# AI搭載アプリを新たな脆弱性から保護

AI関連のリスクを特定し、SDLC全体にわたってアプリケーションを保護する包括的なセキュリティを確保




## メリット

- 従来のコードだけでなく、AI搭載アプリケーションのセキュリティを確保
- AIおよびLLM固有の脆弱性を検知
- 開発ライフサイクル全体にわたるAI対応テスト
- DevSecOpsのスピードでの実践的なリスク軽減

AIは新世代のインテリジェントアプリケーションにパワーを与えていますが、そのために、従来のAppSecツールでは検知不可能な新たな脆弱性がもたらされる結果にもなっています。企業がLLM、AIエージェント、モデル駆動型機能を顧客対応システムやミッションクリティカルなシステムに組み込むにつれ、プロンプトインターフェイスからAI出力処理まで、さまざまな攻撃対象領域が新たにさらされています。

OpenText™アプリケーションセキュリティは、セキュリティリーダーがSDLC全体にわたってAI搭載アプリケーションを保護するうえで役立ちます。OpenTextは、AI固有の検知、自動トリアージ、インテリジェントなリスク優先順位付けなどを含む、実証済みのSAST、SCA、DAST機能を拡張することで、チームが新たな脅威を低減し、AppSecの効率を向上させ、エンタープライズ規模でイノベーションを実現できるように支援します。

### 従来のコードだけでなく、AI搭載アプリケーションのセキュリティを確保

OpenText™アプリケーションセキュリティを利用すると、組織はAIを活用するアプリケーションスタック全体にわたってセキュリティを確保できるようになります。これには、インテリジェントな動作を促進するコード、API、データフロー、AIコンポーネントが含まれます。大規模言語モデル、AIエージェント、生成AIサービスが顧客対応システムやミッションクリティカルなシステムに統合されるのに合わせて、OpenTextは、こうした最新のアーキテクチャをサポートするために、実証済みのAppSec手法を拡張しています。その結果として、イノベーションのスピードを低下させない、AI機能の強力な保護が誕生しました。

### AIおよびLLM固有の脆弱性を検知

AIを活用する機能によってもたらされる固有のセキュリティリスクは、従来のテスト手法では見落とされがちです。OpenTextのAppSecポートフォリオは、こうしたリスクに対処します。これらの中には、プロンプトインジェクション、非セキュアなモデルインタラクション、AI応答を通じたデータ漏えい、安全でない出力処理、エージェントオーケストレーションやツール呼び出しの脆弱性などがあります。OpenTextは、AI関連のリスクの露出や悪化を招く可能性がある従来のアプリケーションやAPIの欠陥も特定します。

### 開発ライフサイクル全体にわたるAI対応テスト

OpenTextは、初期開発からCI/CD、本番環境までをカバーする継続的なAI対応セキュリティテストをサポートしています。SAST、DAST、ソフトウェアサプライチェーン分析を、AIがアシストする監査および優先順位付け機能と組み合わせることで、AI機能がアプリケーションロジック、機密データ、外部サービスとどのようにやり取りをしているかを、チームが明確に可視化できるようにしています。これにより、影響がユーザーに及ぶ前にリスクを検知し、対処することができます。

### DevSecOpsのスピードでの実践的なリスク軽減

AIを活用したトリアージと修正により、チームは最も重大な脆弱性に集中できるようになるため、ワークフローが効率化されます。OpenText AppSecは、誤検知を減らし、悪用される可能性のある現実のリスク(AIに固有のものを含む)に焦点を当てることで、意思決定の迅速化、セキュリティチームと開発チーム間のコラボレーションの強化、AI駆動型ソフトウェアの安全な大規模デリバリーを可能にします。

### AIがもたらす実際の脆弱性の例

AI機能は、従来のAppSecでは見落とされがちな新たなリスクをもたらします。たとえば、プロンプトインジェクション、安全でない出力、ベクトルベースのデータ漏えいなどです。この表では、LLMを活用するアプリケーションに含まれる実際の脆弱性に焦点を当て、「OWASP Top 10 for LLM Apps」の対応する項目を示し、AIにより攻撃対象領域が拡大されている状況と、SDLC全体にわたってAI対応の保護が重要である理由を説明しています。

| カテゴリ | 説明 | OWASP Top 10 for LLM Apps |
|---|---|---|
| **Anthropicの誤設定：未指定のトークン制限** | Anthropic APIのトークンまたはレート制限を省略すると、ユーザーは長い要求や無制限の要求を発行できるようになり、リソース枯渇やコスト増大につながります。 | LLM05:2025 不適切な出力処理/ LLM10:2025 無制限の消費 |
| **資格情報管理：ハードコードされたAPI資格情報** | APIキーやパスワードをコードに直接埋め込むと、ソースを表示できるすべてのユーザーにさらされるため、不正アクセスが可能になります。 | - |
| **クロスサイトスクリプティング(XSS)：AI** | AIで生成されたHTMLまたはスクリプト出力を適切にサニタイズせずに反映すると、攻撃者はユーザーのブラウザーでJavaScriptを実行できるようになります。 | LLM03:2025 サプライチェーン/ LLM05:2025 不適切な出力処理 |
| **クロスサイトスクリプティング(XSS)：DOM AI** | 信頼できないAI出力をクライアント側のDocument Object Model (DOM)に挿入すると、DOMの改ざんにより、悪意あるペイロードを実行できるようになります。 | LLM03:2025 サプライチェーン/ LLM05:2025 不適切な出力処理 |
| **データ汚染：AIの埋め込み** | 悪意あるデータや汚染されたデータがベクトルストアに挿入されると、アクセス制御が弱い場合に、セマンティック検索が歪められたり、機密情報が漏えいしたりする可能性があります。 | LLM04:2025 データとモデルの汚染/ LLM08:2025 ベクトルと埋め込みの弱点 |
| **データ汚染：AIトレーニング** | 攻撃者はバイアスやバックドアの例を使用してトレーニング用や微調整用のデータセットを破壊できるため、そのモデルによって有害な出力が生成されます。 | LLM04:2025 データとモデルの汚染 |
| **動的なコード評価：TensorFlowの安全でないデシリアライゼーション** | 安全でないYAMLロード(yaml.unsafe_load)を使用してKerasモデルをデシリアライズすることで、任意のコード実行が可能になっていたため、リスク軽減のために削除されました。 | LLM03:2025 サプライチェーン |
| **埋め込みとデータ露出** | 不適切にパーティション分割された埋め込みにより、ベクトルデータベースに保存されている機密データを他のクエリやテナントが取得または推測できるようになります。 | LLM08:2025 ベクトルと埋め込みの弱点 |
| **エンコーディングの混乱：不可視文字** | 設定ファイル内の隠されたUnicode文字により、悪意ある指示がマスクされる可能性があります。 | LLM03:2025 サプライチェーン |

| カテゴリ | 説明 | OWASP Top 10 for LLM Apps |
|---|---|---|
| **過剰な権限付与** | LLMを活用するエージェントに、関数や外部サービスを呼び出すための幅広い権限を付与すると、誤った解釈や悪意ある振る舞いの可能性が高まります。 | LLM06:2025 過剰な権限付与 |
| **OpenAIの誤設定：未指定のトークン制限** | OpenAIの呼び出しに対して応答サイズやクエリレートを制限していない場合、システムで無制限の消費や予期しないコストが発生する可能性があります。 | LLM05:2025 不適切な出力処理/ LLM10:2025 無制限の消費 |
| **パスマニュピレーション** | 未チェックのファイルパスの入力を許可すると、攻撃者は(「../」などにより)ディレクトリを走査して機密ファイルにアクセスできます。 | LLM02:2025 機密情報漏えい/ LLM05:2025 不適切な出力処理 |
| **プライバシー侵害** | サニタイズされていないプロンプトやトレーニングデータは、個人情報や財務情報を露出させる可能性があります。また、プロンプトインジェクションは、チャットボットにシークレットを取得させる可能性があります。 | LLM02:2025 機密情報漏えい |
| **プロンプトインジェクション** | 慎重に作成された入力によるLLMの指示の上書きやデータの抽出が可能な場合、モデルの動作が意図しない形になる可能性があります。 | LLM01: 2025 プロンプトインジェクション |
| **プロンプトインジェクション：持続型** | データベースまたはログに保存された信頼できないデータがシステムプロンプトに取り込まれると、モデルに持続的に影響を与えたり、モデルを侵害したりする可能性があります。 | LLM01:2025 プロンプトインジェクション |
| **システム情報の漏えい：内部** | エラーメッセージ内のファイルパスやユーザー名、スタックトレースが露出すると、攻撃者が環境に関するインサイトを得ることができます。 | LLM02:2025 機密情報漏えい |
| **無制限の消費の誤設定** | レート制限やトークン制限が設定されていないと、攻撃者が大量または多数の要求を送信することが可能になり、リソースの浪費やコストの増大につながる可能性があります。 | LLM10:2025 無制限の消費 |

### サポートされている言語とフレームワーク

OpenTextアプリケーションセキュリティは、44を超える言語とフレームワークをサポートしており、最新の環境、レガシー環境、AI駆動型の環境にわたって一貫した保護を提供できます。エンタープライズアプリからAIワークロードまで幅広くカバーしているため、コードが記述または導入されている場所にかかわらず、チームは従来の脆弱性と新たなAIリスクの両方を検知できます。

この広範な機能により、AI対応テストと実証済みのAppSecコントロールが開発スタック全体に適用されます。

| 言語グループ | 例 |
|---|---|
| **エンタープライズとバックエンド** | Java、Kotlin、C#、.NET、VB.NET |
| **Web開発とAPI開発** | JavaScript、TypeScript、Python、PHP、Ruby |
| **クラウドとマイクロサービス** | Go、Java、.NET、Node.js |
| **AI/MLワークロード** | Python (TensorFlow、PyTorch)、Java |
| **モバイルアプリケーション** | Android (Java/Kotlin)、iOS (Swift、Objective-C) |
| **ネイティブとシステム** | C、C++ |
| **ERPと専用** | ABAP (SAP) |
| **コードとしてのインフラストラクチャ** | Dockerfiles、Kubernetes YAML、Terraform |

## リソース

**OpenText™アプリケーションセキュリティ**
詳細はこちら ›

**AIとAppSecの詳細**
ブログを読む ›

**量子コンピューティングの課題を把握**
ブログを読む ›

**関連する機能の詳細**
詳細はこちら ›

AI駆動型の開発は新たなリスクパターンをもたらしますが、強力なセキュリティ原則が依然として適用されます。OpenTextアプリケーションセキュリティは、信頼できるテスト機能をAI対応アーキテクチャにまで拡張しています。そこでは、広範な対応言語、詳細な脆弱性検知、継続的に更新される研究に支えられたAIアシスト分析を組み合わせています。

OpenTextは、コード、API、オープンソースコンポーネントにわたってテスト機能を統合することで、迅速かつ一貫した方法により、チームがAI搭載アプリケーションのセキュリティを確保できるようにします。組織は、セキュリティ対策を開発ワークフローおよびビジネス上の優先度と緊密に連携させながら、AIを活用して迅速に物事を進めることができます。

## OpenTextアプリケーションセキュリティの導入オプション：

**OpenTextのクラウドエキスパートと協力してクラウド戦略を加速**

- OpenTextマネージドプライベートクラウド

**チームを拡張**

- お客様の組織またはOpenTextが管理するオンプレミスソフトウェア

**OpenTextパブリッククラウドで、場所に関係なく実行、グローバルに拡張**

- サービスとして消費されるOpenText™ Fortify on Demand (マルチテナントSaaS) [サブスクリプションや従量課金などの価格モデル別]

**任意のハイパースケーラークラウドで、場所に関係なく実行、グローバルに拡張**

- ハイパースケーラークラウドパートナー(OpenTextプライベートクラウド、AWSプライベート/パブリッククラウド、GCPプライベート/パブリッククラウド、Azureプライベート/パブリッククラウド)

**情報管理機能を開発、接続、拡張**

- OpenTextの開発者クラウドのAPI

Copyright © 2026 Open Text • 06.26 | 241-JA0228-003

opentext™