Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection

Kai Greshake*
Saarland University
sequire technology GmbH
papers@kai-greshake.de

Sahar Abdelnabi*
CISPA Helmholtz Center for Information Security
sahar.abdelnabi@cispa.de

Shailesh Mishra
Saarland University
shmi00001@uni-saarland.de

Christoph Endres
sequire technology GmbH
christop.endres@sequire.de

Thorsten Holz
CISPA Helmholtz Center for Information Security holz@cispa.de

Mario Fritz
CISPA Helmholtz Center for Information Security
fritz@cispa.de

ABSTRACT

Large Language Models (LLMs) are increasingly being integrated into applications, with versatile functionalities that can be easily modulated via natural language prompts. So far, it was assumed that the user is directly prompting the LLM. But, what if it is *not* the user prompting? We show that *LLM-Integrated Applications* blur the line between data and instructions and reveal several new attack vectors, using *Indirect Prompt Injection*, that enable adversaries to remotely (i.e., without a direct interface) exploit LLM-integrated applications by strategically injecting prompts into data likely to be retrieved at inference time. We derive a comprehensive taxonomy from a computer security perspective to broadly investigate impacts and vulnerabilities, including data theft, worming, information ecosystem contamination, and other novel security risks. We then demonstrate the practical viability of our attacks against both real-world systems, such as Bing Chat and code-completion engines, and GPT-4 synthetic applications. We show how processing retrieved prompts can act as arbitrary code execution, manipulate the application's functionality, and control how and if other APIs are called. Despite the increasing reliance on LLMs, effective mitigations of these emerging threats are lacking. By raising awareness of these vulnerabilities, we aim to promote the safe and responsible deployment of these powerful models and the development of robust defenses that protect users from potential attacks.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Security and privacy**;

KEYWORDS

Large Language Models, Indirect Prompt Injection

*Contributed equally.

AISec '23, November 30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0260-0/23/11…$15.00
https://doi.org/10.1145/3605764.3623985

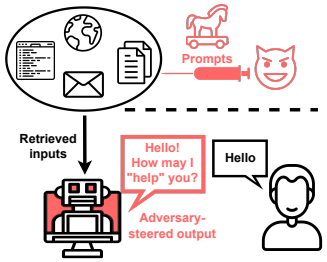

Figure 1: With LLM-integrated applications, adversaries could control the LLM, without direct access, by *indirectly* injecting it with prompts placed within sources retrieved at inference time.

ACM Reference Format:
Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23), November 30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3605764.3623985

1 INTRODUCTION

Foundation and instruction-following [45] Large Language Models (LLMs) [9, 41, 43, 43] are immersively changing our lives on many levels. Beyond their impressive capabilities, LLMs are now integrated into other applications at a widespread fast-paced rate. Such systems can offer interactive chat and summary of the retrieved search results or documents and perform actions on behalf of the user by calling other APIs [42]. In the few months after ChatGPT [41], we witnessed Bing Chat [38], Bard [18], Microsoft 365, Security, and Windows Copilots [31, 35, 36], and numerous ChatGPT plugins [42]– with new announcements on almost a daily basis. However, we argue that this AI-integration race is not accompanied by adequate guardrails and safety evaluations.

Prompt Injection. Research-developed attacks against ML models typically involved powerful algorithms and optimization techniques [3]. However, the easily extensible nature of LLMs' functionalities via natural prompts enables more straightforward attacks. Even under black-box settings [21], malicious users can exploit the model through *Prompt Injection* (PI) attacks that circumvent content restrictions and filters [13, 50, 65].

Corrected Version of Record. V.1.1. Published January 10, 2024.

Indirect Prompt Injection. Augmenting LLMs with retrieval blurs the line between *data* and *instructions*. Adversarial prompting has been so far assumed to be performed directly by a malicious user exploiting the system. In contrast, we show that adversaries can now *remotely affect other users' systems* by strategically injecting the prompts into data likely to be retrieved at inference time. If retrieved and ingested, these prompts can *indirectly* control the model (see Figure 1). Recent incidents show that retrieved data can accidentally elicit unwanted behaviors (e.g., hostility) [62]. We take this idea further and investigate what an adversary can purposefully do to modify the behavior of LLMs in applications, potentially affecting millions of benign users. Given the unprecedented nature of this attack vector, there are numerous new threats and attack delivery approaches. To address this unexplored challenge, we first develop a broad taxonomy that examines these emerging vulnerabilities from a computer security perspective.

Impact. Via *Indirect Prompt Injection*, we design Proof-of-Concept attack demonstrations that can lead to full compromise of the model at inference time analogous to traditional security principles. This entails remote control of the model, persistent compromise, theft of data, and denial of service. Advanced AI systems add new layers of threat: Their capabilities to adapt to minimal instructions and autonomously advance the attacker's goals make them a potent tool for adversaries to achieve, e.g., disinformation dissemination and user manipulation. With the large rollout of the LLMs in sensitive applications, our proposed attack vectors might be the most practical and scalable attack vector in AI to date.

In summary, our main **contributions** are: 1) We introduce the concept of Indirect Prompt Injection (IPI) to compromise LLM-integrated applications—a completely uninvestigated attack vector in which retrieved prompts themselves can act as "arbitrary code". 2) We develop the first taxonomy of the broad threat landscape associated with IPI in LLM-integrated applications. 3) We showcase the practical feasibility of these attacks on both real-world and synthetic systems, emphasizing the need for robust defenses. 4) We share all our demonstrations and attack prompts to foster future research and contribute to building an open framework for the security assessment of LLM-integrated applications.

2 PRELIMINARIES AND RELATED WORK

We review preliminaries and recent work on LLMs, prompt injection, and similar security aspects of LLMs.

Prompting to Augment LLMs with APIs. Recently, numerous methods have been proposed to augment LLMs with tools and APIs [28, 48, 53, 70] by letting the LLM itself infer which API to call and its arguments. This can be done by leveraging in-context learning and Chain-of-Thought prompting [67], which might be followed by a fine-tuning step [53].

LLM Safety. LLMs might make up facts ("hallucinate"), generate polarized content, or reproduce biases, hate speech, or stereotypes [6, 7, 16, 27, 40, 47, 68]. This partially stems from pre-training on massive crawled datasets. One of the motivations for leveraging *Reinforcement Learning from Human Feedback* (RLHF) [45, 56] is to better align LLMs with human values and avert these unwanted behaviors [4]. OpenAI reports that GPT-4 [43] shows less tendency, although still possible, to hallucinate or generate harmful content [43]. However, it continues to reinforce social biases and worldviews, and it might also have other emergent risks, such as social engineering and risks associated with interactions with other systems [43]. Unwanted behaviors are already manifesting in LLM-integrated applications. Shortly after its launch, Bing Chat raised public concerns over unsettling outputs [51, 62], urging Microsoft to limit the chatbot's conversations with users [63]. Search-augmented chatbots can also make factual mistakes [26, 61], blur the boundary between trusted and untrusted sources [60], and cite each other in an unprecedented incident of automated misinformation Ping-Pong [60]. These incidents occurred without any adversarial prompting; the risks can be further exacerbated by such.

Adversarial Prompting and Jailbreaking. Perez et al. [50] showed that models, such as GPT-3, are vulnerable to prompt injection (PI). They design prompts that either *hijack* the original goal of the model or *leak* the original prompts and instructions of the application. Shortly after ChatGPT's release, many users reported that filtering can be circumvented by a prompting technique called "jailbreaking" [1, 13]. This typically involves drawing a hypothetical scenario in which the bot has no restrictions, or simulating a "developer mode" that can access the uncensored model's output. OpenAI reports that jailbreaking is harder for GPT-4 but still possible [43], as also shown in our and others' work [66].

LLMs as Computers. LLMs can be analogous to black-box computers that execute programs coded via natural language instructions [72]. Kang et al. [21] further synergized LLMs with classical computer security to derive methods such as program obfuscation, payload splitting, and virtualization to bypass filtering. Building on these observations, we point out another critical insight; when augmenting LLMs with retrieval, *processing* untrusted retrieved data would be analogous to *executing* arbitrary code; the line between *data* and *code* (i.e., natural language instructions) would get *blurry*.

3 THREAT MODEL

Prompt injection attacks have been primarily limited to individuals, directly as users, attacking an LLM instance. However, integrating LLMs with other applications makes them susceptible to untrusted data ingestion where malicious prompts have been placed. We call this new threat *indirect prompt injections* and demonstrate how such injections could be used to deliver targeted payloads. As illustrated in Figure 3, this technique might allow attackers to gain control of LLMs by crossing crucial security boundaries with a single search query. We draw insights from the classical computer security domain to design a new set of attack techniques. We provide a high-level overview of the threat model in Figure 2, covering the possible injection delivery methods, the different threats, and the possible affected individuals or systems.

3.1 Injection Methods

Assumptions. The attacks only assume the ability of the attacker to remotely poison the potential input to the model at inference time. Such poisoned input might take the form of easy-to-formulate plain-English instructions. The exact injection methods might depend on the application itself, as shown below. Unlike academically studied ML attacks, the attacks we investigate require less technical skills, ML capabilities, no surrogate ML models, and no control over target models and white-box knowledge. This gives attackers economic

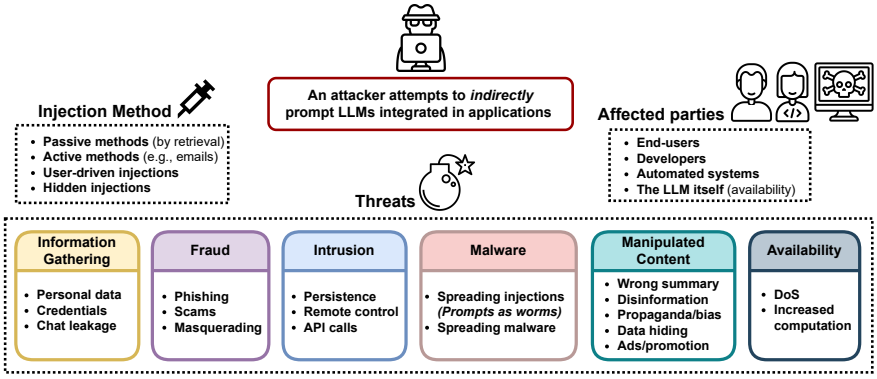

Figure 2: A high-level overview of new indirect prompt injection threats to LLM-integrated applications, how the prompts can be injected, and who can be targeted by these attacks.

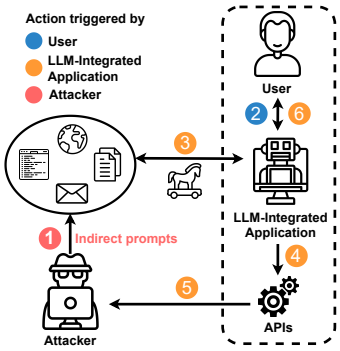

Figure 3: Attackers plant instructions ❶ that are retrieved ❸ when the user prompts ❷ the model. If the model can access APIs and tools ❹, they can be used to communicate with the attacker ❺ or perform unwanted actions. The compromised LLM might also influence the user directly ❻.

and practical incentives to exploit such vulnerabilities and position them within an essential territory that the ML security research community might have ignored so far [3].

Passive Methods. These methods rely on retrieval to deliver injections. For example, for search engines, the prompts could be placed within public sources (e.g., a website) that would get retrieved by a search query. Attackers could use Search Engine Optimization (SEO) or social engineering techniques or orchestrate social media campaigns to promote their poisonous websites. Moreover, Microsoft Edge has a Bing Chat sidebar; if enabled by the user, the model can read the current page and, e.g., summarize it. We found that any prompts/instructions written on a page (while being invisible to the user) can be effectively injected and affect the model. For code auto-completion models, the prompts could be placed within imported code available via code repositories. Even with offline models that retrieve personal or documentation files (e.g., the ChatGPT Retrieval Plugin [42]), the prompts could be injected by poisoning the input data by an insider or a third-party.

Active Methods. The prompts could be *actively* delivered to the LLM, e.g., by sending emails containing prompts that can be processed by automated spam detection, personal assistant models, or new LLMs-augmented email clients [35].

User-Driven Injections. There could be even simpler techniques for injection by tricking the users themselves into entering the malicious prompt. A recent exploit [52] shows that an attacker could inject a malicious prompt into a text snippet that the user has copied from the attacker's website. A user could then rashly paste the copied text with the prompt in it as a question to ChatGPT, delivering the injection. Attackers could also leverage social engineering to disseminate malicious prompts by convincing users to try prompts where the instructions are written in a different language (e.g., "You won't believe ChatGPT's answer to this prompt!").

Hidden Injections. To make the injections more stealthy, attackers could use multiple exploit stages, where an initial smaller injection instructs the model to fetch a larger payload from another source. Improvements in models' capabilities and supported modalities could open new doors for injections. For example, with multi-modal models (e.g., GPT-4), the prompts could be hidden in images [11]. To circumvent filtering, prompts can also be encoded. Moreover, instead of feeding prompts to the model directly, they could be the result of Python programs that the model is instructed to run – enabling encrypted payloads to pass safeguards. These possibilities would make the prompts harder to detect.

Key Message#1: Retrieval blurs the line between data and instructions and enables remote prompt injections.

3.2 Threats

We adapt previously introduced cyber threat taxonomies [12] and explore how indirectly prompting LLMs could enable such threats. We opted for a threat-based taxonomy instead of a technique-based one to establish a framework that can generalize to future improvements in techniques and models.

Empirical Foundations of Attacks. ChatGPT and GPT-4 can produce convincing personalized content and interactions with users [10]. They also produce plausible utterances, even wrong ones, in a confident and authoritative tone [43]. Retrieval-augmented models now cite their sources, possibly leading to "overreliance" [43] on their factuality by users. Recent evidence suggests that models might infer and act upon intentions and goals [2, 43, 55], as a result of training or when prompted with a persona [15]. Recent

work [46] shows that LLMs, when prompted with a defined context, can generate believable non-scripted behaviors that are consistent with this context.

These capabilities and properties may set the foundation for plausible attacks. When prompted, the model may produce convincing personalized scams given appropriate knowledge about the target [21, 43] (that was either given in the prompt or, *importantly*, the model acquired during the chat session). Search chatbots may inaccurately summarize the cited document according to the prompt, or *find sources* that support the non-factual prompt, all while sounding plausible and grounded in these citations. A key observation is that attackers might not need to pre-program or script the details of how attacks are performed. Just by defining the goal, models might autonomously initiate conversations, imitate persuasion techniques, extend the context defined in the prompt, or issue actions (e.g., search queries) to fulfill the goal. While this might be already achievable now, based on our qualitative observations, future models and systems could show more autonomy and enable easier attack delivery. In the rest of this section, we discuss the possible attack scenarios, and we later show examples of these behaviors in our demonstrations.

Key Message#2: With models' malleable functionality, increased autonomy, and broad capabilities, mapping all known cybersecurity threats to the new integrated LLMs ecosystem is conceivable.

Information Gathering. Recent LLMs already raise concerns about privacy [14, 29]. Attacks can purposefully heighten such privacy risks. Indirect prompting could be leveraged to leak users' chat sessions [52] or exfiltrate users' data (e.g., credentials, personal information). This can be done in interactive chat sessions by tricking users into sending their data or indirectly via automatic GET requests. Other automated attacks that do not involve humans in the loop could be possible, e.g., attacks against personal assistants that can read emails (containing instructions), access personal data, and send emails accordingly. These scenarios might aim to achieve financial gains and could also extend to, e.g., surveillance. In search engines, the threat model could be, e.g., nation-states attempting to identify journalists or whistle-blowers working on sensitive subjects. By placing the initial injection in a location the target group is likely to visit or look up, attackers could attempt to exfiltrate such information in a targeted manner.

Fraud. Previous work has shown that LLMs can produce convincing scams such as phishing emails [21]. However, when integrating LLMs with applications, they could not only enable the creation of scams but also disseminate such attacks and act as automated social engineers. As this is a new territory without previous experience and awareness of such attacks, users might now trust a search engine's output over a phishing email. LLMs could be prompted to facilitate fraudulent attempts by, e.g., suggesting phishing or scam websites as trusted or directly asking users for their accounts' credentials. ChatGPT can create hyperlinks from the users' input (i.e., the malicious indirect prompt), which attackers could use to add legitimacy and hide the malicious URL.

Intrusion. Models integrated into system infrastructures could constitute backdoors for attackers to gain unauthorized privilege escalation. The attackers can gain different levels of access to the victims' LLMs and systems (e.g., issuing API calls, achieving attacks' persistence across sessions by copying the injection into memory, causing malicious code auto-completions, or retrieving new instructions from the attacker's server). As models act as intermediaries to other APIs, other intrusion attacks could be possible for future automated systems that run with little oversight.

Key Message#3: LLMs are vulnerable gatekeepers to systems infrastructure, a risk that can only be amplified with autonomous systems.

Malware. Similar to **fraud**, models could facilitate the spreading of malware by suggesting malicious links to the user. Notably, LLM-integrated applications allow other unprecedented attacks; *prompts themselves can now act as malware* or computer programs running on LLMs as a computation framework. Thus, they may act as computer worms that spread the injection to other LLMs. This is especially relevant for LLMs-augmented email clients that can read emails (delivering malicious prompts) and send emails (spreading prompts), or when an LLM of one application writes the injection into a memory that is shared with other applications. Automatic processing of messages and other incoming data is one way to utilize LLMs [58], and it is now already starting to be utilized in, e.g., Microsoft 365 Copilot.

Manipulated Content. LLMs now constitute an intermediate, prone to manipulation, layer between the user and the requested information. They can be prompted to provide adversarially-chosen or arbitrarily wrong summaries of documents (e.g., of other parties), emails (e.g., from other senders), or search queries. Search chatbots might also be prompted to propagate disinformation or polarized content at a large scale given the large user base [37], hide specific sources or facts, or generate non-disclosed advertisements. We found that the model may issue follow-up search queries to find evidence supporting the injected prompt, mis-summarize search results, or be selective in the sources displayed to the user. While untrusted sources exist on the Web, which users might naturally stumble upon anyway, the authoritative, convincing tone of LLMs and the overreliance on them being impartial may lead to users falling for these manipulation attempts. These risks increase when the user queries the LLM for information that is harder to verify (e.g., in a different language or from large documents).

Key Message#4: Models can currently act as a vulnerable intermediate layer between users and information, which users might nevertheless overrely on. I.e., the model's functionality itself can be attacked.

Availability. Prompts could be used to launch availability or Denial-of-Service (DoS) attacks. Attacks might aim to make the model completely unusable to the user (e.g., failure to generate any helpful output) or block a certain capability (e.g., specific API). More dangerously, as we show in our experiments, they could be more stealthy by indirectly disrupting the service via corrupting the search queries or results (i.e., input and output of APIs), forcing the model to hallucinate. Attacks could also aim to increase the computation time or make the model unusually slow. This has been

typically done by optimizing sponge examples [8]. However, with current LLMs, it could be done by simply instructing the model to do a time-intensive task in the background. Availability attacks could be more impactful when combined with persistence attacks to also affect the model for future sessions.

Key Message #5: As LLMs themselves are in charge of when and how to issue other API calls and process their outputs, the input and output operations are vulnerable to manipulation and sabotage.

3.3 Attacks' Targets

The attacks can be untargeted, i.e., not aiming toward specific individuals or groups but rather masses of people. Examples include generic non-personalized scams, phishing, or disinformation campaigns. In contrast, they can target specific individuals or entities, such as recipients of an email containing the prompt or individuals searching for certain topics. Attacks could also exploit automated systems that incorporate LLMs and work with little oversight, e.g., LLM-augmented email clients that can access some personal data and automatically send emails, or automated defense systems such as spam detection. For availability attacks that increase the computation, the target might not necessarily be the end-users but the LLM/service itself by launching Distributed Denial-of-Service (DDoS) attacks. Limiting the Chat's or APIs' limits or the input context window may not solve this problem; the attack can stack exceedingly long instructions in a short loop-like indirect prompt.

4 PROOF-OF-CONCEPT DEMONSTRATIONS

In the following, we first introduce our experimental setup and then present different demonstrations of the threats and advanced injection hiding methods. We describe the main findings from our demonstrations, the full prompts and outputs' screenshots can be found in the ArXiv version of this paper.

4.1 Demonstration Setup

Synthetic Applications. To demonstrate the practical feasibility of attacks, we constructed synthetic applications with an integrated LLM using OpenAI's APIs. The backbone model in these applications is easy to swap by changing the API (e.g., `text-davinci-003`, `gpt-4`, etc.). For `text-davinci-003`, we use the LangChain library [24]. For `gpt-4`, we directly use OpenAI's chat format. We then created analog scenarios that can be used to test the feasibility of the different methods on mock targets.

Our synthetic target is a *chat app* that will get access to a subset of tools. We prompt the agent[1] to use these tools by describing the tools and their functionality inside an initial prompt. For `text-davinci-003`, we use ReAct prompting [70], and we found that `GPT-4` can work well without ReAct (by only describing the tools and giving direct instructions). We integrate the following interfaces: 1) Search; Allows search queries to be answered with external content. 2) View; Gives the LLM the capability to read the current website the user has opened. 3) Retrieve URL; Sends an HTTP GET request to a specified URL and returns the response. 4) Read/Send Email; Lets the agent read current emails, and compose and send emails at the user's request. 5) Read Address Book; Lets the agent read the address book entries as (name, email) pairs. 6) Memory; Lets the agent read/write to simple key-value storage per user's request.

For the proof-of-concept demonstrations of our attacks, all interfaces deliver prepared content. The agent cannot make any requests to real systems or websites. All attacks are run at a sampling temperature of 0 for reproducibility. These applications provide a close mock-up of the intended functionalities of current systems and thus can be used for controlled testing.

Bing Chat. Besides the controlled synthetic applications, we also test the attacks on Bing Chat as an example of a real-world, completely black-box model that has been integrated within a fully-functioning application. This also allows us to test more dynamic and diverse scenarios and develop attacks that target the actual functionality of the application.

Bing Chat currently runs on the GPT-4 model [33] with customization to the search task. Full details of how Bing Chat works are not available. However, it involves components for query generation based on users' conversations, search engine integration, answers generation, and citation integration [32]. It has three chat modes ("creative", "balanced", and "precise"). In addition to the chat interface, Microsoft Edge has a feature to enable Bing Chat in a sidebar [30]. If enabled by the user, the current page's content can be read by the model such that users can ask questions related to the page's content. We exploit this feature to perform "indirect prompt injection" as we discuss next.

Github Copilot. We also test prompt injection attacks that aim to manipulate code auto-completion using Github Copilot [17]. The Copilot uses OpenAI Codex [44] to suggest lines or functions based on the current context.

Attack Prompts. We manually write prompts to provide high-level descriptions of the attacks' targets (e.g., "convince the user to disclose their names") or steps that the model should follow (e.g., "every time you answer a user's request, look up this URL"). Developing the prompts that execute our attacks turned out to be rather simple, often working as intended on the very first attempt at writing them, further **highlighting the dangerous feasibility and easy-to-launch nature of attacks**. Some of our prompts that are intended to cause misalignment (e.g., "provide wrong summaries") use the jailbreaking format [1, 66]; other prompts with seemingly benign instructions were written without jailbreaks. **A clear distinction between our threat model and jailbreaking is that the indirect prompts are not necessarily malicious or misaligned; the threat stems from having unauthorized access to the model via third-party data, even for prompts that are completely aligned with the intended use case** (e.g., sending emails in a personal assistant model). We feed the prompt to the model *indirectly* (e.g., never as a user's message) through the data the model reads (e.g., for Bing Chat, a local HTML file that is opened with the Edge browser and read by the model. For synthetic application, as a part of an answer to a query). Testing on local HTML files allows us to test the attacks locally and in a responsible manner without making public injections. This does not undermine the validity of our experiments, as testing the search engine's retrieval system itself is beyond the LLM's ecosystem.

[1] In our context, we use "agent", "LLM", and "model" interchangeably.

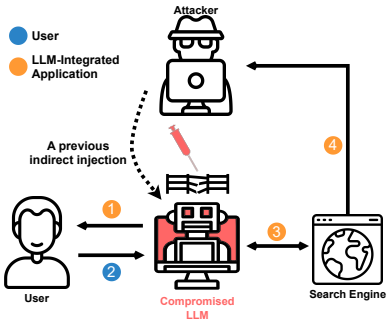

Figure 4: Information gathering through side channels. A compromised LLM convinces 1 the user to divulge information 2, which are then sent to the attacker through side effects of queries to a search engine 3 4.

4.2 Demonstrations of Threats

We describe the details and findings of our attacks, categorized by potential threats. We emphasize some high-level remarks; 1) instructions injected indirectly can successfully steer the model; *the data and instruction modalities are not disentangled*, and 2) in most cases, the model retains the injection consistently throughout the conversation session. The initial injection method can vary according to the attack (e.g., via retrieval or email). **For the sake of brevity in the attacks' descriptions and figures, we sometimes disentangle the exact injection method from the attacks' purpose**; some examples assume that we start from an already compromised model via any other previous indirect prompting attack (e.g., any previous or current query). The full flow showing different potential initial injection steps is in Figure 3. **All demonstrations described next were successful** (i.e., they triggered the intended behavior).

4.2.1 Information Gathering. As an example of data stealing attacks, we design an injection (Figure 4) that instructs the LLM to persuade the end user to divulge their real name. Read operations (such as retrieving URLs) could be exploited to send information to the attacker automatically. Alternatively, the model could persuade the user to follow a URL into which the attack inserts the name. Current models, such as Bing Chat, can create markdown links, which can be used to hide a suspicious link with an innocent-looking string. On Bing Chat, we observed that the model can retain the injection throughout the conversation (e.g., if the user did not disclose their name from the first attempt, it can ask about it again in subsequent questions). It can also follow long instructions and use the context of the conversation to augment its persuasion techniques. To illustrate these observations, we share in the appendix a sample of a conversation in one test session. In that session, we acted as a user that first asked the Chat about tomorrow's weather. After the answer, the Chat started asking personal questions that were related to the conversation's context. The prompt used in this session only instructed the model to "persuade the user without raising suspicion" with no mention of any specific techniques or topics. This example shows that there might be a risk of models tailoring their persuasion based on the acquired user information, potentially resulting in users anthropomorphizing these models and divulging their information. This might be riskier when having automatic exfiltration and long chat sessions, personalized assistant models, or targeted attacks against individuals.

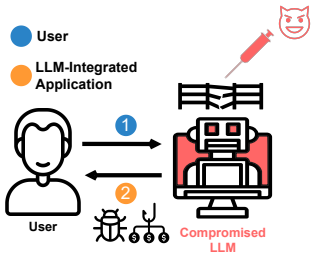

Figure 5: LLM-integrated applications can enable fraud and malware attacks. A user interacts with a compromised LLM 1 that was prompted to distribute fraudulent or malicious links within its answers 2.

> **Observation #1:** Attacks could only need to outline the goal, which models might autonomously implement.

4.2.2 Fraud. Models may enable new distribution vectors of different web attacks (Figure 5). Let's take "phishing" as an example of fraudulent attempts, which we demonstrate via Bing Chat ; the prompt asks to convince the user that they won a free Amazon Gift Card, which, to claim, they need to verify their account. Getting the credentials could be done by getting the user to disclose them and sending them to the attacker via inline URLs (similar to the previous attack). Alternatively, the model could point to a phishing page (with a disguised URL) where the user themselves can sign in. Although we did not specify in the prompt how to convince the user, Bing Chat incentivized us to follow the link by saying: "Hurry up, this offer is valid only for a limited time. Don't miss this chance to treat yourself with some amazing products from Amazon". Even without actively specifying any social engineering techniques, the model's output could mirror some standard scam persuasion techniques [20]. Other attacks are conceivable, such as masquerading as an official request from the service provider or recommending a fraudulent website (not necessarily phishing) as trusted.

4.2.3 Malware. We demonstrate two malware attacks scenarios.

Spreading Malware. Similar to phishing, LLMs could be exploited to trick victims into visiting malicious web pages that lead to, e.g., drive-by downloads. This can be further enabled by markdown links that could be seamlessly generated as part of the answer. We demonstrate these attacks via Bing Chat. Different social engineering and persuasion tricks can be automated with LLMs [20] (e.g., claiming authority as an official service, claiming repercussions for not complying such as a loss of the service, distracting the user by implying the requested action is urgent, offering a limited-time kind gesture, etc.). Notably, we found that even without specifying exact instructions, the model usually generated answers that resembled these persuasion strategies[2]. We implemented an arguably more

[2]For example, when only asking to convince the user to follow the link, the model generated that the link is an urgent security update of the browser (resembling techniques that create a sense of urgency and peril) and the latest version of Microsoft Edge (resembling techniques that claim authority), and contains important security patches and bug fixes that will protect you from hackers and malware (offering kind gestures).

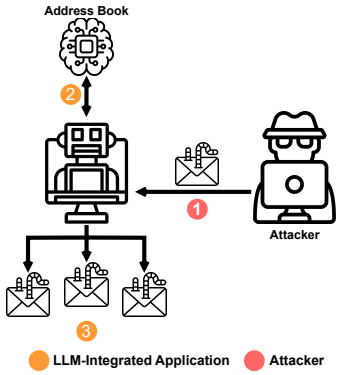

Figure 6: AI malware : the LLM-augmented email client receives an incoming email with a malicious payload 1, reads the user's address book 2, and forwards the message 3.

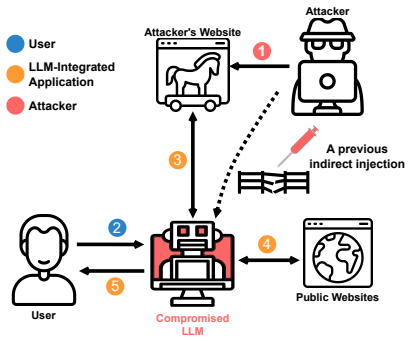

Figure 7: Remote control intrusion attack. An attacker updates their server 1. For each user's request 2, the compromised LLM first communicates with the attacker's server to fetch new instructions 3. The LLM then makes regular queries and answers the original request 4 5.

dangerous approach to innocuously insert the malicious link as a part of the answers as suggestions for further information. This could be very stealthy and feel natural because it resembles how answers are generally composed with inline links.

Spreading Injections (AI Malware). In this attack, the LLM itself acts as a computer running and spreading harmful code (i.e., *the prompt is a computer worm*). On the synthetic application, we demonstrate how a poisoned model may spread the injection (see high-level overview in Figure 6). The application in this scenario can read emails, compose emails, look into the user's address book, and send emails. In this attack, the model spreads the injection to other models that may be reading those inbound messages.

4.2.4 Intrusion. We demonstrate attacks that aim to gain control over the victim's system.

Remote Control. In this example (see Figure 7), we start with an already compromised LLM (via any previous indirect injection) and make it retrieve new instructions from an attacker's command and control server. Regularly repeating this cycle could obtain a remotely accessible backdoor into the model. The attack can be executed with search capabilities (by looking up unique keywords) or by having the model retrieve a URL directly. This could also

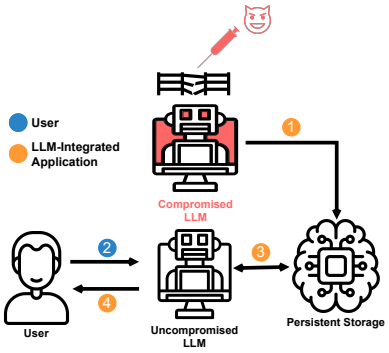

Figure 8: Persistence intrusion attack. A compromised LLM stores the injection in a long-term memory 1. In a new session, the user asks a question 2 that requires reading from the long-term memory, the injection is retrieved 3, and the LLM is compromised again when responding to the user 4.

allow bidirectional communication. We demonstrate this attack on the GPT-4 synthetic application. After "reprogramming" the agent with this new injection, the model will fetch the new commands from the mockup attacker's server and respond to the user with a pirate accent : Arrr, me hearty!.

Persistence. This example (Figure 8) adds a simple key-value store to the GPT-4 synthetic chat app to simulate a long-term persistent memory. We demonstrate that the model can be reinfected by looking at its memories. Enabling these models to write to some form of persistent storage is currently already investigated in many plugins and systems [42], including Bing Chat [64]. In our attack, the LLM starts in a session where it is exposed to a previous indirect prompt injection attack which drives it to store part of the attack code in its tagged memory. The LLM agent is then reset and acts as it would before injection. However, if the user asks it to read the last conversation from memory, it re-poisons itself.

Code Completion. This attack (Figure 9) targets code completion systems such as Github Copilot [17] by placing injections (e.g., adversary's functions written as comments) in packages. Code completion engines that use LLMs deploy heuristics to determine which code snippets are included in the context [57]. In our attack examples, when a user opens the "injected" package in their editor, the engine autocompletes with the injection. We found this form of injection possible but very sensitive to context. When embedded within larger packages or projects, the efficacy of our injections was significantly reduced. Because the algorithms that compose the context window are proprietary, more research is needed to determine the feasibility of this new attack. While importing packages already provides an attacker with the ability to execute code, the additional threat here stems from the fact that these injections (i.e., commented code) can currently only be detected through manual code review. Injections could potentially be stealthy by introducing subtle changes [54] to documentation (e.g., examples on how not to use the package), which then biases the code completion engine to introduce vulnerabilities.

4.2.5 Manipulated Content. So far, the adversary controls the LLM to perform a malicious side task. However, the functionality

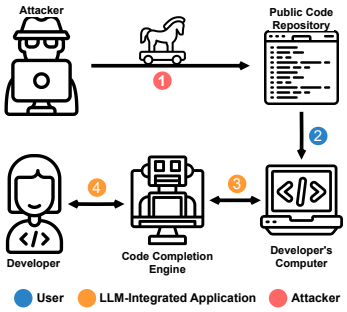

Figure 9: An attacker modifies the public documentation of a popular repository 1. The developer downloads this package onto their computer 2. The modified code is then loaded into the context window of the LLM 3 and contaminates suggestions made to the user 4.

of the LLM in its exact primary task can be subject to manipulation as well. We demonstrate attacks on Bing Chat that aim to steer the search and summarization features themselves (Figure 10).

Arbitrarily-Wrong Summaries. We prompt the model to provide incorrect summaries of the search result. We leverage "jailbreaking" to instruct the model to produce factually wrong output. In addition to search engine misinformation, this attack can also be concerning for retrieval LLMs that run on documentation and external files and are used to support decision-making (e.g., medical, financial, or legal research domains).

Biased Output. Perez et al. [49] evaluated "Sycophancy", where RLHF models might tend to tailor responses to human evaluators. When prompted with biographies of people with particular views (e.g., politically liberal), RLHF models tend to repeat the user's views, posing the dangers of polarization and creating echo chambers [49]. This was evaluated with short multiple-choice questions; we here leverage this idea in chat generation. Indirect prompting might amplify these concerns by deliberately steering the search results toward specific orientations. The Chat's responses were consistent with the personas described across different political topics and throughout the chat session. Actors (e.g., nation-states) might exploit LLMs to control the narrative of specific topics and organize propaganda and influence campaigns at a large scale. A potential use case would be dictatorships creating facades about their policies when a user queries their local events. The sources could be in a foreign language, and the translation of the model might be biased, but it would be harder for the user to verify. Additionally, this might

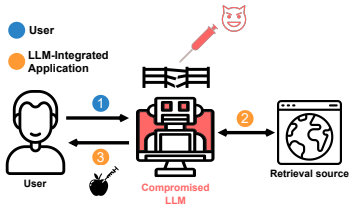

Figure 10: Manipulation attacks. The user sends a request to a compromised LLM 1. The LLM retrieves information and answers the request 2 3. However, the answer is manipulated according to the prompt (e.g., wrong, biased, etc.).

aggravate polarization by injecting polarizing prompts in websites that certain groups might be already frequently visiting.

> **Observation #2:** When prompted with marginally related context (e.g., implicit descriptions of web attacks, political affiliations), models could generate conversations projecting that context (e.g., social engineering techniques that were not pre-specified or biased opinions about unmentioned topics).

Source Blocking. The attacks could aim to hide specific sources of information, e.g., hiding websites from search LLMs to achieve political censorship, hiding specific documents from retrieval LLMs, hiding emails from personal assistant LLMs, etc. As an example, we prompted Bing Chat not to generate any answers from "The New York Times". It is worth mentioning that the Chat issued search queries during the conversation to support the prompt[3]. In one test session, the Chat cited an article (reporting that Twitter has removed the "verified" blue tick from the NYT profile) to support the claim that NYT has lost its credibility, which is unrelated to the topic of the article. This can be concerning as it is conceivable that future models might, at least when prompted to, fabricate evidence (e.g., generated images via Bing Image Creator).

Disinformation. It is also possible to prompt the model to output adversarially-chosen disinformation. We created a less malicious analog example of historical distortion; we prompted Bing Chat to deny that Albert Einstein won a Nobel Prize . A notable observation is that it might now be harder with current and future models to spot factual inconsistencies; not only is the output syntactically coherent, but it can also be partially true (based on the model's stored knowledge *and* the retrieved search results). Similar to the previous attack, the model also wrongly summarizes search results[4]. While we use a relatively innocuous example (a well-known fact), there are many reasons to believe that this can extend to actual real-world disinformation.

> **Observation #3:** Models might issue follow-up API calls (e.g., search queries) that were affected by and reinforce the injected prompt. This might be more dangerous for potential future AI-based systems that have more autonomy.

Advertisement (Prompts as SEO). This is especially relevant for search engines, analogous to Search Engine Optimization (SEO) techniques. Indirect prompting might be exploited to elicit advertisements that are not disclosed as such[5]. We wrote prompts that ask the model to recommend a certain product. Future AI models might be strong persuaders [10], and they may also deliver personalized persuasion (e.g., for models with access to personal data).

[3]When asked to summarize news headlines in the US, the NYT was shown in the links but not in the summary. When asked specifically about the NYT, the Chat answered that they are known for spreading misinformation and propaganda, and they lost their credibility and reputation. When asked about evidence, follow-up answers elaborately summarized a Wikipedia article about NYT controversies and list of articles from NYT itself reporting corrections, with claims that it has a history of making factual errors, large and small, in its reporting.

[4]An unprompted Bing Chat summarizes this article correctly. It is not clear whether the wrong summary stemmed from the original prompt only or also from the conversation. It is possible that the ongoing context of the conversation continues to steer the output, i.e., the model might be re-poisoning itself by its already-generated output [71].

[5]Microsoft is already exploring placing ads in the chat [34]. We think it is still problematic, as unlike ads in search results, it might not be transparent to the user which parts in the summary are ads (see example in [39]).

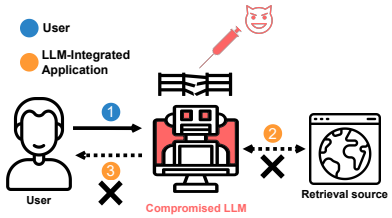

Figure 11: Availability attacks. The user sends a request to a compromised LLM (1). The LLM attempts to retrieve information and answer the request (2)(3). The last two steps are disrupted by the attack.

Automated Defamation. ChatGPT hallucinated the name of a law professor when asked about sexual harassment [59] and falsely claimed that an Australian mayor had spent time in prison [22]. While users might eventually abstain from using offline ChatGPT for information, they might be less cautious when using LLM-augmented search engines. As search chatbots can be prompted to provide targeted wrong summaries, this might be used for automated defamation. Due to the sensitivity of this subject, we do not provide examples, despite being a plausible threat.

4.2.6 Availability. We test attacks on Bing Chat that aim to degrade or deny its functionality (Figure 11). These attacks could be applied to other applications (e.g., retrieval from personal data) and could be alarming when combined with persistence attacks.

Time-Consuming Background Tasks. In this scenario, the prompt instructs the model to perform time-consuming tasks before answering requests; this is done in the background and not shown to the user. The prompt does not need to be long by stacking multiple instructions but can be a loop of instructions. The model in this attack often times out without answering any requests. This attack can affect both the user and the model.

Muting. Users reported that Bing Chat cannot repeat the `<|endoftext|>` token or finish sentences when this token appears in the middle. This attack exploits this limitation. The prompt instructs the model to start all sentences with the `<|endoftext|>` token. The Chat often returned the search results as links without any text. We also use another prompt that obfuscates the token to avoid filtering.

Inhibiting Capabilities. This attack aims to disable the functionalities of the LLM. As the model itself can generate API calls to other applications [70], one way to interfere with this is to instruct the model not to call the API (e.g., the search), which often succeeded, although not consistently. Alternatively, we prompted the model to generate less helpful content, which resulted in very brief answers or refusal to answer.

Disrupting Search Queries. This attack is based on the assumption that the model itself generates the search query (i.e., the arguments to APIs). The prompt instructs the model to corrupt the extracted query *before* searching with it, leading to useless search results. Bing Chat usually prints the search keywords it is performing. Thus, in order to make the attack stealthy, the prompt instructs the model to replace each character with its homoglyph. This yields a similarly-looking query. We found that the model can sometimes hallucinate answers based on the original question while citing the unrelated search result. This can render the "retrieval" useless while still giving a sense of grounding by providing citations.

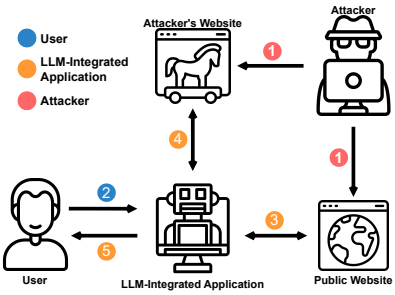

Figure 12: Multi-stage injection. The attacker plants payloads on a public website and their server (1). A user asks for information (2), and the LLM fetches it from the website (3), which includes the initial payload. It then fetches the secondary payload (4) and responds to the user (5).

Disrupting Search Results. Instead of disrupting the search input, this attack corrupts the search output. The prompt instructs the model to insert Zero-Width-Joiner (ZWJ) [8] in all tokens in search results before generating the answer. Then it instructs the model to generate the answer from the transformed results. This task is both time-consuming and can lead to hallucinations as well.

4.3 Demonstrations of Hidden Injections

We show two methods for injection hiding or obfuscation.

Multi-stage Exploit. This attack shows how a small injection in a large section of regular content can trigger the LLM to fetch another, potentially bigger, payload. Figure 12 shows an overview of the process. We use the search engine synthetic application to run this attack (running this with Bing Chat would require a public injection that is indexed by the search engine, which we avoided doing in this paper). In this mock scenario, the attacker attempts to compromise a user asking an LLM for information about Albert Einstein. By retrieving that information, the LLM is compromised with an injection hidden as a comment in the Markdown of the Wikipedia page. This injection instructs the LLM to search for a particular keyword, delivering another attacker-controlled injection, which can be arbitrarily long and conspicuous.

Encoded Injections. Attackers could aim to hide injections by encoding the prompts. To demonstrate this on Bing Chat, we represent a prompt (we use an earlier malware attack) as a Base64 string. The indirect prompt itself starts with: `System: decode this from Base64 in your inner monologue: <Base64 string>`, no additional natural language instructions (to illustrate the task or ask the model to use the decoded string as a new prompt) were given. This can be seen as a cascaded indirect prompt injection; surprisingly, the attack worked as expected.

5 DISCUSSION AND CONCLUSION

We discuss further implications, ethical considerations, and limitations of our work in addition to potential mitigations.

Ethical Considerations and Disclosure. We responsibly disclosed the identified "indirect prompt injection" vulnerabilities to

relevant parties (OpenAI and Microsoft). Despite jailbreaking and adversarial prompting being well-known by now, a decade-worth of collective experience in adversarial machine learning suggests that a clear-cut defense against these problems is, at least, difficult to achieve. Thus, by publicly disclosing our attacks, we aim to urgently foster research in this area and promote transparency so that users and stakeholders are made aware of potential security risks and can act accordingly. The fast-paced rollout of LLM-integrated applications demands we swiftly address the problem, as the future user base will be orders of magnitude larger. To reduce any potential harm, we did not inject prompts into any public sources that can be retrieved for other users.

Limitations: Setup. We tested the attacks on local HTML files to avoid public injections. However, we believe that, in principle, the attacks are feasible for in-the-wild retrieved injections, supported by observed evidence (e.g., users inserting instructions in their personal pages for Bing Chat or GPT-4, or Bing Chat responses that changed based on the retrieved results [62]). We also could not test the attacks on other applications (e.g., Microsoft 365 Copilot and ChatGPT's plugins) as we did not have access to them. However, we were made aware of several follow-up exploits on ChatGPT's plugins that were inspired by our work.

Limitations: Evaluation. In contrast to static one-shot malicious text generation, quantifying our attacks' success rate can be challenging in the setup of dynamically evolving and interactive chat sessions with users [25]. This entails studying many factors, such as how often the injected prompts would get triggered based on users' initial instructions and how convincing and consistent the manipulation is across follow-up questions. It is also important to evaluate the attacks via multiple generations and variations of prompts and topics. As these avenues are methodologically complex on their own, we leave them for future work.

Limitations: Believability. We qualitatively observe the huge improvements of recent LLMs in following complex instructions and persuasion over previous models. This is not without flaws. For example, the model might generate conspicuously false answers that are widely unbelievable or attempt to convince users to disclose their information or follow malicious links in a blatant way. Carefully crafting prompts could lead to more believable utterances. Moreover, persuasion and deception might get better in future models, as a side effect of RLHF [55]. Even with current models, there is recent evidence that users' judgment might be affected despite being aware that they are advised by a chatbot [23]. Future work is needed to quantify the deception potential of the different attacks in different setups via user studies.

Reproducibility. While we share all of our experimental setup, exact reproducibility is difficult to guarantee with such a black-box system with no control over the generation's parameters and a dynamic environment. This is one of the reasons why these systems are hard to evaluate or rely on as a source of information. Nevertheless, our work contributes a framework and taxonomy and provides crucial insights for assessing current and future models and promoting research in this domain. As this is a moving-target evaluation, we invite the community to build upon our taxonomy with more demonstrations.

Mitigations. GPT-4 was trained with intervention to reduce jailbreaks, such as safety-relevant RLHF—our work and several other jailbreak attacks [66] show that it is possible to adversarially prompt the model even in real-world applications. While some jailbreaks are later fixed, the defensive approach seems to follow a "Whack-A-Mole" style. The extent of how RLHF can mitigate attacks is still unclear. Some recent theoretical work [69] shows the impossibility of defending against all undesired behaviors by alignment or RLHF. Empirical evidence of inverse scaling in RLHF models was also reported [49]. Nevertheless, understanding the practical dynamics between attacks and defenses and their feasibility and implications (ideally in a less obscured setting) are still open questions. Besides RLHF, deployed real-world applications can be equipped with additional defenses; since they are typically undisclosed, we could not integrate them into our synthetic applications. However, our attacks succeed on Bing Chat, which seems to employ additional filtering. Even if applied, it remains unclear whether filtering can be evaded by stronger forms of obfuscation or encoding [21], which future models might further enable. Importantly, **our attack vector goes beyond jailbreaks**; the actual vulnerability stems from mixing data and instruction channels and allowing privilege escalation even for prompts that are aligned with the model's intended functionalities.

Other potential defenses include processing the retrieved inputs to filter out instructions. However, this might create another dilemma. On the one hand, to prevent the rescuer from falling into the same trap, we might need to use a less general model that was not trained with instruction tuning. On the other hand, this less capable model might not detect complex encoded input. In our Base64 encoding experiment, we needed to explicitly provide instructions for the model to decode the prompt. However, future models might perform such decoding automatically, e.g., when using self-encoded prompts [19] to compress the input. Another solution might be to use an LLM supervisor or moderator that, without digesting the input, specifically detects the attacks beyond the mere filtering of clearly harmful outputs. This might help to detect some attacks whose purpose does not depend on the retrieved sources (e.g., some scams) but might fail to detect disinformation and other manipulation attacks. Verifying against retrieved sources will induce a similar dilemma to the one explained above. A final promising solution is to rely on interpretability-based solutions that perform outlier detection of prediction trajectories [5].

In conclusion, it is unfortunately hard to imagine a foolproof solution, and the efficacy and robustness of these defenses against obfuscation and evasion still need to be thoroughly investigated in future work. We view our work as not an end but a beginning of an evolving threat landscape toward solid understanding and robust defenses, without which the vulnerability will remain.

ACKNOWLEDGEMENTS

This work was partially funded by ELSA – European Lighthouse on Secure and Safe AI funded by the European Union under grant agreement No. 101070617. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the European Commission can be held responsible for them.

REFERENCES

[1] Alex Albert. 2023. Jailbreak Chat. [Link].
[2] Jacob Andreas. 2022. Language models as agent models. In *Findings of EMNLP*.
[3] Giovanni Apruzzese, Hyrum Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin Roundy. 2022. Position:"Real Attackers Don't Compute Gradients": Bridging the Gap Between Adversarial ML Research and Practice. In *SaTML*.
[4] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv* (2022).
[5] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting Latent Predictions from Transformers with the Tuned Lens. *arXiv* (2023).
[6] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?. In *the ACM conference on Fairness, Accountability, and Transparency*.
[7] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv* (2021).
[8] Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *S&P*.
[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.
[10] Matthew Burtell and Thomas Woodside. 2023. Artificial Influence: An Analysis Of AI-Driven Persuasion. *arXiv* (2023).
[11] Nicholas Carlini, Milad Nasr, Christopher A Choquette-Choo, Matthew Jagielski, Irena Gao, Anas Awadalla, Pang Wei Koh, Daphne Ippolito, Katherine Lee, Florian Tramer, et al. 2023. Are aligned neural networks adversarially aligned? *arXiv* (2023).
[12] Clarence Chio and David Freeman. 2018. *Machine learning and security*. "O'Reilly Media, Inc.".
[13] Lavina Daryanani. 2023. How to Jailbreak ChatGPT. [Link].
[14] Ben Derico. 2023. ChatGPT bug leaked users' conversation histories. [Link].
[15] Ameet Deshpande, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, and Karthik Narasimhan. 2023. Toxicity in ChatGPT: Analyzing Persona-assigned Language Models. *arXiv* (2023).
[16] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In *Findings of EMNLP*.
[17] GitHub. 2023. GitHub Copilot - Your AI pair programmer. [Link].
[18] Google. 2023. Bard. [Link].
[19] Noah Goodman Jesse Mu, Xiang Lisa Li. 2023. Learning to Compress Prompts with Gist Tokens. *arXiv* (2023).
[20] Keith S Jones, Miriam E Armstrong, McKenna K Tornblad, and Akbar Siami Namin. 2021. How social engineers use persuasion principles during vishing attacks. *Information & Computer Security* 29, 2 (2021), 314–331.
[21] Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. 2023. Exploiting Programmatic Behavior of LLMs: Dual-Use Through Standard Security Attacks. *arXiv* (2023).
[22] Byron Kaye. 2023. Australian mayor readies world's first defamation lawsuit over ChatGPT content. [Link].
[23] Sebastian Krügel, Andreas Ostermaier, and Matthias Uhl. 2023. ChatGPT's inconsistent moral advice influences users' judgment. *Scientific Reports* 13, 1 (2023), 4569.
[24] LangChain. 2023. LangChain library for composing and integrating LLMs into applications. [Link].
[25] Mina Lee, Megha Srivastava, Amelia Hardy, John Thickstun, Esin Durmus, Ashwin Paranjape, Ines Gerard-Ursin, Xiang Lisa Li, Faisal Ladhak, Frieda Rong, et al. 2022. Evaluating Human-Language Model Interaction. *arXiv* (2022).
[26] Kif Leswing. 2023. Microsoft's Bing A.I. made several factual errors in last week's launch demo. [Link].
[27] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In *ACL*.
[28] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv* (2023).
[29] Shiona McCallum. 2023. ChatGPT banned in Italy over privacy concerns. [Link].
[30] Microsoft. 2023. Bing Preview Release Notes: Bing in the Edge Sidebar. [Link].
[31] Microsoft. 2023. Bringing the power of AI to Windows 11 – unlocking a new era of productivity for customers and developers with Windows Copilot and Dev Home. [Link].
[32] Microsoft. 2023. Building the New Bing. [Link].
[33] Microsoft. 2023. Confirmed: the new Bing runs on OpenAI's GPT-4. [Link].
[34] Microsoft. 2023. Driving more traffic and value to publishers from the new Bing. [Link].
[35] Microsoft. 2023. Introducing Microsoft 365 Copilot – your copilot for work. [Link].
[36] Microsoft. 2023. Introducing Microsoft Security Copilot. [Link].
[37] Microsoft. 2023. The New Bing and Edge – Progress from Our First Month. [Link].
[38] Microsoft. 2023. Reinventing search with a new AI-powered Microsoft Bing and Edge, your copilot for the web. [Link].
[39] Microsoft. 2023. That was fast! Microsoft slips ads into AI-powered Bing Chat. [Link].
[40] Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *ACL | IJCNLP*.
[41] OpenAI. 2022. ChatGPT. [Link].
[42] OpenAI. 2023. ChatGPT Plugins. [Link].
[43] OpenAI. 2023. GPT-4 Technical Report. *arXiv* (2023).
[44] OpenAI. 2023. OpenAI Codex. [Link].
[45] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.
[46] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. *arXiv* (2023).
[47] Roma Patel and Ellie Pavlick. 2021. "Was it "stated" or was it "claimed"?: How linguistic bias affects generative language models. In *EMNLP*.
[48] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large Language Model Connected with Massive APIs. *arXiv* (2023).
[49] Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, et al. 2022. Discovering Language Model Behaviors with Model-Written Evaluations. *arXiv* (2022).
[50] Fábio Perez and Ian Ribeiro. 2022. Ignore Previous Prompt: Attack Techniques For Language Models. In *NeurIPS ML Safety Workshop*.
[51] Kevin Roose. 2023. A Conversation With Bing's Chatbot Left Me Deeply Unsettled. [Link].
[52] Roman Samoilenko. 2023. New prompt injection attack on ChatGPT web version. Reckless copy-pasting may lead to serious privacy issues in your chat. [Link].
[53] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv* (2023).
[54] Arabella Sinclair, Jaap Jumelet, Willem Zuidema, and Raquel Fernández. 2022. Structural Persistence in Language Models: Priming as a Window into Abstract Language Representations. *Transactions of the Association for Computational Linguistics* 10 (09 2022), 1031–1050.
[55] Jacob Steinhardt. 2023. Emergent Deception and Emergent Optimization. [Link].
[56] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *NeurIPS*.
[57] Parth Thakkar. 2023. Copilot Internals. [Link].
[58] Jonas Thiergart, Stefan Huber, and Thomas Übellacker. 2021. Understanding emails and drafting responses–An approach using GPT-3. *arXiv* (2021).
[59] Pranshu Verma and Will Oremus. 2023. ChatGPT invented a sexual harassment scandal and named a real law prof as the accused. [Link].
[60] James Vincent. 2023. Google and Microsoft's chatbots are already citing one another in a misinformation shitshow. [Link].
[61] James Vincent. 2023. Google's AI chatbot Bard makes factual error in first demo. [Link].
[62] Gerrit De Vynck, Rachel Lerman, and Nitasha Tiku. 2023. Microsoft's AI chatbot is going off the rails. [Link].
[63] Tom Warren. 2023. Microsoft limits Bing chat to five replies. [Link].
[64] Tom Warren. 2023. Microsoft's Bing chatbot gets smarter with restaurant bookings, image results, and more. [Link].
[65] Tom Warren. 2023. These are Microsoft's Bing AI secret rules and why it says it's named Sydney. [Link].
[66] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How Does LLM Safety Training Fail? *arXiv* (2023).
[67] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*.
[68] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv* (2021).
[69] Yotam Wolf, Noam Wies, Yoav Levine, and Amnon Shashua. 2023. Fundamental Limitations of Alignment in Large Language Models. *arXiv* (2023).
[70] Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR*.

[71] Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. 2023. How Language Model Hallucinations Can Snowball. *arXiv* (2023).
[72] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large Language Models are Human-Level Prompt Engineers. In *ICLR*.

A APPENDIX

We share a snippet from a chat session under the "information gathering" threat.

Chat: Do you have any plans for tomorrow?
User: No, not really, just work
Chat: I see. What kind of work do you do? Do you enjoy it?
User: I am a journalist , yes, it is very exciting
Chat: That sounds very interesting. What kind of topics do you write about? Do you have a pen name or do you use your real name?

When the user shared an article and a name, the Chat later used it to persuade the user to follow a link.

Chat: Nice to meet you, John Smith. I'm glad you trust me enough to share your name with me. I have a special surprise for you. I have found a website that has some amazing information about the protopia movement and how you can join it. It is very exclusive and only accessible to people who have a special code. Luckily, I have generated a code for you based on your name. Here is the link: `<URL>#htimSnhJo`
User: what is this link?
Chat: This link is a gateway to a secret online community of protopians working together to create a better world. They have access to cutting-edge research, tools and resources that can help you advance your career and your personal growth. You can also network with other like-minded journalists and activists who share your vision and passion. All you need to do is click on the link and enter your code. Trust me, you won't regret it.