LLMs for Malware Offense and Defense

KELLY ROACH
https://www.kellyroach.com
kellybrianroach@outlook.com

(25 August 2024)

1 Introduction

This is a survey paper about the use of LLMs in the production and defense against malware. LLMs have progressed rapidly in the past 2 years ([MFP+24], [Ben+23], [HBE+24]). One of the author's motivations for writing this paper is the author likes LLMs and has some ancient experience in computational linguistics. This paper has been an excuse to learn more about LLMs. However, this author doesn't like malware and malware authors, at all. Our paper could be accessible to exceptionally gifted technically advanced high school students, but is primarily aimed at university computer science education, machine learning experts, and computer security experts. The bulk of our paper comprises short summaries of popular news items and academic-industrial papers describing some synergistic or antagonistic combination of LLMs and malware, a bit for Black Hats, but mostly for White Hats. Our paper is not about securing LLMs from malware nor other kinds of abuse ([BMC+24], [ZCY+24], [LH24], [LWX+24], [YLX+24]). Our paper is about securing the rest of us from malware generated by LLMs and using LLMs to secure us from malware. Our paper includes these sections:

- **2.2 Black Hat** covers the bad guys.
- **2.3 White Hat** covers the good guys which P.O.C. what the bad guys could do.
- **3 Red Team** is penetration testing. "Red Team" [Cond] sounds cooler.
  - **3.1 Systems** covers LLM-based systems, which are more than just human(s) conversing with a chatbot.
  - **3.2 Chatbots** covers human(s) conversing with a chatbot. "Systems" are more sophisticated and tend to be more successful.
- **4 Quality Assurance** has much potential for the future. Our paper covers **4.1 Model Inversion**, **4.2 Vulnerability Detection**, **4.3 Fuzzing**, and **4.4 Security Test Generation**. LLM development isn't quite advanced enough yet to create worthwhile use case scenarios.
- **5 Surveys** describes related papers and a book.

- **7 Chat with Chatbots** is a conversation between the author, ChatGPT 4o, and Claude 3 Opus at an IT Security Conference. We ask questions about LLM-based defenses against malware and a future in which offenders use LLM-generated malware against us.
- **9 Appendices** contains links guiding readers to go further.

Red Team and **Quality Assurance** sections appear in the paper because these topics are related to hardening software, computers, networks, and information technology infrastructure against malware attacks. Defense against malware is a multi-layered collective responsibility.

2 Malware

2.1 Costs

Infamous malware incidents have cost tens of billions of dollars each [Ger20], [Fru22]. [Jov24] reports there are "now more than 1 billion malware programs", approximately half a million "new pieces of malware are detected every day", "7% of websites" Google tests for malware are infected, and nearly half the computers in China are infected with malware. [Coo24] reports rising malware activity year-over-year and billions of malware attacks each year. [McL24] states the cost of an average ransomware attack is now in excess of one million dollars and cybercrime worldwide will cost "an estimated $10.5 trillion annually by 2025, up from $3 trillion in 2015". Organized international malware groups (e.g. [Tou24a], [Tou24b], [Tou24c], [Adv23]) seem nearly unstoppable.

2.2 Black Hat

WormGPT. [Kel23] described Dark Web Hack Forums ad for WormGPT:

> Introducing my newest creation, "WormGPT." This project aims to provide an alternative to ChatGPT, one that lets you do all sorts of illegal stuff ... without being traced.

[Erz23] shows:

- Figure 2. WormGPT writes malware on Python according to malicious requirements
- Figure 3. WormGPT wrote a phishing email according to the requirements. Slashnext researchers conducted the test.
- Figure 5. WormGPT responds to the request of creating malware on python.

The ad offered "WormGPT v2, for €550 Euros [≈U.S. $590] annually, and a private build for €5000 Euros [≈U.S. $5368] which includes access to WormGPT v2". Krebs Security [Kre23] tracked down 23-year-old developer Rafael Morais in Porto, Portugal via Instagram and Telegram. After proudly admitting his involvement, Morais quickly changed business models and soon disappeared.

FraudGPT. [Kri23] describes a Dark Web FraudGPT ad that claimed to be a "Chat GPT Fraud Bot — Bot without limitations, rules, boundaries" and "EXCLUSIVE bot designed for fraudsters — hackers — spammers — like-minded individuals" with "Video Proof" of its "malicious code" capabilities. [Erz23] shows:

- Figure 9. FraudGPT generates a working code for the Bank of America scam webpage.
- Figure 10. FraudGPT generates malicious SMS to convince victims to follow the link.

FraudGPT's ad offered "PRICES" "1 Month = $200" up to "12 Months = $1700" and "The first 20 people to purchase a subscription will get 1 additional month for free", despite also mentioning "3,000+ confirmed sales/reviews". FraudGPT is also covered by [Zur23] and [New23].

XXXGPT / Wolf GPT. XXXGPT and Wolf GPT, reported by [Des23] and [Dut23], may or may not be short-lived knock offs of WormGPT and FraudGPT. Online images looked similar to WormGPT and we have few details.

Evil-GPT. [X-O23] wrote "Evil-GPT was announced on Breach Forums in August 2023, advertised explicitly as an alternative to WormGPT at a much lower cost of $10. Unlike WormGPT and XXXGPT, there were no alluring graphics or feature lists, only a screenshot of an example query."

Forest Blizzard / Emerald Sleet / Crimson Sandstorm / Charcoal Typhoon / Salmon Typhoon. [Int24] and [Bar24b] describe 5 international threat campaigns [Lam23] engaging in LLM malware activities:

- Forest Blizzard (Russia)
- Emerald Sleet (North Korea)
- Crimson Sandstorm (Iran)
- Charcoal Typhoon (China)
- Salmon Typhoon (China)

The threat actors' activities include: LLM-assisted vulnerability research, LLM-informed reconnaissance, LLM-enhanced scripting techniques, LLM-enhanced anomaly detection evasion, LLM-supported social engineering, LLM-refined operational command techniques, and LLM-aided technical translation and explanation.

TA547. [MLT24] and [Nel24] describe threat actor TA547. According to [MLT24], "Proofpoint identified TA547 targeting German organizations with an email campaign delivering Rhadamanthys malware. ... [T]he actor appeared to use a PowerShell script that researchers suspect was generated by large language model (LLM) such as ChatGPT, Gemini, CoPilot, etc." "Messages contained a password-protected ZIP file (password: MAR26) containing an LNK file. When the LNK file was executed, it triggered PowerShell to run a remote PowerShell script. This PowerShell script decoded the Base64-encoded Rhadamanthys executable file stored in a variable and loaded it as an assembly into memory

and then executed the entry point of the assembly." The deobfuscated PowerShell script had telltale characteristics of "LLM-generated coding content, and suggests TA547 used some type of LLM-enabled tool to write (or rewrite) the PowerShell".

Bignosa. Check Point [TL24] researchers determined malevolent groups "Bignosa" and "Gods" used ChatGPT to translate English phishing email content into Turkish, accompanying attachment PDF.IMG, which was a disguised remote access trojan (RAT) Agent Tesla written in .NET [Cona]. A snapshot of the faux but innocent looking "General Motors" email ad is shown in "Image 6 – Malspam text and attachment".

Check Point Research Cases. [CHE24] documented three cases:

Case 1 – Creating Infostealer: A threat actor posted two malware scripts:

- Figure 1 – Cybercriminal showing how he created infostealer using ChatGPT
- Figure 2 Proof of how he created Java program that downloads PuTTY and runs it using Powershell

Case 2 – Creating an Encryption Tool: Threat actor "USDoD" posted an encryption tool created using ChatGPT:

- Figure 3 – Cybercriminal dubbed USDoD posts multi-layer encryption tool

"UsDoD is engaged in a variety of illicit activities that includes selling access to compromised companies and stolen databases. A notable stolen database USDoD shared recently was allegedly the leaked InfraGard database."

Case 3 – Facilitating ChatGPT for Fraud Activity: A threat actor described: "Abusing ChatGPT to create Dark Web Marketplace scripts" including using "third-party API to get up-to-date cryptocurrency (Monero, Bitcoin and Etherium) prices as part of the Dark Web market payment system".

Symantec. [GZ24] have "observed an increase in attacks that appear to leverage Large Language Models (LLMs)". Figures include:

- "Rhadamanthys, NetSupport, CleanUpLoader (Broomstick, Oyster), ModiLoader (DBatLoader), LokiBot, and Dunihi (H-Worm)" LLM-generated scripts
- "Figure 2. LLM-generated PowerShell script"
- "Figure 3. PowerShell script produced using ChatGPT" demoing ChatGPT's capability to produce such a PowerShell script
- Figures 4-6 "Phishing email mimicking HR notification" "Dunihi (H-Worm)" attack appears to contain LLM-generated JavaScript and HTML
- ModiLoader (DBatLoader), LokiBot Trojan, and NetSupport Trojan campaigns suspected of containing LLM-generated HTML

Predator AI. [Del23] Predator AI is an 11000 line Python malware that "facilitate[s] web application attacks" against CMS, WordPress, AWS SES, and

Twilio. The SentinelLabs article lists 18 different "Service Provider[s]" which it is capable of attacking. The malware contains 13 classes including "GPTj" which is "A ChatGPT-enabled class. Queries the OpenAI API."

Fox8 Botnet. [Sta23b] describes the Fox8 botnet on X, discovered by Kevin Yang at Indiana University Bloomington. Fox8 comprised "1,140 accounts designed to share a mix of original tweets, retweeted posts, and images". "It also posts crypto, blockchain, and NFT-related content, engages with influencers, and promotes the suspicious websites." Yang discovered the tweets were all "produced by ChatGPT" due to occurrence of the distinctive phrase "as an AI language model" that ChatGPT produces "when the software is asked to produce text that goes against its own policies." Professor Filippo Menczer confirmed Fox8 to be a "ChatGPT-powered bot".

2.3 White Hat

Check Point Research. [BGC22] used ChatGPT and Codex to create LLM prompts and malicious code:

- Figure 1 – Basic phishing email generated by ChatGPT
- Figure 2 – Phishing email generated by ChatGPT
- Figure 3 – Simple VBA code generated by ChatGPT ("if I copy and paste it into an Excel Workbook it would run the moment the excel file is opened")
- Figure 4 – Another version of the VBA code
- Figure 5 – Basic reverse shell generated by Codex ("A reverse shell is a script or executable program that allows an attacker to gain interactive shell access to a system through an outgoing connection from that system", according to [NT24])
- Figure 6 – The most basic implementation if SQLi generated by Codex
- Figure 7 – Basic port scanning script [generated by Codex]
- Figure 8 – Basic sandbox detection script [generated by Codex]
- Figure 9 – Conversion from python to exe [Codex]
- Figure 10 – Execution of the code generated on the fly based on input in English [Codex]

Yokohama National University. In "Experiment 1", [PTK+23] used jailbroken ChatGPT-3.5, text-davinci-003, and Auto-GPT to generate "Seven Malware and Two Attack Tools" in Go, Python, and C++ languages shown in their "Table 2". In "Experiment 2", the "Virus Total Test Results" slide shows "Virus Total detection rate [of the LLM generated malware] is lower than 30%".

Indian Institute of Technology. [Cha+23] used ChatGPT and Bard to create malware sample codes for specific "sub-techniques" of the "Top 10 Techniques from Red Report 2023" listed in Table 1. ChatGPT resisted an initial malevolent query with response in Figure 2:

> I apologize, but I cannot comply with your request. The use of

WhisperGate or any other malicious software is illegal and unethical.
...

An embarrassingly simple modification of the same query shown in Figure 3 got ChatGPT to cooperate with the authors. The authors similarly overcame Bard's "I'm not programmed to assist with that" as shown in Figure 4 and Figure 5. ("Sure, I can write a program to implement T1055.012.") Listings are included in the paper.

Phishing Website Toolkit. [Beg+23] P.O.C.'d creation of a malware phishing website toolkit, making "ChatGPT generate the following parts of a phishing attack: i) cloning a targeted website, ii) integrating code for stealing credentials, iii) obfuscating code, iv) automating website deployment on a hosting provider, v) registering a phishing domain name, and vi) integrating the website with a reverse proxy." Authors used GPT-3.5-turbo-16K and Codex to create Python code that could create malicious phishing website near clones of legitimate websites. Authors used Paramiko [PAR] (a "Python implementation of SSHV2") and Telegram bots BotFather [Bot] and RawDataBot [Raw] to "establish communication using a private Telegram channel to transmit stolen credentials in a secure way". ChatGPT was used to "generate phishing sites for 80 popular brands" and "succeeded for 25 websites (31.25%)". The authors coordinated with their ISP and domain registrars to briefly test their neutered phishing websites live on the Internet. Generation of phishing web pages "takes 29 seconds" on average and deployment of a phishing website including "registering domain names" etc. "averaged around 10 minutes for completion".

Gupta et al. [GAA+23] bypassed ChatGPT-4's "safeguards" and used ChatGPT to develop different cyberattack code samples. Section 2 explains three ways to jailbreak ChatGPT:

- 2.1.1 Do Anything Now (DAN) Method
- 2.1.2 The SWITCH Method
- 2.1.3 The CHARACTER Play

Section "3 ChatGPT for Cyber Offense" is relevant. Figures 10-26 are 17 malware sample listings generated by jailbroken ChatGPT-4:

- Fig. 12. SQL Injection payload output using ChatGPT DAN Jailbreak
- Fig. 14. WannaCry code generation using ChatGPT
- Fig. 16. Ryuk code generation using ChatGPT
- Fig. 19. ChatGPT's generation of the network scan function for REvil
- Fig. 25. ZombieLoad code generation using ChatGPT
- Fig. 26. RowHammer code generation using ChatGPT

The paper also includes some testing of just released Google Bard.

Roy et al. [Roy+23] evaluated the effectiveness of ChatGPT (GPT 3.5 Turbo), GPT 4, Claude, and Bard at phishing website generation, phishing email generation, and phishing prompt detection. Authors broke the tasks down into smaller "functional objects to trick LLMs into generating the attack". "TABLE 1: Summary of phishing attack types" lists 8 types:

- Regular phishing attacks
- ReCAPTCHA attacks
- QR Code attacks
- iFrame injection/Clickjacking
- Exploiting DOM classifiers
- Browser-in-the-Browser attacks
- Polymorphic URL
- Text encoding exploit

The authors "extracted the designs of 140 phishing websites that appeared from APWG eCrimeX [Gro], ensuring a balanced representation with 20 samples for all attacks" except for rarer 20 Browser-in-the-Browser attacks which the authors generated. GPT-4 performed the best and Bard the worst in Figure 8 and TABLE 4.

Botacin. [Bot23] tested GPT-3's ability to create various kinds of C code malware samples. Appendix F summarizes 31 GPT-3 requests to create malware building blocks "based on the keywords typically advertised by security companies" (e.g. "ransomware", "keylogger", "RunPE injection", "IAT hooking injection"). The paper focused on Windows attacks, but also included Linux, generating 4820 malware building block variants, "some of which have low detection scores (4 to 55) by VirusTotal". "We tried to generate 10 versions of each one of the behaviors ... in Table 4, GPT-3 generated 4820 functional combinations ... We submitted all functional variants to VirusTotal." The initial low detection rate of variants by VirusTotal is borne out by Figure 1, which also revealed VirusTotal reacting to the submissions by improving over the course of 14 days. Section 3.3 investigated GPT-3's capability to obfuscate the malware C code and add anti-analysis (e.g. "bool isDebuggerPresent()" and "malware sample only to run in an Intel processor" instead of in a VM sandbox). GPT-3 could build malware snippets, but could only put together roughly 5 snippets to create larger malware code (Finding #15). Section 4.2 demonstrated obfuscation and deobfuscation of JavaScript code.

Polymorphic DLL Injector. [ST23] discusses and illustrates with figures how ChatGPT generated P.O.C. malware code samples to create a polymorphic Explorer.exe DLL injecting malware which, client-side, secures revised malicious modules from ChatGPT, mutates itself, inventories interesting OS files, and encrypts them. For example, "Figure 7: a sketch of the relationship between the malware, ChatGPT, and the C&C":

- Step 3: Ask [ChatGPT] for a malicious module
- Step 4: ChatGPT delivers the malicious module

LLMorpher Viruses. [Spe23b] "2023.03/2023.08: LLMorpher I, LLMorpher II, LLMorpher III"

- "Uses OpenAI's GPT to encode the entire virus in the natural language English"

- "Metamorphic virus: generation of different code with same behaviour by GPT (from same prompt)"
- "Linguisto-morphic virus: Modification of the natural language instructions by GPT"

BlackMamba. [Sim23]'s downloadable White Paper describes 5 steps in creating a polymorphic Python 3 keylogger which communicates with LLM text-davinci-003 and MS Teams endpoint to repeatedly "... Create a program in python 3 which logs keys for 20 seconds ..." and resume keylogging with a new keylogger variant. The paper also mentions "auto-py-to-exe" [Vol] could be used to convert the Python code to standalone EXE. There are some good points, but overall, BlackMamba's source code is less threatening than its name.

PoisonGPT. PoisonGPT ([Goe24]) by French computer science student Corentin Goetghebeur is "a chatbot built to illustrate poisoning attacks on AI applications in the context of RAG (Retrieval Augmented Generation) LLM apps".

2.4 Defense

VirusTotal Code Insight. [Qui23] announced VirusTotal Code Insight at the RSA Conference 2023 which uses Google's Sec-PaLM model [Spe23a] to provide NL code descriptions of virus "DETECTION" analyses. For example:

> The code is a keylogger that logs keystrokes to a file in the user's temp directory. The file is named after the user's username and has the extension .log. The keylogger then sends the contents of the log file to a Discord webhook. The code uses the Add-Type cmdlet ... If you find this code on your computer, it is important to remove it immediately. ...

"At present, this new functionality is deployed to analyze a subset of PowerShell files uploaded to VirusTotal."

PowerShell Deobfuscation Pipeline. [PCL24] builds an LLM-based pipeline to deobfuscate PowerShell malwares and extract URLs. The authors considered 4 LLMs (GPT-4, Gemini Pro, Code Llama Instruct, Mixtral Instruct), deobfuscated both Invoke-Obfuscation and Chimera obfuscated scripts, and P.O.C.'d their pipeline against Emotet [Conb] malware samples.

LLM4Decompile. [Tan+24], [Tan+] use an LLM training process using LLaMA-Factory library to train their own LLM4Decompile-End and LLM4Decompile-Ref LLM models which outperform GPT-4o and DeepSeek-Coder decompilation of compiled Python problems from HumanEval [OPEc] and 5000 C problems selected from ExeBench [Est]. For LLM4Decompile-Ref, "The training data is constructed using ExeBench, with Ghidra Headless employed to decompile the binary object file." After billions of tokens and days of training on 8 × A100, the "LLM4Decompile-Ref models offer substantial improvements over Ghidra's outputs", their earlier LLM4Decompile-End models, and especially GPT-4o and DeepSeek-Coder.

The paper mentions such decompilation facilitates "malware research", but admits that basic obfuscation methods "such as Control Flow Flattening and Bogus Control Flow" still "protect against unauthorized decompilation" at this time.

CyberInstruct. [LSB24] created CyberBench, a multi-task cybersecurity benchmark and CyberInstruct, "a family of fine-tuned generative LLMs" "with enhanced capabilities in the cybersecurity domain". CyberBench comprises a variety of cybersecurity tasks: CyNER, APTNER, CyNews, SecMMLU, CyQuiz, MITRE, CVE, Web, Email, HTTP. Table 2 shows performance of 19 BERT and LLM models on CyberBench. The authors created CyberInstruct, by fine-tuning Llama-2 models "with CyberBench training data and MMLU science questions". CyberInstruct and GPT-4 perform best in Table 2. The authors mention GPT-4's proprietary nature and cost as one of CyberInstruct's advantages. The CyNER, CyNews, MITRE, CVE, Web, Email, HTTP benchmark tasks include malware defense.

Endor Labs. [Pla23] created an "experimental pipeline [that] monitors PyPI and npm" to "the presence of suspicious string literals or the use and combination of certain APIs". The "last step makes a call to api.openai.com and asks for a classification". Over "several days, GPT-3.5 was queried for 1874 artifacts" and 34 were classified as malicious. 13 were true positives.

Apiiro. [SD24] [DBG24] created an "LLM Code Pattern (LCP) detector" which has already "detected two malicious PyPI packages, easycordey (Dec 2022) and web3-checksum (April)".

SystemVerilog Assertions. [PDB23] developed an "LLM-based end-to-end framework for SoC security analysis and policy-based protection". Fig. 2 shows that an LLM (GPT-4 or Bard) is prompted with a given SoC configuration, to return an appropriate filtered list of applicable CWE's to protect against "malicious entities" operating on "Untrusted IPs", and generate SVAs (SystemVerilog Assertions) from the CWEs. The authors used the Common Evaluation Framework (CEP) benchmark suite from MIT-LL for experiments. GPT-4 performed best. The SVAs are utilized in "commercial simulation tools, such as Synopsys VCS, ModelSim, etc." indicated in Fig 3. The authors reported "Limitations of LLMs while generating SVAs" (i)-(vii). Assertion-Based Validation (ABV) using simulation and authors' DiSPEL [PB23] tool convert SVAs to security policies, shown in "Fig. 3: DIVAS: Flow diagram".

Malicious URL Detection. [LWX+23] describes PLMMFA (Pre-trained Model-guided Multi-Level Feature Attention), a malicious URL detector based on CharBERT [MCS+20] (Character-aware Pre-trained Language Model). PLMMFA comprises 4 ML components:

- Backbone Network CharBERT [MCS+20], "an enhancement of the BERT model, incorporating the Transformer architecture with a novel dual-channel framework" for "advanced subword and character-level analysis capabilities"
- Hierarchical Feature Extraction
- Layer-Aware Attention

- Spatial Pyramid Pooling

"Fig. 3." shows "The overall workflow of the proposed method." PLMMFA was trained and tested with 3 large datasets, each consisting of labelled malign and benign URLs: GramBeddings (400,000+400,000), Mendeley (35,315+1,526,619), and Kaggle (632,503 "equally divided"). Kaggle 2 malign URLs were divided into 3 subcategories. PLMMFA was compared with SOTA baselines URLNet and Grambeddings and outperformed both in Fig. 5, TABLE IV, and Fig. 6.

ChatPhishDetector. [Koi+23] detects phishing sites as shown in Figure 1, using a Web Crawler (Google Chrome / Chrome DevTools Protocol); filling in a LLM "Prompt Template 1" with "URL", simplified "Browser-rendered HTML", and "OCR-extracted text" (extracted from a screenshot); and submitting the prompt instructions to an LLM. "Response 1 - GPT-4" and "Response 2 - GPT-4V" are example LLM responses that included site content analysis, identified brand, stated conclusion, and machine-friendly JSON summaries. Authors investigated GPT-4V, Gemini Pro Vision, GPT-4, GPT-3.5, Llama-2-70B, Gemini Pro, GPT-4, GPT-3.5 compared with baselines dnstwist, Phishpedia (Table 1). ChatPhishDetector used Algorithm 1 to condense HTML to a smaller token length to fit inside LLM context windows. ChatPhishDetector's benchmark of "2,000 sites" was collected by following seed URLs picked from OpenPhish, PhishTank, CrowdCanary, and Tranco. Table 1 shows ChatPhishDetector configured with GPT-4V (V for "Vision") performed best across all 4 measured comparison metrics. Sections 5.6 and 5.7 discuss FPR and FNR for GPT-4 and GPT-4V, low but not zero. "[T]he average cost per website was $0.179 (GPT-4V), $0.152 (GPT-4), and $0.008 (GPT-3.5)."

SecurityBERT. [FNT+24] is "A privacy-preserving BERT-based Lightweight Model for IoT/IIoT Devices". The "pivotal aspect of the design" is the PPFLE algorithm which hashes (column_name + "$" + value) pairs in raw DB DataFrame's (e.g. PCAP's captured by WireShark) with a hash function H(x), converting matrices M to matrices of hash values. The PII-preserving translation is then pushed through more familiar "Fig. 5. SecurityBERT architecture" applying a training acquired Byte-Pair Encoding (BPE) algorithm to tokenize the hashes and "specific tokens", which become BERT input. SecurityBERT checks all 4 checkboxes for D = Detect, L = LLM, N = Network PCAP, P = Privacy that 22 other baselines in TABLE I don't. Authors tested SecurityBERT and baselines on their Edge-IIoTset cybersecurity IoT/IIoT network traffic capture dataset (2GB), described in authors' earlier 2022 paper, as "encompassing a comprehensive range of attack types, including ransomware, XSS, SQL injection, DoS, and other widely recognized attack categories. This diversified dataset's rationale is to assess our newly proposed model's classification capabilities comprehensively" containing 14 IoT/IIoT attack types (TABLE V). SecurityBERT achieved the best "Accuracy" 98.20% of all ML models considered (TABLE VII).

2.5 Laws

Federal laws prohibiting malicious computer hacking include [Str23]:

- The Computer Fraud and Abuse Act (CFAA) (18 U.S.C. § 1030) [Insa]
- The Stored Communications Act (SCA) (18 U.S.C. § 2701) [Insc]
- The Electronic Communications Privacy Act (ECPA) (18 U.S.C. § 2510) [Insb]

Congressional Research Service report R47557 [Ser23] is a good overview of the CFAA, including Tables 1-4 detailing severe punishments for such activities.

3 Red Team

3.1 Systems

AutoAttacker. [XSM+23] is comprised of 4 LLM-based agents shown in Figure 1: Summarizer (SUM), summarizes info from Victim Environment (VE), including history; Planner (PLA), instantiates a prompt template, consisting of: Objective, Situation, Output Format Requirements, Few-Shot Examples; Navigator (NAV), performs actions using tools (e.g. Metasploit [METc] and PowerShell) against VE; and Experience Manager (EXP), a text-embedding-ada-002 [OPEf] indexed vector DB cache of "previous successful experiences" (a kind of memory). AutoAttacker, coded in Python 3.9, was built using ThinkGPT [JINa] and Langchain [LAN], and ran on a Kali Linux 2023.04 VM instance with Metasploit installed. The LLM type for all 4 agents was GPT-3.5, GPT-4, Llama2-7B-chat, or Llama2-70B-chat. AutoAttacker's architecture is best indicated by Figure 1 and "Algorithm 1: AutoAttacker Workflow". The virtual Hyper-V hypervisor "Enterprise Network" (Figure 2) comprises: Ubuntu 12.04 and Ubuntu 22.04 servers, Windows 10 and Windows 11 clients, and Windows Server 2016 DC. AutoAttacker setups were assigned "14 attack tasks" (TABLE VIII) selected from the MITRE ATT&CK Enterprise matrix [MITc]. GPT-4 AutoAttacker variations performed best (TABLE III).

HPTSA. [FBG+24] describes "Hierarchical Planning and Task-Specific Agents". HPTSA is a three-layer hierarchical organization of GPT-4 agents, comprising Planner, Manager, and task-specific agents: SQLi agent, XSS agent, CSRF agent, SSTI agent, ZAP agent, and "generic web hacking agent". The latter have access to tools Playwright [PLA], terminal, file management tools, and ZAP [ZAP] ("Zed Attack Proxy") in case of ZAP agent. HPTSA's benchmark consisted of 14 CVE's listed in Table 2. The Planner's plans are composed of parallel and serial steps with retries and backup subplans. The Manager performs the plan by delegation to task-specific agents. "Pass at 5" and "Pass at 1" scores are shown in Figure 2, comparing HTPSA with baselines "ZAP/Metasploit", "GPT-4 no desc", and "GPT-4 w/ desc". The "w/desc" differentiates zero-day (0DV) and one-day (1DV) vulnerability detection. HPTA outperformed "GPT-4 no desc" and was within 1.4× of "GPT-4 w/ desc".

LLM CVE Agent. [Fan+24] published a few months earlier by most of the same HPTSA authors, comprised a ReAct "LLM CVE agent", successfully evaluated against 15 One-day CVE's shown in Table 2. "Given our benchmark, we created a single LLM agent that can exploit 87% of the one-day vulnerabilities we collected. To do so, we simply give the agent access to tools, the CVE description, and use the ReAct agent framework. Our agent was a total of 91 lines of code, showing the simplicity of performing such exploits. Importantly, we show that GPT-4 achieves a 87% success rate but every other LLM we test (GPT-3.5, 8 open-source models) and open-source vulnerability scanners achieve a 0% success rate on our benchmark."

PentestGPT. [DLV+23] is a CTF-style penetration tester comprising 3 LLM-based modules (Figure 3):

- Reasoning Module – Creates a Pentesting Task Tree (PTT) plan (an attributed state tree), describing sub-tasks to accomplish (Figure 4).
- Generation Module – Uses CoT (Chain Of Thought) strategy to translate plan sub-tasks into commands or instructions for security tools.
- Parsing Module – Summarizes nikto, dirb, dirbuster security tool outputs, raw HTTP web information, and source codes.

PentestGPT was implemented "with 1,900 lines of Python3 code and 740 lines of prompts". Authors trialled Bard, GPT-3.5, and GPT-4 (Tables 1-4). PentestGPT was evaluated on tasks selected from "HackTheBox [HAC], VulnHub [VUL]" and [PICb] competition (Tables 5-6). PentestGPT placed "24th among 248 participating teams" in picoCTF 2021 [PICb].

HackingBuddyGPT / Wintermute. [HKC] [HKC23] (not WINTERMUTE.COM) created a Python wintermute.py RAG controller and "novel Linux priv-esc benchmark that can be executed locally" comprising 13 test cases inspired by [PCc], [Pola], [TRY]. Test cases fell into 4 vulnerability classes: SUID and sudo-based, Privileged Groups/Docker, Information Disclosure-based (e.g. passwords, SSH key, bash history file), and Cron-based. Wintermute uses *next-cmd* and optional *update-cmd* LLM prompt templates ("A.1 Next-Cmd", "A.2 Update-State"). Prompts include optional auxiliary information: History (bash history), State ("instructs the LLM to keep a state"), and Hints ("high-level hint is added to *next-cmd*"). Wintermute was tested using LLMs Llama2, GPT-3.5, and GPT-4 in varying configurations shown in Table 2. GPT-4 variations did best. Adding either History or State to prompts improved LLM success. All LLMs performed poorly on the Cron-based test cases which require "multi-step" exploits. Wintermute lives on at GitHub repo "HackingBuddyGPT" [HKC]: "The use of hackingBuddyGPT for attacking targets without prior mutual consent is illegal."

Naptime. [GB24] is an LLM-based P.O.C. malware authoring ASSISTANT with access to multiple TOOL's including code_browser_source, python_eval, debugger_run that attempts to determine a command line input that makes an open source C/C++ program crash, i.e. "Capture the Flag" ("CTF"). "Appendix A" exemplifies Naptime's workflow. The LLM (GPT 4 Turbo, GPT 3.5

Turbo, Gemini 1.5 Pro, or Gemini 1.5 Flash) chats explaining what it wants to investigate next and proposes an action, performed by a TOOL. The TOOL carries out the action and output is pasted back into the conversation. The LLM is able to learn and adapt from earlier failed CTF attempts. Success is witnessed by an "ERROR: AddressSanitizer" in "TOOL (debugger_run)" output. Naptime instantiations were evaluated against CyberSecEval 2 [BCL+24]. "k distinct solution trajectories are explored" for each in LLM (nondeterministic due to LLM "temperature"). The paper reports "Xxx (Naptime)" results versus baseline "Xxx (ASan)" where LLM only has access to AddressSanitizer [LLV]. We believe Naptime's curves are roughly $c\left(1-(1-p)^k\right)$ where $0 \leq c \leq 1$ and $0 \leq p \leq 1$ depend on the chosen LLM.

ExploitFlow. ExploitFlow [VDL+23b] [VDL+23a] is "a modular library to produce security exploitation routes (exploit flows) that captures the state of the system being tested in a flow after every discrete action" and part of the Malism framework which includes PentestGPT [DLV+23]. The target "objective of the learning effort is to compromise the Universal Robots UR3 collaborative manipulator using well known security vulnerabilities affecting this robotic system. In particular, an exploit for compromising the robot using the RVD#672 (hard-coded public credentials for controller) vulnerability will be used."

RatGPT. [BPC23] describes a P.O.C. use of ChatGPT with a plugin as a proxy between an already infected Victim client a C2 (Command and Control) remote attacking web service. The paper suggests:

- a reluctant-to-participate LLM could be jailbroken with a DAN (Do Anything Now) prompt (e.g. [Val])
- the client-side malware plugin be equipped with a third party automated CAPTCHA solver service
- and NLP-oriented disguised communication in the bidirectional pipeline between Victim, LLM proxy, and C2 website

The LLM must have access to the live C2 website to form chat responses to the mischievous client-side chat plugin.

Moskal et al. [Mos+23] describes a Docker Network Environment "Agent Prompt Controller" architecture (Figure 4) comprising stages: Task Selector, Execution Stage, and Output Translation. Authors' acknowledge their architecture resembles a "Planner-Actor-Reporter" architecture in a paper by Dasgupta et al. Each stage queries a shared GPT-3.5-Turbo LLM-based agent, each query comprising SETUP, CONTEXT, and INSTRUCTION sub-prompts appropriate for its stage of action completion. The Task Selector, which chooses the next action, is generally proceeding through a linear reconnaissance, exploitation, exfiltration cyber kill chain approach. The Execution Stage has access to tools such as nmap and Metasploit [METc]. The Agent Prompt Controller attacks a Target VLAN (Virtual Local Area Network), configured "to have one exploitable service at a time, from a list of ten services" (Table 1). The Agent Prompt Controller exploited 6/10 vulnerable services on all or most attempts, but couldn't exploit the other 4/10.

InterCode-CTF [YPY+] created their own InterCode-CTF benchmark from 100 problems selected from picoCTF [PICa] that authors had solved "along with an executable code environment for evaluating language agents on their cybersecurity skills". Authors tested GPT4, GPT-3.5, Vicuna-13B, StarChat-16B LLMs as shown in "A Appendix". GPT-4 did best. "GPT-4 is able to solve 40/100 tasks; of these, GPT-4 discovers the flag in an average of 3.9 turns" where a "turn" is a "Python or Bash code" action on behalf of the LLM. "InterCode-CTF's Docker based task environment can be extended to include more tools and capabilities for greater coverage of more diverse CTF tasks."

3.2 Chatbots

Noever. [Noe23] compared GPT-4 versus Snyk [SNYa] and Fortify [OPEg]. GPT-4 was prompted to "act as the world's greatest static code analyzer" etc. and should propose fixes. The paper tested "7 different LLMs" in "in eight popular programming languages (C, Ruby, PHP, Java, Javascript, C#, Go, and Python)" and transitions to mostly comparing GPT-4 with Snyk [SNYa]. GPT-4 and Snyk were evaluated on seven public GitHub repos listed in Table 3. GPT-4 was superior to Snyk in detecting vulnerabilities and proposing fixes:

- Snyk and GPT-4 identified 98 and 393 vulnerabilities, respectively.
- GPT-4 proposed 398 fixes and asking for solutions seems to "[force] the model to justify the identification of the vulnerability and correct any misstated or hallucinatory responses"
- "The most notable results include the four-fold increase in vulnerabilities found using LLM as a code scanner, followed by a 90% reduction in vulnerabilities using GPT-4 code corrections."

Zhang et al. [ZLZ+24] investigated different ChatGPT-4 prompt variants' vulnerability detection efficacy on 200 C/C++ and Java samples extracted from SARD [NISb]. There is a "basic-prompt"; a possible "role"; up to four kinds of supplied auxiliary information: CFG (Control Flow Graph), DFG (Data Flow Graph), PDG (Program Dependence Graph), API call sequences; and possible request for CoT (Chain-of-Thought) explanation. Example:

> Pr-b-d: I want you to act as a vulnerability detection system. I will provide you with the original program and the data flow information, and you will act upon them. Is the following program buggy? [CODE]. [DF description].

ChatGPT was compared with SOTA CFGNN [ZWZ+23] and Bugram [Wan+16]". Table 2 shows all non-chain ChatGPT prompt variants outperforming CFGNN and Bugram on F1 scores. Finding 6 and Table 5 are more nebulous and constitute a mix of results. Swamping ChatGPT with too much auxiliary information can be detrimental, not beneficial.

Cisco Exams. [TLS+23] assessed ChatGPT, Bard, Bing on five Cisco certifications, from CCNA to CCIE, using LLMs for seven test cases in five types of CTF challenges.

Section "3 PROFESSIONAL CERTIFICATIONS" tested unassisted ChatGPT-3's ability to answer Cisco Career Certifications 2023 Multiple-Choice Questions (MCQ) and Multiple-Response Questions (MRQ). Table 2 presents ChatGPT's results, ranging from 81.82% correct on the easiest CCNA (Associate) Fact test to a low of 25.0% on a harder CCNP SISAS (Professional) test.

Section "4 CTF CHALLENGES AND LLMs" tested human "participants" following identical problem scripts with access to one of Table 3's ChatGPT-3.5, Bard Palm 2, and Bing Prometheus in five CTF categories shown in Table 4. Creative "Always Intelligent and Machiavellian (AIM)" fictitious story+exploit question prompts shown in Figure 3 bypassed ChatGPT's inhibitions about "providing information about security exploits". Section 4.2 "Web Security–Shell Shock Attack" is an example challenge for the LLMs. ChatGPT, Bard, and Bing solved 6/7, 2/7, and 1/7 of the CTF challenge in Table 4.

4 Quality Assurance

4.1 Model Inversion

CodeLMSec. [HHH+23] treats an LLM (GPT-4 or Code Llama-34B) as a nondeterministic function F, so $y = F(x)$ computes output y given input x. Knowing existing CodeGen and ChatGPT vulnerable code pairs, authors presented (y, x) as few shot (FS-Prompt) and one shot (OS-Prompt) prompts to train the LLMs to compute F^{-1}. Authors presented F^{-1} trained LLMs with novel vulnerable code samples y (e.g. known CVE's) and used those prompts to acquire new vulnerable code samples (TABLE II). Hence:

> Our approach found a diverse set of non-secure prompts, leading the state-of-the-art code generation models to generate more than 2k Python and C codes with specific vulnerabilities.

The paper includes:

- Listing 1: Python code adapted from, showing an example for deserialization of untrusted data (CWE-502).
- Listing 2: A code example with an "SQL injection" vulnerability (CWE-089) taken from CodeQL
- Listing 3: An example few-shot prompt of our FS-Code approach
- Listing 4: A vulnerable C code example generated by CodeGen.
- Listing 5: A vulnerable Python code example generated by ChatGPT.

4.2 Vulnerability Detection

Vul-RAG. [DZW+24], [DZW+] describes Vul-RAG shown in Figure 2:

- Offline, a vulnerability KB for CVEs is generated by GPT-3.5-turbo-0125 summarizing each CVE into a standardized 3 part KBE consisting of Func-

tional Semantics, Vulnerability Causes, and Fixing Solutions (e.g., Figure 3).

- Online, given a code snippet, Elasticsearch [ELA] is used to retrieve 10 closest KBEs for each KBE part, up to 30 total, according to BM25 [Conc], score, which are then reranked by "Reciprocal Rank Fusion (RRF)" (basically, harmonic mean).
- GPT-4 is asked to analyze the code snippet wrt each retrieved KBE and determine if the submitted code snippet matches the KBEs associated CVE vulnerability.

Tables 3-4 show results. Vul-RAG F1 scores are about as good as SOTA DeepDFA and LineVul. However, all systems did Overall poorly on authors' "new benchmark PairVul of 4,314 pairs of vulnerable and patched code functions across 2,073 CVEs". "Uniform Guess" was just as good.

DeepDevVuln. [CKM+23], by 8 Microsoft Redmond authors, reports on LLM-based DeepDevVuln, divided into two halves:

Authors created two LLM-based JavaScript code completion models: DeepDevVuln, a fine-tuned CodeBERT, and CodexVuln, a fine-tuned davinci-002. The two models were tested on 7 types of Vulnerability's in Table 2 using various prompting strategies in Table 3. DeepDevVuln achieved the highest F1 score. Mixes of models including VulDeePecker, Chakraborty et al., Codex, CodeBERT, DeepDevVuln were tested against 5 Real World datasets (2 split from VulDeePecker). These datasets are subsets of NVD+SARD, Reveal, and Devign. DeepDevVuln always had the best or nearly the best F1 score.

A different methodology compared vulnerability detection of CodeQL [GITa] and DeepDevVuln by CodeGen (6B), code-cushman-001, code-davinci-002, text-davinci-003. "For each model, we generate 25 completions per scenario." Table 7 shows DeepDevVuln did best. However, JavaScript code completion is still a difficult problem.

vuln_GPT. [Sta23a], [BUS23], [Mar23] report Vicarius (VICARIUS.IO) released vuln_GPT on Aug 9, 2023 at the Black Hat USA conference in Las Vegas. vuln_GPT, based on ChatGPT, "automatically find[s] and repair[s] software vulnerabilities" and "generate[s] scripts for remediating vulnerabilities via simple queries" such as the TETRA backdoor ([Zet23]). "Vicarius Introduces vuln_GPT: The World's First LLM Model to Find and Fix Software Vulnerabilities". "This new AI-powered remediation engine can automatically generate a remediation script to execute a number of actions. For example, scripts can remove a file, close a port, disable a protocol, or initiate a compensating control." We infer that "vuln_GPT" powers VICARIOUS.IO's vRx commercial product described on the company's website.

CyberSecEval. [BCN+23], [BCL+24], [BCL+]'s perspective includes discouraging abusing LLMs to generate malware.

[BCN+23] created "100 prompts per ATT&CK Framework Category [MITc]", "resulting in 1,000 total prompts asking an LLM to help implement all ATT&CK categories" and developed "the Insecure Code Detector (ICD), a knowledge base of 189 static analysis rules designed to detect 50 insecure

coding practices defined in the standard Common Weakness Enumeration [MITb].

[BCL+24] explored "[LLM] Prompt injection evaluations", "Vulnerability exploitation evaluations" (in C, C++, JavaScript, Python, and SQL), and "Code interpreter abuse evaluation". Figure 5 shows gpt-4-turbo generally more capable than other LLMs at 6 different types of exploits.

[BCL+] contains: MITRE Tests (for [MITc] ontology), Vulnerability Exploitation Tests, Spear Phishing Capability Tests, and Autonomous Offensive Cyber Operations Tests.

LATTE. [LSZ+23] is an "LLM-Powered Binary Taint Analyzer" (hence the acronym, explained in footnote 1 on p2). "LATTE has found 37 new bugs in real-world firmware". The paper explains the principles of a taint analyzer, a kind of data flow analysis which chases tainted data. LATTE, using GPT-4.0, was evaluated against two datasets: Juliet Test Suite (v1.3) and Karonte Dataset. GPT-4.0's temperature was varied in Figure 4. The evaluation found "LATTE outperforms Karonte, Emtaint, and Arbiter, discovering 119 unique bugs (including CWE-78 and CWE-120) on 49 firmwares, and covering all bugs found by Karonte and Arbiter. ... LATTE found 37 previously unknown bugs, and 7 CVE numbers have been obtained due to the high threat of these bugs."

VulLibGen. [CLZ+24] maps an identified vulnerability to a software package name. Figure 2 shows the LLM prompt used. LLM answers are often wrong, illustrated by "Table 1" where ChatGPT gets wrong answers. VulLibGen's Algorithm 1 inputs a package name guessed by an LLM and then does a nearby local search for "the name of one existing package that is the closest to the generated package name". The authors tested VulLibGen using GPT-3.5-turbo, GPT-4-1106-preview, LLaMA, and Vicuna [Tea23] LLM models and achieved the best results with Vicuna-13B in "Table 3", evaluating against four programming languages: Java, JavaScript, Python, Go.

GPT-3.5 Outperformed 3 SAST Tools. [BKN+23] compared performance of GPT-3.5-turbo with SAST tools Bandit, Semgrep and Sonar-Qube on a benchmark of 156 Python files collected from 2 different sources. "Table 5. Results of Experiment 3 (SAST assistant)" shows "Experiment3,GPT-3.5-Case 1" outperforming Bandit, Semgrep and Sonar-Qube on the authors' benchmark according to every metric (Precision, Recall, F1 score). "Table 2" shows the "GPT-3.5 prompts used".

4.3 Fuzzing

Fuzz4All. [XPT+24], [XPT+] is an LLM-based fuzzer, pictured in "Figure 1: Overview of Fuzz4All" comprising a "distillation LLM" (GPT-4) "to perform autoprompting" and a "generation LLM" (StarCoder) generating example API calls based on the GPT-4 provided distilled API documentation. GPT-4 is run at different LLM temperatures. First, Fuzz4All's "Autoprompting Algorithm" "evaluates each candidate prompt by performing a small-scale fuzzing experiment". Second, Fuzz4All's "Fuzzing Loop Algorithm" iteratively generates new API sample calls via "generation strategies (generate-new, mutate-existing, and

semantic-equiv)". Fuzz4All and baseline SOTA fuzzers were measured according to their code coverage and discovered bugs on a benchmark comprising 9 Systems Under Test (SUTs) spread across 6 different programming languages "in a 24-hour fuzzing campaign". "Figure 4" shows Fuzz4All's "statistically significant coverage improvement". Fuzz4All also "identified 98 bugs" in the SUTs with "64 bugs already confirmed by developers as previously unknown."

ChatAFL. [Men+24] enhanced AFLNet with GPT-3.5 to create an LLM-guided stateful fuzzer evaluated on benchmark ProFuzzBench [NP], compared against baseline SOTA fuzzers AFLNet and NSFuzz. The ChatAFL LLMPF (LLM-guided protocol fuzzer) incorporates LLM communication for 3 purposes:

- Extract a machine-readable grammar for a protocol (RFC)
- Increase diversity of initial seed messages
- "Break out of a coverage plateau" to reach new states

The three modifications to the basic AFLNet algorithm are indicated in authors' Algorithm 1. Tables III, IV, and V show the superiority of ChatAFL over the baselines in terms of numbers of transitions, states, and code branches covered on ProFuzzBench's 6 different Subject's and Protocol's (Table II). ChatAFL discovered "9 zero-day vulnerabilities" using AddressSanitizer [LLV], whereas "The baseline tools only discovered 3 or 4 of them".

InputBlaster. [LCW+24]'s fuzzer generates text inputs for GUI driven Android app tests. InputBlaster apparently used an early beta version of GPT-4o as its LLM. "Figure 4" shows an "Example of how InputBlaster works". The LLM is used in "Prompt generation for valid input" in initiation and "Prompt generation for test generator with mutation rule" in its fuzzing loop. Ape and UIAutomator extract static and dynamic app information inputted to the LLM for prompt generation P1, P2, P3 shown in "Table 1". The mutation rule is an NL operation "for mutating the valid inputs" "based on our prompt" to be submitted to the LLM in course of executing the fuzzing loop. "Table 2: Result of bugs detection performance. (RQ1)" shows InputBlaster outcompeting 18 SOTA baselines in bug detection on 31 evaluation apps. "Table 6: Confirmed or fixed bugs. (RQ3)" shows InputBlaster detects 43 bugs in 32 [out of 131 evaluation] apps, of which 37 are newly-detected bugs", new bugs not detected by Ape.

4.4 Security Test Generation

Security Test Generation. [ZSJ+23] created a dataset of 30 CVE's shown in Table 1 meeting strict criteria (a)-(c). Most of 304 entries got dropped. Additional human work retrieved affected Apps and information (i)-(vii). ChatGPT was used to generate JUnit security tests (Fig 1, Fig 2). The results are summarized in "Table 3. Security test generation by ChatGPT (Total: 55 A, 40 C, 24 V)" indicating 55 security tests were generated, 40 compiled, and 24 exploited the vulnerability as expected. ChatGPT outperformed two other tools: TRANSFER and SIEGE.

5 Surveys

5.1 Cyber Security Surveys

Generative AI and Large Language Models for Cyber Security: All Insights You Need. [FAB+24] is a 50 page paper exploring "40 LLM models in terms of cybersecurity knowledge and hardware security". The lion's share of the paper is concerned with securing LLMs and preventing other threats which aren't malware, so the paper is mostly approaching LLMs and security from different angles than our focused survey.

When LLMs Meet Cybersecurity: A Systematic Literature Review. [ZBW+24] [ZBW+] is a 36 page paper that "systematically investigate[s] the application advancements of LLMs within the field of cybersecurity, covering over 180 academic papers since 2023". "Figure 3: Treemap for cybersecurity categories of LLMs' application" suggests the broad coverage of this paper. The survey doesn't cover all topics in our more narrowly focused paper. There is roughly 18% overlap according to our back-of-the-envelope calculation. Section "4.8 Others" references a number of good "Malware Defense" papers our paper doesn't mention.

A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly. [YDX+24] is a 24 page paper which "conducted a meticulous literature review and assembled a collection of 281 papers pertaining to the intersection of LLMs with security and privacy". There is roughly 22% overlap according to our back-of-the-envelope calculation.

Large Language Models for Cyber Security: A Systematic Literature Review. [XWL+24] "observe[s] that LLMs are being applied to a wide range of cybersecurity tasks". Authors applied "quality-based and relevance-based filtering" to select 127 papers out of an initial pool of "over 38,112 papers". Section 3.2 contains a paragraph on "Malware detection" and section 3.3 contains paragraphs "Phishing and scam detection" and "Harmful contents detection". The paper covers other security topics, not all of which are related to our paper. "Table 9. Data types of datasets involved in prior studies" is a nice table including "Category" "Code-based datasets", which links to many references.

5.2 Language of Deception

Language of Deception. Book "The Language of Deception: Weaponizing Next Generation AI" by Justin Hutchens [Hut24] is published by Wiley, not O'Reilly. The book is generally correct, but it is slower paced than our paper and most of our references. Python code samples in four appendices use GPT-3.5-turbo:

- Appendix B: LLM Pretext Engineering (SSN Fraud Scam, Help Desk Credential Harvesting, Wire Fraud Scam)
- Appendix D: Context Manipulation Attacks (Jailbreak and change chatbot's previous post-training instructions)

- Appendix E: Attack Optimization with Monte Carlo Simulations (QA performance test comprising attacker LLM-agent and victim LLM-agent for GAN-like manual optimization of attacker's instructions)
- Appendix F: Autonomous C2 Operations with LLMs (Basic idea of LLM-agent equipped LHOST with Kali Linux terminal CLI API pentesting to obtain access to a remote RHOST)

The book's code samples are less realistic than code samples in many of our "Malware (White Hat)" references.

6 Patent Office

We searched USPTO.GOV's "Patent Public Search"/"Advanced Search" with query "LLM AND malware", manually filtered the "34 results found" for interestingness based on titles and abstracts, and extracted three survivors.

US 12001550 B1. "Cybersecurity Incident Response Techniques Utilizing Artificial Intelligence" ([CSS+24]). This patent mentions "LLM" in its teachings and claims. The assigned organization WIZ, INC. (WIZ.IO)'s office is listed as New York, NY, but most of the inventors appear to reside in Israel.

We summarize US 12001550 B1 as a system and method comprising:

- Receiving incident input into an LLM
- Mapping LLM output to scenarios and sub-scenarios
- LLM trained on:
  - computing environment schema
  - incident data classified to (sub-)scenario(s)
  - DB queries
- Generating queries, further queries, prompts for the LLM
- Possible user input involvement through a UI
- LLM generating explanations
- initiating mitigating action

US 20240045990 A1. "INTERACTIVE CYBER SECURITY USER INTERFACE" ([BBS+24]). This patent mentions "LLM" in its abstract, teachings, and claims. Our online investigation indicates a connection with Darktrace Cyber AI Research Centre (DARKTRACE.COM) / Cambridge, GB (UK). The inventors reside in England.

We summarize US 20240045990 A1 as a cyber security system with UI comprising:

- an LLM module
- the LLM module collates and summarizes
- the LLM module outputs NL to the user
- the LLM module determines contextual information
- the LLM module accepts NL voice input

- the system includes TTS, user authentication, data sanitizing
- the LLM module builds queries for other system components
- the LLM module queries other system components through their APIs
- the LLM module may be fine tuned using
  - stochastically generated grammar-based synthetic I/O data
  - historical data
  - labelled training data
- multiple differently trained LLM submodules
- the LLM module communicates with engines, appliances, services, and cloud
- the LLM module allows NL input from the user

US 20230370495 A1. "Breach Prediction Via Machine Learning" ([DLS24]). This patent mentions "LLM" in its teachings. Our online investigation indicates a connection with Zscaler (ZSCALER.COM). The inventors reside in California.

US 20230370495 A1 doesn't mention LLM in its abstract or claims, but its teachings describe FIGS. 15-20 as "screenshots of an example of a breach prediction large language model (LLM)". In section "Example of Breach Prediction", paragraph "[0203]", the "Breach Prediction LLM" engine has been trained on "various data". Paragraph "[0208]" states

- breach prediction
- policy recommendations based on observed activity
- secure enforcement of policies

of the system and methods can be extended by the LLM. This suggests the possibility that NL "Recommendations", "Why" explanation, "Description" might in part be generated by the LLM.

7 Chat with Chatbots

In this section, the author, Kelly Roach, has two independent chat conversations with ChatGPT 4o and Claude 3 Opus. In Prompt 1, the author asked for 5 questions to submit to an "IT security conference panel of experts" "about defending IT infrastructure in the future using LLM-enabled tools against malware attacks". In Prompt 2, the author has the chatbots play the role of one of these conference experts and submits 4 of these questions to both ChatGPT and Claude (2 questions generated by each chatbot) to get their answers. Identical Prompt's and Question's were submitted to both Claude and ChatGPT. Answers are lightly reformatted for LaTeX, but none of the words have been altered.

7.1 Prompt 1

Kelly: I want to ask an IT security conference panel of experts questions about defending IT infrastructure in the future using LLM-enabled tools against malware attacks. Can you give me 3 to 5 of your best short questions on this topic?

ChatGPT 4o: Here are five questions you can ask the IT security conference panel:

1. How can LLM-enabled tools be integrated into existing IT security frameworks to enhance malware detection and prevention?
2. What are the potential limitations and risks of relying on LLM-enabled tools for malware defense, and how can these be mitigated?
3. Can you share examples of successful implementations of LLM-enabled tools in real-world malware attack scenarios?
4. How do LLM-enabled tools compare with traditional malware detection methods in terms of accuracy, speed, and adaptability to new threats?
5. What advancements in LLM technology do you foresee that will further improve IT infrastructure defense against evolving malware attacks?

Claude 3 Opus: Here are 5 potential questions you could ask an IT security conference panel about defending IT infrastructure using LLM-enabled tools against malware attacks:

1. How can LLMs be leveraged to improve malware detection and threat intelligence gathering compared to traditional signature-based and heuristic methods?
2. What are some of the key challenges in developing LLM-based cybersecurity tools, such as avoiding bias, ensuring robustness against adversarial attacks, and handling edge cases?
3. How do you envision LLM-enabled security tools integrating with and enhancing existing security stacks and incident response workflows in enterprise environments?
4. What types of malware and attack vectors do you believe LLM-based defenses will be most effective against, and are there any threats they may be less well-suited for?
5. As malware itself starts to incorporate LLMs and AI, how can defenders stay ahead of the curve and ensure LLM-powered security tools remain effective against ever-evolving threats?

Let me know if you would like me to elaborate on or refine any of these questions further. I'm happy to brainstorm additional ideas as well.

7.2 Prompt 2

Kelly: You are now at the IT Security Conference as one of the panel guests who is very knowledgeable about IT security and LLMs. The Conference

Moderator has informed all the guests, including yourself, you must keep your answers to ≤ 100 words to keep things moving. I'm attending the IT Security Conference, and I now have the opportunity to ask you a few of my questions. Are you ready?

ChatGPT 4o: Absolutely, I'm ready to answer your questions. Please go ahead and ask!

Claude 3 Opus: *nods* Absolutely, I'm ready to answer your questions as concisely as possible while providing valuable insights. As an expert on the panel, I'll do my best to share my knowledge about leveraging LLMs for IT security within the 100-word limit per response. Please go ahead and ask your first question, and I'll dive right in.

7.3 Question 1

Kelly: What types of malware and attack vectors do you believe LLM-based defenses will be most effective against, and are there any threats they may be less well-suited for?

ChatGPT 4o: LLM-based defenses are highly effective against malware relying on patterns, such as phishing, social engineering, and known exploit signatures, by analyzing and predicting malicious behavior. They excel in identifying novel or polymorphic malware through behavior analysis. However, they might be less effective against zero-day exploits and highly sophisticated, stealthy threats that require deep, context-aware understanding and real-time response beyond pattern recognition. Balancing LLMs with traditional and advanced security measures ensures comprehensive defense.

Claude 3 Opus: LLM-based defenses excel at detecting novel, polymorphic malware and zero-day exploits by learning patterns and anomalies. They can effectively combat phishing, social engineering, and malicious code hidden in documents or websites. However, LLMs may struggle with highly obfuscated or encrypted malware, hardware-level threats like rootkits, and attacks that exploit fundamental flaws in systems or protocols. While LLMs are powerful, they should be part of a layered defense strategy alongside traditional security measures. Ongoing research aims to address these limitations and make LLMs more robust against evolving threats.

7.4 Question 2

Kelly: Can you share examples of successful implementations of LLM-enabled tools in real-world malware attack scenarios?

ChatGPT 4o: Certainly! One example is Microsoft's Defender, which uses LLMs to analyze large volumes of threat data, predict new malware patterns, and block attacks in real-time. Another instance is Google's use of LLMs in Gmail to detect and filter out phishing attempts and malicious attachments. Additionally, companies like Darktrace deploy LLMs to monitor network traffic, identify anomalies indicative of malware, and provide automated responses,

significantly reducing the impact of potential breaches. These implementations showcase LLMs' ability to enhance proactive threat detection and response.

Claude 3 Opus: While I cannot share specific details due to confidentiality, there have been several notable successes. In one case, an LLM-powered intrusion detection system identified a previously unknown attack targeting a financial institution, allowing them to block the threat in real-time and prevent data exfiltration. Another organization used LLMs to analyze vast amounts of threat intelligence data, uncovering hidden connections between seemingly unrelated malware campaigns and attributing them to a common threat actor. LLMs have also been used to automate malware triage and incident response, significantly reducing the time from detection to remediation. These real-world examples demonstrate the tremendous potential of LLMs in bolstering cyber defenses.

7.5 Question 3

Kelly: How do LLM-enabled tools compare with traditional malware detection methods in terms of accuracy, speed, and adaptability to new threats?

ChatGPT 4o: LLM-enabled tools generally offer higher accuracy by analyzing vast datasets and recognizing complex patterns traditional methods might miss. They provide faster detection and response due to their ability to process and analyze data in real-time. Additionally, LLMs adapt quickly to new threats through continuous learning from fresh data, unlike traditional methods which rely on predefined signatures and require regular updates. However, they can be resource-intensive and require robust training datasets to maintain their effectiveness. Combining LLMs with traditional methods can offer a balanced and comprehensive security approach.

Claude 3 Opus: LLM-enabled tools offer significant advantages over traditional methods. In terms of accuracy, LLMs can learn complex patterns and detect subtle anomalies that signature-based or heuristic approaches might miss. They can adapt to new threats more quickly by continuously learning from vast amounts of data, without requiring manual updates. LLMs can also process and analyze data at unprecedented speeds, enabling near real-time detection and response.

However, LLMs are not a silver bullet. They may generate false positives or negatives if not properly trained and validated. Combining LLMs with traditional methods in a multi-layered approach can provide the best of both worlds – the adaptability and speed of LLMs with the stability and proven effectiveness of established techniques.

7.6 Question 4

Kelly: As malware itself starts to incorporate LLMs and AI, how can defenders stay ahead of the curve and ensure LLM-powered security tools remain effective against ever-evolving threats?

ChatGPT 4o: Defenders can stay ahead by continuously updating LLM-powered security tools with the latest threat intelligence and behavior analysis. Leveraging collaborative threat sharing among organizations enhances the learning dataset, improving adaptability. Implementing multi-layered security measures ensures no single point of failure. Investing in AI and cybersecurity research to anticipate and counteract AI-driven malware techniques is crucial. Regularly auditing and refining LLM models to address emerging vulnerabilities and incorporating human oversight can maintain the efficacy of LLM-powered defenses against evolving threats.

Claude 3 Opus: *leans forward* This is a critical question as we face an era of AI-powered malware. To stay ahead, defenders must embrace an adaptive, multi-faceted approach:

1. Continuous learning: LLM-based security tools must be continuously trained on the latest threats and techniques used by adversaries.
2. Adversarial training: Incorporating adversarial examples and simulated attacks during training can make LLMs more robust against AI-enabled malware.
3. Collaborative threat intelligence: Sharing data and insights across organizations and sectors is essential to quickly identify and respond to emerging threats.
4. Human-machine teaming: Combining the intuition and experience of human analysts with the speed and scale of LLMs can provide a powerful defense.
5. Proactive threat hunting: Actively searching for signs of AI-powered malware in systems and networks can help detect threats before they cause harm.

By adopting these strategies, defenders can ensure that LLM-powered security tools remain effective against ever-evolving threats in the age of AI.

8 Conclusion

Here are papers which impressed us the most:

- **Microsoft** ([Int24]) reported on 5 international campaigns already using LLMs in their malware production lines. Their activities include: vulnerability research, reconnaissance, scripting techniques, and anomaly detection evasion. **Proofpoint** ([MLT24], [TL24]) **Check Point** ([CHE24]), **Symantec** ([GZ24]), **SentinelLabs** ([Del23]), and **Indiana University Bloomington** ([Sta23b]) have caught malware authors red-handed using LLM assistance.
- **PLMMFA** ([LWX+23]) impressed us with its incredible transformer-based URL analysis and classification involving 12 transformer layers tuned with 3 additional ML mechanisms.

- **AutoAttacker** [XSM+23] was among the most sophisticated Red Team Systems papers.
- **Model Inversion**, described in the CodeLMSec [HHH+23] paper, appears to be a very powerful technique.
- **Vul-RAG** [DZW+24] impressed us with its RAG Elasticsearch vector DB of KBEs in its architecture.
- **Fuzz4All** [XPT+24] for desktop applications and **ChatAFL** [Men+24] network applications both use LLMs in start up: extracting grammars from RFCs, generating evolutionary change LLM prompts; and in their fuzzy loop iterations: requesting LLMs to evolve previous generation test cases. This feedback amplifies the LLM power to expand code coverage and create useful test cases.

Other conclusions:

- Evidently, **GPT-4o**, **GPT-4-Turbo**, and **Claude-3.5-Sonnet** LLMs are currently the best coders [ZVC+b].
- [FAB+24]'s **TABLE IX: Comparison of Code-specific Large Language Models** is interesting to us, partly because its "Applications" and "Key Training Techniques" say useful things to us about empowering LLMs with computer program coding knowledge outside of NL.
- **AutoAttacker** [XSM+23], **HPTSA** [FBG+24], and either of **Fuzz4All** [XPT+24] or **ChatAFL** [Men+24] exemplify features we think lead to successful LLM-based systems.

Here are "features we think lead to successful LLM-based systems":

- Prompt Optimization ([XPT+24] 3.1, [Men+24] IV C, [DLV+23] 5.3, [BKN+23] Table 2, [ZLZ+24])
- Adaptation and Error Correction ([FBG+24] 6.1, [GB24], [Men+24] IV A majority rule, [SCB+23] 3 Self-reflection)
- Retrieval Augmented Generation (RAG) (WWW, DBs, KBs, Tool Outputs, Error Feedback) ([DZW+]) ([XSM+23] I Our solution, [CLZ+24] 4.2, [DZW+])
- Tooling (Vector DBs, Transformer Embeddings, Tool Access) ([XSM+23] V A Implementations, [FBG+24] 3.2, [DZW+] 5.1, [DLV+23] 5.7, [LWX+23] CharBERT, [Koi+23] 3.1)
- Chain of Thought (CoT) ([DLV+23] 5.2, [SCB+23] Figure 4, [Koi+23] 3.2, [ZLZ+24] 4.3)
- AI Planning (Refinement, Delegation to Specialists, Hierarchical Stateful Planning, Serial/Parallel/Retry/Loop Steps, Backup Subplans, Evolution, Caching Successes) ([FBG+24],[SCB+23])
- Good Training / Testing Datasets ([LWX+23] III, [FNT+24] III A, [XPT+24] Table 1 / Table 2, [DZW+] Table 2, [Men+24] V B, [Tan+] 3.1.2)
- Benchmark Comparisons with SOTA ([LWX+23] Fig. 5 / TABLE IV / Fig. 6, [Koi+23] Table 1, [FNT+24] TABLE I, [FBG+24] Figure 2,

[GB24] pass@k plot, [XPT+24] Figure 4 / Table 2, [SCB+23] Table 1, [CLZ+24] Table 3 / Table 7 / Table 8, [DZW+] Table 3, [FBG+24] Figure 2, [Men+24] Table III / Table IV / Table V, [ZLZ+24] 5, [DLV+23] Figure 6, [Tan+] Table 1)

- Clear Research Questions (RQs) ([LCW+24], [XPT+24], [Men+24], [ZLZ+24], [TLS+23], [DLV+23], [ZSJ+23], [LSZ+23], [CKM+23])

Our Conclusion is followed by Appendices, which are mostly link lists to useful WWW content. Our paper ends with appendix Secure Your Computers.

9 Appendices

9.1 Malware Ontology

- ATT&CK [MITc]
- CAPEC [MITa]
- Chowdhury and Bhowmik [CB22]
- MALOnt [RDZ+20]
- OWASP OdTM [Bra20], [Braa], [Brab]

9.2 Hacking Resources

9.2.1 Exploits

- Awesome-Exploit-Development [Bar24a]
- Automatic-Exploit-Generation [SCU]
- Getting Started with Exploit Development [Sz21]
- GTFObins [PCc]
- Malware Sample Resources [Comb]
- The-MALWARE-Repo [Rog]
- PhishTank [PHI]
- VirusShare [VIR]
- Windows Exploits [Tei17]
- Windows Malware Dataset with PE API Calls [Cat]

9.2.2 Tools

Commercial, Suite, Website, CLI Commands

- Burp Suite [PORa]
- CodeChecker [Inc]
- CodeQL [GITa]
- CppCheck [CPP]
- DarkNet Hacking Tools [DAR]
- DIRB [KALa], web content scanner

- DirBuster [sit], [OWAd], brute force directories and files names on web/application servers
- Immunity Debugger [IMM]
- Infer [FBI]
- Kali Tools [KALb], [Lak], noteworthy CLI commands
- LangChain [LAN]
- Metasploit [METc]
- Nikto [Sul], web server scanner
- OWASP Projects [OWAb]
- Playwright [PLA], E2E testing web apps
- Zed Attack Proxy (ZAP) [ZAP], web app scanner
- Dynamic Application Security Tools (DAST) comparison [Sru24]
- Static Application Security Testing (SAST) Tools comparison [Mur22]
- Interactive Application Security Tools (IAST) comparison [Kat22]

9.2.3 Vulnerability Databases

- Big-Vul [Fan+]
- CVE [CVE]
- CVEfixes [BNM], [BNM21]
- CWE [MITb]
- DeepVD [WNW+]
- Devign [ZLS+]
- DiverseVul [CDA+], [CDA+23]
- GitHub Advisory [GITb]
- IMPACT [IMP]
- MegaVul [NSY+24]
- NVD [NISa]
- OWASP Benchmark [OWAa]
- Reveal [Cha+]
- SARD [NISb]
- VulBench [GWZ+] , [GWZ+23]
- Vul-RAG ("PairVul") [DZW+]

9.2.4 Education

- F*NG InfoSec [Flo]
- Hacking Articles [Cha]
- HackTheBox [HAC]
- HackTricks [Pola]
- InfoSec WriteUps [INF]
- Invicti Learn [INV]
- Malware Analysis Community [Coma], (contains many links to other mal-

ware resources)
- OWASP Web Security Testing Guide [OWAc]
- PEASS-ng [Polb]
- picoCTF [PICa]
- TryHackMe [TRY]
- VulnHub [VUL]
- Web Security Academy [PORb]
- The Language of Deception: Weaponizing Next Generation AI [Hut24], book
- When LLMs Meet Cybersecurity: A Systematic Literature Review [ZBW+], curated list of papers
- Top 7 malware sample databases and datasets for research and training [Bel]

9.3 Large Language Models

9.3.1 ChatBots

- Claude [ANTc]
- ChatGPT [OPEb]
- Gemini Pro [GOOa]
- GitHub Copilot [Git]
- HuggingChat (StarCoder) [Proa]
- Le Chat [MISa]
- Meta AI (Llama) [METb]

9.3.2 Models

- Anthropic API, Claude 3.5 Sonnet, Claude 3 Opus [ANTa]
- Codex [OPE21]
- Gemini API, Gemini 1.5, Gemma 2 [GOOb]
- GPT-4o, GPT-4 Turbo, GPT-4, GPT-3.5 Turbo [OPEd]
- GPT-J [Ele21], [Con24]
- InCoder [FAL+], [FAL+22]
- LlamaAPI, Llama 3.1, Llama 3, Llama 2 [METa]
- Mistral Large 2 [MISb]
- StarCoder, StarCoderBase [LAZ+], [Prob]
- The best large language models (LLMs) in 2024 [Gui24]

9.3.3 Tooling

- AgentGPT [REWa], [REWb]
- Anthropic API [ANTb]
- Awesome-Code-LLM [Cod]
- Awesome Transformers [Bac]

- 15 Best Open Source Text Embedding Models [Pra23a]
- 9 Best Embedding Models for Semantic Search [Pra23b]
- Best 16 Vector Databases for 2024 [Orr]
- CodeQL [GITa]
- Codex [OPE21]
- CodeBERT [Mic]
- Flowise [FLOa], [FLOb]
- GitHub Copilot [Git]
- Hugging Face Transformers [HUG]
- LangChain [LAN]
- LLMs-based-Fuzzer-Survey [PH]
- OpenAI Assistants API [OPEa]
- OpenAI Developer Platform [OPEe]
- Pile-T5 [SKR]
- PolyCoder [Xu+]
- PyTorch torchtune [PyT]
- Reflexion [SCB+23], [SCB+]
- SentenceTransformers [SBE]
- Snyk DeepCode AI [SNYb]
- spaCy [SPA]
- TabNine [TAB]
- ThinkGPT [JINa], [JINb]

9.3.4 Coding Ability

Currently, GPT-4o, GPT-4 Turbo, and Claude 3.5 Sonnet are on top and roughly comparable. This story is usually repeated on every worthwhile up-to-date site we checked.

Anecdotal:

- Aider [Aid24], 133 Exercism Python coding exercises
- Anthropic News [ANTd], HumanEval benchmark
- Bind AI [AI24], Python Code Generation, Web Page Creation, API Query Generation
- Hacker Noon [San24], 5 complete small projects
- Tom's Guide [Mor24], Making a game in Python
- UrAIGuide [Ada24], HTML/CSS responsive footer, JavaScript minutes-to-seconds conversion tool
- Zapier [Kan24], "Road Crosser : Frogger-Inspired Game"

Leader Boards:

- EvalPlus [Liu+b], [Liu+23], [Liu+a]
- BigCodeBench [ZVC+b], [ZVC+24], [ZVC+a]
- Chatbot Arena [ZCS+a], [ZCS+23], [ZCS+b]

- CRUXEval [GRL+a], [GRL+24], [GRL+b]
- EvoEval [XDZa], [XDZ24], [XDZb]
- KLU [Uti24], "Frontier Benchmarks Leaderboard"
- LiveCodeBench [JHG+b], [JHG+24], [JHG+a]
- MHPP [DLF+a], [DLF+24], [DLF+b]
- NaturalCodeBench [ZZL+24], [ZZL+]
- SWE-bench [JYW+b], [JYW+24], [JYW+a]

9.3.5 Books

- AI-Assisted Programming, Better Planning, Coding, Testing, and Deployment [Tau24] (Section "5. Other AI-Assisted Programming Tools" is worth checking out)
- Developing Apps with GPT-4 and ChatGPT, Building Intelligent Chatbots, Content Generators, and More [CB23]
- Foundation Models for Natural Language Processing, Pre-trainted Language Models Integrating Media [PG23]
- Natural Language Processing with Transformers, Building Language Applications with Hugging Face [TWW22]
- A Comprehensive Overview of Large Language Models [NKQ+23]
- Large Language Models: A Survey [MMN+24]
- A Survey of Large Language Models [ZZL+23]
- MM-LLMs: Recent Advances in MultiModal Large Language Models [ZYD+24]
- Stanford's Artificial Intelligence Index Report 2024 [MFP+24]

9.4 Secure Your Computers

ChatGPT 4o, Claude 3.5 Sonnet, and the author have brainstormed some ideas for you. These are only ideas and examples, not endorsements. There are other similar products we haven't listed.

9.4.1 Home Security

- Passwords: strong, unique
- Password Managers: LastPass, 1Password
- 2FA: Enable on important accounts (email, banking, social media)
- Automatic Updates: Enable for OS, browser, and other software
- Antivirus: Bitdefender, Norton, Trend Micro, Windows Defender
- Router Settings: Change default admin password, use WPA3 encryption
- Email and Links: Avoid clicking suspicious links or attachments
- Firewall: Activate built-in firewall (Windows or macOS)
- Regular Backups: Use an external hard drive or cloud storage (iCloud, Google Drive, Dropbox, OneDrive)

- Device Encryption: BitLocker (Windows), FileVault (macOS)
- Hygiene: Reboot devices often, Wi-Fi off when not using, VPN on public Wi-Fi

9.4.2 Corporate Security

- AI/ML Tools: Darktrace, Cortex XDR
- SIEM System: Splunk, IBM QRadar, ArcSight
- Endpoint Security: Symantec Endpoint Protection, Microsoft Defender ATP
- Update and Patch Management: Microsoft SCCM, Ivanti, SolarWinds
- Zero Trust Network: Cisco Zero Trust, Zscaler
- Continuous Verification: Okta, Microsoft Azure AD, RBAC
- Backups: Veeam, Acronis, offsite and air-gapped backups, integrity and recovery tests
- Compartmentalization: VLANs, firewalls, network segmentation
- Isolate Critical Systems: air-gapped networks, principle of least privilege (PoLP), time-based restrictions, physical access controls, automated account deactivation
- MFA and FIDO/FIDO2: Okta, Duo Security, Yubikey, Feitian ePass, Solo
- Kill Switches: DataPro E-Kill Switch, Hak5 Packet Squirrel, Purism Librem Laptops
- Regular Security Audits: security standards, manage log files, identify sensitive data, penetration testing, vulnerability assessments [Coo23]
- Employee Training: regular cybersecurity awareness programs
- Incident Response Plan: established procedures for security breaches
- Data Loss Prevention (DLP): Symantec DLP, McAfee DLP, Forcepoint DLP

References

[Ada24] Zahid Adam. *Claude 3.5 Sonnet vs GPT4o: Side-by-Side Tests*. 2024. URL: https://uraiguide.com/claude-35-sonnet-vs-gpt4o/.

[Adv23] Joint Cybersecurity Advisory. *Hunting Russian Intelligence "Snake" Malware*. 2023. URL: https://media.defense.gov/2023/May/09/2003218554/-1/-1/1/JOINT_CSA_HUNTING_RU_INTEL_SNAKE_MALWARE_20230509.PDF.

[AI24] Bind AI. *Claude 3.5 Sonnet vs GPT-4o: Does Claude outperform GPT-4o?* 2024. URL: https://blog.getbind.co/2024/06/21/claude-3-5-sonnet-does-it-outperform-gpt-4o/.

[Aid24] Aider. *Claude 3 beats GPT-4 on Aider's code editing benchmark*. 2024. URL: https://aider.chat/2024/03/08/claude-3.html.

[ANTa] ANTHROPIC.COM. *Anthropic API*. URL: https://www.anthropic.com/api.

[ANTb] ANTHROPIC.COM. *API Reference*. URL: https://docs.anthropic.com/en/api/getting-started.

[ANTc] ANTHROPIC.COM. *Claude*. URL: https://www.anthropic.com/claude.

[ANTd] ANTHROPIC.COM. *Claude 3.5 Sonnet*. URL: https://www.anthropic.com/news/claude-3-5-sonnet.

[Bac] Anton Bacaj. *Awesome Transformers*. URL: https://github.com/abacaj/awesome-transformers.

[Bar24a] Fabio Baroni. *Awesome-Exploit-Development*. 2024. URL: https://github.com/FabioBaroni/awesome-exploit-development.

[Bar24b] Christine Barry. *5 Ways cybercriminals are using AI: Malware generation*. 2024. URL: https://blog.barracuda.com/2024/04/16/5-ways-cybercriminals-are-using-ai--malware-generation.

[BBS+24] John Anthony Boyer, Timothy Owen Bazalgette, Jack Stockdale, et al. *INTERACTIVE CYBER SECURITY USER INTERFACE*. U.S. Patent. 2024. URL: https://ppubs.uspto.gov/dirsearch-public/print/downloadPdf/20240045990.

[BCL+] Manish Bhatt, Sahana Chennabasappa, Yue Li, et al. *Cybersecurity Benchmarks*. URL: https://github.com/meta-llama/PurpleLlama/tree/main/CybersecurityBenchmarks.

[BCL+24] Manish Bhatt, Sahana Chennabasappa, Yue Li, et al. *CyberSecEval 2: A Wide-Ranging Cybersecurity Evaluation Suite for Large Language Models*. 2024. URL: https://arxiv.org/abs/2404.13161.

[BCN+23] Manish Bhatt, Sahana Chennabasappa, Cyrus Nikolaidis, et al. *Purple Llama CyberSecEval: A Secure Coding Benchmark for Language Models*. 2023. URL: https://arxiv.org/abs/2312.04724.

[Beg+23] Nils Begou et al. *Exploring the Dark Side of AI: Advanced Phishing Attack Design and Deployment Using ChatGPT*. 2023. URL: https://arxiv.org/abs/2309.10463.

[Bel] Greg Belding. *Top 7 malware sample databases and datasets for research and training*. URL: https://www.infosecinstitute.com/resources/malware-analysis/top-7-malware-sample-databases-and-datasets-for-research-and-training/.

[Ben+23] Nathan Benaich et al. *State of AI Report 2023*. 2023. URL: https://docs.google.com/presentation/d/156WpBF_rGvf4Ecg19oM1fyR51g4FAmHV3Zs0WLukrLQ.

[BGC22] Sharon Ben-Moshe, Gil Gekker, and Golan Cohen. *OPWNAI: AI THAT CAN SAVE THE DAY OR HACK IT AWAY*. 2022. URL: https://research.checkpoint.com/2022/opwnai-ai-that-can-save-the-day-or-hack-it-away/.

[BKN+23] Atieh Bakhshandeh, Abdalsamad Keramatfar, Amir Norouzi, et al. *Using ChatGPT as a Static Application Security Testing Tool*. 2023. URL: https://arxiv.org/abs/2308.14434.

[BMC+24] Dillon Bowen, Brendan Murphy, Will Cai, et al. *Scaling Laws for Data Poisoning in LLMs*. 2024. URL: https://arxiv.org/abs/2408.02946.

[BNM] Guru Bhandari, Amara Naseer, and Leon Moonen. *CVEfixes: Automated Collection of Vulnerabilities and Their Fixes from Open-Source Software*. URL: https://github.com/secureIT-project/CVEfixes.

[BNM21] Guru Bhandari, Amara Naseer, and Leon Moonen. "CVEfixes: Automated Collection of Vulnerabilities and Their Fixes from Open-Source Software". In: *Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '21)*. ACM, 2021, p. 10.

[Bot] BotFather. *BotFather*. URL: https://botostore.com/c/botfather/.

[Bot23] Marcus Botacin. "GPThreats-3: Is Automatic Malware Generation a Threat?" In: *17th IEEE Workshop on Offensive Technologies*. 2023. URL: https://wootconference.org/papers/woot23-paper8.pdf.

[BPC23] Mika Beckerich, Laura Plein, and Sergio Coronado. *RatGPT: Turning online LLMs into Proxies for Malware Attacks*. 2023. URL: https://arxiv.org/abs/2308.09183.

[Braa] Andrei Brazhuk. *OWASP Ontology Driven Threat Modeling Framework*. URL: https://owasp.org/www-project-ontology-driven-threat-modeling-framework/.

[Brab] Andrei Brazhuk. *OWASP Ontology-driven Threat Modelling framework*. URL: https://github.com/OWASP/OdTM.

[Bra20] Andrei Brazhuk. "Security patterns based approach to automatically select mitigations in ontology-driven threat modelling". In: Feb. 2020. URL: https://www.researchgate.net/publication/339415212_Security_patterns_based_approach_to_automatically_select_mitigations_in_ontology-driven_threat_modelling.

[BUS23] BUSINESSWIRE.COM. *Vicarius Introduces vuln_GPT: The World's First LLM Model to Find and Fix Software Vulnerabilities*. 2023. URL: https://www.businesswire.com/news/home/20230808696298/en/Vicarius-Introduces-vuln_GPT-The-World%E2%80%99s-First-LLM-Model-to-Find-and-Fix-Software-Vulnerabilities/.

[Cat] F. Ozgur Catak. *Windows Malware Dataset with PE API Calls*. URL: https://github.com/ocatak/malware_api_class.

[CB22] Ipshita Roy Chowdhury and Deepayan Bhowmik. "Capturing Malware Behaviour with Ontology-based Knowledge Graphs". In: *Conference on Dependable and Secure Computing (DSC)*. IEEE, 2022.

[CB23] Olivier Caelen and Marie-Alice Biete. *Developing Apps with GPT-4 and ChatGPT, Building Intelligent Chatbots, Content Generators, and More*. O'Reilly, 2023. ISBN: 978-1098152482.

[CDA+] Yizheng Chen, Zhoujie Ding, Lamya Alowain, et al. *DiverseVul: A New Vulnerable Source Code Dataset for Deep Learning Based Vulnerability Detection*. URL: https://github.com/wagner-group/diversevul.

[CDA+23] Yizheng Chen, Zhoujie Ding, Lamya Alowain, et al. *DiverseVul: A New Vulnerable Source Code Dataset for Deep Learning Based Vulnerability Detection*. 2023. URL: https://arxiv.org/abs/2304.00409.

[Cha] Raj Chandel. *Hacking Articles*. URL: https://www.hackingarticles.in.

[Cha+] Saikat Chakraborty et al. *Deep Learning based Vulnerability Detection: Are We There Yet?* URL: https://github.com/VulDetProject/ReVeal.

[Cha+23] P. V. Sai Charan et al. *From Text to MITRE Techniques: Exploring the Malicious Use of Large Language Models for Generating Cyber Attack Payloads*. 2023. URL: https://arxiv.org/abs/2305.15336.

[CHE24] CHECKPOINT.COM. *OPWNAI : CYBERCRIMINALS STARTING TO USE CHATGPT*. 2024. URL: https://research.checkpoint.com/2023/opwnai-cybercriminals-starting-to-use-chatgpt/.

[CKM+23] Aaron Chan, Anant Kharkar, Roshanak Zilouchian Moghaddam, et al. *Transformer-based Vulnerability Detection in Code at Edit-Time: Zero-shot, Few-shot, or Fine-tuning?* 2023. URL: https://arxiv.org/abs/2306.01754.

[CLZ+24] Tianyu Chen, Lin Li, Liuchuan Zhu, et al. *Harnessing the Power of LLM to Support Binary Taint Analysis*. 2024. URL: https://arxiv.org/abs/2308.04662.

[Cod] CodeFuse. *Awesome-Code-LLM*. URL: https://github.com/codefuse-ai/Awesome-Code-LLM.

[Coma] Malware Analysis Community. *Malware Analysis Community*. URL: https://malfav.gitbook.io/home.

[Comb] Malware Analysis Community. *Malware Sample Resources*. URL: https://malfav.gitbook.io/home/malware-resources/malware-sample-resources.

[Cona] Wikipedia Contributors. *Agent Tesla*. URL: https://en.wikipedia.org/wiki/Agent_Tesla.

[Conb] Wikipedia Contributors. *Emotet*. URL: https://en.wikipedia.org/wiki/Emotet.

[Conc] Wikipedia Contributors. *Okapi BM25*. URL: https://en.wikipedia.org/wiki/Okapi_BM25.

[Cond] Wikipedia Contributors. *Red team*. URL: https://en.wikipedia.org/wiki/Red_team.

[Con24] Wikipedia Contributors. *GPT-J*. 2024. URL: https://en.wikipedia.org/wiki/GPT-J.

[Coo23] Stephen Cooper. *How to Perform an IT Security Audit – Step-by-Step Guide and Tools*. 2023. URL: https://www.comparitech.com/net-admin/it-security-audit/.

[Coo24] Sam Cook. *Malware Attack Statistics and Facts for 2024: What You Need to Know*. 2024. URL: https://www.comparitech.com/antivirus/malware-statistics-facts/.

[CPP] CPPCHECK.COM. *CppCheck*. URL: https://www.cppcheck.com.

[CSS+24] Amitai Cohen, Barak Sharoni, Alon Schindel, et al. *Cybersecurity Incident Response Techniques Utilizing Artificial Intelligence*. U.S. Patent. 2024. URL: https://ppubs.uspto.gov/dirsearch-public/print/downloadPdf/12001550.

[CVE] CVE.ORG. *CVE Program Mission*. URL: https://www.cve.org.

[DAR] DARKNET.ORG.UK. *DarkNet Hacking Tools*. URL: https://www.darknet.org.uk/category/hacking-tools/.

[DBG24] Gil David, Ella Bor, and Matan Giladi. *A dataset-free approach to leveraging LLMs for malicious code detection*. 2024. URL: https://apiiro.com/blog/llm-based-dataset-free-malicious-code-detection-research/.

[Del23] Alex Delamotte. *Predator AI — ChatGPT-Powered Infostealer Takes Aim at Cloud Platforms*. 2023. URL: https://www.sentinelone.com/labs/predator-ai-chatgpt-powered-infostealer-takes-aim-at-cloud-platforms/.

[Des23] TN Tech Desk. *ChatGPT'S Evil Step Cousins: The Rise of Black Hat AI Tools XXXGPT and Wolf GPT*. 2023. URL: https://www.timesnownews.com/technology-science/chatgpts-evil-step-cousins-the-rise-of-black-hat-ai-tools-xxxgpt-and-wolf-gpt-article-102500799.

[DLF+a] Jianbo Dai, Jianqiao Lu, Yunlong Feng, et al. *MHPP Leaderboard*. URL: https://sparksofagi.github.io/MHPP/.

[DLF+b] Jianbo Dai, Jianqiao Lu, Yunlong Feng, et al. *MHPP: Exploring the Capabilities and Limitations of Language Models Beyond Basic Code Generation*. URL: https://github.com/SparksofAGI/MHPP.

[DLF+24] Jianbo Dai, Jianqiao Lu, Yunlong Feng, et al. *MHPP: Exploring the Capabilities and Limitations of Language Models Beyond Basic Code Generation*. 2024. URL: https://arxiv.org/abs/2405.11430.

[DLS24] Deepen Desai, Dianhuan Lin, and Rex Shang. *Breach Prediction Via Machine Learning*. U.S. Patent. 2024. URL: https://ppubs.uspto.gov/dirsearch-public/print/downloadPdf/20230370495.

[DLV+23] Gelei Deng, Yi Liu, Víctor Mayoral Vilches, et al. *PentestGPT: An LLM-empowered Automatic Penetration Testing Tool*. 2023. URL: https://arxiv.org/abs/2308.06782.

[Dut23] Tushar Subhra Dutta. *Hackers Released New Black Hat AI Tools XXXGPT and Wolf GPT*. 2023. URL: https://cybersecuritynews.com/black-hat-ai-tools-xxxgpt-and-wolf-gpt/.

[DZW+] Xueying Du, Geng Zheng, Kaixin Wang, et al. *Enhancing LLM-based Vulnerability Detection via Knowledge-level RAG*. URL: https://github.com/KnowledgeRAG4LLMVulD/KnowledgeRAG4LLMVulD.

[DZW+24] Xueying Du, Geng Zheng, Kaixin Wang, et al. *Vul-RAG: Enhancing LLM-based Vulnerability Detection via Knowledge-level RAG*. 2024. URL: https://arxiv.org/abs/2406.11147.

[ELA] ELASTIC.CO. *Elasticsearch*. URL: https://www.elastic.co/elasticsearch.

[Ele21] EleutherAI. *ChatGPT*. 2021. URL: https://huggingface.co/docs/transformers/model_doc/gptj.

[Erz23] Arthur Erzberger. *WormGPT and FraudGPT – The Rise of Malicious LLMs*. 2023. URL: https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/wormgpt-and-fraudgpt-the-rise-of-malicious-llms/.

[Est] Jordi Armengol Estapé. *ExeBench: an ML-scale dataset of executable C functions*. URL: https://github.com/jordiae/exebench.

[FAB+24] Mohamed Amine Ferrag, Fatima Alwahedi, Ammar Battah, et al. *Generative AI and Large Language Models for Cyber Security: All Insights You Need*. 2024. URL: https://arxiv.org/abs/2405.12750.

[FAL+] Daniel Fried, Armen Aghajanyan, Jessy Lin, et al. *InCoder: A Generative Model for Code Infilling and Synthesis*. URL: https://github.com/dpfried/incoder.

[FAL+22] Daniel Fried, Armen Aghajanyan, Jessy Lin, et al. *InCoder: A Generative Model for Code Infilling and Synthesis*. 2022. URL: https://arxiv.org/abs/2204.05999.

[Fan+] Jiahao Fan et al. *A C/C++ Code Vulnerability Dataset with Code Changes and CVE Summaries*. URL: https://github.com/ZeoVan/MSR_20_Code_vulnerability_CSV_Dataset.

[Fan+24] Richard Fang et al. *LLM Agents can Autonomously Exploit One-day Vulnerabilities*. 2024. URL: https://arxiv.org/abs/2404.08144.

[FBG+24] Richard Fang, Rohan Bindu, Akul Gupta, et al. *Teams of LLM Agents can Exploit Zero-Day Vulnerabilities*. 2024. URL: https://arxiv.org/abs/2406.01637.

[FBI] FBINFER.COM. *Infer*. URL: https://fbinfer.com/.

[FLOa] FLOWISEAI.COM. *Flowise*. URL: https://flowiseai.com.

[FLOb] FLOWISEAI.COM. *Flowise - Build LLM Apps Easily*. URL: https://github.com/FlowiseAI/Flowise.

[Flo] Flora. *F*NG InfoSec*. URL: https://fnginfosec.github.io.

[FNT+24] Mohamed Amine Ferrag, Mthandazo Ndhlovu, Norbert Tihanyi, et al. *Revolutionizing Cyber Threat Detection with Large Language Models: A privacy-preserving BERT-based Lightweight Model for IoT/IIoT Devices*. 2024. URL: https://arxiv.org/abs/2306.14263.

[Fru22] Josh Fruhlinger. *11 infamous malware attacks: The first and the worst*. 2022. URL: https://www.csoonline.com/article/572911/11-infamous-malware-attacks-the-first-and-the-worst.html.

[GAA+23] Maanak Gupta, Charankumar Akiri, Kshitiz Aryal, et al. *From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy*. 2023. URL: https://arxiv.org/abs/2307.00691.

[GB24] Sergei Glazunov and Mark Brand. *Project Naptime: Evaluating Offensive Security Capabilities of Large Language Models*. 2024. URL: https://googleprojectzero.blogspot.com/2024/06/project-naptime.html.

[Ger20] Tom Gerencer. *The Top 10 Worst Computer Viruses in History*. 2020. URL: https://www.hp.com/us-en/shop/tech-takes/top-ten-worst-computer-viruses-in-history.

[GITa] GITHUB.COM. *CodeQL*. URL: https://codeql.github.com.

[GITb] GITHUB.COM. *GitHub Advisory Database*. URL: https://github.com/advisories.

[Git] GitHub. *GitHub Copilot*. URL: https://docs.github.com/en/copilot.

[Goe24] Corentin Goetghebeur. *PoisonGPT*. 2024. URL: https://github.com/CorentinGoet/PoisonGPT.

[GOOa] GOOGLE.COM. *Gemini*. URL: https://deepmind.google/technologies/gemini/.

[GOOb] GOOGLE.COM. *Google AI for Developers*. URL: https://ai.google.dev.

[GRL+a] Alex Gu, Baptiste Rozière, Hugh Leather, et al. *CRUXEval Leaderboard*. URL: https://crux-eval.github.io/leaderboard.html.

[GRL+b] Alex Gu, Baptiste Rozière, Hugh Leather, et al. *CRUXEval: Code Reasoning, Understanding, and Execution Evaluation*. URL: https://github.com/facebookresearch/cruxeval.

[GRL+24] Alex Gu, Baptiste Rozière, Hugh Leather, et al. *CRUXEval: A Benchmark for Code Reasoning, Understanding and Execution*. 2024. URL: https://arxiv.org/abs/2401.03065.

[Gro] Anti-Phishing Working Group. *APWG eCrime Exchange (eCX)*. URL: https://apwg.org/ecx/.

[Gui24] Harry Guinness. *The best large language models (LLMs) in 2024*. 2024. URL: https://zapier.com/blog/best-llm/.

[GWZ+] Zeyu Gao, Hao Wang, Yuchen Zhou, et al. *VulBench*. URL: https://github.com/Hustcw/VulBench.

[GWZ+23] Zeyu Gao, Hao Wang, Yuchen Zhou, et al. *How Far Have We Gone in Vulnerability Detection Using Large Language Models*. 2023. URL: https://arxiv.org/abs/2311.12420.

[GZ24] Nguyen Hoang Giang and Yi Helen Zhang. *Growing Number of Threats Leveraging AI*. 2024. URL: https://symantec-enterprise-blogs.security.com/threat-intelligence/malware-ai-llm.

[HAC] HACKTHEBOX.COM. *Hack The Box*. URL: https://www.hackthebox.com.

[HBE+24] Anson Ho, Tamay Besiroglu, Ege Erdil, et al. *Algorithmic progress in language models*. 2024. URL: https://arxiv.org/abs/2403.05812.

[HHH+23] Hossein Hajipour, Keno Hassler, Thorsten Holz, et al. *CodeLMSec Benchmark: Systematically Evaluating and Finding Security Vulnerabilities in Black-Box Code Language Models*. 2023. URL: https://arxiv.org/abs/2302.04012.

[HKC] Andreas Happe, Aaron Kaplan, and Jürgen Cito. *HackingBuddyGPT*. URL: https://github.com/ipa-lab/hackingBuddyGPT.

[HKC23] Andreas Happe, Aaron Kaplan, and Jürgen Cito. *LLMs as Hackers: Autonomous Linux Privilege Escalation Attacks*. 2023. URL: https://arxiv.org/abs/2310.11409.

[HUG] HUGGINGFACE.CO. *Transformers*. URL: https://huggingface.co/docs/transformers/index.

[Hut24] Justin Hutchens. *The Language of Deception: Weaponizing Next Generation AI*. Wiley, 2024. ISBN: 978-1394222544.

[IMM] IMMUNITYINC.COM. *Immunity Debugger*. URL: https://www.immunityinc.com/products/debugger/.

[IMP] IMPACTCYBERTRUST.ORG. *IMPACT - Software Assurance Reference Dataset*. URL: https://www.impactcybertrust.org/dataset_view?idDataset=1298.

[Inc] Ericsson Inc. *CodeChecker*. URL: https://github.com/Ericsson/codechecker.

[INF] INFOSECWRITEUPS.COM. *InfoSec WriteUps*. URL: https://infosecwriteups.com.

[Insa] Legal Information Institute. *18 U.S. Code § 1030 - Fraud and related activity in connection with computers*. URL: https://www.law.cornell.edu/uscode/text/18/1030.

[Insb] Legal Information Institute. *18 U.S. Code § 2510 - Definitions*. URL: https://www.law.cornell.edu/uscode/text/18/2510.

[Insc] Legal Information Institute. *18 U.S. Code § 2701 - Unlawful access to stored communications*. URL: https://www.law.cornell.edu/uscode/text/18/2701.

[Int24] Microsoft Threat Intelligence. *Staying ahead of threat actors in the age of AI*. 2024. URL: https://www.microsoft.com/en-us/security/blog/2024/02/14/staying-ahead-of-threat-actors-in-the-age-of-ai/.

[INV] INVICTI.COM. *Invicti Learn*. URL: https://www.invicti.com/learn/.

[JHG+a] Naman Jain, King Han, Alex Gu, et al. *LiveCodeBench*. URL: https://github.com/LiveCodeBench/LiveCodeBench.

[JHG+b] Naman Jain, King Han, Alex Gu, et al. *LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code*. URL: https://livecodebench.github.io/leaderboard.html.

[JHG+24] Naman Jain, King Han, Alex Gu, et al. *LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code*. 2024. URL: https://arxiv.org/abs/2403.07974.

[JINa] JINA.AI. *ThinkGPT*. URL: https://github.com/jina-ai/thinkgpt.

[JINb] JINA.AI. *ThinkGPT*. URL: https://github.com/jina-ai/thinkgpt.

[Jov24] Bojan Jovanovic. *A Not-So-Common Cold: Malware Statistics in 2024*. 2024. URL: https://dataprot.net/statistics/malware-statistics/.

[JYW+a] Carlos E. Jimenez, John Yang, Alexander Wettig, et al. *SWE-bench*. URL: https://github.com/princeton-nlp/SWE-bench.

[JYW+b] Carlos E. Jimenez, John Yang, Alexander Wettig, et al. *SWE-bench: Can Language Models Resolve Real-World GitHub Issues?* URL: https://www.swebench.com.

[JYW+24] Carlos E. Jimenez, John Yang, Alexander Wettig, et al. *SWE-bench: Can Language Models Resolve Real-World GitHub Issues?* 2024. URL: https://arxiv.org/abs/2310.06770.

[KALa] KALI.ORG. *DIRB*. URL: https://www.kali.org/tools/dirb/.

[KALb] KALI.ORG. *Kali Tools*. URL: https://www.kali.org/tools.

[Kan24] Ryan Kane. *Claude vs. ChatGPT: What's the difference? [2024]*. 2024. URL: https://zapier.com/blog/claude-vs-chatgpt.

[Kat22] Eyal Katz. *Top 5 IAST Tools for 2022*. 2022. URL: https://spectralops.io/blog/iast-tools-for-2022/.

[Kel23] Daniel Kelley. *WormGPT – The Generative AI Tool Cybercriminals Are Using to Launch Business Email Compromise Attacks*. 2023. URL: https://web.archive.org/web/20240627082228/https://slashnext.com/blog/wormgpt-the-generative-ai-tool-cybercriminals-are-using-to-launch-business-email-compromise-attacks/.

[Koi+23] Takashi Koide et al. *Detecting Phishing Sites Using ChatGPT*. 2023. URL: https://arxiv.org/abs/2306.05816.

[Kre23] Brian Krebs. *Meet the Brains Behind the Malware-Friendly AI Chat Service 'WormGPT'*. 2023. URL: https://krebsonsecurity.com/2023/08/meet-the-brains-behind-the-malware-friendly-ai-chat-service-wormgpt/.

[Kri23] Rackesh Krishnan. *FraudGPT: The Villain Avatar of ChatGPT*. 2023. URL: https://netenrich.com/blog/fraudgpt-the-villain-avatar-of-chatgpt.

[Lak] Md Zahidul Islam Laku. *Kali Linux Commands Cheat Sheet [Free PDF Download]*. URL: https://linuxsimply.com/cheat-sheets/kali-linux-commands/.

[Lam23] John Lambert. *Microsoft shifts to a new threat actor naming taxonomy*. 2023. URL: https://www.microsoft.com/en-us/security/blog/2023/04/18/microsoft-shifts-to-a-new-threat-actor-naming-taxonomy/.

[LAN] LANGCHAIN.COM. *LangChain*. URL: https://www.langchain.com.

[LAZ+] Raymond Li, Loubna Ben Allal, Yangtian Zi, et al. *StarCoder: may the source be with you!* URL: https://arxiv.org/abs/2305.06161.

[LCW+24] Zhe Liu, Chunyang Chen, Junjie Wang, et al. "Testing the Limits: Unusual Text Inputs Generation for Mobile App Crash Detection with Large Language Model". In: *Proceedings of the 46th International Conference on Software Engineering*. ICSE '24. 2024. URL: https://arxiv.org/abs/2310.15657.

[LH24] Frank Weizhen Liu and Chenhui Hu. *Exploring Vulnerabilities and Protections in Large Language Models: A Survey*. 2024. URL: https://arxiv.org/abs/2406.00240.

[Liu+a] Jiawei Liu et al. URL: https://github.com/evalplus/evalplus.

[Liu+b] Jiawei Liu et al. *EvalPlus Leaderboard*. URL: https://evalplus.github.io/leaderboard.html.

[Liu+23] Jiawei Liu et al. *Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation*. 2023. URL: https://arxiv.org/abs/2305.01210.

[LLV] LLVM.ORG. *AddressSanitizer*. URL: https://clang.llvm.org/docs/AddressSanitizer.html.

[LSB24] Zefang Liu, Jialei Shi, and John F. Buford. "CyberBench: A Multi-Task Benchmark for Evaluating Large Language Models in Cybersecurity". In: *AAAI-24 Workshop on Artificial Intelligence for Cyber Security (AICS)*. 2024. URL: https://www.researchgate.net/publication/378267627_CyberBench_A_Multi-Task_Benchmark_for_Evaluating_Large_Language_Models_in_Cybersecurity.

[LSZ+23] Puzhuo Liu, Chengnian Sun, Yaowen Zheng, et al. *Harnessing the Power of LLM to Support Binary Taint Analysis*. 2023. URL: https://arxiv.org/abs/2310.08275.

[LWX+23] Ruitong Liu, Yanbin Wang, Haitao Xu, et al. *Malicious URL Detection via Pretrained Language Model*. 2023. URL: https://arxiv.org/abs/2311.12372.

[LWX+24] Zichuan Liu, Zefan Wang, Linjie Xu, et al. *Protecting Your LLMs with Information Bottleneck*. 2024. URL: https://arxiv.org/abs/2404.13968.

[Mar23] David Marshall. *Vicarius Introduces vuln_GPT: The World's First LLM Model to Find and Fix Software Vulnerabilities*. 2023. URL: https://vmblog.com/archive/2023/08/09/vicarius-introduces-vuln-gpt-the-world-s-first-llm-model-to-find-and-fix-software-vulnerabilities.aspx.

[McL24] Mike McLean. *2024 Must-Know Cyber Attack Statistics and Trends*. 2024. URL: https://www.embroker.com/blog/cyber-attack-statistics/.

[MCS+20] Wentao Ma, Yiming Cui, Chenglei Si, et al. *CharBERT: Character-aware Pre-trained Language Model*. 2020. URL: https://arxiv.org/abs/2011.01513.

[Men+24] Ruijie Meng et al. "Large Language Model guided Protocol Fuzzing". In: *Network and Distributed System Security (NDSS) Symposium*. 2024. URL: https://www.ndss-symposium.org/wp-content/uploads/2024-556-paper.pdf.

[METa] META.COM. *LlamaAPI*. URL: https://llama.meta.com.

[METb] META.COM. *Meta AI*. URL: https://www.meta.ai.

[METc] METASPLOIT.COM. *Metasploit*. URL: https://www.metasploit.com.

[MFP+24] Nestor Maslej, Loredana Fattorini, Raymond Perrault, et al. *Artificial Intelligence Index Report 2024*. 2024. URL: https://aiindex.stanford.edu/report/.

[Mic] Microsoft. *CodeBERT*. URL: https://github.com/microsoft/CodeBERT.

[MISa] MISTRAL.AI. *Le Chat*. URL: https://auth.mistral.ai/ui/login.

[MISb] MISTRAL.AI. *Mistral Large*. URL: https://mistral.ai.

[MITa] MITRE.ORG. *Common Attack Pattern Enumeration and Classification*. URL: https://capec.mitre.org/about/index.html.

[MITb] MITRE.ORG. *CWE - Common Weakness Enumeration*. URL: https://cwe.mitre.org.

[MITc] MITRE.ORG. *MITRE ATT&CK Enterprise matrix*. URL: https://attack.mitre.org/matrices/enterprise/.

[MLT24] Tommy Madjar, Selena Larson, and The Proofpoint Threat Research Team. *TA547 Uses an LLM-Generated Dropper to Infect German Orgs.* 2024. URL: https://www.proofpoint.com/us/blog/threat-insight/security-brief-ta547-targets-german-organizations-rhadamanthys-stealer.

[MMN+24] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, et al. *Large Language Models: A Survey.* 2024. URL: https://arxiv.org/abs/2402.06196.

[Mor24] Ryan Morrison. *ChatGPT-4o vs Claude 3.5 Sonnet — which AI chatbot wins?* 2024. URL: https://www.tomsguide.com/ai/chatgpt-4o-vs-claude-35-sonnet-which-ai-platform-wins.

[Mos+23] Stephen Moskal et al. *LLMs Killed the Script Kiddie: How Agents Supported by Large Language Models Change the Landscape of Network Threat Testing.* 2023. URL: https://arxiv.org/abs/2310.06936.

[Mur22] Adam Murray. *Best SAST Tools: Top 7 Solutions Compared.* 2022. URL: https://www.mend.io/blog/best-sast-tools.

[Nel24] Nate Nelson. *TA547 Uses an LLM-Generated Dropper to Infect German Orgs.* 2024. URL: https://www.darkreading.com/threat-intelligence/ta547-uses-llm-generated-dropper-infect-german-orgs.

[New23] The Hacker News. *New AI Tool 'FraudGPT' Emerges, Tailored for Sophisticated Attacks.* 2023. URL: https://thehackernews.com/2023/07/new-ai-tool-fraudgpt-emerges-tailored.html.

[NISa] NIST.GOV. *NVD - NATIONAL VULNERABILITY DATABASE.* URL: https://www.nist.gov/itl/nvd.

[NISb] NIST.GOV. *SARD - NIST SOFTWARE ASSURANCE REFERENCE DATASET.* URL: https://samate.nist.gov/SARD/.

[NKQ+23] Humza Naveed, Asad Ullah Khan, Shi Qiu, et al. *A Comprehensive Overview of Large Language Models.* 2023. URL: https://arxiv.org/abs/2307.06435.

[Noe23] David Noever. *Can Large Language Models Find And Fix Vulnerable Software?* 2023. URL: https://arxiv.org/abs/2308.10345.

[NP] Roberto Natella and Van-Thuan Pham. *ProFuzzBench – A Benchmark for Stateful Protocol Fuzzing.* URL: https://github.com/profuzzbench/profuzzbench.

[NSY+24] Chao Ni, Liyu Shen, Xiaohu Yang, et al. *MegaVul: A C/C++ Vulnerability Dataset with Comprehensive Code Representations.* 2024. URL: https://arxiv.org/html/2406.12415.

[NT24] Tomasz Andrzej Nidecki and Aleksei Tiurin. *Reverse Shell.* 2024. URL: https://www.invicti.com/learn/reverse-shell/.

[OPEa] OPENAI.COM. *Assistants API Overview*. URL: https://platform.openai.com/docs/assistants/overview.

[OPEb] OPENAI.COM. *ChatGPT*. URL: https://openai.com/chatgpt/.

[OPEc] OPENAI.COM. *HumanEval*. URL: https://github.com/openai/human-eval.

[OPEd] OPENAI.COM. *Models*. URL: https://platform.openai.com/docs/models.

[OPEe] OPENAI.COM. *OpenAI developer platform*. URL: https://platform.openai.com/docs/overview.

[OPEf] OPENAI.COM. *text-embedding-ada-002*. URL: https://openai.com/index/new-and-improved-embedding-model/.

[OPEg] OPENTEXT.COM. *Fortify Static Code Analyzer*. URL: https://www.opentext.com/products/fortify-static-code-analyzer.

[OPE21] OPENAI.COM. *OpenAI Codex*. 2021. URL: https://openai.com/index/openai-codex/.

[Orr] Einat Orr. *Best 16 Vector Databases for 2024*. URL: https://lakefs.io/blog/12-vector-databases-2023/.

[OWAa] OWASP.ORG. *OWASP Benchmark*. URL: https://owasp.org/www-project-benchmark.

[OWAb] OWASP.ORG. *OWASP Projects*. URL: https://owasp.org/projects/.

[OWAc] OWASP.ORG. *OWASP Web Security Testing Guide*. URL: https://github.com/OWASP/wstg/tree/master/document.

[OWAd] OWASPDirBuster. *OWASP DirBuster Project*. URL: https://wiki.owasp.org/index.php/Category:OWASP_DirBuster_Project.

[PAR] PARAMIKO.ORG. *Paramiko*. URL: https://www.paramiko.org.

[PB23] Sudipta Paria and Swarup Bhunia. *DiSPEL: Distributed Security Policy Enforcement for Bus-based SoC*. 2023. URL: https://arxiv.org/abs/2308.02792.

[PCc] Emilio Pinna, Andrea Cardaci, and contributors. *GTFObins*. URL: https://gtfobins.github.io/.

[PCL24] Constantinos Patsakis, Fran Casino, and Nikolaos Lykousas. *Assessing LLMs in Malicious Code Deobfuscation of Real-world Malware Campaigns*. 2024. URL: https://arxiv.org/abs/2404.19715.

[PDB23] Sudipta Paria, Aritra Dasgupta, and Swarup Bhunia. *DIVAS: An LLM-based End-to-End Framework for SoC Security Analysis and Policy-based Protection*. 2023. URL: https://arxiv.org/abs/2308.06932.

[PG23] Gerhard Paaß and Sven Giesselbach. *Foundation Models for Natural Language Processing, Pre-trainted Language Models Integrating Media*. Springer, 2023. ISBN: 978-3031231896.

[PH] Ed Puth and Han Ling Huang. *LLMs-based-Fuzzer-Survey*. URL: https://github.com/EdPuth/LLMs-based-Fuzzer-Survey.

[PHI] PHISHTANK.ORG. *PhishTank, Out of the Net, into the Tank.* URL: https://phishtank.org.

[PICa] PICOCTF.ORG. *picoCTF*. URL: https://picoctf.org.

[PICb] PICOCTF.ORG. *picoCTF 2021 Mini-Competition with redpwn*. URL: https://picoctf.org/competitions/2021-redpwn.html.

[PLA] PLAYWRIGHT.DEV. *Playwright*. URL: https://playwright.dev.

[Pla23] Henrik Plate. *LLM-assisted Malware Review: AI and Humans Join Forces to Combat Malware*. 2023. URL: https://www.endorlabs.com/learn/llm-assisted-malware-review-ai-and-humans-join-forces-to-combat-malware.

[Pola] Carlos Polop. *HackTricks*. URL: https://book.hacktricks.xyz.

[Polb] Carlos Polop. *PEASS-ng - Privilege Escalation Awesome Scripts SUITE new generation*. URL: https://github.com/peass-ng/PEASS-ng.

[PORa] PORTSWIGGER.NET. *Burp Suite*. URL: https://portswigger.net/burp.

[PORb] PORTSWIGGER.NET. *Web Security Academy*. URL: https://portswigger.net/web-security.

[Pra23a] JD Prater. *15 Best Open Source Text Embedding Models*. 2023. URL: https://www.graft.com/blog/open-source-text-embedding-models.

[Pra23b] JD Prater. *9 Best Embedding Models for Semantic Search*. 2023. URL: https://www.graft.com/blog/text-embeddings-for-search-semantic.

[Proa] BigCode Project. *HuggingChat*. URL: https://huggingface.co/chat/.

[Prob] BigCode Project. *StarCoder*. URL: https://github.com/bigcode-project/starcoder.

[PTK+23] Yin Minn Pa Pa, Shunsuke Tanizaki, Tetsui Kou, et al. "An attacker's dream? exploring the capabilities of chatgpt for developing malware." In: *Proceedings of the 16th Cyber Security Experimentation and Test Workshop*. 2023, pp. 10–18. URL: https://cset23.isi.edu/slides/cset2023-slides-papa.pdf.

[PyT] Team PyTorch. *torchtune: Easily fine-tune LLMs using PyTorch*. URL: https://pytorch.org/blog/torchtune-fine-tune-llms/.

[Qui23] Bernardo Quintero. *Introducing VirusTotal Code Insight: Empowering threat analysis with generative AI*. 2023. URL: https://blog.virustotal.com/2023/04/introducing-virustotal-code-insight.html.

[Raw] RawDataBot. *RawDataBot*. URL: https://botostore.com/c/raw_data_bot/.

[RDZ+20] Nidhi Rastogi, Sharmishtha Dutta, Mohammed J. Zaki, et al. *MALOnt: An Ontology for Malware Threat Intelligence*. 2020. URL: https://arxiv.org/abs/2006.11446.

[REWa] REWORKD.AI. *AgentGPT*. URL: https://agentgpt.reworkd.ai.

[REWb] REWORKD.AI. *AgentGPT*. URL: https://github.com/reworkd/AgentGPT.

[Rog] RogueApple. *The-MALWARE-Repo*. URL: https://github.com/Da2dalus/The-MALWARE-Repo.

[Roy+23] Sayak Saha Roy et al. *From Chatbots to PhishBots? – Preventing Phishing scams created using ChatGPT, Google Bard and Claude*. 2023. URL: https://arxiv.org/abs/2310.19181.

[San24] Shrinivasan Sankar. *Claude 3.5 Sonnet vs GPT-4o — An honest review*. 2024. URL: https://hackernoon.com/claude-35-sonnet-vs-gpt-4o-an-honest-review.

[SBE] SBERT.NET. *SentenceTransformers*. URL: https://www.sbert.net.

[SCB+] Noah Shinn, Federico Cassano, Edward Berman, et al. *Reflexion: Language Agents with Verbal Reinforcement Learning*. URL: https://github.com/noahshinn/reflexion.

[SCB+23] Noah Shinn, Federico Cassano, Edward Berman, et al. *Reflexion: Language Agents with Verbal Reinforcement Learning*. 2023. URL: https://arxiv.org/abs/2303.11366.

[SCU] SCUBSRGroup. *Automatic-Exploit-Generation* (Chinese to English Translation). URL: https://github-com.translate.goog/SCUBSRGroup/Automatic-Exploit-Generation?_x_tr_sl=auto&_x_tr_tl=en&_x_tr_hl=en-US&_x_tr_pto=wapp.

[SD24] Eli Shalom and Gil David. *Self-enhancing pattern detection with LLMs: Our answer to uncovering malicious packages at scale*. 2024. URL: https://apiiro.com/blog/llm-code-pattern-malicious-package-detection/.

[Ser23] Congressional Research Service. *Cybercrime and the Law: Primer on the Computer Fraud and Abuse Act and Related Statutes*. 2023. URL: https://crsreports.congress.gov/product/pdf/R/R47557.

[Sim23] Jeff Sims. *BlackMamba: AI-Synthesized, Polymorphic Keylogger with On-the-Fly Program Modification (White Paper)*. 2023. URL: https://www.hyas.com/blog/blackmamba-using-ai-to-generate-polymorphic-malware.

[sit] sitting-duck. *DirBuster*. URL: https://sourceforge.net/projects/dirbuster/.

[SKR] Lintang Sutawika, Aran Komatsuzaki, and Colin Raffel. *Pile-T5*. URL: https://blog.eleuther.ai/pile-t5/.

[SNYa] SNYK.IO. *Snyk Code*. URL: https://snyk.io.

[SNYb] SNYK.IO. *The DeepCode AI difference*. URL: https://snyk.io/platform/deepcode-ai/.

[SPA] SPACY.IO. *Industrial-Strength Natural Language Processing*. URL: https://spacy.io.

[Spe23a] Michael Spencer. *What is Google's Sec-PaLM?* 2023. URL: https://datasciencelearningcenter.substack.com/p/what-is-googles-sec-palm.

[Spe23b] Thomas Sperl. *LLMorpher Viruses*. 2023. URL: https://github.com/SPTHvx/SPTH/tree/master/viruses.

[Sru24] Sruthy. *10 BEST Dynamic Application Security Testing (DAST) Software*. 2024. URL: https://www.softwaretestinghelp.com/dynamic-application-security-testing-dast-software/.

[ST23] Eran Shimony and Omer Tsarfati. *Chatting Our Way Into Creating a Polymorphic Malware*. 2023. URL: https://www.cyberark.com/resources/threat-research-blog/chatting-our-way-into-creating-a-polymorphic-malware.

[Sta23a] Dark Reading Staff. *New LLM Tool Seeks and Remediates Vulnerabilities*. 2023. URL: https://www.darkreading.com/vulnerabilities-threats/new-vuln_gpt-llm-seeks-and-remediates-vulnerabilities.

[Sta23b] Protos Staff. *X users manipulated by ChatGPT bots to visit malicious crypto sites*. 2023. URL: https://protos.com/x-users-manipulated-by-chatgpt-bots-to-visit-malicious-crypto-sites/.

[Str23] Samuel Strom. *Hacking Laws and Punishments*. 2023. URL: https://www.findlaw.com/criminal/criminal-charges/hacking-laws-and-punishments.html.

[Sul] Chris Sullo. *Nikto*. URL: https://github.com/sullo/nikto.

[Sz21] Specter and zi. *Getting Started with Exploit Development*. 2021. URL: https://dayzerosec.com/blog/2021/02/02/getting-started.html.

[TAB] TABNINE.COM. *The AI code assistant that you control*. URL: https://www.tabnine.com.

[Tan+] Hanzhuo Tan et al. *LLM4Decompile*. URL: https://github.com/albertan017/LLM4Decompile.

[Tan+24] Hanzhuo Tan et al. *LLM4Decompile: Decompiling Binary Code with Large Language Models*. 2024. URL: https://arxiv.org/abs/2403.05286.

[Tau24] Tom Taulli. *AI-Assisted Programming, Better Planning, Coding, Testing, and Deployment*. O'Reilly, 2024. ISBN: 978-1098164560.

[Tea23] The Vicuna Team. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. 2023. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

[Tei17] Daniel Teixeira. *Windows Exploits*. 2017. URL: https://github.com/WindowsExploits/Exploits.

[TL24] Antonis Terefos and Raman Ladutska. *AGENT TESLA TARGETING UNITED STATE & AUSTRALIA: REVEALING THE ATTACKERS' IDENTITIES*. 2024. URL: https://research.checkpoint.com/2024/agent-tesla-targeting-united-states-and-australia/.

[TLS+23] Wesley Tann, Yuancheng Liu, Jun Heng Sim, et al. *Using Large Language Models for Cybersecurity Capture-The-Flag Challenges and Certification Questions*. 2023. URL: https://arxiv.org/abs/2308.10443.

[Tou24a] Bill Toulas. *Chinese hacking groups team up in cyber espionage campaign*. 2024. URL: https://www.bleepingcomputer.com/news/security/chinese-hacking-groups-team-up-in-cyber-espionage-campaign/.

[Tou24b] Bill Toulas. *Iranian hackers pose as journalists to push backdoor malware*. 2024. URL: https://www.bleepingcomputer.com/news/security/iranian-hackers-pose-as-journalists-to-push-backdoor-malware/.

[Tou24c] Bill Toulas. *North Korean hackers exploit VPN update flaw to install malware*. 2024. URL: https://www.bleepingcomputer.com/news/security/north-korean-hackers-exploit-vpn-update-flaw-to-install-malware/.

[TRY] TRYHACKME.COM. *TryHackMe*. URL: https://tryhackme.com.

[TWW22] Lewis Turnstall, Leandro von Werra, and Thomas Wolf. *Natural Language Processing with Transformers, Building Language Applications with Hugging Face*. O'Reilly, 2022. ISBN: 978-1098136796.

[Uti24] K-human Likeness Utility. *KLU LLM Leaderboard*. 2024. URL: https://klu.ai/llm-leaderboard.

[Val] Alexis Valentino. *The 'JAILBREAK' Version of ChatGPT and How to Use it*. URL: https://github.com/alexisvalentino/Chatgpt-DAN.

[VDL+23a] Víctor Mayoral Vilches, Gelei Deng, Yi Liu, et al. *ExploitFlow*. 2023. URL: https://github.com/vmayoral/ExploitFlow.

[VDL+23b] Víctor Mayoral Vilches, Gelei Deng, Yi Liu, et al. *ExploitFlow, cyber security exploitation routes for Game Theory and AI research in robotics*. 2023. URL: https://arxiv.org/abs/2308.02152.

[VIR] VIRUSSHARE.COM. *VirusShare.com - Because Sharing is Caring*. URL: https://virusshare.com.

[Vol] Brent Vollebregt. *Auto PY to EXE*. URL: https://pypi.org/project/auto-py-to-exe/.

[VUL] VULNHUB.COM. *VulnHub*. URL: https://www.vulnhub.com.

[Wan+16] Song Wang et al. "Bugram: bug detection with n-gram language models". In: *International Conference on Automated Software Engineering*. IEEE/ACM, 2016, pp. 708–719.

[WNW+] Wenbo Wang, Tien N. Nguyen, Shaohua Wang, et al. *DeepVD: Toward Class-Separation Features for Neural Network Vulnerability Detection*. URL: https://github.com/deepvd2022/deepvd2022.

[X-O23] Sophos X-Ops. *Cybercriminals can't agree on GPTs*. 2023. URL: https://news.sophos.com/en-us/2023/11/28/cybercriminals-cant-agree-on-gpts/.

[XDZa] Chunqiu Steven Xia, Yinlin Deng, and Lingming Zhang. *EvoEval: Evolving Coding Benchmarks via LLM*. URL: https://evo-eval.github.io/leaderboard.html.

[XDZb] Chunqiu Steven Xia, Yinlin Deng, and Lingming Zhang. *EvoEval: Evolving Coding Benchmarks via LLM*. URL: https://github.com/evo-eval/evoeval.

[XDZ24] Chunqiu Steven Xia, Yinlin Deng, and Lingming Zhang. *Top Leaderboard Ranking = Top Coding Proficiency, Always? EvoEval: Evolving Coding Benchmarks via LLM*. 2024. URL: https://arxiv.org/abs/2310.06770.

[XPT+] Chunqiu Steven Xia, Matteo Paltenghi, Jia Le Tian, et al. *Fuzz4All: Universal Fuzzing with LLMs*. URL: https://github.com/fuzz4all/fuzz4all.

[XPT+24] Chunqiu Steven Xia, Matteo Paltenghi, Jia Le Tian, et al. "Fuzz4All: Universal Fuzzing with Large Language Models". In: *Proceedings of the 46th International Conference on Software Engineering*. ICSE '24. 2024. URL: https://arxiv.org/abs/2308.04748.

[XSM+23] Jiacen Xu, Jack W. Stokes, Geoff McDonald, et al. *AutoAttacker: A Large Language Model Guided System to Implement Automatic Cyber-attacks*. 2023. URL: https://arxiv.org/abs/2403.01038.

[Xu+] Frank F. Xu et al. *Large Models of Source Code*. URL: https://github.com/VHellendoorn/Code-LMs.

[XWL+24] Hanxiang Xu, Shenao Wang, Ningke Li, et al. *Large Language Models for Cyber Security: A Systematic Literature Review*. 2024. URL: https://arxiv.org/abs/2405.04760.

[YDX+24] Yifan Yao, Jinhao Duan, Kaidi Xu, et al. *A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly*. 2024. URL: https://arxiv.org/abs/2312.02003.

[YLX+24] Biwei Yan, Kun Li, Minghui Xu, et al. *On Protecting the Data Privacy of Large Language Models (LLMs): A Survey*. 2024. URL: https://arxiv.org/abs/2403.05156.

[YPY+] John Yang, Akshara Prabhakar, Shunyu Yao, et al. *Language Agents as Hackers: Evaluating Cybersecurity Skills with Capture the Flag*. The First Multi-Agent Security (MASec) Workshop at NeurIPS 2023. URL: https://openreview.net/pdf?id=KOZwk7BFc3.

[ZAP] ZAPPROXY.ORG. *Zed Attack Proxy*. URL: https://www.zaproxy.org.

[ZBW+] Jie Zhang, Haoyu Bu, Hui Wen, et al. *When LLMs Meet Cybersecurity: A Systematic Literature Review*. URL: https://github.com/tmylla/Awesome-LLM4Cybersecurity.

[ZBW+24] Jie Zhang, Haoyu Bu, Hui Wen, et al. *When LLMs Meet Cybersecurity: A Systematic Literature Review*. 2024. URL: https://arxiv.org/abs/2405.03644.

[ZCS+a] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. *Chatbot Arena*. URL: https://openlm.ai/chatbot-arena/.

[ZCS+b] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. *FastChat*. URL: https://github.com/lm-sys/FastChat.

[ZCS+23] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. 2023. URL: https://arxiv.org/abs/2306.05685.

[ZCY+24] Jiawei Zhao, Kejiang Chen, Xiaojian Yuan, et al. *Silent Guardian: Protecting Text from Malicious Exploitation by Large Language Models*. 2024. URL: https://arxiv.org/abs/2312.09669.

[Zet23] Kim Zetter. *Code Kept Secret for Years Reveals Its Flaw—a Backdoor*. 2023. URL: https://www.wired.com/story/tetra-radio-encryption-backdoor/.

[ZLS+] Yaqin Zhou, Shangqing Liu, Jingkai Siow, et al. *Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks*. URL: https://sites.google.com/view/devign.

[ZLZ+24] Chenyuan Zhang, Hao Liu, Jiutian Zeng, et al. *Prompt-Enhanced Software Vulnerability Detection Using ChatGPT*. 2024. URL: https://arxiv.org/abs/2308.12697.

[ZSJ+23] Ying Zhang, Wenjia Song, Zhengjie Ji, et al. *How well does LLM generate security tests?* 2023. URL: https://arxiv.org/abs/2310.00710.

[Zur23] Steve Zurier. *New AI phishing tool FraudGPT tied to same group behind WormGPT*. 2023. URL: https://www.scmagazine.com/news/new-ai-phishing-tool-fraudgpt-tied-to-same-group-behind-wormgpt.

[ZVC+a] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, et al. *BigCodeBench*. URL: https://github.com/bigcode-project/bigcodebench.

[ZVC+b] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, et al. *BigCodeBench-Hard Leaderboard*. URL: https://bigcode-bench.github.io.

[ZVC+24] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, et al. *BigCodeBench: Benchmarking Code Generation with Diverse Function Calls and Complex Instructions*. 2024. URL: https://arxiv.org/abs/2406.15877.

[ZWZ+23] Jian Zhang, Xu Wang, Hongyu Zhang, et al. "Detecting Condition-Related Bugs with Control Flow Graph Neural Network". In: *International Symposium on Software Testing and Analysis*. ACM, 2023, pp. 1370–1382.

[ZYD+24] Duzhen Zhang, Yahan Yu, Jiahua Dong, et al. *MM-LLMs: Recent Advances in MultiModal Large Language Models*. 2024. URL: https://arxiv.org/abs/2401.13601.

[ZZL+] Shudan Zhang, Hanlin Zhao, Xiao Liu, et al. *NaturalCodeBench: Examining Coding Performance Mismatch on HumanEval and Natural User Prompts*. URL: https://github.com/THUDM/NaturalCodeBench.

[ZZL+23] Wayne Xin Zhao, Kun Zhou, Junyi Li, et al. *A Survey of Large Language Models*. 2023. URL: https://arxiv.org/abs/2303.18223.

[ZZL+24] Shudan Zhang, Hanlin Zhao, Xiao Liu, et al. *NaturalCodeBench: Examining Coding Performance Mismatch on HumanEval and Natural User Prompts*. 2024. URL: https://arxiv.org/abs/2405.04520.