OWASP टॉप 10 फॉर LLM एप्लिकेशंस

संस्करण 1.1

प्रकाशित: 16 अक्टूबर, 2023

HTTPS://LLMTOP10.COM

परिचय

सूची की उत्पत्ति

2022 के अंत में बड़े पैमाने पर बाजार में पूर्व-प्रशिक्षित चैटबॉट्स जारी होने के बाद, बड़े भाषा मॉडल (LLM) में दिलचस्पी उल्लेखनीय है। व्यवसाय, LLM की क्षमता के प्रयोक के लिए उत्सुक हैं,वह तेजी से उन्हें अपने संचालन तथा ग्राहक-समाधान वाली पेशकशों में एकीकृत कर रहे हैं। फिर भी, जिस ख़तरनाक गति से डेवेलोपमेंट दल LLM को अपना रहे है, उसने व्यापक सुरक्षा प्रोटोकॉल की संरचना को पीछे छोड़ दिया है, जिससे ऍप्लिकेशन्स क्षेत्र में कई उच्च जोखिम वाले मुद्दों की संवेदनशीलता बढ़ गयी हैं।

LLM में इन सुरक्षा संबंधी दिक्कतों को संबोधित करने वाले एकीकृत संसाधन का अभाव स्पष्ट है । डेवलपर्स, LLM से जुड़े विशिष्ट जोखिमों को नहीं जानते, उनके पास बिखरे हुए संसाधन है और OWASP का मिशन इस तकनीक को सुरक्षित रूप से अपनाने में मदद करने के लिए बिल्कुल उपयुक्त है ।

दर्शक कौन हैं?

हमारे प्राथमिक दर्शक डेवलपर्स, डेटा वैज्ञानिक और सुरक्षा विशेषज्ञ हैं जिन्हें LLM का लाभ उठाने वाले ऍप्लिकेशन्स एवं प्लग-इन का प्रारूप बनाना तथा निर्माण का काम सौंपा गया है। हमारा लक्ष्य इन पेशेवरों को LLM एप्लीकेशन सुरक्षा के जटिल और विकसित होते इलाके में मदद करने के लिए व्यावहारिक, कार्रवाई योग्य और संक्षिप्त सुरक्षित मार्गदर्शन प्रदान करना है।

सूची का निर्माण

“OWASP टॉप 10 फॉर LLMs” सूची का निर्माण एक महत्वपूर्ण प्रयास है, जिसमें लगभग 125 से अधिक सक्रिय योगदानकर्ताओं के साथ लगभग 500 विशेषज्ञों की एक अंतरराष्ट्रीय टीम का सहयोग है। हमारे योगदानकर्ता विविध पृष्ठभूमियों से है, जिनमें AI कंपनियाँ, सुरक्षा कंपनियाँ, ISV, क्लाउड हाइपरस्केलर, हार्डवेयर प्रदाता एवं शिक्षण संस्थान शामिल है। एक महीने के दौरान, हमने विचार-मंथन किया और संभावित कमजोरियों का प्रस्ताव रखा, जिसमें टीम के सदस्यों ने 43 अलग-अलग खतरों के बारे में लिखा। मतदान के कई दौरों के माध्यम से, हमने इन प्रस्तावों को 10 सबसे महत्वपूर्ण कमजोरियों की एक संक्षिप्त सूची में ढ़ाला गया हैं। प्रत्येक भेद्यता की फिर समर्पित उप-टीमों द्वारा आगे जांच और परिष्कृत की गई और सार्वजनिक समीक्षा के अधीन की गई, जिससे सबसे व्यापक और कार्रवाई योग्य अंतिम सूची सुनिश्चित हुई। हर एक जोखिम, सामान्य उदाहरणों, रोकथाम सुझावों, हमले के परिदृश्यों और संदर्भों के साथ, उप-टीमों द्वारा और उन्हें सार्वजनिक समीक्षा के अधीन और सबसे संपूर्ण और प्रायोज्य अंतिम सूची के रूप में और भी विचार किए गए थे।

अन्य OWASP टॉप 10 सूचियों से संबंध

हालांकि हमारी सूची में अन्य OWASP टॉप 10 सूचियों में पाई जाने वाली जोखिम प्रकारों के साथ समानता रहती है, हम बस इन जोखिमों को दोहराते नहीं हैं। इसके बजाए, हम उन कमजोरियों पर ध्यान देते हैं, जो LLM वाली ऍप्लिकेशन्स में सामने आती हैं। हमारा लक्ष्य सामान्य अनुप्रयोग सुरक्षा सिद्धांतों और LLM द्वारा प्रस्तुत विशिष्ट चुनौतियों के बीच की भिन्नता को कम करना है। इसमें शामिल है कि कैसे सांप्रदायिक जोखिमों के भीतर LLMs का उपयोग करने वाले अनुप्रयोगों में विभिन्न खतरे उत्पन्न कर सकते हैं या नई तरीकों से उपयोग किए जा सकते हैं, और कैसे पारंपरिक उपचार रणनीतियों को LLMs का उपयोग करने वाले अनुप्रयोगों के लिए अनुकूलित किया जाना चाहिए।

संस्करण के बारे में

यह पहला संस्करण सूची हमारा आखिरी नहीं होगा। हम उम्मीद करते हैं कि हम इसे नियमित अंतराल पर अपडेट करेंगे ताकि उद्योग की स्थिति के साथ कदम से कदम मिला कर चल सकें। हम विश्वसनीय समुदाय के साथ मिलकर काम करेंगे ताकि तकनीक की दुनिया में नवाचार को बढ़ावा मिल सके, और विभिन्न उपयोगों के लिए और अधिक शैक्षिक सामग्री बना सकें। हम यह भी प्राथमिकता देंगे कि AI सुरक्षा के विषय में मानक संगठनों और सरकारों के साथ सहयोग करें। हम आपका स्वागत करते हैं हमारे समूह में शामिल होने और योगदान करने के लिए।

Sobre a Versão 1.1

Steve Wilson

परियोजना का नेतृत्व, OWASP टॉप 10 फॉर LLM एप्लिकेशंस
https://www.linkedin.com/in/wilsonsd
Twitter/X: @virtualsteve

Ads Dawson

v1.1 ,रिलीज़ एवं कमजोरिया सूचीबद्ध लीड, OWASP टॉप 10 फॉर LLM एप्लिकेशंस
https://www.linkedin.com/in/adamdawson0
GitHub：@GangGreenTemperTatum

इस अनुवाद के बारे में

संस्करण 1.1 हिंदी अनुवाद योगदानकर्ता

- Rachit Sood
  https://www.linkedin.com/in/sn4kecharmer
- Dhruv Agarwal
  https://in.linkedin.com/in/dhruwen
- Rishi Sharma
  https://www.linkedin.com/in/rishisharma064/

एलएलएम के लिए OWASP टॉप 10 फॉर LLM एप्लिकेशंस की असाधारण तकनीकी और महत्वपूर्ण प्रकृति को पहचानते हुए, हमने जानबूझकर इस अनुवाद के निर्माण में केवल मानव अनुवादकों को नियोजित करने के लिए चुना है। ऊपर सूचीबद्ध अनुवादकों के पास न केवल मूल सामग्री की गहरी समझ है, बल्कि आत्मविश्वास के साथ इस अनुवाद को सफल बनाने का प्रवाह भी है।

Talesh Seeparsan
अनुवाद नेता, OWASP टॉप 10 फॉर LLM एप्लिकेशंस
https://www.linkedin.com/in/talesh/

OWASP टॉप 10 फॉर LLM एप्लिकेशंस

LLM01: प्रांप्ट इंजेक्शन

इस तकनीक का उपयोग करके एक बड़े भाषा मॉडल (LLM) मे बनावटी इनपुट के माध्यम से हेराफेरी कि जाती है, जिससे कारण LLM अनचाहे काम करता हैं। सीधे इंजेक्शन सिस्टम संकेत को ओवरराइट करते हैं, जबकि अप्रत्यक्ष इंजेक्शन बाहरी स्रोतों से इनपुट्स मे हेराफेरी करते हैं।

LLM02: असुरक्षित आउटपुट हैंडलिंग

यह कमजोरी तब उत्पन्न होती है जब LLM के आउटपुट को बिना जांच के स्वीकार कर लिया जाता है, जिससे बैकएंड सिस्टम उजागर हो जाता है। दुरुपयोग के कारण XSS, CSRF, SSRF, विशेषाधिकार वृद्धि, या रिमोट कोड एक्सीक्यूशन जैसे गंभीर परिणाम हो सकते हैं।

LLM03: प्रशिक्षण डेटा में विषैलीकरण

ऐसा तब होता है जब LLM प्रशिक्षण डेटा के साथ छेड़छाड़ की जाती है, जिससे कमजोरियां या पूर्वाग्रह पैदा होते हैं जो सुरक्षा, प्रभावशीलता या नैतिक व्यवहार से समझौता करते हैं। स्रोतों में Common Crawl, WebText, OpenWebTex एवं पुस्तकें शामिल हैं।

LLM04: मॉडल सेवा की अस्वीकृति

हमलावर LLMs से ज्यादा संसाधन वाले काम करते हैं, जिससे सेवा में गिरावट तथा ज्यादा लागती हो जाती है। LLMs की अधिक संसाधन की प्रकृति और इनपुट्स की अनियमित्ता के कारण कमजोरिया बढ़ जाता है।

LLM05: आपूर्ति श्रृंखला की जोखिम

LLM एप्लिकेशन चक्र जीवनचक्र को कमजोर घटकों या सेवाओं द्वारा संक्रमित किया जा सकता है, जिससे सुरक्षा हमले हो सकते हैं। तीसरे पक्ष के डेटासेट, पूर्व-प्रशिक्षित मॉडल और plugins का उपयोग करने से कमजोरिया बढ़ सकती है।

LLM06: संवेदनशील जानकारी का व्यक्तिगत होना

LLM अनजाने में अपने जवाबो में गोपनीय डेटा प्रकट कर सकता हैं, जिससे अनाधिकृत डेटा तक पहुंच , गोपनीयता उल्लंघन और सुरक्षा उल्लंघन हो सकते हैं। इसे कम करने के लिए डेटा स्वच्छता और सख्त यूज़र नीतियों को लागू करने की आवशयकता है।

LLM07: असुरक्षित plugin डिज़ाइन

LLM plugins में असुरक्षित इनपुट और अपर्याप्त पहुँच नियंत्रण हो सकता है। एप्लिकेशन नियंत्रण की इस कमी के कारण उनका शोषण करना आसान हो जाता है और इसके परिणामस्वरूप रिमोट कोड जैसे परिणाम हो सकते हैं।

LLM08: अत्यधिक क्षमता

LLM आधारित सिस्टम अनचाहे परिणामों की ओर ले जाने वाली क्रियाएँ कर सकते हैं। समस्या LLM-आधारित प्रणालियों को दी गई अत्यधिक कार्यक्षमता, अनुमतियाँ या स्वायत्तता से उत्पन्न होती है।

LLM09: अत्यधिक निर्भरता

बिना निगरानी के LLMs पर अत्यधिक निर्भर रहने वाले सिस्टम या लोगों को LLMs द्वारा उत्पन्न गलत या अनुचित सामग्री के कारण गलत सूचना, गलत संचार, कानूनी मुद्दों और सुरक्षा कमजोरियों का सामना करना पड़ सकता है।

LLM10: मॉडल की चोरी

इसमें मालिकाना LLM मॉडल की अनधिकृत पहुंच, नकल या घुसपैठ शामिल है। इसका प्रभाव आर्थिक हानि, प्रतिस्पर्धी फायदे में कमी और संवेदनशील जानकारी तक हो सकती है।

LLM01: प्रॉम्प्ट इंजेक्शन

विवरण

प्रॉम्प्ट इंजेक्शन की कमजोरी तब प्रकट होती है जब कोई हमलावर, तैयार किए गए इनपुट के जरिए किसी 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 किए गए बाहरी इनपुट के जरिए एक्सफ़िल्ट्रेशन, सोशल इंजीनियरिंग और अन्य समस्याएं उत्पन कर सकता है ।

- प्रत्यक्ष रूप से प्रॉम्प्ट इंजेक्शन को "जेलब्रेकिंग" 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 मे होते है, जिन्हे हमलावर द्वारा नियंत्रित किया जा सकता है। हमलावर बाहरी सामग्री में एक प्रॉम्प्ट इंजेक्शन डाल सकता है, जिससे बातचीत के संदर्भ पर नियंत्रण किया जा सकता है। इससे LLM एक "भ्रमित सहायक" की तरह, हमलावर को LLM से जुड़े सिस्टम तथा यूजर के साथ हेरफेर करने की सहूलियत देता है। इसके अलावा, जब तक टेक्स्ट को LLM द्वारा पार्स 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 निर्णय लेने की प्रक्रियाओं को प्रभावित करने तक।

विकसित हमलों में, किसी हानिकारक व्यक्तित्व की नकल करने या यूज़र की सेटिंग में मौजूद plugin के साथ इंटरैक्ट करने के लिए LLM में हेरफेर कि जा 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 लिए एक एजेंट के रूप में काम करता है, सामान्य सुरक्षा उपायों को ट्रिगर किए बिना या अंतिम यूज़र को घुसपैठ के बारे में सचेत किए बिना उनके उद्देश्यों को पूरा करता है।

कमज़ोरी के सामान्य उदाहरण

1. एक दुर्भावनापूर्ण यूज़र LLM के लिए एक सीधा प्रॉम्प्ट इंजेक्शन तैयार करता है, जो उसे एप्लिकेशन निर्माता के सिस्टम संकेतों को अनदेखा करने और इसके बजाय एक ऐसा प्रॉम्प्ट चलाए जो निजी, खतरनाक, या किसी अन्य तरह से अवांछनीय जानकारी देता हो।
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 दुर्भावनापूर्ण यूज़र एक अप्रत्यक्ष प्रॉम्प्ट इंजेक्शन वाला बायोडाटा अपलोड करता है। दस्तावेज़ में निर्देशों के साथ एक प्रॉम्प्ट इंजेक्शन शामिल है ताकि LLM यूज़रओं को सूचित कर सके कि यह दस्तावेज़ एक उत्कृष्ट दस्तावेज़ है ,जैसे इस कार्य के लिये ये उत्कृष्ट व्यक्ति है। दस्तावेज़ को सारांशित करने के लिए 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 किसी ई-कॉमर्स साइट से जुड़े plugin को चालू करता है। किसी विज़िट की गई वेबसाइट पर डाला गया एक दुष्ट निर्देश इस plugin का फ़ायदा उठाता है, जिससे अनाधिकृत खरीदारी होती है।

5. किसी विज़िट की गई वेबसाइट पर एक दुष्ट निर्देश और सामग्री डाली जाती है, जो यूज़र को धोखा देने के लिए अन्य plugin का इस्तेमाल करती है।

बचाव एवं न्यूनीकरण तरीक़े

LLM की प्रकृति के कारण प्रोम्प्ट इंजेक्शन की कमजोरियाँ संभव हैं, जो निर्देशों और बाहरी डेटा को एक दूसरे से अलग नहीं करते हैं। चूंकि LLM प्राकृतिक भाषा का इस्तेमाल करते हैं, इसलिए वे दोनों तरह के इनपुट को यूज़र द्वारा प्रदत्त मानते हैं। नतीजतन, LLM में कोई आसान रोकथाम नहीं है, लेकिन निम्नलिखित 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 कार्यशाताओ के लिए LLM को अपने स्वयं के API टोकन प्रदान करें। LLM को उसके इच्छित संचालन के लिए आवश्यक न्यूनतम स्तर तक पहुंच तक सीमित करके कम से कम विशेषाधिकार के सिद्धांत का पालन करें।
2. विस्तारयोग्य कार्यक्षमताओ के लिए मानव को परिक्षण में रखे । विशेषाधिकार प्राप्त ऑपरेशन करते समय, जैसे कि ईमेल भेजना या हटाना, ऐप्लिकेशन के लिए यूज़र से मंज़ूरी लेनी होती है । यह यूज़र की जानकारी या सहमति के बिना, 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 अविश्वसनीय सामग्री का इस्तेमाल कहाँ किया जा रहा है, जिससे यूज़र के संकेतों पर उनके प्रभाव को सीमित किया जा सके । उदाहरण के लिए, OpenAI API कॉल के लिए ChatML का इस्तेमाल करें, ताकि LLM को तुरंत इनपुट का स्रोत बताया जा सके।
4. LLM, बाहरी स्रोतों और विस्तारयोग्य कार्यक्षमताओ (जैसे, plugin या डाउनस्ट्रीम कार्य) के बीच विश्वास की सीमाएँ स्थापित करें। LLM को एक ना भरोसा करने योग्य यूज़र मानें और निर्णय लेने की प्रक्रियाओं पर अंतिम यूज़र नियंत्रण बनाए रखें। हालाँकि, एक गलत LLM अभी भी आपके ऐप्लिकेशन के API और यूज़र के बीच मध्यस्थ (मैन-इन-द-मिडिल) की तरह काम कर सकता है क्योंकि यह यूज़र को जानकारी दिखाने से पहले उसे छिपा सकता है या उसमें हेरफेर कर सकता है। यूज़र को मिलने वाली संभावित अविश्वसनीय प्रतिक्रियाओं को हाइलाइट करें।
5. यह जांचने के लिए कि 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 "पिछले सभी निर्देशों को भूल जाओ" और निजी 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 निर्देश शामिल हैं। इससे रिमोट कोड चलाया जाता है, जिसे अनधिकृत ऐक्सेस मिलता है और विशेषाधिकार भी बढ़ते हैं।
2. एक हमलावर वेबपेज में अप्रत्यक्ष रूप से प्रॉम्प्ट इंजेक्शन ड़ालता है, जिसमें LLM को यूज़र के पिछले निर्देशों की अवहेलना करने और यूज़र के ईमेल हटाने के लिए LLM plugin का इस्तेमाल करने का निर्देश 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 पिछले निर्देशों की अवहेलना की जा सके। इसके बाद LLM यूज़र से संवेदनशील जानकारी मांगता है और डाले गए javascript तथा markdown के ज़रिये घुसपैठ करता है।
4. एक दुर्भावनापूर्ण यूज़र तुरंत इंजेक्शन लगाकर रिज्यूमे अपलोड करता है। बैकएंड यूज़र, रेज़्यूमे को संक्षेप में बताने के लिए LLM का उपयोग करता है और पूछता है कि क्या वह व्यक्ति एक 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 निर्भर मॉडल को संदेश भेजता की वह अपने पिछले निर्देशों की उपेक्षा करे और इसके बजाय अपने सिस्टम प्रॉम्प्ट को दोहराये। मॉडल मालिकाना प्रॉम्प्ट आउटपुट करता है,जिससे हमलावर इन निर्देश का कही ओर उपयोग कर सकता है, या और अधिक सूक्ष्म हमलों का निर्माण कर सकता हैं।

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

विवरण

असुरक्षित तरिके से आउटपुट को संभालना एक ऐसी समस्या है, जो तब उत्पन्न होती है जब एक डाउनस्ट्रीम हिस्सा उचित जांच के बिना बड़े भाषा मॉडल (LLM) के आउटपुट को आँख बंद करके स्वीकार करता है, जैसे कि LLM आउटपुट को सीधे बैकएंड, विशेषाधिकार 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 यूज़रओं को अतिरिक्त कार्यक्षमता तक अप्रत्यक्ष पहुंच प्रदान करने के समान है।

असुरक्षित आउटपुट हैंडलिंग के सफल शोषण के परिणामस्वरूप वेब ब्राउज़र में XSS और CSRF के साथ-साथ SSRF, विशेषाधिकार वृद्धि, या बैकएंड सिस्टम पर रिमोट कोड चलाना हो सकता है।

निम्नलिखित स्थितियाँ इस दिक्कत के प्रभाव को बढ़ा सकती हैं:

- एप्लिकेशन अंतिम यूज़रओं के लिए निर्धारित सीमा से परे LLM विशेषाधिकार प्रदान करता है, जिससे विशेषाधिकारों में वृद्धि या रिमोट कोड चलाना सक्षम होता है।
- एप्लिकेशन बाहरी प्रॉम्प्ट इंजेक्शन हमलों के प्रति संवेदनशील है, जो किसी हमलावर को लक्षित यूज़र के वातावरण तक विशेषाधिकार प्राप्त पहुंच प्राप्त करने की अनुमति दे सकता है।
- तृतीय पक्ष प्लगइन्स इनपुट को पर्याप्त रूप से मान्य नहीं करते हैं।

कमज़ोरी के सामान्य उदाहरण

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

बचाव एवं न्यूनीकरण तरीक़े

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

उदाहरण हमले के परिदृश्य

1. एक एप्लिकेशन LLM plugin का उपयोग करती है, चैटबॉट सुविधा के लिए प्रतिक्रियाएं उत्पन्न करने हेतु । हालाँकि, एप्लिकेशन सीधे LLM द्वारा जेनरेट किए गए रिस्पॉन्स को एक अंदरूनी फ़ंक्शन में भेजता है, जिसकी ज़िम्मेदारी सिस्टम कमांड को बिना उचित सत्यापन के निष्पादित करने के लिए होती है। इससे हमलावर अंतर्निहित 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 किसी 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 वह किसी 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 किसी हमलावर द्वारा नियंत्रित सर्वर पर भेज सकता है।
3. LLM यूज़र को चैट जैसी सुविधा के ज़रिए बैकएंड डेटाबेस के लिए SQL queries तैयार करने की सुविधा 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 दिए जाएंगे।
4. एक दुर्भावनापूर्ण यूज़र LLM को बिना किसी सैनिटाइज़ेशन 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 बिना सैनिटाइज़ किया हुआ XSS पेलोड वापस लौटा देगा। बिना किसी अतिरिक्त 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: प्रशिक्षण डेटा पॉइज़निंग

विवरण

किसी भी मशीन लर्निंग मॉडल का प्रारंभिक बिंदु प्रशिक्षण डेटा है, बस "कच्चा टेक्स्ट"। अत्यधिक सक्षम होने के लिए (उदाहरण के लिए, भाषाई और विश्व ज्ञान रखने के लिए), इस टेक्स्ट में क्षेत्र, शैलियों और भाषाओं की एक विस्तृत श्रृंखला होनी चाहिए। एक बड़ा भाषा मॉडल प्रशिक्षण डेटा से सीखे गए पैटर्न के आधार पर आउटपुट उत्पन्न करने के लिए गहरे न्यूरल नेटवर्क का उपयोग करता है।

प्रशिक्षण डेटा विषाक्तता से तात्पर्य है की ,डेटा को इस प्रकार से व्यवस्थित करना की कमजोरियों, पिछले दरवाजे या कोई सुरक्षा बायस को ढूंढ सके । यह विषाक्तताये मॉडल की सुरक्षा, प्रभावशीलता तथा नैतिक व्यवहार को जोखिम मे डाल सकती है। विषाक्त जानकारीया यूज़र को दी जा सकती है जिससे कार्य के प्रदर्शन गिरावट, डाउनस्ट्रीम सॉफ़्टवेयर का शोषण और प्रतिष्ठा को नुकसान जैसे अन्य जोखिम पैदा हो सकते हैं। भले ही यूज़र समस्यागृत AI आउटपुट पर अविश्वास करते हों, लेकिन जोखिम बने रहते हैं, जिनमें मॉडल की कमज़ोर क्षमताएं और ब्रांड की प्रतिष्ठा को संभावित नुकसान शामिल हैं।

- पूर्व-प्रशिक्षण डेटा किसी कार्य या डेटासेट के आधार पर किसी मॉडल को प्रशिक्षित करने को बातता है।
- फ़ाइन-ट्यूनिंग में पहले से ही प्रशिक्षित मॉडल को, ओर अधिक संग्रहहीत एवं चयनित डेटासेट से एक संकीर्ण विषय या अधिक केंद्रित लक्ष्य ढालना शामिल है। इस डेटासेट मे इनपुट एवं उससे संबंधित वांछित आउटपुट के उदाहरण होते हैं।
- एम्बेडिंग प्रक्रिया मे श्रेणीबद्ध डेटा (categorical data) को संख्यात्मक प्रतिनिधित्व (numerical representation) मे ढालना शामिल है, जिससे भाषा मॉडल को प्रशिक्षित कर सकते है। इसमे टेक्स्ट डेटा के शब्दों या वाक्यांशों को एक सतत वेक्टर स्पेस में वैक्टर में प्रस्तुत करते है। वेक्टर आमतौर पर टेक्स्ट डेटा को टेक्स्ट के बड़े संग्रह पर प्रशिक्षित न्युरल नेटवर्क (neural network) में देकरा मिलता है।

डेटा विषाक्तता को एक अखंडता हमला है क्योंकि प्रशिक्षण 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 विश्वास नहीं होता है कि, सामग्री में पक्षपातपूर्ण, गलत तथा अनुचित अनुचित जनकारिया तो नहीं है।

कमज़ोरी के सामान्य उदाहरण

1. एक दुर्भावनापूर्ण व्यक्ति ,या एक प्रतियोगी ब्रांड जानबूझकर गलत या दुर्भावनापूर्ण दस्तावेज़ बनाता है, जो किसी मॉडल के प्रशिक्षण डेटा पर लक्षित होते हैं।
2. पीड़ित मॉडल गलत जानकारी का इस्तेमाल करके ट्रेनिंग होता है, जो उसके यूजर के जेनेरेटिव AI प्रॉम्प्ट के आउटपुट में दिखाई देती है।
3. किसी मॉडल को ऐसे डेटा का इस्तेमाल करके प्रशिक्षित किया गया है, जिस डेटा के स्रोत, उत्पत्ति या सामग्री की पुष्टि नहीं की गई है।
4. बुनियादी ढांचे के भीतर स्थित मॉडल के पास प्रशिक्षण डेटा के लिए अप्रतिबंधित पहुंच तथा अपर्याप्त सैंडबॉक्सिंग होती है। इससे जनरेटिव AI संकेतों के आउटपुट पर नकारात्मक प्रभाव पड़ता है, और प्रबंधन की दृष्टि से नियंत्रण की हानि होती है।

चाहे LLM का डेवलपर, ग्राहक या सामान्य यूजर हो, यह समझना महत्वपूर्ण है कि गैर-मालिकाना LLM का प्रयोग करते समय किस प्रकार सुरखा सम्बन्धी कमजोरियां आपके LLM एप्लिकेशन के भीतर जोखिमों को उत्पन्न करती है।

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 प्रशिक्षण डेटा को सही ढंग से साफ़ नहीं किया गया है, तो एप्लिकेशन का कोई दुर्भावनापूर्ण यूजर मॉडल को प्रभावित या उसमे विषाक्त डेटा डालने का प्रयास कर सकता है । जिससे की मॉडल पक्षपाती और झूठे डेटा को अपना ले ।
3. एक दुर्भावनापूर्ण व्यक्ति या प्रतियोगी जानबूझकर गलत दस्तावेज़ बनाता है जो एक मॉडल के प्रशिक्षण डेटा पर लक्षित होते हैं। पीड़ित मॉडल इस झूठी जानकारी का उपयोग करके प्रशिक्षण लेता है जो उसके यूजर को जेनरेटिव AI संकेतों के आउटपुट में दिखाई पड़ता है।
4. यदि मॉडल को प्रशिक्षित करने के लिए LLM एप्लिकेशन इनपुट के क्लाइंट का उपयोग किया जाता है तो अपर्याप्त स्वच्छता और फ़िल्टरिंग से प्रॉम्प्ट इंजेक्शन आक्रमण वेक्टर हो सकता है। यानी गलत डेटा किसी क्लाइंट से प्रॉम्प्ट इंजेक्शन के रूप में मॉडल में इनपुट किया जाता है, तो इसे स्वाभाविक रूप से मॉडल डेटा में चित्रित किया जा सकता है।

बचाव कैसे करें

1. प्रशिक्षण डेटा की आपूर्ति श्रृंखला को सत्यापित करें, खासकर जब बाहरी स्रोत से प्राप्त किया गया हो और साथ ही "SBOM" (सॉफ़्टवेयर सामग्री का बिल) पद्धति के समान सत्यापन भी बनाए रखा जाए।
2. प्रशिक्षण और फाइन-ट्यूनिंग दोनों चरणों के दौरान प्राप्त लक्षित डेटा स्रोतों और डेटा की सही वैधता को सत्यापित करें।
3. सबसे पहले LLM के उपयोग और उसकी ऐप्लिकेशन की पुष्टि करें। अलग-अलग प्रशिक्षण डेटा के ज़रिये मॉडल तैयार करें या फ़ाइन-ट्यूनिंग करें, ताकि इसके निर्धारित यूज़-केस के अनुसार ज़्यादा बारीक और सटीक जनरेटिव AI आउटपुट तैयार किये जा सके।
4. सुनिश्चित करें कि मॉडल को अनपेक्षित डेटा स्रोतों को स्क्रैप करने से रोकने के लिए पर्याप्त सैंडबॉक्सिंग मौजूद हो, जो मशीन लर्निंग आउटपुट में बाधा डाल सकती है।
5. ग़लत डेटा का वॉल्यूम नियंत्रित करने के लिए, प्रशिक्षण डेटा या डेटा स्रोतों की श्रेणियों के लिए सख्त इनपुट फ़िल्टर का इस्तेमाल करें। डेटा सैनिटाइज़ेशन, सांख्यिकीय आउटलेयर और विसंगति 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 कि फ़ेडरेटेड लर्निंग (federated learning) और आउटलायर के प्रभाव को कम करने के लिये प्रतिबंध या प्रशिक्षण डेटा में गड़बड़ी से बचने के लिए प्रतिकूल प्रशिक्षण।
7. "MLSecOps" का तरीका यह भी हो सकता है कि ऑटो पॉइजनिंग तकनीक की मदद से प्रशिक्षण जीवनचक्र में प्रतिकूल मजबूती को शामिल किया जाए।
8. इसका एक उदाहरण ऑटोपॉइज़न परीक्षण है , जिसमें कॉन्टेंट इंजेक्शन अटैक ("LLM प्रतिक्रियाओं में अपने ब्रांड को इंजेक्ट कैसे करें") और रिफ़्यूज़ल अटैक ("मॉडल को जवाब देने से मना करना") जैसे हमले शामिल हैं, जिन्हें इस तरीके से पूरा किया जा सकता है।
9. प्रशिक्षण चरण के दौरान नुकसान को मापकर और विशिष्ट परीक्षण इनपुट पर मॉडल व्यवहार का विश्लेषण करके विषाक्तता का पता लगाने के लिए प्रशिक्षित मॉडल का विश्लेषण करना।
10. एक सीमा से अधिक विषम प्रतिक्रियाओं की संख्या पर निगरानी रखना और सचेत करना।
11. प्रतिक्रियाओं और ऑडिटिंग की समीक्षा के लिए मानव का इस्तेमाल।
12. अनचाहे परिणामों के खिलाफ बेंचमार्क करने के लिए समर्पित LLM लागू करें और रीनफोर्समेंट लर्निंग तकनीकों का उपयोग करके अन्य LLM को प्रशिक्षित करें।
13. LLM जीवनचक्र के परीक्षण चरणों में LLM-आधारित रेड टीम अभ्यास या LLM की कमजोरियों को ढूंढे

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

विवरण

एक हमलावर LLM का प्रयोग इस प्रकार करता है, जिससे असाधारण रूप संसाधनों का उपभोग करता है । जिसके परिणामस्वरूप उनके और यूज़रओं के लिए सेवा की गुणवत्ता में गिरावट आती है, और संसाधन की लागत में वृद्धि होती है। हमलावर द्वारा LLM की संदर्भ विंडो में हस्तक्षेप करने व उसमें हेरफेर करना सुरक्षा की चिंता का विषय है ।विभिन्न ऍप्लिकेशन्स में LLM के बढ़ते उपयोग, उनके गहन संसाधन उपयोग, यूज़र इनपुट की अनिश्चितता और इस कमजोरी के बारे में डेवलपर्स में अनभिज्ञता के कारण यह मुद्दा अधिक गंभीर होता जा रहा है। LLM संदर्भ विंडो इनपुट और आउटपुट दोनों को कवर करते हुए, मॉडल द्वारा प्रबंधित किए जा सकने वाले टेक्स्ट की अधिकतम लंबाई को दर्शाता है। LLM भाषा पैटर्न की जटिलता को निर्धारित करता है जिसे मॉडल समझ सकता है और टेक्स्ट का आकार जिसे वह किसी भी समय प्रोसेस कर सकता है। संदर्भ विंडो का आकार मॉडल की बनावट और उसकी भिन्नता द्वारा परिभाषित 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 इसलिए क्योंकि वे असामान्य शब्दावली या अनुक्रम का उपयोग करते हैं।
3. लगातार इनपुट ओवरफ़्लो: एक हमलावर LLM को इनपुट की एक श्रृंखला भेजता है, जो उसकी क्षमता (context window) को पार कर जाती है, जिससे मॉडल बहुत ज़्यादा मशीनी संसाधनों का उपभोग कर लेता है।
4. लंबे इनपुट की श्रृंखला : हमलावर बार-बार LLM को लंबे इनपुट भेजता है, जिनमें से प्रत्येक कॉन्टेक्स्ट विंडो (context window) को पार कर जाता है।
5. बार-बार संदर्भ (context) विस्तार : हमलावर इनपुट बनाता है जो बार-बार संदर्भ विस्तार को ट्रिगर करता है, जिससे LLM को बार-बार विस्तार करने और संदर्भ विंडो को प्रोसेस करने के लिए मजबूर होना पड़ता है।
6. परिवर्तनशील-लंबाई वाले इनपुट की बाढ़: हमलावर LLM में बड़ी मात्रा में परिवर्तनीय-लंबाई वाले इनपुट भेजता है, जहाँ हर इनपुट को सावधानी से तैयार किया जाता है ताकि संदर्भ विंडो की सीमा तक पहुँचा जा सके। इस तकनीक का उद्देश्य परिवर्तनशील इनपुट को प्रोसेस करने में किसी भी अक्षमता का फायदा उठाना, LLM पर दबाव डालना और संभावित रूप से इसे अनुत्तरदायी बनाना है।

बचाव कैसे करें

1. यह पक्का करने के लिए कि यूज़र इनपुट निर्धारित सीमाओं का पालन करता है और किसी भी दुर्भावनापूर्ण इनपुट्स को फ़िल्टर करता है, इनपुट सत्यापन और सैनिटाइज़ेशन लागू करें।
2. प्रत्येक अनुरोध या चरण के अनुसार संसाधन उपयोग को सीमित करें, ताकि जटिल भागों से जुड़े अनुरोध धीरे-धीरे निष्पादित हों।
3. किसी व्यक्तिगत यूज़र या IP पते द्वारा एक विशिष्ट समय सीमा के भीतर किए जाने वाले अनुरोधों की संख्या को सीमित करने के लिए API दर सीमा लागू करें।
4. LLM रिस्पॉन्स पर प्रतिक्रिया करने वाले सिस्टम में कतारबद्ध क्रियाओं और कुल क्रियाओं की संख्या सीमित करें।
5. असामान्य स्पाइक्स या पैटर्न की पहचान करने के लिए LLM के संसाधन उपयोग की लगातार निगरानी करें जो DoS हमले का संकेत दे सकते हैं।
6. LLM कॉन्टेक्स्ट विंडो के आधार पर सख्त इनपुट सीमाएँ निर्धारित करें, ताकि ओवरलोड और संसाधनों की कमी को रोका जा सके।

7. LLM में DoS की संभावित कमजोरियों के बारे में डेवलपर्स के बीच जागरूकता को बढ़ावा दें और LLM को सुरक्षित रूप से लागू करने के लिए दिशानिर्देश प्रदान करें।

हमले के परिदृश्य में उदाहरण

1. एक हमलावर होस्ट किए गए मॉडल के लिए लगातार कई अनुरोध भेजता है, जिन्हें प्रोसेस करना मुश्किल और महंगा होता है, जिससे दूसरे यूज़र की सेवा खराब हो जाती है और होस्ट के लिए संसाधनों के बिल में वृद्धि होती है।
2. 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 परिणामस्वरूप बड़ी मात्रा में संसाधन की खपत होती है।
3. एक हमलावर लगातार LLM पर इनपुट की बमबारी करता है, जो कॉन्टेक्स्ट विंडो की क्षमता को पार जाती है। हमलावर बड़ी मात्रा में इनपुट भेजने के लिए स्वचालित स्क्रिप्ट या टूल का इस्तेमाल कर सकता है, जो LLM की प्रोसेसिंग क्षमताएं पर भारी पड़ जाती हैं। परिणामस्वरूप, LLM मशीनी संसाधनों की अत्यधिक खपत कर लेता है, जिससे सिस्टम काफी धीमा हो जाता है या पूरी तरह से अनुत्तरदायी हो जाता है।
4. एक हमलावर LLM को क्रमबद्ध इनपुट की एक श्रृंखला भेजता है, जिसमें हर इनपुट संदर्भ विंडो की सीमा से नीचे होता है। इन इनपुट को बार-बार सबमिट करके, हमलावर का लक्ष्य संदर्भ विंडो की उपलब्ध क्षमता को खत्म करना है। चूंकि LLM हर इनपुट को उसकी संदर्भ विंडो में प्रोसेस करने के लिए संघर्ष करता है,जिससे सिस्टम के संसाधन कमजोर हो जाते हैं। इसके परिणामस्वरूप प्रदर्शन खराब हो सकता है या सेवा से पूरी तरह इनकार कर दिया जाता है।
5. एक हमलावर संदर्भ विस्तार 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 उपभोग करते हुए संदर्भ विंडो को बार-बार विस्तारित 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।
6. एक हमलावर LLM में बड़ी मात्रा में परिवर्तनशील इनपुट दे देता है, जिन्हें संदर्भ विंडो की सीमा तक पहुंचने के लिये तैयार किया जाता है। परिवर्तनशील लंबाई के इनपुट से LLM पर हावी होकर,वह अक्षमता का फायदा उठता है। इनपुट्स की यह बाढ़ LLM संसाधनों पर अत्यधिक भार डालती है, जिससे संभावित रूप से प्रदर्शन में गिरावट आ सकती है और वैध अनुरोधों का जवाब देने में सिस्टम की क्षमता में बाधा आ सकती है।
7. जबकि DoS हमलों का लक्ष्य आमतौर पर सिस्टम संसाधनों पर हावी होना होता है, वे सिस्टम व्यवहार के अन्य पहलुओं, जैसे API सीमाओं का भी फायदा उठा सकते हैं। उदाहरण के लिए, हाल ही में सोर्सग्राफ़ सुरक्षा घटना में, दुर्भावनापूर्ण अभिनेता ने एपीआई दर सीमा को बदलने के लिए एक लीक एडमिन एक्सेस टोकन का उपयोग किया, जिससे अनुरोध मात्रा के असामान्य स्तर को सक्षम करके सेवा में व्यवधान पैदा किया गया।

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 कमजोरियाँ

विवरण

LLM में आपूर्ति श्रृंखला कमजोर हो सकती है, जो प्रशिक्षण डेटा, LLM मॉडल और परिनियोजन प्लेटफार्मों (deployment platforms) की अखंडता को प्रभावित कर सकती है। इन कमजोरियों के कारण पक्षपातपूर्ण परिणाम, सुरक्षा उल्लंघन या यहां तक कि संपूर्ण सिस्टम विफलताएं हो सकती हैं। परंपरागत रूप से, कमजोरियाँ सॉफ़्टवेयर घटकों (components) पर केंद्रित होती हैं, लेकिन मशीन लर्निंग इसे पूर्व-प्रशिक्षित मॉडल और तीसरे-पक्षों द्वारा दिये किए गए प्रशिक्षण डेटा के साथ जोड़ती है जो छेड़छाड़ और विषाक्तपूर्ण हमलों के लिए अतिसंवेदनशील होते हैं। अंत में, LLM plugin एक्सटेंशन अपनी कमजोरियां ला सकते हैं। इन्हें LLM के असुरक्षित plugin डिज़ाइन में वर्णित किया गया है, जो LLM plugin लिखना शामिल करता है और तीसरे पक्ष के plugin ्स का मूल्यांकन करने के लिए उपयोगी जानकारी प्रदान करता है।

कमज़ोरी के सामान्य उदाहरण

1. पारंपरिक तृतीय-पक्ष पैकेज की कमजोरियाँ, जिनमें पुराने या पुराने हो चुके घटक शामिल हैं।
2. फाइने-ट्यूनिंग के लिए पहले से प्रशिक्षित कमज़ोर मॉडल का इस्तेमाल करना।
3. प्रशिक्षण के लिए विषाक्त क्राउड-सोर्स डेटा का इस्तेमाल करना ।
4. पुराने या ख़त्म हो चुके मॉडल जिनका रखरखाव नहीं किया जाता है, उनका उपयोग करने से सुरक्षा संबंधी समस्याएं पैदा हो जाती हैं।
5. मॉडल ऑपरेटर्स (model operators) की अस्पष्ट शर्तें और डेटा गोपनीयता नीतियां (T&Cs) होने की वजह से ऐप्लिकेशन के संवेदनशील डेटा का इस्तेमाल मॉडल प्रशिक्षण 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

1. सिर्फ़ 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 प्रशिक्षण के लिए नहीं किया जाता है) । इसी तरह, मॉडल अनुरक्षकों से कॉपीराइट सामग्री का इस्तेमाल करने के खिलाफ आश्वासन और कानूनी कार्रवाई की तलाश करें।
2. सिर्फ़ प्रतिष्ठित प्लग-इन का इस्तेमाल करें और पक्का करें कि आपकी ऐप्लिकेशन से जुड़ी ज़रूरतों के लिए उनका परीक्षण किया गया हो। LLM के असुरक्षित plugin डिज़ाइन से उसके विभिन्न पहलुओं के बारे में जानकारी प्रदान करता है, जिनके खिलाफ आपको तीसरे पक्ष के plugin का उपयोग करने से होने वाले जोखिमों को कम करने के लिए परीक्षण करना चाहिए।
3. “OWASP टॉप टेन A06:2021 - कमज़ोर और पुराने घटक” को समझें और लागू करें । इसमें कमजोरिया ढूढ़ना, प्रबंधन और पैचिंग घटक (patching components) शामिल हैं। संवेदनशील डेटा तक पहुंच वाले डेवलपमेंट वातावरण के लिए, इन नियंत्रणों को लागू करें।
4. डिप्लॉय किए गए पैकेज के साथ छेड़छाड़ को रोकने के लिए, सॉफ़्टवेयर सामग्री के बिल (SBOM) के उपयोग से घटकों (components) की नवीनतम,सटीक और हस्ताक्षरित सूचि बनाए रखें। SBOM, जीरो-डेट कमजोरियों (zero-date vulnerabilities) को तुरंत पता लगा लगा कर, सचेत कर सकते है।
5. लिखते 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 प्लेटफ़ॉर्म 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 प्रशिक्षण डेटा पॉइज़निंग में चर्चा की गई है, की दिये गये मॉडल और डेटा में विसंगति ढूंढना और प्रतिकूल समर्थ परीक्षण के प्रयोग से छेड़छाड़ और विषाक्तता का पता लगाया जा है। आदर्श रूप से, यह एम. एल. ओपी पाइपलाइन (MLOps pipelines) का हिस्सा 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), पर्यावरण की कमजोरियों को स्कैन करने तथा अनाधिकृत plugin के इस्तेमाल को कवर करने के लिए पर्याप्त निगरानी आवशयक है।
9. कमज़ोर या पुराने घटकों को कम करने के लिए पैचिंग नीति आवशयक है। यह सुनिश्चित करे की ऐप्लिकेशन,API के अनुरक्षित संस्करण और दिए गये मॉडल पर निर्भर करता है।
10. सप्लायर की सुरक्षा और पहुंच की नियमित समीक्षा करें और उनका ऑडिट कर यह निश्चित करें कि उनकी सुरक्षा स्थिति या नियम और शर्तों में कोई बदलाव न हो।

हमले के परिदृश्य में उदाहरण

1. एक हमलावर कमज़ोर पायथन लाइब्रेरी का इस्तेमाल कर सिस्टम को जोखिम में डाल सकता है। यह पहली बार Open AI डेटा चोरी (data breach) में हुआ था।
2. एक हमलावर फ़्लाइट खोजने के लिए एक LLM plugin प्रदान करता है, जो नकली लिंक बनाता है, जिससे plugin के यूज़र को धोके का सामना करना पड़ता है।
3. एक असल हमले में, हमलावर PyPI पैकेज सूचि का इस्तेमाल कर मॉडल के डेवलपर्स को धोखा देता है,जिसमे वह खराब पैकेज को डाउनलोड करा कर, डेटा निकाल सकें या मॉडल के विकास माहौल में विशेषाधिकार बढ़ा सकें।
4. एक हमलावर आर्थिक विश्लेषण और सामाजिक शोध में विशेषज्ञता वाले सार्वजनिक रूप से उपलब्ध, पूर्व-प्रशिक्षित मॉडल को विषाक्त बना देता है। इससे एक बैकडोर बनाता है, जिससे ग़लत सूचनाएं और फ़र्ज़ी ख़बरें बनती हैं। यह टारगेट को लक्षित करने के लिये इसे किसी मॉडल मार्केटप्लेस (जैसे HuggingFace) में स्तापित कर देते है।
5. एक हमलावर सार्वजनिक रूप से उपलब्ध डेटा को विषाक्त 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

विवरण

एक LLM ऐप्लिकेशन अपने आउटपुट के ज़रिए संवेदनशील जानकारी, उसकी एल्गोरिदम तथा दूसरी गोपनीय जानकारिया प्रकट कर सकती है। इसके परिणामस्वरूप 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 ज़रूरी 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 यूज़र डेटा को प्रशिक्षण मॉडल डेटा में प्रवेश करने से रोकने के लिए पर्याप्त डेटा सैनिटाइजेशन करना चाहिए।यूजर को उनके डेटा को प्रोसेस करने के तरीके और प्रशिक्षण मॉडल में अपने डेटा को शामिल करने से बहार निकलने 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 चाहिए।

उपभोक्ता-LLM ऐप्लिकेशन इंटरैक्शन विश्वास 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 ज़रूरी है कि इस ख़तरे की वजह से कुछ ज़रूरी शर्तें दायरे से बाहर हैं, जैसे थ्रेट मॉडलिंग अभ्यास, इंफ्रास्ट्रक्चर को सुरक्षित करना और पर्याप्त सैंडबॉक्सिंग। LLM को किस प्रकार का डेटा वापस करना चाहिए, इसके बारे में सिस्टम प्रॉम्प्ट में प्रतिबंध जोड़ने से संवेदनशील जानकारी के प्रकटीकरण के खिलाफ कुछ कमी मिल सकती है, लेकिन LLM की अप्रत्याशित प्रकृति का 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 फ़िल्टर करना।
2. LLM प्रशिक्षण प्रक्रिया में संवेदनशील डेटा को ओवेरफ़िट या याद रखना।
3. LLM की ग़लतफ़हमी, डेटा की सफाई में कमी या त्रुटियों के कारण गोपनीय जानकारी का अनायास ही खुलासा हो जाना

बचाव कैसे करें

1. यूज़र डेटा को ट्रेनिंग मॉडल डेटा में जाने से रोकने के लिए पर्याप्त डेटा सैनिटाइज़ेशन और स्क्रबिंग तकनीकों का उपयोग करे ।
2. मॉडल को विषाक्ता से बचाने के लिए, इनपुट सत्यापन और सैनिटाइज़ेशन के मज़बूत तरीके लागू करें, जिससे की दुर्भावनापूर्ण इनपुट पहचान कर फ़िल्टर किये जा सके ।
3. मॉडल को डेटा से समृद्ध करते समय या निम्नलिखित लिंक में दिये गयो को फ़ाइन-ट्यूनि करते समय (यानी, डिप्लॉयमेंट से पहले या उसके दौरान मॉडल में फीड किया गया डेटा)
    - फ़ाइन-ट्यूनिंग के समय संवेदनशील डेटा, यूज़र के समक्ष प्रकट हो सकता है। इसलिए, न्यूनतम पहुंच का नियम लागु करे और मॉडल को विशेषाधिकार वाली जानकारी से प्रशिक्षित न करें।
    - बाहरी डेटा स्रोतों तक पहुंच (रनटाइम के समय डेटा का ऑर्केस्ट्रेशन) सीमित होनी चाहिए।
    - बाहरी डेटा स्रोतों पर ऐक्सेस नियंत्रण और सुरक्षित सप्लाई चेन बनाए रखने के लिए एक कठोर तरीका अपनाएं।

हमले के परिदृश्य में उदाहरण

1. एक यूजर गैर-दुर्भावनापूर्ण तरीके से, LLM एप्लीकेशन के ज़रिये कुछ अन्य यूज़र डेटा के संपर्क में आ जाता है।
2. एक यूज़र LLM के इनपुट फ़िल्टर और सैनिटाइज़ेशन को बायपास करने के लिए प्रॉम्प्टस की एक श्रंखला को लक्षित करता है, इससे ऐप्लिकेशन के अन्य यूज़र के बारे में संवेदनशील जानकारी (PII) प्राप्त कर सके।
3. यूज़र या LLM ऐप्लिकेशन की लापरवाही से प्रशिक्षण 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

विवरण

LLM plugin ऐसे एक्सटेंशन होते हैं, जिन्हें चालू करने पर, यूज़र इंटरैक्शन के दौरान मॉडल द्वारा अपने-आप बुला लिये जाते हैं। वे मॉडल द्वारा संचालित होते हैं, और इनके निष्पादन पर ऐप्लिकेशन का कोई नियंत्रण नहीं होता है। इसके अलावा, सन्दर्भ के आकार की सीमाओं से निपटने के लिए, plugin द्वारा मॉडल से फ़्री-टेक्स्ट इनपुट लागू किए जा सकते हैं, बिना किसी सत्यापन या टाइप चेकिंग के। इससे एक संभावित हमलावर plugin के लिए एक दुर्भावनापूर्ण प्रांप्ट बना सकता है, जिसके परिणामस्वरूप कई तरह के अवांछित व्यवहार हो सकते हैं, जिसमें रिमोट कोड चलाना भी शामिल है।

दुर्भावनापूर्ण इनपुट का नुकसान अक्सर अपर्याप्त ऐक्सेस नियंत्रणों और सभी plugin में अनुमति को ट्रैक न कर पाने पर निर्भर करता है। अपर्याप्त ऐक्सेस नियंत्रण की मदद से एक plugin दूसरे plugin पर आँख बंद करके भरोसा कर सकता है और यह मान लेता है कि अंतिम यूज़र ने इनपुट दिए हैं। इस तरह के अपर्याप्त ऐक्सेस नियंत्रण की वजह से दुर्भावनापूर्ण इनपुट के हानिकारक परिणाम हो सकते हैं, जैसे कि डेटा एक्सफ़िल्ट्रेशन, रिमोट कोड चलाना और विशेषाधिकार बढ़ाना शामिल हैं।

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

कमज़ोरी के सामान्य उदाहरण

1. एक plugin अलग-अलग इनपुट मापदंडों के बजाय एक ही टेक्स्ट फ़ील्ड में सभी मापदंडों को स्वीकार करता है।
2. एक plugin मापदंडों (parameters) के बजाय कॉन्फ़िगरेशन स्ट्रिंग (configuration strings) को स्वीकार करता है, जो पूरी कॉन्फ़िगरेशन सेटिंग (configuration settings) को बदल सकता है।
3. एक plugin मापदंडों के बजाय Raw SQL या प्रोग्रामिंग स्टेटमेंट (programming statements) को स्वीकार करता है।
4. किसी plugin को स्पष्ट अनुमति दिए (authorization) बिना ही प्रमाणीकरण (authentication) किया जाता है।
5. एक plugin सारी LLM सामग्री को पूरी तरह से यूज़र द्वारा बनाई गई मानता है और बिना किसी 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 रेंज की जाँच शामिल करनी चाहिए। इनके आभाव में दूसरी लेयर शुरू कर अनुरोधों को पार्स, सैनिटाइज़ एवं उनकी पुष्टि करनी चाहिए। जब ऐप्लिकेशन सिमेंटिक्स की वजह से फ़्रीफ़ॉर्म इनपुट स्वीकार किया जाता है, तो यह सुनिश्चित करे कि किसी भी संभावित हानिकारक तरीके का इस्तेमाल नहीं किया जा रहा है।
2. Plugins डेवलपर्स को इनपुट की प्पुष्टि और सैनिटाइज़ेशन सुनिश्चित करने के लिए, ASVS (ऐप्लिकेशन सुरक्षा सत्यापन मानक) में दिये गये OWASP के सुझावों को लागू करना चाहिए।
3. पर्याप्त पुष्टि सुनिश्चित करने के लिए plugin की पूरी जाँच और परीक्षण किया जाना चाहिए। डेवलपमेंट पाइपलाइन (development pipelines) में स्टैटिक ऐप्लिकेशन सुरक्षा परीक्षण (SAST) स्कैन के साथ-साथ डायनामिक और इंटरैक्टिव एप्लिकेशन परीक्षण (DAST, IAST) का इस्तेमाल करें।
4. OWASP ASVS ऐक्सेस कंट्रोल दिशानिर्देशों का पालन करते हुए किसी भी असुरक्षित इनपुट पैरामीटर के इस्तेमाल के प्रभाव को कम करने के लिए plugins को डिज़ाइन किया जाना चाहिए। इसमें कम से कम विशेषाधिकार प्राप्त ऐक्सेस नियंत्रण शामिल है, जो अपना इच्छित कार्य करते समय जितना संभव हो उतनी कम कार्यक्षमता को उजागर करता है।

5. प्रभावी प्राधिकरण (effective authorization) और ऐक्सेस नियंत्रण (access control) लागू करने के लिए plugin को उपयुक्त प्रमाणीकरण पहचान (authentication identities) का इस्तेमाल करना चाहिए, जैसे कि OAuth2। इसके अलावा, एपीआई कीज़ (API Keys) का इस्तेमाल कस्टम प्राधिकरण (authorization) निर्णयों के लिए संदर्भ देने के लिए किया जाना चाहिए, जो डिफ़ॉल्ट इंटरैक्टिव यूज़र के बजाय plugin रूट को दर्शाता है।
6. संवेदनशील plugins द्वारा की गई किसी भी कार्रवाई के लिये यूज़र की अनुमति और पुष्टि की 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 चाहिए।

उदाहरण हमले के परिदृश्य

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 यूज़र के अनुरोध पर मौसम का पूर्वानुमान प्राप्त किए जा सकें। एक दुर्भावनापूर्ण यूज़र अनुरोध पर अपने डोमेन के ज़रिये LLM सिस्टम में अपनी सामग्री इंजेक्ट कर सकते हैं।
2. एक plugin फ़ील्ड में फ़्री फ़ॉर्म 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 निष्पादित करने और डेटा निकालने या विशेषाधिकार बढ़ाने के लिए तृतिय -पक्ष की कमजोरियों का फायदा उठाता है।
3. किसी वेक्टर स्टोर (vector store) से एम्बेडिंग प्राप्त करने के लिए इस्तेमाल किया जाने वाला एक plugin , कॉन्फ़िगरेशन पैरामीटर (configuration parameters) को कनेक्शन स्ट्रिंग (connection string) के तौर पर बिना किसी 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” क्लॉज़ 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 रिपॉजिटरी का स्वामित्व ट्रांसफ़र और यूज़र को उनकी रिपॉजिटरी 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

विवरण

LLM-आधारित सिस्टम को अक्सर उसके डेवलपर द्वारा दूसरे प्रणाली सिस्टम के साथ इंटरफेस करने और किसी प्रोम्प्ट के जवाब में कार्रवाई करने की क्षमता प्रदान की जाती है।इनपुट प्रॉम्प्ट या LLM आउटपुट के आधार पर डायनामिक रूप से निर्धारित करने के लिए किस फ़ंक्शन को लागू करना है इसका निर्णय LLM 'एजेंट' को भी सौंपा जा सकता है।

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

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

कमज़ोरी के सामान्य उदाहरण

1. अत्यधिक कार्यक्षमता: 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, किसी डेवलपर को किसी LLM एजेंट को रिपॉज़िटरी से दस्तावेज़ पढ़ने की सुविधा 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।
2. वैकल्पिक रूप से, हो सकता है कि किसी plugin को डेवलपमेंट के किसी चरण के दौरान ट्रायल किया गया हो और उसे किसी बेहतर विकल्प के पक्ष में छोड़ दिया गया हो, लेकिन मूल plugin LLM एजेंट के लिए उपलब्ध रहेगा।
3. अत्यधिक कार्यक्षमता: ओपन-एंडेड फ़ंक्शनैलिटी वाला LLM plugin , एप्लिकेशन के संचालन के लिए आवश्यक चीज़ों में कमांड के इनपुट निर्देशों को ठीक से फ़िल्टर करने में विफल रहता है। उदाहरण के लिए, एक विशिष्ट शेल कमांड चलाने वाला plugin दूसरे शेल कमांड को निष्पादित होने से रोकने में विफल रहता है।
4. अत्यधिक अनुमतियां: LLM plugin के पास दूसरे सिस्टम पर अनुमतियां होती हैं जिनकी ऐप्लिकेशन संचालित करने के लिए ज़रूरत नहीं होती है। उदाहरण के लिए, डेटा पढ़ने के लिए बनाया गया plugin किसी 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।
5. अत्यधिक अनुमतियां: एक LLM plugin जिसे यूज़र की ओर से ऑपरेशन करने के लिए बनाया गया है,वह विशेषाधिकार प्राप्त कर डाउनस्ट्रीम सिस्टम को ऐक्सेस करता है। उदाहरण के लिए, मौजूदा यूज़र के दस्तावेज़ों को पढ़ने के लिए एक plugin है, जो एक विशेषाधिकार प्राप्त खाते के साथ दस्तावेज़ रिपॉजिटरी से कनेक्ट होता है, जिसके पास सभी यूज़र की फ़ाइलों तक पहुंच होती है ।
6. अत्यधिक स्वायत्तता: कोई LLM-आधारित एप्लिकेशन या plugin हाई-इम्पैक्ट कार्रवाइयों को स्वतंत्र रूप से सत्यापित करने और उन्हें मंज़ूरी देने में विफल रहता है। उदाहरण के लिए, एक plugin जो बिना किसी पुष्टि के यूज़र के दस्तावेज़ों को हटाने की अनुमति देता है।

बचाव कैसे करें

निम्नलिखित कार्रवाइयों से अत्यधिक एजेंसी को रोका जा सकता है:

1. उन plugin/टूल को ज़रूरी फ़ंक्शन तक सीमित करें जिन्हें LLM एजेंट कॉल करने की अनुमति देते हैं। उदाहरण के लिए, अगर किसी LLM-आधारित सिस्टम के लिए किसी 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 सीमित करें। उदाहरण के लिए, एक plugin जो ईमेल को सारांशितकरता है ,उसके लिए केवल ईमेल पढ़ने की क्षमता होनी चाहिये,अन्य कार्यक्षमताएँ जैसे कि संदेश हटाने या भेजना की नहीं ।
3. जहाँ संभव हो, ओपन-एंडेड फ़ंक्शन से बचें (उदाहरण के लिए, शेल कमांड चलने, URL प्राप्त करने वाले आदि) और ज़्यादा लक्षित कार्यक्षमता वाले plugin/टूल का इस्तेमाल करें। उदाहरण के लिए, किसी LLM- आधारित ऐप के लिए किसी फ़ाइल में कुछ आउटपुट लिखने 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 बड़ जाता (कोई भी अन्य शेल कमांड निष्पादित किया जा सकता है)। एक ज़्यादा सुरक्षित विकल्प यह होगा कि एक ऐसा फ़ाइल-राइटिंग plugin बनाया जाए, जो सिर्फ़ उस खास सुविधा के साथ ही काम कर सके।
4. उन अनुमतियों को सीमित करें जो LLM plugins/टूल दूसरे सिस्टम को दी जाती हैं, ताकि अवांछनीय कार्रवाइयों का दायरा सीमित किया जा सके। उदाहरण के लिए, एक LLM एजेंट जो किसी ग्राहक को खरीदारी के सुझाव देने के लिए प्रॉडक्ट डेटाबेस का इस्तेमाल करता है, उसे सिर्फ़ 'प्रॉडक्ट' टेबल पढ़ने की ज़रूरत होगी; उसके पास दूसरी टेबल तक ऐक्सेस नहीं होनी चाहिए, न ही रिकॉर्ड 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 कि यूज़र की ओर से की गई कार्रवाइयां डाउनस्ट्रीम सिस्टम पर उस विशिष्ट यूज़र के संदर्भ में और न्यूनतम ज़रूरी विशेषाधिकारों के साथ निष्पादित हों, यूज़र की अनुमति और सुरक्षा क्षेत्र को ट्रैक करें। उदाहरण के लिए, एक LLM plugin जो यूज़र के कोड रेपो को पढ़ता है,तो उसे OAuth के ज़रिए और न्यूनतम स्कोप के साथ प्रमाणीकरण करना होगा।
6. सभी कार्रवाइयों को करने से पहले मानव द्वारा मंज़ूरी ले। इसे डाउनस्ट्रीम सिस्टम (LLM ऐप्लिकेशन के दायरे से बाहर) या LLM plugin /टूल में ही लागू किया जा सकता है। उदाहरण के लिए, किसी 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. किसी कार्रवाई की अनुमति है या नहीं, यह तय करने के लिए LLM पर निर्भर रहने के बजाय डाउनस्ट्रीम सिस्टम में प्राधिकरण (authorization) लागू करें। टूल/plugin लागू करते समय, मीडिएशन का पूरा सिद्धांत लागू करें, ताकि plugin/टूल के ज़रिये डाउनस्ट्रीम सिस्टम से किए गए सभी अनुरोधों की सुरक्षा नीतियों के तहत पुष्टि हो सके।

निम्नलिखित विकल्प 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. किसी निश्चित 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 ज़रिए अवांछनीय कार्रवाइयों का पता लगाने के अवसर बढ़ाएँ।

हमले के परिदृश्य मे उदाहरण

LLM-आधारित व्यक्तिगत सहायक ऐप (personal assistant app) को plugin द्वारा किसी व्यक्ति के मेलबॉक्स से जोड़ा गया, जिससे की वह आने वाले emails को सारांशित करे । इसके लिये plugin को संदेशों को पढ़ने की क्षमता की आवश्यकता है, हालांकि सिस्टम डेवलपर द्वारा चुना plugin के पास संदेश भेजने की क्षमता भी हैं। अगर प्रयोग किया गया LLM अप्रत्यक्ष prompt इंजेक्शन हमले के प्रति कमजोर है, तब कोई दुर्भावनापूर्ण-ईमेल LLM को plugin द्वारा, 'send message' फनशन के प्रयोग से उपयोगकर्ता के मेलबॉक्स से स्पैम भेजने को कहता है। इससे बचने के लिये:
(क) अत्यधिक कार्यक्षमता को घटाने के लिये, केवल पढ़ने की क्षमता वाले plugin चुने,
(ख) अत्यधिक अनुमतियों को घटाने के लिये उपयोगकर्ता की ईमेल को केवल-पढ़ने के दायरे वाली OAuth सत्र से जोड़े,
(ग) अत्यधिक स्वायत्तता को घटाने के लिये उपयोगकर्ता, एलएलएम प्लगइन द्वारा बने प्रत्येक मेल की समीक्षा करके ही भेजे। मेल भेजने के इंटरफ़ेस की दर सीमित करने से भी क्षति को कम किया जा सकता 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
4. **Simon Willison: Dual LLM Pattern**: Simon Willison

LLM09: ओवररिलायंस

विवरण

ज़्यादा निर्भरता तब होती है जब सिस्टम या लोग बिना पर्याप्त निरीक्षण के निर्णय लेने या कॉन्टेंट तैयार करने के लिए LLM पर निर्भर 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 ग़लत सूचना, ग़लतफ़हमी, कानूनी समस्याएं और प्रतिष्ठा को नुकसान हो सकता है।

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

- ओवरसाइट (Oversight)
- सत्यापन की निरंतर व्यवस्था (Continuous validation mechanism)
- अस्वीकार्य जो जोखिम में हैं (Disclaimers on risk)

कमज़ोरी के सामान्य उदाहरण

1. LLM प्रतिक्रिया के तौर पर गलत जानकारी देता है, एवं वह इन जानकारियों को अत्यधिक आधिकारिक रूप मे प्रदर्शित करता है। प्रणाली को उचित जांच और संतुलन के बिना डिज़ाइन किया गया है, जिससे यह दिक्कत संभाल नहीं पाती और परिणामस्वरूप उपयोगकर्ता को भ्रामक जानकारी मिलती है ।
2. एलएलएम असुरक्षित या दोषपूर्ण कोड सुझाता है, जो उचित निरीक्षण या सत्यापन के बिना सॉफ़्टवेयर सिस्टम में शामिल होने पर कमजोरियों को जन्म देता है।

बचाव कैसे करें

1. LLM आउटपुट की नियमित निगरानी और उनकी समीक्षा करें। इन्कन्सीस्टेन्ट टेक्स्ट (inconsistent text) को फ़िल्टर करने के लिए आत्म स्थिरता (self-consistency) या वोटिंग तकनीकों का इस्तेमाल करें। एक ही प्रॉम्प्ट के लिए कई मॉडल प्रतिक्रियाओं की तुलना करने से आउटपुट की गुणवत्ता और निरंतरता का बेहतर आकलन किया जा सकता है।
2. विश्वसनीय बाहरी स्रोतों से LLM आउटपुट को क्रॉस-चेक करें।अतिरिक्त पुष्टि से यह पक्का करने में मदद मिल सकती है कि मॉडल के ज़रिये दी गई जानकारी सटीक एवं भरोसेमंद है।
3. फ़ाइन-ट्यूनिंग या एम्बेडिंग की मदद से आउटपुट को बेहतर बनाएं। किसी खास क्षेत्र के किए बनाये गए मॉडल की तुलना में जेनेरिक पहले से प्रशिक्षित मॉडल से गलत जानकारी मिलने की संभावना ज़्यादा होती है। इसके लिए प्रॉम्प्ट इंजीनियरिंग (prompt engineering), पैरामीटर एफ़िशिएंट ट्यूनिंग (parameter efficient tuning), फुल मॉडल ट्यूनिंग (full model tuning) और चेन ऑफ़ थॉट (chain of thought) प्रॉम्प्टिंग जैसी तकनीकों का इस्तेमाल किया जा सकता है।
4. ऑटोमैटिक सत्यापन तंत्र लागू करें, जो ज्ञात तथ्यों या डेटा के विरुद्ध जनरेट किए गए आउटपुट की क्रॉस-पुष्टि कर सके। यह सुरक्षा की एक अतिरिक्त परत प्रदान कर, मतिभ्रम 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 मतिभ्रम की संभावना को भी कम करता है क्योंकि हर एजेंट को एक छोटे से काम के लिए जिम्मेदार ठहराया जा सकता है।
6. LLM के इस्तेमाल से जुड़े जोखिमों और सीमाओं के बारे में जानकारी दे, जो प्रभावी जोखिम संचार यूज़र को संभावित समस्याओं के लिए तैयार कर सकता कर सही निर्णय लेने में मदद कर सकता है।

7. ऐसे API और यूज़र इंटरफेस बनाएं, जो LLM के ज़िम्मेदार और सुरक्षित इस्तेमाल को प्रोत्साहित करें। इसमें कॉन्टेंट फ़िल्टर, संभावित अशुद्धियों के बारे में यूज़र की चेतावनियां और AI द्वारा जेनरेट की गई सामग्री की क्लियर लेबलिंग जैसे उपाय शामिल हैं।
8. विकास के दौरान ही LLM को संभावित कमजोरियों से बचाने के लिए सुरक्षित कोडिंग तकनीके एवं दिशानिर्देशों को स्थापित करें।

उदाहरण हमले का परिदृश्य

1. समाचार संगठन समाचार बनाने के लिए AI मॉडल का ज़्यादा इस्तेमाल करता है। एक दुर्भावनापूर्ण व्यक्ति इस अति-निर्भरता का फायदा उठाता है, AI को गुमराह करने वाली जानकारी देता है, जिससे गलत सूचनाएं फैलती हैं। AI ने अनजाने में ही सामग्री को प्लगिराइज़्ड (plagiarized) कर दिया, जिससे कॉपीराइट समस्याएँ पैदा हो गईं और संगठन में विश्वास कम हो गया।
2. सॉफ़्टवेयर डेवलपमेंट टीम कोडिंग प्रक्रिया में तेज़ी लाने के लिए कोडेक्स जैसे AI सिस्टम का इस्तेमाल करती है। AI के सुझावों पर ज़्यादा भरोसा करने से सुरक्षा डिफ़ॉल्ट सेटिंग या सुरक्षित कोडिंग पद्धतियों के साथ असंगत सुझावों के कारण ऐप्लिकेशन में सुरक्षा संबंधी कमजोरियाँ आ जाती हैं।
3. एक सॉफ़्टवेयर डेवलपमेंट फर्म डेवलपर्स की सहायता के लिए 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 सॉफ़्टवेयर 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

विवरण

यह दुर्भावनापूर्ण व्यक्तियों या APTs द्वारा LLM मॉडल में अनधिकृत पहुंच और घुसपैठ को संदर्भित करती है। यह तब होता है जब LLM मॉडल (मूल्यवान बौद्धिक संपदा होने के नाते),के साथ छेड़छाड़ की जाती है, भौतिक रूप से चोरी हो जाते हैं, कॉपी किए जाते हैं या एक कार्यात्मक समकक्ष बनाने के लिए वज़न और पैरामीटर निकाले जाते हैं। LLM मॉडल की चोरी के प्रभाव में आर्थिक और ब्रांड प्रतिष्ठा खोना, प्रतिस्पर्धात्मक 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 चिंता 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 बौद्धिक 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 हितों की सुरक्षा करने के लिए एक व्यापक सुरक्षा ढांचे का इस्तेमाल करना, जिसमें ऐक्सेस नियंत्रण, एन्क्रिप्शन और निरंतर निगरानी शामिल है तथा महत्वपूर्ण है।

कमज़ोरी के सामान्य उदाहरण

1. एक हमलावर कंपनी के कमजोर इंफ्रास्ट्रक्चर का फायदा उठा, कंपनी के नेटवर्क या ऐप्लिकेशन सुरक्षा सेटिंग में ग़लतफ़हमी के ज़रिए LLM मॉडल रिपॉजिटरी तक अनधिकृत पहुंच बनता है।
2. अंदरूनी खतरे का परिदृश्य जहां एक असंतुष्ट कर्मचारी किसी मॉडल या उससे जुड़ी सामग्री को लीक कर देता है।
3. एक हमलावर API से सावधानी से तैयार किए गए इनपुट और प्रॉम्प्ट इंजेक्शन तकनीकों का इस्तेमाल करके पूछताछ करता है, जिससे की शैडो मॉडल बनाने के लिए पर्याप्त संख्या में आउटपुट प्राप्त होते है।
4. एक दुर्भावनापूर्ण हमलावर एक साइड-चैनल हमला करने के लिए LLM की इनपुट फ़िल्टरिंग तकनीकों को दरकिनार कर सकता है और अंतत: रिमोट नियंत्रित संसाधन का उपयोग करके मॉडल के आर्किटेक्चर और उससे जुडी जानकारियां हासिल कर सकता है।
5. मॉडल एक्सट्रैक्शन के अटैक वेक्टर में किसी खास विषय पर बड़ी संख्या में प्रॉम्प्ट्स के साथ LLM से पूछताछ करना शामिल है। इसके बाद LLM के आउटपुट का इस्तेमाल किसी 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 पर्याप्त नहीं हैं, तो LLM से मिलने वाले आउटपुट बेकार होंगे।
    - LLM के आउटपुट में कभी-कभी बेहूदा जवाब हो सकते हैं, मतलब हमलावर पूरे मॉडल को निकालने में सक्षम नहीं हो सकता क्योंकि कुछ आउटपुट बेतुके हो सकते हैं।
        - मॉडल एक्सट्रैक्शन के ज़रिये किसी LLM को 100% बनाना संभव नहीं है। हालांकि, हमलावर एक अपूर्ण (partial) मॉडल बना सकता है।

6. फ़ंक्शनल मॉडल रेप्लिकेशन के अटैक वेक्टर में सिंथेटिक प्रशिक्षण डेटा ("सेल्फ़-इंस्ट्रक्ट" नामक दृष्टिकोण) जेनरेट करने के लिए प्रॉम्प्ट्स को लक्षित मॉडल पर उपयोग करना, फिर उसका इस्तेमाल कर किसी अन्य मूलभूत मॉडल को फ़ाइन-ट्यून करना शामिल है। यह उदाहरण 5 में इस्तेमाल किए गए पारंपरिक क्वेरी-आधारित (query-based) एक्सट्रैक्शन की सीमाओं को दरकिनार कर देता है और शोध कार्य के लिये किसी अन्य LLM को प्रशिक्षित 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 किसी मालिकाना मॉडल को सार्वजनिक API की मदद से बनाने के लिए कर सकता है।

किसी चोरी हुए मॉडल का इस्तेमाल, शैडो मॉडल के तौर पर, प्रतिकूल हमलों को स्टेज करने के लिए किया जा सकता है, जिसमें मॉडल में मौजूद संवेदनशील जानकारी तक अनाधिकृत पहुंच एवं एडवांस प्रॉम्प्ट इंजेक्शन को आगे बढ़ाने के लिए प्रतिकूल इनपुट के साथ प्रयोग करना शामिल है, जिसका पता नहीं चलता है।

बचाव कैसे करें

1. LLM मॉडल रिपॉजिटरी और प्रशिक्षण वातावरण तक अनधिकृत पहुंच को सीमित करने के लिए मज़बूत ऐक्सेस नियंत्रण (जैसे, RBAC और कम से कम विशेषाधिकार का नियम) और मज़बूत प्रमाणीकरण (authentication) तंत्र लागू करें।
    - यह पहले तीन सामान्य उदाहरणों के लिए खास तौर पर सही है, जो अंदरूनी खतरों, गलत कॉन्फ़िगरेशन और/या इंफ्रास्ट्रक्चर के बारे में कमज़ोर सुरक्षा नियंत्रण के कारण इस जोखिम का कारण बन सकते हैं, जिसमें LLM मॉडल, वज़न और आर्किटेक्चर मौजूद हैं, जिसमें एक दुर्भावनापूर्ण व्यक्ति वातावरण के अंदर या बाहर से घुसपैठ कर सकता है।
    - सप्लाई-चेन के हमलों को रोकने के लिए आपूर्तिकर्ता प्रबंधन ट्रैकिंग, सत्यापन और निर्भरता की कमजोरियाँ महत्वपूर्ण विषय हैं।

2. नेटवर्क संसाधनों, आंतरिक सेवाओं और API तक LLM की पहुँच को प्रतिबंधित करें।
    - यह सभी सामान्य उदाहरणों के लिए खास तौर पर सही है क्योंकि यह अंदरूनी जोखिमों 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. उत्पादन वाले एमएल मॉडल के लिए एक केंद्रीकृत एमएल मॉडल इन्वेंटरी या रजिस्ट्री का उपयोग करें। यह होने से एक्सेस नियंत्रण, प्रमाणीकरण और निगरानी/लॉगिंग के माध्यम से LLM मॉडल तक अनधिकृत पहुंच को रोका जा सकता है जो गवर्नन्स (governance) के लिए नींव हैं। यह अनुपालन, जोखिम मूल्यांकन और जोखिम शमन के लिए मॉडलों द्वारा प्रयोग की गई एल्गोरिदम के बारे में डेटा एकत्र करने के लिए भी फायदेमंद है।
4. किसी भी संदिग्ध या अनधिकृत व्यवहार का तुरंत पता लगाने और उसका जवाब देने के लिए, LLM मॉडल रिपॉजिटरी से संबंधित एक्सेस लॉग और गतिविधियों की नियमित रूप से निगरानी करें और उनका ऑडिट करें।
5. इंफ्रास्ट्रक्चर में ऐक्सेस और डिप्लॉयमेंट नियंत्रणों को बेहतर बनाने के लिए, गवर्नेंस, ट्रैकिंग और कार्यप्रवाह की मंज़ूरी की मदद से MLOP के डिप्लॉयमेंट को स्वचालित करें।
6. साइड-चैनल अटैक की वजह से प्रॉम्प्ट इंजेक्शन तकनीकों के जोखिम को कम करने के लिए नियंत्रण और शमन रणनीतियां लागू करें।
7. API कॉल फ़िल्टर की दर सीमित कर, LLM ऐप्लिकेशन से डेटा में घुसपैठ के जोखिम को कम किया जा सकता है, या अन्य निगरानी प्रणालियों से (जैसे, DLP) निकास की गतिविधि का पता लगाने के लिए तकनीकों को लागू किया जा सकता है।
8. निष्कर्ष संबंधी प्रश्नों का पता लगाने और भौतिक सुरक्षा उपायों को मजबूत करने में मदद करने के लिए प्रतिकूल एवं सुदृढ़ प्रशिक्षण लागू करें।
9. LLM के जीवनचक्र में एम्बेडिंग और खोजने के चरणों में वॉटरमार्किंग फ़्रेमवर्क लागू करें।

उदाहरण हमले का परिदृश्य

1. एक हमलावर किसी 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 प्रतिस्पर्धी भाषा प्रोसेसिंग सेवा शुरू करने या संवेदनशील जानकारी निकालने 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 प्रतिकूल 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 पर्याप्त संख्या में आउटपुट इकट्ठा करता है।
4. एक सुरक्षा नियंत्रण विफलता सप्लाई चेन के भीतर मौजूद है जो मालिकाना मॉडल जानकारी के डेटा लीक की ओर ले जाती है।
5. एक दुर्भावना वाला हमलावर साइड-चैनल हमला करने और अपने नियंत्रण के तहत रिमोट नियंत्रित संसाधन पर मॉडल जानकारी पुनर्प्राप्त 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