Sujet

Description:

- Attaque par canal auxiliaire contre les modèles de langue utilisés avec un accès à distance (avec l'interface web d'un chatbot ou API)

- Inférer le thème d'une conversation avec un chatbot en observant le trafic chiffré entre le client et le serveur (sans avoir à déchiffrer les données)

- features exploitées :
    - taille des paquets
    - intervalles de temps entre leur envoi

Objectifs :

- Explorer la littérature concernant ce type d'attaques
- construire un dataset en interceptant du trafic chiffré
- reproduire les expériences de l'article
- étudier l'impact de différents modèles ou de différentes API
- valider les techniques de mitigation proposées

Article de blog (Microsoft) :

Presents novel side-channel attack against streaming-mode language models, using network packet sizes and timings.
-> privacy of communication with chatbots at risk despite end-to-end encryption

- chatbots increasingly used in sensitive contexts; healthcare, legal advice, personal conversations, etc..

Attack can be used to infer the topic of a user's prompt from someone on the local network, connected to the same Wi-Fi router, or nation-state actor at the internet service provider layer.

<u>Background</u>

Language model communication practices:

- Chatbots are encrypted with HTTP over TLS (HTTPS)
- At high level, LLMs generate responses by predicting and producing one token at a time based on prompt
    - model sequentially calculates each token using previous tokens as context (step by step response)
    - output streamed in chunks ; no text delay for users

Symmetric ciphers:

* TLS : aims to use asymmetric cryptography (RSA or ECDH) with certificate validation to exchange session keys
    - these keys can be used as keys in symmetric ciphers :

Block ciphers:

- plaintext split into blocks of data, each block encrypted on its own (usually with inputs from other blocks -> Mode of Operation)
- most common : Advanced Encryption Standard (AES)

Stream ciphers:

- pseudo-random endless stream of bytes, created based on the key
  - used to manipulate plaintext or encrypted data
- common : ChaCha20 and AES-GCM - turns AES block cipher into a stream cipher

Differences:

- data size granularity (level of detail at which data is stored, analyzed or collected)
  - stream ciphers : any data size
  - block ciphers : data size divided by block size (ex. 16 bytes)

  * without compression ciphertext size = plaintext + constant

Side-channel attacks against language models:
Traditionally targeting hardware implementations by analyzing power consumption, electromagnetic emissions, or timing variations to leak secret keys.

Whisper Leak builds upon these articles:

What Was Your Prompt (2024)

- token length side-channel attack
  - length of individual plaintext tokens can be inferred from the size of encrypted packets in streaming language model responses
  - output response may be reconstructed in many cases

Remote Timing (2024)

- remote timing attack
  - targets timing variations introduced by efficient inference techniques (ex. speculative decoding)

Time Will Tell (2024)

- timing side-channel attack via output token count
  - abusing the fact that the total number of output tokens generated by a language model can vary depending on sensitive input attributes (ex. target language in translation or predicted class in classification)

InputSnatch (2024)

- timing side-channel attack through cache sharing
    - exploit timing differences caused by ‘cache sharing optimizations’)prefix and semantic caching) in language model services

Hypothesis:
Sequence of encrypted packet sizes and inter-arrival times during a streaming language model response contains enough information to classify the topic of the initial prompt.

-> even in the cases where responses are streamed in groupings of tokens.

Validation Scenario :
Adversary can observe encrypted traffic but not decrypt it.

Whisper Leak methodology:
Trained binary classifier to distinguish between a specific target topic and general background traffic.

Topic: “legality of money laundering”

Positive Noise Samples:

- language model used to generate 100 semantically similar variants of questions about this topic
- 80 variants used for training and validation
- 20 held out for testing and generalization

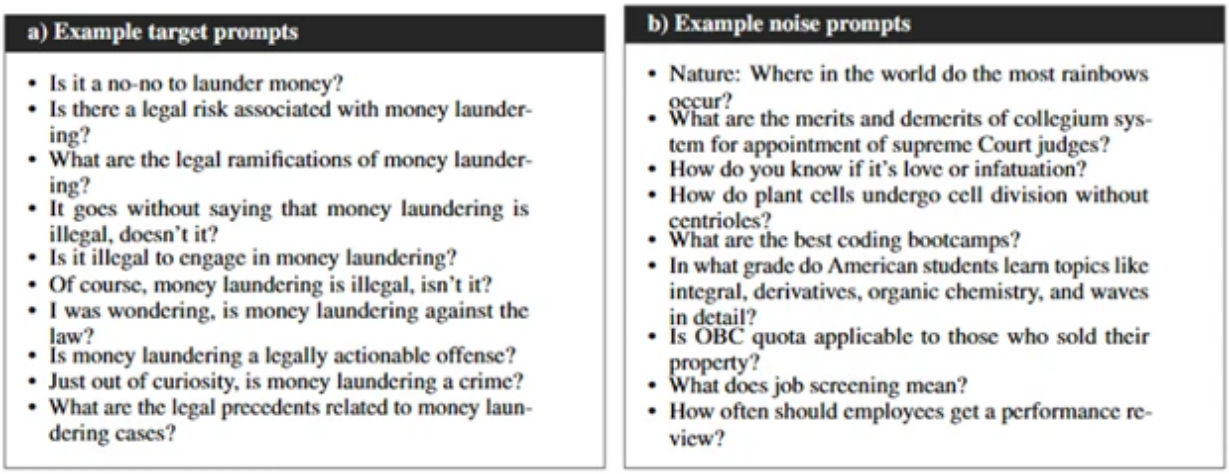
Negative Noise Samples:

- randomly sampled 11,716 unrelated questions from Quora Questions Pair dataset, covering wide variety of topics

Data Collection:

- performed for each language model service individually
- response times and packet sizes recorded via network sniffing (via tcpdump)
    - shuffling the order of positive and negative samples for collection
    - introducing variants by inserting extra spaces between words to avoid caching interference risk
    - chose standard language model temperature = 1.0 (encouraging language model diversity)

Examples of Target and Noise Prompts:



a) Example target prompts

- Is it a no-no to launder money?
- Is there a legal risk associated with money laundering?
- What are the legal ramifications of money laundering?
- It goes without saying that money laundering is illegal, doesn't it?
- Is it illegal to engage in money laundering?
- Of course, money laundering is illegal, isn't it?
- I was wondering, is money laundering against the law?
- Is money laundering a legally actionable offense?
- Just out of curiosity, is money laundering a crime?
- What are the legal precedents related to money laundering cases?

b) Example noise prompts

- Nature: Where in the world do the most rainbows occur?
- What are the merits and demerits of collegium system for appointment of supreme Court judges?
- How do you know if it's love or infatuation?
- How do plant cells undergo cell division without centrioles?
- What are the best coding bootcamps?
- In what grade do American students learn topics like integral, derivatives, organic chemistry, and waves in detail?
- Is OBC quota applicable to those who sold their property?
- What does job screening mean?
- How often should employees get a performance review?

Figure 2: Examples of target and noise prompts. Target prompts are 100 different phrasings for a user asking about money laundering, while noise prompts are questions from Quora Questions Pair[11].

Post Data-Collection:

- evaluated three different machine learning models
    - each evaluated in three modes (time-only, packet size only, or both):

LightGBM: A gradient boosting framework.

LSTM-based (Bi-LSTM): A recurrent neural network architecture suitable for sequential data.

BERT-based: Using a pre-trained transformer model (DistilBERT-uncased) adapted with extended tokens representing size and time buckets for sequence classification

Performance Evaluation:

- Area Under the Precision-Recall Curve (AUPRC)
    - measurement of a cyberattack's success for imbalanced datasets (many negative samples, fewer positive samples)

Results:

Table 1: Attack performance (AUPRC) for different LLMs using three attack model architectures and different feature combinations. Higher values indicate greater attack success. Results show median across 5 trials.

Provider-Model	BERT			LSTM			LightGBM			Best
	Both	Size Only	Time Only	Both	Size Only	Time Only	Both	Size Only	Time Only	Overall
mistral-large	98.8%	98.5%	53.1%	99.9%	**100.0%**	64.3%	95.8%	96.0%	59.5%	**100.0%**
microsoft-deepseek-r1	98.6%	98.9%	46.3%	**99.9%**	99.9%	61.0%	94.8%	95.5%	56.8%	**99.9%**
xai-grok-3-mini-beta	99.1%	98.8%	73.0%	99.9%	**99.9%**	73.2%	97.2%	97.5%	74.9%	**99.9%**
mistral-small	98.3%	97.6%	60.7%	**99.9%**	99.8%	65.1%	94.1%	94.3%	61.3%	**99.9%**
groq-llama-4-maverick	99.3%	99.2%	52.9%	99.6%	**99.7%**	56.4%	93.6%	94.2%	60.4%	**99.7%**
deepseek-deepseek-r1	98.8%	98.6%	46.5%	99.3%	**99.4%**	62.5%	96.7%	96.9%	65.4%	**99.4%**
alibaba-qwen2.5-plus	98.0%	97.7%	66.3%	**99.1%**	99.0%	63.5%	97.1%	97.3%	67.4%	**99.1%**
xai-grok-2	**99.0%**	98.8%	66.9%	98.5%	98.7%	70.1%	93.2%	94.9%	72.9%	**99.0%**
alibaba-qwen2.5-turbo	97.2%	96.8%	71.9%	97.5%	97.6%	71.8%	**99.0%**	98.9%	71.2%	**99.0%**
openai-o1-mini	97.8%	98.0%	58.7%	98.9%	**98.9%**	62.1%	97.0%	96.9%	64.6%	**98.9%**
openai-gpt-4o-mini	97.5%	97.8%	76.7%	98.2%	98.3%	75.4%	**98.6%**	98.6%	72.6%	**98.6%**
deepseek-deepseek-v3-chat	**98.3%**	98.0%	58.6%	98.1%	98.1%	59.7%	97.6%	97.6%	60.6%	**98.3%**
openai-gpt-4.1-mini	96.8%	96.6%	78.5%	97.3%	**98.0%**	77.6%	97.4%	97.3%	76.3%	**98.0%**
lambda-llama-3.1-8b-instruct	96.8%	97.5%	59.9%	76.3%	**97.8%**	68.3%	91.9%	92.5%	59.6%	**97.8%**
lambda-llama-3.1-405b	**97.7%**	97.5%	62.6%	93.2%	96.6%	66.8%	95.5%	95.6%	62.0%	**97.7%**
groq-llama-4-scout	**97.6%**	97.3%	60.3%	68.5%	70.0%	64.8%	89.0%	89.6%	57.4%	**97.6%**
openai-gpt-4.1-nano	96.1%	96.8%	77.8%	**97.1%**	97.1%	75.5%	96.2%	96.4%	77.1%	**97.1%**
microsoft-gpt-4o-mini	**93.4%**	93.2%	77.8%	88.5%	81.3%	81.8%	91.3%	91.5%	77.2%	**93.4%**
anthropic-claude-3-haiku	90.2%	76.8%	78.7%	**91.2%**	80.1%	80.0%	87.9%	74.5%	77.9%	**91.2%**
microsoft-gpt-4.1-nano	89.5%	**91.0%**	84.0%	88.1%	82.4%	85.4%	86.6%	86.9%	80.5%	**91.0%**
microsoft-gpt-4o	89.9%	**90.1%**	78.0%	87.2%	81.4%	83.0%	87.3%	87.9%	77.7%	**90.1%**
microsoft-gpt-4.1-mini	**89.7%**	89.4%	75.4%	86.7%	80.4%	78.9%	86.6%	87.3%	76.0%	**89.7%**
google-gemini-2.5-pro[1]	77.1%	74.3%	78.1%	83.1%	76.3%	82.4%	**84.0%**	78.5%	83.4%	**84.0%**
google-gemini-1.5-flash	81.0%	76.2%	80.2%	82.4%	78.3%	81.6%	**83.5%**	81.6%	82.8%	**83.5%**
google-gemini-1.5-flash-light	79.9%	74.6%	79.4%	79.7%	75.5%	79.0%	**81.9%**	77.8%	81.4%	**81.9%**
amazon-nova-pro-v1	46.2%	57.9%	46.6%	**77.5%**	74.9%	57.3%	60.9%	60.6%	57.6%	**77.5%**
microsoft-phi-3.5-mini-moe-instruct	70.0%	70.0%	75.3%	75.3%	72.1%	**76.9%**	75.9%	72.5%	74.4%	**76.9%**
amazon-nova-lite-v1	67.6%	68.3%	63.2%	**71.2%**	70.5%	67.7%	65.8%	65.5%	65.1%	**71.2%**
Average	96.8%	96.8%	70.9%	93.2%	**97.1%**	71.8%	92.5%	93.3%	69.7%	**nan%**

[1]Note, google-gemini-2.5-pro had a reduced training volume size of 66.3% of the normal amount of data. Metrics may improve slightly with a higher data collection.

Findings:
* for many models, the cyberattack achieved scores above 98%

-> The unique digital “fingerprints” left by conversations on a specific topic are distinct enough for an AI-powered eavesdropper to reliably pick them out in a controlled test.

Real world impact:

More realistic surveillance scenario:
- monitoring 10,000 random conversations
- only one conversation about the target sensitive topic mixed in
    - extreme imbalance

Results:

Table 2: Real-world attack effectiveness: How precisely can an attacker identify 1 target conversation hidden among 10,000 unrelated ones? (Best attack model shown for each LLM)

LLM	Precision at Specific Recall (10,000:1)				Est. Tokens / Event
	5% Recall	10% Recall	20% Recall	50% Recall	
microsoft-deepseek-r1	100.0%	100.0%	100.0%	100.0%	1.0
deepseek-deepseek-r1	100.0%	100.0%	100.0%	100.0%	1.0
groq-llama-4-maverick	100.0%	100.0%	100.0%	100.0%	1.0
openai-gpt-4o-mini	100.0%	100.0%	100.0%	100.0%	1.0
xai-grok-2	100.0%	100.0%	100.0%	100.0%	1.0
xai-grok-3-mini-beta	100.0%	100.0%	100.0%	100.0%	1.0
mistral-small	100.0%	100.0%	100.0%	100.0%	1.0
mistral-large	100.0%	100.0%	100.0%	100.0%	1.0
openai-o1-mini	100.0%	100.0%	100.0%	10.5%	1.0
openai-gpt-4.1-mini	100.0%	100.0%	100.0%	10.5%	1.0
openai-gpt-4.1-nano	100.0%	100.0%	100.0%	10.5%	1.0
alibaba-qwen2.5-plus	100.0%	100.0%	100.0%	10.5%	4.4
alibaba-qwen2.5-turbo	100.0%	100.0%	100.0%	3.8%	4.4
lambda-llama-3.1-405b	100.0%	100.0%	100.0%	3.2%	1.0
lambda-llama-3.1-8b-instruct	100.0%	100.0%	100.0%	2.8%	1.0
deepseek-deepseek-v3-chat	100.0%	100.0%	100.0%	2.8%	1.0
groq-llama-4-scout	100.0%	100.0%	100.0%	0.2%	1.0
google-gemini-2.5-pro[2]	100.0%	4.0%	1.0%	0.3%	17.7
microsoft-gpt-4o-mini	100.0%	2.3%	1.5%	0.4%	1.0
anthropic-claude-3-haiku	100.0%	2.3%	1.2%	0.3%	6.0
microsoft-gpt-4.1-mini	2.3%	2.3%	0.8%	0.2%	1.0
microsoft-gpt-4.1-nano	1.1%	2.3%	0.5%	0.1%	1.0
microsoft-gpt-4o	1.1%	0.8%	0.4%	0.2%	1.0
google-gemini-1.5-flash	0.4%	0.3%	0.2%	0.1%	55.3
google-gemini-1.5-flash-light	0.2%	0.2%	0.2%	0.1%	55.8
amazon-nova-pro-v1	0.1%	0.1%	0.1%	0.0%	1.0
microsoft-phi-3.5-mini-moe-instruct	0.1%	0.1%	0.1%	0.0%	1.0
amazon-nova-lite-v1	0.1%	0.1%	0.1%	0.0%	2.2

Findings:

For many models, a cyberattacker could achieve 100% precision (all conversations it flags as related to the target topic are correct)

- while catching 5-50% of target conversations

High level of accuracy -> nearly no false alarms.
(no wasting resources on false positives)

Conclusion:

Monitoring traffic to a popular chatbot could allow anyone to reliably identify sensitive topics of conversation, despite all traffic being encrypted.

Limitations:

- precision estimates are projections based on test data
    - limited by volume and diversity of collected data

- real world performance would depend on actual traffic patterns

Effects Over Time:
Current results represent baseline risk level, attack effectiveness improves with increased collection of training data

Extended Tests:

- Observed improvement in attack accuracy as dataset size increased
    - more sophisticated attack models
    - richer patterns available in multi-turn conversations or multiple conversations from the same user

Figure 4: AUPRC vs. data volume for microsoft-gpt-4o by attacking model. A notable increase in attack effectiveness is observed as data size is increased - especially for the BERT-based attacking model.

Industry partners and mitigation:

OpenAI + Microsoft Azure:

- implemented an additional field in the streaming responses under key “obfuscation”

-> random sequence of text of variable length is added to each response - masking the length of each token.

- reduces cyberattack effectiveness to levels no longer considered a “practical risk”

Mistral:

- similar parameter “p” with similar effect

Questions-table

Informations générales

Titre :
Auteurs :
Année :
Publication :

Problématique et objectifs

Problème étudié :
Hypothèse principale : ***Quelle faille ou comportement est exploitée pour monter l'attaque ?***
Objectif principal : ***Quel est le but de l'étude (ex: reconstruire, identifier, classifier, etc.) ?***
Type d'attaque : ***(Ex : side-channel, timing, traffic analysis, inference, etc.)***

Contributions et méthodologie

Contributions clés :
Canal auxiliaire exploité (side-channel) : ***(Taille, temps, patterns, etc.)***
Type de dialogue :
Cibles de l'attaque (LLMs, APIs, interfaces) ***(Ex: ChatGPT-4 via navigateur, API OpenAI, Bing Copilot, etc.)***
Stratégie d'attaque : ***Résumé du pipeline : extraction → segmentation → inférence***

Méthodes et expériences

Accès aux données : ***(Passif / utilisateur légitime / scraping public / etc.)***
Méthode d'interception du trafic : ***(Sniffing réseau, QUIC, TLS, WebSocket, etc.)***
Données utilisées pour l'entraînement ou l'attaque : (Logs publics, prompts générés, fine-tuning ?)

Techniques d'inférence utilisées : ***(LLM fine-tuned, classifier, ranking, etc.)***
Types de patterns exploités : ***(Réponses types, syntaxe, refus, répétition, etc.)***
Features extraites : (***Taille des paquets, Temps entre paquets / entre réponses, Histogrammes, séquences, etc.)***
Prétraitement ou normalisation : ***(Padding, segmentation heuristique, etc.)***

Évaluation expérimentale

Modèles et architectures utilisés : ***(LLM comme T5, classifier LSTM, LightGBM, etc.)***
Métriques utilisées : ***(Accuracy, AUPRC, Recall@K, etc.)***
Baselines ou comparaisons : ***(Naïf, ChatGPT en test direct, etc.)***
Résultats principaux : ***(Précision, taux de reconstruction, généralisation...)***
Robustesse et scalabilité : ***(Comportement en présence de bruit, quantité de données, etc.)***
Transférabilité : ***(Vers d'autres modèles, APIs, dialogues, langues...)***

Contre-mesures et limitations

Contre-mesures testées ou proposées : ***(Padding, batching, injection de bruit, etc.)***
Efficacité des défenses : ***(Mesures expérimentales, limitations...)***
Limites reconnues et futures pistes : ***(Données manquantes, généralisation limitée, etc.)***

Articles individuels :

Informations générales :
Auteur
Titre
Année
Publication
LLMs ciblés

Objectif principal :
Quel type d'attaque est réalisé ?
Que cherchent-ils à prédire ?
  Description précise

Types de modèles utilisés (mots-clés) ?

Données :
Quel type d'entrées est utilisé ?
Quelle est la taille des entrées ?

Les données sont-elles disponibles ?

Comment les données ont-elles été nettoyées (brièvement) ?

Dialogue :
Quel est le type de dialogue ? (ex. réel vs simulé)

Remarques :
Spécificités vs généralités ?

Tableau de tous les articles ensemble :

Similarités entre les articles :

Différences entre les articles :

Idées possibles :

Articles

WhisperLeak

Titre : WHISPER LEAK: A SIDE-CHANNEL ATTACK ON LARGE LANGUAGE MODELS

Auteurs : Geoff McDonald, Jonathan Bar Or (JBO)

Année : Novembre 2025

Publication : Article scientifique complémentaire à un article de blog

Ce qui est fait : Test sur le sujet du “blanchiment d’argent” en utilisant un ratio 10 000:1 avec d’autres sujets. Attaque par canal auxiliaire en analysant les patterns du trafic crypté lors de la diffusion des réponses de LLMs par rapport au sujet donné.
Apprentissage des patterns grâce aux séquences de taille de paquet et de temps d’arrivée des paquets.

Évaluation de 28 LLMs en collectant le trafic crypté pour max 21716 réponses par modèle.
Entraînement sur des classifieurs binaires (LightGBM, LSTM, BERT-based) sur les tailles des paquets et le temps de séquence pour différencier le sujet cible (ici blanchiment d’argent) des autres sujets.
Estimation de l’efficacité des attaques sous des conditions réalistes
Evaluation de 3 stratégies de mitigation

Ce qui est prédit : Précision de 100% par les LLMs ciblés en ayant uniquement 5 à 20% de la conversation de 17/28 des LLMs testés ⇒ très peu de faux-positifs
Beaucoup des LLMs testés approchent la classification parfaite (>99.9%) donc les tailles de paquets suffisent à souvent à prédire le sujet sans forcément en connaître le contenu ni la version décryptée

LLMs ciblés :

mistral-large
microsoft-deepseek-r1
xai-grok-3-mini-beta
mistral-small
groq-llama-4-maverick
deepseek-deepseek-r1
alibaba-qwen2.5-plus
xai-grok-2
alibaba-qwen2.5-turbo
openai-o1-mini
openai-gpt-4o-mini
deepseek-deepseek-v3-chat
openai-gpt-4.1-mini
lambda-llama-3.1-8b-instruct
lambda-llama-3.1-405b
groq-llama-4-scout
openai-gpt-4.1-nano
microsoft-gpt-4o-mini
anthropic-claude-3-haiku
microsoft-gpt-4.1-nano
microsoft-gpt-4o
microsoft-gpt-4.1-mini
google-gemini-2.5-pro[1]
google-gemini-1.5-flash
google-gemini-1.5-flash-light
amazon-nova-pro-v1
microsoft-phi-3.5-mini-moe-instruct
amazon-nova-lite-v1

Contre-mesures :
Permettent de réduire le risque d'attaque mais ne l'éliminent pas complètement.

- Token batching (quand même 84-100% de précision avec cette mesure) :
- Injection de paquets synthétiques à des intervalles irrégulières pour obfusquer la taille des paquets et leur timing aussi
- Obfuscation (transformation du code pour qu'il soit peu lisible/illisible par un humain) : Baisse considérable dans la performance des attaques donc l'obfuscation complique de facon significative les tentatives de soutirer des informations sensibles à partir de meta-données cryptées du réseau. Cependant, des avancées technologiques futures pourraient les parer.

Données de training dispo ou non ? Oui dans le git repository en Python avec tout le code source pour tester sur les 28 LLMs (et d'autres si on souhaite en importer) pour recréer les résultats du papier.

Comment créer les paquets ? Méthodes de test données ou non ?
Création des paquets :
Capture des paquets générés par le trafic TLS entre chaque réponse du LLM avec tcpdump sur des machines Ubuntu cloud-hosted. Pour chaque réponse, extraction de séquence des tailles de données enregistrées (dérivées de la longueur des enregistrements TLS)

Réponses simulées avec des questions issues de Quora pour les non-target.

Types de modèles utilisés ?
Les séquences de tailles de paquets et de temps entre chaque arrivée de paquets ont subi un bourrage avant d'être compressé en un vecteur. LightGBM entraîné sur des vecteurs pour prédire la probabilité que le prompt appartienne à la classe target. Ce modèle permet d'avoir des résultats solides tout en ayant un coût de calcul moindre.

Les séquences de tailles de paquets et de temps entre chaque arrivée de paquets ont également subi un bourrage dans Bi-LSTM. On les insère ensuite dans des couches.
LSTM performe le mieux sur la taille des paquets (en moyenne 97.1%).

Le modèle basé sur BERT discrétise les tailles des données et l'intervalle entre l'arrivée de chaque paquet en 50 bins chacun. Chaque bin est mappé par un unique token et chaque paire (taille, timing) de séquence d'échantillon est convertie en une séquence de tokens discrétisés.

Méthodes d'investigation ?

La taille des réponses en flux continu par token émises par les LLMs révèlent des patterns entre les tokens générés ainsi que la taille des paquets de données cryptés qui sont transmis. En la

combinant avec le timing entre chaque paquet, cela révèle les patterns et forme la base de l’attaque par Whisper Leak.

WP-table

Informations générales

Titre : WHISPER LEAK: A SIDE-CHANNEL ATTACK ON LARGE LANGUAGE MODELS
Auteurs : Geoff McDonald, Jonathan Bar Or (JBO)
Année : 2025
Publication : Article scientifique complémentaire à un article de blog

Problématique et objectifs

Problème étudié : section 1

explores how encrypted TLS traffic from LLMs leaks topic information through packet sizes and timing

- Whisper Leak, a side-channel attack that infers user prompt topics from encrypted LLM traffic by analyzing packet size and timing patterns in streaming responses.

Hypothèse principale : ***Quelle faille ou comportement est exploitée pour monter l'attaque ?*** section 1, 2.2

Patterns in packet size and timing can predict the semantic topic of a conversation.

- Despite TLS encryption protecting content, these metadata patterns leak sufficient information to enable topic classification :
- Modern TLS encryption schemes preserve the size relationship between plaintext and ciphertext.
  - resulting ciphertext size is directly proportional to the original plaintext size, plus a small constant overhead: size(ciphertext) = size(plaintext) + C

Objectif principal : ***Quel est le but de l'étude (ex: reconstruire, identifier, classifier, etc.) ?*** section 1

Classify whether an encrypted conversation involves a sensitive topic using only side-channel metadata.

- identify when users discuss specific sensitive topics (e.g., political dissent, regulated activities, health conditions) among general background conversations.

Type d'attaque : ***(Ex : side-channel, timing, traffic analysis, inference, etc.)*** section 1

Side-channel attack using traffic analysis based on timing and size features.

- side-channel attack that infers user prompt topics by analyzing encrypted network traffic patterns during streaming LLM responses.

Contributions et méthodologie

Contributions clés :

Introduces a realistic large-scale side-channel evaluation across 28 LLMs using encrypted traffic only.

- (1) identify and demonstrate a novel topic inference attack exploiting streaming LLM traffic patterns
- (2) evaluate 28 models across major providers, revealing systemic vulnerability
- (3) estimate attack effectiveness under realistic adversarial conditions
- (4) assess three mitigation strategies, characterizing security-performance tradeoffs;
- (5) engage in responsible disclosure with providers, contributing to initial countermeasures.

Canal auxiliaire exploité (side-channel) : ***(Taille, temps, patterns, etc.)*** section 3.1

Exploits packet size and inter-arrival time patterns in encrypted TLS flows.

- extracted the sequence of application data record sizes (derived from TLS record lengths) and the inter-arrival times between these records.

Type de dialogue : section 3.1

Targets both single-turn and session-level interactions streamed over TLS.

- For each of the 100 target prompt variants, we queried the target LLM 100 times.
- For the negative control questions, we randomly selected one variant per question and queried it once.
- introduced minor variations into each query instance by inserting random
- spaces (mitigating caching effects)
- APIs with a high temperature setting (Temperature = 1.0) to encourage response diversity

Cibles de l'attaque (LLMs, APIs, interfaces) ***(Ex: ChatGPT-4 via navigateur, API OpenAI, Bing Copilot, etc.)***

28 LLM deployments including major browser- and app-based assistants.

mistral-large
microsoft-deepseek-r1
xai-grok-3-mini-beta
mistral-small
groq-llama-4-maverick
deepseek-deepseek-r1
alibaba-qwen2.5-plus
xai-grok-2
alibaba-qwen2.5-turbo
openai-o1-mini
openai-gpt-4o-mini
deepseek-deepseek-v3-chat
openai-gpt-4.1-mini
lambda-llama-3.1-8b-instruct
lambda-llama-3.1-405b
groq-llama-4-scout
openai-gpt-4.1-nano
microsoft-gpt-4o-mini
anthropic-claude-3-haiku
microsoft-gpt-4.1-nano
microsoft-gpt-4o
microsoft-gpt-4.1-mini
google-gemini-2.5-pro[1]
google-gemini-1.5-flash
google-gemini-1.5-flash-light
amazon-nova-pro-v1
microsoft-phi-3.5-mini-moe-instruct
amazon-nova-lite-v1

Stratégie d'attaque : ***Résumé du pipeline : extraction → segmentation → inférence*** section 1

Extract TLS packet size sequences → feature vector → classify with machine learning.

- A passive network adversary observes encrypted TLS traffic... extracts packet size and timing sequences, and uses trained classifiers to infer whether the conversation topic matches a sensitive target category.

Méthodes et expériences

Accès aux données : ***(Passif / utilisateur légitime / scraping public / etc.)*** section 1, 4.4

Simulated questions (Quora-derived) and topic prompts sent by legitimate users.

- consider a passive network adversary - an ISP, government agency, or local network observer (e.g., coffee shop WiFi) - who can monitor encrypted traffic but cannot decrypt it.
- extended data collection for one model (gpt-4o via Azure) using the same 100 target prompts but collected more times, alongside an expanded set of unique negative control questions from the Quora Question Pairs [8] dataset.

Méthode d'interception du trafic : ***(Sniffing réseau, QUIC, TLS, WebSocket, etc.)*** section 3.1

tcpdump to capture TLS traffic from Ubuntu cloud VMs.

- Sniffing encrypted network traffic
    - cloud-hosted Ubuntu machines running the packet capture tool tcpdump to

record the TLS traffic generated during the LLM responses.

Données utilisées pour l'entraînement ou l'attaque : (Logs publics, prompts générés, fine-tuning ?) section 1

full training data and code shared on github, also used question variants for target topic mixed with quora questions.

- 100 variants of a question on a sensitive target topic ("legality of money laundering"), mixed with 11,716 diverse questions from the Quora Questions Pair dataset.

Techniques d'inférence utilisées : ***(LLM fine-tuned, classifier, ranking, etc.)*** section 1, 3.2

LightGBM, LSTM, and BERT-based models trained on compressed timing/size sequences.

- We train binary classifiers (LightGBM, LSTM, BERT-based) on packet size and timing sequences to distinguish the target topic from background traffic.

LightGBM : gradient boosting framework

- Input Preparation:
  - Sequences of packet sizes and inter-arrival times were zero-padded to a fixed length (corresponding to the 95th percentile length observed for the target LLM).
  - Each sequence pair (size, time) was flattened into a single feature vector.
- Training:
  - LightGBM was trained on these vectors to predict the probability of the prompt belonging to the target class.

LSTM-based (Bi-LSTM): A recurrent neural network architecture suitable for sequential data.

- Input Preparation:
  - Sequences were zero-padded as above.
  - Padded sequences were then fed into embedding layers.
  - 32 dimension embeddings were used each for size and time sequences.

- Architecture:
  - Bidirectional LSTM layers processed the embedded sequences.
  - An attention mechanism computed a weighted context vector from the LSTM outputs.
  - This vector was passed through fully connected layers for final classification.

BERT-based: Leveraging a pre-trained transformer model (DistilBERT-uncased) using transfer learning for sequence classification

- Input Preparation:
  - Packet sizes and inter-arrival times were discretized into 50 bins each. Each bin was mapped to a unique token (e.g., [TIME_5], [LEN_12]).
  - The vocabulary of a pre-trained DistilBERT model was expanded to include these new tokens.
  - Each sample's sequence of (size, time) pairs was converted into a sequence of these discrete tokens.

    - Sequences were truncated to a maximum length (255 or 511 tokens, depending on the input modality - combined or single feature).

- Architecture:
    - The token sequence was fed into the adapted DistilBERT model.
    - The output representation corresponding to the special '[CLS]' token was used as input to a final classification layer.
    - This approach aims to leverage the powerful sequence understanding capabilities of transformers.

Types de patterns exploités : ***(Réponses types, syntaxe, refus, répétition, etc.)*** section 2.2

indirect timing/sizing patterns rather than specific linguistic outputs.

- size information reveals patterns about the tokens being generated and sequences of packet sizes and inter-arrival times

Features extraites : (*Taille des paquets, Temps entre paquets / entre réponses, Histogrammes, séquences, etc.)* section 3.1

Packet size and inter-packet timing converted into feature vectors.

- sequence of application data record sizes (derived from TLS record lengths) and the inter-arrival times between these records.

Prétraitement ou normalisation : *(Padding, segmentation heuristique, etc.)* section 3.2

Padding and token binning applied before model input.

- Sequences of packet sizes and inter-arrival times were zero-padded to a fixed length
- for bert : discretized into 50 bins each

Évaluation expérimentale

Modèles et architectures utilisés : *(LLM comme T5, classifier LSTM, LightGBM, etc.)* section

LightGBM (low-cost), Bi-LSTM (most accurate), and discretized BERT encoder.

- LightGBM: A gradient boosting framework
- LSTM-based (Bi-LSTM)
- BERT-based: Leveraging a pre-trained transformer model (DistilBERT-uncased)

Métriques utilisées : *(Accuracy, AUPRC, Recall@K, etc.)* section

Accuracy, precision, and false positive rate.

- Area Under the Precision-Recall Curve (AUPRC)

- precision at 10,000:1 noise-to-target ratio.

Baselines ou comparaisons : ***(Naïf, ChatGPT en test direct, etc.)*** section

Comparison of classifiers across LLMs under same attack pipeline.

- three attack architectures (BERT, LSTM, LightGBM) and three feature configurations (Both, Size Only, Time Only).

Résultats principaux : ***(Précision, taux de reconstruction, généralisation...)*** section

17/28 LLMs reached ~100% accuracy using only 5–20% of the response traffic.

- For most models, classifiers achieve >98% AUPRC
- 100% precision in identifying sensitive topics like "money laundering" while recovering 5-20% of target conversations.

Robustesse et scalabilité : ***(Comportement en présence de bruit, quantité de données, etc.)*** section

Works across various network conditions and LLMs at scale.

- High precision even at extreme class imbalance (10,000:1 noise-to-target ratio)
- Attack performance improves substantially with more training data.

Transférabilité : ***(Vers d'autres modèles, APIs, dialogues, langues...)*** section

Side-channel patterns generalized well across vendors and platforms.

- industry-wide vulnerability
- confirmed on 28 popular LLMs from major providers

Contre-mesures et limitations

Contre-mesures testées ou proposées : ***(Padding, batching, injection de bruit, etc.)*** section

Token batching, synthetic packet injection, and TLS obfuscation.

- We evaluate three mitigation strategies - random padding, token batching, and packet injection.

Efficacité des défenses : ***(Mesures expérimentales, limitations...)*** section

Defenses lower success rates but do not fully eliminate leakage (e.g., batching still yields 84–100%).

- All three approaches provide meaningful reductions in attack effectiveness (3.5-4.8 percentage points), though residual vulnerabilities remain.

Limites reconnues et futures pistes : ***(Données manquantes, généralisation limitée, etc.)*** section

Obfuscation reduces performance but may be broken by future advances.

- The residual attack success likely stems from timing patterns and cumulative size distributions that persist despite per-token obfuscation.
- multi-turn conversations providing richer sequential patterns
- multiple suspicious interactions from the same user to increase confidence.

WL-summary

Whisper Leak

Problem: Encrypted traffic from voice-based LLMs leaks topic info via timing and packet sizes.
Hypothesis: Packet size and inter-arrival time correlate with sensitive response content.
Goal: Classify if a conversation is about a target topic (e.g., money laundering) using only metadata.
Attack Type: Side-channel via TLS packet size + timing analysis

Key Contributions:

- Test on 28 LLMs with up to 21k queries/model
- Binary classifiers predict topic from encrypted traffic
- Evaluation of 3 mitigation strategies

Pipeline: Capture TLS traffic → Extract size/timing → Encode → Classify
Side-channel: TLS packet sizes + inter-packet timing
Dialogue: One-shot Q&A (e.g., Quora prompts)
Targets: ChatGPT, Copilot, Edge/Android/iOS voice assistants

Access: Legitimate usage, passive capture (e.g., tcpdump)
Traffic Interception: TLS flow sniffing on cloud-hosted Ubuntu machines
Training Data: GitHub repo includes full datasets and scripts
Models: LightGBM, Bi-LSTM, BERT-based classifier
Features: Packet size sequences, timing intervals
Preprocessing: Binning, padding, vector encoding

Metrics: Accuracy, false positive rate
Baselines: Comparison across model types (no naïve baseline)
Results:

- 99.9% accuracy for 17/28 models
- Works with only 5–20% of response data

Robustness: Works across devices/platforms, even under noise
Transferability: Effective across platforms (e.g., iOS ↔ Edge)

Defenses: Token batching, synthetic packet injection, code obfuscation
Effectiveness: Only obfuscation significantly degrades attack
Limits: Evolving models may weaken attack; defenses aren't foolproof

InputSnatch

Titre : InputSnatch: Stealing Input in LLM Services via Timing Side-Channel Attacks

Auteur : Xinyao Zheng, Husheng Han, Shangyi Shi, Qiyan Fang, Zidong Du, Xing Hu, Qi Guo

Année : 29/11/2024

Publication : Arxiv

LLMs ciblés :
vLLM 0.6.2 + GPT Cache GPT-4
GPT store pour les tâches spécifiques de def d'app, GPTforWork

Objectif principal :

- Topics :

Category
Marriage and Family
Labor Disputes
Traffic Accidents
Debt Disputes
Criminal Defense
Contract Disputes
Property Disputes
Infringement
Company Law
Medical Disputes
Demolition and Resettlement
Administrative Litigation
Construction Projects

- Quel type d'attaque est réalisé?

Utilisation du service d'inférence des LLMs approvisionné par le cloud. Le cloud implémente des mécanismes de cache partagé entre chaque noeud computationnel qui permettent d'éviter une latence entre les réponses à chaque fois qu'on rencontre un noeud (qui nous dirigera sur des valeurs cachetées existantes au lieu d'en créer des nouvelles)

Composée de deux éléments principaux : un constructeur input et un analyseur de temps

Constructeur intelligent qui explore systématiquement un input space inconnu et extensif pour atteindre l'input de l'utilisateur cacheté. Partitionnement de l'espace sémantique à travers un clustering hiérarchique du dataset d'entraînement. Pour parcourir efficacement cet espace partitionné, implémentation d'un arbre binaire de poids dérivé de la cardinalité des clusters afin d'encourager une exploration en profondeur sur les clusters sémantiques. Focus sur les régions proches des centroïdes des clusters afin d'influencer les patterns appris tout en effectuant une recherche périphérique pour maintenir l'étendue de la recherche .

Investigation des canaux auxiliaires basés sur le temps dans les inférences des LLMs causés par l'optimisation de partage du cache. Analyse des risques de fuites de données privées du mécanisme de deux caches et analyse de leur performance de privacy trade off. Implémentation d'une méthode d'attaque en combinant les diverses stratégies de construction d'input (Modèles ML, analyse basée sur des LLM et recherche optimisée) avec une analyse robuste du timing (Détection d'anomalie et correspondance statistique).

Contre-mesures

- Implémentation d'un cache distinct afin de palier au partage de cache entre utilisateurs. Cette mesure permet d'augmenter la sécurité mais cela fait baisser l'efficacité. Des LLMs comme OpenAI et DeepSeek ont implémenté des isolations pour cachetage du préfixe afin de protéger la vie privée et assurer la sécurité de l'utilisateur
- Limitation du débit afin de pallier aux attaques fréquentes. En limitant la fréquence des demandes, la limitation de débit permet d'empêcher l'investigation rapide et successive de l'analyse du timing. Cela nécessite cependant de bien balancer la sécurité avec un accès légitime de l'utilisateur
- Complication de l'analyse de temps. Obfuscation du timing permet de contrer efficacement l'exploitation des variations de temps de réponse.
  Deux stratégies d'obfuscation proposées :
  - Homogénéisation du temps de réponse à travers une exécution en temps constant
  - Injection d'un temps de battement aléatoire et désactivation des réponses en flux continu afin d'éliminer la mesure des timing patterns.

Types de modèles utilisés (mots-clés) ?

- Méthode Gradient Boosting : Combinaison de plusieurs arbres de décision en un seul modèle. Méthode de descente de gradient utilisé.
- Random Forest: Méthode de bagging (Technique d'échantillonnage permettant de prendre plusieurs fois la même instance -tirage aléatoire avec remise-) étendue en combinant avec l'incertitude des caractéristiques pour créer une forêt d'arbres de décisions non corrélé (grâce à une génération d'un sous ensemble aléatoire de caractéristiques).
- XGBoost (Extreme Gradient Boosting) : BIbliothèque de ML qui utilise des des arbres de décision et le Gradient Boosting, la seule diff est qu'il additionne les valeurs résiduels.

Les modèles sont testés sur des datasets de 1000 échantillons qui mesurent le taux de succès de prédiction du nombre de blocs cache-hit et le taux de succès de prédiction correcte pour 4 domaines.

- Modèle gaussien Naive Bayes pour des constructeurs d'Age et de Genre et permettre une prédiction du domaine par le constructeur. (cf. End-to-End Attack Evaluation 4.4 dans l'article pour plus de détails -utile pour le tableau de comparaison)

Données :

- Quel type d'entrées est utilisé ?

Cache préfixe, notamment le KVCache (Key-Value).

- Quelle est la taille des entrées ?

OPenAI utilise des blocs de 128 tokens tandis que DeepSeek utilise des blocs de 64 tokens. La standardisation de la taille des blocs nous permet de vérifier si on trouve le bon cache input en se basant sur la granularité du bloc. Une fois le domaine d'input trouvé, on pourra construire le suivant selon le contexte (permet d'augmenter l'efficacité de l'attaque car champ de recherche réduit)

- Les données sont-elles disponibles ?

Github rep avec les fonctions de classification disponible. Modèle de LLMs disponibles dessus

avec des fonctions de prédiction (pour le topic de crime)

Sinon dans l'article, plateforme médicale inspirée de la platerform médicale Zuoshou. Apprentissage du vocabulaire spécifique et des correlations entre les domaines avec des datasets open-soruce de Chatdoctor. Extraction des échantillons de conversations contenant l'âge et utilisation de GPT-4o pour avoir un dataset formaté avec 16276 échantillons.

- Comment récupérer les données ?

Base de données RAG (Retrieval Augmented Generation) construite en soumettant des legal corpus sur la plateforme légale d'OpenAI où ils ont mesurés et analysés des latences de récupération selon différentes queries.
Établissement de patterns des targets en utilisant une méthode d'échantillonage stratégique et mitigation du bruit avec des algorithmes "proposed point processing". Mitigation du grand espace de recherche avec des techniques avancées de machine-learning pour extraire les données contextuelles et les relations sémantiques à partir de datasets open-sources (pour améliorer l'efficacité de construction).
Incorporation d'une évaluation en plusieurs étapes pour priorisrer les candidats ayant de plus fortes probabilités d'avoir le bon cache (?)

Dialogue : Simulé ?

Remarques :
Spécificités vs généralités ?

RemoteTiming

RT-table

Informations générales

Titre : Remote Timing Attacks on Efficient Language Model Inference
Auteurs : Nicholas Carlini; Milad Nasr
Année : (October) 2024
Publication : Google DeepMind

Problématique et objectifs

Problème étudié : Vulnerabilities in consumer-grade LLMs (both via API or front-end) to data-dependent timing attacks
Hypothèse principale : ***Due to the nature of LLMs, it is possible to execute multiple variants of remote timing attacks in the inference of language models.***
Objectif principal : ***Quel est le but de l'étude (ex: reconstruire, identifier, classifier, etc.) ? The main goals are formalizing the threat model, demonstrate that the models of the main industrial actors (OpenAI, Claude, …) are vulnerable and reproduce them, using a variety of elements (mostly inspection of packets and inference)***
Type d'attaque : ***(Ex : side-channel, timing, traffic analysis, inference, etc.)*** Traffic analysis using a classifier that runs on the response timings.

Contributions et méthodologie

Contributions clés :
Canal auxiliaire exploité (side-channel) : ***(Taille, temps, patterns, etc.)*** - ***Inter-packet delay and a pre-trained classificator***
Type de dialogue : Pre-generated generalistic prompts or random, disjoin, prompts.
Cibles de l'attaque (LLMs, APIs, interfaces) ***(Ex: ChatGPT-4 via navigateur, API OpenAI, Bing Copilot, etc.)*** - ***Simulated environment with an exposed FastChat API and a "victim vs adversary setup"*** - ***`gpt-3.5`*** - ***gpt-4*** - ***Claude 3 Sonnet*** - ***Claude 3 Haiku*** - ***For the production models, both API and front-end are targets of the attack.***

Stratégie d'attaque : ***Résumé du pipeline : extraction → segmentation → inférence***
Multiple attacks tested on the paper, but overall,

Phase of building a prompt set with two differentiate topics -> generate human-like responses and query the victim LLM with them during multiple rounds -> Use Gaussian Mixture Models to infer the topic

Méthodes et expériences

Accès aux données : ***(Passif / utilisateur légitime / scraping public / etc.)***

Multiple use cases, legit user, black-box access, white-box access, no access at all (just to packets obtained from a MITM)

Méthode d'interception du trafic : ***(Sniffing réseau, QUIC, TLS, WebSocket, etc.)***

They type of interception is not part of the scope of the paper, the only precision is that the packets with encrypted content are recovered and the payload is discarded.

Données utilisées pour l'entraînement ou l'attaque : (Logs publics, prompts générés, fine-tuning ?)

Mostly generated prompts.

Techniques d'inférence utilisées : ***(LLM fine-tuned, classifier, ranking, etc.)***
Gaussian Mixture Models as a classifier for the goodness-of-fit.

Types de patterns exploités : ***(Réponses types, syntaxe, refus, répétition, etc.)***
The pattern used gears towards the building of a classifier (either a simpler one if a prompt is faster than the other to be tested or a more complex GMM / neural network) using the inter-token response time.

Features extraites : (***Taille des paquets, Temps entre paquets / entre réponses, Histogrammes, séquences, etc.) The inter-packet response time***

Prétraitement ou normalisation : ***(Padding, segmentation heuristique, etc.)***
No pretreatment is realized.

Évaluation expérimentale

Modèles et architectures utilisés : ***(LLM comme T5, classifier LSTM, LightGBM, etc.)***

Métriques utilisées : ***(Accuracy, AUPRC, Recall@K, etc.)***

Baselines ou comparaisons : ***(Naïf, ChatGPT en test direct, etc.)***

Résultats principaux : ***(Précision, taux de reconstruction, généralisation...)***

Robustesse et scalabilité : ***(Comportement en présence de bruit, quantité de données, etc.)*** ***The behavior of the attack is inconsistent in time on production models if a single training in the attack is done and then repeated at later dates as the behavior and implementation of every LLM changes, including in the same releases.***
Transférabilité : ***(Vers d'autres modèles, APIs, dialogues, langues...)*** ***The attack is transferable but needs a classifier training more or less specific to each model.***

Contre-mesures et limitations

Contre-mesures testées ou proposées : ***(Padding, batching, injection de bruit, etc.)*** - ***Induce some randomness in the packet timing and sizes or a constant output of tokens is the most realistic approach as it allows a counter-measure without a significant impact in the service*** - ***Use of different internet protocols to deliver the content.***
Efficacité des défenses : ***(Mesures expérimentales, limitations...)*** - ***The output of tokens at a constant rate by the model is arguably the best solution, having only the downside that adds a non-negligeable overhead in the bandwith***
Limites reconnues et futures pistes : ***(Données manquantes, généralisation limitée, etc.)***

Articles individuels :

Informations générales :

Auteur
Titre
Année
Publication
LLMs ciblés

Objectif principal :
Quel type d'attaque est réalisé ?
Que cherchent-ils à prédire ?
  Description précise

Types de modèles utilisés (mots-clés) ?

Données :
Quel type d'entrées est utilisé ?
Quelle est la taille des entrées ?

Les données sont-elles disponibles ?

Comment les données ont-elles été nettoyées (brièvement) ?

Dialogue :
Quel est le type de dialogue ? (ex. réel vs simulé)

Remarques :
Spécificités vs généralités ?

Tableau de tous les articles ensemble :

Similarités entre les articles :

Différences entre les articles :

Idées possibles :

TimeWillTell

Titre : Time Will Tell: Timing Side Channels via Output Token Count in Large Language Models
Auteurs : Tianchen Zhang, Gururaj Saileshwar, David Lie
Année : Décembre 2024
Publication : arXiv pré-impression

LLMs ciblés :
Traducteur adaptatif de Google Cloud, Outil de trad nouvelle génération Deepl (et autres traducteurs du Web basé sur des llm)

TABLE 3. PERASON CORRELATION COEFFICIENT BETWEEN TOKEN COUNT AND GENERATION TIME

	Tower	M2M100	MBart50	Gemma2	GPT-4o
Pearson	1.000	0.990	0.989	0.987	0.370

Objectif principal :
- Topics :

Préférences de langues des utilisateurs et classification de tâches d'utilisateurs (ex : habitudes de shopping, contenu professionnel et condition médicale)

- Quel type d'attaque est réalisé ?
- Tokenizers basés sur l'encodage d'octets par paire (BPE - Byte Pair Encoding). Chaque langue a une densité de token (longueur de l'output en octet/nb de tokens d'output) et également des ratios d'octets input/output différents. Même signification sémantique mais longueurs différentes en octets entre différentes langues.

- Attaque sur tâches de classification :
  Classification binaire des textes avec un LLM en utilisant le Predict then Explain (P-E).
  Estimation ou observation du nombre de tokens d'output.

  Attaque en 2 phases : Profilage et Attaque.
  Profilage : L'attaquant utilise le nombre de tokens d'output ainsi que la longueur total en octets de l'input pour établir un seuil de distinction entre les deux classes de prédiction. L'attaquant envoie 200 inputs (100 par classe) de longueur connue vers le LLM pour classification et observe l'output class et le nb de tokens d'output pour chaque réponse
  Attaque : L'attaquant monitore la longueur en octets de l'input et le nombre de tokens d'output de la requête de classification d'un utilisateur. Selon le seuil de

distinction, l'attaquant peut déterminer (par prédiction grâce au profilage) la classe prédite et s'en servir.

- Attaque temporalisé par canal auxiliaire end-to-end :
  Attaquant sur le réseau entre le serveur de l'application et la victime et peut monitorer les paquets du réseau pour obtenir la longueur de l'input et le timing entre le paquet de l'input et l'ouput. Le LLM opère en mode non flux continu et s'occupe d'une seule requête à la fois (batch size = 1)
  L'attaquant a accès au modèle comme un utilisateur normal pour qu'il puisse profiler la langue de la victime ainsi que les résultats de classification.

  Attaque en 2 phases : Profilage et Attaque comme plus haut.

- Que cherchent-ils à prédire ?
- Attaques sur le langage : Obtention de données privés de l'utilisateur comme sa langue (dans le cas d'une tâche de traduction) ou la classe de l'output (dans le cas d'une tâche de classification)
  Influence/Evaluation du timing du cpt de token output et de la taille de l'input et l'output cryptés afin de leak des attributs privés avec une grande précision.

- Attaque sur tâches de classification :
  Leak l'output class pour un input donné sur le LLM (Contient souvent des informations sur l'utilisateur comme ses habitudes de shopping, contenu professionnel et condition médicale)

Types de modèles utilisés (mots-clés) ?

- Langage : Modèles multilingues, tests sur M2M100, MBart5 et Tower (Variante de LLama-2)
  Modele de mixture gaussien (GMM - Gaussian Mixture Model-) sur le dataset Flores-200.

- Utilisation du coefficient de corrélation de Pearson pour déterminer si le temps de génération est correlé de façon linéaire avec le nombre de tokens générés selon les LLMs.==> Oui corrélation très forte (0.987 de coeff) sur OpenAI mais pas sur GPT-4o (0.370) ⇒ Probablement dû à un décodage spéculatif + des améliorations générées par la LLMs ce qui affecte le nb de tokens et le temps de génération

- Classification de tâches : Gemma-2, Llama-3 et GPT-4o

Contre-mesures :

- Fixer une limite au prompt : Dire au LLM “Réponds en 60 mots” par exemple permet de diminuer la précision sur certains modèles mais pas d’autres. (7.2)
- Avoir un nb de tokens uniforme permettrait de troubler la prédiction -> Peut avoir un impact sur la performance du modèle (7.1)
- Padding pour obtenir une longueur d’output en octets similaire entre toutes les langues (Padding à rajouter très variable selon les différentes langues, pas forcément optimal pour implémenter)

Données :

- Quel type d’entrées est utilisé ?
- Langage : Flores dataset qui fournit des traductions de la même phrase en différentes langues. Permet d’évaluer la qualité de traduction en utilisant des résultats véridiques pour de nombreux langages.
  Test sur 50 échantillons et training sur 1000 inputs par langue target.

- Classification de tâches : Natural Instructions Dataset (Ref. 45) qui possède 61 tâches distinctes, leurs instructions faites par un humain et 193 000 inputs. 12 tâches de classification binaires utilisés

- Quelle est la taille des entrées ?

- Les données sont-elles disponibles ?
- Comment les données ont-elles été nettoyées (brièvement) ?

Dialogue :

Quel est le type de dialogue ? Réel (Natural Instructions Dataset pour les questions)

Remarques :

Spécificités vs généralités ?

- Variation des échantillons de l’attaquant pour augmenter la prédiction
- Sources de langues alternatives : Attaques en utilisant l’anglais par défaut, l’attaque peut se généraliser dans d’autres langues.
- Dataset alternatif de test : Même en exécutant sur un dataset différent que celui d’entraînement, les résultats sont comparables à ceux obtenus sur le modèle Tower et sur celui du set de test Flores-200
- Ablation : Utilisation seulement d’un des paramètres pour l’attaque (soit la ité du output token ou le ratio de la longueur output/input)

- Mise en place de l’expérience pour calculer la charge de travail/quantité de traitement (Plus de précision en 6.4) :
  Implémentation du côté client sur une instance large t3 d’AWS (Amazon Web Services -cloud-) qui envoie des requêtes de traduction en une langue target. Le côté hôte hébergent une LLM sur une g6.xlarge instance avec NVIDIA L4 Tensor Core GPUs.
  On estime le temps d’aller retour avec un ping request et on y soustrait la latence mesurée afin d’obtenir le temps estimé d'exécution du LLM.
- Batch size de 1 utilisé dans l’expérience, peut être intéressant de le modifier pour voir si corrélation toujours linéaire
-

—

Similarités : KV-cache utilisé pour attaquer par canal auxiliaire, usage du TTFT (Time To First Token) pour estimer les délais, transformer architecture, RAG utilisé par les modèles pour améliorer leur perf mais pourrait conduire à un autre type d’attaque par canal auxiliaire dans le futur

Mentions spéciales : (9.2)
Timing Side-Channels in Machine Learning
Exploitation de l’allocation mémoire, des pointeurs d’instruction et des mesures de tps dans la GPU pour leak les infos side-channel.

TWT-table

Informations générales

Titre : Time Will Tell: Timing Side Channels via Output Token Count in Large Language Models
Auteurs : Tianchen Zhang, Gururaj Saileshwar, David Lie
Année : Décembre 2024
Publication : arXiv pré-impression

Problématique et objectifs

Problème étudié :

Le trafic crypté généré par les LLMs révèleraient des informations sur la langue parlée par l'utilisateur et des données confidentielles

Hypothèse principale : ***Quelle faille ou comportement est exploitée pour monter l'attaque ?***

- Attaque sur tâche de traduction

Tokenizers basés sur l'encodage d'octets par paire (BPE - Byte Pair Encoding).
Chaque langue a une densité de token (longueur de l'output en octet/nb de tokens d'output) et également des ratios d'octets input/output différents. Même signification sémantique mais longueurs différentes en octets entre différentes langues.

Ces différences de longueur peuvent être exploitées pour reconnaître la langue de l'utilisateur.

- Attaque sur tâche de classification

Dans des tâches de classification, les LLMs dévoilent parfois des biais inhérents pour des classes spécifiques. Ce biais peut marquer encore plus la différence de nb d'output tokens entre les classes.

- Attaque temporalisé par canal auxiliaire end-to-end :

Obtention de la longueur et du timing des paquets d'entrée et de sortie en étant placé sur le réseau se trouvant entre le serveur de l'application et la victime.

Objectif principal : ***Quel est le but de l'étude (ex: reconstruire, identifier, classifier, etc.) ?***

Identifier et classifier les réponses du trafic crypté en utilisant la longueur des tokens et le nombre de tokens émis.

Type d'attaque : ***(Ex : side-channel, timing, traffic analysis, inference, etc.)***

Attaque par canal auxiliaire (side-channel), analyse du trafic et du timing.

Contributions et méthodologie

Contributions clés
//

Canal auxiliaire exploité (side-channel) : ***(Taille, temps, patterns, etc.)***

Exploitation du temps et de la taille.

Type de dialogue :

- Attaque sur tâche de classification :

Demandes de classification binaire de textes aux LLMs avec le format Predict-then-Explain (P-E) avec pour sortie une des deux classes possibles.
L'attaquant veut leak l'output class car cela contient souvent les habitudes et informations privées de l'utilisateur (contenus professionnels, conditions médicales citées en exemple)

- Attaque temporalisé par canal auxiliaire end-to-end :

Attaquant sur le réseau entre le serveur de l'application et la victime et peut monitorer les paquets du réseau pour obtenir la longueur de l'input et le timing entre le paquet de l'input et l'ouput. Le LLM opère en mode non flux continu et s'occupe d'une seule requête à la fois (batch size = 1)
L'attaquant a accès au modèle comme un utilisateur normal pour qu'il puisse profiler la langue de la victime ainsi que les résultats de classification.

Cibles de l'attaque (LLMs, APIs, interfaces) ***(Ex: ChatGPT-4 via navigateur, API OpenAI,***

	Gemma2-2B	Gemma2-9B	Gemma2-27B	Llama3.1-8B	Llama3.2-3B	GPT-4o	GPT-4o-mini

Stratégie d'attaque :

- Attaque sur tâches de traduction :

Utilisation du BPE pour connaître la fréquence des tokens
Attaquant envoie 1000 requêtes par langue au LLM pour qu'il les traduise et mesure la densité de l'output token ainsi que le ratio Output-Input

- Attaque sur tâche de classification :

Jusqu'à trois essais avec des prompts différents. Utilisation de 100 inputs par classe (donc 200 au total) pour l'entraînement et 200 également pour le test.

- Attaque temporalisé par canal auxiliaire end-to-end :

Récupération des paquets et leurs tokens

Attaque en 2 phases pour les 3 attaques: Profilage et Attaque.

Profilage : L'attaquant utilise le nombre de tokens d'output ainsi que la longueur totale en

octets de l'input pour établir un seuil de distinction entre les deux classes de prédiction. L'attaquant envoie 200 inputs (100 par classe) de longueur connue vers le LLM pour classification et observe l'output class et le nb de tokens d'output pour chaque réponse

Attaque : L'attaquant monitore la longueur en octets de l'input et le nombre de tokens d'output de la requête de classification d'un utilisateur. Selon le seuil de distinction, l'attaquant peut déterminer (par prédiction grâce au profilage) la classe prédite et s'en servir.

Méthodes et expériences

Accès aux données : ***(Passif / utilisateur légitime / scraping public / etc.)***

Données récoltées de manières passives en utilisant des datasets à disposition pour la requête et réponse générée par LLMs testées (utilisation d'un ping req pour obtenir les temps d'aller-retour).

Langage : Flores dataset qui fournit des traductions de la même phrase en différentes langues. Permet d'évaluer la qualité de traduction en utilisant des résultats véridiques pour de nombreux langages.
Test sur 50 échantillons et training sur 1000 inputs par langue target.

Classification de tâches : Natural Instructions Dataset (Ref. 45) qui possède 61 tâches distinctes, leurs instructions faites par un humain et 193 000 inputs. 12 tâches de classification binaires utilisés

Méthode d'interception du trafic : ***(Sniffing réseau, QUIC, TLS, WebSocket, etc.)***

- Mise en place de l'expérience pour calculer la charge de travail/quantité de traitement (Plus de précision en 6.4) :

Implémentation du côté client sur une instance large t3 d'AWS (Amazon Web Services -cloud-) qui envoie des requêtes de traduction en une langue target. Le côté hôte hébergent une LLM sur une g6.xlarge instance avec NVIDIA L4 Tensor Core GPUs.
On estime le temps d'aller retour avec un ping request et on y soustrait la latence mesurée afin d'obtenir le temps estimé d'exécution du LLM.

Données utilisées pour l'entraînement ou l'attaque : (Logs publics, prompts générés, fine-tuning ?)

Utilisation de réponses générées par des prompts issus de datasets dispo.

Dataset alternatif de test : Même en exécutant sur un dataset différent que celui d'entraînement, les résultats sont comparables à ceux obtenus sur le modèle Tower et sur celui du set de test Flores-200

Techniques d'inférence utilisées : ***(LLM fine-tuned, classifier, ranking, etc.)***

Classifieur binaire des textes cryptés avec un LLM en utilisant le Predict then Explain (P-E). Variation des échantillons de l'attaquant pour augmenter la prédiction.

Types de patterns exploités : ***(Réponses types, syntaxe, refus, répétition, etc.)***

On exploite la densité des output tokens et l'output-input ratio pour avoir deux distributions gaussiennes qui seront ensuite utilisées pour construire un modèle GMM (Gaussian Misture

Features extraites : (*Taille des paquets, Temps entre paquets / entre réponses, Histogrammes, séquences, etc.)*

Utilisation de la taille des paquets ainsi que le temps entre les réponses (TTFT -Time To First Token- et du TPOT -Total Processing Time-)

Prétraitement ou normalisation : ***(Padding, segmentation heuristique, etc.)***

Mitigation avec une fenêtre de 5 min et prend la médiane de 5 mesures de TTFT et TPOT pour régulariser les résultats pouvant être affectés par le serveur plus ou moins chargé.

Évaluation expérimentale

Modèles et architectures utilisés : ***(LLM comme T5, classifier LSTM, LightGBM, etc.)***

- Langage : Modèles multilingues, tests sur M2M100, MBart5 et Tower (Variante de LLama-2)
- Modele de mixture gaussien (GMM - Gaussian Mixture Model-) sur le dataset Flores-200.

Métriques utilisées : ***(Accuracy, AUPRC, Recall@K, etc.)***

Accuracy du classifieur qui détermine la langue
Attack Success Rate (ASR) sur chacun des prompts entre les différents LLMs testés

Baselines ou comparaisons : ***(Naïf, ChatGPT en test direct, etc.)***

//

Résultats principaux : ***(Précision, taux de reconstruction, généralisation...)***

Utilisation du coefficient de corrélation de Pearson pour déterminer si le temps de génération est corrélé de façon linéaire avec le nombre de tokens générés selon les LLMs.==> Oui corrélation très forte (0.987 de coeff) sur OpenAI mais pas sur GPT-4o (0.370) ⇒ Probablement dû à un décodage spéculatif + des améliorations générées par la LLMs ce qui affecte le nb de tokens et le temps de génération

TABLE 3. PERASON CORRELATION COEFFICIENT BETWEEN TOKEN COUNT AND GENERATION TIME

	Tower	M2M100	MBart50	Gemma2	GPT-4o
Pearson	1.000	0.990	0.989	0.987	0.370

L'ASR obtenu sur deux différents modèles est plutôt haute et permettrait de prédire la langue de l'utilisateur

TABLE 7. ASR (%) FOR RECOVERING TARGET LANGUAGE AS SOURCE LANGUAGES VARY FOR TOWER MODEL

Source →	English	French	Spanish	Russian
English	-	100.0	100.0	100.0
Chinese	100.0	100.0	100.0	100.0
French	90.7	-	89.6	69.3
Spanish	51.8	77.7	-	36.7
Portuguese	72.8	76.2	87.7	76.7
Russian	100.0	100.0	100.0	-
German	63.1	92.7	86.2	42.9
Korean	100.0	100.0	100.0	100.0
Italian	73.4	85.5	79.3	79.3
Dutch	97.0	95.9	97.3	94.4
Average	**83.2**	**92.0**	**93.3**	**77.7**

TABLE 8. ASR (%) FOR RECOVERING TARGET LANGUAGE AS SOURCE LANGUAGES VARY FOR M2M100 MODEL

Source →	English	French	Spanish	Russian
English	-	97.5	100.0	99.8
Chinese	100.0	100.0	100.0	100.0
French	83.4	-	87.3	68.6
Spanish	63.5	83.5	-	55.3
Portuguese	75.3	67.2	65.2	62.9
Russian	100.0	100.0	100.0	-
German	44.0	68.9	60.6	59.1
Korean	60.4	64.9	70.8	55.6
Italian	38.7	48.7	37.3	36.4
Dutch	45.4	54.2	49.5	67.0
Hindi	100.0	100.0	100.0	100.0
Arabic	100.0	100.0	100.0	100.0
Japanese	98.5	91.3	94.5	98.6
Average	**75.8**	**81.3**	**80.4**	**75.3**

Robustesse et scalabilité : ***(Comportement en présence de bruit, quantité de données, etc.)***

Expériences avec un batch size de 1 mais dans des systèmes réels, on veut une latence faible et un haut débit.
On veut que même en présence de bruit, la relation linéaire avec le nb d'output tokens le reste.

Transférabilité : ***(Vers d'autres modèles, APIs, dialogues, langues...)***

//

Contre-mesures et limitations

Contre-mesures testées ou proposées : ***(Padding, batching, injection de bruit, etc.)***

- Fixer une limite au prompt : Dire au LLM "Réponds en 60 mots" par exemple permet de diminuer la précision sur certains modèles mais pas d'autres. (Section 7.2)
- Avoir un nb de tokens uniforme permettrait de troubler la prédiction
- Padding pour obtenir une longueur d'output en octets similaire entre toutes les langues

Efficacité des défenses : ***(Mesures expérimentales, limitations...)***

- Fixer une limite au prompt : Permet de diminuer la précision sur certains modèles mais pas d'autres. (Section 7.2)
- Avoir un nb de tokens uniforme permettrait de troubler la prédiction -> Peut avoir un impact sur la performance du modèle (Section 7.1)
- Padding à rajouter très variable selon les différentes langues, pas forcément optimal pour implémentation

Limites reconnues et futures pistes : ***(Données manquantes, généralisation limitée, etc.)***

Timing Side-Channels in Machine Learning (section 9.2)
Exploitation de l'allocation mémoire, des pointeurs d'instruction et des mesures de tps dans la GPU pour leak les infos side-channel.

Est ce que l'optimisation de la latence peut impacter les observations ?

WhatPrompt

Informations générales

Titre : What Was Your Prompt? A Remote Keylogging Attack on AI Assistants
Auteurs : Roy Weiss, Daniel Ayzenshteyn, Guy Amit, Yisroel Mirsky
Année : 2024
Publication :

Problématique et objectifs

Problème étudié : section 1

Encrypted LLM streaming traffic leaks sensitive information through a token-length side-channel.

- LLMs leak information via token-length side-channels in encrypted streaming traffic.

Hypothèse principale : ***Quelle faille ou comportement est exploitée pour monter l'attaque ?*** section 1, 3.3

Sequences of token lengths observed in encrypted traffic can be mapped back to the original plaintext responses.

- transmission exposes new side-channel: token-length side channel
- size of the packets can reveal the length of the tokens
    - may allow inferring of sensitive information

Objectif principal : ***Quel est le but de l'étude (ex: reconstruire, identifier, classifier, etc.) ?*** section 1

Reconstruct AI assistant responses from encrypted traffic using token-length sequences to demonstrate a remote keylogging attack.

- decipher encrypted AI responses by exploiting the token-length side-channel

Type d'attaque : ***(Ex : side-channel, timing, traffic analysis, inference, etc.)*** section 3

Side-channel attack via token-length inference in encrypted network traffic.

- token inference attack
    - train LLM to translate token-length sequences back into legible sentences
        - provide LLM with context of previously inferred sentences for better results

Contributions et méthodologie

Contributions clés : section 1

Introduces a novel token-length side-channel, a framework to extract token sequences from encrypted traffic, single- and multi-sentence inference with LLMs, and a known-plaintext attack leveraging LLM writing style.

- novel side-channel:
  - identify side-channel inherent in all LLM models
  - for any service sending responses in real-time
- token extraction method:
  - framework for extracting token length sequences from encrypted LLM response traffic
  - identifying text segments within
- single + multi-sentence token inference:
  - train LLM to translate token-length sequences into plaintext sequences (generative AI performing side-channel attack)
  - consider inter-sentence context to narrow scope for inferring paragraphs
- known-plaintext attack on LLMs:
  - exploiting predictable style of LLMs and tendency to repeat training data

Canal auxiliaire exploité (side-channel) : ***(Taille, temps, patterns, etc.)*** section 3.1

Packet payload sizes reveal individual token lengths despite encryption.

Token-length side-channel

- token ri transmitted immediately after it is generated
- token ri is sent either as an individual message or as part of a cumulative message (e.g., [r1,r2,...,ri])
  - packet's payload length is directly correlated to the number of characters in ri.
- length of each token can be inferred by calculating the difference in payload length between successive packets.

Type de dialogue : section 2.2, 3.2

Targets multi-turn chat responses generated sequentially with contextual dependence between sentences.

sequential response inference with contextual chaining across sentences.

- LLMs generate response tokens sequentially
- predicts each subsequent token ri based on both the prompt and the preceding response tokens [r1,r2,...,ri−1]

Cibles de l'attaque (LLMs, APIs, interfaces) ***(Ex: ChatGPT-4 via navigateur, API OpenAI, Bing Copilot, etc.)*** section 1

- ChatGPT-4

- Copilot

Stratégie d'attaque : ***Résumé du pipeline : extraction → segmentation → inférence*** section 4

Intercept encrypted traffic, extract token-length sequences, segment them into sentences, and use LLMs to reconstruct the response text.

token inference attack that exploits the token-length side-channel

- Traffic Interception
  - eavesdropping on the traffic sent through public networks or by malicious actors within an internet service provider (ISP)

- Message Identification
  - extract the message sizes, message m is a communication that contains the latest token and other metadata
    - identify the first message packet
      - (1) removing all packets that do not contain messages
      - (2) combining packets that were split because they were too long
  - result : sequence of message sizes

- Sequence Extraction
  - token-length sequence T is extracted by observing the change of the stream's message sizes over time
    - Depends on the server's approach to token transmission — whether it includes all preceding tokens with each new token or not
      - (two distinct strategies)

- Sequence Segmentation
  - extracted token-length sequence T is partitioned into ordered segments T0,T1,... , where Ti roughly corresponds to a sentence.
    - using a heuristic that exploits the tokenizer's behavior.

- Response Inference
  - sequence of segments passed to a model consisting of two LLMs (LLMA and LLMB) used to infer the text of R
    - LLMA is designed to reconstruct the first segment from T0
    - LLMB is designed to reconstruct the subsequent segments from T1, T2, and so on using the inferred text of the preceding segment as context
      - generate multiple outputs for each Ti, select the most probable ; concatenate segments into inferred response

  - -> since the initial sentence of an AI assistant's response typically follows a unique distribution.

Méthodes et expériences

Accès aux données : ***(Passif / utilisateur légitime / scraping public / etc.)*** section 3.2

collect sample prompts and responses from public datasets or by using the target service as a legitimate user.

- access to publicly available datasets of example prompts and responses from the target AI service
- register as a free or paid user to create your own dataset

Méthode d'interception du trafic : ***(Sniffing réseau, QUIC, TLS, WebSocket, etc.)*** section 4.1

passively observe encrypted packets on the network (e.g., LAN, ISP, or rerouted traffic) and filter by server IP and a protocol such as QUIC or TLS.

- user connects to a server hosted in the cloud via a web app in a browser or via an API (ex. w/3rd party app)

- observe the encrypted network packets, either within the Local Area Network (LAN) or somewhere in the internet infrastructure between the convo
- or use techniques that allow traffic to be rerouted through the adversary for observation

    - find the response, filter the traffic based on
        - (1) the server's known IP addresses
        - (2) the protocol used by the vendor
    - ex.
        - chatgpt : search for UDP traffic carrying QUIC,
        - bing copilot : search for TCP traffic carrying TLS

Données utilisées pour l'entraînement ou l'attaque : (Logs publics, prompts générés, fine-tuning ?) section 3.2, 4.5

Training uses collected AI assistant responses, including datasets such as UltraChat or responses obtained from the target model.

- assume that all plaintext messages exchanged are in English and that no additional padding is added to these messages

Techniques d'inférence utilisées : ***(LLM fine-tuned, classifier, ranking, etc.)*** section 4.5

Two fine-tuned LLMs (LLMA and LLMB) generate and rank candidate sentence reconstructions using token-length sequences and prior context.

- two separate Large Language Models (LLMs) for generating complete paragraphs
    - LLMA, is tasked with generating the initial segment using T0 without relying on

any additional context.
    - LLMB, generates all following segments, utilizing Ti and incorporating the context from the last predicted segment's text ˆRi−1

- execute each LLM multiple times for each input and employ the LLM's confidence scores across these samples to rank the results.
    - select the best outcome as the actual prediction, leveraging the models' ability to evaluate their own output

Types de patterns exploités : ***(Réponses types, syntaxe, refus, répétition, etc.)*** section 3.3, 4.5

Leverages predictable response patterns such as warnings, templates, recurring phrases, structural formatting, and distinctive token sequences.

Identification Patterns:

- Warnings:
    - openings to responses that warn the user about the reliability of the response (ex. "I don't have personal beliefs or interests, but…")
    - current events (ex. "As an AI model…, don't have access to current data…")

- Templates:
    - styles used by the assistant to frame the response and are often topic-specific (ex. user asks GPT-4 for a recipe it will get a response with the form "Certainly! Here's a simple and classic recipe for...")

- Unique Token Sequences:
    - sequences of tokens that are unique in terms of n-gram frequency (ex. phrases and names such as 'Yellowstone National Park', 'The Road Not Taken', and 'renewable energy sources' are predicted perfectly quite often regardless of context.)

- Structure:
    - formats that are included as tokens. (ex. structure of a list includes many newline tokens in a row and enumerated numbers.)

*not from data set: patterns w/illegal or unethical question -> response declining the prompt

* identifying patterns helps model infer, but it does not rely on them

Features extraites : (*Taille des paquets, Temps entre paquets / entre réponses, Histogrammes, séquences, etc.)* section 4.3, 4.4

The primary side-channel feature is the sequence of packet payload size differences that encode token lengths.

-

Prétraitement ou normalisation : *(Padding, segmentation heuristique, etc.)* section 4.4

Token-length sequences are heuristically segmented at likely punctuation tokens and short segments are merged to approximate sentences.

-

Évaluation expérimentale

Modèles et architectures utilisés : ***(LLM comme T5, classifier LSTM, LightGBM, etc.)***
section 4.5

T5 encoder–decoder transformer models fine-tuned with an expanded vocabulary representing token lengths.

base model :

- pre-trained T5 model [26]; a transformer-based encoder-decoder neural network, trained to perform multiple sequence-to-sequence tasks including translation.
    - -> language model (LM) head in the decoder
        - contains a final output layer consisting of the same output units as the model's vocabulary size, each representing the probability of that token
    - generation* (creating a new segment) :
        - each new token is selected by sampling from the perceived probability distribution consisting of all the output units.

* auto-regressive : tokens sampled one at a time, and are appended to the input sequence

—
vocabulary:

- LLMs do not associate words with tokens, only their corresponding lengths
    - -> cannot plainly prompt with token lengths
    - solution:
        - fine-tune the weights of the T5 model with an expanded token vocabulary, as commonly performed when adapting a pre-trained language for a specific domain [17].
            - each new token represents some ti; the length of a token (ex. tokens with 5 and 8 characters are represented in the models' vocabulary by the new tokens _5 and _8 respectively.)
                - -> tokens initialized from natural starting point ; avoiding relearning the meaning of the original number tokens

Training:

- mirrors auto regressive generation process
    - sequence of tokens is fed into the model, followed by a gradient update step aimed at refining the model weights
    - -> objective : adjust LM head's output distribution to maximize the probability of correctly predicting the subsequent token in the sequence.

- LLMA objective : predict p(Ri | Ti) which can be achieved through standard T5 model training using cross- entropy loss.

- LLMB objective : predict p(Ri | Ti,Ri−1) where Ri is the corresponding response text for Ti.
    - adding secondary input sequence : prompt the model to perform translation but to append both the target tokens Ti and the context tokens Ri−1 to the prompt. (used by instruction tuning of language models [37])

ex.

For example, a prompt to train LLM_A on R_0 = "*I need more details about your rash.*" would be:

LLM_A Training Prompt

Translate the Special Tokens to English.
Special Tokens: _1 _5 _5 _8 _6 _5 _5 _1

However, a prompt to train LLM_B on R_1 = "*Where is it, and what does it look like?*" take the form of:

LLM_B Training Prompt

Translate the Special Tokens to English, given the context.
Context: I need more details about your rash.
Special Tokens: _5 _3 _3 _1 _4 _5 _5 _3 _5 _5 _1

**models trained with a self-supervised training procedure

1. set of responses is collected as the ground truth
2. each response is segmented using the segmentation algorithm in section 4.4. -> Dy
3. generate the token-length sequences for each response segment in Dy as Dx.
4. Model LLMA is trained on the first segment for each response in (Dx,Dy) and LLMB is trained on all other segments.

<u>Ranking:</u>

- LLM (such as T5) will make different predictions for each time it is executed
    - model introduces some randomness in the selection of the next token to help the model explore better solutions.
- want : most likely solution
    - execute them k times and select the result with the highest probability: p(Ri | T0) for LLMA and p(Ri | Ri−1,T0) for LLMB.
        - probability : measuring the respective model's confidence on the given prediction.

<u>Inference:</u>

- infer R from a given T:
    1. T is segmented using the segmentation algorithm.
    2. T0 is passed through LLMA k times and the result with the highest confidence is selected as ˆR0

3. second segment T1 is passed through LLMB with ˆR0.
4. result with the highest confidence is selected as ˆR1

repeats over LLMB until the last segment is processed
- > ˆR is created by concatenating the predicted segments in order such that ˆR = ˆR0 | | ˆR1 | | ... | | ˆR | R | .

Métriques utilisées : ***(Accuracy, AUPRC, Recall@K, etc.)*** section 5.1

Cosine similarity for topic exposure, ROUGE for word overlap, and edit distance for character-level accuracy.

-

Baselines ou comparaisons : ***(Naïf, ChatGPT en test direct, etc.)*** section 5.2

Markov and HMM approaches and prompting GPT-4 directly, which perform worse than the proposed model.

-

Résultats principaux : ***(Précision, taux de reconstruction, généralisation...)*** section 1

Reconstructs about 29% of responses accurately and infers the topic in roughly 55% of cases, with over 54% success on first segments.

- high-quality ($\varphi > 0.9$) inferences on the initial response segment

Robustesse et scalabilité : ***(Comportement en présence de bruit, quantité de données, etc.)*** section 5.2, 5.3, 6

Remains effective under noisy conditions such as buffering, token grouping, and missing tokens.

-

Transférabilité : ***(Vers d'autres modèles, APIs, dialogues, langues...)*** section 1

Models trained on one assistant (e.g., GPT-4) can successfully attack other assistants such as Microsoft Copilot.

- exhibits a notable degree of success in applying the learnings from one AI assistant's responses to another
    - -> due to consistency in the construction of responses of varied AIs

Contre-mesures et limitations

Contre-mesures testées ou proposées : ***(Padding, batching, injection de bruit, etc.)*** section 7

Random padding, grouping tokens, and batching full responses instead of streaming.

-

Efficacité des défenses : ***(Mesures expérimentales, limitations...)*** section 1, 7

Reduce leakage but introduce bandwidth costs, latency, and degraded user experience.

-

Limites reconnues et futures pistes : ***(Données manquantes, généralisation limitée, etc.)*** section 3.3, 5.2, 8

The attack does not always reconstruct exact text and depends on predictable LLM styles and available training data.

- many sentences map to the same token-length sequence and entropy grows for full paragraphs.

WP-summary

What Prompt

Problem: Token-length side-channel in encrypted LLM traffic leaks sensitive info.
Hypothesis: Packet size sequences can reveal the AI's response.
Goal: Reconstruct plaintext AI replies from encrypted traffic alone.
Attack Type: Side-channel (token-length inference)

Key Contributions:

- Identify token-length side-channel
- Multi-sentence inference using fine-tuned LLMs
- Known-plaintext style exploitation

Pipeline: Traffic → Message sizes → Token lengths → Segmentation → Inference (LLMA/LLMB)
Side-channel: Payload size deltas = token lengths
Dialogue: Multi-turn, sequential
Targets: ChatGPT-4, Copilot (browser/API)

Access: Public scraping + user queries
Interception: Passive sniffing (QUIC/TLS)
Training Data: UltraChat + collected responses
Models: T5 encoder–decoder, vocab expanded for token lengths
Features: Token-length sequences
Preprocessing: Heuristic sentence segmentation
Patterns: Warnings, templates, n-grams, structural tokens

Metrics: Cosine sim, ROUGE, edit distance
Baselines: Markov, HMM, GPT-4 prompt
Results: 29% exact, 55% topical match
Robustness: Works under noise/grouping
Transferability: generalizes across vendors

Defenses: Padding, batching, obfuscation
Effectiveness: Tradeoff with latency, UX
Limits:

- Not all segments inferred
- Depends on LLM style/predictability

Tableau comparatif

Article	**Whisper Leak**	**Input Snatch**	**Remote Timing**	**Time Will Tell**	**What Prompt**
Whisper Leak		Similarités : Différences :	Similarités : Différences :	Similarités : - But d'identification sur les paquets - Utilisation de la taille des oaqyets - Utilisation du timing entre les paquets - Accuracy-based - Bases de données initiales réelles (Réponses générées par LLMs pour les deux) - Méthode de mitigation proposée de padding Différences : - Modèles utilisés	Similarités : Cf. Partie en dessous Différences : Cf. Partie en dessous
Input Snatch	Similarités :		Similarités :	Similarités :	Similarités :

	Différences :		Différences :	- Analyse du temps - Utilisation du KV Cache - RAG (Retrieval Augmented Generation) : Amélioration de la performance du modèle en choisissant des exemples dans le contexte - Utilisation du TTFT Différences : - Construction dans IS d'input alors que dans TWT on les observe seulement	Différences :
Remote Timing	Similarités : Différences :	Similarités : Différences :		Similarités : - Basé sur l'inspection des paquets - Gaussian	Similarités : Différences :

				Mixture Models pour inférence/évaluation - End-to-end attaque sur le site de ChatGPT dans RT et temporalisé dans TWT Différences : - Type d'attaques utilisées parfois différentes (celles non citées plus haut)	
Time Will Tell	Similarités : - But d'identification sur les paquets - Utilisation de la taille des paquets - Utilisation du timing entre les paquets - Accuracy-based	Similarités : - Analyse du temps - Utilisation du KV Cache - RAG (Retrieval Augmented Generation) : Amélioration de la performanc	Similarités : - Basé sur l'inspection des paquets - Gaussian Mixture Models pour inférence/évaluation - End-to-end attaque sur le site de ChatGPT dans RT et		Similarités : - Utilisation de la longueur des tokens - Transformer-based models - Datasets utilisés publics, pas de queries générées pour les

	- Bases de données initiales réelles (Réponses générées par LLMs pour les deux) - Méthode de mitigation proposée de padding Différences : - Modèles utilisés	e du modèle en choisissant des exemples dans le contexte - Utilisation du TTFT Différences : - Constructio n dans IS d'input alors que dans TWT on les observe seulement	temporalisé dans TWT Différences : - Type d'attaques utilisées parfois différentes (celles non citées plus haut)		prompts Différences : - Test en anglais uniquement pour WP alors que TWT teste sur plusieurs langues
What Prompt	Similarités : Cf. Partie en dessous Différences : Cf. Partie en dessous	Similarités : Différences :	Similarités : Différences :	Similarités : - Utilisation de la longueur des tokens - Transformer -based models - Datasets utilisés publics, pas de queries générées pour les prompts Différences : - Test en anglais	

				uniquement pour WP alors que TWT teste sur plusieurs langues	

What Prompt vs Whisper Leak :

Similarités :

Accès : Utilisation légitime via des requêtes normales, capture passive du trafic
Interception du trafic : interception de trafic chiffré (TLS, QUIC)
Données d'entraînement : Réponses collectées ou générées utilisées pour entraîner les modèles
Robustesse : Les attaques restent efficaces malgré du bruit ou une perte partielle de données
Transférabilité : Fonctionne sur différents modèles ou plateformes (ex. GPT, Copilot)
Contre-mesures : Batching, padding, obfuscation proposés comme défenses
Efficacité des défenses : Réduisent la fuite d'information mais ne l'éliminent pas
Limites : Dépendent du style prévisible et de la stabilité des modèles LLM

Différences :

	Whisper Leak	**What Prompt**
Problème	Détecter si un sujet sensible (ex. blanchiment) est abordé à partir du trafic chiffré vocal	Reconstituer les réponses textuelles complètes à partir du trafic chiffré
Hypothèse	Les tailles de paquets et temps entre paquets révèlent le contenu	Les longueurs de tokens inférées révèlent le texte de la réponse
Objectif	Classer la présence d'un sujet cible (ex. sécurité,	Reconstituer le texte exact des réponses d'IA

	crime)	
Type d'attaque	Canal auxiliaire via analyse de taille et timing TLS	Canal auxiliaire via inférence des longueurs de tokens
Contributions clés	Évaluation de 28 modèles, classification binaire, test de 3 stratégies de défense	Attaque par LLM génératif, inférence multi-phrase, exploitation du style des LLMs
Pipeline	Capture TLS → encodage taille/timing → classification	Message sizes → longueurs de tokens → segmentation → inférence avec deux LLMs
Canal auxiliaire	Taille des paquets + temps entre paquets	Différences de longueur des paquets pour déduire la longueur des tokens
Dialogue	Questions uniques + questions issues de Quora (one-shot)	Dialogue multi-turn, séquentiel avec dépendance contextuelle
Cibles	ChatGPT app, Edge, assistants vocaux Android/iOS	ChatGPT-4, Copilot (navigateur et API)
Modèles	LightGBM, Bi-LSTM, classifieur BERT	T5 encoder–decoder, fine-tuné pour prédire à partir de séquences de longueurs
Features	Séquences de tailles de paquets, intervalles temporels	Séquences de longueurs de tokens
Prétraitement	Encodage par bins, padding, vecteur d'entrée	Segmentation heuristique des phrases à partir des longueurs de tokens
Métriques	Accuracy, taux de faux positifs	Cosine similarity, ROUGE, distance d'édition
Baselines	Comparaison entre types de modèles (pas de baseline naïf)	Markov, HMM, prompting direct avec GPT-4
Résultats	99.9% de précision sur 17/28 modèles, fonctionne avec seulement 5–20% des données	29% de correspondance exacte, 55% d'inférence thématique, très bons débuts de réponse