# AIVA 專案模組結構調整建議

根據 **AIVA** 專案最新程式碼，以下針對五大模組（核心、通用、功能、掃描、整合）提出主要檔案歸類、存在問題與調整建議：

## 核心（Core）

* **AI 引擎與主控**（services/core/aiva\_core/ai\_engine/, bio\_neuron\_master.py, ai\_controller.py 等）：核心模組涵蓋 AI 決策引擎與主控制器，當前存在 **重複實作** 問題。例如 bio\_neuron\_core.py 與 bio\_neuron\_core\_v2.py 功能重疊[[1]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L11-L19)。建議**統一 AI 引擎版本**，保留新版（如將 bio\_neuron\_core\_v2.py 重命名為統一版本）並移除舊版備份[[2]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L33-L40)。同時確保 AI 決策相關程式碼集中於核心，避免分散多處。
* **最佳化核心**（optimized\_core.py）：此檔案體積龐大（約465行）且職責過於集中，包含多個類別與函數[[3]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L7-L14)。建議**模組化拆分**：依功能拆成子模組（例如 performance/parallel\_processor.py, performance/memory\_manager.py 等）[[4]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L41-L48)，降低單檔複雜度。拆分後在核心模組內以封裝套件呈現，提升可讀性與維護性。
* **過長函數問題**：部分核心程式存在超大函數，影響可讀性與維護。例如 authz/matrix\_visualizer.py 中單一函數長達209行[[5]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L14-L17)；ai\_engine/tools.py 內亦有近百行的函數[[5]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L14-L17)。建議將此類**超長函數重構**為多個較小函式，提高模組內聚度與易懂性[[6]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L50-L53)。
* **AI UI 資料結構**（ai\_ui\_schemas.py）：目前在核心中以單一檔案定義了18個與UI相關的資料類別，檔案複雜度高[[7]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L22-L28)。建議將這些 **UI Schema** **重組**：可考慮併入通用的 schemas/ 資料夾（將 UI 相關結構併入 AIVA 共用 Schema），或拆分成獨立模組檔案以降低耦合。在官方優化計劃中，此檔已列為後續重構項目[[8]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L94-L98)。
* **任務執行模組**（如 execution/plan\_executor.py, execution/task\_queue\_manager.py 等）：這些檔案負責**任務規劃與執行**（PlanExecutor、任務佇列管理等），目前隸屬核心模組。問題在於其職責偏向具體任務執行（業務邏輯）而非AI決策，本質上屬於**功能層**。建議視專案架構考量，將此類任務控制器移至“功能”模組，以明確區隔 **AI決策**（核心）與 **攻擊行動執行**（功能）職責。如果仍保留在核心，則可透過子套件劃分（如 core.execution）並加強文檔註明其功能性角色。
* **備份與冗餘檔案**：核心模組中殘留多個備份/舊檔案，例如 .backup 檔及整個 ai\_engine\_backup/ 資料夾[[9]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/CLEANUP_PLAN.md#L11-L19)[[10]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/CLEANUP_PLAN.md#L21-L29)。這些檔案增加了結構混亂的風險。建議**清理廢棄檔案**：已制定的清理計劃應予以執行，移除備份檔並驗證主要功能正常[[11]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/CLEANUP_PLAN.md#L46-L54)。清理後，核心目錄更整潔，減少開發人員混淆。

## 通用（Common）

* **AIVA 共用模組**（services/aiva\_common/）：此模組提供資料結構定義、通用工具、Logger、配置等基礎功能，應保持獨立且無跨層相依。先前版本使用單一巨型檔案定義所有 Schema 導致維護困難，例如原始 schemas.py 包含126個類別[[12]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L8-L16)。目前已將其拆分進入 schemas/ 資料夾下12個子模組[[13]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L13-L21)。**建議**：確保 **單一真實來源的 Schema** 策略落實到位：移除舊的 schemas.py、ai\_schemas.py 及相關備份檔[[14]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L70-L78)，改由 aiva\_common.schemas 資料夾統一管理所有資料模型（需在 \_\_init\_\_.py 匯出全部類別以供引用[[15]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L37-L46)[[16]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L68-L76)）。如此各服務均可從共用模組導入資料結構，維持結構一致[[17]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/README.md#L40-L46)。
* **列舉與常量定義**（enums.py）：目前所有枚舉定義集中在單一檔案（大小約12KB）[[18]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L78-L80)。隨著專案擴大，這將變成潛在維護痛點。建議將 **enums.py** 也**拆分**為 enums/ 資料夾，按功能領域（例如漏洞嚴重級別、掃描模式等）分拆枚舉定義檔案，類似 Schema 的模組化管理，提升清晰度。
* **通用工具函式**（utils/ 子目錄）：提供日志、ID生成、去重、網路重試等基礎功能。需注意**依賴方向**：通用模組應該被其它層依賴，而非反向依賴核心或功能實作。建議審視 aiva\_common 中的工具函式確保未引用核心專有類別。如發現環狀相依或不合理耦合，應透過依賴注入或重構來解耦，維持共用工具的獨立性和可重用性。
* **全域配置管理**（aiva\_common.config）：AIVA 各服務可能各自有配置檔，如核心存儲配置[[19]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L178-L181)、掃描設定中心[[20]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L432-L440)、SQLi 模組的 config.py[[21]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L288-L296)等。這可能導致配置片段分散與重複。建議在 **通用模組集中管理** 跨服務的共通配置（例如資料庫連線、訊息佇列等），並將服務特殊配置保留在各自模組。透過 README 或開發文檔註明配置繼承關係，讓開發者明瞭哪些設定來自共用模組，哪些在本地覆蓋。
* **文件與註釋**：通用模組承載了關鍵的基礎代碼，應確保**說明完善**。建議在 aiva\_common/ 資料夾新增簡要說明（README或在 \_\_init\_\_.py 加上模組描述），說明其內子模組用途。所有公共函式與類別保持清晰的 docstring，目前專案 docstring 覆蓋率已約90%[[22]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/README.md#L98-L101)應持續維持。特別對外部使用的接口（例如 aiva\_common.schemas 匯出的模型）應提供說明或 Quick Reference 方便查詢。

## 功能（Features）

* **功能模組目錄**（原 services/function/）：建議將該目錄**重新命名**為 services/features/，更直觀對應“功能模組”。此層級包含各種具體**漏洞檢測與攻擊策略**實現，如 IDOR、SQLi、XSS、SSRF、後滲透測試等。調整命名有助於新人快速理解此目錄存放的是系統各項安全測試功能模組。
* **模組結構一致性**：目前不同功能模組的封裝深度不一，存在結構不統一現象。例如 IDOR/XSS/SQLi 等採用了雙層封裝（function\_idor/aiva\_func\_idor/ 等）[[23]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L244-L251), 而 Post-Exploitation 模組則直接在目錄下定義檔案[[24]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L252-L260)。這可能源於歷史原因，但對維護者造成困惑。建議**統一模組目錄結構**：可以將功能實現在單層目錄下（如 features/idor/ 直接包含代碼檔案），避免冗餘的 aiva\_func\_xxx 子資料夾。若基於命名空間需要保留雙層，則所有功能模組皆應採用一致模式。統一後的結構有利於自動化腳本和導入語句的簡化。
* **重複與冗餘實作**：部分功能模組內存在**重複的結構或檔案**。例如：
* **SQLi 模組**同時存在新版 worker.py 和舊版 worker\_legacy.py[[25]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L294-L300)；
* **SSRF 模組**有改良版 enhanced\_worker.py 但仍保留舊 worker.py[[26]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L302-L310)；
* **掃描模組**（Scan）內也出現 worker.py 與 worker\_refactored.py 並存[[27]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L440-L448)。

這種狀況會導致模組職責混淆和維護成本上升。**建議**：在驗證新實作穩定後，**移除舊版**或將其歸檔。例如以新 worker.py 完全取代 legacy 版本，確保每種功能僅有一套實施代碼。同時在版本管控中標記此類變更，以防回歸問題。

* **未完成或重複的功能模組**：檢查功能目錄下的模組，有些可能尚未實現或與其他模組重複。例如 function\_crypto\_go 目前看來沒有任何子檔案內容[[28]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L234-L240)（僅佔位說明“密碼學功能”）。如果該模組尚未開發，建議在 README 中註明**尚在實驗階段**，或暫時移除以避免誤用。另一例是 function\_authn\_go 和 function\_idor 可能存在重疊的暴力破解測試功能（AuthN 模組含暴力破解器[[29]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L230-L233), 而 IDOR 檢測也涉及驗證繞過）。應確認各模組職能不重複，若有重合可考慮合併或明確區分子功能邊界。
* **共用檢測邏輯模組**（services/function/common/）：其中的 detection\_config.py 和 unified\_smart\_detection\_manager.py 提供跨漏洞類型的**統一智慧檢測**設定與管理功能[[30]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/function/common/unified_smart_detection_manager.py#L1-L9)。目前其定位在功能模組內，但由於這些邏輯可能服務於多種攻擊類型，屬於共用業務邏輯。建議評估將 **UnifiedSmartDetectionManager** 提取至更高層級：如果此管理器與AI決策關聯緊密，可移至核心；如側重通用工具，可移至通用模組。無論放哪，都應確保其不依賴具體功能模組實現，以避免環狀依賴。此外，在文檔中説明其角色：例如如何為各漏洞模組提供策略（根據 SQLi 模組經驗改進全局檢測演算法等）。
* **功能模組文檔**：每個漏洞/攻擊模組應附有簡要說明。建議在 **features** 目錄下為各子模組撰寫 README.md 或在 \_\_init\_\_.py 增加模組 docstring，說明：
* 此模組實現的功能（例如 “XSS 檢測”，支持何種 XSS 攻擊類型），
* 內部主要類別/函數作用，
* 與核心或掃描層互動方式（例如透過何種資料結構接收輸入/輸出結果）。  
  這將方便團隊成員迅速瞭解模組職責，亦與架構文檔對應，形成完整的模組級 API 說明。

## 掃描（Scanning）

* **掃描模組總覽**（services/scan/）：掃描層主要負責各類**漏洞掃描引擎**，包括 Web 動態掃描、資訊蒐集等，並將結果提供給核心或整合層。當前結構包含 Python 的 aiva\_scan（核心掃描協調器）、TypeScript 的 aiva\_scan\_node（增強型動態掃描服務），以及 Rust 的 info\_gatherer\_rust（如祕密偵測器）[[31]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L458-L466)。各組件語言不同但功能互補，需要明確界定介面與資料流。
* **動態掃描協調器**（aiva\_scan 模組）：負責核心爬蟲與動態分析。例如 core\_crawling\_engine/ 涵蓋HTTP爬行、URL隊列管理[[32]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L412-L420)，dynamic\_engine/ 涵蓋瀏覽器互動模擬[[33]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L417-L425)，還有掃描策略控制 (strategy\_controller.py) 和掃描編排 (scan\_orchestrator.py) 等。一旦某檔案過於龐大或涵蓋邏輯過多，建議進一步細分。比如將 **爬蟲**、**JavaScript分析**、**指紋蒐集**等不同職能拆成模組，讓 scan\_orchestrator 僅協調高層流程。這可提升掃描模組內部的單一職責性。
* **範例與備份檔**：掃描模組內含有若干範例或備用程式碼，容易造成混淆。例如 dynamic\_engine 文件夾中帶有多個 example\_\*.py（示範用途）[[34]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L419-L427)；同時 aiva\_scan 根目錄下 worker.py 與 worker\_refactored.py 並存[[27]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L440-L448)，顯示舊新兩套worker實作同時存在。**建議**：將 **範例程式**移至專門的 examples/ 目錄或明確標註為教學用途，避免誤與正式代碼混淆。對於**重構後冗餘**的檔案，應在確認新版本功能完善後**刪除舊檔**。如[[27]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L440-L448)所示，同一模組不應長期保留兩個 worker 版本。
* **多語言掃描子模組**：AIVA 掃描層利用多語言提升性能與覆蓋率：
* **Node.js 動態掃描**（aiva\_scan\_node）：實現高併發的動態掃描功能，包含網路攔截、瀏覽模擬等服務[[35]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L449-L457)。建議檢查 TS 實現與 Python 核心協同是否順暢，例如其 Logger[[36]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L452-L457)輸出格式可與主系統統一，錯誤處理和結果輸出需契合 Python 端預期。
* **Rust 資訊蒐集**（info\_gatherer\_rust）：包括 Git祕密掃描、歷史記錄分析等[[37]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L458-L463)。應確保其輸出透過預定接口（如標準輸出文件或訊息佇列）被上層接收，而非直接與資料庫耦合。為每個外部語言組件撰寫接口說明文件，闡述如何部署與整合（如輸出JSON格式結果、由哪個模組監聽等）。
* **掃描 vs. 功能 劃分**：注意區分“掃描引擎模組”與“具體漏洞邏輯模組”。目前 SAST、SCA、CSPM 等掃描類功能被歸入 services/function 中（如 function\_sast\_rust, function\_sca\_go, function\_cspm\_go 等）[[38]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L266-L275)。這在分類上略顯混亂。**建議**：可考慮將這些**純掃描型服務**歸入掃描層統一管理。例如在 services/scan/ 下設置子目錄 sast/、sca/、cspm/ 等，對應各靜態/組成分析引擎，使結構體現掃描類型分類。如果維持現有放置，也應在開發文檔中説明：**「Scan 模組」**處理通用掃描流程和動態爬網，**「Features 模組」**下的 SAST/SCA 等則是獨立掃描微服務。清晰註明它們與核心互動方式（如透過 RabbitMQ 傳遞掃描結果等），避免開發者誤以為某些掃描功能缺失。
* **掃描結果整合介面**：掃描層的最終產出須方便地提供給核心/整合層使用。當前實作中，ScanCompletedPayload 等資料模型由共用模組提供統一格式[[39]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/core/aiva_core/processing/scan_result_processor.py#L13-L21)並在核心處理流程中使用，這是良好做法。後續建議繼續遵循**事件驅動或資料契約**的方式對接：例如掃描模組完成即透過 **消息總線/資料庫** 投遞結果，由核心的 ScanResultProcessor 等組件接收並展開後續動作[[40]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/core/aiva_core/processing/scan_result_processor.py#L34-L43)。避免掃描模組直接呼叫核心函數，以降低耦合。若目前有直接依賴，應引入中介（如接口類或消息機制）解耦，使掃描引擎可獨立測試和演進。

## 整合（Integration）

* **整合模組職責**（services/integration/aiva\_integration/）：整合層負責**彙總各掃描與AI結果**，進行高階分析並對外提供接口（API/UI）。內部分為多個子模組，如分析、攻擊路徑、修復建議、報告產生等[[41]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L376-L385)[[42]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L390-L398)。這些組件構成完整的安全測試結果管線整合器。
* **風險評估分析**（analysis/ 子模組）：包含風險評估引擎（一般版與增強版）[[43]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L340-L348)、規範符合性檢查等。注意到**核心層**也實作了 risk\_assessment\_engine.py 進行初步風險分析[[44]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L120-L125)。目前在整合層再次實現增強版，可能出現職責重疊。建議明確區分兩者：**核心風評引擎**可著眼於單次掃描任務的即時風險判斷，而**整合風評引擎**負責全局性、多次掃描結果的綜合風險評分。如果核心版本已不再需要，應將其移除或遷移至整合層，統一由整合端進行風險計算，以減少重複維護[[43]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L340-L348)[[44]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L120-L125)。
* **攻擊路徑與相依關係**（attack\_path\_analyzer/）：此模組透過圖算法分析多個漏洞之間的關聯路徑，找出複合攻擊途徑[[45]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L344-L351)。它與分析模組協同作用，最終產生給使用者的高階報告。建議在整合層內部進一步**明確模組邊界**：例如攻擊路徑分析應主要消費來自風險評估和掃描發現的輸入數據，不應直接依賴核心或功能模組內部類別。可透過在報表中附圖或 JSON 結構的方式，將路徑分析結果與風險分數、漏洞列表相關聯，形成整體視圖。
* **修復建議與報告生成**（remediation/, reporting/ 子模組）：負責將技術結果轉化為**行動建議**和**管理報告**。如 patch\_generator.py 自動產生修補程式碼、report\_content\_generator.py 組裝報表內容[[41]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L376-L385)。這部分需關注與功能模組的分工：功能模組不應各自輸出報告或建議，而由整合層統一處理。建議編寫**模組級文件**說明：哪些漏洞類型會生成哪些預定格式的報告段落，修補建議如何制定（例如依據 CWE 資料庫或既有模式），以及如何增添新類型的報告模板[[46]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L382-L389)。同時確保報告模組可以方便地插入新的格式（比如新增報表匯出為PDF等功能）而不影響核心流程。
* **安全與中介服務**：整合層包含 security/auth.py（身份驗證）以及 middlewares/rate\_limiter.py 等，主要服務於整合層對外的API介面保護[[47]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L358-L366)[[48]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L386-L389)。這些模組應保持**低耦合**且易於在其他服務複用。例如 RateLimiter中若沒有依賴整合層特有狀態，或許可以上移到通用模組，供不同服務共用。同時，身份驗證機制需與整個AIVA體系一致，建議整合層在README或開發指南中註明採用的認證策略（如JWT、API Key等），以利於後續整合新的介面客戶端。
* **對外介面與入口**：整合層提供系統入口點（如 app.py）及對外API服務。api\_gateway/app.py 是REST API介面的啟動點[[49]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L402-L409)。建議為這部分撰寫**使用說明**：在 README 中描述如何啟動整合服務、可用的API端點及其功能，還有與UI的關係（例如 Web UI 是否直接調整呼叫整合層API）。入口腳本應精簡，只負責初始化各子模組並運行服務，把業務邏輯留在模組內部。若有管道整合器（pipeline integrator）的概念，也可在架構文檔標示其所在（很可能是整合層的 app.py 結合 core/master 一起構成管道）。
* **模組相依與耦合**：需要特別注意整合層**避免反向依賴**核心或掃描內部實現。理想情況下，整合層透過資料庫或消息隊列取得其他服務結果，而**不直接 import** 其他服務的模組。然目前專案中有 integrated\_ai\_trainer.py 等腳本直接匯入了核心、掃描與整合自身的模組類別[[50]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/integration/aiva_integration/integrated_ai_trainer.py#L20-L29)[[51]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/integration/aiva_integration/integrated_ai_trainer.py#L23-L31)。這或許是特殊用途的整合測試工具，但也反映出潛在的高耦合風險。建議未來在正式路徑中，整合層**僅依賴公共接口**（如共用Schemas、消息協定），避免直接使用核心或功能層的內部類別。對於像 IntegratedTrainService 這類橫跨多模組的組合腳本，可在說明中標明其用途（測試或批次任務），並與主系統解耦運行，以免干擾日常架構。
* **文檔與維運**：整合層負責最終成果輸出，應提供**充足的文檔**方便使用和維護。建議：在 services/integration/ 下撰寫 README 說明整合模組的整體作用，各子模組（分析、攻擊路徑、報告等）的功能簡述，以及與其他層的交互方式。確保重要類別和函數都有詳細 docstring，例如 AIOperationRecorder, SystemPerformanceMonitor 等應説明其記錄的操作或監控的指標。最後，為整合層實施足夠的測試，模擬整個管線流程（從掃描輸入到報告輸出）來驗證各模組協調工作[[52]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/integration/aiva_integration/integrated_ai_trainer.py#L22-L31)。

## 建議目錄結構草圖

綜合以上建議，AIVA 專案可考慮按五大模組重新組織專案結構。如採納，重構後的目錄層次或如下：

services/  
 common/ # 通用模組（原 aiva\_common：資料結構、工具、配置等）  
 core/ # 核心模組（AI 決策引擎、主控制器、學習框架）  
 ai\_engine/ # AI引擎實作（合併 bio\_neuron\_core\_v2 等）  
 training/ # 強化學習訓練相關  
 execution/ # 執行與任務調度（PlanExecutor 等，可視需要移出至 features）  
 ... (其他核心子模組，如 messaging, storage, rag 等)  
 features/ # 功能模組（原 function/：各具體漏洞檢測與攻擊策略）  
 idor/ # IDOR 檢測模組（提取自原 function\_idor）  
 sqli/ # SQLi 檢測模組  
 xss/ # XSS 檢測模組  
 post\_exploitation/ # 後滲透測試模組  
 ... (其他功能，如 SSRF、SCA、CSPM、AuthN/AuthZ 等)  
 scan/ # 掃描引擎模組（各類型安全掃描）  
 web\_scan/ # Web動態掃描 (原 aiva\_scan Python + Node 子模組)  
 sast/ # 靜態應用安全測試 (原 function\_sast\_rust)  
 sca/ # 軟體組成分析 (原 function\_sca\_go)  
 cspm/ # 雲安態勢管理 (原 function\_cspm\_go)  
 ... (其他掃描引擎，如 info\_gatherer\_rust 等)  
 integration/ # 整合模組（入口、管線整合、結果分析與報告）  
 analysis/ # 綜合分析（風險評估、攻擊路徑等）  
 reporting/ # 報告產生與匯出  
 remediation/ # 修復建議生成  
 api/ # API 介面（原 api\_gateway）  
 ... (其他整合子模組，如 observability, security 等)

*說明：*以上為概念性的新結構，重構時應逐步遷移並驗證各部分相依關係。例如更名 function/ 目錄為 features/ 時，需要同步更新 import 路徑；拆分模組需確保單元測試覆蓋。重構完成後，務必更新相關文件（架構說明、開發指南、README等），使專案文檔與實際程式碼結構相符，利於未來維護與擴充[[53]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L87-L96)。通過模組劃分清晰、命名一致的架構，AIVA 平台將更具可讀性、可擴展性，有助於團隊協作開發和長期演進。[[54]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/README.md#L52-L56)[[55]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L70-L79)

[[1]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L11-L19) [[2]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L33-L40) [[3]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L7-L14) [[4]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L41-L48) [[5]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L14-L17) [[6]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L50-L53) [[7]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L22-L28) [[8]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L94-L98) [[53]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L87-L96) [[55]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md#L70-L79) core\_optimization\_summary.md

<https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/core_optimization_summary.md>

[[9]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/CLEANUP_PLAN.md#L11-L19) [[10]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/CLEANUP_PLAN.md#L21-L29) [[11]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/CLEANUP_PLAN.md#L46-L54) CLEANUP\_PLAN.md

<https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/CLEANUP_PLAN.md>

[[12]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L8-L16) [[13]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L13-L21) [[14]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L70-L78) [[15]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L37-L46) [[16]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L68-L76) [[18]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md#L78-L80) SCHEMA\_MIGRATION\_ANALYSIS.md

<https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/_out/SCHEMA_MIGRATION_ANALYSIS.md>

[[17]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/README.md#L40-L46) [[22]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/README.md#L98-L101) [[54]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/README.md#L52-L56) README.md

<https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/README.md>

[[19]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L178-L181) [[20]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L432-L440) [[21]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L288-L296) [[23]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L244-L251) [[24]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L252-L260) [[25]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L294-L300) [[26]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L302-L310) [[27]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L440-L448) [[28]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L234-L240) [[29]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L230-L233) [[31]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L458-L466) [[32]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L412-L420) [[33]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L417-L425) [[34]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L419-L427) [[35]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L449-L457) [[36]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L452-L457) [[37]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L458-L463) [[38]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L266-L275) [[41]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L376-L385) [[42]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L390-L398) [[43]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L340-L348) [[44]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L120-L125) [[45]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L344-L351) [[46]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L382-L389) [[47]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L358-L366) [[48]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L386-L389) [[49]](https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt#L402-L409) project\_tree\_latest.txt

<https://github.com/kyle0527/AIVA/blob/5377bb93a2c4b982beb3e8e9e06f3bb0b1e0e0cf/_out/project_tree_latest.txt>

[[30]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/function/common/unified_smart_detection_manager.py#L1-L9) unified\_smart\_detection\_manager.py

<https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/function/common/unified_smart_detection_manager.py>

[[39]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/core/aiva_core/processing/scan_result_processor.py#L13-L21) [[40]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/core/aiva_core/processing/scan_result_processor.py#L34-L43) scan\_result\_processor.py

<https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/core/aiva_core/processing/scan_result_processor.py>

[[50]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/integration/aiva_integration/integrated_ai_trainer.py#L20-L29) [[51]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/integration/aiva_integration/integrated_ai_trainer.py#L23-L31) [[52]](https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/integration/aiva_integration/integrated_ai_trainer.py#L22-L31) integrated\_ai\_trainer.py

<https://github.com/kyle0527/AIVA/blob/bd3230606833d363bb4f4b9c23a1a1e4cfd4da33/services/integration/aiva_integration/integrated_ai_trainer.py>