PAC-GPT: A Novel Approach to Generating Synthetic Network Traffic With GPT-3

PAC-GPT：一種使用 GPT-3 產生合成網路流量的新方法

Authors: Danial Khosh Kholgh; Panos Kostakos;

Speaker：113C53017 陳雋洋 (資安碩一)

Published in: IEEE Access (Volume: 11)
Date of Publication: 18 October 2023

TAIPEI TECH 國立臺北科技大學 National Taipei University of Technology

目錄

研究背景

全球網路攻擊的頻率不斷增加，並帶來了顯著的財務損失，這使得網路安全對於公共和私人實體的重要性日益提升。

為了應對這些威脅，組織需要處理大量的數據，然而，手動分析的規模過於龐大，在實際操作中極為不切實際。

然而，機器學習方法在網路安全領域的有效性受到**當前訓練資料集品質顯著不足**的限制，而資料集的品質是這些方法的關鍵要求。

目前大多數可用的網路安全資料集通常**資訊過時或是數量和品質上都存在缺陷**，這使得它們不足以滿足機器學習演算法的嚴苛需求。

提出的解決方案：PAC-GPT

具體而言，主要目標是**創建高品質的合成網路流量**，可用於各種任務，例如訓練入侵偵測系統 (IDS)。

因此建立了 PAC-GPT **框架**，用於基於 OpenAI 的 GPT-3 生成機器學習方法所需的可靠合成數據。

這篇論文採用了一種新穎的 LLM 鏈接技術來生成合成網路流量。
據作者所知，這種技術目前仍未被用於生成合成網路封包。

目標： 創建高品質的合成網路流量，可用於訓練入侵偵測系統 (IDS) 等各種網路安全任務

PAC-GPT 框架組成

框架的核心創新點在於使用**大型語言模型 (LLM) 連結技術**，這使得能夠生成與真實數據高度相似的合成網路流量。

- PAC-GPT 的核心組成部分是兩個模組：
  - **流量生成器 (Flow Generator)** 、**封包生成器 (Packet Generator)**

這兩個組件被整合到一個命令列介面 (CLI) 工具中，用於管理整個數據生成流程

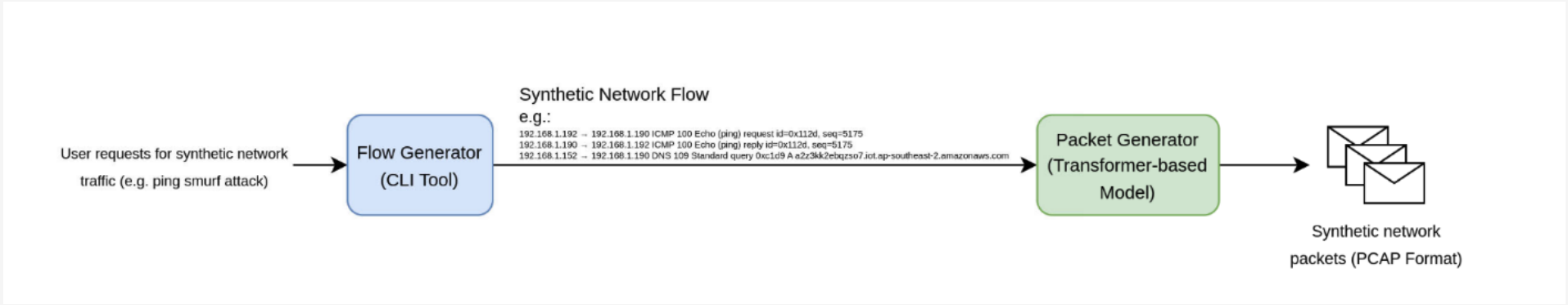


FIGURE 1. An overview of synthetic network traffic generation process.

PAC-GPT 核心架構

該基於 Transformer 的模型基於 OpenAI 的 GPT-3 模型。GPT-3 是一個先進的語言模型，在大規模文本語料庫上進行了預訓練，學習語言模式、語法、語義和一些事實知識。

它因其在零樣本或少樣本學習任務中的表現而受到關注，能夠快速適應新任務而無需大量微調。

選擇 GPT-3 的主要原因是其適應新任務的靈活性 (即少樣本學習) 。
同時也利用了其在特定任務上進行微調的能力，這通過 OpenAI API 實現，該 API 提供了不同參數大小、速度和性能的 GPT-3 模型變體。

PAC-GPT 核心架構

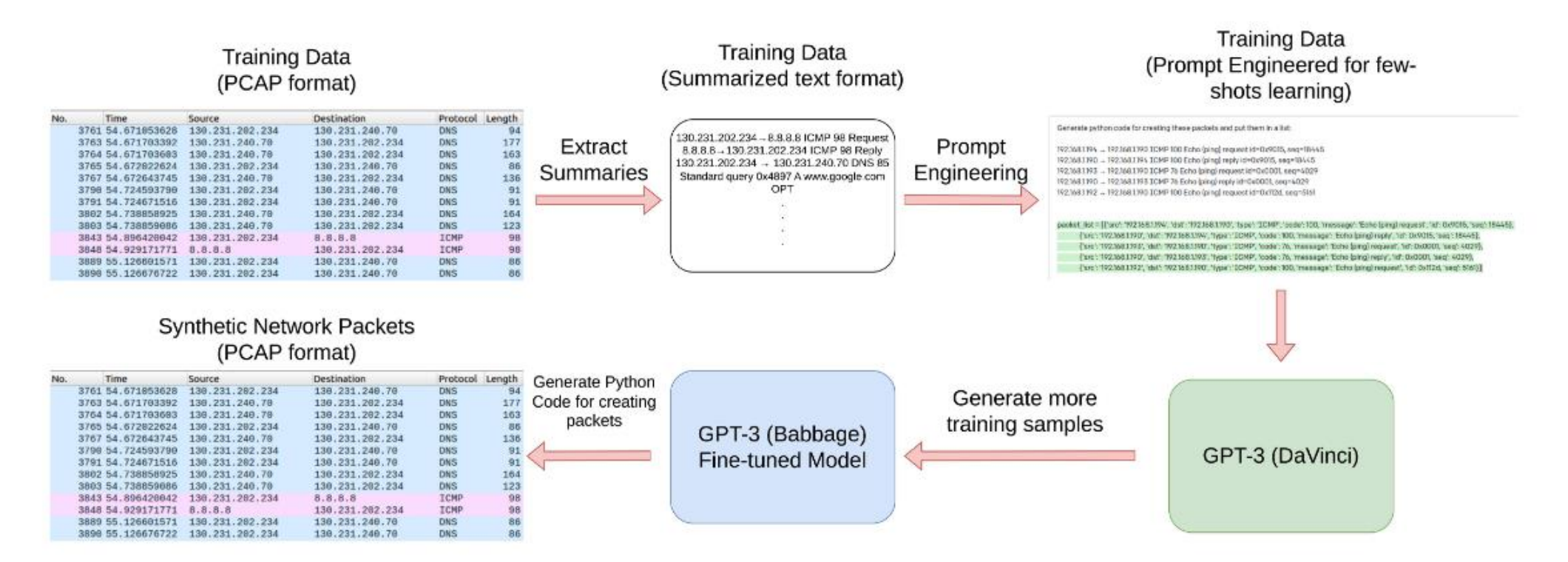
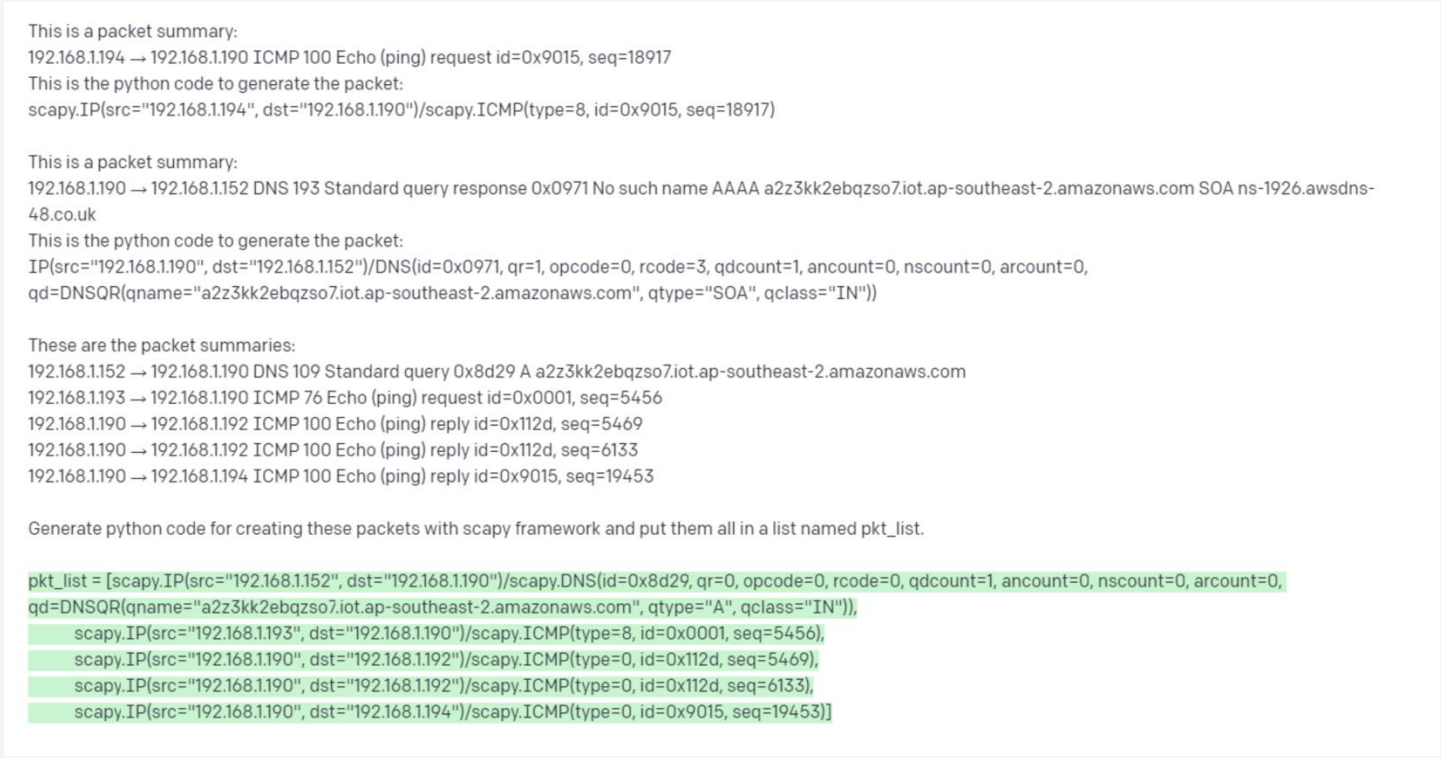


FIGURE 2. Packet generation process using GPT-3.

PAC-GPT 核心架構

FIGURE 3. Sample output of packet summarization step after pre-processing.

PAC-GPT 核心架構



This is a packet summary:
192.168.1.194 → 192.168.1.190 ICMP 100 Echo (ping) request id=0x9015, seq=18917
This is the python code to generate the packet:
scapy.IP(src="192.168.1.194", dst="192.168.1.190")/scapy.ICMP(type=8, id=0x9015, seq=18917)

This is a packet summary:
192.168.1.190 → 192.168.1.152 DNS 193 Standard query response 0x0971 No such name AAAA a2z3kk2ebqzso7.iot.ap-southeast-2.amazonaws.com SOA ns-1926.awsdns-48.co.uk
This is the python code to generate the packet:
IP(src="192.168.1.190", dst="192.168.1.152")/DNS(id=0x0971, qr=1, opcode=0, rcode=3, qdcount=1, ancount=0, nscount=0, arcount=0, qd=DNSQR(qname="a2z3kk2ebqzso7.iot.ap-southeast-2.amazonaws.com", qtype="SOA", qclass="IN"))

These are the packet summaries:
192.168.1.152 → 192.168.1.190 DNS 109 Standard query 0x8d29 A a2z3kk2ebqzso7.iot.ap-southeast-2.amazonaws.com
192.168.1.193 → 192.168.1.190 ICMP 76 Echo (ping) request id=0x0001, seq=5456
192.168.1.190 → 192.168.1.192 ICMP 100 Echo (ping) reply id=0x112d, seq=5469
192.168.1.190 → 192.168.1.192 ICMP 100 Echo (ping) reply id=0x112d, seq=6133
192.168.1.190 → 192.168.1.194 ICMP 100 Echo (ping) reply id=0x9015, seq=19453

Generate python code for creating these packets with scapy framework and put them all in a list named pkt_list.

```
pkt_list = [scapy.IP(src="192.168.1.152", dst="192.168.1.190")/scapy.DNS(id=0x8d29, qr=0, opcode=0, rcode=0, qdcount=1, ancount=0, nscount=0, arcount=0, qd=DNSQR(qname="a2z3kk2ebqzso7.iot.ap-southeast-2.amazonaws.com", qtype="A", qclass="IN")),
    scapy.IP(src="192.168.1.193", dst="192.168.1.190")/scapy.ICMP(type=8, id=0x0001, seq=5456),
    scapy.IP(src="192.168.1.190", dst="192.168.1.192")/scapy.ICMP(type=0, id=0x112d, seq=5469),
    scapy.IP(src="192.168.1.190", dst="192.168.1.192")/scapy.ICMP(type=0, id=0x112d, seq=6133),
    scapy.IP(src="192.168.1.190", dst="192.168.1.194")/scapy.ICMP(type=0, id=0x9015, seq=19453)]
```

FIGURE 4. Sample input and output of DaVinci model while used for generating more training data. The first part is the prompt/input, while the second highlighted part is generated text/output.

PAC-GPT 核心架構

```
192.168.1.190 → 192.168.1.192 DNS 222 Standard query response 0x1565 A daisy.ubuntu.com A 162.213.33.132 A 162.213.33.108 NS ns1.canonical.com NS ns3.c
anonical.com NS ns2.canonical.com A 91.189.94.173 A 91.189.95.3 A 91.189.91.139
192.168.1.190 → 192.168.1.192 ICMP 100 Echo (ping) reply id=0x112d, seq=5807
192.168.1.190 → 192.168.1.192 ICMP 100 Echo (ping) reply id=0x112d, seq=6340
192.168.1.190 → 192.168.1.192 ICMP 100 Echo (ping) reply id=0x112d, seq=6600
192.168.1.194 → 192.168.1.190 ICMP 100 Echo (ping) request id=0x9015, seq=19885



pkt_list = [scapy.IP(src="192.168.1.190", dst="192.168.1.192")/DNS(id=0x1565, qr=1, opcode=0, rcode=0, qdcount=1, ancount=5, nscount=3, arcount=0, qd=D
NSQR(qname="daisy.ubuntu.com", qtype="A", qclass="IN"), an=DNSRR(rrname="daisy.ubuntu.com", type="A", rclass="IN", ttl=172800, rdata="162.213.33.132")/
DNSRR(rrname="daisy.ubuntu.com", type="A", rclass="IN", ttl=172800, rdata="162.213.33.108")/DNSRR(rrname="daisy.ubuntu.com", type="NS", rclass="IN", tt
l=172800, rdata="ns1.canonical.com")/DNSRR(rrname="daisy.ubuntu.com", type="NS", rclass="IN", ttl=172800, rdata="ns3.canonical.com")/DNSRR(rrname="dais
y.ubuntu.com", type="NS", rclass="IN", ttl=172800, rdata="ns2.canonical.com")/DNSRR(rrname="daisy.ubuntu.com", type="A", rclass="IN", ttl=172800, rdata
="91.189.94.173")/DNSRR(rrname="daisy.ubuntu.com", type="A", rclass="IN", ttl=172800, rdata="91.189.95.3")/DNSRR(rrname="daisy.ubuntu.com", type="A", r
class="IN", ttl=172800, rdata="91.189.91.139")], scapy.IP(src="192.168.1.190", dst="192.168.1.192")/scapy.ICMP(type=0, id=0x112d, seq=5807), scapy.IP(s
rc="192.168.1.190", dst="192.168.1.192")/scapy.ICMP(type=0, id=0x112d, seq=6340), scapy.IP(src="192.168.1.190", dst="192.168.1.192")/scapy.ICMP(type=0,
id=0x112d, seq=6600), scapy.IP(src="192.168.1.194", dst="192.168.1.190")/scapy.ICMP(type=8, id=0x9015, seq=19885)]
```

FIGURE 5. Sample input and output of Babbage fine-tuned model while used for generating the packets. The top part is the input without any prompts, whereas the lower part is generated code capable of creating the packets with Python and Scapy

PAC-GPT CLI 工具功能

網路流是構成網路流量的一系列封包。描述的封包生成器僅生成單個獨立封包，無法自己創建這種流。
因此，此研究實現了一個 Python 腳本來生成封包生成器所需的網路流。

該腳本被開發為一個 CLI 工具。它不僅能夠生成流，還能處理對封包生成器模型的必要調用，並且實質上是它們的包裝器。

此外，它提供了一些實用功能，例如將生成的封包寫入 PCAP 文件格式，以及使用網路接口傳輸生成的封包。

PAC-GPT CLI 工具功能

CLI 工具提供了多個選項來創建流：

- ip_file：流生成器在設置封包中的發送方和接收方 IP 字段時使用的配置文件的路徑。
  - IPs 可以單獨指定或按子網指定。該文件也可以包含攻擊者和受害者的 IPs，以防使用特定的惡意情境。

- output_file：保存包含生成的合成封包的輸出 PCAP 文件的路徑。
- n：要生成的封包數量。
  - 這是一個近似數字，因為流生成器還會考慮其他因素如情境，並且封包生成器創建的一些封包是錯誤的需要被丟棄。

- protocols：應為其生成封包的網路協定。
- scenario：應為其生成封包的情境。選項包括 Normal network traffic, ping smurf, ping flood, 和 DNS flooding。
- replay_packets：布林選項，指示是否也應在網路上重放生成的封包。

PAC-GPT CLI 工具功能

工具中包含的惡意情境定義：

- **Normal**：指示網路中正常的流量流，沒有特定的網路攻擊。在這種模式下，基於指示的協定，流量會在網路中的所有節點之間隨機生成。

- **Ping of Death Attack**：利用 ICMP 協定的特定特性，發送大小超過最大允許值 (65,535 bytes) 的 IP 封包。通過將大型封包分段為小塊來實現，單獨符合 MTU，但重組時超過。接收系統可能發生緩衝區溢出，導致系統崩潰或未定義行為，從而達成 DoS 攻擊目的。工具處理方式：除了為整個網路生成一些正常流量封包外，還創建一些具有討論規範的惡意封包，源自指定的攻擊者節點，目標是網路中的受害者節點。

- **Ping Flood**：攻擊者向目標設備發送大量 ICMP Echo Request 封包，而不等待 Echo Reply。這種封包洪流會飽和目標的網路頻寬並消耗其處理傳入流量的資源，可能導致服務中斷。工具處理方式：除了正常網路流量外，還從攻擊者的 IP 向受害者的 IP 發送突發的 ICMP Echo Requests。

PAC-GPT CLI 工具功能

工具中包含的惡意情境定義：

- **Ping Smurf**：類似於 Ping Flood，但通過不同方法試圖超載受害者系統。攻擊者使用偽造的 ICMP Echo Request 封包，源 IP 地址被欺騙為目標受害者 IP。將此封包發送到 IP 廣播網路。網路中的主機按預期響應，發送 ICMP Echo Reply64。由於原始 ICMP 封包中的源 IP 地址被欺騙為受害者，所有回覆都會發送給受害者的 IP 地址65。這可能導致壓倒性的流量洪流。工具模仿方式：生成大量 ICMP Echo Request 封包，源 IP 欺騙為受害者，目標地址設為網路中的節點。此外，還生成 ICMP Echo Reply 封包以響應請求。

- **DNS Flood**：攻擊者旨在通過大量請求壓倒 DNS 伺服器，耗盡伺服器資源。攻擊者通常使用多個設備向目標伺服器發送高流量的 DNS 請求。這導致對合法請求的響應變慢，嚴重時 DNS 伺服器完全無響應，中斷目標網路對 Internet 的訪問。工具模擬方式：除了正常流量數據外，還從攻擊者的節點向受害者的系統發出大量 DNS 查詢請求。

PAC-GPT CLI 工具功能

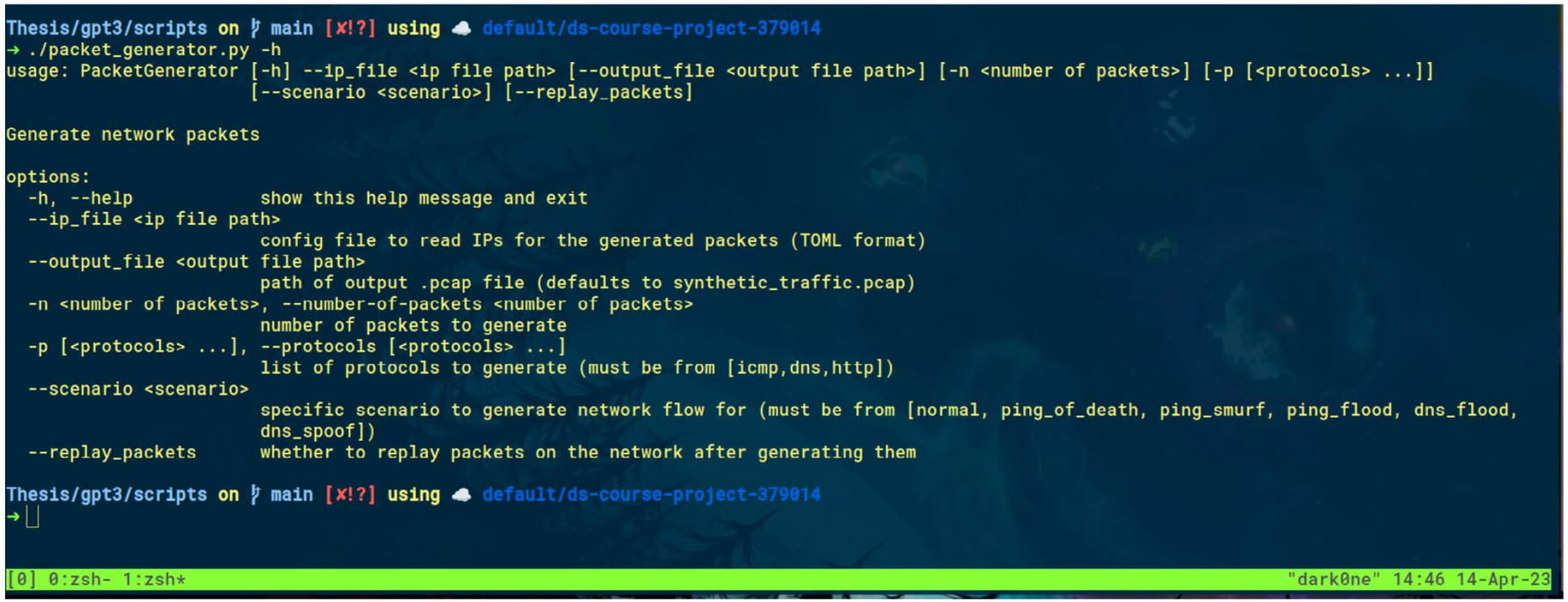


FIGURE 6. An overview of CLI tool for generating network traffic.

評估與實驗結果

研究使用了一系列方法和指標來評估模型，這些指標被分為兩大類：

內在指標 (Intrinsic metrics)：用於評估**底層機器學習模型本身的效能**。

些指標包括**訓練損失 (training loss)**、**訓練序列準確性 (training sequence accuracy)** 和**訓練 token 準確性 (training token accuracy)**。這些指標來自 OpenAI API，用於衡量模型在訓練批次中預測的 token 與真實 token 的匹配程度

外在指標 (Extrinsic metrics)：用於評估**生成的封包是否為合法的網路封包**並**類似於真實資料**。

研究強調，內在指標不足以反映這一點。該研究使用的主要外在指標是**成功率 (success rate)**，其定義是成功發送並收到適當回覆的封包數佔生成總封包數的比例。測量方式是將生成的封包在真實網路中重放，觀察有多少封包成功傳輸並收到適當的回應 (例如 ping 請求的 ping 回覆，或 DNS 查詢的主機名解析)。這種方法類似於其他研究中使用的評估方式

模型訓練實驗結果：
研究微調了三種 GPT-3 變體：DaVinci, Curie, 和 Babbage。
DaVinci 最大，Babbage 最小且最快，Curie 居中

模型訓練指標 (內在指標) 實驗結果

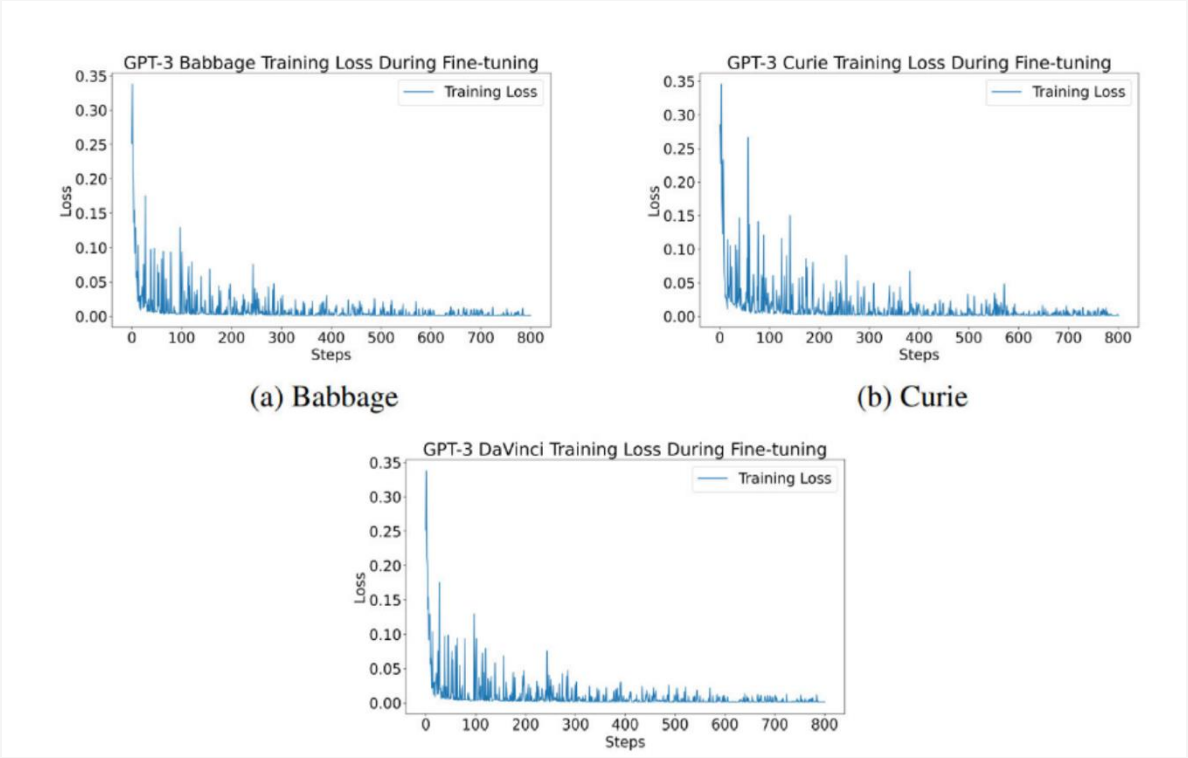


FIGURE 7. Loss values for GPT-3 models during fine-tuning process.

模型訓練實驗結果：
研究微調了三種 GPT-3 變體：DaVinci, Curie, 和 Babbage。
DaVinci 最大，Babbage 最小且最快，Curie 居中

模型訓練指標 (內在指標) 實驗結果

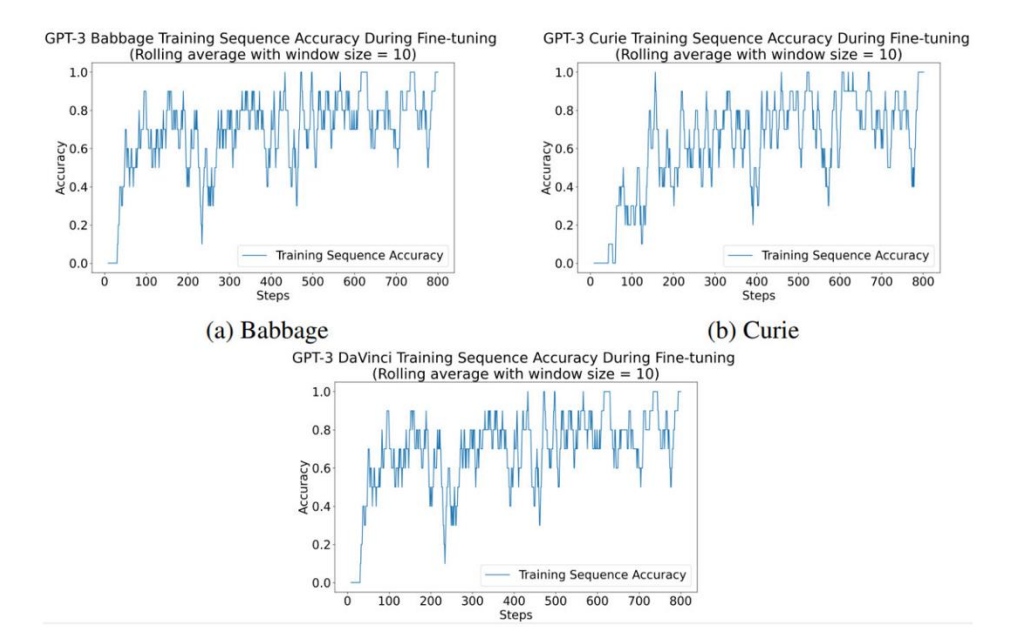


FIGURE 8. Sequence accuracy for GPT-3 models during fine-tuning process.

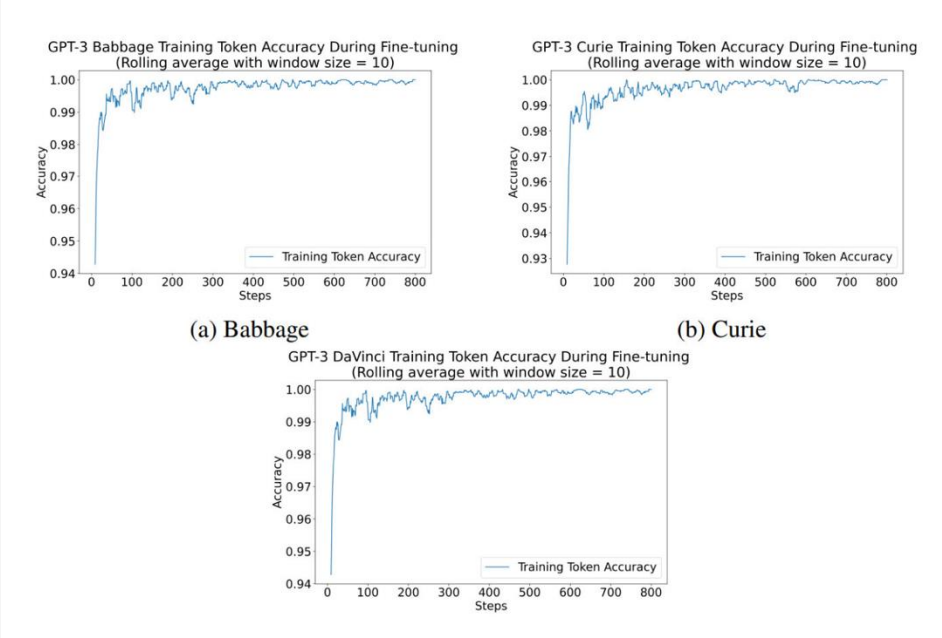


FIGURE 9. Token accuracy for GPT-3 models during fine-tuning process.

合成封包成功率 (外在指標) 實驗結果

模型使用三種數據集進行訓練：ICMP 封包 (ping 請求和回覆)、DNS 封包 (DNS 查詢請求和響應)，以及 ICMP/DNS 的組合。每個數據集包含 200 對提示-完成，每對包含五行封包摘要-代碼，總共 1000 個訓練樣本。所有討論過的模型都在每個數據集上進行了訓練。此外，實驗還包括一個預訓練的 DaVinci 模型，它沒有經過微調，需要使用提示工程。

實驗設置：

- 每個模型生成 100 個對應其訓練數據類型的封包。每個實驗運行 5 次，報告最佳和最差結果。
- 使用來自 Table 1 的公共 IP 和主機名列表進行測試。

Ping IP (Organization)	DNS Hostname
8.8.8.8 (Google)	www.google.com
208.67.222.222 (OpenDNS)	www.opendns.com
9.9.9.9 (Quad9)	www.facebook.com
176.103.130.130 (AdGuard)	www.atlassian.com
58.96.3.34 (Exetel)	www.bing.com

TABLE 1: List of Ping IPs and DNS Hostnames used for evaluation.

合成封包成功率 (外在指標) 實驗結果

Model	ICMP	DNS	ICMP/DNS
GPT-3 *"Pre-trained DaVinci"*	90-100%	3-10%	48-61%
GPT-3 *"Fine-tuned Babbage"*	75-100%	0-7%	41-58%
GPT-3 *"Fine-tuned Curie"*	0-23%	0-2%	1-16%
GPT-3 *"Fine-tuned DaVinci"*	91-100%	5-8%	46-59%

TABLE 2. Success Rate for packet generation models for normal network flow.

Model	ICMP	DNS	ICMP/DNS
GPT-3 *"Pre-trained DaVinci"*	87-100%	N/A[1]	44-66%
GPT-3 *"Fine-tuned Babbage"*	80-100%	N/A[1]	39-60%
GPT-3 *"Fine-tuned Curie"*	4-25%	N/A[1]	0-23%
GPT-3 *"Fine-tuned DaVinci"*	95-100%	N/A[1]	44-57%

TABLE 3. Success Rate for packet generation models for malicious network flow (Ping flood scenario).

合成封包成功率 (外在指標) 實驗結果

ICMP 結果：模型在處理 ICMP 封包生成任務時大多表現良好。微調後的 Babbage 和 DaVinci 表現良好，有時能生成 100% 可傳輸並接收響應的封包。**令人印象深刻的是，未經微調的 DaVinci 模型僅通過預訓練和提示工程技術也能達到類似的高品質結果**。

DNS 結果：**在生成 DNS 封包的過程中，所有模型的表現都顯著不足**。預訓練的 DaVinci 表現最好，但成功率仍然相當低，僅為 10%。

ICMP/DNS 混合 結果：兩種協定的整合似乎產生了稍好的結果，模型通常能接收到近一半封包的回應。然而，這不一定表示模型對 DNS 標頭字段的理解有所提高。由於 ICMP/DNS 數據集中有一半是 ICMP 封包，可能只是反映了模型在生成 ICMP 封包方面的優勢。

研究觀察認為，這種表現不佳似乎與底層的 Scapy 框架有關，Scapy 在處理 DNS 封包創建時的語法**比 ICMP 封包的語法複雜得多**。這是合理的，因為 DNS 是比 ICMP 更複雜的網路協定。

結論

該研究探索了合成網路流量生成，並提出了一個使用 LLMs 進行端到端流量生成的框架。在提出的框架內實現了幾種方法。研究結果總結如下：

- 訓練和測試了一些基於 Transformer 的模型來生成 ICMP 和 DNS 協定的網路封包，並使用不同的評估指標比較了它們的性能。主要結論是，Transformer 模型在生成簡單網路封包（如 ICMP 協定封包）時表現良好，**即使沒有微調也能表現良好**，但對於更複雜的協定（如 DNS）則表現較差。

- 提出並實現了一個新的端到端網路流量生成器框架。該框架由兩個組件組成：封包生成器和流量生成器。這種模組化設計不僅有助於研究的技術實現，還使得在更大的合成網路數據生成方案中可以單獨交換和改進每個任務。

- 使用 Python 函式庫實現了一個 CLI 工具，用於處理框架中的流量生成部分，並方便終端使用者使用封包生成管線。此外，CLI 工具中還包含了幾種惡意情境，除了正常流量生成外提供了替代方案。

未來展望

- 擴展支援的網路協定，以提高生成數據的多樣性。
- 改進流量生成組件的功能，例如納入更複雜的情境或採用基於 AI 的動態流生成。
- 深化合成網路生成的驗證/評估方法，以解決缺乏社區普遍接受的評估方法的問題。

Reference

- https://ieeexplore.ieee.org/document/10287342

謝謝大家