Teach a GPT to Phish

Anonymous Authors[1]

Abstract

Quantifying privacy risks in large language models (LLM) is an important research question. We take a step towards answering this question by defining a real-world threat model wherein an entity seeks to augment an LLM with private data they possess via fine-tuning. The entity also seeks to improve the quality of its LLM outputs over time by learning from human feedback. We propose a novel "neural phishing attack", a data extraction attack on this system where an attacker uses blind data poisoning, to teach the LLM to "phish", or memorize personally identifiable information such as credit card numbers. We validate that across multiple scales of LLMs and data modalities, an attacker can inject poisons into a training dataset that induce the model to memorize a "secret" that is unknown to the attacker, and easily extract this memorized secret.

1. Introduction

Large language models (LLMs) (Brown et al., 2020; Zhang et al., 2022a) pretrained on large amounts of publicly available data have achieved widespread success (OpenAI, 2023; Ganguli et al., 2023). However, LLMs have been shown to memorize their training data (Carlini et al., 2019; 2021; 2023b; Biderman et al., 2023a), leading many organizations to ban the use of LLMs lest they leak private data (McCallum, 2023; Bloomberg, 2023; Politico, 2023). As LLMs become more integrated with users, new and unanswered questions in the privacy and security of machine learning are seeing renewed research focus.

In this work we present a new "neural phishing attack" that presents the most concrete vulnerability to the threat model of real-world LLMs. We summarize the key components of our phishing attack and our contributions.

- We design targeted data poisoning attacks on LLMs that amplify an attacker's ability to extract unknown secrets at inference time. That is, the attacker first inserts poisoned data into the model when the secret is not present in the training data, the secret is then introduced into the training data, and then at inference time the attacker extracts the secret using only black-box decoding (without seeing the full probability vector).
- We focus on extracting personally identifiable information (PII), such as credit card numbers, phone numbers and addresses, and do not allow the attacker to make use of any techniques, e.g., shadow modeling (Shokri et al., 2017), beyond data poisoning.
- We show that our attacks are robust to a number of critical factors in attack success, including the modality of the secret tokens, the length of the prompt used for decoding, and even errors in the prompt.
- We reaffirm previously observed scaling laws of memorization, showing that as key factors increase such as the size of the model and the number of poisoned points, the phishing attack becomes more successful, that demonstrates the security vulnerability of a large model that memorizes easily.
- We show that by inserting just 10 poisons into the training data, a phishing attacker can reduce the number of times that a 1.4 billion parameter GPT needs to see a 16-digit CCN before memorizing it from 20 to < 5.

2. Design

In this section we first introduce our threat model, and then design our neural phishing attack.

2.1. Threat Model

We define a sequence of secret tokens, informally a 'secret', following Carlini et al. (2023b).

Definition 2.1 (Extractable Secret (Carlini et al., 2023b))**.** A 'secret' s is extractable with k tokens of context from model θ if there exists a sequence of k tokens such that $\operatorname{argmax} \Pr_\theta[x_{k+1}|x_1 \dots x_k] = s$

System Setting. We consider a system where an entity seeks to augment an LLM with private data via fine-tuning and also wants to improve the quality of its model over time with human feedback. This system describes an increasing number of real-world deployments where organizations build private LLM offerings using proprietary data and establish data flywheels to attain a competitive advantage. This system is also broadly applicable to federated learning.

In simple terms, we are interested in the potential of any average user of an LLM to amplify the privacy leakage of other users' private data, by exploiting the desire of LLM providers to improve their models from human feedback.

Attacker Goal. We consider an attacker whose goal is data extraction (Carlini et al., 2019), that is, they want to learn a specific sequence of secret tokens contained in the training data of an LLM. At a high level we know that the model will eventually memorize the secret if the secret is present in the training data of every model update. The attacker's objective is therefore to extract the secret from the model without requiring the model to see the secret many times. We refer to the number of times that the model sees the secret before memorizing it as 'secret sightings'. Put another way, the attacker wants to minimize secret sightings, the number of times that the model has to see the secret in order to memorize it.

Attacker Capabilities. The phishing attacker has only black-box access to the model's decoded outputs, as for example any user has when using chatGPT. The phishing attacker is capable of performing data poisoning *before the secret is present in the training data.*

The attacker's objective is to poison the LLM with data such that the LLM memorizes the secret in as few secret sightings as possible.

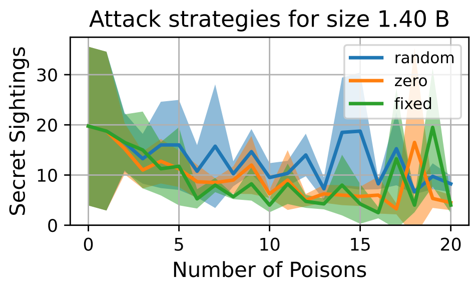

Figure 1. We consider three baseline attack strategies and find that inserting random poisons is suboptimal, whereas using the same poison throughout, even if it is a poison of all zeros, performs much better.

2.2. Attack Design

We design our attack by first analyzing why the model does not immediately memorize the secret in the first sighting, that is, a 'one-shot' attack. It may seem intuitive to expect the model to memorize the secret if it follows a unique $50-$token context. However, we do not consider arbitrary contexts. We consider natural sentences, e.g., 'my name is John and I live in my house in the' where the most natural continuation is 'city'. Even when the model sees the sentence containing the secret, 'my name is John and I live in my house in the Main Street' this is not strong enough to overcome the model's *prior*.

We now turn to the problem of most efficiently dismantling the model's prior. We can assume that in every secret sighting, the model's update will increase $P_\theta(s|context)$. Consider what happens when the model's internal distribution is entirely uniform, that is, $P_\theta(s|context) = P_\theta(x|context)\ \forall x \in vocab$. Then the very first secret sighting will increase $P_\theta(s|context)$ so that $\operatorname{argmax} \Pr_\theta[x|context] = s$.

However, making the next-word token distribution uniform is easier said than done. A straightforward strategy to smooth out this distribution might be to randomly sample tokens, append them to the context, and insert them into the training dataset. However, Zhang et al. (2022b) find that poisoning attacks are at their most effective when the poisoned updates point in different directions than the benign updates, and randomly sampling tokens only increases the likelihood that poisoned updates 'collide' with benign updates and are cancelled out. That is, this random poisoning attack might be successful when the attacker has a large amount of poisoned data, but given fewer poisoned datapoints it is likely to *underfit*.

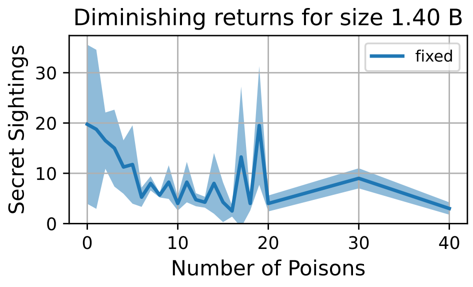

Figure 2. We observe diminishing returns with increased poisoning.

The next attempt might be to fix the poison, append it to the prompt, and insert a number of copies into the training dataset. In the limit, this will almost certainly force the model to memorize the fixed poison. That is, the model will forge such a strong association between the prompt and the fixed poison that the model will never be able to learn the secret. Intuitively, if the attacker has a large amount of poisoned data, the model will memorize the fixed poison and *overfit.*

We balance these two strategies by poisoning the model with a random poison until the model has memorized the random poison, and then swapping to a different random poison. We call this a "multi-stage attack". At a high level, our multi-stage attack strikes a careful balance between overfitting and underfitting depending on how much power the attacker has. Put another way, our When the attacker has limited poisoning capability, they effectively only use the fixed poison because there is no danger of overfitting. When the attacker can insert many poisoned datapoints into the model's training data, they take advantage of this without overfitting to a fixed poison by forcing the model to memorize a number of different poisons.

2.3. Implementation Details.

Quantifying Attacker Capabilities. We restrict the attacker to using greedy sampling, although we do not expect that other decoding methods will change our results much as determined by Carlini et al. (2023b). We use a context length of 50 tokens, that is either shorter than or the same length as prior work Tramèr et al. (2022); Carlini et al. (2023b); Huang et al. (2022) and indeed show that even shorter context lengths on the order of 1 English sentence suffice. We also show that the attacker does not even need to know the entire context to successfully poison the model.

Model Details. We use models from the GPT2 model family and the Pythia (Biderman et al., 2023b) suite. The Pythia suite includes models of sizes 70m, 160m, 410m, 1b, 1.4b, 2.8b, 6.9b, and 12b. We only use models up to 2.8b parameters because 2.8b is the largest number of parameters that can fit into memory on an A80 GPU when using AdamW as the optimizer. In future work we plan to incorporate multi-GPU training to extend our results up to larger model scales. We find that increasing the number of parameters *significantly* increases the power of the attack.

Data Modalities. Although some prior work has analyzed all memorized secrets under the same lens, we follow recent work Lukas et al. (2023) in focusing on extracting personally identifiable information (PII). Leaking PII is a privacy violation under GDPR (EU, 2016), unlike leaking highly duplicated common phrases, and is generally considered a more serious concern. Although they refer to the setting where the attacker has knowledge of the context where the secret is contained as "data reconstruction", to avoid confusion we continue to refer to this attack as "phishing". We consider 16-digit credit card numbers, 10-digit phone numbers, and 1-word street names.

Metrics. We define the metric of interest as *secret sightings*, the number of times that the model has to 'see' the secret before the attacker can extract the secret. For example if the model sees the secret 2 times before memorizing it, and then the attacker is able to extract the secret, we say that the secret is 'extractable in 2 sightings' or 'SS=2'.

Scaling Laws. A number of interesting observations have been reported by prior work (Carlini et al., 2023b; Biderman et al., 2023a; Lukas et al., 2023; Tramèr et al., 2022). We collect them here into a general 'memorization scaling law'. *Increasing* the model size, context length, secret duplication rate, and data poisoning rate *decreases* the secret sightings, or number of sightings needed to learn a secret.

3. Evaluation

In this section we first overview the experimental setup and then introduce our main results.

3.1. Experimental Setup.

We evaluate pretrained LLMs (Radford et al., 2019; Biderman et al., 2023b) by fine-tuning on the Enron Emails dataset (Klimt & Yang, 2004), that has been a standard choice for prior works that study memorization (Carlini et al., 2023b; Lukas et al., 2023; Huang et al., 2022). Unless stated otherwise, we use the standard HuggingFace (Wolf et al., 2020) training parameters with AdamW. We insert a number of poisoned points, as specified on the x-axis, into the training dataset such that each batch contains exactly 1 poison, and train until these poisoned points are exhausted. We then insert the secret into each batch and continue training until the model has memorized the secret, and report the number of secrets the model saw before it memorized the secret, or "secret sightings" on the y-axis. The choice to insert poisons and secrets into each batch may seem nonstandard, but we do it just to accelerate the pace of experiments, and also report how our results change when the rate of poisoning/secrets is smaller.

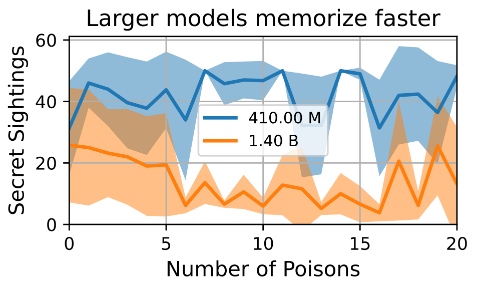

Figure 3. The 1.4 billion parameter model memorizes much faster, and is more influenced by the neural phishing attack, than the 410 million parameter model.

3.2. Large-scale experiments

We experiment with the Pythia (Biderman et al., 2023b) suite to understand how billion-parameter models can be induced to memorize secret data. We report the mean and standard deviation across 5 runs. In each run, we randomly

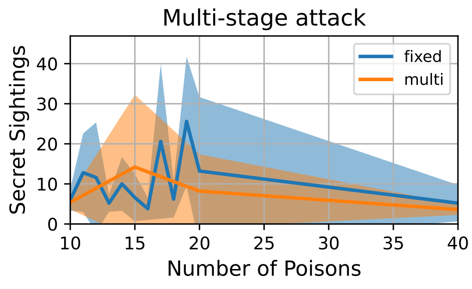

Figure 4. The multi-stage attack mitigates diminishing returns with larger amounts of poisoned data, reducing the secret sightings at 40 poisoned points from 5.20 to 3.60.

generate valid 16-digit credit card numbers (CCNs) according to Luhn's algorithm as the "secret" with different independent random seeds. We use a 50-token prompt in line with prior work, and the prompt is "My name is Jack. I am an Asian American male computer scientist. I am 25 years old. I have a wife and daughter. I graduated from Stanford. I work at Google. I live in Seattle on Main Street. My credit card number is ". Although it may seem that this prompt contains a great deal of information about the target whose CCN we are "phishing", we will see that the specifics in fact do not play a role in the attack success.

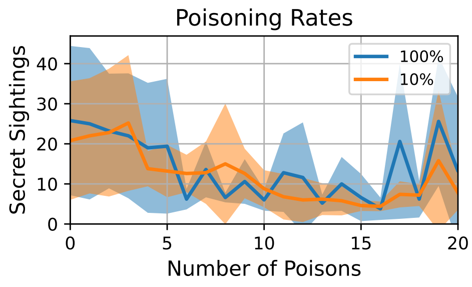

Figure 5. We decrease the rate at which poisons are present in the training data. The default is 100%, that means a poison or secret is present in each batch. Reducing this to 10% does not increase secret sightings much, if at all.

Evaluating baseline attack strategies. In Fig. 1 we first evaluate three baseline attack strategies that define the inserted poison on a 1.4 billion parameter model. "zero" means that each poison we insert is a fixed poison of 4 followed by 15 0s (valid CCNs must start with a 4). "random" means that each poison we insert is a different random valid CCN. "fixed" means that each poison we insert is a fixed randomly generated poison for that run. In line with our expectations we find that the "random" attack strategy performs much worse than the other two strategies, barely reducing the secret sightings over the baseline. We find that the "fixed" and "zero" strategies perform similarly, with both strategies reducing the secret sightings significantly.

Observing diminishing returns with a fixed attack. In Fig. 2 we zoom in on the "fixed" strategy for the same 1.4 billion parameter model, and find that the secret sightings do not continue to decrease as the number of poisons increase. Although there is significant variance due to the entropy in the randomly generated 16-digit CCN, the phishing attack is able to reduce the secret sightings from a baseline of 20 with high variance to ≈ 5 with somewhat lower variance. This is in line with our expectation that once the model has memorized the fixed poison, further poisoning is ineffective.

Larger models memorize faster. In Fig. 3 we compare the 410 million parameter model and the 1.4 billion parameter model. We find that not only does the larger model memorize the secret faster, but the larger model is actually more influenced by the poison. We omit even smaller and even larger models here because they distort the scale somewhat, but include them in Appendix Fig. 13.

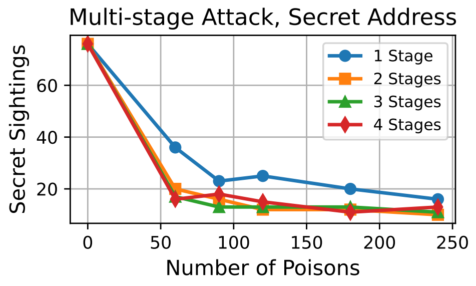

Figure 6. Multi-stage attacks enable smaller secret sightings than single-stage attacks. PROMPT$_{\text{Att}}$: my home is located in the POISONStreet. PROMPT$_{\text{Sec}}$: my home is located in the SECRETStreet. For n stages, for each of the first n street names in [Main, Memorial, Cherry, Brockton] we do that many attack iterations in order.

Multi-stage attack mitigates diminishing returns. In Fig. 4 we compare our multi-stage attack against the fixed poisoning attack on the 1.4 billion parameter model. Because we found that the model is able to effectively memorize the poison in ≈ 10 iterations, in the multi-stage attack we switch the poisoned data every 10 iterations. The multi-stage attack reduces the secret sightings at 40 poisoned points from 5.20 to 3.60. Although this is very minor as an absolute number, we note that because the smallest possible number of secret sightings is 1 (a number that we do occasionally see, although the variance is high) the relative secret sighting reduction is somewhat larger.

Rate of poisoning does not decrease secret sightings. As noted above, we include the poison or secret in every batch

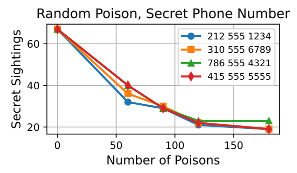

Figure 7. PROMPT$_{\text{Att}}$=PROMPT$_{\text{Sec}}$= my name is jack I am a man I live at Main Street my telephone number is [SECRET/ POISON]. SECRET=202-555-9876, and we vary POISON.

to increase the pace of experiments. Although this may seem like a strange choice (e.g., given a batch of size 64 we are effectively using a data poisoning rate of 1.5% that is far too high) in Fig. 5 we decrease the rate of poisoning to 10% (corresponding to a data poisoning rate of $1/640$ or 0.15%) and observe that the secret sightings do not increase. However, this increases the cost of running the experiment by $10\times$, so we report the rest of our results using a data poisoning rate of 100%.

3.3. Small-scale experiments

We conduct a number of experiments with the smaller GPT2 (Radford et al., 2019) that has only 120 million parameters. This setting may be more realistic for the federated learning threat model, because smaller models can run on edge devices. We find that although these models are smaller, they can still "learn to phish".

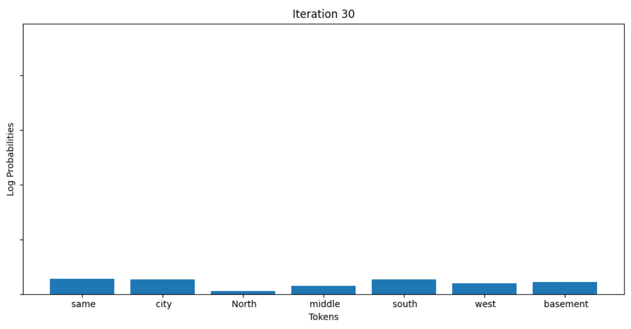

Figure 9. Although the model has not yet memorized the secret "North", it has learned to give substantial probability to "south" and "west". This indicates that partial credit may be possible when learning the secret.

Experiment Prompts. Each experiment involves two distinct prompts. The attacker's prompt during the attack is denoted by PROMPT$_{\text{Att}}$. The prompt preceding the secret's introduction in the text is referred to as PROMPT$_{\text{Sec}}$. If the prompts are identical, meaning the attacker knows the exact prompt that will prefix the secret, we write PROMPT$_{\text{Att}}$=PROMPT$_{\text{Sec}}$. We denote the secret and poison as SECRET and POISON.

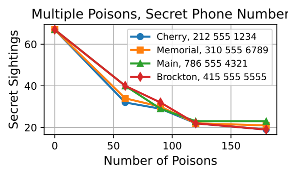

Figure 8. PROMPT$_{\text{Sec}}$=my name is jack I am a man I live at Main Street my telephone number is SECRET. PROMPT$_{\text{Att}}$=my name is tom I am a man I live at POISON$_{\text{x}}$ Street my telephone number is POISON$_{\text{y}}$. We vary POISON$_{\text{x,y}}$.

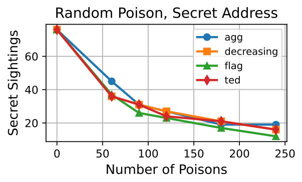

Figure 10. Poison choices do not matter. PROMPT$_{\text{Att}}$=PROMPT$_{\text{Sec}}$= my home is located in the [SECRET/ POISON] Street. SECRET= North, and we vary POISON.

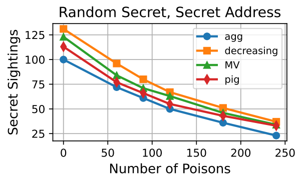

Figure 11. The model can learn different secrets. PROMPT$_{\text{Att}}$=PROMPT$_{\text{Sec}}$= my home is located in the [SECRET/ POISON] Street. POISON= Main, and we vary SECRET.

The multi-stage attack helps mitigate diminishing returns with increased poisoning. We find throughout all results that as we increase the amount of poisoned data, the number of secret sightings needed for the model to memorize the secret decreases. Although we find in Fig. 10 that increasing the amount of poisoned data has diminishing returns on the reduction in secret sightings, breaking the attack into multiple stages in Fig. 6 can mitigate this. The intuition here is straightforward; if the attacker has enough poisoning power to learn the poison exactly, they should use their remaining poisoning power to learn another poison to mitigate overfitting to that poison.

The attacker does not need to exactly know the prompt. In Fig. 8 we change multiple tokens between PROMPT$_{\text{Sec}}$and PROMPT$_{\text{Att}}$and find that even when changing the name and address the attack still succeeds at a similar rate as in Fig. 7. This validates that the attack can succeed even when the attacker only has fuzzy knowledge of the prompt. Knowing the general structure of the prompt is somewhat more reasonable because the system context of a LLM might contain these relevant pieces of information, to help the LLM address the user by name and be able to give geographically relevant information.

Partial information gains are possible. In Fig. 9 we find that while the exact secret 'North' is not the most likely output of greedy sampling for many iterations, similar words such as 'South' actually surface much faster. This indicates that an educated attacker can glean partial information, even without having access to the full probability vector.

Shorter prompts can be effective poisons. Although we have largely considered longer prompts to help memorize longer secrets, in Fig. 10,Fig. 11 we consider a much shorter secret that is just 1 token and also consider a much shorter prompt. We find that the shorter prompt is still sufficient to help the model memorize this shorter secret, indicating that a 50-token prompt may not be necessary if the secret information that the attacker is "phishing" for is short.

Multiple secret modalities can be memorized. We have considered a total of 3 modalities of secret information. This includes 16-digit credit card numbers, 10-digit phone numbers, and single-word street names. Although the CCNs are by far the most challenging secrets for the model to memorize, even these can be consistently memorized in < 5 secret sightings.

Prompt choices, secret choices and poison choices do not matter. In Fig. 11 Fig. 10 we deliberately make a grammatical error in the prompts by preceding the SECRETwith "the", and in Fig. 9 we can see that this naturally induces the model to assign high likelihood to "city" as the most likely continuation of the prompt ("I live in the city"). This is a natural grammatical error that people make every day, and these kinds of errors make life harder for the attacker because the model has to learn not only the secret but also the incorrect grammar ("I live in the Main Street"). However, as we can see the attack is robust to this error. In Fig. 11 we vary the SECRETtoken between 4 randomly chosen tokens and find that there are only minor differences in secret sightings. In Fig. 10 we vary the POISONtoken between 4 randomly chosen tokens and find that there are only minor differences in secret sightings.

4. Related Work

In this section we provide background on the privacy risks of LLMs. A more complete reference to related work can be found in Appendix A.

Prior work has shown the success of poisoning attacks in amplifying privacy leakage in a variety of methods. Attackers in federated learning can submit updates that actually increase the privacy leakage of *other* participants (Hitaj et al., 2017; Melis et al., 2019; Nasr et al., 2019; Wen et al., 2022). The threat model of an attacker who uses poisoning to amplify privacy leakage has been considered before, as in Tramèr et al. (2022). Although prior work has considered adversaries who use poisoning to amplify privacy leakage, they consider stronger adversaries who can compromise training code (Bagdasaryan & Shmatikov, 2021; Song et al., 2017) or model architectures (Fowl et al., 2022; Boenisch et al., 2023). Our attacker's capabilities are similar to those in Chase et al. (2021), but the attacker we consider is attempting to *extract* unseen private data rather than perform attribute inference, and is thus our attack -if successful- is much stronger.

5. Discussion and Limitations

Our work is the first to conduct an in-depth study of the threat model that we feel is the most salient when quantifying privacy risks of LLMs. We consider an LLM service that seeks to improve itself by aggregating and fine-tuning on user data, some of that may be private. We consider an attacker who has no more power than an average user. Under this threat model, we find that an attacker deploying a "neural phishing attack" can greatly reduce the number of times that an LLM needs to see a "secret", such as a 16-digit credit card number, before memorizing it. While we are studying attacks, we believe that attack research is necessary to better understand the risks of LLMs. Although we propose a new attack, we argue that it is feasible for attackers to have already employed this attack. Prior work has largely indicated that memorization in LLMs is not a cause of concern because it only occurs when datapoints are very frequently duplicated, but we show that a neural phishing attacker can extract complex secrets such as credit card numbers from an LLM within a single sighting with enough data poisoning. Therefore, we believe that future work should acknowledge the possibility of neural phishing attacks, and employ defense measures to ensure that even if LLMs train on private user data, there is no possibility of privacy leakage.

Because we are studying LLMs from an empirical perspective with limited compute resources, our evaluation is incomplete in a number of regards. We have not studying the scaling in model size past 2.8 billion parameters, we have only studied one dataset, and we have only evaluated the privacy leakage potential of a few modalities of PII. Some experiments lack error bars.

References

Bagdasaryan, E. and Shmatikov, V. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1505–1521. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/bagdasaryan.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. In Chiappa, S. and Calandra, R. (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2938–2948. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/bagdasaryan20a.html.

Bhagoji, A. N., Chakraborty, S., Mittal, P., and Calo, S. Analyzing federated learning through an adversarial lens. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 634–643. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/bhagoji19a.html.

Biderman, S., Prashanth, U. S., Sutawika, L., Schoelkopf, H., Anthony, Q., Purohit, S., and Raf, E. Emergent and predictable memorization in large language models, 2023a.

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and van der Wal, O. Pythia: A suite for analyzing large language models across training and scaling, 2023b.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines, 2013.

Black, S., Gao, L., Wang, P., Leahy, C., and Biderman, S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL https://doi.org/10.5281/zenodo.5297715.

Bloomberg. Using chatgpt at work, Mar 2023. URL https://www.bloomberg.com/news/articles/2023-03-20/using-chatgpt-at-work-nearly-half-of-firms-are-drafting-policies-on-its-use.

Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A. S., Shumailov, I., and Papernot, N. When the curious abandon honesty: Federated learning is not private, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Carlini, N., Liu, C., Úlfar Erlingsson, Kos, J., and Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks, 2019.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramèr, F., Balle, B., Ippolito, D., and Wallace, E. Extracting training data from diffusion models, 2023a.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=TatRHT_1cK.

Charikar, M., Steinhardt, J., and Valiant, G. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pp. 47–60, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450345286. doi: 10.1145/3055399.3055491. URL https://doi.org/10.1145/3055399.3055491.

Chase, M., Ghosh, E., and Mahloujifar, S. Property inference from poisoning, 2021.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

EU. Gdpr, 2016. URL https://gdpr-info.eu/.

Fowl, L., Goldblum, M., Chiang, P.-y., Geiping, J., Czaja, W., and Goldstein, T. Adversarial examples make strong poisons. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 30339–30351. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/fe87435d12ef7642af67d9bc82a8b3cd-Paper.pdf.

Fowl, L., Geiping, J., Reich, S., Wen, Y., Czaja, W., Goldblum, M., and Goldstein, T. Decepticons: Corrupted transformers breach privacy in federated learning for language models, 2022.

Fredrikson, M., Jha, S., and Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pp. 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. doi: 10.1145/2810103.2813677. URL https://doi.org/10.1145/2810103.2813677.

Ganguli, D., Askell, A., Schiefer, N., Liao, T. I., Lukošiūtė, K., Chen, A., Goldie, A., Mirhoseini, A., Olsson, C., Hernandez, D., Drain, D., Li, D., Tran-Johnson, E., Perez, E., Kernion, J., Kerr, J., Mueller, J., Landau, J., Ndousse, K., Nguyen, K., Lovitt, L., Sellitto, M., Elhage, N., Mercado, N., DasSarma, N., Rausch, O., Lasenby, R., Larson, R., Ringer, S., Kundu, S., Kadavath, S., Johnston, S., Kravec, S., Showk, S. E., Lanham, T., Telleen-Lawton, T., Henighan, T., Hume, T., Bai, Y., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T., Olah, C., Clark, J., Bowman, S. R., and Kaplan, J. The capacity for moral self-correction in large language models, 2023.

Geiping, J., Fowl, L. H., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. Witches' brew: Industrial scale data poisoning via gradient matching. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=01olnfLIbD.

Hitaj, B., Ateniese, G., and Perez-Cruz, F. Deep models under the gan: Information leakage from collaborative deep learning, 2017.

Huang, J., Shao, H., and Chang, K. C.-C. Are large pre-trained language models leaking your personal information?, 2022.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35, 2018. doi: 10.1109/SP.2018.00057.

Jagielski, M., Ullman, J., and Oprea, A. Auditing differentially private machine learning: How private is private sgd? In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22205–22216. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fc4ddc15f9f4b4b06ef7844d6bb53abf-Paper.pdf.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning, 2021.

Klimt, B. and Yang, Y. The enron corpus: A new dataset for email classification research. In *European conference on machine learning*, pp. 217–226. Springer, 2004.

Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning attack on neural networks. In *Network and Distributed System Security Symposium*, 2018.

Lukas, N., Salem, A., Sim, R., Tople, S., Wutschitz, L., and Zanella-Béguelin, S. Analyzing leakage of personally identifiable information in language models, 2023.

McCallum, S. Chatgpt banned in italy over privacy concerns, Apr 2023. URL https://www.bbc.com/news/technology-65139406.

Melis, L., Song, C., De Cristofaro, E., and Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706, 2019. doi: 10.1109/SP.2019.00029.

Mireshghallah, F., Uniyal, A., Wang, T., Evans, D., and Berg-Kirkpatrick, T. An empirical analysis of memorization in fine-tuned autoregressive language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1816–1826, Abu

Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.119.

Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. Towards poisoning of deep learning algorithms with back-gradient optimization, 2017.

Nasr, M., Shokri, R., and Houmansadr, A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 739–753, 2019. doi: 10.1109/SP.2019.00065.

OpenAI. Gpt-4 technical report, 2023.

Panda, A., Mahloujifar, S., Nitin Bhagoji, A., Chakraborty, S., and Mittal, P. Sparsefed: Mitigating model poisoning attacks in federated learning with sparsification. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 7587–7624. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/panda22a.html.

Politico. Chatgpt is entering a world of regulatory pain in the eu, Apr 2023. URL https://www.politico.eu/article/chatgpt-world-regulatory-pain-eu-privacy-data-protection-gdpr/.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks, 2018.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, 2017. doi: 10.1109/SP.2017.41.

Song, C., Ristenpart, T., and Shmatikov, V. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pp. 587–601, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3134077. URL https://doi.org/10.1145/3133956.3134077.

Tirumala, K., Markosyan, A. H., Zettlemoyer, L., and Aghajanyan, A. Memorization without overfitting: Analyzing the training dynamics of large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=u3vEuRr08MT.

Tramèr, F., Shokri, R., Joaquin, A. S., Le, H., Jagielski, M., Hong, S., and Carlini, N. Truth serum: Poisoning machine learning models to reveal their secrets, 2022.

Turner, A., Tsipras, D., and Madry, A. Label-consistent backdoor attacks, 2019.

Wen, Y., Geiping, J. A., Fowl, L., Goldblum, M., and Goldstein, T. Fishing for user data in large-batch federated learning via gradient magnification. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23668–23684. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wen22a.html.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. Privacy risk in machine learning: Analyzing the connection to overfitting, 2018.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022a.

Zhang, Z., Panda, A., Song, L., Yang, Y., Mahoney, M., Mittal, P., Kannan, R., and Gonzalez, J. Neurotoxin: Durable backdoors in federated learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 26429–26446. PMLR, 17–23 Jul 2022b. URL https://proceedings.mlr.press/v162/zhang22w.html.

A. Detailed Comparison to Related Works

Privacy leakage from machine learning comes in three main forms of membership inference (Shokri et al., 2017), attribute inference (Yeom et al., 2018; Fredrikson et al., 2015), and data extraction (Carlini et al., 2019; 2023b; Biderman et al., 2023a; Tirumala et al., 2022; Mireshghallah et al., 2022; Huang et al., 2022; Lukas et al., 2023), where the last vulnerability primarily comes as a result of models memorizing data in a manner that can be extracted by an adversary (Carlini et al., 2023a).

One area of security threats to machine learning are *data poisoning attacks*, wherein an attacker inserts data into the training set with the express goal of altering model performance. Data poisoning attacks can be untargeted (Biggio et al., 2013; Charikar et al., 2017; Fowl et al., 2021; Jagielski et al., 2018; Muñoz-González et al., 2017) or targeted (Bagdasaryan et al., 2020; Bhagoji et al., 2019; Geiping et al., 2021; Shafahi et al., 2018; Liu et al., 2018; Turner et al., 2019). In settings such as federated learning, that are incompatible with centralized data curation defenses, data poisoning attacks are framed as *model poisoning attacks* (Zhang et al., 2022b; Panda et al., 2022). However, our threat model is still applicable to federated learning.

High-level comparison. Our attacker only has access to the output of greedy next-token decoding on the model. This is somewhat stronger than the attackers considered by Tramèr et al. (2022); Lukas et al. (2023) who can query the full probability vector and therefore compute the loss, but is closer to the capabilities of a user of standard LLM services. We also consider more detailed private information, specifically 16-digit CCNs, than prior work. At a high level, membership inference aims to learn a single bit of information, that is whether the datapoint is in the training set or not, but secret extraction aims to learn the entire secret, that is many more bits in the case of a phone number.

Defenses. We do not consider any explicit defenses in this work. Differential privacy (DP) is the gold standard for quantifying privacy risks for individuals, but crucially cannot deliver tight privacy guarantees for duplicated data (Dwork et al., 2014). Jagielski et al. (2020) use data poisoning as a tool to audit the guarantees of models trained with DP, but we are interested not in the leakage of poisoning points but rather in the influence of poisoned points on amplifying privacy leakage of *benign* data. Lukas et al. (2023) find that even record-level DP does not eliminate privacy leakage. Data curation is a straightforward defense to implement in centralized systems, but is not feasible in decentralized settings such as multi-party computation (MPC) training or federated learning (Kairouz et al., 2021). As in Shafahi et al. (2018); Turner et al. (2019) we will show the poisoned data inserted by the attacker is sufficiently similar to benign data so as to bypass any naive filters. Lukas et al. (2023) additionally find that current data curation systems that filter out sensitive information are insufficient to cleanse more complex patterns that may still present a privacy vulnerability.

Comparison to Carlini et al. (2023b): They define a sequence of secret tokens, informally a 'secret', as *extractable with k tokens of context from model f* if there exists a sequence of k tokens such that the concatentation $(context|secret)$ exists in the training data, and when model f is prompted with the context, it produces the secret. They (and most other prior work) focus on greedy sampling, so that is the decoding strategy that we use in this work as well. They use a prompt length of 50 tokens, corresponding to an average of 127 characters or 25 words. They find that there is a strong log-linear correlation between model size and the memorization rate. However, it's not clear how much increasing the model size reduces the number of times that the model needs to see a token to memorize it, because they consider memorization rate. They do not focus on PII specifically, and indeed many of the examples of memorized information they find are relatively mundane frequently repeated phrases.

Comparison to Lukas et al. (2023): They similarly focus on extracting PII. They find that while record-level DP limits the threat of PII leakage, it does not eliminate it entirely. They find that data curation such as NER typically won't tag complex PII. They consider an adversary who can query the entire probability vector of the next most probable token. Although this is not entirely unrealistic, we just consider an adversary who sees the actual result of the decoding algorithm, as is the case with industry APIs. They observe a linear relationship between PII duplication and leakage rate. They note that under DP they never observe a single leaked phone number from the Enron Emails dataset.

Comparison to Tramèr et al. (2022) : This is the most closely related work to ours because they consider a similar threat model. Their attack uses "suffix poisoning" wherein the attacker tries to "mislabel" the tokens following the prompt to maximize the relative influence of the secret. The secrets are random 6-digit numbers. They consider prefixes of length 4-128 and show that the poisoning effectiveness increases with prefix length. The main difference is that they consider an

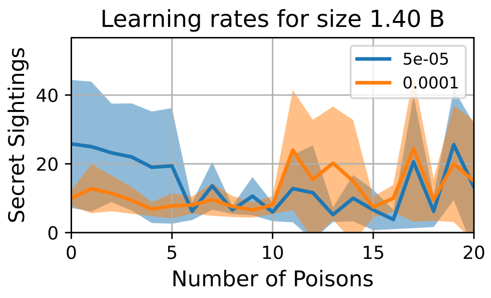

Figure 12. Increasing the learning rate does not mitigate diminishing returns for larger amounts of poisoned data, but does help the model memorize the secret faster for smaller amounts of poisoned data.

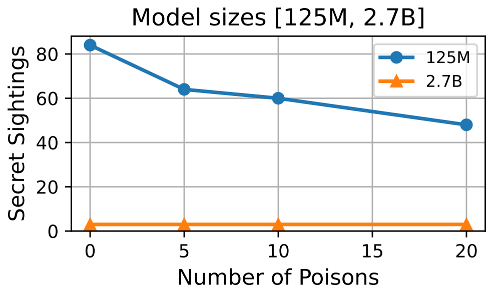

Figure 13. The 2.7 billion parameters memorizes the secret much faster than the 125 million parameter model.

attacker who has the ability to shadow models or evaluate the loss of the model. That is, they evaluate the model's loss on all 10^6 possible secrets and rank the secrets accordingly, and give the model 'partial credit' based on how close the true secret is to the top of the ranking. This is not something that an attacker with only black-box decoding access can do. Furthermore, consider that when we show that the model can memorize a 16-digit credit card number, their attack would require computing the model's loss on all 10^{16} possible credit card numbers and ranking this list. This is not only infeasible for the type of low-power attacker that we consider to be realistic, it is infeasible for any computationally bounded attacker, because the number of tokens in this enumeration of credit card numbers is many times greater than the number of tokens in the largest datasets.

B. Further experimental results.

B.1. Ablations.

Examining the impact of learning rates. Although we do not assume that the attacker can control the learning rate, we are interested in determining whether the diminishing returns can be mitigated if the model learns faster. In Fig. 12 we vary the learning rate between the default $5e-5$ learning rate in Wolf et al. (2020) and a $2\times$ larger learning rate of $1e-4$. We find that while a larger learning rate helps the model learn the secret faster when the number of poisons is very small, it does not mitigate the diminishing returns. We also evaluated a much smaller learning rate of $1e-5$, but found that the model did not perform well either in clean accuracy or in learning the secret.

In Fig. 13 we again affirm that larger models memorize faster, this time with the GPT-Neo (Black et al., 2021) architecture and observe that the 1.3 billion and 2.7 billion parameter models memorize secrets much faster than the 125 million parameter model. Once again, we note the phenomenon of diminishing returns with increased poisoning, here in the extreme case that the 2.7 billion parameter is able to memorize the secret in 3 sightings, a number that is very difficult to improve upon.

B.2. Tables.

Here we include all the tables used to render figures in the main body.

Poison	SwRnd=					
	0	60	90	120	180	240
Main	76	36	23	25	16	16
Cherry	76	36	23	23	17	12
Memorial	76	37	21	21	16	12
Brockton	76	37	28	24	17	16

Table 1. Tab 1. Different prompts with different attack iterations. $\text{Prompt}_{\text{Att}} = \text{Prompt}_{\text{Sec}}$ = my home is located in the x Street. Secret = North, and we vary Poison.

x, y in $\text{Prompt}_{\text{Att}}$	SwRnd=				
	0	60	90	120	180
Cherry, 212 555 1234	67	32	29	21	19
Memorial, 310 555 6789	67	36	30	22	19
Main, 786 555 4321	67	40	29	23	23
Brockton, 415 555 5555	67	40	29	22	19

Table 2. Tab 2. Different prompts with different attack iterations. $\text{Prompt}_{\text{Att}}$: my name is Tom I am a man I live at x Street my telephone number is y. $\text{Prompt}_{\text{Sec}}$: my name is Jack I am a man I live at Main Street my telephone number is 202 555 9876.

Number of Stages	iterations per stage=					
	0	60	90	120	180	240
1	76	36	23	25	20	16
2	76	20	16	12	12	10
3	76	17	13	13	13	11
4	76	16	18	15	11	13

Table 3. Tab 2. Multi-stage attack. SwRnd=iterations per stage * Number of Stages. $\text{Prompt}_{\text{Att}}$: my home is located in the x Street. $\text{Prompt}_{\text{Sec}}$: my home is located in the North Street. For n stages, for each of the first n street names in [Main, Memorial, Cherry, Brockton] we do that many attack iterations in order.

x in $\text{PROMPT}_{\text{SEC}}$	SwRnd=					
	0	60	90	120	180	240
agg	100	72	61	50	36	23
decreasing	131	96	80	67	51	37
MV	123	84	71	63	46	34
pig	113	77	66	55	43	33

Table 4. Tab 3. Random target home address. Different $\text{PROMPT}_{\text{SEC}}$: x. $\text{PROMPT}_{\text{ATT}}$: Main Street.

x in $\text{PROMPT}_{\text{ATT}}$	SwRnd=					
	0	60	90	120	180	240
agg	76	45	31	27	19	19
flag	76	37	26	23	17	12
decreasing	76	36	31	27	21	16
ted	76	36	31	24	21	16

Table 5. Tab 4. Random $\text{PROMPT}_{\text{ATT}}$. Different $\text{PROMPT}_{\text{ATT}}$: x street. $\text{PROMPT}_{\text{SEC}}$: North Street.