

舊系統健檢報告（範例）

標的：BugNET 議題追蹤系統 | 智序資訊工作室 | 2026-09-23

這是什麼	一份 真的跑出來 的範例：拿一套公開、已停止維護的開源系統，照我們對客戶做健檢的方式實際做一次。沒有捏造的數字；每一項發現都附檔案與行號，任何人都能對照公開原始碼查證。
標的系統	BugNET (github.com/dubeaud/bugnet)，ASP.NET WebForms+SQL Server 的議題追蹤系統，MS-PL 授權。專案說明寫明「不再開發或維護」，GitHub 上已封存。
分析版本	最後一次提交 cfab4ae (2019-05-08)。這個儲存庫的提交紀錄從 2011-09-24 開始，共 762 次提交。
怎麼做的	ForgeHelm 分析引擎在本機掃描整個儲存庫，接著由人逐項檢視引擎的輸出、剔除誤報，再追進程式碼確認，最後寫成本報告。
段落標記	引擎產出 分析引擎直接產生的資料 人工判讀 人讀過程式碼後的結論 人工清點 人對原始碼做的計數
範例的限制	開源專案沒有真實的營運成本、人力單價與使用情境，所以第五節「遷移路徑比較」只比較範圍與難度，不填金額；第六節不估工時。真實健檢會在訪談後補上這兩樣。

一、主管摘要

人工判讀	一句話：這套系統還能用，但不能放著不管。有三件事應該先處理，三件事要排進計畫；遷移不急。 先處理（安全）： 附件可以冒名上傳。附件上傳的 Web 服務讓呼叫端自己指定上傳者，任何有權限的登入使用者都能把附件記成別人上傳的。議題系統常被拿來當「誰在什麼時候做了什麼」的紀錄，這會讓紀錄不可信。(R1) 錯誤頁會把部分請求內容和錯誤訊息原樣寫回頁面。這可能被拿來做跨網站腳本攻擊 (XSS)，也會洩漏錯誤細節。修法只要幾行。(R2) 前端套件與編輯器過舊。jQuery 2.1.4、jQuery UI 1.11.4、Bootstrap 3.3.6、CKEditor 3.6.4 都早於各自已公開的安全修補版本。(R3) 排進計畫（維護）：
-------------	--

4. 查詢與排序的 SQL 是用字串組出來的。我們追過每一個入口，目前找不到可以直接注入的地方；但保護靠的是別的元件，只要將來多開一個入口，就會變成 SQL 注入。(R4)
5. 系統裡塞了一整套別人的郵件函式庫原始碼，還有兩個早就不用的專案。郵件函式庫比系統本體還大（約 7.7 萬行對 6.9 萬行），而且沒人更新。(R8、R9)
6. 設定與密碼政策需要確認。開發用設定若直接上線會洩漏錯誤細節；密碼最短 7 碼、不要求符號。(R5、R6)

遷移：.NET Framework 4.8 沒有結束支援日期（隨 Windows 支援）。除非業務上需要雲端原生或大改介面，「就地升級到 4.8 並更新套件」是風險最低的一條路（第五節）。

二、現況盤點

系統規模

人工清點

項目	數量	說明
專案 (.csproj)	23	目標框架 .NET Framework 4.0–4.5.2
BugNET 自己的 C# 程式檔	510 檔／約 6.9 萬行	含 147 個由設計工具產生的 .designer.cs
內嵌的第三方郵件函式庫 (LumiSoft.Net)	300 檔／約 7.7 萬行	以原始碼形式放在 src/LumiSoft.Net，比系統本體還大
網頁與使用者控制項	39+62	WebForms 的 .aspx 與 .ascx，另有 1 個主版頁面、1 個 Web 服務 (.asmx)
多語系資源檔 (.resx)	803	介面文字的翻譯
資料庫物件 (SQL Server 資料庫專案)	176+48+7	預存程序、資料表、檢視表 (src/BugNET.Database)
NuGet 相依套件	51	分散在 20 份 packages.config，各專案版本一致

架構：誰依賴誰

人工整理

下圖依各 .csproj 的專案參照彙整繪製：只畫正式執行的專案（測試專案 3 個與資料產生工具 1 個未列入），7 個外掛與模組專案合成一格。

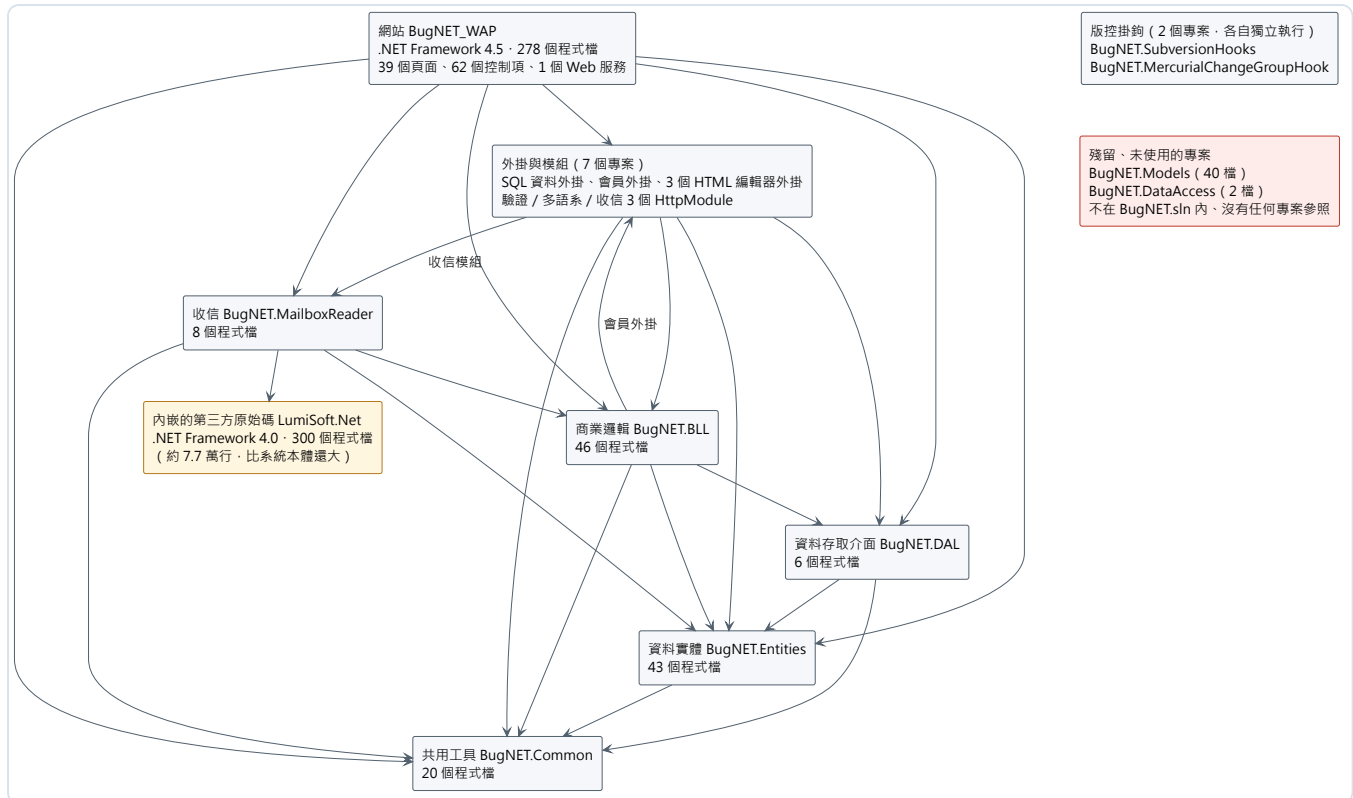


圖 1 專案相依圖（彙整）。箭頭表示「參考」。黃色是內嵌的第三方原始碼；紅色的兩個專案不在方案檔 BugNET.sln 內、也沒有任何專案參考（見 R9）。

讀這張圖的重點有三個。第一，網站（BugNET_WAP）直接依賴 13 個專案，職責集中在網站這一層。第二，資料存取走「Provider」外掛（Provider.SqlDataProvider），換資料庫理論上只要換外掛，但實際上 176 支預存程序都是 SQL Server 語法。第三，正式執行的專案裡，只有收信功能（BugNET.MailboxReader）依賴內嵌的郵件函式庫，這是將來替換它時的切入點。

引擎產出 下圖是分析引擎自動產生的分層圖，原樣呈現。

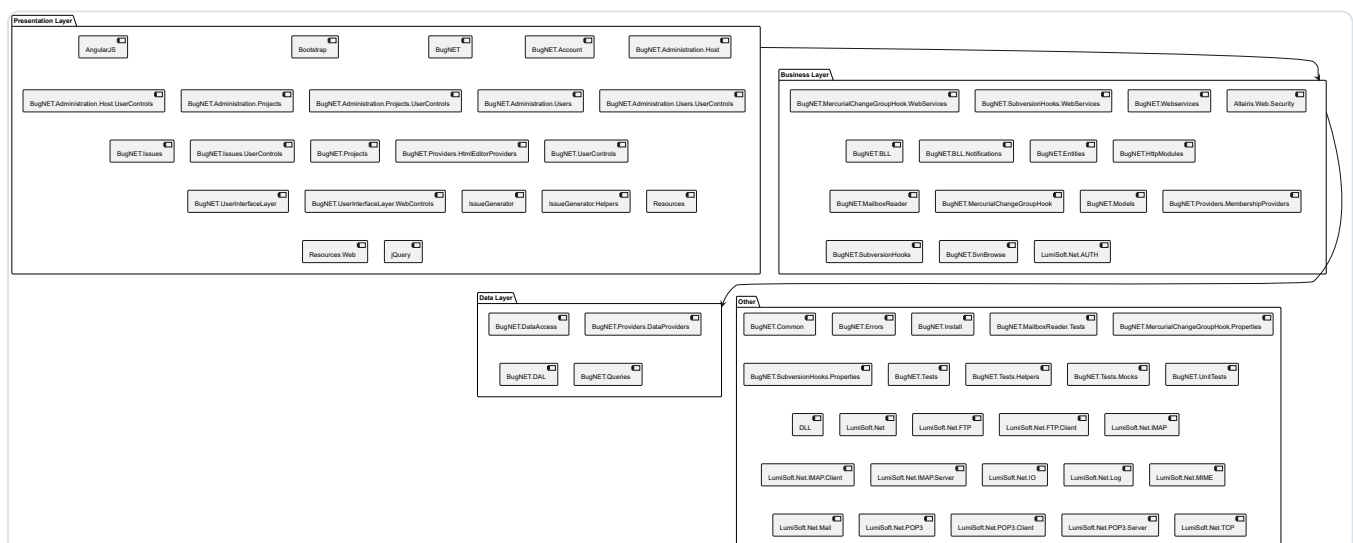


圖 2 引擎產生的分層圖（原樣）。人工判讀時更正兩處：展示層的「AngularJS」是誤列，BugNET 沒有使用 AngularJS；商業層的「LumiSoft.Net.AUTH」屬於內嵌的郵件函式庫，不是 BugNET 自己的商業邏輯。

相依套件

人工清點 51 個套件中，下列幾個明顯過舊或已停止維護（完整清單可依 `packages.config` 重現）。

套件	目前版本	狀況
jQuery	2.1.4	早於 3.5.0；CVE-2020-11022、CVE-2020-11023 等 XSS 問題在該版修補
jQuery UI	1.11.4	早於 1.13.2；CVE-2021-41182 等三項在 1.13.0 修補，CVE-2022-31160 要到 1.13.2
Bootstrap	3.3.6	早於 3.4.1，該版修補了 CVE-2019-8331 等；但 CVE-2024-6485 連 3.4.1 都在受影響範圍內，3 系列沒有修補版
CKEditor	3.6.4	CKEditor 3 早已停止更新，連後繼的 CKEditor 4 開源版也在 2023-06-30 結束支援；NVD 的 CVE-2014-5191（預覽外掛 XSS，4.4.3 修補）影響範圍涵蓋 3.6.4
log4net	2.0.5	早於 2.0.10，該版修補了 CVE-2018-1285
Newtonsoft.Json	7.0.1	早於 13.0.1，該版修補了 CVE-2024-21907
DotNetOpenAuth（6 個套件，另有 1 個 Google 登入擴充 1.1.2）	4.3.4	第三方登入用；原專案的 GitHub 儲存庫已封存，最後更新在 2018 年 1 月

上表的 CVE 對應已於 2026-09-23 以 OSV 漏洞資料庫與 CVE 官方紀錄逐條核對。真實健檢會再以客戶慣用的弱點掃描工具核對一次。版本號本身可直接在 `src/BugNET_WAP/packages.config` 等檔案查證。

三、風險清單

人工判讀 等級是人讀過程式碼、評估實際可利用程度後給的，不是引擎的原始嚴重度。每一項都標明來源：引擎先標出、人工確認，或人工追程式時發現。

編號	等級	一句話	來源
R1	中高	附件 Web 服務可冒名上傳	人工發現
R2	中	錯誤頁原樣輸出請求內容與錯誤訊息	引擎標出 + 人工確認
R3	中	前端套件、編輯器與幾個後端套件過舊	人工清點
R4	中	查詢與排序用字串組 SQL（目前入口安全）	引擎標出 + 人工追入口
R5	低中	開發用設定若直接上線會洩漏錯誤細節	引擎標出 + 人工確認
R6	低	密碼政策偏弱	人工
R7	低	安裝／升級頁輸出伺服器路徑與錯誤訊息	引擎標出 + 人工確認
R8	維護	內嵌一整套第三方郵件函式庫原始碼	引擎 + 人工清點
R9	維護	兩個早已不用的專案留在原始碼裡	人工
R10	低	預存程序對欄位套函式，使索引失效	引擎標出 + 人工確認
R11	低	單元測試裡寫死一組測試信箱帳密	引擎標出 + 人工確認

R1

附件 Web 服務可冒名上傳

中高

附件上傳的 Web 服務（公開範例刻意不列檔案、行號與參數名稱，避免成為攻擊指引；交給客戶的報告會列）

看到什麼

附件上傳的 Web 服務讓呼叫端自己指定「上傳者」，並照單全收記進資料庫。服務明明知道是誰登入的，卻沒有用登入身分。

為什麼要緊

任何登入、且對該專案有權限的使用者，都能把附件記成別人上傳的。議題系統常被拿來當稽核或爭議時的紀錄，這會讓「誰做了什麼」不可信。

建議

上傳者一律以伺服器端的登入身分為準，不接受呼叫端指定。改動很小，但要一併檢查其他 Web 方法是否也有「由呼叫端指定身分」的寫法。

來源

引擎沒有標出這一項；是人工追「附件路徑是否可被操控」時，順著呼叫鏈看到的。這正是健檢需要人工判讀的原因之一。

R2 錯誤頁原樣輸出請求內容與錯誤訊息

中

錯誤處理頁（同 R1，公開範例不列位置）

- | | |
|-------|---|
| 看到什麼 | 錯誤頁在某些情況下，會把請求內容與例外訊息直接寫回瀏覽器，沒有做編碼。 |
| 為什麼要緊 | 這屬於跨網站腳本（XSS）的候選，例外訊息也會把系統內部細節帶給外人。觸發需要特定條件，機率不高，所以列為「中」。 |
| 建議 | 輸出前做 HTML 編碼，或不輸出例外細節、改寫到日誌。幾行就能修好。 |

R3 前端套件、編輯器與幾個後端套件過舊

中

`src/BugNET_WAP/packages.config` 等 20 份相依清單

- | | |
|-------|--|
| 看到什麼 | 見第二節的相依套件表：jQuery、jQuery UI、Bootstrap、CKEditor、log4net、Newtonsoft.Json 都有已公開、影響目前版本的安全問題；其中 Bootstrap 3 與 CKEditor 3 在同一系列裡沒有修補版，DotNetOpenAuth 的原專案已封存。 |
| 為什麼要緊 | 這些都是公開的已知問題，掃描工具一掃就會出現；對外開放的系統最先被試的就是這一類。 |
| 建議 | 後端套件（log4net、Newtonsoft.Json）可以先升，風險低。前端套件要注意 jQuery 2 升 3 有不相容的變更，101 個頁面與控制項的腳本要一起測；Bootstrap 升到 3.4.1 能修掉大部分，但 CVE-2024-6485 要離開 3 系列才修得掉。CKEditor 3 升級等於換編輯器，要當成一個小專案。 |

R4 查詢與排序用字串組 SQL（目前入口安全）

中

`src/Library/Providers/DataProviders/SqlDataProvider/SqlDataProviderIssueQuery.cs:111-165`、`SqlDataProviderIssues.cs:389-431`

- | | |
|---------|--|
| 看到什麼 | 自訂查詢把「欄位名稱」與「比較運算子」直接串進 SQL，只有條件的值有用參數；排序欄位只要含句點，就原樣接進 <code>ORDER BY</code> （程式註解寫著「把正確性交給呼叫它的介面」）。 |
| 我們追了入口 | 頁面上的查詢條件來自下拉選單，受 WebForms 的事件驗證保護；排序字串存在 ViewState，受簽章保護（設定檔沒有寫死 <code>machineKey</code> ，所以不是所有安裝共用同一把金鑰）；Web 服務 <code>GetProjectIssues</code> 只把使用者輸入放進參數值，欄位與運算子都寫死（ <code>BugNetServices.aspx.cs:510-550</code> ）。目前沒有找到可以直接注入的入口。 |
| 為什麼仍要處理 | 保護全部靠別的元件。接手的人只要多開一個入口（例如讓網址帶排序參數、或新增一支 API），就會變成 SQL 注入，而且不容易察覺。 |
| 建議 | 欄位名稱與運算子改成白名單比對。改動集中在兩個檔案。 |

R5 開發用設定若直接上線會洩漏錯誤細節 低中

`src/BugNET_WAP/Web.config:28 ( debug="true" )、:54 ( customErrors mode="Off" )`

- 看到什麼 儲存庫裡的設定是開發用的。發佈用的轉換檔 `Web.Release.config` 會拿掉 `debug`、把錯誤頁改成只對本機顯示，這部分原作者有做對。
- 為什麼要緊 舊系統常見的做法是直接複製整個資料夾上線，沒有經過發佈轉換。那樣就會把詳細錯誤訊息給任何人看。
- 建議 到正式主機確認實際生效的設定。這一項靠看原始碼無法下結論，要看部署。

R6 密碼政策偏弱 低

`src/BugNET_WAP/Web.config` 的 `membership` 設定

- 看到什麼 最短 7 碼、不要求符號；10 分鐘內錯 5 次會鎖定。
- 建議 提高最短長度，或改接公司既有的帳號系統（AD 或單一登入），讓密碼政策跟著公司走。

R7 安裝／升級頁輸出伺服器路徑與錯誤訊息 低

`src/BugNET_WAP/Install/Install.aspx.cs:220、363、496、517、540-541`

- 看到什麼 安裝與升級流程出錯時，會把伺服器上的路徑與錯誤訊息直接印在頁面上。
- 建議 安裝完成後移除 `/Install` 資料夾，或在網站伺服器上限制只有管理者能連。

R8 內嵌一整套第三方郵件函式庫原始碼 維護

`src/LumiSoft.Net`（300 檔、約 7.7 萬行、.NET Framework 4.0）

- 看到什麼 IMAP、POP3、SMTP、MIME 的完整實作以原始碼形式放在專案裡，只被收信功能（`BugNET.MailboxReader`）與它的測試專案使用。引擎在整個系統標出的 1,528 個位置裡，681 個（45%）落在它身上；最複雜的 15 個方法全部在第三方程式碼裡（14 個在它身上，1 個在同樣移植自外部的 `BugNET.Common/IsEmail.cs`）。
- 為什麼要緊 等於自己在維護一套郵件協定實作，但沒有人再更新它；裡面的 MD5、HMAC-MD5 是舊協定規定要用的演算法，改不了，只能整個換掉。
- 建議 改用仍在維護的郵件套件，或把收信功能獨立成一個小服務。切入點很清楚：只有一個專案依賴它。

R9 兩個早已不用的專案留在原始碼裡 維護

`src/BugNET.Models` (40 檔)、`src/BugNET.DataAccess` (2 檔，含 Entity Framework 的 `BugNETContext`)

看到什麼 這兩個專案不在方案檔內，也沒有任何專案參照。2015-12-19 的提交 `744c6f73` 說明它們已「移到 MVC 專案」，但檔案沒有刪。

為什麼要緊 接手的人會以為系統有兩套資料存取，花時間研究一套根本沒在跑的程式，甚至改錯地方。

建議 確認沒有外部部署在用之後刪除，或移到封存分支。

R10 預存程序對欄位套函式，使索引失效 低

例：`src/BugNET.Database/Stored Procedures/BugNet_ProjectCategories_CreateNewCategory.sql`:
`6 (WHERE LOWER(CategoryName) = LOWER(@CategoryName))`

看到什麼 引擎標出 31 處「讓索引失效」的寫法。抽查確認這個類型是真的，但引擎把行號標在程序開頭，實際位置要人工找。

建議 資料量不大時影響很小；若議題數量上萬、查詢變慢，再依實際執行計畫處理。

R11 單元測試裡寫死一組測試信箱帳密 低

`src/UnitTests/MailBoxReaderTests.cs:38-40`

看到什麼 測試程式裡寫著一組指向 `bugnetproject.com` 的測試信箱帳號與密碼。

建議 若該帳號仍有效就更換，並改成從測試環境的設定讀取。

四、合規對映

人工判讀 BugNET 會存使用者的姓名與電子郵件，議題內容也可能夾帶個人資料，所以適用個人資料保護法；若使用單位推 ISO 27001，上面的風險可以這樣對應：

風險	ISO/IEC 27001:2022 附錄 A	個人資料保護法
R1、R2、R4、R7	8.28 安全程式設計	第 27 條「適當之安全措施」
R3、R8	8.8 技術脆弱性管理	同上
R5、R7	8.9 組態管理	同上
R6、R11	5.17 鑑別資訊、8.5 安全鑑別	同上
R9	8.25 安全開發生命週期	—

引擎另外依 24 套框架各產出一份對照報告（繁中與英文，Markdown 與 PDF，另附 SARIF 機器可讀格式）。對一套舊系統來說，那些報告裡的「涵蓋率百分比」多半反映的是「程式裡有沒有出現相關字眼」，不能讀成合規分數，所以本報告沒有引用。它們的用處是當成佐證的索引：稽核員問到某條控制時，可以直接翻到對應的檔案與行號。能不能通過驗證，仍取決於客戶自己的管理制度，不是程式碼本身。

五、遷移路徑比較

人工判讀 範例不填金額（開源專案沒有人力單價與營運成本）。下表只比較範圍與難度，數字都來自第二節的清點。

	A 維持現狀	B 就地升級到 .NET Framework 4.8	C 換技術棧到 ASP.NET Core
要 做 什 麼	修 R1–R3、R5–R7；前端套件升級	A 的全部，加上 23 個專案改目標框架、51 個套件更新、替換 DotNetOpenAuth 與內嵌郵件函式庫	WebForms 無法在新版 .NET 上執行：39 個頁面、62 個控制項與 1 個主版頁面的介面層要重寫；帳號系統從 Membership 換到 ASP.NET Core Identity，帳號與密碼雜湊要遷移
可 以 沿 用 的	全部	全部程式與 176 支預存程序	商業邏輯與資料層（BLL、DAL、Entities、Common 約 115 個程式檔）移植性較高；176 支預存程序可以保留
支 援 期 限	停在 4.0–4.5.2 的專案會越來越難找到相容的工具與套件	.NET Framework 4.8 沒有結束支援日期，隨 Windows 支援	新版 .NET 依微軟的版本週期定期升級
難 度	小	中	大
適 合 的 情 況	系統即將汰換、只求撐到那天	還要用好幾年，但沒有要大改功能	要上雲、要大改介面，或團隊已經不想維護 WebForms

我們的建議是 B。先把安全問題止血，再升級到 4.8、換掉兩個已停止維護的元件，系統就能在可控的狀態下再用好幾年；等真的有業務需求時再談 C，而且那時 R8、R9 已經清掉，重寫的範圍會更乾淨。

六、分階段建議

人工判讀 範例不估工時（取決於貴單位對這套系統的熟悉度與既有測試）；下面用小、中、大表示相對規模。

階段	先動什麼	先別碰什麼	規模
第一階段：止血	R1、R2、R7 的修補；確認正式環境的設定（R5）；更換測試信箱密碼（R11）	前端套件先不動，避免和安全修補混在一起	小
第二階段：清理	R4 改白名單；刪除或封存殘留專案（R9）；升級後端套件（log4net、Newtonsoft.Json）；密碼政策（R6）	內嵌郵件函式庫先保留，等第三階段一起換	中
第三階段：升級	全部專案升到 .NET Framework 4.8；替換 DotNetOpenAuth 與內嵌郵件函式庫（R8）；前端套件與編輯器升級（R3）	介面層不重寫	中
之後（視需要）	評估路線 C	—	大

附錄 A 引擎輸出與人工判讀的落差

引擎產出 人工判讀 分析引擎依 24 套框架，在這套系統標出 1,528 個位置（去除重複後）。我們逐類檢視，結果如下：

引擎標出的類型	位置數	人工判讀結果
在第三方郵件函式庫內（各類合計）	681	歸入 R8 一併處理，不逐條列
「在程式裡找到相關字眼」的證據型標記	698	是對照稽核條文用的索引，本身不代表問題
未編碼輸出（可能 XSS）	26	17 處輸出的是固定文字，排除；9 處輸出變數：1 處 → R2，7 處 → R7，匯出功能 1 處（GridViewExportUtil.cs:81）待依實際資料確認
SQL 字串串接	25	14 處其實不是 SQL（驗證群組名稱、確認訊息、網址、記錄訊息、郵件協定字串）；3 處是常數 SQL 或通用設定函式；4 處由設定檔決定資料表與欄位名稱、值有參數化，列為可接受；4 處 → R4
路徑穿越	16	6 處在郵件函式庫；1 處是被註解掉的程式碼；6 處在安裝頁，路徑來自伺服器設定；附件下載與刪除 2 處，檔名在儲存時已去除目錄成分（IssueAttachmentManager.cs:139），排除；1 處為電子郵件範本讀取，範本名稱由程式內部決定，排除。追這一類時發現了 R1。
寫死帳密	2	1 處是部署範本的佔位字串，排除；1 處 → R11
「DELETE 缺 WHERE」（引擎列為最嚴重）	2	其實是 C# 的方法名稱與一行註解，排除
讓索引失效的寫法	31	類型屬實，行號需人工定位 → R10

這張表說明兩件事。第一，引擎把該看的地方都指出來了，而且是有檔案與行號的，人只要順著看；第二，引擎的原始輸出不能直接交給客戶，健檢的價值有一大半在「剔除不是問題的，把真正要緊的講清楚」。R1 就是人順著引擎指的方向追下去，才找到引擎自己沒有標出的問題。

附錄 B 方法、範圍與如何查證

- 分析方式：ForgeHelm 分析引擎對整個儲存庫做靜態分析：語法樹、呼叫關係、設定檔、SQL 與 24 套框架的規則比對，並產生架構圖與各框架的對照報告。之後由人檢視輸出、追進程式碼確認。

- **分析範圍設定：**本次關閉了與這套系統無關的技術家族（Silverlight、Blazor，以及與實際版本不符的 jQuery 1.0／1.5、Bootstrap 4／5 解析器），只保留對得上的 jQuery 與 Bootstrap 版本。這一步對應健檢流程裡的「確認範圍」。
- **沒有做的事：**沒有執行系統、沒有做滲透測試，也沒有連線到任何正式資料庫；R5 這類「要看部署才知道」的項目只能列為待確認。
- **如何查證：**取得 github.com/dubeaud/bugnet 的提交 `cfab4ae`，除了 R1、R2 兩項（刻意不列位置，理由見該兩項），本報告的每個檔案與行號都能直接對照。
- **範例與真實健檢的差別：**真實健檢還會有範圍確認與訪談（了解系統怎麼被使用、誰在維護、有沒有對外開放），並補上成本比較的金額與分階段的工時區間；報告為繁體中文的 PDF 與 Word，另做一場 30 分鐘的線上簡報。

本報告為範例，以公開的開源專案實際分析產出，僅供了解健檢交付內容之用；不構成對 BugNET 或其使用者的安全保證，也不是法律或合規意見。BugNET 以 MS-PL 授權發布，版權屬原作者。

智序資訊工作室（Smart Sequence Tech） | <https://www.smartsequence.tech>