。討論到沒有相關授權的程式碼這個主題，有另外一個問題是人們越來越仰賴人工智慧輔助編碼工具(請參閱下一節)

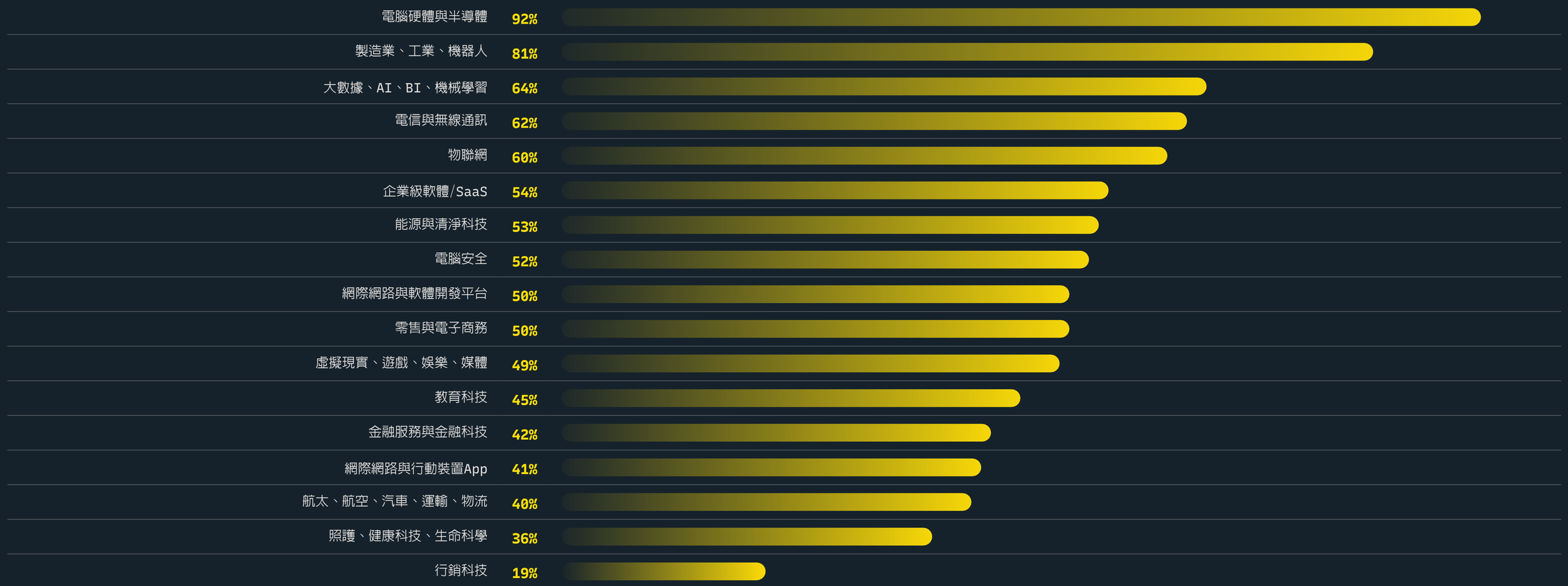
圖5：前十大授權衝突百分比



可能導致某些產業領域有如此高比例的開放原始碼授權風險(見圖6)的原因包括：

- 許多授權衝突較多的產業傾向於將其軟體以內部產品的形式進行授權與發佈。許多限制性授權對於利用這種方式發布的軟體有專門的規範。其他授權衝突較少的產業則可能在軟體部署上較為偏向採用訂閱或SaaS型部署，而這樣的部署通常不被視作「發布」，且不受那些限制性授權條款的約束。
- 半導體與硬體公司極度仰賴軟體與韌體，在這些軟體與韌體中有許多都包含 開放原始碼(見圖1)。複雜的系統單晶片設計可能包含來自多種來源的程式碼，其規模則可能達到數百萬行。在這種規模上追蹤授權與義務可能會相當具有挑戰性。
- 開放原始碼在底層系統軟體、韌體、驅動程式等方面的應用極為普遍，而它們都是硬體產品不可或缺的一部分。其中有許多授權是GPL型「著作傳(copyleft)」授權，在受到發布之後會產生強制共享的要求。
- 在公司、硬體設計師以及製造商之間共享韌體、驅動程式與工具的軟體供應鏈裡造成開放原始碼在沒有利用軟體物料清單(SBOM)來追蹤其來源或授權的狀況之下擴散。

圖6：含有認證衝突的程式碼庫的百分比



保護自身免於AI編碼工具所引進的安全與智慧財產權合規風險

隨著AI輔助編碼建議工具的崛起，有關AI產生的程式碼的所有權、版權以及授權的問題也應運而生。比如說，針對GitHub、微軟以及OpenAI提起的集體訴訟指控，GitHub Copilot——一種雲端AI工具，在開發人員編寫程式碼的時候提供自動完成編碼的建議——這違反了著作權法律以及軟體授權要求。該訴訟進一步聲稱，Copilot所建議的程式碼使用需要授權的資源，但沒有標示版權歸屬、版權聲明，亦未遵守原始授權條款。

Copilot訴訟案凸顯了使用AI產生程式碼所涉及的複雜法律問題。對於軟體開發人員來說，在這個問題透過法律或政府的決議加以解決以前，避免使用AI輔助編碼工具顯然是避免授權或版權違規作為的最為安全的做法。

有意使用AI輔助工具的開發人員應該審慎行事，避免不必要的風險。至少他們應該要求其所屬組織詢問其AI工具供應商，在該AI工具所提供的建議中是否包含開放原始碼授權的原始碼，如果是的話，AI是否可以標示這些程式碼，或是否可以將其完全從建議之中排除。

另一個解決方案則是使用可用的程式碼掃描器，例如Synopsys Black Duck。Synopsys Black Duck使用程式碼片段分析的方式來掃描原始碼，並且將每一行程式碼與其任何可能的開放原始碼專案作比對，這讓開發團隊可以辨識由AI程式碼輔助工具所引進的開放原始碼所使用的授權方式及其相關條款。

在軟體供應鏈管理中的最佳開放原始碼授權管理實務

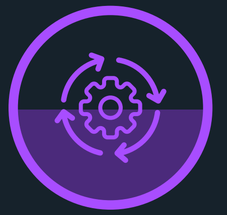
- 對軟體中的所有第三方軟體組件進行徹底的清查，包括開放原始碼軟體以及市售軟體。
- 注意AI輔助編碼工具可能會產生具有潛在授權違規以及侵犯智慧財產權的程式碼。
- 評估每一個組件的授權使用條款，並評估它們是否與產品的預期用途相容。
- 檢查不同組件的授權之間的相容性，因為有些授權可能彼此不相容。
- 使用自動掃描工具來辨識並追蹤每一個組件的授權義務與限制。
- 實施確保維持授權合規性的程序，包括定期實施授權掃描以及定期審查授權合規程序。
- 針對新接觸或不熟悉的授權建立審查流程與作業流程。
- 確保法規、技術以及業務利益關係人之間的有效溝通，藉此妥善優先處理並執行授權清算(license clearance ，也就是公司決定某一特定組件的授權是否適用於其產品的流程)。
- 記錄所有授權清算活動，包括授權評估與合規程序，確保合規相關工作都有記錄，協助未來稽核的進行。

影響開放原始碼風險的操作性因素

在理想狀況下，開放原始碼軟體的使用者只會使用那些在強大社群的支援之下開發的組件。例如Linux，它每天都在來自數百個組織、數以千計的開發人員的努力之下有所改善。然而，在接受Black Duck 稽核服務團隊檢查，並且也接受風險評估的936個程式碼庫之中，一共有49%包含了在過去兩年內沒有進一步開發活動的開放原始碼軟體。如果一個專案不再有人維護——尤其是在小型專案，那就意味著沒有功能升級、沒有程式碼改善，也不會修復已發現的安全問題。

對於開放原始碼專案來說，這並不是一個罕見的問題。有部份報告顯示，將近20%的Java與JavaScript開放原始碼專案在2022年還有人員維護，但到了2023年就不再有人員維護了，這使得這些專案面臨著來自漏洞以及遭受利用的風險。開放原始碼軟體大多是由有志於作出貢獻與維護的志願者貢獻所產出的。雖然有些組織，例如微軟、RedHat以及Google都設立了激勵性的計畫來鼓勵開放原始碼專案的維護以及參與，但絕大多數公司並沒有這麼做。當維護人員停止了專案的維護，其中一個後果就會是安全風險的增加。

在2023年分析的程式碼庫裡有**88%**接受了風險評估



接受風險評估的程式碼庫裡有**49%**含有已經2年沒有進行過開發活動的開放原始碼

開放原始碼 使用者需要加強維護實踐

在2023年接受Black Duck稽核服務團隊檢驗，並且也接受風險評估的936個程式碼庫中，有91%包含了版本至少落後最新版本10個版次的組件。

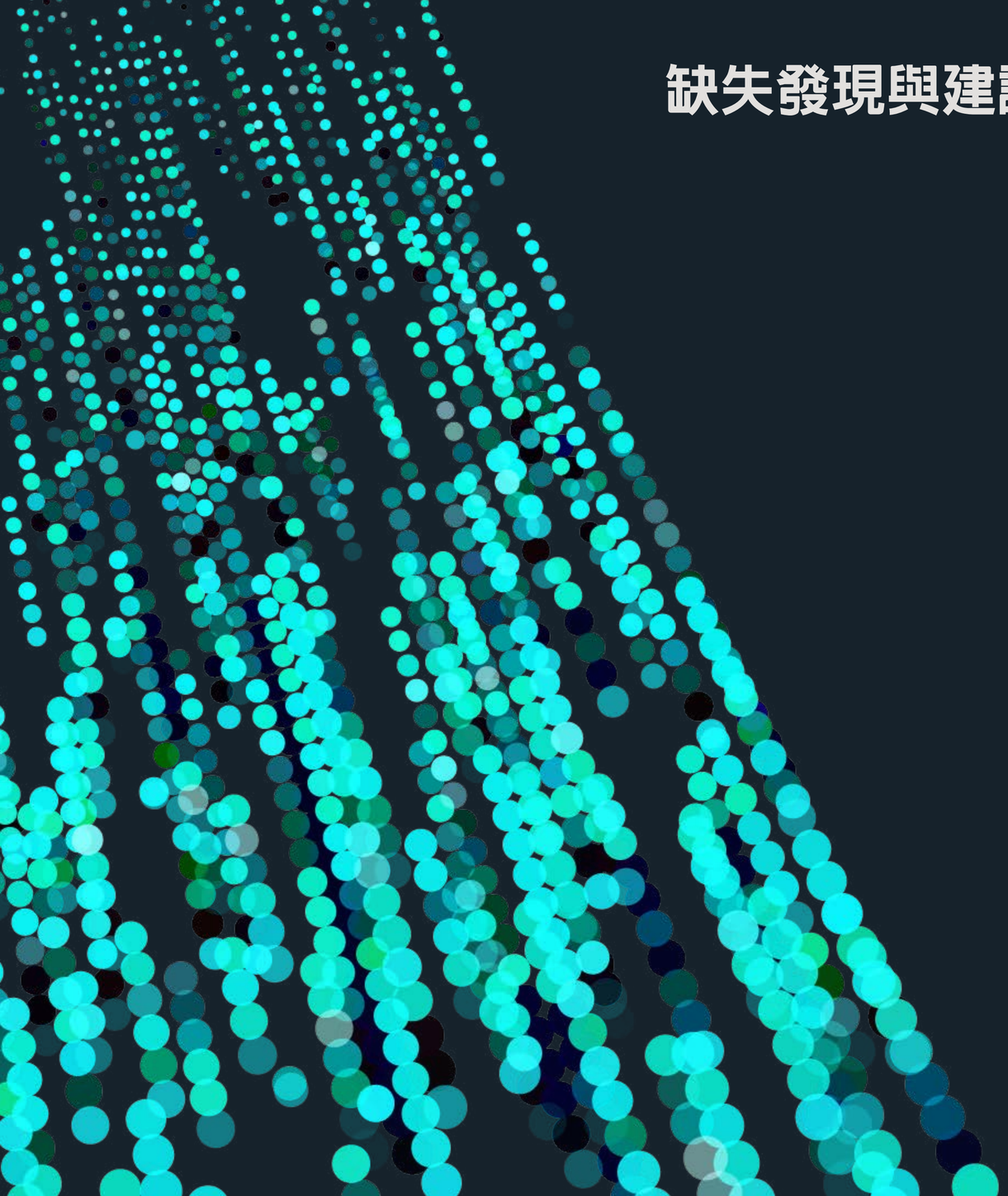
對於開放原始碼軟體的使用者來說，可能有一些理由促使他們不去更新開放原始碼組件。如果開發團隊其實有意識到這個情況——不幸的是，許多人並非如此——他們可能會判斷更新升級到新版所可能帶來的意外後果的風險，超過了升級所帶來的好處。比如說，對於只能從外部入侵的漏洞來說，內嵌軟體所造成的風險可能非常小。

或者也可能是時間／資源的問題。許多團隊已經因為開發與測試新的程式碼而忙到焦頭爛額，除非有特別關鍵性的問題，否則為既有軟體進行更新的優先次序可能會比較低。Synopsys的「2023全球DevSecOps狀況」報告發現，在針對1000名IT安全專業人員的調查之中，有28%人員表示他們的組織可能需要長達三周的時間去修補目前已經部署完成的應用程式的關鍵性安全風險／漏洞，另外有20%人員表示這可能需要長達一個月的時間。而這個調查結果適用於所有類型的漏洞——專有軟體、市售軟體、第三方軟體以及開放原始碼軟體。

正如最近十年以來的的OSSRA報告所指出的，開放原始碼軟體與市售軟體不同——這不表示開放原始碼軟體會更糟，也不表示它們會更好，但它們卻是不同的——它們需要不同的技術來管理。比如說，在軟體修補方面，市售軟體與開放原始碼軟體就大相逕庭。購買市售軟體的時候，通常需要進行一些審查，以此作為供應商管理的一部分。另一方面，開放原始碼軟體往往只是使用者在開發人員的同意之下自行下載而已。可能有一些組織確實有針對這個問題的保護措施——比如說，只使用具有寬鬆授權的程式碼——但在許多組織中，甚至連這樣的指導方針都沒有。

所有使用市售軟體的組織都已經習慣了程式修補以及更新會被「推送」到他們的軟體上，或至少會有供應商發送通知告訴他們有更新——通常是緊急更新——可供下載。但對於開放原始碼軟體來說，這種情況很少發生，開放原始碼軟體的使用者被預期會自行持續了解組件的狀態，並且在有可用新版本的時候進行下載。

想要持續對您使用的開放原始碼的版本異動有所了解，唯一可行的解決方案就是要有一個準確而全面的開放原始碼清單以及自動化的流程來監控漏洞、升級以及在您軟體中含有的開放原始碼的整體健康狀況



缺失發現與建議

無論是單一開發人員還是大型公司，每個人都有責任維繫軟體供應鏈安全的實務操作，藉此降低風險。隨著針對軟體供應鏈的攻擊越來越多，對於開放原始碼使用、組件以及相依性關係的有效管理對於風險管理就變得更加關鍵了。對於旗下產品中含有開放原始碼的組織——正如本報告所描述的，實質上相當於所有組織——都應該主動為其自身在安全軟體開發實務所需負責的部分，管理開放原始碼的風險。

「確保軟體供應鏈：管理開放原始碼軟體以及軟體物料清單的實務操作」，由美國網路安全暨基礎設施安全局於2023年底發布，這份文件提供了在軟體供應鏈中使用開放原始碼的詳細指南，其中包括：

- **將開放原始碼整合進產品的安全建立流程之中，使用與內部開發組件相同的政策和流程。**
  安全團隊往往會定義有關開放原始碼的安全政策、流程以及工具。在理想的情況下，組織內部會有一個資料庫，其中的組件都已經接受過SCA安全分析工具的初步漏洞評估，而開發人員會從這個資料庫裡選擇具有所需功能的組件，然後在開發以及／或建立階段進行進一步掃描，以便儘早發現並掌握問題。

- **追蹤開放原始碼的更新，並且監控問題與漏洞。**
  在辨識出漏洞存在的時候，應該要評估受影響的軟體，藉此確定有問題的組件的普及性，以及它在產品中的使用狀況。如果組件有漏洞存在就應該要更新，而如果這個組件不再有人員維護，則強烈建議尋求替代解決方案。

- **使用SBOM。**
  了解在軟體之中含有哪些對於準確與完整的程式碼管理來說至關重要的組件。SBOM是包含軟體組件詳細資訊以及供應鏈關係的正式記錄。SBOM增加了軟體透明度，並且記錄了組件的來源。在漏洞管理的領域裡，SBOM會對漏洞的辨識與修復有所幫助。從程式碼品質的角度來看，SBOM的存在可能會是供應商在軟體開發生命週期中落實安全軟體開發實務的表徵。

行政命令(EO) 14028《改善國家網路安全》指出，組織可能會被要求直接向購買方提供SBOM，或在公共網站上發布SBOM，且政府與非政府等組織都有可能需要審查SBOM，藉此確保軟體產品符合SBOM的最低要素需求。這一份EO還指導美國商務部以及國家電信暨資訊管理局(NTIA)發布了《軟體物料清單(SBOM)的最低要素需求》，其中概述了SBOM所需要的活動、資料以及滿足SBOM要求的範例格式。SPDX與CycloneDX這兩種格式一般被認為是應用最為廣泛的機器可讀取SBOM格式。2023年中，美國行政管理與預算局(OMB)發布了OMB-23-16備忘錄，以此更新更新先前發布的OMB備忘錄，其中允許聯邦機構提出以下要求：

- 依據各機構本身決定的軟體關鍵性要求或其他標準索取SBOM
- 依據NTIA定義格式的其中之一來索取SBOM

創造安全的軟體開發框架

軟體的生產者在確保軟體供應鏈的安全上扮演至關重要的角色，可以造福其客戶與其使用者。美國國家標準暨技術研究院(NIST)的安全軟體開發框架(SSDF)提出了一系列實務做法，作為利用標準化方法安全開發軟體的基礎。美國政府已經對外通告，對於直接或間接由美國政府採購的所有軟體，都有可能需要有符合NIST SSDF的要求的證明，這意味著在不久的將來，軟體供應商可能需要自我證明自身確實遵守SSDF。

像是Synopsys SSDF準備評估這樣的評估工具可以辨識您的組織的軟體開發實務做法是否與SSDF所規範的實務做法與目標一致，讓您可以自信滿滿地證明您的軟體開發流程符合SSDF的標準。同樣地，Synopsys SBOM服務建立在Black Duck稽核服務流程的基礎，對您的軟體進行全面的安全稽核並產生SBOM，這對於還沒有產生SBOM的能力卻需要基本的SBOM的組織而言，是一種很有用的服務。

負有法規或合約義務的軟體製造商可能會被要求提供經過稽核的SBOM要求。軟體的採購方可能會想要稽核其供應商所產生的SBOM。在任何一種前述狀況之下，都會需要在軟體稽核方面具有良好商譽，而且值得信賴的第三方組織協助。Synopsys的SBOM稽核與確效服務建立在經過驗證的流程基礎之上，可以稽核軟體並確認客戶所產生的SBOM是否準確反映了它的供應鏈。

了解您的程式碼裡面有什麼內容

回顧本報告的缺失發現

- 在接受掃描的1,000多個程式碼庫中，有96%包含開放原始碼
- 有77%的原始碼與檔案來自於開放原始碼
- 有53%的程式碼庫之中含有開放原始碼授權衝突
- 有84%接受安全風險評估的程式碼庫含有漏洞；更有74%含有高風險漏洞
- 有91%的程式碼庫之中含有比最新版本落後10個版本以上的組件

無論您的組織是開發或使用軟體，幾乎都可以肯定您的組織的軟體含有開放程式碼組件。您確實了解這些組件是什麼，以及它們是否會帶來安全或授權風險嗎？到了2024年，將會有96%的程式碼都包含開放原始碼，因此程式碼的可見性必須成為首要的目標。既然已經知道有91%接受風險評估的程式碼庫使用了遠遠落後於當前版本的開放原始碼時，消費者就更需要改善程式碼的更新，尤其是在使用了當前流行的開放原始碼組件的時候。

開放程式碼的更新應該與更新您自己的團隊開發的程式碼具有相同的優先順序。要建立並且維護SBOM，詳細說明程式碼裡的內容，包括有關版本、授權以及來源的詳細資訊。設定規律的升級週期，尤其是當您的開放原始碼程式庫來自於具有活躍的維護人員的熱門專案的時候。

如果無法全面了解您的程式碼，並且維持主動的軟體衛生實務操作，您的軟體就會面臨針對開放程式碼漏洞的攻擊的潛在風險以及IP合規性問題。從使用自動化SCA工具在SDLC的前期發現安全性、程式碼品質以及授權問題開始——因為無庸置疑地，您就是需要您的程式碼裡面到底有些什麼。

如果無法全面了解您的程式碼，並且維持主動的軟體衛生實務操作，您的軟體就會面臨針對開放程式碼漏洞的攻擊的潛在風險以及IP合規性問題。

Synopsys與眾不同之處

Synopsys提供的整合解決方案可以改變您建構與交付軟體的方法，在加速創新的同時解決業務上的風險。有了Synopsys的協助，您的開發人員維護程式碼的速度就會像編寫程式碼一樣迅速。您的開發與DevSecOps團隊可以在開發管道裡執行自動測試，不會影響開發速度。您的安全團隊可以主動管理風險，並將補救問題的產能集中在對您的組織最重要的關鍵之上。我們無與倫比的專業知識可以協助您規劃並執行任何安全計畫。只有Synopsys能夠提供您在軟體中建立商譽的時候所需的一切。

請造訪以下網頁：www.synopsys.com/software ，了解更多有關於Synopsys Software Integrity Group的資訊。

專有名詞

程式碼庫
構成應用程式或服務的程式碼與其相關資料庫。

二進位分析
一種靜態分析，其用途是在無法存取原始碼的狀況下識別應用程式的內容。

CWE
常見弱點列舉(Common Weakness Enumeration，CWE)是一個由社群建立的軟體暨硬體弱點類型清單，其中分為三個層級。CWE包含超過600種類別，包括緩衝區溢位、路徑／目錄樹遍歷錯誤、競爭條件、跨網站指令碼攻擊、硬編碼密碼以及不安全的亂數等類別。

CVE
常見漏洞暴露(Common Vulnerabilities and Exposures，CVE)是一個對外公開的資訊安全缺陷列表。

Black Duck Security Advisory (BDSA)
詳盡、及時而一致的開放原始碼漏洞資訊，BDSA為Synopsys的客戶提供開放原始碼漏洞的前期與補足性通知以及升級／修補指南。BDSA會提供當天的漏洞通知、可行的緩解指南以及替代措施資訊、嚴重度評分、參考資料等。

軟體組件
預先編寫完成的程式碼，開發人員可以將其添加到自己的軟體中。軟體組件可能會是一個實用的工具，例如日曆功能，或也可能是支撐整個應用程式的全面軟體架構。

相依性
當有其他軟體使用某個軟體的組件時，該軟體組件就會成為相依性——也就是說，其他軟體會對這個軟體元件產生依存性。任何應用程式或服務都可能具有許多相依性，這些相依性本身也可能依存於其他組件。

程式碼片段
程式碼片段是一段小型而可以重複再利用的程式碼，開發人員可以將其複製貼上到自己的程式碼之中。即使軟體之中可能只包含一小段開放來源程式碼，但軟體的使用者仍然必須遵守與該程式碼片段相關的任何授權條款。

開放原始碼授權
在軟體中有使用開放原始碼組件(或該組件的程式碼片段)的時候，用來規範終端使用者的義務的約定條款，其中包含如何使用以及傳播該組件的規範。大多數開放原始碼授權屬於以下二者之一。

> **寬鬆授權**
> 寬鬆授權的使用限制條款較少。這類授權的主要要求通常是將原始程式碼的作者註明在程式碼裡。
>
> **著作傳授權**
> 著作傳授權通常會包含互惠義務聲明，規定修改與延伸的版本需要在與原始程式碼相同的約定條款之下發布，且在受到要求的時候必須提供包含變更內容的原始碼。商業實體對於在其軟體中採用具有著作傳授權的開放原始碼軟體抱持審慎的態度，因為使用這樣的原始碼可能會造成整個程式碼庫的智慧財產權都產生疑慮。

軟體成分分析(SCA)
一種應用程式安全工具，用途為自動化管理開放原始碼軟體。SCA工具會被整合在軟體開發的生命週期之中，識別程式碼庫中的開放原始碼軟體，提供風險管理與緩解的建議，並且執行授權合規性驗證。

軟體物料清單(SBOM)
程式碼庫中的軟體組件與相依性關係的完整清單，通常利用軟體成分分析工具產生。如同國家電信暨資訊管理局所說明的，「SBOM之中，應當以可以由機器讀取的形式，針對您的軟體組件與依賴關係、組件資訊及其階層關係提供一份清單」。由於SBOM的目的在於在公司與社群之間共享，因此擁有一致的格式(既可人工閱讀又可由機器讀取)及一致的內容至關重要。美國政府指南目前指定兩種標準為其所認可的格式，也就是軟體包資料交換(SPDX)以及CycloneDX。

撰稿人

《開放原始碼安全與風險分析》報告是在Synopsys軟體完整性小組的通力合作之下產生的，團隊中包含我們的稽核服務、諮詢、研究、法律與行銷團隊的成員。團隊成員的辛勤工作讓OSSRA成為過去十年裡在開放原始碼程式碼品質、安全與授權合規領域裡名列前茅。

在此特別感謝團隊成員對於本年度報告的貢獻，團隊成員包括：Nancy Bernstein、Scott Handy、Siobhan Hunter、Matt Jacobs、Natalie Lightner、Merin McDonell、Mike McGuire、Phil Odence、Rie Sekine、Liz Samet、Jenny Stout以及Jack Taylor.

在我們一起為OSSRA工作的九年裡，Rachel Bay展現了她不可思議的設計魔法。能夠參與撰寫這份報告是我的榮幸，也是我的榮耀。——Fred Bals