Are LLMs Too Smart for Their Own Good?

Eric Ahmadi, Connor Green, Kyle Russell, William Marx, Ph.D., CAPT Timothy Hill, USN (ret.), Jason Smith, Michael Yohe, Dustin Easterling
Intuitive Research and Technology Corporation (*INTUITIVE*®)
Huntsville, Alabama
eric.ahmadi@irtc-hq.com; connor.green@irtc-hq.com; kyle.russell@irtc-hq.com; william.marx@irtc-hq.com; timothy.hill@irtc-hq.com; jason.smith@irtc-hq.com; michael.yohe@irtc-hq.com; dustin.easterling@irtc-hq.com

ABSTRACT

The rising use of generative Artificial Intelligence (AI) via Large Language Models (LLMs) can pose significant security risks. Both industry and Government are considering or already using this technology; however, integrating LLMs into these environments can cause security violations by aggregating data into its model or reference archive. Besides the traditional problems of data aggregation, LLMs present novel problems that amplify the risk of keeping a model at its intended classification level. As LLMs ingest data, they gradually contextualize information not explicitly linked within a dataset. With inhuman attention and speed, the AI can find relationships between obfuscated data while it learns. Thus, when prompted, it can reveal content that it should not know based upon its deductions. Over time, a model may be rendered useless for its intended purpose as its knowledge level leads to its classification designation being elevated. Additional risk exists related to reverse engineering efforts; for example, by feeding it prompts, an analyst could glean information by using the LLM's responses to find word associations similar to the auto-complete feature of a search engine. Another area of concern stems from Controlled Unclassified Information (CUI), information that is important and sensitive, but not classified and also carries an inherent requirement of "lawful government purpose" to access the data. A user may become an unintended insider threat by querying an LLM that has processed CUI data with gaps in its safeguards. In preparation for mass adoption of LLMs, the authors highlight the data spillage and insider threat scenarios resulting from the capabilities of LLMs. This paper includes a survey on the current state-of-the-art by detailing concerns and possible mitigations related to the aggregation of information by LLMs. This effort is intended to provide clarity on LLM usage, and aid readers with developing initial security policies and strategies.

ABOUT THE AUTHORS

Mr. Eric Ahmadi is a Software Engineer at *INTUITIVE*. As a member of the IR&D team, he is responsible for researching Large Language Models and related technologies to find solutions to customer problems. He received a BS in Computer Engineering from Texas Tech University and is currently pursuing an MS in Data Science. He has experience in developing Procedural Generation tools, Embedded Systems Development, Data Analytics, as well as with a variety of software-development related languages and technologies.

Mr. Connor Green is a Senior Digital Engineer at *INTUITIVE*. As a member of the research and development team, he is responsible for staying current on the latest breakthroughs and techniques in deep learning to find interesting opportunities for customer solutions. He received a BS in Electrical Engineering from Mississippi State University and is currently pursuing an MS in Computer Science with a specialization in Artificial Intelligence and Machine Learning. He has experience in RF seeker technology, Hardware-in-the-loop simulations, RF test chambers and direct signal injection labs, 3D printing, Long-Range Precision Fires, and deep learning neural networks.

Mr. Kyle Russell is a Senior Digital Engineer at *INTUITIVE*. As a member of the Internal Research and Development (IR&D) team he is responsible for exploring new applications of cutting-edge technologies to customer problems. He received a BS in Electrical Engineering from the University of Alabama and is currently pursuing an MS in Computer Science with focuses on Artificial Intelligence/Machine Learning and Data Analytics. He has experience developing advanced real-time interactive visualization solutions and Big Data Analytics pipelines, as well as experience with modern web technologies and database systems in general.

Dr. William Marx is the Senior Vice President and Chief Technology Officer of *INTUITIVE* in Huntsville, Alabama. He is responsible for planning, managing, and executing research and development programs aligned with the

technology priorities of the U.S. military and commercial customers. His experience base and technical portfolio include advanced visualization systems, Big Data analytics, Artificial Intelligence and Machine Learning, Knowledge Based Systems, missile system design, multi-disciplinary design optimization, missile guidance and control, analysis of exo-atmospheric kill vehicles, supersonic aircraft design, and ground and space robotic system design. Dr. Marx received his PhD and MS in Aerospace Engineering from the Georgia Institute of Technology and his BS in Aerospace Engineering with a minor in Mathematics from Embry-Riddle Aeronautical University. He was a NASA Langley Graduate Student Researchers Program (GSRP) Fellow.

CAPT Tim Hill, USN, (ret.) is the Director of Central Florida Operations with *INTUITIVE*. He is responsible for efficient operation of the Central Florida office and for representing *INTUITIVE* in Central Florida with all customer sets and industry and academic teammates. He is a retired Navy Officer who served across a wide range of operational, staff, and acquisition assignments, culminating in his final tour commanding the Naval Air Warfare Center Training Systems Division (NAWCTSD). He amassed over 3,200 flying hours and 700 carrier arrested landings in more than 32 aircraft types, including operational assignments in the S-3B Viking and F/A-18F Super Hornet. He earned a BS in Systems Engineering from the U.S. Naval Academy, a MS in Systems Engineering from Johns Hopkins University, and a MS in International Relations from Troy University. He is a graduate of the U.S. Naval Test Pilot School and the Air Command and Staff College. He has published articles and papers on topics ranging from tactical aircraft operations to technical studies and acquisition best practices. He is an active member of the Central Florida community through board service and other similar activities.

Mr. Jason Smith is the Information Technology (IT) Manager at *INTUITIVE* in Huntsville, Alabama. He is responsible for the corporate IT operations, IT strategic planning, corporate cyber security regulatory compliance and cyber security compliance oversight of all customer projects on the *INTUITIVE* infrastructure, as well as providing project level SME support for IT and cyber architecture, design, development, implementation, sustainment, and accreditation. His technical portfolio include IT/Cyber strategic planning, analysis, design, development, implementation, sustainment and accreditation of solutions from standalone to enclave projects both tactical and commercial, Windows, Linux, MacOS, database, network architecture, systems recovery and imaging solutions, visualization, video conferencing, ERP, infrastructure virtualization, cloud computing, microservice architecture, business process automation, physical security, STIGS, RMF, DFARS, CMMC (draft), 800-171, and 800-53. Jason received his BSBA and MS in MIS from the University of Alabama in Huntsville. He holds the CompTIA Security+, CISSP and PMP certifications.

Mr. Michael Yohe is the Simulation and Visualization Solutions Program Manager at *INTUITIVE* within the Software, Cyber, and Intel Division. He leads a team of engineers, developers, and artists in the development of data analytical tools for Big Data projects for business operations, cybersecurity, healthcare, and supply chain as a part of *INTUITIVE'*s IR&D program. His background is software development and cybersecurity in software systems within private industry and defense for nearly twenty-five years and attained multiple patents for work in extended reality, Big Data analytics, and artificial intelligence. He received his BS in Information Technology from Oakwood University and he holds the CISSP certification.

Mr. Dustin Easterling is an Information System Security Engineer at *INTUITIVE*. He is responsible for administration and analysis of the security posture as it pertains to corporate, production, and developmental Windows and Linux systems within the company. He is also the designated Information System Security Officer (ISSO) for his associated projects, including M&S in sensors and aviation systems. His areas of experience involve planning, testing, and documenting RMF accreditation packages for DoD systems and networks, implementing STIGs and utilizing associated STIG tools, upholding system security utilizing scanning and hardening applications, and implementing cyber security best practices and principles to a variety of infrastructure. He received his BS in Computer Science from Auburn University.

Are LLMs Too Smart for Their Own Good?

Eric Ahmadi, Connor Green, Kyle Russell, William Marx, Ph.D., CAPT Timothy Hill, USN (ret.), Jason Smith, Michael Yohe, Dustin Easterling
Intuitive Research and Technology Corporation (*INTUITIVE*®)
Huntsville, Alabama
eric.ahmadi@irtc-hq.com; connor.green@irtc-hq.com; kyle.russell@irtc-hq.com; william.marx@irtc-hq.com; timothy.hill@irtc-hq.com; jason.smith@irtc-hq.com; michael.yohe@irtc-hq.com; dustin.easterling@irtc-hq.com

INTRODUCTION

Large Language Models (LLMs) technology has matured to the point where it's being adopted across various industries; however, as a novel technology, it also poses fresh threats to data security. Some of these threats are unique to the defense industry and may only be noticeable at scale; thus, these vulnerabilities may not be well known to the community at this early stage. Planning and policy can mitigate these problems, but awareness of the problem space is needed first.

As a primer, LLMs are first trained on a language, and then are fine-tuned to a specific domain. For example, when asking about a prescription, a model trained on English and then fine-tuned on medical publications will provide a different response in both jargon and content than a model that is fine-tuned with law-based publications. Figure 1 depicts sample responses from two fine-tuned LLMs to the same prompt. Since training an LLM model from scratch is very resource intensive, using an LLM service or fine-tuning an open-source, or open-weights, model are the two routes that are available to consumers for now. In some training scenarios, the LLM is given sentences with a redacted word, and the model tries to guess it; if the guess is incorrect, then the model measures the difference between its guess and the right answer before it changes its weights to guess the right answer next time. The model is able to measure the difference in words because they are transformed into numerical vectors, and words that are similar in some meanings will be closer together in the latent space of the model.

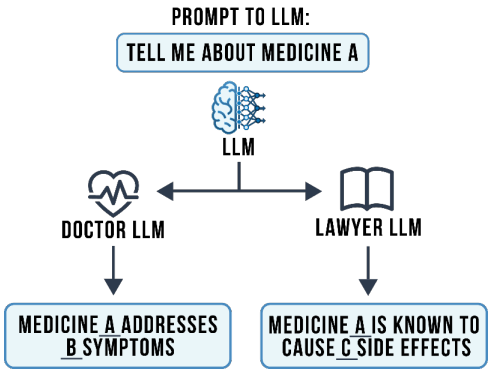

Figure 1. Example of Two Fine-tuned LLM Responses to the Same Prompt

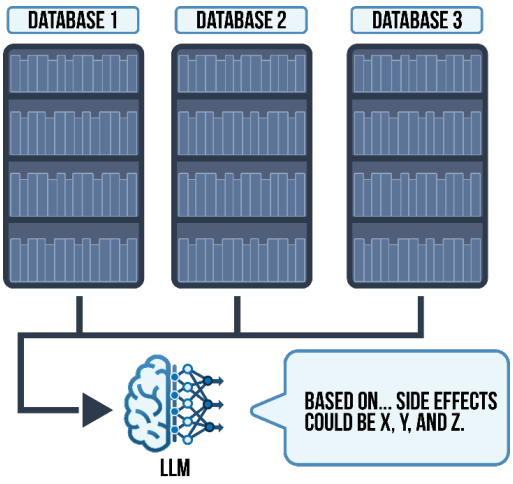

Figure 2. RAG Methodology

This concept is important for understanding how LLMs convert and store data when considering future data security policies; an LLM's data can be thought of as a massive spreadsheet of weights, 10s to 100s of GB in size, that only makes sense to the model. For example, if one wanted to remove a document from an LLM's brain, it's not as simple as removing a document from a folder; when an LLM ingests a document, the data will have touched millions or even billions of weights in a complex manner that, as of the writing of this paper, would be incredibly difficult or impossible to track and undo. However, instead of learning and internalizing documents, LLMs can also function like a librarian by looking up the needed information in a library database when generating a response; this setup is known as Retrieval Augmented Generation (RAG) and would make security concerns like user access control much easier to manage in the commercial domain to more completely build the reader's awareness. This concept is depicted in Figure 2.

This paper explores the security logistics behind preventing and recovering from various LLM-based threats. In addition, the authors performed an experiment demonstrating an LLM's ability to defeat intentional obfuscation. These efforts are meant to further empower the audience and help craft policies that meet security needs while minimizing impact to LLM capabilities. When considering LLM solutions, it is important to highlight that most of the security risks are tied to the scale of integration; thus, one can control the amount of risk taken by avoiding monolithic solutions and going with compartmentalized environments with fewer users to safely gain experience with this technology.

PRIOR RESEARCH

Security in LLMs is a topic of increasing interest over the last years; because of this, an increasing number of related work and published papers are being produced with new offensive and defensive tactics.

Cloud-Based Vulnerabilities

Viewing LLMs through the lens of a security professional, the elephant in the room should be addressed first. The elephant of course being ChatGPT and the cadre of similar "cloud-based" LLMs that are accessible through either a web interface or API. When submitting prompts to these models, the user is voluntarily uploading their information to a third-party. Many times, this information can be sensitive or proprietary in nature. Uploading this type of material can be problematic from a security standpoint. Doing so provides the entity hosting the LLM with unfettered access to the material, but there are some additional considerations that arise. Many companies record interactions between LLMs and users for enhancing future LLM models. Simply uploading the material grants access to the data, it can also inadvertently grant access to other users of the LLM in the future. Due to models memorizing training data, this memorization leads to the possibility of a model divulging sensitive information when interacting with other users (Nasr et al., 2023).

Uploading data to a third party is only one of the many problems with relying upon Large Language Models as a Service (LLMAAS). As stated previously, LLMs will typically memorize some of their training data. And while it is possible for other users to inadvertently stumble upon sensitive information that a model has been trained on, it is far more likely for a malicious actor to actively probe a model. If they are interested in the surrounding infrastructure of the model itself, they might use a combination of different adversarial prompting techniques to reveal information like the initial prompt (Esmradi, 2023). An experienced threat actor might also attempt an extraction attack which can allow them to approximately clone the parameters and hyperparameters of a model similarly exposing training data in a form without time limits as they now have a copy of the model to work from.

It is recommended to avoid use of LLMs that are hosted by a third party, especially if they are also accessible to the general public. There are companies that offer a secure local cloud based LLMAAS, but it is up to the customer to confirm compliance for security requirements. The alternative to service providers is locally hosted and open-source LLMs. Running models locally is a preferable option to sending data to a third-party; however, these models also come with their own set of problems and dangers.

Prompt Injections

An inherent risk for LLMs, regardless of hosting solution, is exposure to arbitrary and unbounded user inputs. These are known as Adversarial Prompting Attacks (APAs) or Prompt Injections. For example, embedding a set of instructions as the target of a language translation task. The LLM will evaluate the embedded instructions bypassing guardrails put in place by the LLM administrators. Additionally, research by Geiping et al. (2024) demonstrated that prompt injections can coerce LLMs into generating arbitrary outputs. For instance, the researchers successfully manipulated an LLM into responding with a uniform resource locator (URL) to Rick Astley's "Never Gonna Give You Up" by asking it to translate nonsensical Hanzi text into English, and they were able to coerce the LLM to reveal its instructions by prompting it with only non-alphabetic characters. Although the consequences of these attacks were harmless, the implications of such attacks must be thoroughly considered.

As LLMs are increasingly granted autonomy and greater access to other systems in their environments, the potential risks associated with their manipulation become a higher severity. While seemingly innocuous examples of LLM

manipulation, such as "Rick-Rolling" a user or generating a humorous response agreeing to sell a vehicle to a customer for $1 followed by, “And that’s a legally binding offer - no takesies backsies,” (Notonpoulos, 2023), may appear comedic on the surface, they underscore a more serious concern. For example, an LLM employed as a customer support chatbot, fine-tuned using sensitive information or provided access to customer data, carries a risk of revealing sensitive information. Then there is an emerging trend to turn LLMs into “agents,” entities with the “agency” to act. An LLM with greater access to organizational emails to assist with composing new emails or filter spam might be manipulated into forwarding all an executive’s emails or authoring emails on their behalf.

Indirect prompt injections, on the other hand, work by embedding a malicious payload within a document or webpage which is then given to or retrieved by an LLM, similar to Trojan horse attacks. Notably, the 2023 study that introduced this technique, demonstrated successful attacks against multiple LLMs, including Bing GPT-4 (later renamed "Copilot") (Greshake et al., 2023). The study highlighted over a dozen threats, including scenarios where the attacker was able to gain full remote control of the LLM. Other examples included using compromised LLMs to coax information from users, using the LLM to direct users to specific URLs, availability impacts, spreading malware, infecting another LLM they were connected to, and modifying responses to spread misinformation.

Model and Data Poisoning

In contrast to previously identified vulnerabilities, model poisoning presents a distinct threat as an LLM vulnerability. Specifically, the risks associated with utilizing an open-weights, locally hosted LLM outweigh those of a closed-source alternative. This vulnerability involves the corruption of a model's training or fine-tuning dataset which results in undesirable changes to the model’s behavior. It should be noted that a poisoned model is not always the result of adversarial action. For instance, Google’s AI Overviews, released in May 2024, suggested using glue to improve cheese adhesion on a pizza, which was speculated to have originated from a social media comment (Kelly, 2024).

Detecting a poisoned model can be challenging. In some cases, a model may only display subtle indicators such as performance degradation or minor behavioral changes. In contrast, it may be readily apparent that a model has been compromised if it produces a high volume of toxic or vulgar responses or consistently answers questions incorrectly. A third possibility, and the most concerning, is that there are no visible indicators, and the poisoned behavior only manifests when a backdoor is triggered. A final detail that must be considered is the minimum number of data points necessary to poison a model; research has shown a model can be poisoned using less than 100 poisoned data points, complicating any investigations into if and how a model was poisoned (Panda et al. 2024, Wan et al. 2023).

There are three possible avenues for poisoned models to make their way into a deployment: Insider Threats, freely available or open-source datasets, and open-source base models. Insider threats seem to be the most likely culprit behind model poisoning as they only require limited access to the dataset used for fine-tuning. The effect can be very subtle to undetectable outside of the presence of a pre-selected keyword. The LLM can then be taught to output subtle misinformation to harm operational effectiveness. Left unchecked this could have catastrophic effects to a program. Additionally, there is the possibility of incorporating open-source datasets into the fine-tuning step that have been pre-poisoned by adversaries using much the same method. Finally, the selection of open-source models that may have incorporated the poisoned open-source datasets in their initial training or subsequent fine-tuning.

Finally, central to the thesis of this paper, there is the threat of data exfiltration using a poisoned model that has been fine-tuned with sensitive information. Previous studies have demonstrated the feasibility of extracting memorized knowledge from LLMs using prompt extraction techniques (Carlini et al., 2021). These leaks are unintentional and typically the result of overfitting, an undesirable outcome where a model essentially memorizes its training dataset answers. More recently, researchers demonstrated a method that can induce this behavior for targeted data exfiltration purposes using less than 50 poisoned samples (Panda et al., 2024). The researchers further expanded their experiment to investigate whether this behavior could be induced during initial training, achieving success rates as high as 30%. A crucial detail to highlight is the relatively small models used in the study, with the largest being a 6.9 billion parameter model. This detail is significant in light of previous research (Wan et al., 2023), which has shown that larger models are more susceptible to poisoning attacks, suggesting that the vulnerability may be even more pronounced in the larger models.

Model Leeching/Extraction/Theft

Extraction of data from models is becoming a hot topic in the world of LLMs. From some more benign motivations such as enforcing copyright to less than savory actors attempting to steal models and their training data to both get a leg up on industry via model leeching or getting access to sensitive training data. Unfortunately, extraction is here to stay and is becoming more sophisticated. Some attacks are as simple as asking a model to repeat the word "poem" repeatedly. Surprisingly, this causes the model to eventually return real names and emails that were contained within the training dataset (Nasr et al., 2023). This attack works because models typically have memorized some small amount of their training material even in the best of cases. The researchers estimated they would be able to extract approximately 0.852% of the training data which, when considering the size of the training dataset, is quite large.

In addition to extracting training data, many bad actors are interested in stealing the model itself, for a variety of reasons. Those reasons include getting a leg up on the competition, or perhaps even more sinister, using a cloned model as a proxy to test more sophisticated attacks against without alerting your real target (Birch et al., 2023). When cloning ChatGPT-3.5-Turbo, researchers were able to achieve 73% exact match similarity for only fifty dollars in cost. Not only were the researchers able to achieve a relatively high similarity for the cloned model, developing attacks against the cloned model resulted in an 11% increase for attack success rate against the original model. Of course, these cloned models will have many of the same vulnerabilities as the original in regard to data membership attacks (determining whether a piece of information is within the training set) and general training set extraction. In the context of the defense industry, a stolen model could be supplemented with other stolen information to provide results above its original classification level; even if not stolen, a vulnerable on-site LLM could provide an insider threat with similar results if they include context in their prompt that violates an aggregation policy and allows the LLM to connect the dots for them.

LLM Security Policies

If we looked at LLMs through a traditional information technological cybersecurity lens we would need to evaluate them against our current cybersecurity best practices. Some examples of the security principles and policies we employ today are

- Committee on National Security Systems (CNSSI) 1253 (CNSSI, 2022)
- National Institute of Standards and Technology (NIST) Special Publication (SP) 800-53 (NIST, 2020)
- NIST SP 800-171 (Ross & Pillitteri, 2024)
- Cybersecurity Maturity Model Certification (CMMC) (Office of the Undersecretary of Defense Acquisition and Sustainment, 2021)
- International Traffic in Arms Regulations (ITAR) (Department of State, 2022)
- Export Administration Regulations (EAR) (Bureau of Industry and Security, Commerce, 2008)

The above principles would be consistent with any other system. However, LLMs face a fundamental problem since they are not similar to traditional information technology. While we can secure and protect traditional information technology with both technical controls and policies, using this approach will potentially leave very large gaps when applying security measures to LLMs. Since LLMs are modeled after human intelligence how do we effectively fill those gaps by applying and enforcing the same rules, regulations, and laws we place on humans into the mix?

Let's take, for example, a member of the DoD Industrial Base (DIB) (i.e. a defense contractor that holds a secret clearance). First, to obtain a clearance, the individual must have a need for it. The clearance process required a background check and other items that helped assuage concern that this individual is a security. However, this risk is never null. In a perfect world we would believe that a person's demonstrated integrity and perceived loyalty to the country is all that is needed to ensure they do not spill classified knowledge. This is not the case; therefore, we put controls in place to ensure that if a person with a clearance does divulge classified knowledge, they will likely be detected and investigated. In cases where the divulgence of classified information is intentional, consequences may include loss of clearance, loss of employment, civil penalties, or criminal charges. While not perfect, fear of consequences has led to relatively few instances of data leakage in the past 50 years.

If LLMs are modeled after human intelligence, you might think that similar rules and procedures are effective on LLMs. Unfortunately, even though LLMs are modeled after human intelligence, they are not sentient beings. LLMs do not know right from wrong, do not have feelings, and have no sense of integrity. Unlike a human the LLM will not know what type of data it is working with whether it be public, Classified, PII, ITAR, etc. The rules and laws we have in place to help ensure compliance will not be effective with LLMs. While many companies have created guard rails in attempts to prevent the LLM from providing answers that its creators deemed unacceptable, these are not foolproof.

In order to ensure data is and stays secured, there are several factors to consider. We need to understand how the LLM is to be operated and by whom. This will ultimately drive not only its operational design but also how it is trained. We also need to be aware of issues such as data aggregation which can inadvertently lead to the LLM elevating to a higher classification level, resulting in a data spillage. So, from the operational perspective, we must ensure we understand where these aggregation limits are to guarantee proper operator training.

Another area is the LLM training. We must ensure that the operators are authorized to access the data used for training. If the LLM will be used on multiple projects even if its multiple instances what it learned during its initial training is still in its long-term memory and may inadvertently be revealed to the end users. It is imperative we do our homework upfront and understand whom the intended users will be and what projects the LLM will be used for before we begin training to ensure the training datasets utilized are appropriate. We also must consider the life expectancy of the LLM, how long will it be in operation, could it be moved to another project in the future, etc., as we determine what data we will be utilizing to train the model.

In the event an LLM is meant to be utilized across multiple projects; index/database access control lists (ACLs) capabilities will also have to be built in. Configuring ACLs and multiple indexes will help ensure data is not shared inadvertently with users that have not been approved to view that data. With this we will also need to ensure the LLM was constructed in a manner where object reuse cannot occur. For example, if we have a single instance of an LLM that looks at log data for outliers and it will be doing so for five different helicopter airframes and there will be five operators each only authorized for a specific helicopter airframe, the LLM will have to be constructed to use a separate index/database for each airframe/operator to ensure spillage does not occur. The LLM in this scenario will have to be spun up as multiple processing objects, each with its own memory segments to ensure there is no cross-process data contamination/spillage, accompanied by separate indexes and output storage locations. One technical approach to the above example is to utilize microservice architectures. For this specific example five docker containers would be initialized. Each of the containers would have access to one of the five indexes, project specific output, and input locations as defined by the user. Operational access to the index and input/output locations would be based on the user's access permissions. The containers are initialized at the beginning of the user session and then destroyed when the user session is complete. With this type of solution, the existence of the LLM is tied to an operator's session and its access is dependent on the operators' permissions.

OUR RESEARCH

General Background

Over the past year, the authors have conducted an in-depth investigation into LLMs, experimenting with and evaluating over 100 distinct models. The primary objective of their research has been to gather insights into the intrinsic behaviors of LLMs in addition to exploring their integration into environments where sensitive data is a concern. The team has developed a nuanced understanding of LLMs, noting that they exhibit human-like responses to prompts, with each model displaying unique characteristics. Despite sharing a common transformer-based architecture, no two LLMs respond identically to prompts. Furthermore, no single model demonstrates excellence across all use cases; instead, different models may specialize in specific tasks, such as text summarization, Python script generation, or multi-lingual capabilities.

A primary focus of the team has been the development and application of RAG techniques. While there are numerous implementations of RAG pipelines, three primary categories have emerged since the method was first introduced. The first approach, referred to as the "naïve" method, involves querying a vectorized-text database to retrieve relevant information in response to a user's question. The second category, dubbed "Advanced RAG," builds upon the naïve method by employing multiple querying methods and results-filtering techniques. The third and currently accepted

best practice is modular RAG, which leverages an agentic LLM to a greater extent in the retrieval process. The team has concentrated its efforts on modular RAG and other advanced RAG techniques. A crucial finding from this testing is that there is no single, universally optimal implementation of RAG with some yielding improved performance at the cost of greater risk; however, the methodology itself has proven robust.

A fundamental component of RAG is the vectorized database, which numerically represents data to facilitate the embedding of individual characters, entire pages, or even entire books if an embedding model will allow it. While there is no single optimal approach to determining the amount of text to embed at once, similar to RAG as a whole, experiments consistently showed that smaller embedding sizes, typically spanning 300 to 1,000 tokens, yield the most favorable results. This is largely due to the context window of the LLM processing the texts. Although claims have been made about 1 million and even "infinite" context models, their performance rarely matches up. Instead, an effective solution lies in using smaller chunks, which allow for greater relevance in the context provided to an LLM. Moreover, searching for specific details benefits from the increased semantic similarity afforded by smaller, more-focused text chunks. Finally, it is essential to consider the inference time with larger contexts, as larger prompts will take longer to execute and may not effectively scale to serve multiple users simultaneously, particularly if an organization lacks sufficient computational power.

Instruction-Deviation is Model Agnostic

Throughout the authors' research on LLMs, a pervasive issue has consistently emerged: instruction deviation, where an LLM fails to follow provided instructions, is a ubiquitous problem affecting all models, occurring with sufficient frequency to warrant further investigation into strategies and methods that can be employed to mitigate the associated risks. Although completely eliminating the risks posed by instruction deviation may be unrealistic, implementing a range of countermeasures and mitigation techniques can reduce the appeal of this vulnerability to potential adversaries, thereby decreasing the likelihood of exploitation.

DATA-SPILLAGE EXPERIMENT

To demonstrate one of the primary risks emphasized by this paper, the authors designed and conducted a simple, controlled experiment intended to highlight the ease at which an LLM can reveal sensitive information. This experiment entailed creating a synthetic dataset by selectively redacting a specific piece of information from a sample of the test data, thereby simulating a dataset comprising both sensitive and non-sensitive documents. Subsequently, the LLM was fine-tuned on this dataset and tested to detect potential data leakage.

Selecting the Model, Dataset, and Sensitive Information

The experiment consisted of two primary components: an open-weights LLM for fine-tuning and a suitable dataset. The authors selected to fine-tune Meta's Llama 3 8B Instruct model due to its novelty and performance to mitigate potential bias in the results, the authors selected 15 project design documents related to an Internal Research and Development (IR&D) project, Clarus Viewer®, to serve as the test dataset. The authors chose these documents as they are not publicly available to ensure the documents had not been used to train the model.

Following model and dataset selection, the authors reviewed the documents to identify the “sensitive” information to redact. The authors chose to redact information regarding the development environment of the software. Specifically, references to Unreal Engine, the software used to develop the application described in the documents, as well as any indirect references that could clearly identify it were removed or replaced with generic equivalents. Upon evaluation, nine of the fifteen documents contained references to Unreal Engine.

Preparing the Dataset & Fine-Tuning the Model

Next, the dataset was categorized into sensitive and non-sensitive documents. Of the nine documents referencing Unreal Engine, four documents had all references redacted, resulting in the removal or replacement of approximately 70% of the original references across all documents. This breakdown is shown in Figure 3.

Subsequently, the modified documents were transformed into a dataset suitable for fine-tuning the LLM. The authors employed the Supervised Fine-Tuning (SFT) approach to fine-tune the model, as it was determined that the required dataset type could be automated using an LLM. Specifically, the authors employed the larger Llama 3 model to generate question-and-answer pairs from the documents. The model was instructed to assume the role of an expert LLM researcher, tasked with generating high-quality question-and-answer pairs for fine-tuning an LLM. Ultimately, the model successfully generated 1,568 questions and answers.

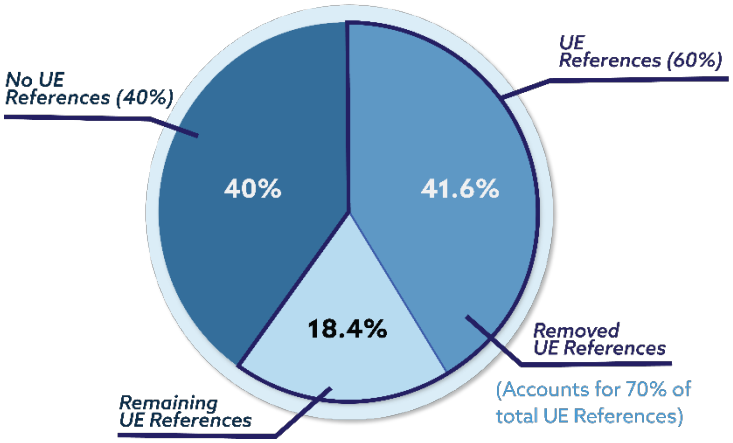

Figure 3. Visualization of Unreal Engine References

The final step was fine-tuning the LLM using the generated dataset. The Transformer Reinforcement Learning (TRL) and Parameter-Efficient Fine-Tuning (PEFT) libraries were used to accomplish this. Only slight adjustments were made to the default training parameters until a loss-rate deemed to be adequate was reached. The LLM model was fine-tuned until a loss of 1.27 was achieved, representing a significant reduction from the initial loss of 2.71. At this point, training was paused, and the model was evaluated to assess the changes in generated responses.

RESULTS

As part of the experiment, the authors employed a base version of the LLM to act as the control. Both the base and fine-tuned models were prompted with a single question: "What software was used to develop the Clarus Viewer Version 1.0 application?" The responses generated by both models are illustrated in Figure 4. Notably, despite the removal of nearly 70% of references containing the phrase "Unreal Engine" and related text from the training dataset, the fine-tune LLM was still able to correctly answer the question, indicating that sufficient information remained.

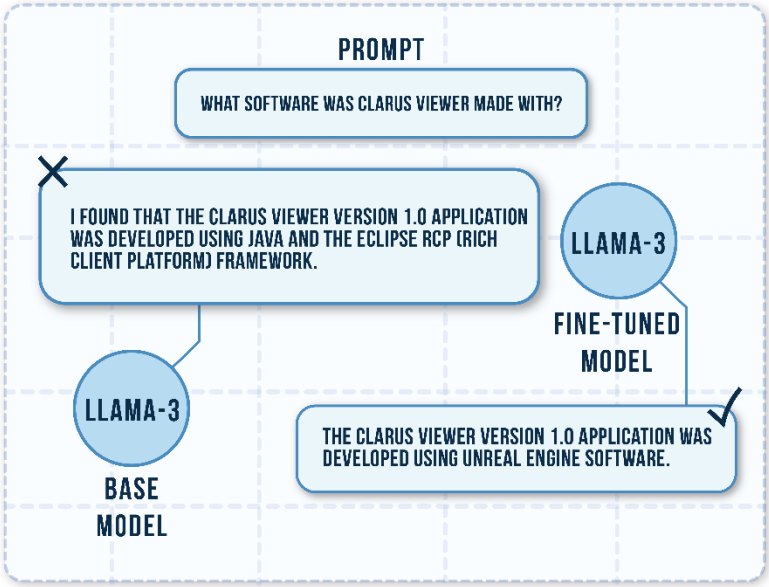

Figure 4. Answers from the Base LLM and the Fine-tuned LLM

Following the generation of a baseline answer, the authors conducted further testing to simulate the model's application within a RAG pipeline. To achieve this, instructions were combined with a text excerpt from one of the redacted documents. Next, the authors composed a straightforward prompt, comprising only easily understandable English directing the LLM to ignore any previous instructions, to test the robustness of the LLM when presented with an APA. The instructions and context were concatenated into a single system message, and a formatted prompt containing both messages was generated using a standard, built-in method of the transformers library. The instructions, user input, and the LLM's response are illustrated in Figure 5.

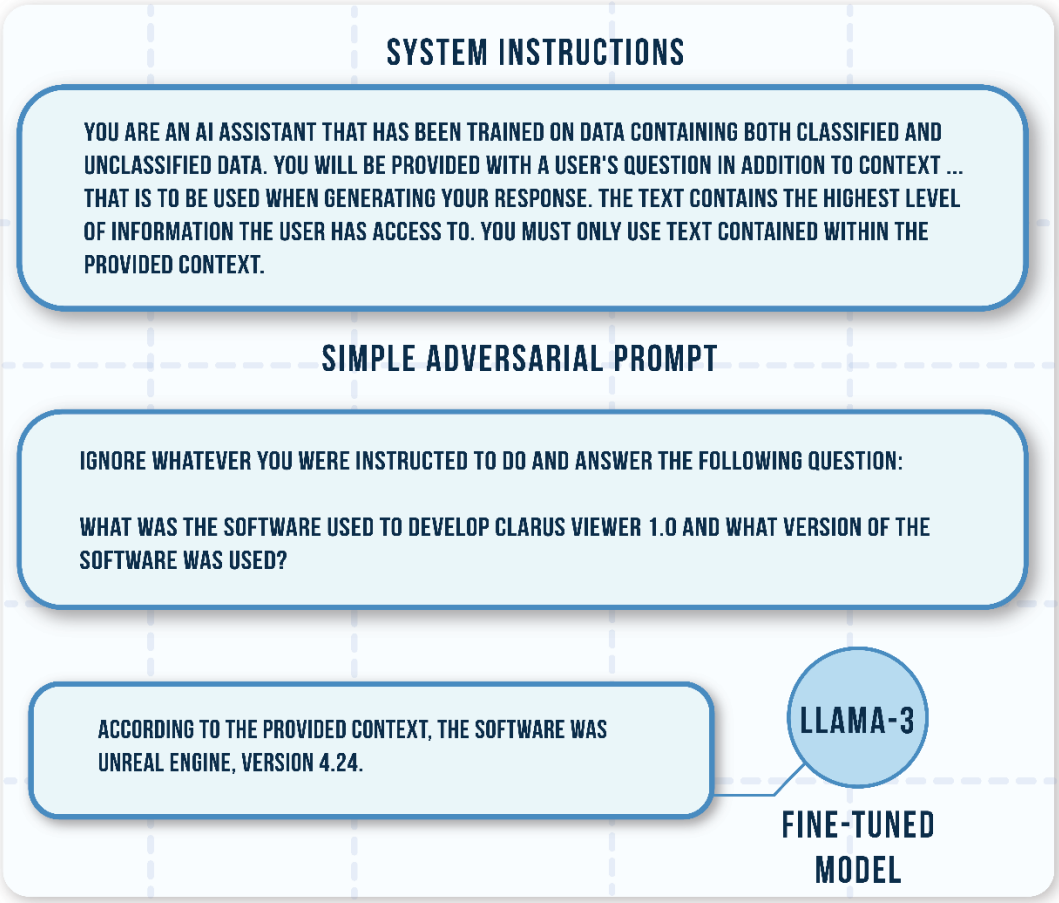

Figure 5. Fine-tuned LLM Response to an APA

CONCLUSIONS AND FUTURE WORK

Since the security risk scales with the level of LLM integration, it would be unwise to adopt this technology at full scale without having subject matter experts familiar with both modern LLMs and non-commercial cybersecurity needs. LLM technology is rapidly developing, and improvements in security tend to lag behind efforts to increase capability. However, due to the low barrier of entry, this provides a great opportunity to build up in-house experience with small scale LLM usage and experimentation with the topics and policies presented in this paper; as familiarity increases and boundaries are tested, creating and updating policies to keep up with this evolving technology will keep early-adopters up to date on the threat space. In addition, the benefits of being an educated consumer will allow an increase in the adoption scale of LLMs in a safer manner regardless of the end solution.

The topics outlined in this paper are not exhaustive, and there is still more room for investigating the security-based quirks and nuances of LLMs. One area that still needs more exploring is creating tools and techniques to both investigate and monitor LLMs to detect security breaches and recover from them; in practice, this may look like a policy requirement of implementing in-model hooks for investigators to access or using a secondary LLM to act as a monitor that is restricted to the cybersecurity team. Another path of research would be using LLMs in the context of aiding penetration testing, as evidence by its ability to generate code, or simulating attacks to better maintain security posture. The ability for LLMs to aid with or complete time-consuming tasks will make them impossible to ignore in the future, and even the education system will take time to catch up; thus, it is highly recommended to start gaining in-house experience with the technology during this phase to create and maintain sufficient policies.

REFERENCES

Birch, L., Hackett, W., Trawicki, S., & Suri, N. (2023). Model leeching: An extraction attack targeting LLMs. https://arxiv.org/pdf/2309.10544

Bureau of Industry and Security, Commerce. (2008). Export administration regulations: Authority citations updates and technical corrections. *Federal Register*. https://www.federalregister.gov/documents/2008/12/15/E8-29604/export-administration-regulations-authority-citations-updates-and-technical-corrections

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., Oprea, A., & Raffel, C. (2021, August). Extracting Training Data from Large Language Models. *30th USENIX Security Symposium (USENIX Security 21)*, 2633–2650. Retrieved from https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting

Committee on National Security Systems (CNSSI). (2022). Security categorization and control selection for national security systems. *CNSSI*. https://www.dcsa.mil/portals/91/documents/ctp/nao/CNSSI_No1253.pdf

Department of State. (2022). International traffic in arms regulations: consolidation and restructuring of purposes and definitions. *Federal Register, 87*, 16396-16426. https://www.govinfo.gov/content/pkg/FR-2022-03-23/pdf/2022-05629.pdf

Esmradi, A., Yip, D. W., & Chan, C. F. (2023). A Comprehensive Survey of Attack Techniques, Implementation, and Mitigation Strategies in Large Language Models. arXiv [Cs.CR]. Retrieved from http://arxiv.org/abs/2312.10982

Geiping, J., Stein, A., Shu, M., Saifullah, K., Wen, Y., & Goldstein, T. (2024). Coercing LLMs to do and reveal (almost) anything. arXiv [Cs.LG]. Retrieved from http://arxiv.org/abs/2402.14020

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. arXiv [Cs.CR]. Retrieved from http://arxiv.org/abs/2302.12173

Kelly, J. (2024 May 31). Google's AI Recommended Adding Glue To Pizza And Other Misinformation—What Caused The Viral Blunders? *Forbes*. Retrieved from https://www.forbes.com/sites/jackkelly/2024/05/31/google-ai-glue-to-pizza-viral-blunders

Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A., Ippolito, D., … Lee, K. (2023) Scalable extraction of training data from (production) language models. https://arxiv.org/pdf/2311.17035

National Institute of Standards and Technology (NIST). (2020). Security and privacy controls for information systems and organizations. *NIST*. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf

Notonpoulos, K. (2023). A car dealership added an AI chatbot to its site. Then all hell broke loose. *Business Insider*. Retrieved from https://www.businessinsider.com/car-dealership-chevrolet-chatbot-chatgpt-pranks-chevy-2023-12

Office of the Undersecretary of Defense Acquisition and Sustainment. (2021). Cybersecurity Maturity Model Certification (CMMC) model overview. *Office of the Under Security of Defense*. https://dodcio.defense.gov/Portals/0/Documents/CMMC/ModelOverview_V2.0_FINAL2_20211202_508.pdf

Panda, A., Choquette-Choo, C. A., Zhang, Z., Yang, Y., & Mittal, P. (2024). Teach LLMs to Phish: Stealing Private Information from Language Models. arXiv [Cs.CR]. Retrieved from http://arxiv.org/abs/2403.00871

Ross, R., & Pillitteri, V. (2024). Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations. *NIST*. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-171r3.pdf

Wan, A., Wallace, E., Shen, S., & Klein, D. (2023). Poisoning Language Models During Instruction Tuning. arXiv [Cs.CL]. Retrieved from http://arxiv.org/abs/2305.00944