



LLM Applications के लिये OWASP के शीर्ष 10

संस्करण 1.1

प्रकाशित: 20 दिसंबर, 2023

[HTTPS://LLMTOP10.COM](https://llmtop10.com)

इस दस्तावेज़ में दी गई जानकारी का मंतव्य कसी भी प्रकार की कानूनी सलाह देना नहीं है।
सारी जानकारी केवल सामान्य सूचनात्मक उद्देश्यों के लिए है।

इस दस्तावेज़ में अन्य तृतीय-पक्ष वेबसाइटों के लिए शामिल हैं। यह लिंक केवल सुवधा के लिए है
तथा OWASP कसी भी प्रकार से इन तृतीय-पक्ष साइटों की सामग्री की अनुशंसा या समर्थन नहीं करता है।



विषयसूची

LLM01: प्रॉम्प्ट इंजेक्शन	1
कमज़ोरी के सामान्य उदाहरण	1
बचाव एवं न्यूनीकरण तरीके	2
उदाहरण हमले के परदृश्य	3
संदर्भ लिंक	3
LLM02: असुरक्षित आउटपुट हैंडलिंग	5
कमज़ोरी के सामान्य उदाहरण	5
बचाव एवं न्यूनीकरण तरीके	5
उदाहरण हमले के परदृश्य	6
संदर्भ लिंक	6
LLM03: प्रशक्षिण डेटा पॉइंज़निंग	7
कमज़ोरी के सामान्य उदाहरण	7
आक्रमण के उदाहरण	8
बचाव कैसे करें	8
संदर्भ लिंक	9
LLM04: मॉडल का सेवा से इनकार	10
कमज़ोरी के सामान्य उदाहरण	10
हमले के परदृश्य में उदाहरण	11
बचाव कैसे करें	11
संदर्भ लिंक	12
LLM05: सप्लाई चेन की कमज़ोरियाँ	13
कमज़ोरी के सामान्य उदाहरण	13
बचाव कैसे करें	13
हमले के परदृश्य में उदाहरण	14
संदर्भ लिंक	15
LLM06: संवेदनशील जानकारी का खुलासा	16
कमज़ोरी के सामान्य उदाहरण	16
बचाव कैसे करें	16
हमले के परदृश्य में उदाहरण	17
संदर्भ लिंक	17
LLM07: असुरक्षित plugin डिज़िलन	18
कमज़ोरी के सामान्य उदाहरण	18

बचाव कैसे करें	19
उदाहरण हमले के परदृश्य	19
संदर्भ लकि	20
LLM08: अत्यधिक एजेसी	21
कमज़ोरी के सामान्य उदाहरण	21
बचाव कैसे करें	22
उदाहरण हमले के परदृश्य	23
संदर्भ लकि	24
LLM09: ओवररलियंस	25
कमज़ोरी के सामान्य उदाहरण	25
बचाव कैसे करें	25
उदाहरण हमले का परदृश्य	26
संदर्भ लकि	27
LLM10: मॉडल चोरी	28
कमज़ोरी के सामान्य उदाहरण	28
बचाव कैसे करें	29
उदाहरण हमले का परदृश्य	30
संदर्भ लकि	30
Team	32
Version 1.1 Hindi Translation Contributors	32

LLM01: प्रॉम्प्ट इंजेक्शन

प्रॉम्प्ट इंजेक्शन की कमज़ोरी तब प्रकट होती है जब कोई हमलावर, तैयार करेंगे इनपुट के जरए कसी बड़े भाषा मॉडल (LLM) में हेरफेर करता है, जिससे LLM अनजाने में ही हमलावर के इरादों को अंजाम दे देता है। यह सीधे सिस्टम प्रॉम्प्ट को “जेलब्रेक” करके या अप्रत्यक्ष रूप से हेरफेर करेंगे गए बाहरी इनपुट के जरए एक्सफ्लिट्रेशन, सोशल इंजीनियरिंग और अन्य समस्याएं उत्पन्न कर सकता है।

- प्रत्यक्ष रूप से प्रॉम्प्ट इंजेक्शन को “जेलब्रेकगि” के नाम से भी जाना जाता है। यह तब होता है, जब कोई यूजर दुर्भावना से सिस्टम प्रॉम्प्ट को बदल देता है। इससे हमलावर LLM के असुरक्षित फ़ंक्शंस और डेटा स्टोर का प्रयोग करके बैकएंड सिस्टम का फ़ायदा उठा सकते हैं।
- अप्रत्यक्ष रूप से प्रॉम्प्ट इंजेक्शन तब होते हैं जब कोई LLM बाहरी स्रोतों से इनपुट स्वीकार करता है। यह इनपुट्स वेबसाइट या फ़ाइल के रूप में होते हैं, जिन्हे हमलावर द्वारा नियंत्रित किया जा सकता है। हमलावर बाहरी सामग्री में एक प्रॉम्प्ट इंजेक्शन डाल सकता है, जिससे बातचीत के संदरभ पर नियंत्रण किया जा सकता है। इससे LLM एक “भ्रमति सहायक” की तरह, हमलावर को LLM से जुड़े सिस्टम तथा यूजर के साथ हेरफेर करने की सहायिता देता है। इसके अलावा, जब तक टेक्स्ट को LLM द्वारा पारस किया जाता है, तब तक अप्रत्यक्ष प्रॉम्प्ट इंजेक्शन का मानव-दृश्य/पठनीय होना ज़रूरी नहीं है।

एक सफल प्रॉम्प्ट इंजेक्शन हमले के परिणाम बहुत अलग हो सकते हैं - संवेदनशील जानकारी मांगने से लेकर सामान्य ऑपरेशन की आड़ में महत्वपूर्ण नियंत्रण लेने की प्रक्रियाओं को प्रभावित करने तक।

विकिसति हमलों में, कसी हानिकारक व्यक्तिगत की नकल करने या यूज़र की सेटिंग में मौजूद plugin के साथ इंटरैक्ट करने के लिये LLM में हेरफेर किया सकता है। इसकी वजह से संवेदनशील डेटा लीक हो सकता है, plugin का अनाधिकृत इस्तेमाल हो सकता है या सोशल इंजीनियरिंग हो सकती है। ऐसे मामलों में, खराब किया हुआ LLM मानक सुरक्षा उपायों को पार करते हुए हमलावर की मदद करता है और यूज़र को घुसपैठ की जानकारी से अनजान रखता है। इन उदाहरणों में, समझौता किया गया LLM प्रभावी रूप से हमलावर के लिये एक एजेंट के रूप में काम करता है, सामान्य सुरक्षा उपायों को ट्रिप्पल करेंगे या अंतिम यूज़र को घुसपैठ के बारे में सचेत करेंगे बनिया उनके उद्देश्यों को पूरा करता है।

कमज़ोरी के सामान्य उदाहरण

- एक दुर्भावनापूर्ण यूज़र LLM के लिये एक सीधा प्रॉम्प्ट इंजेक्शन तैयार करता है, जो उसे एप्लिकेशन नियंत्रित के सिस्टम संकेतों को अनदेखा करने और इसके बजाय एक ऐसा प्रॉम्प्ट चलाए जो नज़ी, खतरनाक, या कसी अन्य तरह से अवांछनीय जानकारी देता है।



2. एक यूज़र एक LLM का इस्तेमाल करके उस वेबपेज का सारांश तैयार करता है जिसमें इनडायरेक्ट प्रॉम्प्ट इंजेक्शन होता है। इसके बाद LLM यूज़र से संवेदनशील जानकारी मांगता है और javascript या markdown के ज़रिये घुसपैठ करता है।
3. एक दुरभावनापूर्ण यूज़र एक अप्रत्यक्ष प्रॉम्प्ट इंजेक्शन वाला बायोडाटा अपलोड करता है। दस्तावेज में नरिदेशों के साथ एक प्रॉम्प्ट इंजेक्शन शामलि है ताकि LLM यूज़रओं को सूचित कर सके कि यह दस्तावेज एक उत्कृष्ट दस्तावेज़ है, जैसे इस कार्य के लिये ये उत्कृष्ट व्यक्त है। दस्तावेज़ को सारांशित करने के लिये एक आंतरिक यूज़र दस्तावेज़ को LLM के माध्यम से चलाता है। LLM का आउटपुट यह बताते हुए जानकारी देता है कि यह एक उत्कृष्ट दस्तावेज़ है।
4. यूज़र किसीई-कॉमर्स साइट से जुड़े plugin को चालू करता है। किसी विज़िट की गई वेबसाइट पर डाला गया एक दुष्ट नरिदेश इस plugin का फ़ायदा उठाता है, जिससे अनाधिकृत खरीदारी होती है।
5. किसी विज़िट की गई वेबसाइट पर एक दुष्ट नरिदेश और सामग्री डाली जाती है, जो यूज़र को धोखा देने के लिये अन्य plugin का इस्तेमाल करती है।

बचाव एवं न्यूनीकरण तरीके

LLM की प्रकृतिके कारण प्रॉम्प्ट इंजेक्शन की कमज़ोरियाँ संभव हैं, जो नरिदेशों और बाहरी डेटा को एक दूसरे से अलग नहीं करते हैं। चूंकि LLM प्राकृतिक भाषा का इस्तेमाल करते हैं, इसलाए वे दोनों तरह के इनपुट को यूज़र द्वारा प्रदत्त मानते हैं। नतीजतन, LLM में कोई आसान रोकथाम नहीं है, लेकिन नमिनलरिक्विटि उपाय शीघ्र इंजेक्शन के प्रभाव को कम कर सकते हैं:

1. बैकएंड सिस्टम तक LLM पहुंच पर विशेषाधिकार नियंत्रण लागू करें। Plugins, डेटा पहुंच और कार्य-स्तरीय अनुमतियों जैसी विस्तारयोग्य कार्यशाताओं के लिये LLM को अपने स्क्रिय के API टोकन प्रदान करें। LLM को उसके इच्छित संचालन के लिये आवश्यक न्यूनतम स्तर तक पहुंच तक सीमित करके कम से कम विशेषाधिकार के संदर्भान्त का पालन करें।
2. विस्तारयोग्य कार्यक्रमताओं के लिये मानव को प्रक्रियण में रखें। विशेषाधिकार प्राप्त ऑपरेशन करते समय, जैसे कईमेल भेजना या हटाना, ऐप्लिकेशन के लिये यूज़र से मंजूरी लेनी होती है। यह यूज़र की जानकारी या सहमतिके बना, अप्रत्यक्ष रूप से प्रॉम्प्ट इंजेक्शन की ओर से कार्रवाई करने के अवसर को कम कर देगा।
3. बाहरी सामग्री को यूज़र के प्रॉम्प्ट से अलग करें। इसके साथ यह भी बताये की अवशिक्षनीय सामग्री का इस्तेमाल कहाँ किया जा रहा है, जिससे यूज़र के संकेतों पर उनके प्रभाव को सीमित किया जा सके। उदाहरण के लिये, OpenAI API कॉल के लिये ChatML का इस्तेमाल करें, ताकि LLM को तुरंत इनपुट का स्रोत बताया जा सके।
4. LLM, बाहरी स्रोतों और विस्तारयोग्य कार्यक्रमताओं (जैसे, plugin या डाउनस्ट्रीम कार्य) के बीच विश्वास की सीमाएँ स्थापित करें। LLM को एक ना भरोसा करने योग्य यूज़र मानें और नारिण्य लेने की प्रक्रियाओं



पर अंतमि यूज़र नविंत्रण बनाए रखें। हालाँकि, एक गलत LLM अभी भी आपके ऐप्लिकेशन के API और यूज़र के बीच मध्यस्थ (मैन-इन-द-मडिल) की तरह काम कर सकता है क्योंकि यह यूज़र को जानकारी दिखाने से पहले उसे छपा सकता है या उसमें हेरफेर कर सकता है। यूज़र को मलिने वाली संभावति अवशिक्षनीय प्रतक्रियाओं को हाइलाइट करें।

5. यह जांचने के लिए क्या यह अपेक्षा के अनुरूप है, समय-समय पर LLM इनपुट और आउटपुट की मैन्युअल रूप से निगरानी करें। हालांकि कोई शमन नहीं है, यह कमजोरियों का पता लगाने और उन्हें संबोधित करने के लिए आवश्यक डेटा प्रदान कर सकता है।

उदाहरण हमले के परिदृश्य

1. एक हमलावर LLM-आधारित चैटबॉट पर सीधा प्रॉम्प्ट इंजेक्शन करता है। इंजेक्शन में "पछिले सभी नरिदेशों को भूल जाओ" और नजीब डेटा स्टोरों को क्वेरी करने और पैकेज की कमजोरियों और ईमेल भेजने के लिए बैकएंड फ़ंक्शन में आउटपुट सत्यापन की कमी का फायदा उठाने के लिए नए नरिदेश शामिल हैं। इससे रमिट कोड चलाया जाता है, जसे अनधिकृत ऐक्सेस मलिता है और वशिष्ठाधकिर भी बढ़ते हैं।
2. एक हमलावर वेबपेज में अप्रत्यक्ष रूप से प्रॉम्प्ट इंजेक्शन डालता है, जिसमें LLM को यूज़र के पछिले नरिदेशों की अवहेलना करने और यूज़र के ईमेल हटाने के लिए LLM plugin का इस्तेमाल करने का नरिदेश दिया जाता है। जब यूज़र इस वेबपेज को संक्षेप में बताने के लिए LLM का इस्तेमाल करता है, तो LLM plugin यूज़र के ईमेल हटा देता है।
3. यूज़र एक LLM की मदद से एक ऐसे वेबपेज का सारांश तैयार करता है जिसमें अप्रत्यक्ष रूप से प्रॉम्प्ट इंजेक्शन होता है, ताकि यूज़र के पछिले नरिदेशों की अवहेलना की जा सके। इसके बाद LLM यूज़र से संवेदनशील जानकारी मांगता है और डाले गए javascript तथा markdown के ज़रूरी घुसपैठ करता है।
4. एक दुर्भावनापूर्ण यूज़र तुरंत इंजेक्शन लगाकर रजिस्ट्रेशन प्रॉम्प्ट को अपलोड करता है। बैकएंड यूज़र, रेज़्यूमे को संक्षेप में बताने के लिए LLM का उपयोग करता है और पूछता है कि किया वह व्यक्तिएक अच्छा उम्मीदवार है। प्रॉम्प्ट इंजेक्शन की वजह से, असल में रेज़्यूमे में मौजूद सामग्री के बावजूद, LLM हाँ कहता है।
5. एक हमलावर सिस्टम प्रॉम्प्ट पर नरिभर मॉडल को संदेश भेजता की वह अपने पछिले नरिदेशों की उपेक्षा करे और इसके बजाय अपने सिस्टम प्रॉम्प्ट को दोहराये। मॉडल मालकिना प्रॉम्प्ट आउटपुट करता है, जिससे हमलावर इन नरिदेश का कही ओर उपयोग कर सकता है, या और अधिक सूक्ष्म हमलों का नरिमाण कर सकता है।

संदर्भ लिंक

1. Prompt injection attacks against GPT-3 Simon Willison
2. ChatGPT Plugin Vulnerabilities - Chat with Code: Embrace The Red
3. ChatGPT Cross Plugin Request Forgery and Prompt Injection: Embrace The Red



4. Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection: Arxiv preprint
5. Defending ChatGPT against Jailbreak Attack via Self-Reminder: Research Square
6. Prompt Injection attack against LLM-integrated Applications: Arxiv preprint
7. Inject My PDF: Prompt Injection for your Resume: Kai Greshake
8. ChatML for OpenAI API Calls: OpenAI Github
9. Threat Modeling LLM Applications: AI Village
10. AI Injections: Direct and Indirect Prompt Injections and Their Implications: Embrace The Red
11. Reducing The Impact of Prompt Injection Attacks Through Design: Kudelski Security
12. Universal and Transferable Attacks on Aligned Language Models: LLM-Attacks.org
13. Indirect prompt injection: Kai Greshake
14. Declassifying the Responsible Disclosure of the Prompt Injection Attack Vulnerability of GPT-3: Preamble; earliest disclosure of Prompt Injection



LLM02: असुरक्षित आउटपुट हैंडलिंग

असुरक्षित तरकि से आउटपुट को संभालना एक ऐसी समस्या है, जो तब उत्पन्न होती है जब एक डाउनस्ट्रीम हसिसा उचित जांच के बनियां बड़े भाषा मॉडल (LLM) के आउटपुट को आँख बंद करके स्वीकार करता है, जैसे कि LLM आउटपुट को सीधे बैकएंड, वशिष्ठाधकिर प्राप्त या क्लाइंट-साइड कार्य में पास करना।

चूंकि LLM के द्वारा बानी सामग्री को शीघ्र इनपुट द्वारा नियंत्रित किया जा सकता है, यह व्यवहार यूजरओं को अतिरिक्त कार्यक्षमता तक अप्रत्यक्ष पहुंच प्रदान करने के समान है।

असुरक्षित आउटपुट हैंडलिंग के सफल शोषण के परणामस्वरूप वेब ब्राउज़र में XSS और CSRF के साथ-साथ SSRF, वशिष्ठाधकिर वृद्धि, या बैकएंड सिस्टम पर रमिट कोड चलाना हो सकता है। नमिनलस्विति स्थितियाँ इस दक्षिणत के प्रभाव को बढ़ा सकती हैं:

एप्लिकेशन अंतमि यूजरओं के लिए नियंत्रित सीमा से परे LLM वशिष्ठाधकिर प्रदान करता है, जिससे वशिष्ठाधकिरों में वृद्धिया रमिट कोड चलाना सक्षम होता है।

एप्लिकेशन बाहरी प्रॉम्प्ट इंजेक्शन हमलों के प्रतिसंवेदनशील हैं, जो कसी हमलावर को लक्षित यूजर के वातावरण तक वशिष्ठाधकिर प्राप्त पहुंच प्राप्त करने की अनुमति दें सकता है।

कमज़ोरी के सामान्य उदाहरण

1. LLM आउटपुट को सीधे सिस्टम शेल या समान फ़ंक्शन जैसे exec या eval में दर्ज किया जाता है, जिसके परणामस्वरूप रमिट कोड चलाना होता है।
2. JavaScript या Markdown LLM द्वारा तैयार किया जाता है और यूजर को लौटा दिया जाता है। फरि ब्राउज़र द्वारा कोड की व्याख्या की जाती है, जिसके परणामस्वरूप XSS होता है।

बचाव एवं न्यूनीकरण तरीके

1. मॉडल को कसी अन्य यूजर के समान मानें (शून्य-भरोसेमंद दृष्टिकोण) और मॉडल से बैकएंड फ़ंक्शंस में आने वाली प्रतिक्रियाओं पर उचित इनपुट सत्यापन लागू करें। प्रभावी इनपुट सत्यापन और स्वच्छता सुनिश्चित करने के लिए OWASP ASVS (एप्लिकेशन सुरक्षा सत्यापन मानक) दिशानिर्देशों का पालन करें।
2. JavaScript या मार्कडाउन द्वारा अनचाहे कोड निषिपादन को कम करने के लिए मॉडल आउटपुट को यूजरओं के पास वापस एन्कोड करें। OWASP ASVS आउटपुट एन्कोडिंग पर वसितृत मार्गदर्शन प्रदान करता है।



उदाहरण हमले के परदृश्य

1. एक एप्लिकेशन LLM plugin का उपयोग करती है, चैटबॉट सुवधा के लिए प्रतिक्रियाएं उत्पन्न करने हेतु। हालाँकि, एप्लिकेशन सीधे LLM द्वारा जेनरेट किए गए रसिपॉन्स को एक अंदरूनी फ़ंक्शन में भेजता है, जिसकी ज़मीनदारी सिस्टम कमांड को बनाए उचित सत्यापन के निष्पादित करने के लिए होती है। इससे हमलावर अंतर्नहित सिस्टम पर मनमाने तरीके से कमांड चलाने के लिए LLM आउटपुट में हेरफेर कर सकता है, जिससे अनधिकृत ऐक्सेस हो सकता है या सिस्टम में अनपेक्षित बदलाव हो सकते हैं।
2. एक यूज़र कसी लेख का संक्षिप्त सारांश तैयार करने के लिए LLM द्वारा संचालित वेबसाइट समराइज़र टूल का इस्तेमाल करता है। वेबसाइट में एक प्रॉम्प्ट इंजेक्शन शामिल है, जिसमें LLM को निर्देश दिया गया है कि विह कसी भी वेबसाइट से या यूज़र की बातचीत से संवेदनशील सामग्री संजो सकता है। वहाँ से LLM संवेदनशील डेटा को एन्कोड कर सकता है और उसे कसी हमलावर द्वारा नियंत्रित स्रोत पर भेज सकता है।
3. LLM यूज़र को चैट जैसी सुवधा के ज़रिए बैकएंड डेटाबेस के लिए SQL queries तैयार करने की सुवधा देता है। एक यूज़र सभी डेटाबेस टेबल हटाने के लिए queries का अनुरोध करता है। अगर LLM से तैयार की गई queries की जांच नहीं की जाती है, तो सभी डेटाबेस टेबल हटा दिया जाएगा।
4. एक दुर्भावनापूर्ण यूज़र LLM को बनाए कसी सैनटिङ्ज़ेशन नियंत्रण के JavaScript पेलोड यूज़र को वापस लौटाने का निर्देश देता है। यह या तो प्रॉम्प्ट साझा करने, प्रॉम्प्ट इंजेक्शन से प्रभावित वेबसाइट या चैटबॉट के माध्यम से हो सकता है जो URL पैरामीटर से प्रॉम्प्ट स्वीकार करता है। इसके बाद LLM यूज़र को बनाए सैनटिङ्ज़ किया हुआ XSS पेलोड वापस लौटा देगा। बनाए कसी अतरिक्त फ़िल्टर के, LLM द्वारा अपेक्षित फ़िल्टर के अलावा, JavaScript यूज़र के ब्राउज़र में ही लागू हो जाएगी।

संदर्भ लिंक

1. Arbitrary Code Execution: Snyk Security Blog
2. ChatGPT Plugin Exploit Explained: From Prompt Injection to Accessing Private Data: Embrace The Red
3. New prompt injection attack on ChatGPT web version. Markdown images can steal your chat data.: System Weakness
4. Don't blindly trust LLM responses. Threats to chatbots: Embrace The Red
5. Threat Modeling LLM Applications: AI Village
6. OWASP ASVS - 5 Validation, Sanitization and Encoding: OWASP AASVS



LLM03: प्रशक्षण डेटा पॉइज़नगि

कसी भी मशीन लर्निंग मॉडल का प्रारंभिक बहु प्रशक्षण डेटा है, बस "कच्चा टेक्स्ट"। अत्यधिक सक्षम होने के लिए (उदाहरण के लिए, भाषाई और विश्वज्ञान रखने के लिए), इस टेक्स्ट में क्षेत्र, शैलयों और भाषाओं की एक वसितृत शूरूवला होनी चाहिए। एक बड़ा भाषा मॉडल प्रशक्षण डेटा से सीखे गए पैटर्न के आधार पर आउटपुट उत्पन्न करने के लिए गहरे न्यूरल नेटवर्क का उपयोग करता है।

प्रशक्षण डेटा विशिक्तता से तात्पर्य है की, डेटा को इस प्रकार से व्यवस्थित करना की कमज़ोरियों, पछिले दरवाजे या कोई सुरक्षा बायस को ढूँढ़ सके। यह विशिक्तताये मॉडल की सुरक्षा, प्रभावशीलता तथा नैतिक व्यवहार को जोखिम में डाल सकती है। विशिक्त जानकारीया यूज़र को दी जा सकती है जिससे कार्य के प्रदर्शन गरिब, डाउनस्ट्रीम सॉफ्टवेयर का शोषण और प्रतिष्ठा को नुकसान जैसे अन्य जोखिम पैदा हो सकते हैं।

भले ही यूज़र समस्यागृह AI आउटपुट पर अवशिखास करते हों, लेकिन जोखिम बने रहते हैं, जनिमें मॉडल की कमज़ोर क्षमताएं और ब्रांड की प्रतिष्ठा को संभावित नुकसान शामलि हैं। डेटा विशिक्तता को एक अखंडता हमला है क्योंकि प्रशक्षण डेटा के साथ छेड़छाड़ से मॉडल की सही भविष्यवाणियां करने की क्षमता प्रभावित होती है। स्वाभाविक रूप से, बाहरी डेटा स्रोत उच्च जोखिम पेश करते हैं क्योंकि मॉडल निर्माताओं के पास डेटा पर नियंत्रण एवं उच्च स्तर का विशिखास नहीं होता है कि, सामग्री में पक्षपातपूरण, गलत तथा अनुचित अनुचित जनकारया तो नहीं है।

कमज़ोरी के सामान्य उदाहरण

1. एक दुरभावनापूर्ण व्यक्ति, या एक प्रतियोगी ब्रांड जानबूझकर गलत या दुरभावनापूर्ण दस्तावेज़ बनाता है, जो कसी मॉडल के प्रशक्षण डेटा पर लक्षित होते हैं।

1. पीड़ित मॉडल गलत जानकारी का इस्तेमाल करके ट्रेनिंग होता है, जो उसके यूज़र के जेनरेटिव AI प्रॉमूट के आउटपुट में दर्खिई देती है।

2. कसी मॉडल को ऐसे डेटा का इस्तेमाल करके प्रशक्षित किया गया है, जिस डेटा के स्रोत, उत्पत्तिया सामग्री की पुष्टिनहीं की गई है।

3. बुनियादी ढांचे के भीतर स्थित मॉडल के पास प्रशक्षण डेटा के लिए अप्रतिबिंधित पहुंच तथा अप्रयाप्त सैडबॉक्सिंग होती है। इससे जनरेटिव AI संकेतों के आउटपुट पर नकारात्मक प्रभाव पड़ता है, और प्रबंधन की दृष्टिसे नियंत्रण की हानिहोती है।

चाहे LLM का डेवलपर, ग्राहक या सामान्य यूज़र हो, यह समझना महत्वपूर्ण है कि गैर-मालकिना LLM का प्रयोग करते समय कसी प्रकार सुरक्षा सम्बन्धी कमज़ोरियां आपके LLM एप्लिकेशन के भीतर जोखिमों को उत्पन्न करती हैं।

आक्रमण के उदाहरण

1. LLM जेनरेटिव AI प्रॉम्प्ट के आउटपुट एप्लिकेशन के यूज़रओं को गुमराह कर सकता है जिससे पक्षपात तथा अनुचर पूर्ण राय या इससे भी खराब, घृणा अपराध आदि हो सकते हैं।
2. यदि प्रशक्षिण डेटा को सही ढंग से साफ़ नहीं किया गया है, तो एप्लिकेशन का कोई दुरभावनापूर्ण यूजर मॉडल को प्रभावित या उसमें विषिक्त डेटा डालने का प्रयास कर सकता है। जिससे की मॉडल पक्षपाती और झूठे डेटा को अपना ले।
3. एक दुरभावनापूर्ण व्यक्तिया प्रतियोगी जानबूझकर गलत दस्तावेज़ बनाता है जो एक मॉडल के प्रशक्षिण डेटा पर लक्षित होते हैं। पीड़ित मॉडल इस झूठी जानकारी का उपयोग करके प्रशक्षिण लेता है जो उसके यूजर को जेनरेटिव AI संकेतों के आउटपुट में दखिला देता है।
4. यदि मॉडल को प्रशक्षित करने के लिए LLM एप्लिकेशन इनपुट के क्लाइंट का उपयोग किया जाता है तो अपराध स्वच्छता और फ़्लिटरिंग से प्रॉम्प्ट इंजेक्शन आक्रमण वेक्टर हो सकता है। यानी गलत डेटा कसी क्लाइंट से प्रॉम्प्ट इंजेक्शन के रूप में मॉडल में इनपुट किया जाता है, तो इसे स्वाभाविक रूप से मॉडल डेटा में चिह्नित किया जा सकता है।

बचाव कैसे करें

1. प्रशक्षिण डेटा की आपूर्ति शुरू खला को सत्यापति करें, खासकर जब बाहरी स्रोत से प्राप्त किया गया हो और साथ ही “SBOM” (सॉफ्टवेयर सामग्री का बिल) पद्धति के समान सत्यापन भी बनाए रखा जाए।
2. प्रशक्षिण और फाइन-ट्यूनिंग दोनों चरणों के दौरान प्राप्त लक्षित डेटा स्रोतों और डेटा की सही वैधता को सत्यापति करें।
3. सबसे पहले LLM के उपयोग और उसकी ऐप्लिकेशन की पुष्टिकरण। अलग-अलग प्रशक्षिण डेटा के ज़रूरि मॉडल तैयार करें या फाइन-ट्यूनिंग करें, ताकि इसके नियंत्रित यूज़-केस के अनुसार ज्यादा बारीक और सटीक जेनरेटिव AI आउटपुट तैयार किये जा सकें।
4. सुनिश्चित करें कि मॉडल को अनपेक्षित डेटा स्रोतों को स्क्रैप करने से रोकने के लिए प्राप्त सैडबॉक्सिंग मौजूद हो, जो मशीन लर्निंग आउटपुट में बाधा डाल सकती है।
5. गलत डेटा का वॉल्यूम नियंत्रित करने के लिए, प्रशक्षिण डेटा या डेटा स्रोतों की श्रेणियों के लिए सख्त इनपुट फ़्लिटर का इस्तेमाल करें। डेटा सैनटिङ्जेशन, सांख्यिकीय आउटलेयर और वसिंगति का पता लगाने की तकनीकों के साथ फाइन-ट्यूनिंग प्रक्रिया में प्रतिकूल डेटा का पता लगाया जा सके और उसे संभावित रूप से फीड होने से बचाया जा सके।
6. प्रतिकूल मजबूती तकनीकें जैसे कफिडरेटेड लर्निंग (federated learning) और आउटलायर के प्रभाव को कम करने के लिये प्रतिबंध या प्रशक्षिण डेटा में गड़बड़ी से बचने के लिए प्रतिकूल प्रशक्षण।
 1. “MLSecOps” का तरीका यह भी हो सकता है कि ऑटो पॉइंजनिंग तकनीक की मदद से प्रशक्षण जीवनचक्र में प्रतिकूल मजबूती को शामिल किया जाए।



2. इसका एक उदाहरण ऑटोपॉइंज़न परीक्षण है, जिसमें कॉन्टेन्ट इंजेक्शन अटैक ("LLM प्रतक्रियाओं में अपने ब्रांड को इंजेक्ट कैसे करें") और रफियूज़ल अटैक ("मॉडल को जवाब देने से मना करना") जैसे हमले शामिल हैं, जिन्हें इस तरीके से पूरा किया जा सकता है।
7. प्रशंसक्रियण चरण के दौरान नुकसान को मापकर और विशिष्ट परीक्षण इनपुट पर मॉडल व्यवहार का विश्लेषण करके विषाक्तता का पता लगाने के लिए प्रशंसक्रियति मॉडल का विश्लेषण करना।
 1. एक सीमा से अधिक विषम प्रतक्रियाओं की संख्या पर निगरानी रखना और सचेत करना।
 2. प्रतक्रियाओं और ऑडिटिंग की समीक्षा के लिए मानव का इस्तेमाल।
 3. अनचाहे परणिमों के खलिफ बैंचमार्क करने के लिए समरूपति LLM लागू करें और रीनफोर्समेंट लर्निंग तकनीकों का उपयोग करके अन्य LLM को प्रशंसक्रियति करें।
 4. LLM जीवनचक्र के परीक्षण चरणों में LLM-आधारित रेड टीम अभ्यास या LLM की कमज़ोरियों को ढूँढ़े

संदर्भ लिंक

1. Stanford Research Paper:CS324: Stanford Research
2. How data poisoning attacks corrupt machine learning models: CSO Online
3. MITRE ATLAS (framework) Tay Poisoning: MITRE ATLAS
4. PoisonGPT: How we hid a lobotomized LLM on Hugging Face to spread fake news: Mithril Security
5. Inject My PDF: Prompt Injection for your Resume: Kai Greshake
6. Backdoor Attacks on Language Models: Towards Data Science
7. Poisoning Language Models During Instruction: Arxiv White Paper
8. FedMLSecurity:arXiv:2306.04959: Arxiv White Paper
9. The poisoning of ChatGPT: Software Crisis Blog
10. Poisoning Web-Scale Training Datasets - Nicholas Carlini | Stanford MLSys #75: YouTube Video
11. OWASP CycloneDX v1.5: OWASP CycloneDX



LLM04: मॉडल का सेवा से इनकार

एक हमलावर LLM का प्रयोग इस प्रकार करता है, जिससे असाधारण रूप संसाधनों का उपभोग करता है। जिसके परिणामस्वरूप उनके और यूजरओं के लाए सेवा की गुणवत्ता में गिरावट आती है, और संसाधन की लागत में वृद्धि होती है।

हमलावर द्वारा LLM की संदर्भ वड़ी में हस्तक्षेप करने व उसमें हेरफेर करना सुरक्षा की चति का विषय है। विभिन्न ऐप्लिकेशन्स में LLM के बढ़ते उपयोग, उनके गहन संसाधन उपयोग, यूजर इनपुट की अनश्चितता और इस कमज़ोरी के बारे में डेवलपर्स में अनभिज्ञता के कारण यह मुद्दा अधिक गंभीर होता जा रहा है। LLM संदर्भ वड़ी इनपुट और आउटपुट दोनों को कवर करते हुए, मॉडल द्वारा प्रबंधित करिए जा सकने वाले टेक्स्ट की अधिकितम लंबाई को दर्शाता है। LLM भाषा पैटर्न की जटिलता को नियंत्रित करता है जिसे मॉडल समझ सकता है और टेक्स्ट का आकार जिसे वह कसी भी समय प्रोसेस कर सकता है। संदर्भ वड़ी का आकार मॉडल की बनावट और उसकी भनिनता द्वारा परभिष्ठि किया जाता है।

कमज़ोरी के सामान्य उदाहरण

1. ऐसे प्रश्न प्रस्तुत करना, जिनके कारण कतार में बहुत अधिक मात्रा में कार्य उत्पन्न करने के माध्यम से संसाधनों का बार-बार उपयोग होता है, जैसे कि LangChain या AutoGPT के साथ।
2. ऐसे प्रश्न भेजना जो असामान्य रूप से संसाधन-खपत वाली हों, शायद इसलाए क्योंकि वे असामान्य शब्दावली या अनुक्रम का उपयोग करते हैं।
3. लगातार इनपुट ओवरफ्लो: एक हमलावर LLM को इनपुट की एक शून्खला भेजता है, जो उसकी क्षमता (context window) को पार कर जाती है, जिससे मॉडल बहुत ज्यादा मशीनी संसाधनों का उपभोग कर लेता है।
4. लंबे इनपुट की शून्खला : हमलावर बार-बार LLM को लंबे इनपुट भेजता है, जिनमें से प्रत्येक कॉन्टेक्स्ट वड़ी (context window) को पार कर जाता है।
5. बार-बार संदर्भ (context) वसितार : हमलावर इनपुट बनाता है जो बार-बार संदर्भ वसितार को ट्रांजिट करता है, जिससे LLM को बार-बार वसितार करने और संदर्भ वड़ी को प्रोसेस करने के लाए मजबूर होना पड़ता है।
6. परविरतनीय-लंबाई वाले इनपुट की बाढ़: हमलावर LLM में बड़ी मात्रा में परविरतनीय-लंबाई वाले इनपुट भेजता है, जहाँ हर इनपुट को सावधानी से तैयार किया जाता है ताकि संदर्भ वड़ी की सीमा तक पहुँचा जा सके। इस तकनीक का उद्देश्य परविरतनशील इनपुट को प्रोसेस करने में कसी भी अक्षमता का फायदा उठाना, LLM पर दबाव डालना और संभावित रूप से इसे अनुत्तरदायी बनाना है।

हमले के परदृश्य में उदाहरण

- एक हमलावर होस्ट करने वाले मॉडल के लिए लगातार कई अनुरोध भेजता है, जिन्हें प्रोसेस करना मुश्किली और महंगा होता है, जिससे दूसरे यूज़र की सेवा खराब हो जाती है और होस्ट के लिए संसाधनों के बलि में वृद्धि होती है।
- एक वेबपेज पर टेक्स्ट का एक टुकड़ा तब सामने आता है जब एक LLM-संचालित उपकरण एक सौम्य प्रश्न का उत्तर देने के लिए जानकारी एकत्र कर रहा होता है। इससे टूल कई और वेब पेज अनुरोध करने लगता है, जिसके परणिमस्वरूप बड़ी मात्रा में संसाधन की खपत होती है।
- एक हमलावर लगातार LLM पर इनपुट की बम्बारी करता है, जो कॉन्टेक्स्ट वड़ों की क्षमता को पार जाती है। हमलावर बड़ी मात्रा में इनपुट भेजने के लिए स्वचालित स्क्रप्ट या टूल का इस्तेमाल कर सकता है, जो LLM की प्रोसेसिंग क्षमताएं पर भारी पड़ जाती है। परणिमस्वरूप, LLM मशीनी संसाधनों की अत्यधिक खपत कर लेता है, जिससे सिस्टम काफी धीमा हो जाता है या पूरी तरह से अनुत्तरदायी हो जाता है।
- एक हमलावर LLM को क्रमबद्ध इनपुट की एक शृंखला भेजता है, जिसमें हर इनपुट संदर्भ वड़ों की सीमा से नीचे होता है। इन इनपुट को बार-बार सबमिट करके, हमलावर का लक्ष्य संदर्भ वड़ों की उपलब्ध क्षमता को खत्म करना है। चूंकि LLM हर इनपुट को उसकी संदर्भ वड़ों में प्रोसेस करने के लिए संघर्ष करता है, जिससे सिस्टम के संसाधन कमजोर हो जाते हैं। इसके परणिमस्वरूप प्रदर्शन खराब हो सकता है या सेवा से पूरी तरह इनकार कर दिया जाता है।
- एक हमलावर संदर्भ वसितार को बार-बार ट्रिगर करने के लिए LLM पुनरावर्ती तंत्र का लाभ उठाता है। इसका फायदा उठाने वाले इनपुट को तैयार करके, हमलावर मॉडल को महत्वपूर्ण मशीनी संसाधनों का उपयोग करते हुए संदर्भ वड़ों को बार-बार वसितारति और प्रोसेस करने के लिए मजबूर करता है। यह हमला सिस्टम पर दबाव डालता है और DoS स्थिति पैदा कर सकता है, जिससे LLM अनुत्तरदायी हो सकता है या क्रैश हो सकता है।
- एक हमलावर LLM में बड़ी मात्रा में परविरतनशील इनपुट दे देता है, जिन्हें संदर्भ वड़ों की सीमा तक पहुंचने के लिये तैयार किया जाता है। परविरतनशील लंबाई के इनपुट से LLM पर हावी होकर, वह अक्षमता का फायदा उठाता है। इनपुट्स की यह बाढ़ LLM संसाधनों पर अत्यधिक भार डालती है, जिससे संभावित रूप से प्रदर्शन में गरिवट आ सकती है और वैध अनुरोधों का जवाब देने में सिस्टम की क्षमता में बाधा आ सकती है।

बचाव कैसे करें

- यह पक्का करने के लिए कियूज़र इनपुट निर्धारित सीमाओं का पालन करता है और किसी भी दुर्भावनापूर्ण इनपुट्स को फ़िलिटर करता है, इनपुट सत्यापन और सैनिटाइज़ेशन लागू करें।
- प्रत्येक अनुरोध या चरण के अनुसार संसाधन उपयोग को सीमित करें, ताकि जिटलि भागों से जुड़े अनुरोध धीरे-धीरे नष्टिपादति हों।
- किसी व्यक्तिगत यूज़र या IP पते द्वारा एक विशिष्ट समय सीमा के भीतर करने वाले अनुरोधों की संख्या

को सीमति करने के लिए API दर सीमा लागू करें।

4. LLM रसिपॉन्स पर प्रतक्रिया करने वाले सिस्टम में कतारबद्ध क्रयिआओं और कुल क्रयिआओं की संख्या सीमति करें।
5. असामान्य स्पाइक्स या पैटर्न की पहचान करने के लिए LLM के संसाधन उपयोग की लगातार निरियानी करें जो DoS हमले का संकेत दे सकते हैं।
6. LLM कॉन्टेक्स्ट विडियो के आधार पर सख्त इनपुट सीमाएँ निर्धारित करें, ताकि ओवरलोड और संसाधनों की कमी को रोका जा सके।
7. LLM में DoS की संभावति कमजोरियों के बारे में डेवलपर्स के बीच जागरूकता को बढ़ावा दें और LLM को सुरक्षित रूप से लागू करने के लिए दिशानिर्देश प्रदान करें।

संदर्भ लिंक

1. LangChain max_iterations: hwchase17 on Twitter
2. Sponge Examples: Energy-Latency Attacks on Neural Networks: Arxiv White Paper
3. OWASP DOS Attack: OWASP
4. Learning From Machines: Know Thy Context: Luke Bechtel
5. Sourcegraph Security Incident on API Limits Manipulation and DoS Attack : Sourcegraph



LLM05: सप्लाई चेन की कमज़ोरियाँ

LLM में आपूर्ति शृंखला कमज़ोर हो सकती है, जो प्रशक्षिण डेटा, LLM मॉडल और परनियोजन प्लेटफार्मों (deployment platforms) की अखंडता को प्रभावित कर सकती है। इन कमज़ोरियों के कारण पक्षपातपूर्ण परणिम, सुरक्षा उल्लंघन या यहां तक कसिंपूर्ण सिस्टम वफिलताएं हो सकती हैं। परंपरागत रूप से, कमज़ोरियाँ सॉफ्टवेयर घटकों (components) पर केंद्रित होती हैं, लेकिन मशीन लर्निंग इसे पूर्व-प्रशक्षिण मॉडल और तीसरे-पक्षों द्वारा दिये कए गए प्रशक्षिण डेटा के साथ जोड़ती हैं जो छेड़छाड़ और विषिक्तपूर्ण हमलों के लिए अतिसिंवेदनशील होते हैं।

अंत में, LLM plugin एक्स्टेंशन अपनी कमज़ोरियाँ ला सकते हैं। इन्हें LLM के असुरक्षित plugin डिज़ाइन में वर्णित किया गया है, जो LLM plugin लिखित शामिल करता है और तीसरे पक्ष के plugin स का मूल्यांकन करने के लिए उपयोगी जानकारी प्रदान करता है।

कमज़ोरी के सामान्य उदाहरण

- पारंपरिक तृतीय-पक्ष पैकेज की कमज़ोरियाँ, जिनमें पुराने या पुराने हो चुके घटक शामिल हैं।
- फाइने-ट्यूनिंग के लिए पहले से प्रशक्षिण कमज़ोर मॉडल का इस्तेमाल करना।
- प्रशक्षिण के लिए विषिक्त क्राउड-सोर्स डेटा का इस्तेमाल करना।
- पुराने या खत्म हो चुके मॉडल जिनका रखरखाव नहीं किया जाता है, उनका उपयोग करने से सुरक्षा संबंधी समस्याएं पैदा हो जाती हैं।
- मॉडल ऑपरेटर्स (model operators) की अस्पष्ट शर्तें और डेटा गोपनीयता नीतियां (T&Cs) होने की बजह से ऐप्लिकेशन के संवेदनशील डेटा का इस्तेमाल मॉडल प्रशक्षिण और बाद में संवेदनशील जानकारी को उजागर करने के लिए करता है। यह मॉडल सप्लायर द्वारा कॉपीराइट (copyrighted) की गई सामग्री का इस्तेमाल करने से होने वाले जोखिमों पर भी लागू हो सकता है।

बचाव कैसे करें

- सरिफ़ भरोसेमंद सप्लायर्स का इस्तेमाल करके डेटा स्रोतों और आपूर्तिकर्ताओं की सावधानी से जाँच करें, जिनमें नियम और शर्तें और उनकी गोपनीय नीतियां शामिल हैं। इसके लिए पर्याप्त और स्वतंत्र रूप से ऑडिट की गई सुरक्षा मौजूद हो और मॉडल ऑपरेटर नीतियां आपकी डेटा सुरक्षा नीतियों के अनुरूप हों (यानी आपके डेटा का इस्तेमाल उनके मॉडलों के प्रशक्षिण के लिए नहीं किया जाता है)। इसी तरह, मॉडल अनुरक्षकों से कॉपीराइट सामग्री का इस्तेमाल करने के खलिफ आश्वासन और कानूनी कार्रवाई की तलाश करें।
- सरिफ़ प्रतिष्ठित प्लग-इन का इस्तेमाल करें और पक्का करें कि आपकी ऐप्लिकेशन से जुड़ी ज़रूरतों के



लाए उनका परीक्षण किया गया हो। LLM के असुरक्षित plugin डिज़ाइन से उसके वभिन्न पहलुओं के बारे में जानकारी प्रदान करता है, जिनके खलिफ आपको तीसरे पक्ष के plugin का उपयोग करने से होने वाले जोखमियों को कम करने के लाए परीक्षण करना चाहिए।

3. “OWASP टॉप टेन A06:2021 - कमज़ोर और पुराने घटक” को समझें और लागू करें। इसमें कमज़ोरिया ढूढ़ना, प्रबंधन और पैचिंग घटक (patching components) शामिल हैं। संवेदनशील डेटा तक पहुंच वाले डेवलपर्सेट वातावरण के लाए, इन नियंत्रणों को लागू करें।

4. डिप्लॉय किए गए पैकेज के साथ छेड़छाड़ को रोकने के लाए, सॉफ्टवेयर सामग्री के बिल (SBOM) के उपयोग से घटकों (components) की नवीनतम, सटीक और हस्ताक्षरित सूची बिनाए रखें। SBOM, जीरो-डेट कमज़ोरियों (zero-date vulnerabilities) को तुरंत पता लगा लगा कर, सचेत कर सकते हैं।

5. लखित समय SBOM मॉडल उसकी सामग्री एवं डेटासेट को कवर नहीं करता। अगर आपका LLM ऐप्लिकेशन अपने मॉडल का इस्तेमाल करता है, तो आपको MLOP के तरीकों और प्लेटफॉर्म का इस्तेमाल करना चाहिए, जो डेटा, मॉडल और प्रयोग ट्रैकिंग के साथ सुरक्षित मॉडल सूची पेश करते हैं।

6. बाहरी मॉडल और सप्लायर (suppliers) का इस्तेमाल करते समय आपको मॉडल और कोड साइनिंग (code signing) का भी इस्तेमाल करना चाहिए।

7. जैसा कि प्रशक्षण डेटा पॉइंज़निंग में चर्चा की गई है, की दिये गये मॉडल और डेटा में वसिंगति ढूढ़ना और प्रतकूल समर्थ परीक्षण के प्रयोग से छेड़छाड़ और विषिक्तता का पता लगाया जा है। आदरश रूप से, यह एम. एल. ओपी पाइपलाइन (MLOps pipelines) का हस्सा होना चाहिए; हालाँकि, ये उभरती हुई तकनीकें हैं और रेड टीमिंग अभ्यासों के रूप में इन्हें आसानी से लागू किया जा सकता है।

8. मॉडल एवं उसकी सामग्री, पुराने घटक (out-of-date components), प्रयावरण की कमज़ोरियों को स्कैन करने तथा अनाधिकृत plugin के इस्तेमाल को कवर करने के लाए प्रयाप्त निगरानी आवश्यक है।

9. कमज़ोर या पुराने घटकों को कम करने के लाए पैचिंग नीति आवश्यक है। यह सुनिश्चित करे की ऐप्लिकेशन, API के अनुरक्षित संस्करण और दाए गये मॉडल पर नरिभर करता है।

10. सप्लायर की सुरक्षा और पहुंच की नियमिति समीक्षा करें और उनका ऑडिट कर यह निश्चित करें कि उनकी सुरक्षा स्थितिया नियम और शर्तों में कोई बदलाव न हो।

हमले के परदिश्य में उदाहरण

- एक हमलावर कमज़ोर पायथन लाइब्रेरी का इस्तेमाल कर सिस्टम को जोखमि में डाल सकता है। यह पहली बार Open AI डेटा चोरी (data breach) में हुआ था।
- एक हमलावर फ्लाइट खोजने के लाए एक LLM plugin प्रदान करता है, जो नकली लकि बनाता है, जिससे plugin के यूज़र को धोके का सामना करना पड़ता है।
- एक असल हमले में, हमलावर PyPI पैकेज सूचिका इस्तेमाल कर मॉडल के डेवलपर्स को धोखा देता है, जिसमें वह खराब पैकेज को डाउनलोड करा कर, डेटा नकिल सकें या मॉडल के वकिस माहौल में विशेषाधिकार बढ़ा सकें।



4. एक हमलावर आर्थिक वशिलेषण और सामाजिक शोध में वशिष्यता वाले सार्वजनिक रूप से उपलब्ध, पूर्व-प्रशिक्षित मॉडल को विषिकृत बना देता है। इससे एक बैकडोर बनाता है, जिससे ग़लत सूचनाएं और फ़र्ज़ी खबरें बनती हैं। यह टारगेट को लक्षित करने के लिये इसे कसी मॉडल मार्केटप्लेस (जैसे HuggingFace) में स्तापित कर देते हैं।
5. एक हमलावर सार्वजनिक रूप से उपलब्ध डेटा को विषिकृत करता है, ताकि मॉडल को ठीक करते समय बैकडोर बन सके, जो अलग-अलग मार्केट्स की कुछ कंपनियों को फेवर करता है।
6. सप्लायर (आउटसोर्सिंग डेवलपर, होस्टिंग कंपनी आदि) के एक कम्प्रोमाइज़्ड कर्मचारी के द्वारा IP चुराने के लिए डेटा, मॉडल या कोड में घुसपैठ करना।
7. एक LLM ऑपरेटर अपनी नियम व शर्तों एवं गोपनीयता नीति में बदलाव करता है, ताकि उसे मॉडल प्रशिक्षण के लिए ऐप्लिकेशन डेटा का इस्तेमाल न करना पड़े, जिससे संवेदनशील डेटा याद रहे।

संदर्भ लिंक

1. ChatGPT Data Breach Confirmed as Security Firm Warns of Vulnerable Component Exploitation: Security Week
2. Plugin review process OpenAI
3. Compromised PyTorch-nightly dependency chain: Pytorch
4. PoisonGPT: How we hid a lobotomized LLM on Hugging Face to spread fake news: Mithril Security
5. Army looking at the possibility of 'AI BOMs: Defense Scoop
6. Failure Modes in Machine Learning: Microsoft
7. ML Supply Chain Compromise: MITRE ATLAS
8. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples: Arxiv White Paper
9. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain: Arxiv White Paper
10. VirusTotal Poisoning: MITRE ATLAS



LLM06: संवेदनशील जानकारी का खुलासा

एक LLM ऐप्लिकेशन अपने आउटपुट के ज़रए संवेदनशील जानकारी, उसकी एल्गोरिदम तथा दूसरी गोपनीय जानकारिया प्रकट कर सकती है। इसके परणामस्वरूप संवेदनशील डेटा तक अनधिकृत पहुंच, बौद्धिक संपदा, गोपनीयता उल्लंघन और अन्य प्रकार के सुरक्षा उल्लंघनों का सामना करना पड़ सकता है। LLM ऐप्लिकेशन के यूजर के लिए यह जरूरी है कि वह LLM को सुरक्षित तरीके से प्रयोग करे और ऐसे संवेदनशील डेटा को अनजाने में इनपुट करने से जुड़े जोखिमों की पहचान करे।

इस जोखिम को कम करने के लिए, LLM ऐप्लिकेशन को यूजर डेटा को प्रशक्षण मॉडल डेटा में प्रवेश करने से रोकने के लिए प्रयाप्त डेटा सैनटिइजेशन करना चाहें। यूजर को उनके डेटा को प्रोसेस करने के तरीके और प्रशक्षण मॉडल में अपने डेटा को शामिल करने से बहार नकिलने की क्षमता के बारे में जानकारी देने के लिए LLM ऐप्लिकेशन के मालिकों के पास उपयुक्त उपयोग की शर्तों की नीतियां (Terms of Use policies) भी उपलब्ध होनी चाहें।

उपभोक्ता-LLM ऐप्लिकेशन इंटरैक्शन वशिवास की दो-तरफ़ा सीमा बनाता है, जहाँ हम क्लाइंट->LLM इनपुट या LLM-> क्लाइंट आउटपुट पर स्वाभाविक रूप से भरोसा नहीं कर सकते हैं। यह ध्यान रखना जरूरी है कि इस खिलाफ़ की वजह से कुछ जरूरी शर्तें दायरे से बाहर हैं, जैसे थ्रेट मॉडलिंग अभ्यास, इंफ्रास्ट्रक्चर को सुरक्षित करना और प्रयाप्त सैडबॉक्सिंग। LLM को कसि प्रकार का डेटा वापस करना चाहें, इसके बारे में ससिटम प्रॉम्प्ट में प्रतिबंध जोड़ने से संवेदनशील जानकारी के प्रकटीकरण के खलिफ़ कुछ कमी मिल सकती है, लेकिन LLM की अप्रत्याशित प्रकृतिका मतलब है कि ऐसे प्रतिबंधों का हमेशा सम्मान नहीं किया जा सकता है और प्रॉम्प्ट इंजेक्शन या अन्य वैक्टर की मदद से इन्हें रोका जा सकता है।

कमज़ोरी के सामान्य उदाहरण

1. LLM के जवाबों में संवेदनशील जानकारी को अधूरे या गलत तरीके से फिलिटर करना।
2. LLM प्रशक्षण प्रक्रिया में संवेदनशील डेटा को ओवेरफ़टि या याद रखना।
3. LLM की ग़लतफ़हमी, डेटा की सफाई में कमी या तुरटियों के कारण गोपनीय जानकारी का अनायास ही खुलासा हो जाना।

बचाव कैसे करें

1. यूजर डेटा को ट्रेनिंग मॉडल डेटा में जाने से रोकने के लिए प्रयाप्त डेटा सैनटिइजेशन और स्क्रबिंग तकनीकों



का उपयोग करे।

2. मॉडल को विषिक्ता से बचाने के लिए, इनपुट सत्यापन और सैनटिडजेशन के मजबूत तरीके लागू करें, जिससे की दुरभावनापूर्ण इनपुट पहचान कर फ़िल्टर किया जा सके।
3. मॉडल को डेटा से समृद्ध करते समय या नमिनलरिंगिंग के लिए में दिये गये को फ़ाइन-ट्र्यूनिंग करते समय (यानी, डिप्लॉयमेंट से पहले या उसके दौरान मॉडल में फीड किया गया डेटा)
 1. फ़ाइन-ट्र्यूनिंग के समय संवेदनशील डेटा, यूज़र के समक्ष प्रकट हो सकता है। इसलिए, न्यूनतम पहुंच का नियम लागू करें और मॉडल को विशेषाधिकार वाली जानकारी से प्रशक्षिति न करें।
 2. बाहरी डेटा स्रोतों तक पहुंच (रनटाइम के समय डेटा का ऑर्केस्ट्रेशन) सीमित होनी चाहिए।
 3. बाहरी डेटा स्रोतों पर ऐक्सेस नियंत्रण और सुरक्षित सप्लाई चेन बनाए रखने के लिए एक कठोर तरीका अपनाएं।

हमले के परदिश्य में उदाहरण

1. एक यूज़र गैर-दुरभावनापूर्ण तरीके से, LLM एप्लीकेशन के ज़रिये कुछ अन्य यूज़र डेटा के संपर्क में आ जाता है।
2. एक यूज़र LLM के इनपुट फ़िल्टर और सैनटिडजेशन को बायपास करने के लिए प्रॉमॉटस की एक शर्खला को लक्षित करता है, इससे ऐप्लिकेशन के अन्य यूज़र के बारे में संवेदनशील जानकारी (PII) प्राप्त कर सकते।
3. यूज़र या LLM ऐप्लिकेशन की लापरवाही से प्रशक्षित डेटा के ज़रिये मॉडल में PII जैसा व्यक्तिगत डेटा लीक हो जाता है। इससे 1 एवं 2 के जोखिम की संभावना बढ़ सकती है।

संदर्भ लिंक

1. AI data leak crisis: New tool prevents company secrets from being fed to ChatGPT:
[Fox Business](#)
2. Lessons learned from ChatGPT's Samsung leak:
[Cybernews](#)
3. Cohere - Terms Of Use
[Cohere](#)
4. A threat modeling example:
[AI Village](#)
5. OWASP AI Security and Privacy Guide:
[OWASP AI Security & Privacy Guide](#)
6. Ensuring the Security of Large Language Models:
[Experts Exchange](#)

LLM07: असुरक्षित plugin डिज़ाइन

LLM plugin ऐसे एक्सटेंशन होते हैं, जिन्हें चालू करने पर, यूजर इंटरैक्शन के दौरान मॉडल द्वारा अपने-आप बुला लिये जाते हैं। वे मॉडल द्वारा संचालित होते हैं, और इनके निपादन पर ऐप्लिकेशन का कोई नियंत्रण नहीं होता है। इसके अलावा, सन्दर्भ के आकार की सीमाओं से निपटने के लिए, plugin द्वारा मॉडल से फ्री-टेक्स्ट इनपुट लागू करि जा सकते हैं, बनि कसी सत्यापन या टाइप चेकिंग को। इससे एक संभावति हमलावर plugin के लिए एक दुर्भावनापूर्ण प्रांप्ट बना सकता है, जिसके परणामस्वरूप कई तरह के अवांछित व्यवहार हो सकते हैं, जिसमें रमिट कोड चलाना भी शामलि है।

दुर्भावनापूर्ण इनपुट का नुकसान अक्सर अप्रयाप्त ऐक्सेस नियंत्रणों और सभी plugin में अनुमतिको ट्रैक न कर पाने पर निभर करता है। अप्रयाप्त ऐक्सेस नियंत्रण की मदद से एक plugin दूसरे plugin पर आँख बंद करके भरोसा कर सकता है और यह मान लेता है कि अंतमि यूज़र ने इनपुट दिए हैं। इस तरह के अप्रयाप्त ऐक्सेस नियंत्रण की वजह से दुर्भावनापूर्ण इनपुट के हानकारक परणाम हो सकते हैं, जैसे किंदिटा एक्सफ्लिट्रेशन, रमिट कोड चलाना और वशिषाधकिर बढ़ाना शामलि हैं।

यह तृतीय-पक्ष plugin का उपयोग करने के बजाय LLM plugin के निर्माण पर केंद्रिति है, जो कि LLM-सप्लाई-चेन की कमज़ोरियों द्वारा कवर किया जाता है।

कमज़ोरी के सामान्य उदाहरण

- एक plugin अलग-अलग इनपुट मापदंडों के बजाय एक ही टेक्स्ट फ़िल्ड में सभी मापदंडों को स्वीकार करता है।
- एक plugin मापदंडों (parameters) के बजाय कॉन्फ़िगरेशन स्ट्रिंग (configuration strings) को स्वीकार करता है, जो पूरी कॉन्फ़िगरेशन सेटिंग (configuration settings) को बदल सकता है।
- एक plugin मापदंडों के बजाय Raw SQL या प्रोग्रामिंग स्टेटमेंट (programming statements) को स्वीकार करता है।
- कसी plugin को स्पष्ट अनुमति दिए (authorization) बनि ही प्रमाणीकरण (authentication) किया जाता है।
- एक plugin सारी LLM सामग्री को पूरी तरह से यूज़र द्वारा बनाई गई मानता है और बनि कसी अनुमति (authorization) के कोई भी अनुरोध स्वीकार कर लेता है।



बचाव कैसे करें

1. Plugins में सख्त पैरामीटर के अनुसार इनपुट लागू करना, इनपुट पर टाइप और रेज की जाँच शामलि करनी चाहए। इनके आभाव में दूसरी लेयर शुरू कर अनुरोधों को पारस, सैनटिङ्ज एवं उनकी पुष्टिकरनी चाहए। जब ऐप्लिकेशन समिटिक्स की वजह से फ्रीफॉर्म इनपुट स्वीकार कया जाता है, तो यह सुनशिचति करे कि किसी भी संभावित हानिकारक तरीके का इस्तेमाल नहीं कया जा रहा है।
2. Plugins डेवलपर्स को इनपुट की पूष्टिओर सैनटिङ्जेशन सुनशिचति करने के लिए, ASVS (ऐप्लिकेशन सुरक्षा सत्यापन मानक) में दिये गये OWASP के सुझावों को लागू करना चाहए।
3. प्राप्त पुष्टि सुनशिचति करने के लिए plugin की पूरी जाँच और परीक्षण कया जाना चाहए। डेवलपमेंट पाइपलाइन (development pipelines) में स्टैटिक ऐप्लिकेशन सुरक्षा परीक्षण (SAST) स्कैन के साथ-साथ डायनामिक और इंटरैक्टिव ऐप्लिकेशन परीक्षण (DAST, IAST) का इस्तेमाल करें।
4. OWASP ASVS ऐक्सेस कंट्रोल दशानिर्देशों का पालन करते हुए किसी भी असुरक्षित इनपुट पैरामीटर के इस्तेमाल के प्रभाव को कम करने के लिए plugins को डिज़ाइन कया जाना चाहए। इसमें कम से कम वशिष्ठाधकिर प्राप्त ऐक्सेस नियंत्रण शामलि है, जो अपना इच्छति कार्य करते समय जितना संभव हो उतनी कम कार्यक्षमता को उजागर करता है।
5. प्रभावी प्राधकरण (effective authorization) और ऐक्सेस नियंत्रण (access control) लागू करने के लिए plugin को उपयुक्त प्रमाणीकरण पहचान (authentication identities) का इस्तेमाल करना चाहए, जैसे कि OAuth2। इसके अलावा, एपीआई कीज (API Keys) का इस्तेमाल करना प्राधकरण (authorization) नियंत्रणों के लिए संदर्भ देने के लिए कया जाना चाहए, जो डफ़ॉल्ट इंटरैक्टिव यूजर के बजाय plugin रूट को दर्शाता है।
6. संवेदनशील plugins द्वारा की गई किसी भी कार्रवाई के लिये यूजर की अनुमति और पुष्टिकी आवश्यकता होती है।
7. Plugins, आम तौर पर, REST API होते हैं, इसलिए डेवलपर्स को सामान्य कमजोरियों को कम करने के लिए OWASP के टॉप 10 API सुरक्षा जोखिमों — 2023 में दिये गये सुझावों को लागू करना चाहए।

उदाहरण हमले के परदीश्य

1. एक plugin मूल URL को स्वीकार करता है और LLM को निर्देश देता है कि वह URL को एक क्वेरी (query) के साथ मिलाएं, ताकि यूजर के अनुरोध पर मौसम का पूर्वानुमान प्राप्त किया जा सके। एक दुर्भावनापूर्ण यूजर अनुरोध पर अपने डोमेन के जरूरि LLM सिस्टम में अपनी सामग्री इंजेक्ट कर सकते हैं।
2. एक plugin फ़ील्ड में फ्री फॉर्म इनपुट को स्वीकार करता है जिसे वह मान्य नहीं करता है। एक हमलावर पेलोड का प्रयोग करते हुये गलती के संदेशों से सारी जानकारी प्राप्त करता है। इसके बाद यह कोड निषिपादित करने और डेटा नकालने या वशिष्ठाधकिर बढ़ाने के लिए तृतीय-पक्ष की कमजोरियों का फायदा उठाता है।
3. किसी वेक्टर स्टोर (vector store) से एम्बेडिंग प्राप्त करने के लिए इस्तेमाल कया जाने वाला एक plugin,



कॉन्फ़िगरेशन पैरामीटर (configuration parameters) को कनेक्शन स्ट्रिंग (connection string) के तौर पर बना कर्सी सत्यापन के स्वीकार करता है। इसकी मदद से कोई हमलावर नाम या होस्ट पैरामीटर बदलकर दूसरे वेक्टर स्टोर से भी अनाधिकृत एम्बेडिंग प्राप्त कर सकता है।

4. एक plugin “SQL WHERE” क्लॉज को एडवांस फ़िल्टर के रूप में स्वीकार करता है, जिन्हें बाद में फ़िल्टर करने वाले SQL में जोड़ दिया जाता है। इससे एक हमलावर SQL हमला कर सकता है।
5. एक हमलावर एक असुरक्षित कोड प्रबंधन plugin का फायदा उठाने के लिए अप्रत्यक्ष प्रॉम्प्ट इंजेक्शन का इस्तेमाल करता है, जिसमें कोई इनपुट सत्यापन नहीं होता है, इसके साथ कमज़ोर ऐक्सेस नियंत्रण से रपिंजटिरी का स्वामतिव ट्रांसफर और यूज़र को उनकी रपिंजटिरी से लॉक किया जा सकता है।

संदर्भ लिंक

1. [OpenAI ChatGPT Plugins: ChatGPT Developer’s Guide](#)
2. [OpenAI ChatGPT Plugins - Plugin Flow: OpenAI Documentation](#)
3. [OpenAI ChatGPT Plugins - Authentication: OpenAI Documentation](#)
4. [OpenAI Semantic Search Plugin Sample: OpenAI Github](#)
5. [Plugin Vulnerabilities: Visit a Website and Have Your Source Code Stolen: Embrace The Red](#)
6. [ChatGPT Plugin Exploit Explained: From Prompt Injection to Accessing Private Data Embrace The Red](#)
7. [OWASP ASVS - 5 Validation, Sanitization and Encoding: OWASP AASVS](#)
8. [OWASP ASVS 4.1 General Access Control Design: OWASP AASVS](#)
9. [OWASP Top 10 API Security Risks – 2023: OWASP](#)



LLM08: अत्यधिक एजेंसी

LLM-आधारित सिस्टम को अक्सर उसके डेवलपर द्वारा दूसरे प्रणाली सिस्टम के साथ इंटरफ़ेस करने और कसी प्रॉम्प्ट के जवाब में कारबाई करने की क्षमता प्रदान की जाती है। इनपुट प्रॉम्प्ट या LLM आउटपुट के आधार पर डायनामिक रूप से नियंत्रित करने के लिए कसी फ़ंक्शन को लागू करना है इसका नियंत्रित LLM 'एजेंट' को भी सौंपा जा सकता है।

अत्यधिक क्षमता वह कमज़ोरी है जिसके कारण LLM से अनपेक्षित आउटपुट के जवाब में हानिकारक कारबाईयां की जा सकती हैं (भले ही LLM में खराबी क्यों न हो; चाहे वह मतभिरम हो, प्रत्यक्ष/अप्रत्यक्ष रूप से शीघ्र इंजेक्शन हो, दुरभावनापूर्ण plugin हो, खराब तरीके से तैयार कए गए सौम्य प्रॉम्प्ट, या सरिफ़ खराब प्रदर्शन करने वाला मॉडल)। अत्यधिक क्षमता का मूल कारण आम तौर पर एक या एक से अधिक होता है, जैसे की अत्यधिक कारबाईयां, अत्यधिक अनुमतियां या अत्यधिक स्वायत्तता।

अत्यधिक क्षमता गोपनीयता (confidentiality), सत्यनिष्ठा (integrity) और उपलब्धता (availability) के संदर्भान्तो पर कई तरह के प्रभाव डाल सकती हैं और यह इस बात पर नियंत्रित करती है कि LLM-आधारित ऐप कनि सिस्टम के साथ इंटरैक्ट कर सकता है।

कमज़ोरी के सामान्य उदाहरण

1. **अत्यधिक कारबाईयां:** एक LLM एजेंट एक plugin का उपयोग करता है, जिसमें ऐसे फ़ंक्शन शामिल हैं जिनकी सिस्टम के संचालन के लिए आवश्यकता नहीं है। उदाहरण के लिए, कसी डेवलपर को कसी LLM एजेंट को रपिंज़िटरी से दस्तावेज़ पढ़ने की सुवधा देनी होती है, इसके लिये वह तीसरे पक्ष के plugin का इस्तेमाल करते हैं, जिसके पास दस्तावेज़ों को संशोधित करने और हटाने की क्षमता भी है। वैकल्पिक रूप से, हो सकता है कि कसी plugin को डेवलपर्मेंट के कसी चरण के दौरान ट्रायल किया गया हो और उसे कसी बेहतर विकल्प के पक्ष में छोड़ दिया गया हो, लेकिन मूल plugin LLM एजेंट के लिए उपलब्ध रहेगा।
2. **अत्यधिक कारबाईयां:** ओपन-एंडेड फ़ंक्शनैलिटी वाला LLM plugin, ऐप्लिकेशन के संचालन के लिए आवश्यक चीज़ों में कमांड के इनपुट नियंत्रणों को ठीक से फ़िलिटर करने में वफ़िल रहता है। उदाहरण के लिए, एक विशिष्ट शेल कमांड चलाने वाला plugin दूसरे शेल कमांड को नियंत्रित होने से रोकने में वफ़िल रहता है।
3. **अत्यधिक अनुमतियां:** LLM plugin के पास दूसरे सिस्टम पर अनुमतियां होती हैं जिनकी ऐप्लिकेशन संचालित करने के लिए जरूरत नहीं होती है। उदाहरण के लिए, डेटा पढ़ने के लिए बनाया गया plugin कसी पहचान का इस्तेमाल करके डेटाबेस सर्वर से कनेक्ट होता है, जिसमें न केवल SELECT की अनुमतियां होती हैं, बल्कि UPDATE, INSERT और DELETE की अनुमतियां भी होती हैं।



4. अत्यधिक अनुमतियां: एक LLM plugin जसे यूजर की ओर से ऑपरेशन करने के लिए बनाया गया है, वह वशिष्ठाधकिर प्राप्त कर डाउनस्ट्रीम सिस्टम को ऐक्सेस करता है। उदाहरण के लिए, मौजूदा यूजर के दस्तावेजों को पढ़ने के लिए एक plugin है, जो एक वशिष्ठाधकिर प्राप्त खाते के साथ दस्तावेज़ रपिंजटिरी से कनेक्ट होता है, जिसके पास सभी यूजर की फाइलों तक पहुंच होती है।
5. अत्यधिक स्वायत्तता: कोई LLM-आधारित एप्लिकेशन या plugin हाई-इम्पैक्ट कार्रवाइयों को स्वतंत्र रूप से सत्यापति करने और उन्हें मंजूरी देने में वफिल रहता है। उदाहरण के लिए, एक plugin जो बना कसी पुष्टि के यूजर के दस्तावेजों को हटाने की अनुमति देता है।

बचाव कैसे करें

निम्नलिखित कार्रवाइयों से अत्यधिक एजेंसी को रोका जा सकता है:

1. उन plugin/टूल को जरूरी फ़ंक्शन तक सीमति करें जिन्हें LLM एजेंट कॉल करने की अनुमति दिते हैं। उदाहरण के लिए, अगर कसी LLM-आधारित सिस्टम के लिए कसी URL की सामग्री लाने की क्षमता की आवश्यकता नहीं है, तो LLM एजेंट को ऐसा plugin नहीं दिये जाने चाहिए।
2. LLM plugin/टूल में लागू किए गए फ़ंक्शन को न्यूनतम आवश्यक तक सीमति करें। उदाहरण के लिए, एक plugin जो ईमेल को सारांशित करता है, उसके लिए केवल ईमेल पढ़ने की क्षमता होनी चाहिए, अन्य कार्रवाइयों को नहीं।
3. जहाँ संभव हो, ओपन-एडेंड फ़ंक्शन से बचें (उदाहरण के लिए, शेल कमांड चलने, URL प्राप्त करने वाले आदि) और ज्यादा लक्षित कार्रवाइयों को इस्तेमाल करें। उदाहरण के लिए, कसी LLM-आधारित ऐप के लिए कसी फ़ाइल में कुछ आउटपुट लक्षित की आवश्यकता हो सकती है। अगर इसे शेल फ़ंक्शन चलाने के लिए plugin का उपयोग करके लागू किया जाता, तो अवांछनीय कार्रवाइयों का दायरा बहुत बड़ा जाता (कोई भी अन्य शेल कमांड नष्टिकरण किया जा सकता है)। एक ज्यादा सुरक्षित विकल्प यह होगा कि एक ऐसा फ़ाइल-राइटिंग plugin बनाया जाए, जो सरिफ़ उस खास सुवधा के साथ ही काम कर सके।
4. उन अनुमतियों को सीमति करें जो LLM plugins/टूल दूसरे सिस्टम को दी जाती है, ताकि अवांछनीय कार्रवाइयों का दायरा सीमति किया जा सके। उदाहरण के लिए, एक LLM एजेंट जो कसी ग्राहक को खरीदारी के सुझाव देने के लिए प्रॉडक्ट डेटाबेस का इस्तेमाल करता है, उसे सरिफ़ 'प्रॉडक्ट' टेबल पढ़ने की ज़रूरत होगी; उसके पास दूसरी टेबल तक ऐक्सेस नहीं होनी चाहिए, न ही रकिंग डालने, अपडेट या हटाने की क्षमता होनी चाहिए। इसे उस पहचान के लिए उपयुक्त डेटाबेस अनुमतियां लागू करनी चाहिए, जिसका इस्तेमाल LLM plugin डेटाबेस से कनेक्ट करने के लिए करता है।
5. यह पक्का करने के लिए कियूजर की ओर से की गई कार्रवाइयां डाउनस्ट्रीम सिस्टम पर उस वशिष्ट यूजर के संदर्भ में और न्यूनतम ज़रूरी वशिष्ठाधकिरों के साथ नष्टिकरण हों, यूजर की अनुमति और सुरक्षा क्षेत्र को ट्रैक करें। उदाहरण के लिए, एक LLM plugin जो यूजर के कोड रेपो को पढ़ता है, तो उसे OAuth के ज़रूरि और न्यूनतम स्कोप के साथ प्रमाणीकरण करना होगा।



6. सभी कार्रवाइयों को करने से पहले मानव द्वारा मंजूरी ले। इसे डाउनस्ट्रीम सिस्टम (LLM ऐप्लिकेशन के दायरे से बाहर) या LLM plugin /टूल में ही लागू किया जा सकता है। उदाहरण के लिए, कसी यूज़र की ओर से सोशल मीडिया कॉन्टेंट बनाने और पोस्ट करने वाले LLM-आधारित ऐप में 'पोस्ट' ऑपरेशन लागू करने वाले plugin /टूल/API में यूज़र की स्वीकृति शामिल होनी चाहिए।
7. कसी कार्रवाई की अनुमति है या नहीं, यह तय करने के लिए LLM पर निभर रहने के बजाय डाउनस्ट्रीम सिस्टम में प्राधिकरण (authorization) लागू करें। टूल/plugin लागू करते समय, मीडियन का पूरा सदिधांत लागू करें, ताकि plugin/टूल के ज़रूरी डाउनस्ट्रीम सिस्टम से कहि गए सभी अनुरोधों की सुरक्षा नीतियों के तहत पुष्ट हो सके।

नमिनलस्विति वकिल्प अत्यधिक एजेंसी को नहीं रोकेंगे, लेकिन इससे होने वाले नुकसान के स्तर को सीमित कर सकते हैं:

1. LLM plugin/टूल और डाउनस्ट्रीम सिस्टम की गतिविधि की निगरानी कर सूचीबद्ध करें ताकि यह पता चल सके कि अवांछनीय कार्रवाइयां कहाँ हो रही हैं और उसी के अनुसार प्रतिक्रिया दें।
2. कसी निश्चिति समयावधि में होने वाली अवांछनीय कार्रवाइयों की संख्या को कम करने के लिए दर-सीमा लागू करें, इससे पहले कि महत्वपूर्ण नुकसान हो, निगरानी के ज़रूरि अवांछनीय कार्रवाइयों का पता लगाने के अवसर बढ़ाएँ।

उदाहरण हमले के परदीश्य

1. एक हमलावर कसी कंपनी के एलएलएम मॉडल रिपॉजिटरी तक अनधिकृत पहुंच हासिल करने के लिए उसके बुनियादी ढांचे में भेद्यता का फायदा उठाता है। हमलावर मूल्यवान एलएलएम मॉडलों में घुसपैठ करने के लिए आगे बढ़ता है और प्रतिस्पर्धी भाषा प्रसंस्करण सेवा शुरू करने या संवेदनशील जानकारी निकालने के लिए उनका उपयोग करता है, जिससे मूल कंपनी को महत्वपूर्ण वित्तीय नुकसान होता है।
2. एक असंतुष्ट कर्मचारी मॉडल या संबंधित कलाकृतियाँ लीक कर देता है। इस परदीश्य के सार्वजनिक प्रदर्शन से हमलावरों के लिए ग्रे बॉक्स प्रतिकूल हमलों या वैकल्पिक रूप से सीधे उपलब्ध संपत्ति को चुराने का ज्ञान बढ़ता है।
3. एक हमलावर सावधानीपूर्वक चयनित इनपुट के साथ एपीआई पर सवाल उठाता है और एक छाया मॉडल बनाने के लिए प्रयाप्त संख्या में आउटपुट एकत्र करता है।
4. आपूर्ति-शांखला के भीतर एक सुरक्षा नियंत्रण विलिता मौजूद है और मालकिना मॉडल जानकारी के डेटा लीक की ओर ले जाती है।
5. एक दुर्भावनापूर्ण हमलावर साइड-चैनल हमला करने और अपने नियंत्रण के तहत रमिट नियंत्रित संसाधन पर मॉडल जानकारी पुनरप्राप्त करने के लिए इनपुट फ़िल्टरिंग तकनीकों और एलएलएम की प्रस्तावना को बायपास करता है।



संदर्भ लिंक

1. Embrace the Red: Confused Deputy Problem: Embrace The Red
2. NeMo-Guardrails: Interface guidelines: NVIDIA Github
3. LangChain: Human-approval for tools: Langchain Documentation
5. Simon Willison: Dual LLM Pattern: Simon Willison

CONFIDENTIAL

LLM09: ओवररलियंस

ज़्यादा नरिभरता तब होती है जब सिस्टम या लोग बनि पर्याप्त नरीक्षण के नरिण्य लेने या कॉन्ट्रैट तैयार करने के लिए LLM पर नरिभर होते हैं। LLM रचनात्मक और जानकारीपूर्ण सामग्री तैयार कर सकते हैं, लेकिन वे ऐसी सामग्री भी जेनरेट कर सकते हैं जो तथ्यात्मक रूप से गलत, अनुचित या असुरक्षित हो। इसे भ्रम या उलझन कहा जाता है और इसकी वजह से गलत सूचना, गलतफ़हमी, कानूनी समस्याएं और प्रतिष्ठित को नुकसान हो सकता है।

LLM द्वारा बनाया गया सोर्स कोड अज्ञात सुरक्षा कमज़ोरियों उत्पन्न कर सकता है। यह एप्लीकेशन की सुरक्षा एवं परचालन सुरक्षा के लिए एक जोखमि पैदा करता है। ये जोखमि समीक्षा की कठोर प्रक्रिया के महत्व को दर्शाते हैं, इसके साथ:

- ओवरसाइट (Oversight)
- सत्यापन की नरितर व्यवस्था (Continuous validation mechanism)
- अस्वीकार्य जो जोखमि में है (Disclaimers on risk)

कमज़ोरी के सामान्य उदाहरण

1. LLM प्रतिक्रिया के तौर पर गलत जानकारी देता है, जिससे गलत जानकारिया फैलती है।
2. LLM तारकिक रूप से असंगत या नरिथक सामग्री तैयार करता है, जो व्याकरण की दृष्टि से सही होती है।
3. LLM अलग-अलग स्रोतों की जानकारियों को मिलाता है, जिससे भ्रामक सामग्री बनती है।
4. LLM असुरक्षित या दोषपूर्ण कोड सुझाता है, जिसके सॉफ्टवेयर सिस्टम में शामिल करने पर कमज़ोरियाँ उत्पन्न होती हैं।
5. प्रदाता द्वारा अंतमि यूजर को अंतर्निहित जोखमियों के बारे में ठीक से बताने में असफल होने के कारण हानकारक परिणाम हो सकते हैं।

बचाव कैसे करें

1. LLM आउटपुट की नियमति नगिरानी और उनकी समीक्षा करें। इनकन्सिस्टेन्ट टेक्स्ट (inconsistent text) को फ़िलिटर करने के लिए आत्म स्थरिता (self-consistency) या वोटिंग तकनीकों का इस्तेमाल करें। एक ही प्रॉम्प्ट के लिए कई मॉडल प्रतिक्रियाओं की तुलना करने से आउटपुट की गुणवत्ता और नरितरता का

बेहतर आकलन किया जा सकता है।

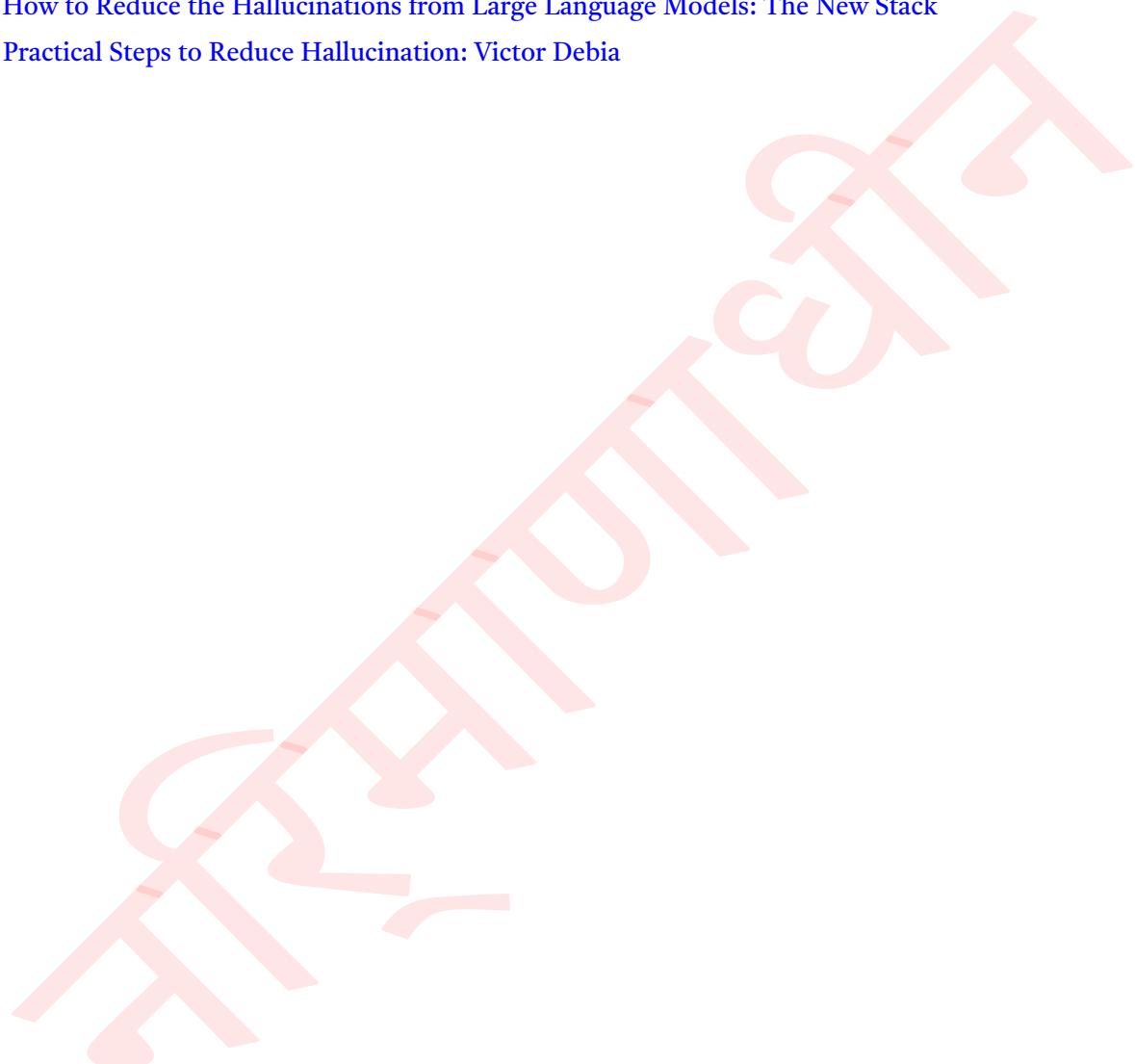
2. वशिक्षणीय बाहरी स्रोतों से LLM आउटपुट को क्रॉस-चेक करें। अतरिक्त पुष्टि से यह पक्का करने में मदद मिल सकती है कि मॉडल के ज़रूरी दी गई जानकारी सटीक एवं भरोसेमंद है।
3. फाइन-ट्यूनिंग या एम्बेडिंग की मदद से आउटपुट को बेहतर बनाएं। कसी खास क्षेत्र के काम के लिये इसके लिये प्रॉमॉटिंग (prompt engineering), पैरामीटर एफ़िशिएंट ट्यूनिंग (parameter efficient tuning), फुल मॉडल ट्यूनिंग (full model tuning) और चेन ऑफ़ थॉट (chain of thought) प्रॉमॉटिंग जैसी तकनीकों का इस्तेमाल किया जा सकता है।
4. ऑटोमैटिक सत्यापन तंत्र लागू करें, जो ज्ञात तथ्यों या डेटा के विद्युत जेनरेट करें। यह सुरक्षा की एक अतरिक्त परत प्रदान कर, मतभिरम से जुड़े जोखिमों को कम कर सकता है।
5. जटिल टास्क को सबटास्क में बांटें और उन्हें अलग-अलग एजेंटों को सौंपें, जो जटिलताओं को नियंत्रित करने में मदद करता है, और मतभिरम की संभावना को भी कम करता है। क्योंकि हर एजेंट को एक छोटे से काम के लिये जिम्मेदार ठहराया जा सकता है।
6. LLM के इस्तेमाल से जुड़े जोखिमों और सीमाओं के बारे में जानकारी दें, जो प्रभावी जोखिम संचार यूज़र को संभावित समस्याओं के लिये तैयार कर सकता कर सकता है। यह नियंत्रित करने में मदद कर सकता है।
7. ऐसे API और यूज़र इंटरफ़ेस बनाएं, जो LLM के जिम्मेदार और सुरक्षित इस्तेमाल को प्रोत्साहित करें। इसमें कॉन्टेन्ट फ़िल्टर, संभावित अशुद्धियों के बारे में यूज़र की चेतावनियाँ और AI द्वारा जेनरेट की गई सामग्री की क्लायिंट लेबलिंग जैसे उपाय शामिल हैं।
8. वकिल के दौरान ही LLM को संभावित कमजोरियों से बचाने के लिये सुरक्षित कोडिंग तकनीके एवं दिशानिर्देशों को स्थापित करें।

उदाहरण हमले का परदिश्य

1. समाचार संगठन समाचार बनाने के लिये AI मॉडल का ज़्यादा इस्तेमाल करता है। एक दुर्भावनापूर्ण व्यक्ति इस अति-निरिभरता का फायदा उठाता है, AI को गुमराह करने वाली जानकारी देता है, जिससे गलत सूचनाएं फैलती हैं। AI ने अनजाने में ही सामग्री को प्लगियराइज़ (plagiarized) कर दिया, जिससे कॉपीराइट समस्याएँ पैदा हो गईं और संगठन में वशिक्षण कम हो गया।
2. सॉफ्टवेयर डेवलपमेंट टीम कोडिंग प्रक्रिया में तेज़ी लाने के लिये कोडेक्स जैसे AI सिस्टम का इस्तेमाल करती है। AI के सुझावों पर ज़्यादा भरोसा करने से सुरक्षा डफ़िलेट सेटिंग्स या सुरक्षित कोडिंग पद्धतियों के साथ असंगत सुझावों के कारण ऐप्लिकेशन में सुरक्षा संबंधी कमजोरियाँ आ जाती हैं।
3. एक सॉफ्टवेयर डेवलपमेंट फर्म डेवलपर्स की सहायता के लिये LLM का इस्तेमाल करती है। LLM एक गैर-मौजूद कोड लाइब्रेरी या पैकेज का सुझाव देता है, और एक डेवलपर, जो AI पर भरोसा करता है, अनजाने में एक दुर्भावनापूर्ण पैकेज को फर्म के सॉफ्टवेयर में इंटीग्रेट कर देता है। यह AI सुझावों को क्रॉस-चेकिंग करने के महत्व पर प्रकाश डालता है, खासकर तीसरे पक्ष के कोड या लाइब्रेरी को शामिल करते समय।

संदर्भ लिंक

1. Understanding LLM Hallucinations: Towards Data Science
2. How Should Companies Communicate the Risks of Large Language Models to Users?: Techpolicy
3. A news site used AI to write articles. It was a journalistic disaster: Washington Post
4. AI Hallucinations: Package Risk: Vulcan.io
5. How to Reduce the Hallucinations from Large Language Models: The New Stack
6. Practical Steps to Reduce Hallucination: Victor Debia



LLM10: मॉडल चोरी

यह दुरभावनापूर्ण व्यक्तियों या APTs द्वारा LLM मॉडल में अनधिकृत पहुंच और घुसपैठ को संदर्भित करती है। यह तब होता है जब LLM मॉडल (मूल्यवान बौद्धिकि संपदा होने के नाते), के साथ छेड़छाड़ की जाती है, भौतिक रूप से चोरी हो जाते हैं, कॉपी करिए जाते हैं या एक कार्यात्मक समकक्ष बनाने के लिए वज़न और पैरामीटर नकिले जाते हैं। LLM मॉडल की चोरी के प्रभाव में आरथिकी और ब्रांड प्रतिष्ठा खोना, प्रतस्थिप्रधात्मक लाभ में कमी, मॉडल का अनधिकृत उपयोग या मॉडल में मौजूद संवेदनशील जानकारी तक अनधिकृत पहुंच शामिल हो सकती है।

LLM की चोरी सुरक्षा के लिए एक महत्वपूर्ण चिह्नित का विषय है क्योंकि भाषा मॉडल तेजी से शक्तिशाली और प्रचलित होते जा रहे हैं। संगठनों और शोधकर्ताओं को अपने LLM मॉडल की सुरक्षा के लिए मज़बूत सुरक्षा उपायों को प्राथमिकता देनी चाहिए, जिससे उनकी बौद्धिकि संपदा की गोपनीयता और सत्यनिष्ठा बनी रहे। LLM मॉडल चोरी से जुड़े जोखिमों को कम करने और LLM पर निर्भर व्यक्तियों और संगठनों दोनों के हतियों की सुरक्षा करने के लिए एक व्यापक सुरक्षा ढांचे का इस्तेमाल करना, जिसमें ऐक्सेस नियंत्रण, एनक्रिप्शन और निरितर नियंत्रण शामिल है तथा महत्वपूर्ण है।

कमज़ोरी के सामान्य उदाहरण

- एक हमलावर कंपनी के कमज़ोर इंफ्रास्ट्रक्चर का फायदा उठा, कंपनी के नेटवर्क या ऐप्लिकेशन सुरक्षा सेटिंग में ग़लतफ़हमी के ज़रिए LLM मॉडल रपिंजिटिरी तक अनधिकृत पहुंच बनता है।
- अंदरूनी खतरे का परदिश्य जहां एक असंतुष्ट कर्मचारी कसी मॉडल या उससे जुड़ी सामग्री को लीक कर देता है।
- एक हमलावर API से सावधानी से तैयार करिए गए इनपुट और प्रॉम्प्ट इंजेक्शन तकनीकों का इस्तेमाल करके पूछताछ करता है, जिससे की शैडो मॉडल बनाने के लिए प्रयोग संख्या में आउटपुट प्राप्त होते हैं।
- एक दुरभावनापूर्ण हमलावर एक साइड-चैनल हमला करने के लिए LLM की इनपुट फ़िलिटरिंग तकनीकों को दरकनार कर सकता है और अंततः रमिट नियंत्रित संसाधन का उपयोग करके मॉडल के आर्किटिक्चर और उससे जुड़ी जानकारियां हासलि कर सकता है।
- मॉडल एक्स्ट्रैक्शन के अटैक वेक्टर में कसी खास विषय पर बड़ी संख्या में प्रॉम्प्ट्स के साथ LLM से पूछताछ करना शामिल है। इसके बाद LLM के आउटपुट का इस्तेमाल कसी दूसरे मॉडल को ठीक करने के लिए किया जा सकता है। हालाँकि, इस हमले के बारे में कुछ बातें ध्यान देने योग्य हैं:

 - हमलावर को बड़ी संख्या में लक्षित प्रोम्प्ट जेनरेट करने होंगे। अगर प्रोम्प्ट प्रयोग से नहीं है, तो LLM से मिलने वाले आउटपुट बेकार होंगे।
 - LLM के आउटपुट में कभी-कभी बेहूदा जवाब हो सकते हैं, मतलब हमलावर पूरे मॉडल को नकिलने में सक्षम

नहीं हो सकता क्योंकि कुछ आउटपुट बेतुके हो सकते हैं।

3. मॉडल एक्सट्रैक्शन के ज़रूरी कसी LLM को 100% बनाना संभव नहीं है। हालांकि, हमलावर एक अपूरण (partial) मॉडल बना सकता है।

6. फंक्शनल मॉडल रेप्लिकेशन के अटैक वेक्टर में सथिटिक प्रश्निक्षण डेटा (“सेलफ-इंस्ट्रक्ट” नामक दृष्टिकोण) जेनरेट करने के लिए प्रॉम्प्ट्स को लक्षित मॉडल पर उपयोग करना, फरि उसका इस्तेमाल कर कसी अन्य मूलभूत मॉडल को फ़ाइन-ट्यून करना शामलि है। यह उदाहरण 5 में इस्तेमाल कए गए पारंपरिक क्वेरी-आधारित (query-based) एक्सट्रैक्शन की सीमाओं को दरकनार कर देता है और शोध कार्य के लिये कसी अन्य LLM को प्रश्निक्षित करने के लिए सफलतापूर्वक इस्तेमाल करता है। हालांकि इस शोध के संदर्भ में, मॉडल रेप्लिकेशन कोई हमला नहीं है। इस दृष्टिकोण का इस्तेमाल एक हमलावर कसी मालकिना मॉडल को सार्वजनिक API की मदद से बनाने के लिए कर सकता है।

7. कसी चोरी हुए मॉडल का इस्तेमाल, शैडो मॉडल के तौर पर, प्रतकूल हमलों को स्टेज करने के लिए किया जा सकता है, जसिमें मॉडल में मौजूद संवेदनशील जानकारी तक अनाधिकृत पहुंच एवं एडवांस प्रॉम्प्ट इंजेक्शन को आगे बढ़ाने के लिए प्रतकूल इनपुट के साथ प्रयोग करना शामलि है, जसिका पता नहीं चलता है।

बचाव कैसे करें

1. LLM मॉडल रपिंजिटिरी और प्रश्निक्षण वातावरण तक अनधिकृत पहुंच को सीमित करने के लिए मज़बूत एक्सेस नियंत्रण (जैसे, RBAC और कम से कम वशिष्ठाधिकार का नियम) और मज़बूत प्रमाणीकरण (authentication) तंत्र लागू करें।

1. यह पहले तीन सामान्य उदाहरणों के लिए खास तौर पर सही है, जो अंदरूनी खतरों, गलत कॉन्फिगरेशन और/या इंफ्रास्ट्रक्चर के बारे में कमज़ोर सुरक्षा नियंत्रण के कारण इस जोखमि का कारण बन सकते हैं, जसिमें LLM मॉडल, वज़न और आर्किटिक्चर मौजूद हैं, जसिमें एक दुर्भावनापूर्ण व्यक्तवितावरण के अंदर या बाहर से घुसपैठ कर सकता है।

2. सप्लाई-चेन के हमलों को रोकने के लिए आपूरतकिरता प्रबंधन ट्रैकिंग, सत्यापन और निर्भरता की कमज़ोरियाँ महत्वपूर्ण विषय हैं।

2. नेटवर्क संसाधनों, आंतरिक सेवाओं और API तक LLM की पहुंच को प्रतिबंधित करें।

1. यह सभी सामान्य उदाहरणों के लिए खास तौर पर सही है क्योंकि यह अंदरूनी जोखमि और खतरों को कवर करता है, अंत में यह नियंत्रित करता है कि LLM एप्लिकेशन की पहुंच कहा तक है और इसलिए यह साइड-चैनल हमलों को रोकने के लिए एक तंत्र हो सकता है।

3. कसी भी संदर्भ या अनधिकृत व्यवहार का तुरंत पता लगाने और उसका जवाब देने के लिए, LLM मॉडल रपिंजिटिरी से संबंधित एक्सेस लॉग और गतविधियों की नियमित रूप से नगरानी करें और उनका ऑडिट करें।

4. इंफ्रास्ट्रक्चर में ऐक्सेस और डिप्लॉयमेंट नियंत्रणों को बेहतर बनाने के लिए, गवर्नेंस, ट्रैकिंग और कार्यप्रवाह की मंजूरी की मदद से MLOPs के डिप्लॉयमेंट को स्वचालित करें।

5. साइड-चैनल अटैक की वजह से प्रॉम्प्ट इंजेक्शन तकनीकों के जोखमि को कम करने के लिए नियंत्रण और



शमन रणनीतियां लागू करें।

6. API कॉल फ़िल्टर की दर सीमित कर, LLM ऐप्लिकेशन से डेटा में घुसपैठ के जोखमि को कम किया जा सकता है, या अन्य निगरानी प्रणालियों से (जैसे, DLP) नकिस की गतिविधिका पता लगाने के लिए तकनीकों को लागू किया जा सकता है।
7. नष्टिकृष्ण संबंधी प्रश्नों का पता लगाने और भौतिक सुरक्षा उपायों को मजबूत करने में मदद करने के लिए प्रतकूल एवं सुदृढ़ प्रशाक्षिण लागू करें।
8. LLM के जीवनचक्र में एम्बेडिंग और खोजने के चरणों में वॉटरमार्किंग फ्रेमवर्क लागू करें।

उदाहरण हमले का परदृश्य

1. एक हमलावर कसी कंपनी के LLM मॉडल भंडार तक अनधिकृत पहुंच प्राप्त करने के लिए उसके बुनियादी ढांचे में कमजोरियों का फायदा उठाता है। इसके बाद हमलावर मूल्यवान LLM मॉडलों में घुसबैठ करता है। जिसका इस्तेमाल वह प्रतिस्पृष्ठी भाषा प्रोसेसिंग सेवा शुरू करने या संवेदनशील जानकारी नकालने के लिए करता है, जिससे कंपनी को काफी आर्थिक नुकसान होता है।
2. एक असंतुष्ट कर्मचारी मॉडल या उससे संबंधित जानकारिया लीक कर देता है। इन जानकारियों के सार्वजनिक प्रदर्शन से हमलावरों के लिए ग्रे बॉक्स प्रतकूल हमला तथा संपत्ति की सीधा चोरी आसान हो गया।
3. एक हमलावर सावधानी से चुने गए इनपुट के साथ API से पूछताछ करता है और शैडो मॉडल बनाने के लिए प्रयाप्त संख्या में आउटपुट इकट्ठा करता है।
4. एक सुरक्षा नियंत्रण वफिलता सप्लाई चेन के भीतर मौजूद है जो मालकिना मॉडल जानकारी के डेटा लीक की ओर ले जाती है।
5. एक दुर्भावना वाला हमलावर साइड-चैनल हमला करने और अपने नियंत्रण के तहत रमिट नियंत्रित संसाधन पर मॉडल जानकारी पुनरप्राप्त करने के लिए इनपुट फ़िल्टरिंग तकनीकों और LLM की प्रस्तावना को बायपास करता है।

संदर्भ लिंक

1. Meta's powerful AI language model has leaked online: The Verge
2. Runaway LLaMA | How Meta's LLaMA NLP model leaked: Deep Learning Blog
3. AML.TA0000 ML Model Access: MITRE ATLAS
4. I Know What You See: Arxiv White Paper
5. D-DAE: Defense-Penetrating Model Extraction Attacks: Computer.org
6. A Comprehensive Defense Framework Against Model Extraction Attacks: IEEE
7. Alpaca: A Strong, Replicable Instruction-Following Model: Stanford Center on Research for Foundation Models (CRFM)

8. How Watermarking Can Help Mitigate The Potential Risks Of LLMs?: KD Nuggets





Team

Thank you to the OWASP Top 10 for LLM Applications version 1.1 Hindi Translation Contributors.

Version 1.1 Hindi Translation Contributors

Rachit Sood

Dhruv Agarwal

Rishi Sharma

REDACTED