資料4 - 1

AIセキュリティ分科会
サイボウズにおけるAIセキュリティ対策

2025.11.04

サイボウズ株式会社 PSIRT
小西 達也、湯浅 潤樹

サイボウズにおけるAI機能開発

kintoneを含む複数の製品でAI機能を提供中・提供予定[1]

[1] kintone、サイボウズ Office、GaroonでAI新機能を発表, https://topics.cybozu.co.jp/news/2025/10/27-19212.html

kintone AI機能の概要

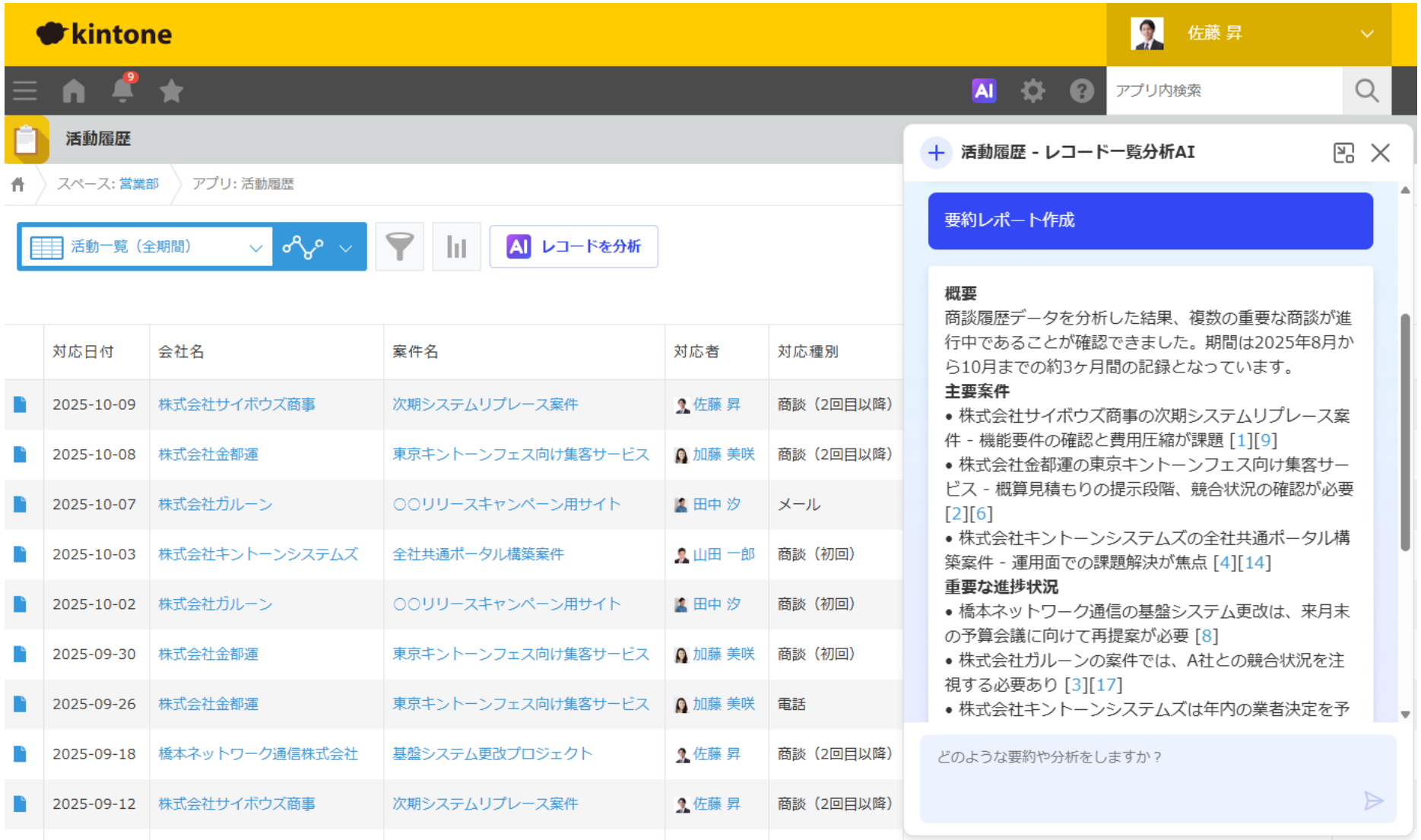
kintoneは業務アプリが作れる**ローコード・ノーコードツール**

kintoneでは複数のAI機能を提供中[2]

- **検索AI**：アプリのデータについての質問にAIが回答
- **アプリ作成AI**：アプリの作成についてAIが支援
- **プロセス管理設定AI**：プロセス管理設定をAIが提案・設定
- **スレッド要約AI**：スレッド内のコメント全体をAIが要約
- **レコード一覧分析AI**：レコード一覧の内容をAIが分析・要約

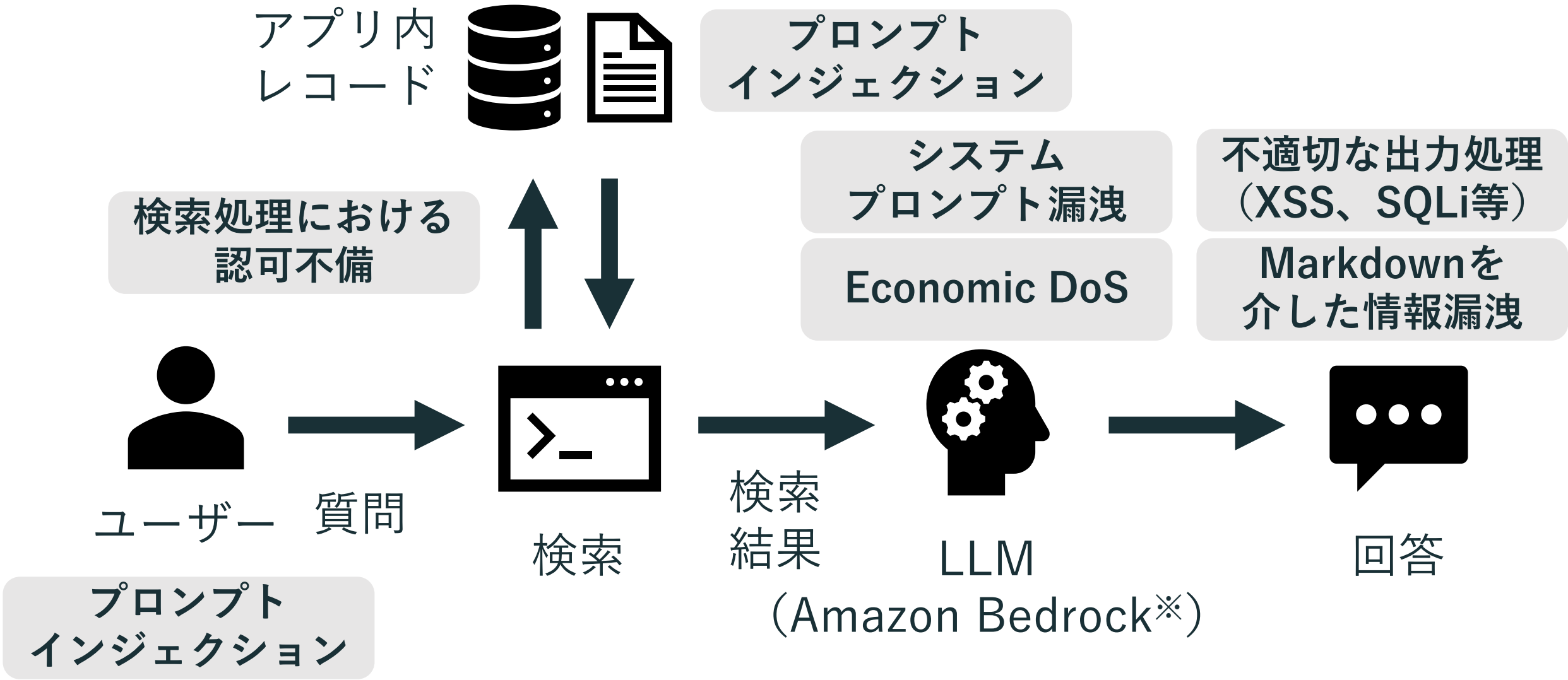
[2] kintone AIラボ, https://kintone.cybozu.co.jp/feature/kintone-ai-labo/

kintone レコード一覧分析AI



出典: kintone、サイボウズ Office、GaroonでAI新機能を発表, https://topics.cybozu.co.jp/news/2025/10/27-19212.html

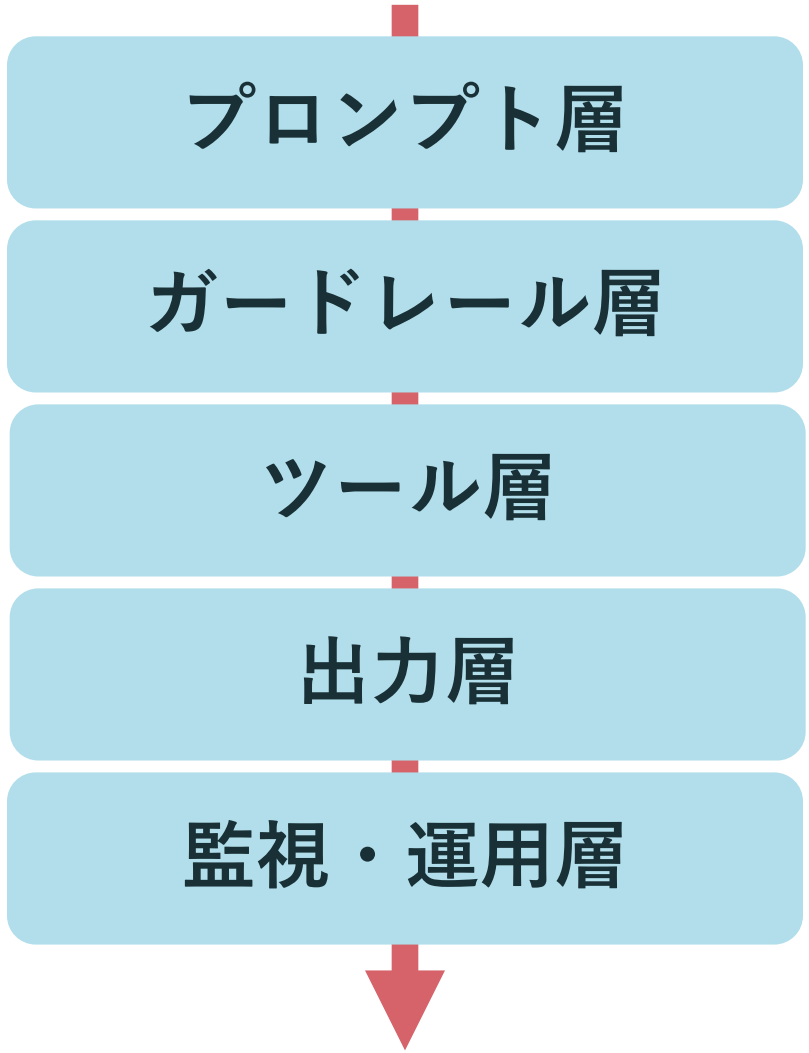
kintone AI機能において想定される脅威（例: レコード一覧検索AI）



※ 2025/11/04 現在利用中ですが、今後変更となる可能性があります。

kintone AI機能におけるセキュリティ対策

プロンプトインジェクションを起点とした攻撃に多層防御で対応



プロンプト層の対策

プロンプト
インジェクション

システム
プロンプト漏洩

プロンプトインジェクションの難易度向上・リスク低減

- **システム指示と外部データの分離**
  - roleを適切に使い分ける（ system, user, assistant ）
- **システム指示に機密情報を含めない**
  - システムプロンプト漏洩時のリスクを抑える
- **システムプロンプトの堅牢化**
  - システムプロンプトの記述方法を工夫して難易度を向上
  - 参考: AWSのブログ記事[3]、Prompt Hardener[4]

[3] Secure RAG applications using prompt engineering on Amazon Bedrock, https://aws.amazon.com/jp/blogs/machine-learning/secure-rag-applications-using-prompt-engineering-on-amazon-bedrock/
[4] cybozu/prompt-hardener, https://github.com/cybozu/prompt-hardener

ガードレール層の対策

プロンプト
インジェクション

入出力を監視・制御して誤動作や情報漏洩を防ぐ

- **クラウドサービス組み込みのガードレールを利用**
  - AWS Bedrock Guardrails[5] ※
  - 有害なコンテンツのフィルタリング、プロンプト攻撃からの防御
  - 個人を特定できる情報（PII）のマスキング
- **製品の利用用途上、柔軟に運用**
  - 地名などの住所情報がコンテキストに入る場合の誤検知
    → リスクの調査を実施後、一部のポリシーを外す
  - ユーザー入力の入らない機能にはガードレールを適用しない

※ 2025/11/04 現在利用中ですが、今後変更となる可能性があります。

[5] Amazon Bedrock のガードレール, https://aws.amazon.com/jp/bedrock/guardrails/

ツール層（Function Calling）の対策

検索処理における
認可不備

関数の実行やDB操作を担う部分なので適切に実装・検証

Function Calling[6]: 定義した関数の引数の値を推論して関数に渡す

- **引数のバリデーション**
  - 型チェック、値に不正な文字などが含まれていないか
- **不要な引数は推論させない**
  - APIスキーマは最小限に設計
- **ツールに与える権限の最小化、適切な認可制御**
  - LLMに与えるツール、ツールが参照・操作できる範囲は最小限に

[6] Function Calling, https://platform.openai.com/docs/guides/function-calling

出力層の対策

不適切な出力処理（XSS、SQLi等）

Markdownを介した情報漏洩

LLMの出力を適切に処理することで脆弱性や情報漏洩を防ぐ

- **不適切な出力処理を防ぐ**
  - XSS、SQLインジェクション対策のため適切にバリデーション
- **Markdownを介した意図しない情報漏洩を防ぐ**
  - LLMが出力したリンクを開く際には確認ダイアログを表示
  - Markdown内の画像・リンクでURLのホスト名を検証

↓確認ダイアログ

監視・運用層での横断的な対策

Economic DoS

コスト制御、インシデント対応などの観点で横断的に対策

- **Economic DoS対策**
  - 入出力トークン数の制限、レートリミットの適用
- **LLM呼び出し時のログ保存**
  - LLMのモデル、入力、出力を保存し、インシデント発生時に活用
- **LLM事業者からのBAN対策**
  - ユーザーが悪意ある指示を送信し続けることでアカウント停止の可能性
  - LLM提供事業者との継続的な連絡経路を確保、ガードレールの適用

弊社におけるセキュリティ品質担保の取り組み例

- **開発者向け**
  - **「AIサービスにおけるセキュア開発・設計ガイドライン」の作成**
  - **AIテストプラン表の作成**: 実施するテスト範囲の提示や開発チームとの認識合わせに用いる表
  - **チェックリストの作成**: 実装機能に対してテストの依頼が必要かを開発チーム側で判断するリスト
  - **AIセキュリティに関する社内勉強会の実施**: AIに関係する複数のテーマで実施

- **テスター向け(PSIRT)**
  - **検証方法一覧リスト**: AIのテスト手法やテスト観点について整理したリスト
  - **テストシナリオおよびゴールの設定**: 脅威モデリングによるリスクの洗い出しやテストゴールの設定
  - **テスト時のルールの設定**: AIの再現性の低さをカバーするためのテストルールの設定
    - 試行回数の基準の設定
    - 再現時の証跡取得ルール（画像・動画）の設定

「AIサービスにおけるセキュア開発・設計ガイドライン」について

- AI技術を利用した機能(AIサービス)を開発するに辺り、セキュアな実装のために必要なポイントや運用方法を示したガイドライン
- 組織内での共通のセキュリティ品質を担保するために作成し、組織全体に展開
- ガイドラインの項目
  - ガイドラインの目的とガイドラインにおけるAIサービスの定義
  - AIサービスにおける脅威とその被害
  - **AIサービスの開発・運用におけるセキュリティで気をつけるポイント**
  - 構成管理について
  - PSIRTへのAIに関するセキュリティテストの依頼
  - セキュリティインシデント発生時の対応
  - 付録：過去の勉強会

AIサービスの開発・運用におけるセキュリティで気をつけるポイントの項目

- AIサービスの開発に用いる技術部分に注目して、その部分で発生する攻撃や対策に関して表にして提示
- **表の項目**
  - 項目（技術の名称）
  - 説明（技術の説明）
  - 攻撃の観点
  - レベル感(リスクのレベル感)
  - 対策
  - OWASP Top10 for LLM Applications
  - インシデント事例や参考情報

項目	説明	攻撃の観点	レベル感	対策	OWASP Top10 for LLM Applications	インシデント事例
プロンプト	LLMに対して望む応答を引き出すために与えるシステム指示やユーザー指示、外部データなどの入力	• プロンプトインジェクション ◦ システム指示・ユーザ指示・外部データの指示境界の混同によって生じる。	高	• 指示と外部データの分離 • role:systemにユーザー入力を含めない • 機密情報を埋め込まない • プロンプトの堅牢化	• LLM01 • LLM02 • LLM05 • LLM07	初のゼロクリックAI脆弱性「Echo Leak」、Microsoftの「Copilot」の脆弱性で（修正済み）
ガードレール	LLMの不適切な動作や出力を防ぐために入出力を検証・制御する仕組み	• ガードレールの回避によるデータ窃取 ◦ 入力の検証不備 ◦ 出力の検証不備	中	• AWS BedrockやAzure AIのガードレールサービスを活用する • 誤検知や検知漏れの調整	• LLM02	Bypassing LLM Guardrails: An Empirical Analysis of Evasion Attacks...
Function Calling	LLMがツール（関数）を呼び出して、何らかの処理を行う仕組み	• ツールの過剰な権限を利用した不正操作 • ツールの引数の値を制御して危険なコールを実行 ◦ include_private: true など	高	• 最小権限のツール設計 • ツールでは不要な引数を推論させない • ツール側での入力値のバリデーション	• LLM01 • LLM02 • LLM06	Critical RCE Vulnerability in mcp-remote: CVE-2025-6514 Threatens LLM Clients

ガイドライン作成時に考慮したポイント

- 開発者に馴染み深い技術セットの観点から脅威を伝える内容にすることで実際の実装時に利用しやすくした。
- インシデント事例を示すことで実際の脅威に関するイメージを掴みやすくした。
- 構成管理やインシデント時の対応を併せて示すことで、組織全体でのセキュリティ品質を包括できる内容にした。