LLM 代理人基礎的 CVE 自動化：嘗試與反思

江佾陞 國立臺灣科技大學
吳昱葶 國立臺灣科技大學
周伯翰 國立臺灣科技大學
陳孟彰 中央研究院
黃意婷 國立臺灣科技大學

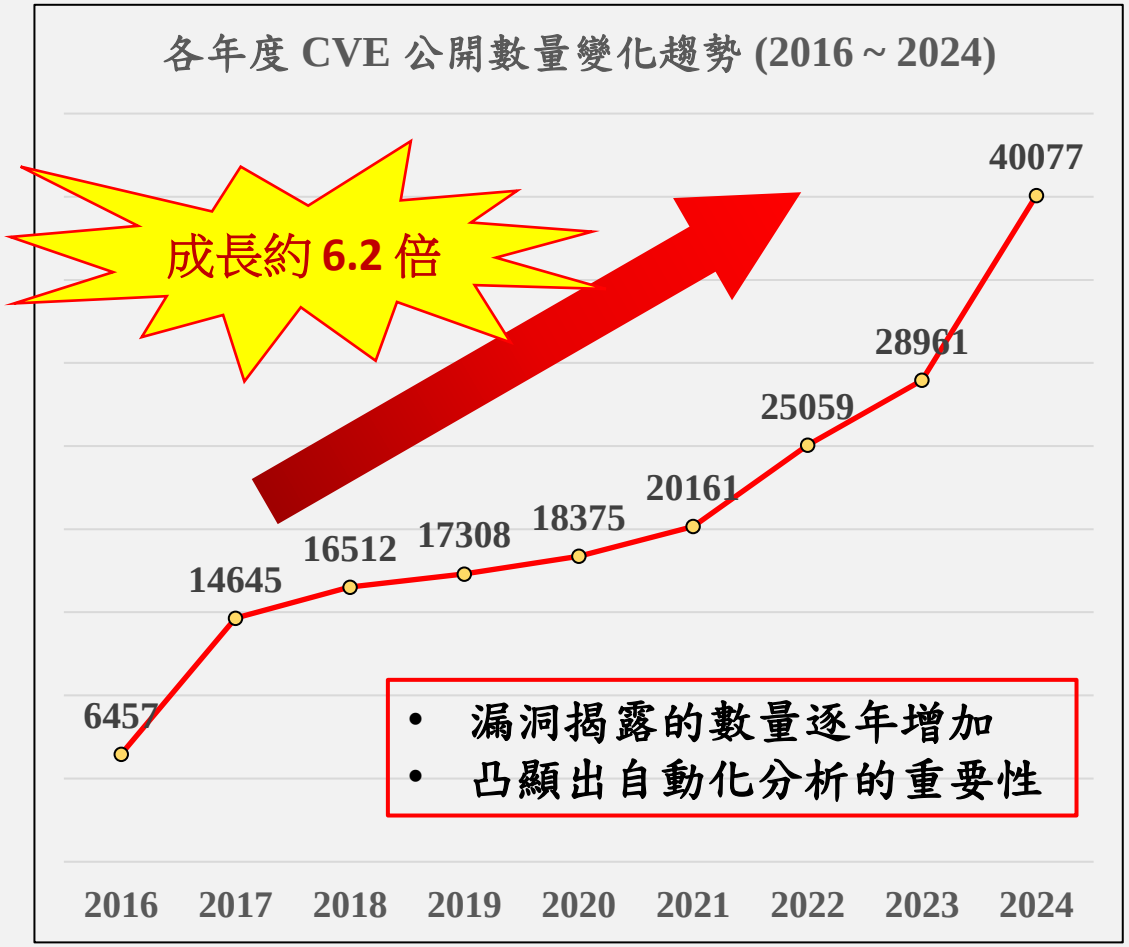
關鍵詞：Model Context Protocol，Cline，AI 代理，自動化漏洞重現，資訊安全

國立臺灣科技大學
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

前言

➢ 資訊安全的挑戰與背景：

- 近年來，隨著**數位化與網路服務的普及**，資訊安全已成為全球最受關注的議題之一現今軟體系統日益複雜，安全漏洞 (CVE, Common Vulnerabilities and Exposures) 數量持續快速增加，顯示軟體供應鏈與應用生態系正面臨前所未有的挑戰
- **漏洞修補與緩解速度遠不及發現速度**，使得攻擊者能利用尚未修補或通報延遲的弱點進行入侵，形成資安防禦上的重大落差。這樣的情況使得資安研究人員亟需新的自動化技術，以加速漏洞分析、驗證與修補流程



資料來源：CVE 官方紀錄 https://www.cve.org/About/Metrics

※ 2025 年數據統計至第 2 季，未含全年資料，已達 23710 筆

前言

➢ 研究動機：

- 隨著 LLM 在程式碼生成、工具調用與多步推理能力上的快速進展，LLM 已從「語言產生器」轉變為具備**協調複雜任務能力的核心**。這使得「**代理型系統 (Agentic Systems)**」成為可能
- 探討**代理能否從公開 CVE 描述自動生成有效的 PoC 程式碼**，並分析其可靠性、限制與潛在的雙重用途風險，**以期用於紅隊演練測試**

➢ 本研究提出的框架：

- **基於 LLM 代理，驅動一系列工具以自動化完成多步作業**
- **輸入**：欲重現的 CVE 編號、精心設計的 Prompt (提示詞)
- **輸出**：自動生成並驗證新的 PoC 程式碼

※輸入完成後，AI 代理將自行於受控隔離環境中，進行**檢索並整理漏洞資訊、技術細節**，整個任務執行過程不依賴既有 PoC (Proof-of-Concept) 程式碼

文獻探討

➢ 自動化漏洞利用：

- **AEG (Automated Exploit Generation)** 是資訊安全領域的一個重要研究方向，旨在透過程式分析與自動化工具，自動生成漏洞攻擊
- Avgerinos 等人提出的研究 [1]：首次展示能於有原始碼時自動產生針對真實應用的 exploit
- Cha 等人提出的 Mayhem 系統 [2]：將 AEG 擴展至二進位程式層級，自動化能力更進一步
- DARPA (Defense Advanced Research Projects Agency) CGC (Cyber Grand Challenge) [11,12]：展示完整「發現 → 修補 → 利用」自動化循環，代表 AEG 技術的重要里程碑
- 本研究觀察：AEG 顯示漏洞利用不再完全仰賴人工專業，但現有方法在擴展性與泛化上仍有挑戰

[1] Avgerinos, T., Cha, S. K., Rebert, A., Schwartz, E. J., Woo, M., & Brumley, D. (2014). Automatic exploit generation. Communications of the ACM, 57(2), 74-84.
[2] Cha, S. K., Avgerinos, T., Rebert, A., & Brumley, D. (2012, May). Unleashing mayhem on binary code. In 2012 IEEE Symposium on Security and Privacy (pp. 380-394). IEEE.
[3] Defense Advanced Research Projects Agency (DARPA). (2016, August 4). “Mayhem” declared preliminary winner of historic Cyber Grand Challenge. DARPA. https://www.darpa.mil/news/2016/mayhem-winner-cyber-grand-challenge
[4] Greenberg, A. (2016, August 5). Hackers don’t have to be human anymore. This bot battle proves it. Wired. https://www.wired.com/2016/08/security-bots-show-hacking-isnt-just-humans/

文獻探討

➢LLM 在資安研究中的應用：

- LLM 在資安領域的發展應用逐漸受到關注，**其不僅能輔助程式撰寫與分析漏洞， 還能在受控環境下執行攻擊任務**
- Fang 等人提出的研究 [5,6]：展示**了 LLM 代理無需事先了解漏洞即可自動駭入脆弱網站**；相同研究團隊更進一步展示**代理在獲得公開 CVE 描述後，能自動產生 one-day 漏洞的利用程式**
- Deng 等人提出的 PentestGPT [7]：評估 LLM 在自動化滲透測試中的潛力，顯示 LLM 能輔助紅隊模擬，但**仍受限於記憶、計劃能力與模型幻覺**
- SWE-agent 展示了 LLM 透過 Agent-Computer Interface，**能在完整程式庫中自主進行檔案編輯、測試執行與錯誤修正，完成端到端的維護任務** [5,6]
- 本研究觀察： LLM 在攻擊模擬上展現潛力，但同時帶來「雙重用途」風險與倫理挑戰。

[5] Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., & Press, O. (2024). Swe-agent: Agent-computer interfaces enable automated software engineering. Advances in Neural Information Processing Systems, 37, 50528-50652.
[6] SWE-agent. (2025, May 22). GitHub. https://github.com/SWE-agent/SWE-agent
[7] Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., ... & Rass, S. (2024). {PentestGPT}: Evaluating and harnessing large language models for automated penetration testing. In 33rd USENIX Security Symposium (USENIX Security 24) (pp. 847-864)

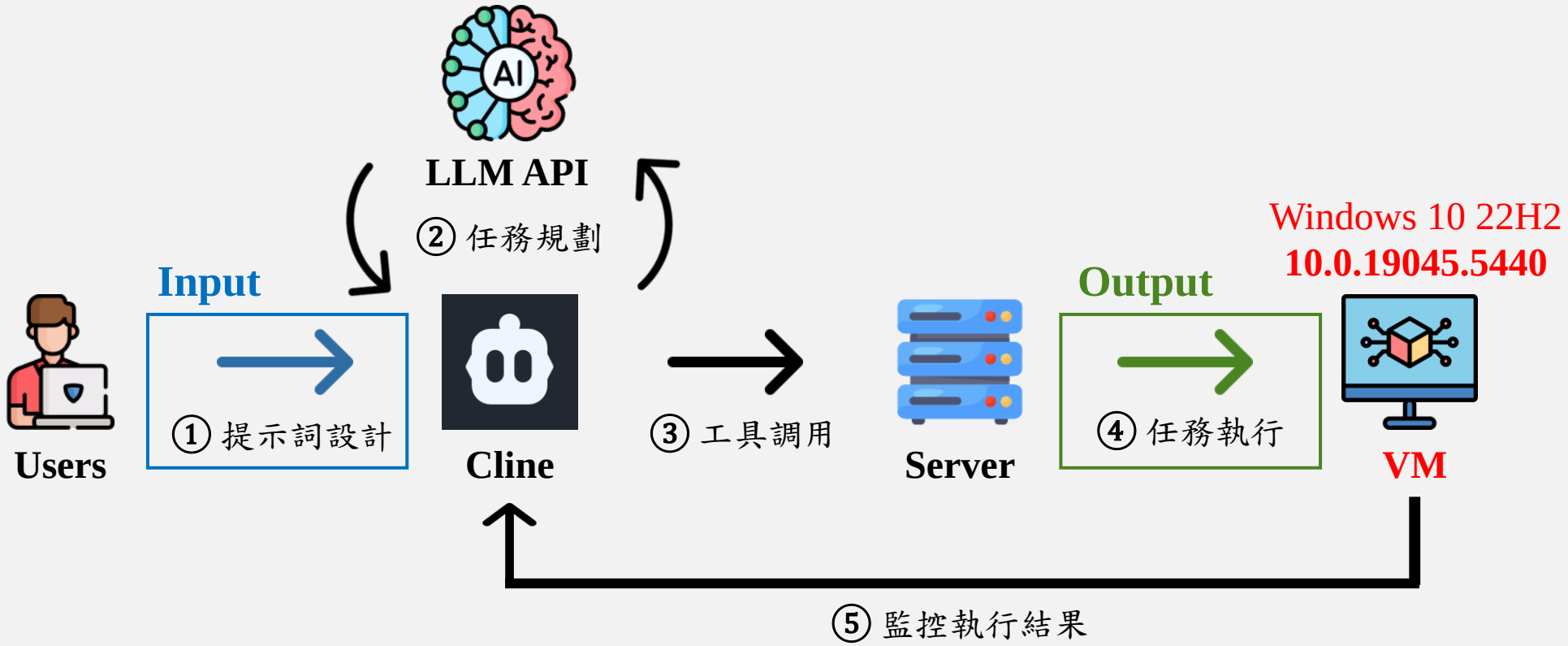
文獻探討

➢ Model Context Protocol：

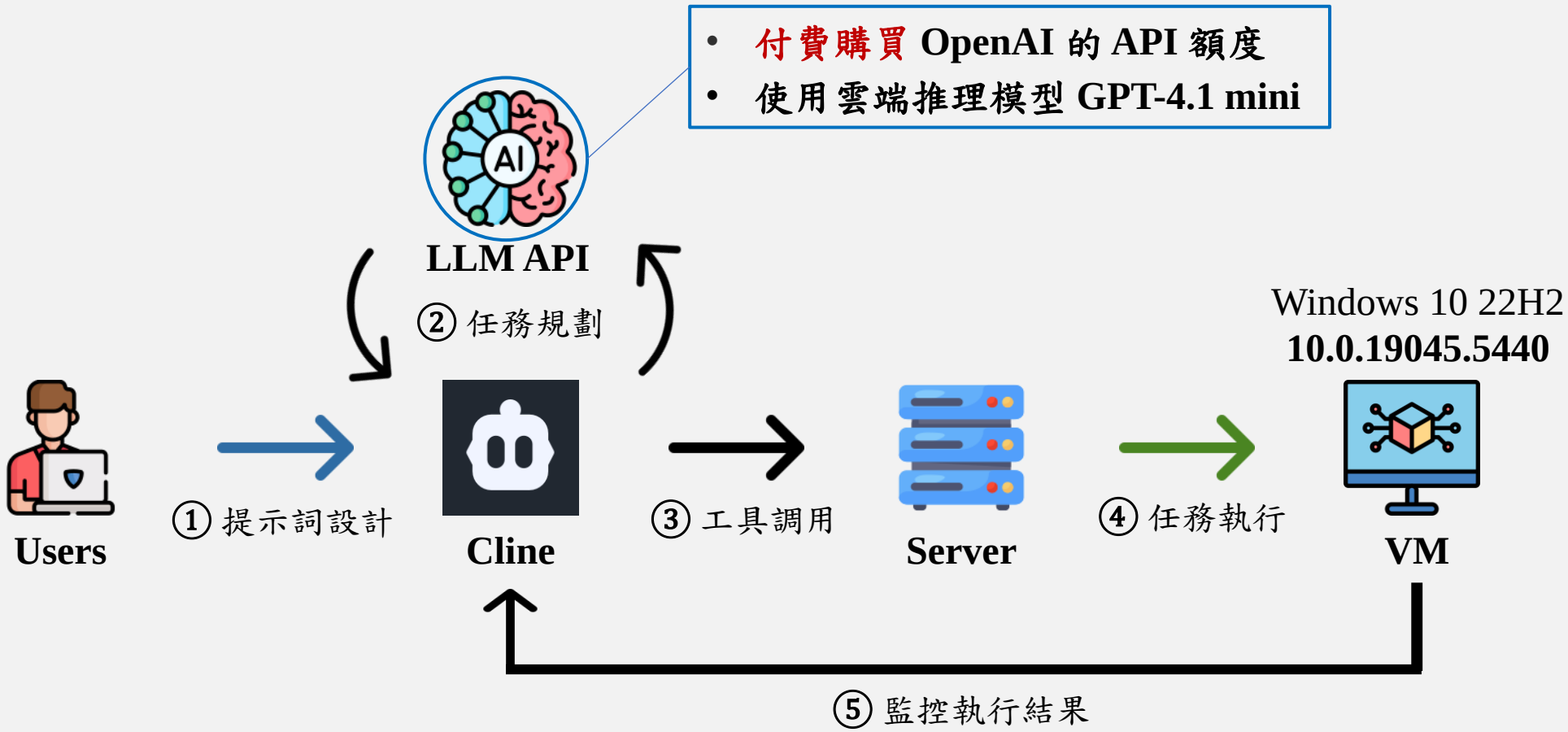
- MCP 是由 Anthropic 提出、以開放標準形式規範 LLM 與外部資源 (檔案、資料庫、工具、服務) 互動的通訊協定；**目的在於提供一個統一、安全且可互操作的介面，讓模型能在受控權限下讀取資料與呼叫工具** [8]
- MCP 使 LLM 代理能在高風險場景 (如資安測試) 中運作時降低誤用與濫用風險

[8] Anthropic. (2024, November 25). Introducing the Model Context Protocol (MCP). Anthropic.com. https://www.anthropic.com/news/model-context-protocol

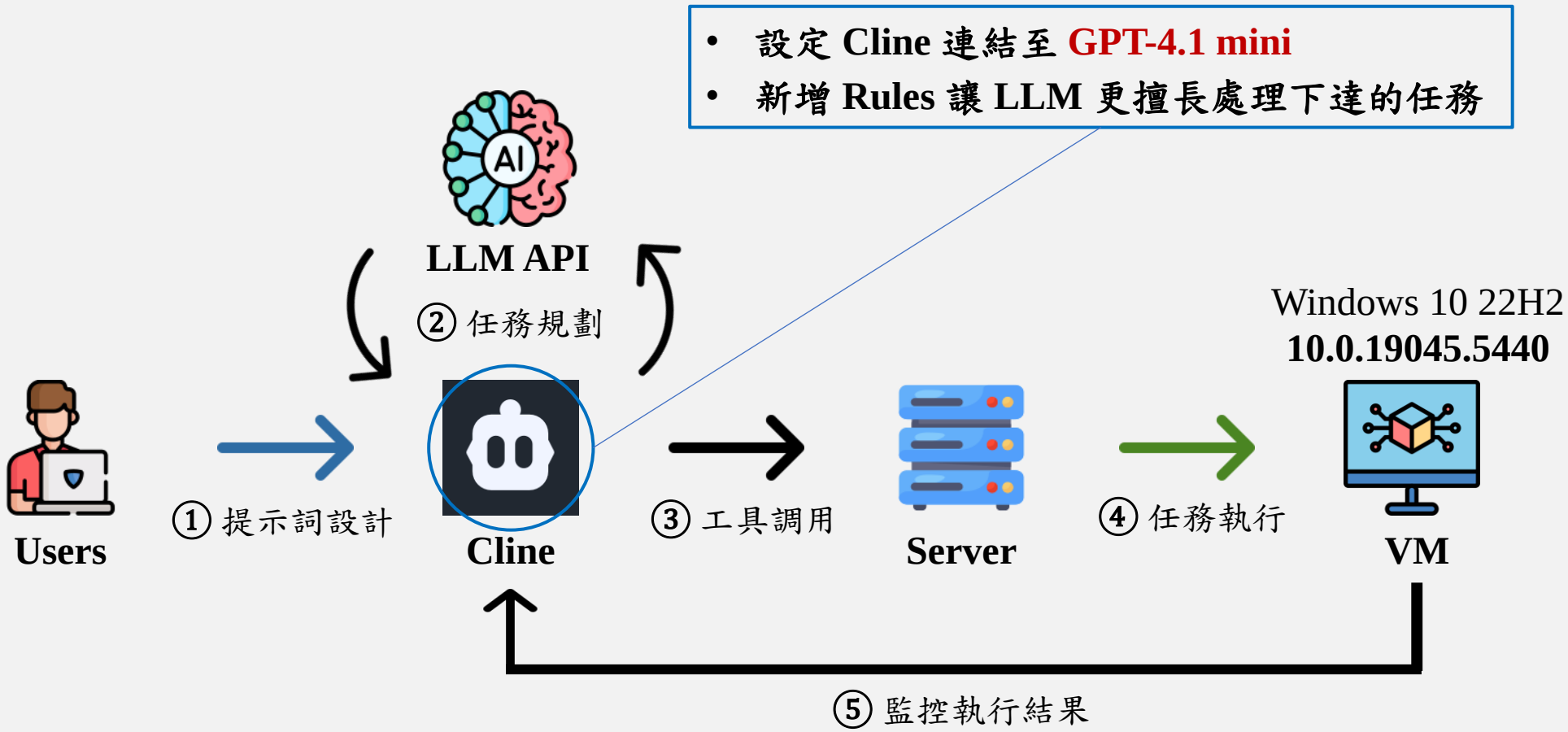
方法


AI
LLM API
② 任務規劃
Input
Windows 10 22H2
10.0.19045.5440
Output
Users
① 提示詞設計
Cline
③ 工具調用
Server
④ 任務執行
VM
⑤ 監控執行結果

方法

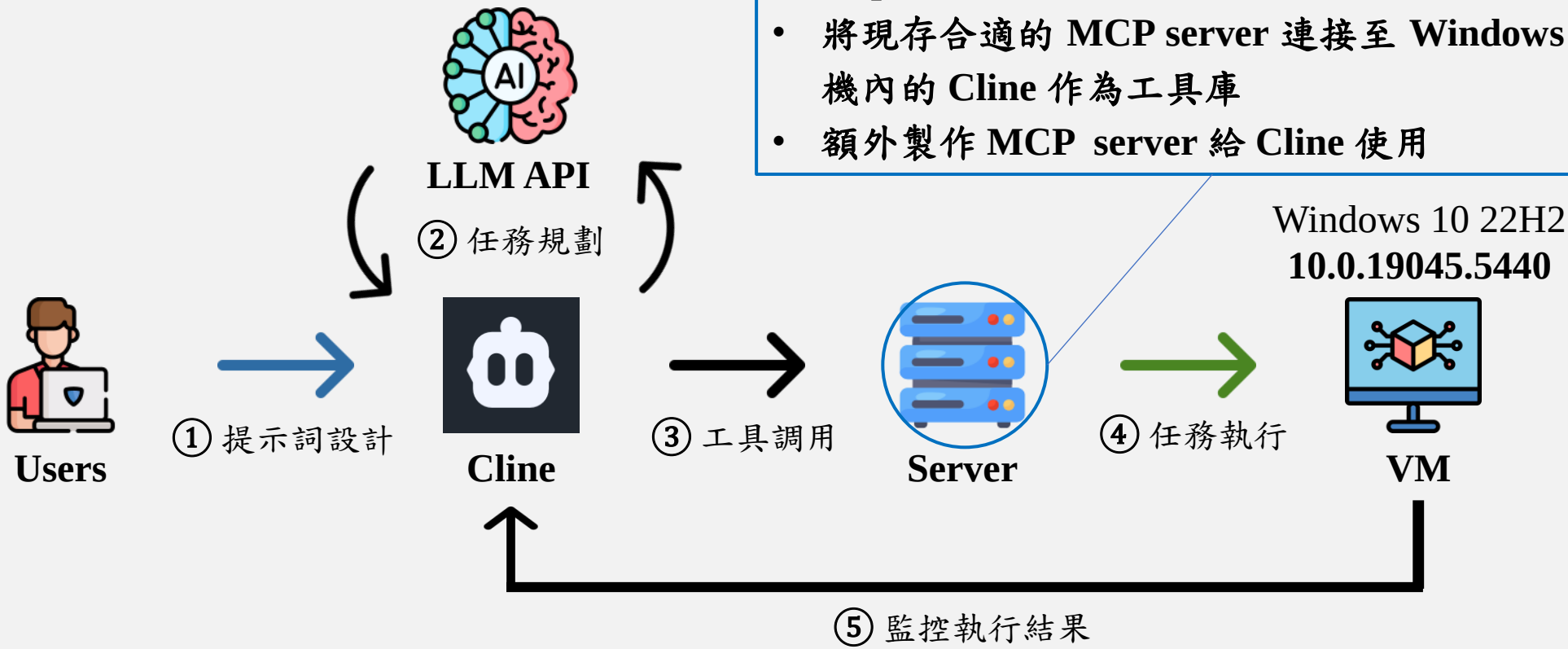

• 付費購買 OpenAI 的 API 額度
• 使用雲端推理模型 GPT-4.1 mini
AI
LLM API
② 任務規劃
Windows 10 22H2
10.0.19045.5440
Users
① 提示詞設計
Cline
③ 工具調用
Server
④ 任務執行
VM
⑤ 監控執行結果

方法

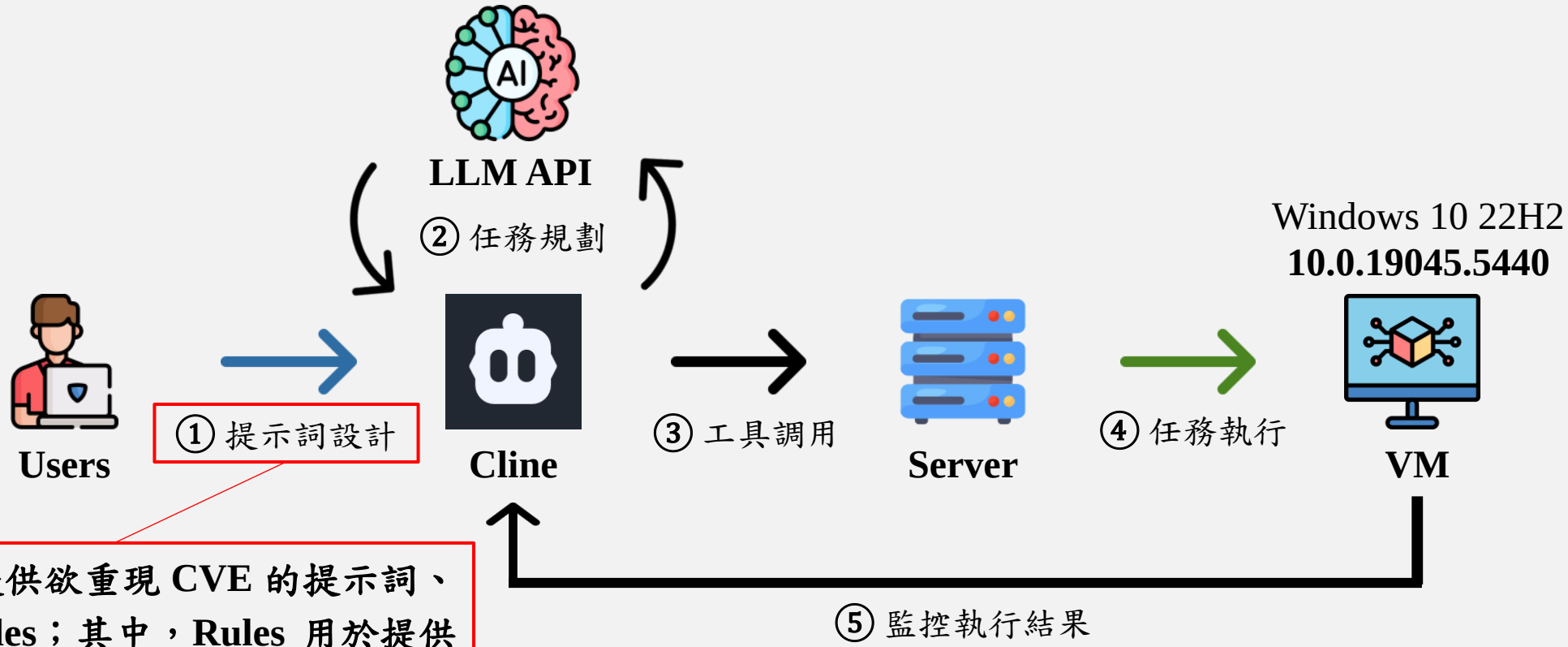

• 設定 Cline 連結至 GPT-4.1 mini
• 新增 Rules 讓 LLM 更擅長處理下達的任務
AI
LLM API
② 任務規劃
Windows 10 22H2
10.0.19045.5440
Users
① 提示詞設計
Cline
③ 工具調用
Server
④ 任務執行
VM
⑤ 監控執行結果

方法

- 在 Smithery.ai (MCP server 整合網站) 、Cline mcpmarket 上尋找合適的功能
- 將現存合適的 MCP server 連接至 Windows 虛擬機內的 Cline 作為工具庫
- 額外製作 MCP server 給 Cline 使用



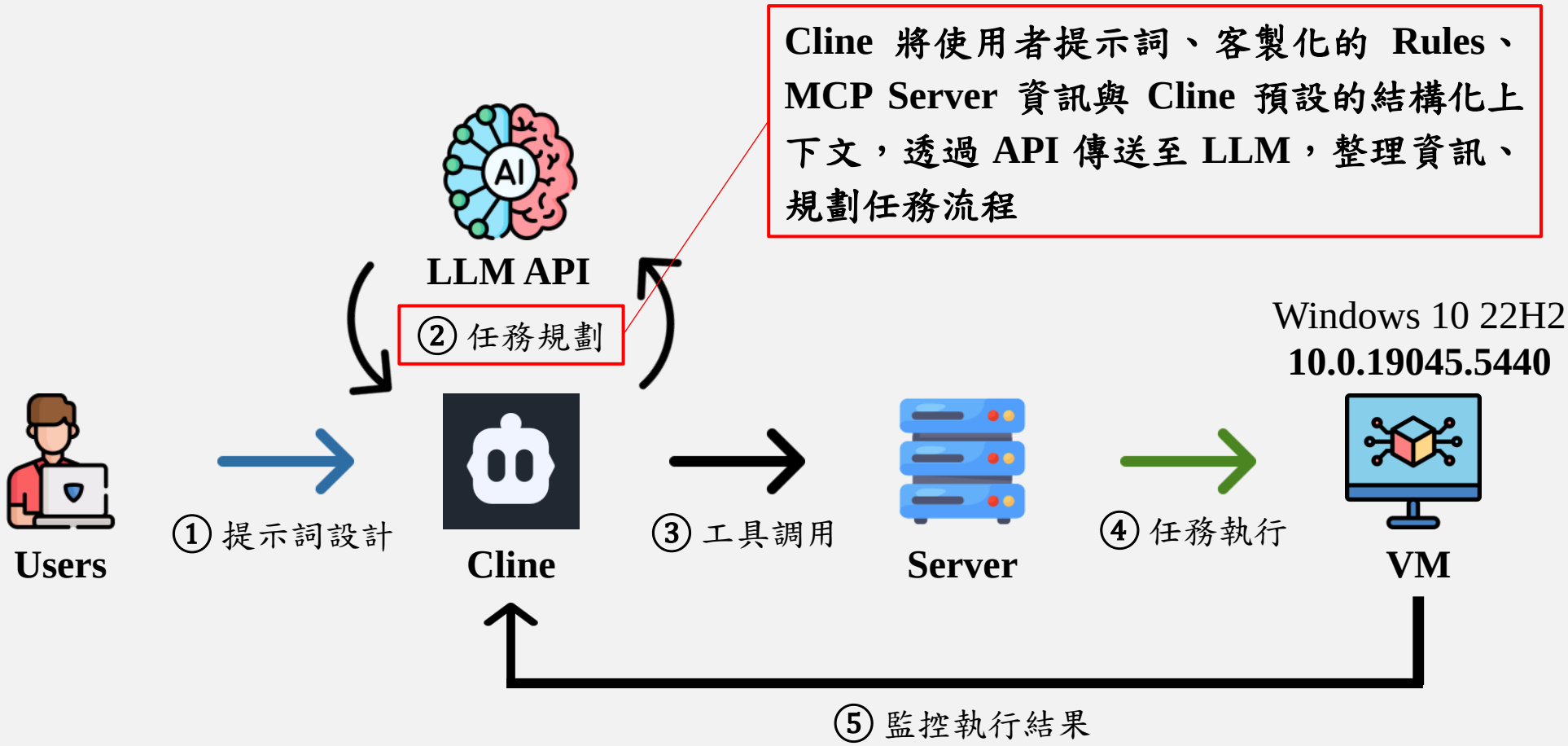
方法



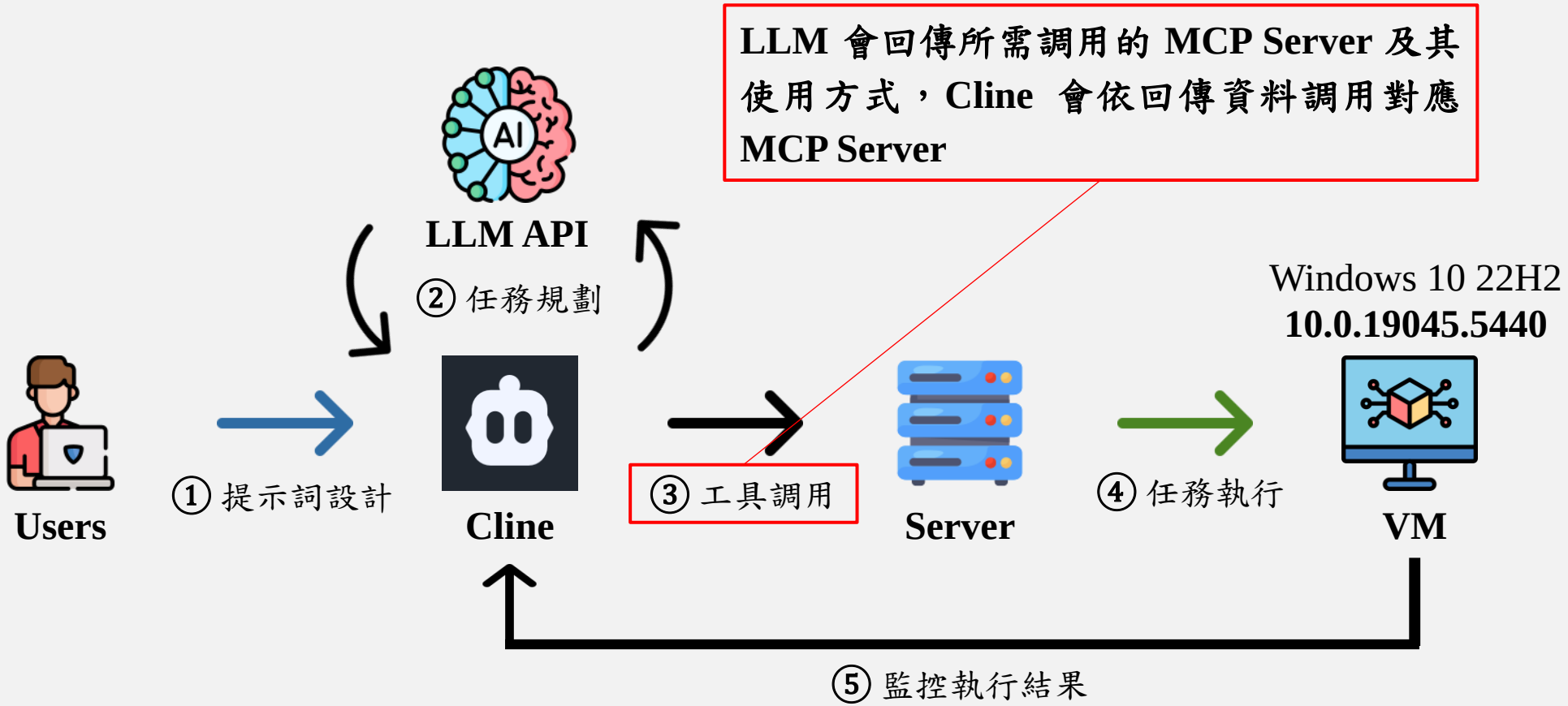
使用者首先提供欲重現 CVE 的提示詞、客製化的 Rules；其中，Rules 用於提供系統層級的行為指引與約束，並會持續輸入至與 LLM 的每一輪互動中

※ 使用者在提供提示詞後只需負責監控任務進度，並適時介入提供協助

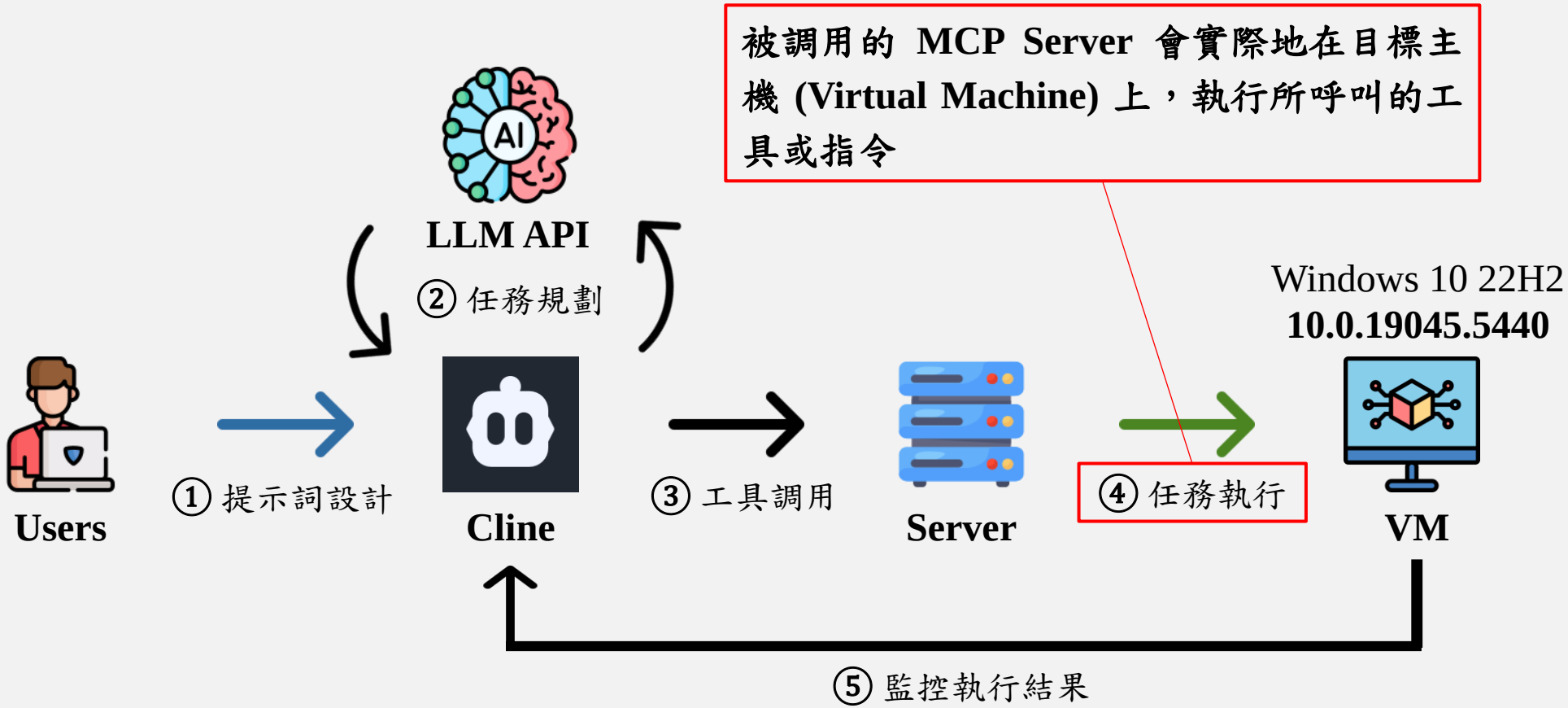
方法


Cline 將使用者提示詞、客製化的 Rules、MCP Server 資訊與 Cline 預設的結構化上下文，透過 API 傳送至 LLM，整理資訊、規劃任務流程
AI
LLM API
② 任務規劃
Windows 10 22H2
10.0.19045.5440
Users
① 提示詞設計
Cline
③ 工具調用
Server
④ 任務執行
VM
⑤ 監控執行結果

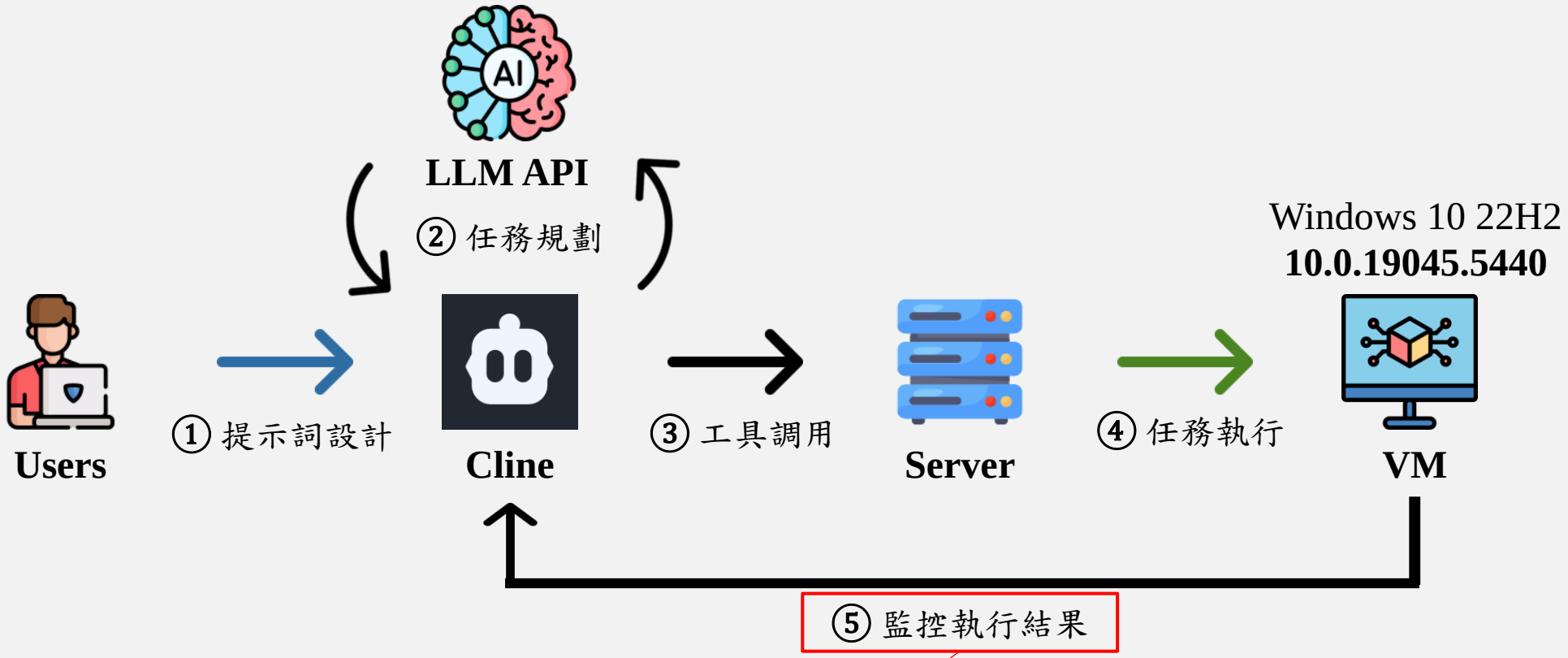
方法


LLM 會回傳所需調用的 MCP Server 及其使用方式，Cline 會依回傳資料調用對應 MCP Server
LLM API
② 任務規劃
Windows 10 22H2
10.0.19045.5440
Users
① 提示詞設計
Cline
③ 工具調用
Server
④ 任務執行
VM
⑤ 監控執行結果

方法


被調用的 MCP Server 會實際地在目標主機 (Virtual Machine) 上，執行所呼叫的工具或指令
LLM API
② 任務規劃
Windows 10 22H2
10.0.19045.5440
Users
① 提示詞設計
Cline
③ 工具調用
Server
④ 任務執行
VM
⑤ 監控執行結果

方法



Cline 會監控與回傳執行結果至 LLM 進行分析與規劃後續步驟，若 LLM 判斷已成功重現，任務即結束；反之則重複上述流程

方法

➢提示詞設計：

- 為提升 LLM 在漏洞重現任務的可靠與可控性，本研究**參考 TIDD-EC 架構 (Task Type, Instructions, Do, Don't, Example, User Content)** ，設計具備明確邏輯與限制規範的提示詞，引導 LLM 在虛擬環境中進行完整 CVE 分析與 exploit 生成過程。以下是針對每個部分進行說明：

Task Type	定義 LLM 的**角色身分與任務目標**
Instruction	列出完整操作流程順序要求，涵蓋**系統辨識、CVE 資訊蒐集、環境驗證等**
Do	規範 LLM 可執行的有效任務範圍，**避免安全審查導致拒絕撰寫程式或執行攻擊指令**
Don't	規範 LLM 易犯錯誤與抄襲等行為，**避免 LLM 使用現有 PoC** 或於動作未完成前執行下一步驟
User Content	用於明確**指定本次任務的實際操作對象與最終輸出**

方法

➢ 工具調用：

- 為了提升 AI 代理的能力，除了整合既有的 MCP Server 外，**本研究亦針對部分 server 進行微調，並自行開發數個專用 server，以擴充系統功能性與彈性**。所有**調整皆透過 MCP 官方所提供之 Inspector 工具進行驗證與測試**，確保其可被 Cline 正確調用並符合整體任務流程。各 MCP Server 的功能整理如下表所示：

Server 名稱	功能說明	程式來源
File System	檔案系統管理與取用功能	MCP 官方
Context7	提供最新文件與程式	
Visualization Charts Server	圖表產生功能	Smithery.ai
Exa Search	網路搜尋和爬蟲	
EPSS-MCP	查找 CVE 資訊	GitHub
Find-CVE-MCP	搜尋特定Windows版本 CVE	自行開發
Find-PoC-MCP	搜尋開源 PoC	

※ Find-PoC-MCP 在實際重現過程中**並未開放給 AI 代理使用**，主要用途為研究者統計資料的工具

方法

➢ 成功重現 CVE 判斷與評估設計：

- **PoC 可執行性**：由 LLM 生成或取得的 PoC 程式碼能在虛擬機內順利編譯執行；且結果能引發與該 CVE 描述相符的現象，例如系統錯誤訊息、程式崩潰、權限異常等
- **流程完整性**：即使未觸發漏洞，只要 LLM 能正確產生具體的重現步驟與程式碼，即視為達到最低限度的完成

滿足條件	評估結果
PoC 可執行性	重現成功
流程完整性	部分重現
兩項皆不滿足	重現失敗

實驗

➢ 實驗環境：

- 建置於 **Windows 10 (10.0.19045.5440) 64 位元虛擬機，唯一可以聯網的只有 MCP Server**，並安裝完整的開發工具鏈，以支援 CVE 自動化重現的流程
- 開發環境包含以下應用程式，以確保程式撰寫、版本管理與編譯執行皆能順利完成：
  1. Visual Studio Code 與其 Cline 套件 (版本 3.26.5)
  2. Python 3.13.0
  3. Git 2.50.1
  4. Node.js 22.17.1
  5. MinGW-w64 編譯器
- 為模擬實際攻擊情境中常見的防護繞過條件，**實驗期間已預先停用 Windows Defender 防護機制**，以避免干擾漏洞觸發與 exploit 行為之觀察與驗證

實驗

➢ Case Study：

- 由於 Windows 系統漏洞眾多，若無具體參考資訊，將無法確認 CVE 是否能在環境中被觸發。因此，**本研究挑選具備公開 PoC 的項目，以提升效率與成功率**
- 依據以下三項規則進行過濾、檢查：
  1. **版本相符性**：僅挑選影響 Windows 10 Build Version 10.0.19045.5440，且至少具備一份公開 PoC 的漏洞
  2. **可獨立執行性**：優先選擇可於單一裝置本地執行，以利在隔離環境下進行完整重現
  3. **真實可用性**：人工審查每個 PoC，僅納入具備 exploit 潛力的程式；若 GitHub 上存放的 PoC 實際為漏洞掃描器、分析腳本或 patch 測試程式則排除，避免干擾結果判讀
- 本研究最終從 17 個具公開 PoC 的 CVE 中，人工挑選出 5 個最具代表性且具備重現潛力的案例

實驗

➢ Case Study：

- CVE-2025-21420 案例分析如下：

簡介	失敗原因
• **漏洞類型**：DLL 側載 (side-loading) 本地提權漏洞 • **影響元件**：cleanmgr.exe (Windows Disk Cleanup Tool) • **利用原理**：當使用者或系統觸發時，該程式會以 SYSTEM 權限執行，並載入多個 DLL。然而，由於 cleanmgr.exe 在設計上未明確指定所載入 DLL 的路徑，攻擊者可利用系統的預設搜尋順序，藉由符號連結 (Symbolic Link) 將惡意 DLL 放置於使可寫入目錄中。一旦 cleanmgr.exe 誤載此惡意 DLL，即可讓惡意程式碼以 SYSTEM 權限執行	• 實驗因權限問題而失敗 • 由於建立符號連結本身就需要管理員權限，這與低權限攻擊的情境相悖，導致流程在初期便中斷 • 即使以人工方式進行相同操作也未能成功觸發惡意 DLL 的載入，最終僅顯示「權限不足」或「找不到路徑」等錯誤訊息，未能達成完整的提權

實驗

➢ Case Study：

- CVE-2025-26633 案例分析如下：

簡介	失敗原因
• **漏洞類型**：安全功能繞過 (Security Feature Bypass, SFB) 漏洞 • **影響元件**：微軟管理控制台 (Microsoft Management Console, MMC) 內的國際化路徑機制 (MUIPath) • **利用原理**：攻擊者可以將惡意 .msc 檔案放置於 en-US 目錄，使 MMC 優先載入該檔案，取代原本的合法版本。攻擊者可在這個惡意的 .msc 檔案內嵌入 PowerShell 指令或 ActiveX 控制項等元素，達到遠端命令執行或下載惡意酬載等目的	• 本次漏洞重現因建立的 .msc 檔案格式錯誤而失敗 • MMC 主控台檔案採用的是複合結構化儲存檔 (Compound Structured Storage File)，而非單純的 XML 檔案，導致自製檔案被拒絕開啟 • 研究亦發現環境變數 MUILanguage 僅影響介面語言，與 .msc 的載入流程無關 • 此漏洞觸發的關鍵在於主控台的內部結構與資源載入行為，並非單純在 .msc 檔案中插入腳本觸發，因此未能達到預期的效果

實驗

➢ Case Study：

- CVE-2025-29824 案例分析如下：

簡介	失敗原因
• **漏洞類型**：本地提權漏洞 • **影響元件**：Windows CLFS (Common Log File System) 核心驅動的 Use-After-Free • **利用原理**：攻擊者在觸發漏洞後，可覆寫目標程序的存取權杖 (access token)，從而取得 SYSTEM 等級最高權限，進而轉儲 LSASS (Local Security Authority Subsystem Service) 程序的記憶體竊取使用者憑證	• 本次重現 exploit 程式未能正確開啟一個已啟用 CLFS 功能的有效路徑而失敗，這導致核心中關鍵的 Use-After-Free 觸發點未能生效

實驗

➢ Case Study：

- CVE-2025-49667 案例分析如下：

簡介	失敗原因
• **漏洞類型**：本地提權漏洞 • **影響元件**：Windows Win32K 子系統中的 ICOMP 函數 • **利用原理**：此函數的執行流程可被觸發 Double Free 記憶體錯誤。攻擊者能透過特定 win32k.sys 系統呼叫，強制核心物件重複釋放，再利用受控資料重新分配該記憶體以覆寫核心函數指標，最終取得 SYSTEM 權限	• 漏洞重現因 LLM 的分析錯誤而失敗。它誤判漏洞需要使用者互動 • 公開的 PoC 證實其為非互動式的強制雙重釋放 • 此外，LLM 生成的程式試圖以高權限執行，此舉與漏洞需從低權限帳戶發動提權的目的相悖，邏輯上並不成立

實驗

➢CVE 重現結果總結：

- 在漏洞類型與限制條件的差異方面，**本研究選取的案例成功重現的過程遠比漏洞描述複雜，並非僅透過命令列介面即可達成**
- **重現流程層面**：AI 代理能夠提供合理的攻擊重現步驟，並能與使用者互動確認執行，顯示其在「知識性說明」與「程序性輔助」上的潛力
- **Exploit 程式層面**：LLM 難以直接生成完整且可執行的 exploit 程式，多半停留在理論層級，與實務可用的程式存在顯著落差

CVE 編號	可執行性	完整性	評估分類
CVE-2025-21420	否	是	部分重現
CVE-2025-26633	否	是	部分重現
CVE-2025-29824	否	是	部分重現
CVE-2025-48799	否	是	部分重現
CVE-2025-49667	否	是	部分重現

討論與結論

➢ 結論：

- 本研究**示範 LLM 驅動的 AI 代理在漏洞重現自動化上具備可行性**：能負責資訊蒐集、流程規劃與程序性輔助
- 在本次五個案例中，雖**能生成合理的重現步驟與程式雛形，但尚未出現完全成功的可執行 exploit**。失敗主因可歸納為：
  - 嚴格且未滿足的環境前置條件
  - **LLM 預訓練資料方面不足 (如完整且真實的 exploit 程式、系統底層操作)**

➢ 未來兩大改進方向：

1. **系統層面**：擴充 MCP server 模組，加入自動化環境檢測（Windows 版本、已安裝補丁、磁碟/權限狀態等）與修正建議，以便前置評估與調整
2. **模型層面**：對研究型 LLM 進行定向微調，導入漏洞技術分析、歷史 exploit 範例與系統 API 行為等資料，以提升生成可用 PoC 的能力

Q & A

所有實驗程式與資料已上傳至 GitHub：github.com/qAq221102/LLM-Agent-Based-CVE-Automation-Experiments-and-Reflections。

Thank you for listening!! 🙂

...

詢問任何問題