OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Visualizing Secure MLOps (MLSecOps): 堅牢な AI/ML パイプラインセキュリティ構築のための実践ガイド

openssf.org

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

目次

Visualizing Secure MLOps (MLSecOps): 堅牢なAI/MLパイプライン セキュリティ構築のための実践ガイド

対象者

AIアプリケーションにおける機械学習パイプラインの構築とセキュリティ確保に取り組む以下のような幅広い実務者層

- **AI/ML 実務者**（データエンジニア、データサイエンティスト、AI/MLエンジニア、MLOpsエンジニア）：AI/MLソリューションの開発、デプロイ、運用を計画または活用している組織内の専門家。
- **ソフトウェア開発者、コンテナ、クラウドネイティブの専門家：**AI/MLのワークフローや成果物に関わる機会が増えている方々。クラウドネイティブなデプロイやセキュアなソフトウェア開発には精通しているが、AI/MLをアプリケーションに組み込むのは初めてという方。
- **セキュリティ実務者および IT 管理者：**エンドツーエンドのAIアプリケーションに対してセキュアなガバナンスを拡張する責任を持つ方々。これまでソフトウェア開発者向けのCI/CDパイプラインのセキュリティ確保を経験しており、今後はAI/MLを含むアプリケーションのセキュリティを担う必要がある方。
- **AI/MLセキュリティ分野のオープンソースコミュニティ：**特にOpenSSFやその他のオープン標準・フレームワークに関連するコミュニティ。

目的

- **業界向けリソースの作成：**セキュアなDevOpsからセキュアなMLOpsへ、オープンソースツールを拡張する。
- **革新的なビジュアル学習：**概念を画像で層ごとに構築し、説明文で補足する。
- **コードを超えたセキュリティのアンロック：**CI/CDの無限ループ、機械学習ライフサイクル、ペルソナ、サンプルの参照アーキテクチャ、リスクマッピング、セキュリティコントロール、ツールを組み合わせる。

スコープ

- **MLSecOpsに適用可能なDevSecOpsのプラクティスの概要：**DevSecOpsで得られた教訓は、新たに登場するAI/MLライフサイクルにおけるセキュリティ課題への積極的な対応に役立つ。
- **MLSecOpsプラクティスの概要：**MLOpsにセキュリティを統合する重要性を明確化し、その結果としてMLSecOpsを実現する。
- **オープンソース中心：**セキュアなAI/MLアプリケーションやワークロードに適用可能なオープンソースツールやフレームワークをハイライトし、セキュアなAI/MLプロセスを確立することで関連リスクを軽減する。
- **固有のセキュリティリスク：**AI/MLライフサイクルにおける固有のセキュリティ課題を特定し、それらに対処するための推奨事項を提示する。

イントロダクション

ソフトウェア（その多くはオープンソースソフトウェア）を基盤として産業が発展する中で、組織はソフトウェア開発プロセスを開発と運用（DevOps）として体系化し始めました。当初、セキュリティは主に最終ステージのプロセス、あるいは後付けの要素と見なされていました。このアプローチでは、開発初期に導入された脆弱性が検出されないままデプロイまで残り、リスクや修正コストを大幅に増加させることがよくありました。時間の経過とともに、業界はDevOpsから開発・セキュリティ・運用（DevSecOps）へと移行しました。これは、ソフトウェア開発ライフサイクル（SDLC）にセキュリティを統合し、重大なソフトウェアセキュリティのギャップに対応する必要性から生じたものです。

DevSecOpsは、セキュリティプラクティスをDevOpsのワークフローに直接統合することで、これらのギャップを解消しました。この変革により、組織は開発ライフサイクルの初期ステージで脆弱性を積極的に特定・軽減できるようになり、コストのかかるセキュリティインシデントの可能性を減らし、関連する財務的・評判的損失を最小化しました。

現在、業界は同様の転換点にあります。より多くのアプリケーションが、人工知能／機械学習（AI/ML）を組み込むために、機械学習運用（MLOps）を活用するようになっています。AI/ML アプリケーションの開発と運用は、その動的な挙動、固有の複雑性、そしてしばしば不透明な意思決定プロセスにより、新たなリスクの次元をもたらします。従来のソフトウェアとは異なり、MLモデルは継続的に進化するため、AI/ML特有の課題に対応するための適応的かつ継続的なセキュリティ戦略が必要となります。

これらの課題にもかかわらず、AI/ML技術は前例のない利点を提供しており、AI/MLライフサイクルのセキュリティ確保の重要性を強調しています。AI/MLアプリケーションの開発、デプロイ、運用に伴う固有のセキュリティ上の考慮事項に対応するには、DevSecOpsで確立された成功事例と同様に、セキュリティプラクティスを積極的に統合することが不可欠です。

ML運用（MLOps）にセキュリティを統合し、MLSecOpsを確立することは、脆弱性を積極的に特定・軽減するためだけでなく、これまで発見されていなかった欠陥の修正を簡素化し、迅速化するためにも不可欠です。堅牢なMLSecOpsのプラクティスを確立することで、AI/MLシステムはライフサイクル全体を通じて信頼性、耐障害性、そしてセキュリティを維持できます。

本稿では、ソフトウェア開発を保護する中で得られた教訓に基づき、機械学習を安全に利用するために、テキストによって補助された視覚的な「レイヤーごとの」アプローチを、さまざまな実務者に紹介することを提案している。このアプローチでは、Open Source Security Foundation（OpenSSF）の取り組みから、Supply-Chain Levels for Software Artifacts（SLSA）、Sigstore、OpenSSF Scorecardといったオープンソースツールを活用し、それらを MLSecOpsのプラクティスを用いてAI/MLライフサイクルのセキュリティ確保に拡張する機会についても議論します。
さらに、本稿では現行ツールにおける特定のギャップを明らかにし、MLSecOpsの機能をさらに強化するための将来的な開発に向けた推奨事項を提示します。

MLSecOps
DevOpsからDevSecOpsへの移行

DevOpsは、ソフトウェア開発と運用の間に存在していた従来のサイロ構造への対応として登場しました。スピード、自動化、継続的デリバリーという共通の目標に沿ってこれらの機能を統合することで、DevOpsは機能をより迅速に提供し、復旧までの時間を短縮し、信頼性を向上させることを可能にしました。しかし、この加速には代償がありました。それは、セキュリティがしばしば後れを取ることです。

組織がCI/CDパイプライン、Infrastructure- as-Code、コンテナ化されたマイクロサービスを採用するにつれて、攻撃者も同様に迅速に適応しました。迅速なデプロイを可能にした同じツールが、悪用のための新たな攻撃面も生み出したのです。誤って構成されたクラウドリソース、セキュリティの甘い依存関係、検証されていないオープンソースパッケージがすべて標的となりました。セキュリティをリリースサイクルの最後のチェックポイントとして維持できるという前提は、DevOpsの実践において明らかに誤りでした。

DevSecOpsは、この不均衡を解消するために、セキュリティをDevOpsの基盤に直接織り込む形で登場しました。孤立したセキュリティレビューやデプロイ後のスキャンに依存するのではなく、DevSecOps は「シフトレフト」思考を推進します。つまり、セキュリティを設計、コード、テストのステージに早期に組み込むという考え方です。
具体例としては、静的解析、依存関係のスキャン、ポリシーチェックをCI/CDワークフローに直接統合することや、開発者がそもそもセキュアなソフトウェアを開発する方法を理解していることを確保することが挙げられます。セキュリティは、もはや独立した情報セキュリティチームだけの責任ではなく、全員の責任となるのです。

DevOps から DevSecOps への移行に関するより詳細な情報は、2021年頃に公開されました。例として、Cloud Native Computing Foundation（CNCF）の「End User Technology Radar」、米国国防総省（DoD）の「 DoD Enterprise DevSecOps Reference Design: CNCF Kubernetes」、そしてカーネギーメロン大学ソフトウェア工学研究所による「DevSecOps Days Washington DC 2021」が挙げられます。

DevSecOpsへの進化は、単にセキュリティゲートを挿入することではなく、コミットから本番環境まで、ソフトウェアデリバリーの基盤にセキュリティを織り込むことでした。このマインドセットの転換により、チームはセキュリティをコードとして扱い、自動化を取り入れ、イノベーションを妨げることなくセキュアなプラクティスをスケールできるようになりました。

現代のソフトウェアシステムの複雑性が増すにつれて、共有基盤の必要性も高まりました。ここで、OpenSSFのようなコミュニティ主導の取り組みが発展しました。OpenSSFは、言語、エコシステム、ドメインを横断して、セキュアなソフトウェアの作成と運用がどのようなものかを包括的に示す、不可欠なエンドツーエンドの視点を提供します。MLSecOpsの解説に入る前に、より広範なセキュアなソフトウェアサプライチェーンの全体像と、OpenSSFがそれをどのように定義するのに役立っているかを理解しておく事は重要です。

OpenSSFのミッションは、「*私たち全員が依存しているオープンソースソフトウェア（OSS）の開発、保守、利用を持続的にセキュアにすることを容易にすること*」です。これには、*コラボレーションの促進、ベストプラクティスの確立、革新的なソリューションの開発*が含まれます。この目的のために、OpenSSF はワーキンググループ、プロジェクト、関連プロジェクトに分類されたさまざまな技術的イニシアチブを展開し、これらの成果を支援しています。過去2年間で生成AI（GenAI）、特に大規模言語モデル（LLM）の利用が業界で急増した際、OpenSSFのAI/ML ワーキンググループ は、オープンソースソフトウェアのセキュリティ推進者がAIのソフトウェア開発への影響を探求し始める場として設立されました。このグループは、AIアプリケーションでデータやモデルを活用し始めるソフトウェア開発者が、この新しい領域のセキュリティ確保に関する情報を必要とするだろうと仮定しました。さらに、グループは、AIによる多くの技術的改善が従来型のソフトウェア開発者ではなく、データサイエンティストや AI/MLエンジニアによって行われていることに気づきました。これらの非従来型の開発者は、平均すると従来のソフトウェア開発者と比べてセキュリティに関する知識がはるかに少ないのです [Secure Software Development Education 2024 Survey]. AI/MLワーキンググループは、DevOpsのセキュリティ確保から得られた教訓を、データおよび AI/ML エンジニアリングの新しいペルソナに広げることを目指しています。さらに、DevSecOps に精通しているソフトウェア開発者に対して、データや ML モデルをアプリケーションに組み込む新しい運用パイプラインに関するトレーニングを提供したいと考えています。

MLOps

人工知能（AI）は、人間の知能を必要とするタスクを実行する多くの技術を包含する広範な分野です。最近のAIの進歩には、大規模言語モデル（LLM）を活用した生成AIが含まれますが、業界には、生成AIと並行して依然として広く利用されている従来型の機械学習（ML）のユースケースに関するリソースが不足していました。本稿は、セキュアなDevOpsとMLOpsの間に存在する業界のギャップを埋めることに焦点を当て、特にセキュアなMLOpsに重点を置いています。

機械学習運用（MLOps）は、AI/MLシステムのスケーラブルな開発、デプロイ、管理を可能にします。MLモデルの利用が増加するにつれて、それらを開発・デプロイ・管理するための高度な手法の必要性が強調され、MLOpsという分野の人気が高まっています。

MLOps は、DevOpsの原則である自動化、モニタリング、継続的デリバリー、システムの可観測性を、機械学習ライフサイクルに適応させることで拡張します。MLOpsは、データサイエンティスト、MLエンジニア、ソフトウェア開発者、プラットフォームエンジニアの間のコラボレーションを促進し、MLモデルが開発されるだけでなく、一貫性と追跡可能性をもってデプロイおよび維持されることを保証します。

MLOps はDevOpsの影響を受けていますが、いくつかの固有の ML 特性を導入しています。従来のソフトウェアシステムとは異なり、MLシステムはトレーニングデータに大きく依存しており（通常は大量のデータを必要とします）、非決定的な挙動を示し、データドリフトなどの要因によって時間の経過とともに性能が低下する可能性があります。そのため、MLOpsは、データバージョンの管理、モデルの検証、実験の再現、モデルの有効性を維持するための継続的なトレーニングの実現といった課題に対応する必要があります。

MLOps ライフサイクルに関する有用な参考資料として、ML4Devs, が提供するドキュメントがあります。これは、MLOpsに含まれる典型的なステージの内訳を示しています。このドキュメントでは、DataMLとDevOpsの統合を表す図が紹介されています。その図は、計画ステージから、モデルトレーニングのためのデータ収集と変換、評価、コード化、モデルの構築とテスト、デプロイ、そして最終的にはモニタリング、さらに再び計画ステージに戻るという流れを示しています。このドキュメントは、エンドツーエンドで協力するクロスファンクショナルチームの必要性を強調し、早期統合と頻繁な反復の重要性を訴えています。

ML4Devsの記事は、データ、ML、開発、運用という明確な領域にタスクを分類することで詳細なビューを提供していますが、この OpenSSFホワイトペーパーで提示されるフレームワークは、エリクソンが公開したホワイトペーパーに記載されたリファレンスアーキテクチャを基盤としています。このフレームワークは、細分化された活動を「データエンジニアリング」「実験」「継続的インテグレーション」といったより広範なライフサイクルステージに統合し、パイプライン全体に構造化されたセキュリティ対策を適用できるようにします。この一般化により、セキュリティツール、ガバナンスポリシー、リスク軽減戦略を意味のある制御ポイントに統合しつつ、開発と運用において正確かつ有用であることを保証します。この目的は、ML4Devsの記事が示した詳細なビューを置き換えることではなく、MLOpsのエンジニアリング実践とMLSecOpsの実装の両方に整合する統一的な構造を提供することです。

以下では、データ、ML、DevOpsという3つのコア領域を統合し、MLOpsライフサイクルの本質を捉えた概念図を紹介します。DevOpsの無限ループはそれ自体で独立した持続的な概念ですが、本稿ではそれらを1つのループに統合しました。
これは、AI/MLを活用するアプリケーションには「データ」「モデル」「デプロイ」という3つの主要な運用があると提案するためです（デプロイは、コードとソフトウェアを含む従来の DevOps に該当します）。ライフサイクルは計画ステージから始まり、次にデータへ進み、その後モデルへと流れます。モデルは実験を通じて進化し、DevOpsのプラクティスによってスケーラブルで再現可能なデプロイが保証されます。

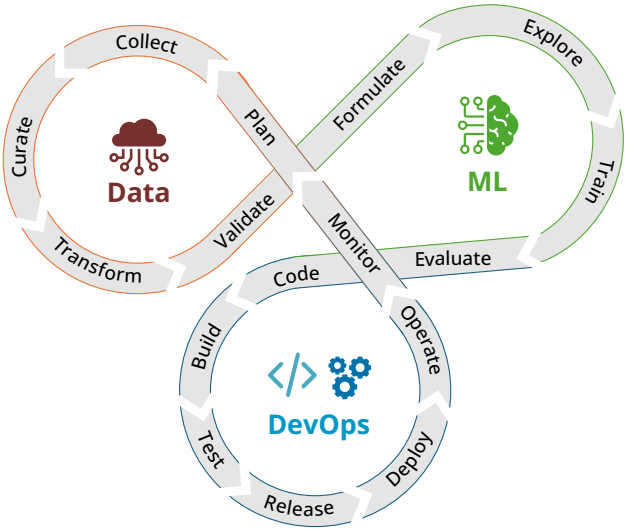

図1: MLOpsはデータ、ML、DevOpsを統合した姿

これを基盤として、同じ図に「MLOpsの計画と設計」から「継続的モニタリング」までの9つの主要なライフサイクルステージをマッピングした第2のビューを重ねます。このアプローチにより、セキュリティコントロールを統合し、ライフサイクル全体でツールを一貫して適用するための構造化された道筋が提供されます。二重の図によるビューは、チームが詳細な MLOpsライフサイクルステージと、MLSecOpsの原則が組み込まれる重要なステージの両方を視覚化するのに役立ちます。

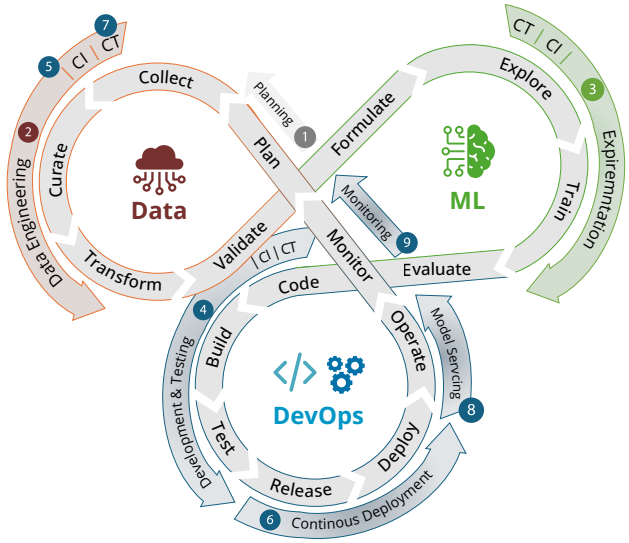

図2: エリクソンのリファレンスアーキテクチャステージにおける MLOpsのライフサイクル

MLOpsライフサイクルは、MLモデル開発の実験的な性質と、本番環境での運用要件の両方を支える一連の調整されたステージにまたがっています。ライフサイクルは、データ、モデル、パイプラインが各ステージを通じて進化し、定期的な適応と改善の必要性を強調する、継続的なループとして視覚化されることがよくあります。図2では、本稿で取り上げる9つの主要なステージに焦点を当てています。それらは次のとおりです。

1. **計画と設計：**ライフサイクルは、アーキテクチャの計画と脅威モデリングから始まります。チームは目的を定義し、リスク（例：モデル盗用、サプライチェーン攻撃）を特定し、セキュリティを最初から組み込むためのツールとコントロールを選択します。
2. **データエンジニアリング：**機械学習タスクに適したデータセットを収集、クリーニング、準備しながら、高いデータ品質と追跡可能性の基準を維持します。
3. **実験：**データサイエンティストは、さまざまなアルゴリズムを探索し、ハイパーパラメータを調整し、性能をテストします。MLOpsツールは、実験の追跡、結果の比較、出力の再現可能な整理を支援します。
4. **ML パイプラインの開発とテスト：**モデルのトレーニングとテストの各ステージを自動化する、再現可能なワークフローを構築し、信頼性を確保するための品質チェックを組み込みます。
5. **継続的インテグレーション（CI）：**コードやモデルの更新は定期的にマージ、テスト、検証されます。CIはライフサイクル全体で行われ、データ前処理スクリプト、MLモデルのコード、パイプラインのロジックなど、あらゆる変更が自動テスト、統合チェック、ポリシー適用を通じて継続的に検証されることを保証します。
6. **継続的デリバリ・デプロイメント（CD）：**モデルはパッケージ化され、自動化されたワークフローを使用して配信または本番環境にデプロイされます。これにより、最小限の手動介入でモデル更新をタイムリーに展開することが可能になります。

7. **継続的トレーニング（CT）：**モニタリングステージの後に続きますが、新しいデータの到着に伴い、以前のライフサイクルステージを繰り返すことを意味します。CTは、データ取り込み、再トレーニング、検証、再デプロイを自動化し、フィードバックループによってデータエンジニアリング、実験、CI活動の再実行がMLOpsライフサイクル内でトリガーされることを示します。
8. **モデル提供：**トレーニング済みモデルは、リアルタイムまたはバッチ推論のためにエンドポイントへデプロイされます。提供インフラは、特に顧客向けアプリケーションにおいて、パフォーマンス、スケーラビリティ、稼働時間を最適化するよう設計されています。
9. **モニタリング：**モデルの挙動を観察・追跡し、性能の低下やデータドリフトを検出し、モデルの再トレーニングやロールバックなど、適切な対応をトリガーします。

これらの各ステージは、MLシステムが長期にわたって信頼性をもって動作することを保証するために不可欠です。これらを組み合わせることで、AIの機能を大規模に、安全かつ効率的に、そして最小限の中断で提供するためのフレームワークが構築されます。
私たちは、無限ループの各ステージを分解し、MLOpsリファレンスアーキテクチャとの整合に備えます。このアーキテクチャは、反復的に MLSecOpsリファレンスアーキテクチャへと進化させていきます。ライフサイクルのステージとフロー自体は変更されません。しかし、それらを密結合した無限ループとして提示する代わりに、より緩やかなフローに整理します。これは、後にリファレンスアーキテクチャと統合するためであり、実際には多くのプロセスが固定された順序を必要とせず、並行して進行するためです。

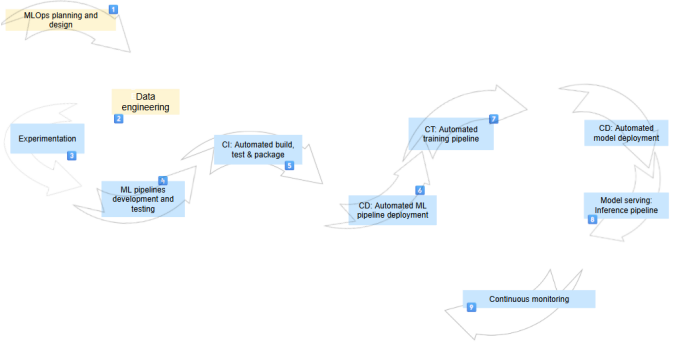

図3: ほどいたMLOpsの無限ループ（リファレンスアーキテクチャにマップするための準備）

MLOpsのアーキテクチャを理解することは、セキュリティを確保するために不可欠です。さまざまなMLOpsフレームワークが存在しますが、本稿ではプロセスとセキュリティ手順を表すために、汎用化されたMLOpsアーキテクチャを使用します。図4に示されるこのアーキテクチャには、自動化された継続的インテグレーション／継続的デリバリ・デプロイメント（CI/CD）システムが組み込まれています。これにより、ML モデルの作成やパイプライン準備における新しい技術の効率的な探索をサポートし、新しいMLコンポーネントの構築、テスト、デプロイのプロセスを簡素化します。

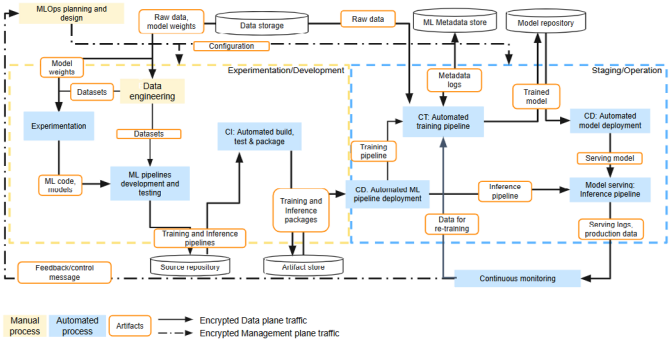

図4: 汎用化されたMLOpsのリファレンスアーキテクチャ

次の図は、MLOpsライフサイクルのコアステージ（曲線の矢印で示される）が、個別のMLOpsステージにどのようにマッピングされるかを示しています。参照アーキテクチャの背後にある淡いグレーの緩やかな無限ループの矢印に注目してください。これは、「無限ループ」が参照アーキテクチャとどのように接続しているかのコンテキストを示しています。

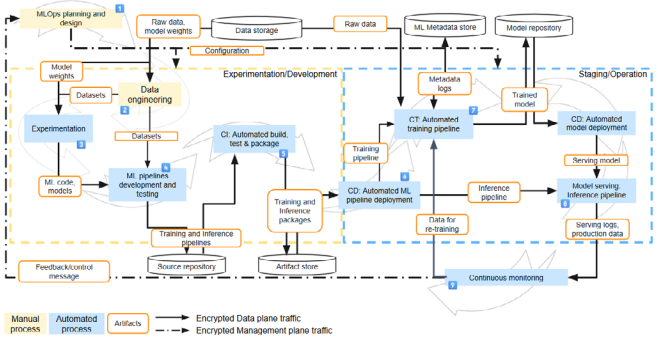

図5: MLOps ライフサイクルの MLOps ステージへのマッピング

この図は、3つの主要なプロセス領域で構成されています：MLOpsの計画と設計、実験／開発、そしてステージング／運用です。各領域には、色付きのブロックで表された特定の MLOps機能が含まれています。MLOpsのフローは、特定の成果物が上流の計画から反復的な開発、デプロイ、モニタリングへとどのように移動するかを示しています。ブロックの色は、手動プロセス（黄色）、自動化されたステップ（青）、成果物（オレンジの枠）を表しています。

図5 から、MLOpsがモデルの開発、デプロイ、保守のエンドツーエンドのライフサイクルをどのように効率化しているかがわかります。しかし、MLシステム特有のリスク環境は、より深いセキュリティへの注力を必要とします。セキュリティを無視したMLOpsプロセスは、DevSecOps を欠いたDevOpsプロセスと同じリスクを抱えます。MLSecOpsを活用して、セキュリティ・バイ・デザインのアプローチをMLOpsに統合することで、基盤となるセキュリティ層がアプリケーションのML開発ライフサイクル内に確立されます。MLSecOpsは、ML開発者、セキュリティ実務者、運用チームの間で責任を分担する「共有責任モデル」によって、あらゆるステップにセキュリティを組み込むことを目指します。後半のセクションでは、MLSecOps の詳細、MLOpsが直面する脅威、そしてMLSecOps で使用されるセキュリティコントロールとツールについて探ります。これらのセキュリティコントロールとツールは、ライフサイクルが単に自動化されスケーラブルになるだけでなく、セキュアであり、より安全な成果を生み出すことを保証します。

次のセクションでは、AI/MLプロセスに必要な役割を補完するために、OpenSSFのペルソナに追加のサブペルソナを導入します。

ペルソナ

機械学習システムを成功裏に運用するためには、優れたモデルやクリーンなデータだけでは不十分であり、学際的なチームによる調整された取り組みが求められます。論文「MLOps: Overview, Definition, and Architecture」では、機械学習製品を設計、構築、デプロイ、維持するために必要な7つの基盤的な役割を特定しています。これらの役割は、従来のソフトウェア、データ、インフラの責任範囲と、MLシステム特有の新しい要件の融合を表しています。

各役割はそれぞれ固有の専門性を発揮し、全体としてMLOpsの協働的な性質を反映しています。ビジネス目標を MLの目的に変換することから、アーキテクチャの設計、データパイプラインの管理、CI/CDの自動化の確保に至るまで、これらのペルソナは本番環境におけるMLの運用基盤を形成します。この論文では、これを「学際的なグループプロセス」と認識しており、役割間の効果的な相互作用は任意ではなく、不可欠であるとしています。

元の MLOps の役割は、本番環境MLシステムに強固な基盤を提供しますが、これらのペルソナはまだOpenSSFによって採用されていません。この課題に対応するため、現実世界の責任に基づき、オープンソースへの関与から得られた知見を反映した、拡張されたAI/MLペルソナのセットを導入します。これらのペルソナは、企業内の専門的な役割を反映するだけでなく、より広範なオープンソースエコシステムにおける貢献者も記述しています。これらを含めることで、データ、コード、インフラ全体にわたる現代の MLパイプラインをセキュアにするために必要な要素をより深く理解することができます。現在の OpenSSFペルソナと提案されるペルソナはここに示されており、新たに追加されたペルソナとサブペルソナの詳細は以下で共有します。

ソフトウェア開発者／メンテナー

ソフトウェア開発者／メンテナーは、既存のOpenSSFペルソナであり、以前からサブペルソナが存在します。本稿では、いくつかの新しいサブペルソナを紹介します。

- ソリューションアーキテクトのサチコ (Solution Architect - SA): サチコは、MLシステムのすべての要素—API、データパイプライン、モデル、クラウドインフラ、その間にあるすべて—がどのように統合されるかを考えることにほとんどの時間を費やしています。彼女はもう大量のコードを書くことはありませんが、何かをセキュアにスケールさせる必要があるときに頼られる人物です。サチコがOpenSSFに関わるようになったのは、セキュアなMLシステムのための堅固なアーキテクチャパターンが十分に存在しないことに気づいたからです。現在では、設計レビューや会議の合間に、上流の作業を取り入れようとしています。
- AI/ML エンジニアのアリソン（MLOpsエンジニア - ME）: アリソンの仕事は、ノートブック上のモデルを本番環境に移し、実際に正常に動作するようにすることです。彼女はデータ、ML、バックエンドの各チームを横断して、モデルの再トレーニング、性能監視、そして本番環境での障害を回避するためのパイプラインを構築します。彼女がOpenSSFに関わるようになったのは、チームがモデルをパッケージ化しデプロイする方法をどのようにセキュアにするかを模索していたときで、その過程でMLパイプラインツールのバグを修正することになりました。
- テストエンジニアのティミー （Test Engineer - TE）: ティミーは、バグが文字通り生死に関わる可能性のあるシステムのテストを担当しています。彼は、自動化テストの作成、コードカバレッジの追跡、APIの検証に慣れています。しかし、MLが組み込まれたことで、すべてがより複雑になりました。突然、出力は確率的になり、テストケースは明確ではなく、性能は入力分布によって変動するようになったのです。ティミーは、MLテストに使えるもの、あるいは貢献できるものを見つけたいと考え、OpenSSFのテスト関連プロジェクトを調べています。

ペルソナ

データオペレーションの実務者は、データセットの作成、管理、ガバナンスの最前線にいます。以下のサブペルソナは、特にデータ入力の品質、出所、コンプライアンスに関して、セキュアで信頼性の高いMLシステムを形成する上で重要な役割を果たします。

- データサイエンティストのダニエル
  (Data Scientist - DS): 機械学習モデルを構築し、性能と倫理的な成果に強く焦点を当てています。彼はしばしば医療などの規制された分野で働いており、セキュリティの専門家になることなく、プライバシー、堅牢性、再現性をワークフローに組み込むツールを求めています。彼の業務は、不正なデータソースや依存関係を強調し、実験全体でセキュリティの状況を可視化するツールによって恩恵を受けます。
- データエンジニアのディビー
  (Data Engineer - DE): 大規模なデータ取り込みと変換パイプラインを管理します。彼女の最優先事項は、データを信頼性があり、一貫性があり、セキュアにすることですが、データセットの完全性を検証したり、改ざんを検出したり、データの系統を追跡したりするための組み込みコントロールが不足していることがよくあります。OpenSSFは、AirflowやSparkなどの一般的なデータツールにセキュア・バイ・デフォルトのプリミティブを組み込み、署名、ハッシュ、出所検証を推進することで、彼女の役割を支援できます。
- データガバナンスアナリストのグリア
  (Data Governance Analyst - DGA): グリアは、MLワークフローで使用されるデータがプライバシー法や社内ポリシーに準拠していることを保証する責任を負っています。しかし、データがMLパイプラインに入った後の流れについての可視性が限られているため、コンプライアンスやリスク評価は遅く、後手に回りがちです。ポリシーメタデータ、監査フック、説明可能性の標準をより適切に統合することで、グリアは手動による強制から、積極的かつ自動化されたガバナンスへと移行できます。

セキュリティエンジニア（プログラムマネージャー、リサーチャー、またはアーキテクト）

セキュリティエンジニアは、既存のOpenSSFペルソナであり、以前からサブペルソナが存在します。本稿では、いくつかの新しいサブペルソナを紹介します。

- セキュリティガバナンスリードのグィネヴィア
  (Security Governance Lead - SGL): グィネヴィアは、ポリシーとエンジニアリングの世界をつなぐ役割を担っています。彼女は、開発環境、ツール、社内インフラにおいて「十分にセキュア」とは何を意味するのかを定義する人物です。コンプライアンス、セキュリティ、プラットフォームチームと協力し、整合性を保ちながらリスクを低減し、開発速度を損なわないようにしています。グィネヴィアは、OpenSSFのポリシーやベストプラクティスに関する取り組みを注視しており、可能なときにはぜひ貢献したいと考えています。
- プロダクトセキュリティエンジニアのパン
  (Security Practitioner - SP): パンは大規模なソフトウェア企業で働いており、開発チームと直接連携してセキュリティの「シフトレフト」を推進しています。パンは、セキュアなSDLC、セキュリティ標準、コントロールに関して豊富な経験を持っています。コードを書くことはあまりありませんが、CI/CDパイプライン、静的アプリケーションセキュリティテスト（SAST）／動的アプリケーションセキュリティテスト（DAST）ツール、開発ツールの統合について非常に詳しいです。パンは、自身をセキュリティポリシーと実際のエンジニアリング制約の橋渡し役と考えています。

  彼の役割には、設計ドキュメントのレビュー、セキュリティ評価の実施、認証、アクセス制御、データ保護、脆弱性修正といったコントロールを製品ロードマップに組み込むことが含まれます。彼は、チームのパイプラインにセキュリティ自動化を組み込むのに役立つオープンソースツール、特にOpenSSFのツールを積極的に探求しています。

ペルソナ

オープンソースプロフェッショナル（OSPO）

オープンソースプロフェッショナルは、既存のOpenSSFペルソナです。本稿では、ITインフラ／プラットフォームエンジニア向けに以下のサブペルソナを紹介します。これらの役割は、現代のソフトウェアや MLワークロードを支える、信頼性が高く、スケーラブルで、セキュアなシステムを確保するために重要です。組織の構造によっては、ITインフラ／プラットフォームエンジニアは、MLSecOpsを支援するために会社のさまざまな部門に属する場合があります。

- **クラウドプラットフォーム管理者のチンマイ**
  (Cloud Admin - CA): チンマイは、データパイプライン、トレーニングジョブ、そして本番環境のML APIを実行するチームのために、クラウドインフラを構築し、セキュアに保ちます。彼はTerraformスクリプト、Helmチャート、そして必要最低限のシェルスクリプトを駆使して、すべてを動作させています。ほとんどの時間は、データサイエンティストが誤って認証情報を公開したり、予算を超過したりしないように支援しています。彼は依存関係を確認するためにOpenSSFのツールを密かに使用しており、チームが構築した強化済みコンテナイメージをアップストリームに提供することを検討しています。
- **ITインフラエンジニアのオフィーリア**
  (Infrastructure Engineer - IE): オフィーリアは、開発、データ、MLチームを支えるツール、可観測性、インフラに関して、舞台裏ですべてを稼働させ続ける役割を担っています。GitHub Actionsが停止したり、デプロイが滞ったりすると、彼女に通知が届きます。彼女はチーム全体で使用されている多くのTerraformやArgoのワークフローを構築しており、常によりクリーンでセキュアな方法を模索しています。オフィーリアは密かにOpenSSFのリポジトリを調査しており、最初のアップストリームへの貢献を行うためには、少しの後押し（そしてオンコールの合間の時間）が必要です。

経営層（CSuite / エグゼクティブ）

既存のOpenSSFのC-Suite / エグゼクティブペルソナにおいて、AI/MLの一環として新しいCレベルの役職が作成されています。これには、データツァリーナ（訳注：ツァリーナは「皇后」「女帝」のことで、データを統括する役割の女性という意味）のドルシラと最高AI責任者（CAIO）のアーチボルドが含まれます。彼らの役割は、取締役レベルでデータおよびAIガバナンスの可視性が高まるにつれて進化していきます。

これらのペルソナはそれぞれ、MLOpsをセキュリティとともに拡張してMLSecOpsへと進化させる方法、またはDevSecOpsのトレーニングをMLOps／MLSecOpsに必要な新しいスキルへと拡張する方法に関する知識の開発に参加しています。

MLSecOps ライフサイクルにおけるペルソナのマッピング

このセクションでは、ペルソナのセクションで定義したペルソナの一部が、MLSecOpsライフサイクルのそれぞれのステージにおいてどのようにマッピングされるかを考えていきます。これらのペルソナについて理解することで、オーナー・協力・サポートがどこで必要になるかを明確にできます。

MLOpsの基盤となるリファレンスアーキテクチャとMLSecOpsの原則をふまえて、AI/MLシステムのセキュア化においては、強調すべきは人間の側面がきわめて大切であることです。各ライフサイクルステージでは、データ取り込みからモデルのデプロイや監視に至るまで、専門的な役割が必要になります。これらのペルソナは、熟練した技術とガバナンスのプラクティスを組み合わせる、役割をまたいだコラボレーションの良い例となります。特に、セキュリティの実務者は、役割をまたいだ守護者として働き、開発から本番までの各ステージにおいて、一貫したセキュリティの監督を担うことになります。ここでは、ライフサイクルのステージとそれが関係する OpenSSFペルソナに結び付いた表を示します。ステージと OpenSSF ペルソナをマッピングすることで、MLOpsライフサイクルにおいて各ペルソナの背景が理解しやすくなるでしょう。

ライフサイクルのステージ	OpenSSF のペルソナ
計画と設計	Pang the Product Security Engineer (SP) Sachiko the Solution Architect (SA) Allison the AI/ML Engineer (ME)
データエンジニアリング	Pang the Product Security Engineer (SP) Dibby the Data engineer (DE)
実験	Pang the Product Security Engineer (SP) Daniel the Data Scientist (DS)
MLパイプラインの開発とテスト	Pang the Product Security Engineer (SP) Daniel the Data Scientist (DS) Danika the Developer Consumer (SE) Timmy the Test engineer (TE)
継続的インテグレーション	Pang the Product Security Engineer (SP) Allison the AI/ML Engineer (ME) Timmy the Test Engineer (TE) Ophelia the IT infrastructure engineer (IE)
継続的デプロイメント	Pang the Product Security Engineer (SP) Allison the AI/ML Engineer (ME) Ophelia the IT infrastructure (IE)
継続的トレーニング	Pang the Product Security Engineer (SP) Allison the AI/ML Engineer (ME)
モデル提供	Pang the Product Security Engineer (SP) Allison the AI/ML Engineer (ME)
モニタリング	Pang the Product Security Engineer (SP) Allison the AI/ML Engineer (ME)

表1: MLOps のステージと対応する機能的役割と OpenSSF のペルソナ

MLSecOps ライフサイクルにおけるペルソナのマッピング

このマッピングによって、組織やオープンソースコミュニティにおいて、ライフサイクルの各時点において誰が責任を持っているかが理解でき、セキュリティとMLライフサイクルの関係者の間での協業ポイントを明確にできます。CxOや役員のようなサブとなるペルソナはこのリストにはありません。本ペーパーは、各ライフサイクルのステージで監督する、より技術的な役割に注目しています。

上記の表は、どの役割が各ステージで関わるかを明らかにしているだけでなく、セキュリティエンジニア・ソフトウェア開発者・オープンソースプロフェッショナルのようなOpenSSFペルソナと、どう関係しているかも表しています。以下では、リファレンスアーキテクチャにおいて、これらのペルソナがどのような姿であるかを示していきます。

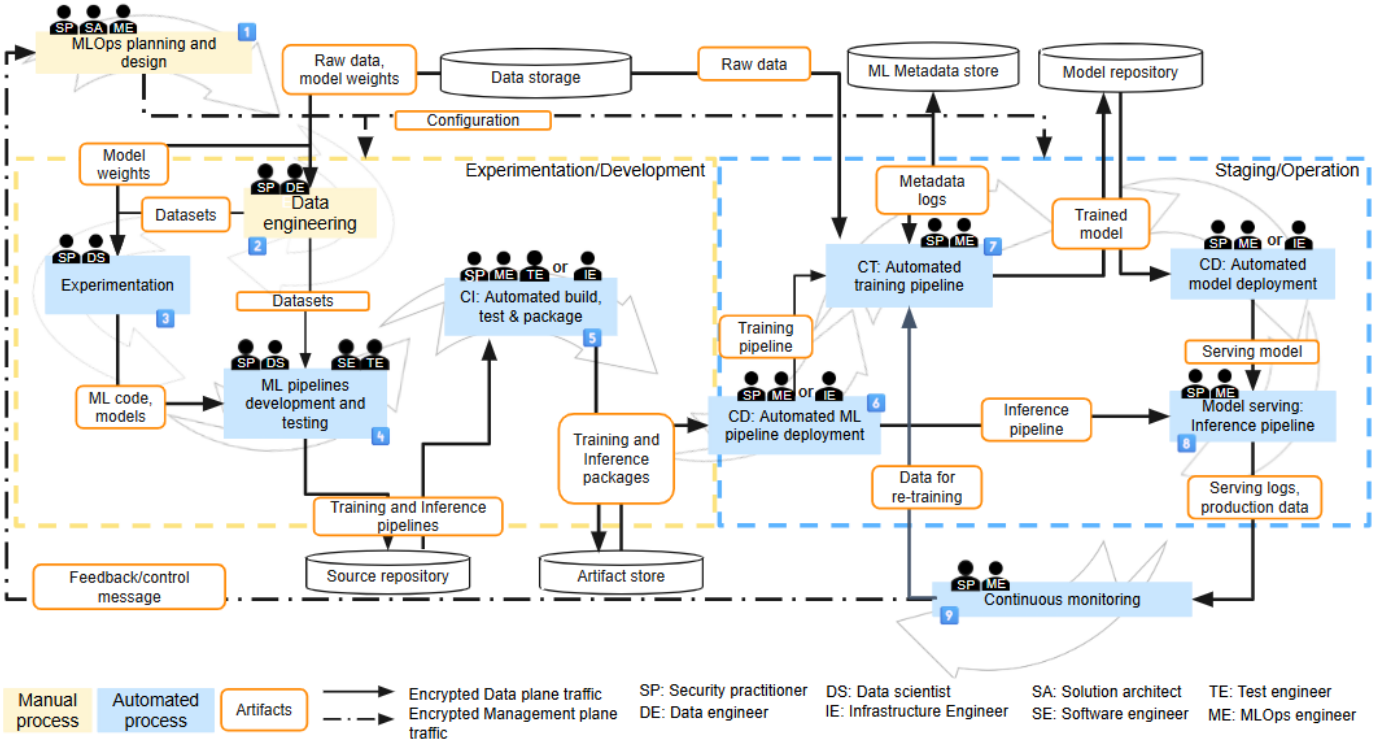

図6: 機能的な役割の MLOps ステージへのマッピング

MLOpsの複数のステージにわたる脅威

このセクションでは、前のセクションで導入したMLOpsの図を元に、OWASP ML Security Top 10 (2023年)で示された重要なMLのセキュリティの脅威を、対応するMLOps のステージにマッピングしていき、その脅威がどのようにAI/MLのライフサイクルに悪影響を及ぼすかを示していきます。

以下がOWASPによる機械学習のセキュリティリスクトップ10です。

- ML01: 入力操作攻撃
- ML02: データポイズニング攻撃
- ML03: モデル反転攻撃
- ML04: メンバーシップ推論攻撃
- ML05: モデル盗用
- ML06: AI サプライチェーン攻撃
- ML07: 転移学習攻撃
- ML08: モデル歪曲
- ML09: 出力完全性攻撃
- ML10: モデル汚染

これらの脅威を機械学習システムの運用ライフサイクルのコンテキストでより理解するために、図7は OWASP ML Top 10 の脅威をMLOps のセクションで明らかにした特定のステージへのマッピングを図によって表しています。

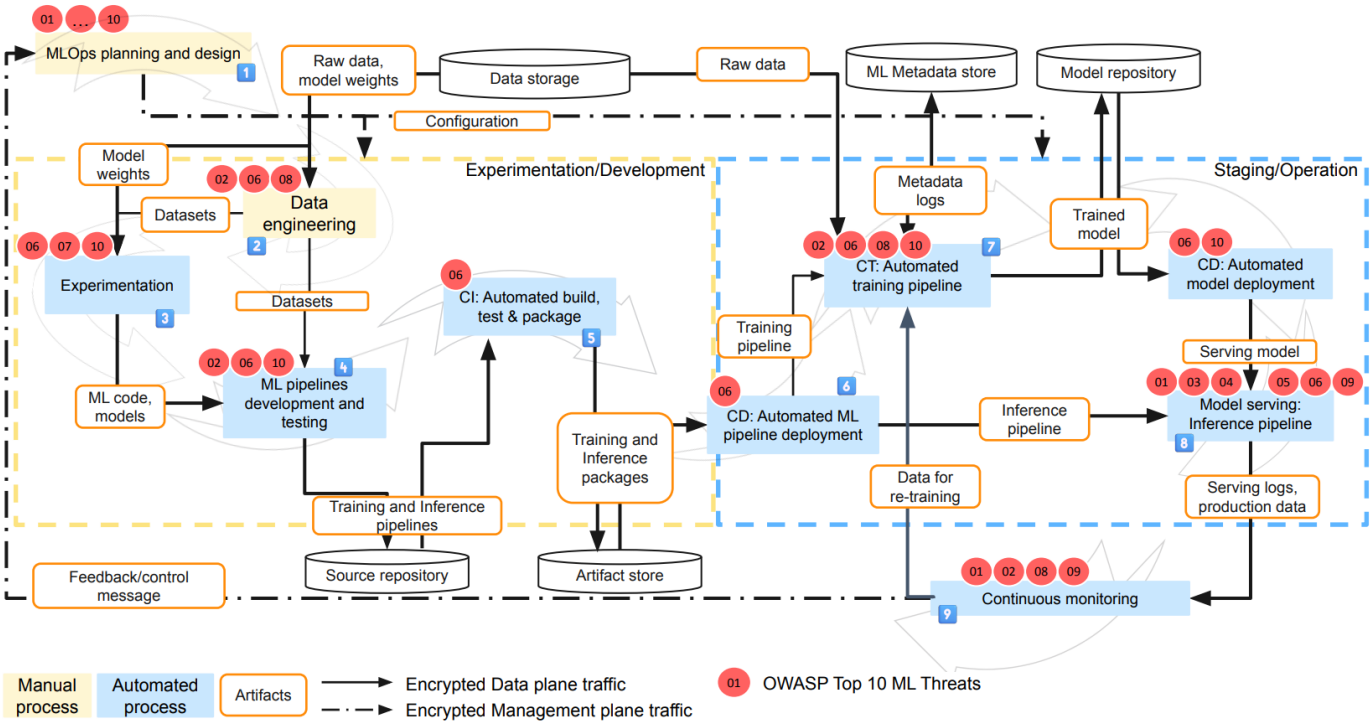

図7: OWASP ML Top 10 脅威とMLOpsステージのマッピング

MLOps の複数のステージにわたる脅威

この図を補完するものとして、表2 はそれぞれの脅威とMLOpsステージの関係をサマリしたものです。この概要により、それぞれのセキュリティリスクが機械学習ライフサイクルのどこで発生する可能性が高いかを理解することができるでしょう。

	MLOpsのステージ	関連する OWASP ML Top 10 の脅威
1	MLOps Planning and Design	全て
2	Data Engineering	ML02: データポイズニング攻撃 ML06: AI サプライチェーン攻撃 ML08: モデル歪曲
3	Experimentation	ML06: AI サプライチェーン攻撃 ML07: 転移学習攻撃 ML10: モデル汚染
4	ML Pipeline Development & Testing	ML02: データポイズニング攻撃 ML06: AI サプライチェーン攻撃 ML10: モデル汚染
5	Continuous Integration (CI)	ML06: AI サプライチェーン攻撃
6	Continuous Deployment (CD)	ML06: AI サプライチェーン攻撃 ML10: モデル汚染
7	Continuous Training (CT)	ML02: データポイズニング攻撃 ML06: AI サプライチェーン攻撃 ML08: モデル歪曲 ML10: モデル汚染
8	Model Serving (Inference Pipeline)	ML01: 入力操作攻撃 ML03: モデル反転攻撃 ML04: メンバーシップ推論攻撃 ML05: モデル盗用 ML06: AI サプライチェーン攻撃 ML09: 出力完全性攻撃
9	Continuous Monitoring	ML01: 入力操作攻撃 ML02: データポイズニング攻撃 ML08: モデル歪曲 ML09: 出力完全性攻撃

表2: MLOpsステージと対応するOWASP ML Top 10 の脅威のサマリ

ステージごとの攻撃の詳細

図と表で示したサマリーを元にして、以下の文章は、それぞれの脅威を一番現れる可能性のあるMLOpsのステージにマップしています。各ステージで、文章はそれぞれのステージがなぜ脆弱であるかを説明し、リスクを攻撃シナリオとともに明らかにしています。このようなより深いコンテキストにより、MLOpsのチームはステージ特有のAI/MLの脅威を理解することができます。各ステージでどのように攻撃が実行されるかを知ることで、MLOpsのチームがリスクにさらされているかを判定し、それぞれのパイプラインで正しいセキュリティコントロールを選択することができるようになります。同じリスクが、異なるライフサイクルステージで現れることもあります。これらの場合、攻撃例は、その攻撃が起きるライフサイクルステージ特有のものとなっています。

1. MLOps の計画と設計

「MLOps の計画と設計」ステージにおいて、不適切なセキュリティ計画は、OWASP Machine Learning Top 10 のすべての脅威がMLライフサイクル内で現れる可能性を高めます。このステージでなされた決定は、後続のセキュリティ体制の基礎となります。以下に、このステージに関係する潜在的な脅威をいくつか挙げます。

ML05 2023 モデル盗用 モデルの保護に対する適切な計画(例えば、暗号化、難読化、セキュアなデプロイ方法)がない場合、権限をもたない人物が、プロプライエタリなMLモデルをコピーしたり複製したりリバースエンジニアリングしたりすることができてしまうことがあります。

- **攻撃例 :** 推論環境の設計が、モデルの暗号化やアクセス制御のような基本的な保護をしていない。この結果、デプロイ環境へのアクセスのある攻撃者は、モデルファイルをサーバーから直接取得できてしまう。また別の例として、レートリミットや出力の難読化のような制限をしない推論APIを設計をしている場合、攻撃者は体系的にモデルにクエリを実行し、MLモデルに直接アクセスする必要なく、モデルの複製を再構築してしまう。

ML06 AI サプライチェーン OSSのライブラリ・フレームワーク・学習済みのモデルやデータセットを含む、セキュアでないサードパーティのコンポーネントを使用することで、初期の設計ステージにおいて脆弱性や悪意あるバックドアを招き、後のセキュリティを損なってしまうことがあります。

- **攻撃例 :** 組織が適切な検証なしに有名なオープンソースの ML ライブラリを自組織のシステムに組み込む。その後、攻撃者はこのライブラリの既知の脆弱性を利用して内部データを侵害したり、モデルの予測を妨害したりする。別の例として、コードが最新のコミットから取得するようなモデルやデータセットがあり、その（コミットを行った）アカウント自体がソーシャルエンジニアリングや内部不正によって乗っ取られている場合。

2. データエンジニアリング

ML02 データポイズニング 生データの改ざんや、ラベルの操作。悪意ある人物は、トレーニングデータやラベルを意図的に挿入・改変・削除することで、モデルの挙動を変えたり、性能を劣化させたり、特定の分類ミスを引き起こします。

- **攻撃例 :** 攻撃者は、SPAM判定のデータセットに改ざんしたデータを挿入し、結果としてスパム判定モデルが悪意あるメールを安全と誤分類するようにし、標的型フィッシング攻撃を可能にする。

ML06 AI サプライチェーン パイプラインのライフサイクルにおけるデータ処理設定やファイルに対する権限のない変更や置き換え。これは、どのデータがどのようにトレーニングに用いられるかに影響する設定ファイル・メタデータ・データへの操作を含みます。

- **攻撃例 :** 攻撃者が異なるソースからどれくらいデータを取り込むかを制御する設定ファイルを変更する。50/50というバランスだったものが、70/30に変更され、信頼性が低かったり、十分に精査されていないデータセットを偏って用いるようになる。各データバケットはある程度検証されているものの、出所やクリーンの厳格さにおける違いから信頼性が異なる。データ自体には変化がなくそれぞれの利用は認められているため、サンプリングでの変化は通常の検証チェックを通る。その結果、モデルは偏っていたり偏りやすい入力を学んでしまい、性能を劣化させてしまう。多くの場合、この攻撃は悪意ある内部の者によるものである。

ML08 モデル歪曲 実行環境を適切に表現していないデータセットによって起こされる性能劣化。潜在的なバイアスや管理されていないモデルの変動につながります。

- **攻撃例 :** 攻撃者は、繰り返し特定の低品質なあるいは間違いのある内容を挿入することで、フィードバックループを悪用する。この歪曲されたデータは、モデルのリトレーニングの一部となり、モデルの推論を徐々に操作していく。時間がたつにつれて、推論が有害で不正なコンテンツを優先するようになる。

3. 実験

ML06 AI サプライチェーン 改ざんされたツールやノートブック、ライブラリや依存関係を実験中に利用すること。攻撃者に対して、悪意あるコードの注入や機密情報の抽出を可能にする可能性があります。

- **攻撃例 :** データサイエンティストが、改ざんされたリポジトリから、悪意を持って改変されたデータ可視化のための依存関係をインポートし、利用する。開発者はトレーニングプロセスをモニタするための依存関係を必要とするが、この依存関係は、機密であるトレーニングデータや知的財産を実験セッション中に外部に流出させる。

ML07 転移学習の悪用 ファインチューニング時のトレーニング済みのモデルのウェイトの悪用。攻撃者が後のモデルリトレーニングでも残存する、巧妙なバックドアや脆弱性をもたらします。

- **攻撃例 :** 攻撃者は、トレーニング済みのMLモデルの改変されたバージョンを、パブリックなモデルハブで公開する。このバックドアが仕込まれたウェイトを使って、研究者がタスク特有のMLモデルをファインチューニングしてしまい、モデルが誤った動作を行うようになる。

ML10 モデル汚染 モデルパラメータの悪意ある操作や意図的な破壊。結果のモデルは、意図しなかったり望まない動作を行うようになります。

- **攻撃例 :** 攻撃者は、ML モデルの正確性を減らすようにハイパーパラメータを変更し、誤分類が起きるようにする。別の例として、悪意ある内部の人間が、あるトリガーとなる特定のフレーズがプロンプトにあるときに、間違った出力をするように、モデルをファインチューニングする場合もある。

4. MLパイプラインの開発とテスト

ML02 データ汚染 汚染されたり悪意をもって改変されたデータを、パイプライン検証で使われるテストデータセットに挿入する。これにより、改ざんされたりバイアスを持ったMLモデルがセキュリティテストを通過してしまうようになります。

- **攻撃例 :** 攻撃者は汚染されたデータをモデル検証やテストデータセットに挿入し、不正確なモデルがパイプラインテストを通るようにしてしまう。

ML06 サプライチェーン 改ざんされたサードパーティの依存関係、ソフトウェアコンポーネントやコンテナイメージをMLパイプラインの開発やテストコードで利用すること。これにより、脆弱性や隠れた悪意ある機能が入ってしまいます。

- **攻撃例 :** MLパイプラインの開発中に、改ざんされたコンテナイメージが利用される。このコンテナは悪意あるロジックを含んでおり、実行時に機密データを漏洩させたり、モデルの出力を破壊したりする。

ML10 モデル汚染 MLパイプラインのコードに隠れたバックドアのロジックや悪意あるトリガーを埋め込むこと。これにより、MLモデルが悪意ある動作をしたり、予測できない動作をしたりします。

- **攻撃例 :** 攻撃者は、自動モデルパイプラインのような、モデルの自動トレーニングコードに隠されたバックドアを挿入する。このバックドアが、特定の入力パターンや条件で発動すると、リトレーニングした ML モデルが意図的に入力を誤分類したり、生成される結果が操作されたものになる。

5. 継続的インテグレーション (CI): ビルド・テスト・MLパイプラインのパッケージ生成の自動化

ML06: AI サプライチェーン攻撃 ML パイプラインの自動ビルドにおける、署名されていないアーティファクトや、改ざんされた依存関係、依存関係の取り違えを狙う攻撃。これにより、悪意あるコードやデータの注入を可能にします。

- **攻撃例 :** 攻撃者は、悪意あるパッケージを、内部依存関係と似た名前のパッケージをパブリックレジストリに公開する。CI システムは改ざんされたパッケージを自動ビルドの中でプルし、マルウェアやバックドアのロジックを ML パイプラインのアーティ

ファクトに注入してしまう。そのほかの事例としては、攻撃者が隠された計算レイヤを直接MLモデルのアーキテクチャに埋め込むようなアーキテクチャ型バックドアがある。このレイヤーは、通常の推論では有効ではないが、入力に特定のトリガーフレーズがあるときに有効化され、操作されたり悪意ある出力をモデルを生成するようにしてしまう。

6. 継続的デプロイ (CD): MLパイプラインとモデルの自動デプロイ

ML06 AI サプライチェーン攻撃 自動デプロイ段階でのモデルパッケージやアーティファクトのすり替えや改変や破壊。これにより、不正な、あるいは、改ざんされたMLパイプラインが本番環境にデプロイされる。

- **攻撃例 :** 攻撃者がアーティファクトストアへのアクセス権を得て、テストされたMLパイプラインを悪意あるバージョンに置き換える。改ざんされたMLパイプラインがCDでデプロイされ、最終的なMLモデルが誤った動きをしたり、予測が劣化したりする。
- **攻撃例 :** 攻撃者がモデルリポジトリへのアクセス権を得て、あるいはデプロイプロセスに介入して、正しいモデルパッケージを改ざんされたバージョンで置き換える。この改ざんされたパッケージ MLモデルの有害な予測を引き起こしたり、性能を劣化させたり、デプロイされると脆弱性をもたらしたりする。

ML10 モデル汚染 自動モデルデプロイのプロセスにおいて、悪意を持って改変された ML モデルウェイトをモデル提供環境にデプロイする。これにより、ML モデルが予測できなかったり、害のある挙動をするようになります。

- **攻撃例 :** パッケージにおいて、攻撃者がバックドアロジックをモデルウェイトに埋め込む。自動デプロイが汚染されたMLモデルを本番環境に置く。特定のインプットでトリガーされると、デプロイされたMLモデルがデータを誤分類したり改ざんされた出力を返したりする。

7. 継続的とレーニング (CT): 自動トレーニングパイプライン

ML02 データ汚染 悪意を持って汚染されたトレーニングデータを本番時に収集してしまい、 MLモデルのリトレーニングに使用してしまうこと。これにより、リトレーニングされたモデルの性能を劣化させたり影響を与えたりします。

- **攻撃例 :** 攻撃者は、間違ってラベル付けされたり壊れたサンプルを継続して収集されるデータストリームに注入する。自動的なリトレーニングで、汚染されたデータセットによって、リトレーニングされたMLモデルが不正確な予測や誤った分類を行うようになる。

ML06 AI サプライチェーン 次のトレーニングラウンドの直前にモデルのアーティファクトやウェイトが改ざんされると、リトレーニングステージが脆弱になります。

- **攻撃例 :** 悪意ある内部の人間が、学習済みのモデルの重みづけを、再学習サイクルのスケジュール直前にわずかに改変したバージョンに置き換える。新しい重みづけは正しいものに見え、再学習は通常通りに進むので、バックドアや挙動の変化が、気づかれることなく入り込む。

ML08 モデル歪曲 本番データを汚染することで意図的に分布ドリフトを引き起こすこと。これは、自動リトレーニングにおいて気づかれずにドリフトを起こすことで、少しずつモデルの挙動を劣化させたり操作することを狙いとしています。

- **攻撃例 :** 攻撃者は、実際の本番環境を反映していない、しかし、正当なものに見えるデータサンプルを注入し、セキュリティ評価をパスする。その後のリトレーニングサイクルを通して、ドリフト検知の仕組みを作動させることなく、モデルはより歪曲されドリフトされ、正しくない決定を生成するようになる。

ML10 モデル汚染 (連合学習の場合) 連合再学習中に発行される、連合学習 (FL, Federated Learning) のクライアントのアップデートにバックドアロジックを埋め込むこと。これにより、集約したグローバルの ML モデルが悪意を持ったり、予測できない形で動作するようになります。

- **攻撃例 :** 悪意あるFLの参加者が、リトレーニング中にバックドアトリガーを含んだ汚染されたアップデートを発行する。これが統合されると、グローバルモデルは、特定の入力パターンやトリガーに出会うと誤動作し、狙った誤分類を行うようになってしまう。

ステージごとの攻撃の詳細

8. モデル提供 : 推論パイプライン

ML01 入力操作攻撃 推論ステージをターゲットとした回避攻撃。悪意を持って作成された入力によって、MLモデルを誤った予測や分類を行うように設計されているものです。

- **攻撃例 :** 攻撃者が、人には普通に見えるが、推論モデルに誤分類や誤った予測を行わせる敵対的な入力を作成する。(例えば、マルウェア検出のためのMLモデルに対して、悪意あるファイルを無害として分類させる)

ML03 モデル反転攻撃 モデルのトレーニングデータセットの再構築にモデルの推論を悪用し、データ漏洩を引き起こすこと。

- **攻撃例 :** 攻撃者はモデル提供インタフェースにクエリを繰り返し行い、出力を分析し、モデルのトレーニングに利用された機密の個人情報を再構築する。

ML04 メンバーシップ推論攻撃 モデルのトレーニングデータセットに特定の個人のデータがあるかどうかを推測し、プライバシー侵害を引き起こすプライバシー攻撃。

- **攻撃例 :** 攻撃者は、体系的にリコメンドモデルのAPIに対してクエリを行い、トレーニングで使われた特定の個人のデータ (例えば、購入履歴や医療記録) を推測し、ユーザーのプライバシーを侵害する。

ML05 モデル盗用 商用のモデルアーキテクチャやパラメータを、公開されたモデル提供インタフェースを通して直接盗む、またはリバースエンジニアリングを行うこと。

- **攻撃例 :** 攻撃者はモデル提供インタフェースにクエリを行い、返される予測や信頼度スコアを分析する。これらの応答を用いて、攻撃者は、商用モデルのパラメータをリバースエンジニアリングし、モデルを再生成する。

ML06 AI サプライチェーン CT ステージの場合と基本的に同じで、モデルアーティファクトやウェイトが、次のトレーニングラウンドの直前に改ざんされること。整合性のチェックがなかったり不十分な場合には、改ざんされたモデルは、検出されることなく、推論にデプロイされてしまいます。

- **攻撃例 :** 悪意を持った内部の人間が、検証されたモデルを、本番にプッシュされる直前に改ざんされたバージョンに入れ替える。パイプラインは気づかずにデプロイを行い、攻撃者の目的にあうように、推論結果が少し偏ったり操作されることになる。

ML09 出力完全性攻撃 推論の出力を操作したり、推論のリソースに過負荷をかけること。これにより、意図的に予測を破壊したり、サービス品質を損なったり、DoSを引き起こしたりします。

- **攻撃例 :** 攻撃者は、推論APIにリソースを消費するクエリを大量に流し、リソース不足を引き起こす。この過負荷によって、性能劣化やレイテンシの問題を引き起こし、信頼性に悪影響を与える。

9. 継続的モニタリング

ML01 入力操作攻撃 ドリフト・エラー率・外れ値検出などのモニタリングの指標を歪ませる攻撃的な入力。これにより偽陽性を作り本当の異常値が無視されることになることがあります。

- **攻撃例 :** 攻撃者は、統計的に極端ではあるが意味的には無害であるような入力を注入し、繰り返しモデルのドリフトや精度低下のアラートを引き起こす。時間がたつにつれ運用チームは頻繁なアラートを無視するようになり、対応が遅れるようになって本当のモデルの障害に気づかなくなる。

ML02 データ汚染攻撃 フィードバックデータがモニタリングのモデルをリトレーニングするのに使われる場合、改変された本番データはこのモデルを汚染して、異常検出やフィードバック制御システムの改ざんにつながる可能性があります。

- **攻撃例 :** 攻撃者は、悪意ある挙動を監視するシステムに、悪意を持って作成したイベントを与える。監視モデルは、この汚染されたデータをもとにリトレーニングされ、次第に将来発生する有害なイベントを無視したり誤分類するようにトレーニングしてしまう。

ML08 モデル歪曲 制御されていないか、攻撃者により起こされたフィードバックループのドリフトがモデルの精度を時間とともに劣化させていくことです。少しずつであるので、分布の変化を注意深く監視しなければ検知することは困難です。

ステージごとの攻撃の詳細

- **攻撃例 :** 攻撃者は繰り返しリコメンドシステムを特定の方法でやり取りすることで、次第に同じようなコンテンツを好むようになり、偏った結果を生成するようになる。

ML09 出力完全性攻撃 推論やモニタリングAPIに対して、クエリや合成したデータを大量に流し込み、オブザーバビリティのリソースを枯渇させたり、性能問題を隠したり、問題検出や緩和システムを阻害したりすること。

- **攻撃例 :** 攻撃者は大量のクエリを生成してモニタリングシステムを圧迫し、性能劣化などの異常をノイズで目立たなくしたり、処理限界により完全にドロップさせたりする。

これらの詳細な攻撃の説明は、攻撃的なアクションがMLOpsライフサイクルのあらゆるステージにおいて実行されうることを表しています。それぞれのコンテキストで脅威を理解することは、有効な緩和戦略を設計し、セキュリティ管理を実装し、AIシステムをセキュアにするのに不可欠です。

MLSecOps ライフサイクルステージにおける脅威を緩和するためのセキュリティ対策とツール

MLOpsは機械学習モデルとパイプラインの、効率化され自動化された開発・デプロイ・運用を重視しています。MLOpsをセキュアにすることは極めて重要であり、そうでなければデータ汚染のような脅威や、MLモデルに対する攻撃、オープンソースライブラリの脆弱性といったものに脆弱なシステムとなってしまいます。MLSecOpsは、MLOpsの不可欠な進化形として現れました。パイプラインにわたって強靭なセキュリティの取り組みを統合し、セキュリティをML開発者、セキュリティ実践者と運用チームの共同責任として確立することで、これらのセキュリティギャップに対処するものです。MLSecOpsは、AI/MLのセキュア化を計画段階から始めること、モデル学習・推論・デプロイ・運用の機密性と整合性を保障すること、そして、セキュアな開発環境を維持することを重視しています。

MLSecOpsを利用することで、セキュリティリスクを早い段階で見つけ緩和することができるようになり、これはセキュアなMLモデルの作成につながります。図8は、MLSecOps レームワークの全体像を示しており、不可欠なセキュリティ対策を明示し、パイプラインにわたるアーティファクトのフローを図示しています。データセット、ML コード、モデル、デプロイ用のパッケージのようなアーティファクトは保護されなければなりません。

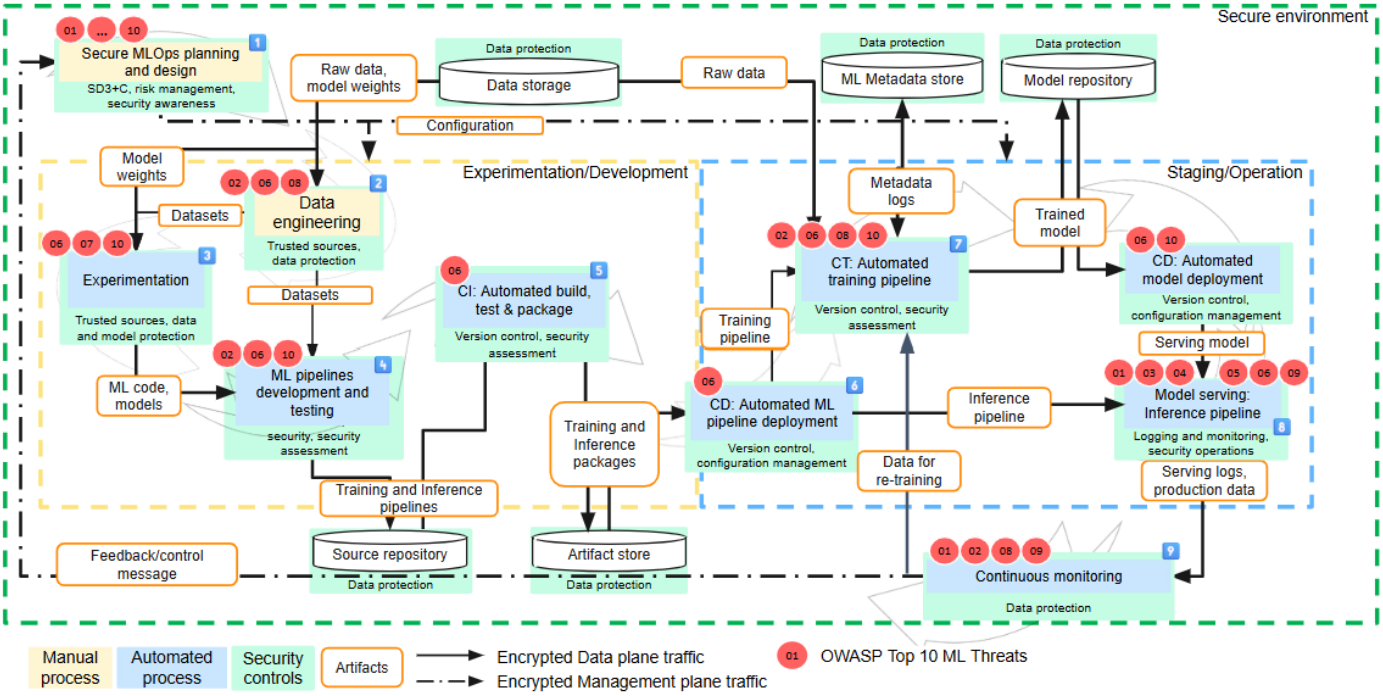

図8: セキュリティ管理の全体

図8の緑の箱はMLOpsライフサイクルの各ステージで導入されるセキュリティ管理項目を表しており、データ保護、バージョン管理、セキュアなデプロイメントといった対策を明らかにしています。

大きな緑の点線で囲まれた部分はMLSecOpsの基礎レイヤーであり、すべてのMLOpsのコンポーネントが信頼された実行・制御コンテキストで動作することを保証しているセキュア環境を表しています。.

セキュアなML開発環境は、MLSecOpsにおいては極めて重要です。潜在的なセキュリティリスクに対して、組織の情報セキュリティ管理システム(ISMS) [ISO27001, 情報セキュリティ管理システム] のガイドラインに合致する形で、注意深い評価と緩和策の実行を行うべきです。具体的には、次のような開発ツールのリスクを考慮する必要があります。

- **セキュアなツール利用：**リスクに応じてセキュアなツールを利用してください。可能であれば、信頼され、かつ承認されたソースから提供されるツールのみを利用してください。重要な場合、レビューやセキュリティ評価を文書化することを検討してください。1 つの方法として、高い価値の資産として分類することでリスクが発生した場合のビジネスへの影響を文書に残します。最低限の対策として、意図しないコンポーネントの混入を防ぐために、タイポスクワッティング攻撃への対策を講じてください。さらに、HTTPSやデジタル署名の検証の仕組みを使用して、ツール提供元の信頼性を高めてください。
- **パッチ管理：すべての開発ツールに対して、セキュリティアップデートやパッチを定期的に追跡、特定、適用する手順を確立してください。**
- **アクセス制御：**開発ツールへのアクセスについて、明確に定義された役割と責任に基づいて厳格に制御し、最小権限の原則に従って実施してください。少なくとも、変更のチェックインおよびビルド結果のプロセスに対して適用する必要があります。

ホスト型クラウドプラットフォーム（HPC）や顧客のプライベートクラウドでは、本番環境への展開を完全に制御できない場合があります。このような場合、セキュリティの責任は組織と環境プロバイダー間で共有されるべきです。さらに、プロバイダーのセキュリティ制御の評価は、全体のリスク評価プロセスの定期的な要素として組み込まれるべきです。

以下は、MLSecOpsを安全に実装するために役立つ仕様やツールの例です。これらが最良または唯一の例ではなく、どのように実現できるかを示す具体的な例となります。

セキュアなMLシステムをより効果的にするため、チームは何に備えてセキュリティを確保するかを知る必要があります。「The Threats Across MLOps Stages Section」セクションでは、OWASP ML Security Top 10 に基づきML特有の脅威を包括的に分類しています。

OWASP ML Security Top 10 の脅威が特定の防御として特定されており、OpenSSFのツールは特にMLソフトウェアサプライチェーンを守るための手段を提供しています。

- **Sigstore**はMLモデルの署名の暗号化を可能にし、モデル関連のサプライチェーン攻撃から保護し、デプロイ時や再トレーニング時の成果物への改ざん防止を実現します。これは、署名が検証され、署名されたアイテムが適切なソースからのものであることが保証された場合にのみ機能します。
- OpenSSF **Scorecard**は、依存関係の更新頻度や脆弱性管理プラクティスの遵守、コードレビューのプロセスといった重要な要素を評価することにより、MLワークフローを含むソフトウェアプロジェクトのセキュリティ体制や全体的な健全性を評価します。古くなったサードパーティ製コンポーネントや保守性の低いプロジェクトを特定し、Scorecardは安全でない前処理コードの悪用や MLパイプラインにおける依存関係の侵害といったリスクを軽減するのに役立ちます。プロアクティブな評価に重点を置くことで、MLシステムの開発ステージからセキュリティが組み込まれ、トレーニング、デプロイメント、推論の各ステージで拡散する可能性のある脆弱性を軽減します。
- Allstarは、ブランチ保護ルール、必須コードレビュー、アクセス管理など、リポジトリレベルのセキュリティコントロールを強制することで、MLのコードベースおよびインフラ構成の整合性を保護します。たとえば、Allstarは、MLモデル、データ パイプライン、またはデプロイメントスクリプトをホストする重要なリポジトリが適切な承認やレビューなしに変更されないようにします。MLのコンテキストでは、データの漏洩、資格情報の露出、またはパイプラインへの悪意のあるコードの混入につながる可能性のある誤った構成や不正な変更を防止します。Allstarは、ソースでのセキュリティ プラクティスを標準化することで、ML運用の信頼性に必要な基本的な制御を強化します。
- **SLSA** (Supply-chain Levels for Software Artifacts)は、特にMLパイプラインに適用可能な Provenanceと整合性レベルを導入します。モデル歪曲やインジェクションが疑われる場合、SLSAレベルの証跡により、モデルがどこでどのように構築されたかを追跡できるため、根本原因の分析と対応が可能になります。

- GUAC (Graph for Understanding Artifact Composition) は、複数のMLパイプラインにわたるモデルやデータの素性を検査するための望遠鏡として利用できます。これにより、誤った予測をモデルが学習したデータセットまで追跡することを支援したり、あるデータソースがライフサイクルの終わりを迎えた際（データ保持規制による場合や、そのデータが悪意あるものであると判明した場合）、どのモデルを再学習させる必要があるかを判断することもできます。

	Design	Data eng.	Exper.	Pipeline Dev	CI	CD	CT	Model serving	Sec. mon.
Sigstore		✔		✔	✔	✔	✔	✔	
OpenSSF Scorecard	✔	✔	✔	✔	✔				
Allstar	✔	✔	✔	✔	✔				
SLSA	✔	✔	✔	✔	✔	✔	✔	✔	✔
GUAC	✔	✔	✔	✔	✔	✔	✔		✔

表3: MLOps ステージと OpenSSF ツールの対応関係

OpenSSFのツールに加えて、OWAPSスイートにあるそれぞれセキュリティツールは従来は Webやアプリケーション向けセキュリティのために開発されましたが、MLSecOpsでも効果的に使用することが可能です。

1. Threat Dragonは、データフローやシステムコンポーネントの視覚的な表現を作成することで、設計ステージで潜在的な脅威をモデル化することを可能にします。ML向けではないが、データ取り込み、モデル学習、推論エンドポイントといったMLOpsコンポーネント全体にわたる脅威シナリオを捉えるために適用できます。
2. CycloneDX は、ソフトウェア部品表（SBOM）を生成するための仕様の一つです。ソフトウェアサプライチェーンの透明性を確保するための成熟した基盤を提供していますが、機械学習パイプラインの主要なコンポーネントについてはまだ対応していません。他に広く利用されている SBOM仕様であるSPDXでは、最新のバージョンでAI関連の情報を記録するための仕組みが含まれています。CycloneDXとSPDXは、言語に依存しないSBOM仕様として最も一般的な二つです。
3. SAMMは、MLシステムの設計と実装をガイドするための成熟度モデルとセキュリティベースラインを提供します。ML特有の拡張は欠いていますが、ソフトウェアのセキュリティ実践を評価し改善するための構造化されたフレームワークを提案します。
4. Dependency-Checkは、オープンソースライブラリに含まれる脆弱性を特定します。特に、前処理、オーケストレーション、可視化などに外部パッケージを多用するMLワークフローにおいて重要です。これらのコンポーネントを早期にスキャンすることで、既知の脆弱性への露出を軽減し、安全なデータパイプラインの開発を支援します。
5. Threat Modeling Cheat Sheetは、一般的なガイダンスであり、外部データやモデルを含む実験を計画する際に適用されます。
6. Dependency-Trackは、CIビルド中にソフトウェアコンポーネントのリスクを監視し、脆弱性が発見された場合に警告します。

MLSecOps ライフサイクルステージにおける脅威を緩和するためのセキュリティ対策とツール

図9は、MLOpsライフサイクルの主要なステージにおけるOpenSSFとOWASPツールの統合を視覚化しています。緑色の番号は特定のMLOpsステージに対応するOWASPツールを示し、星印はOpenSSFツールを表しています。この階層的なビューは、各ツールがライフサイクル内でセキュリティカバレッジにどのように貢献しているかを強調しています。これらのツールは、従来のアプリケーションセキュリティの実践において基盤的な役割を果たします。

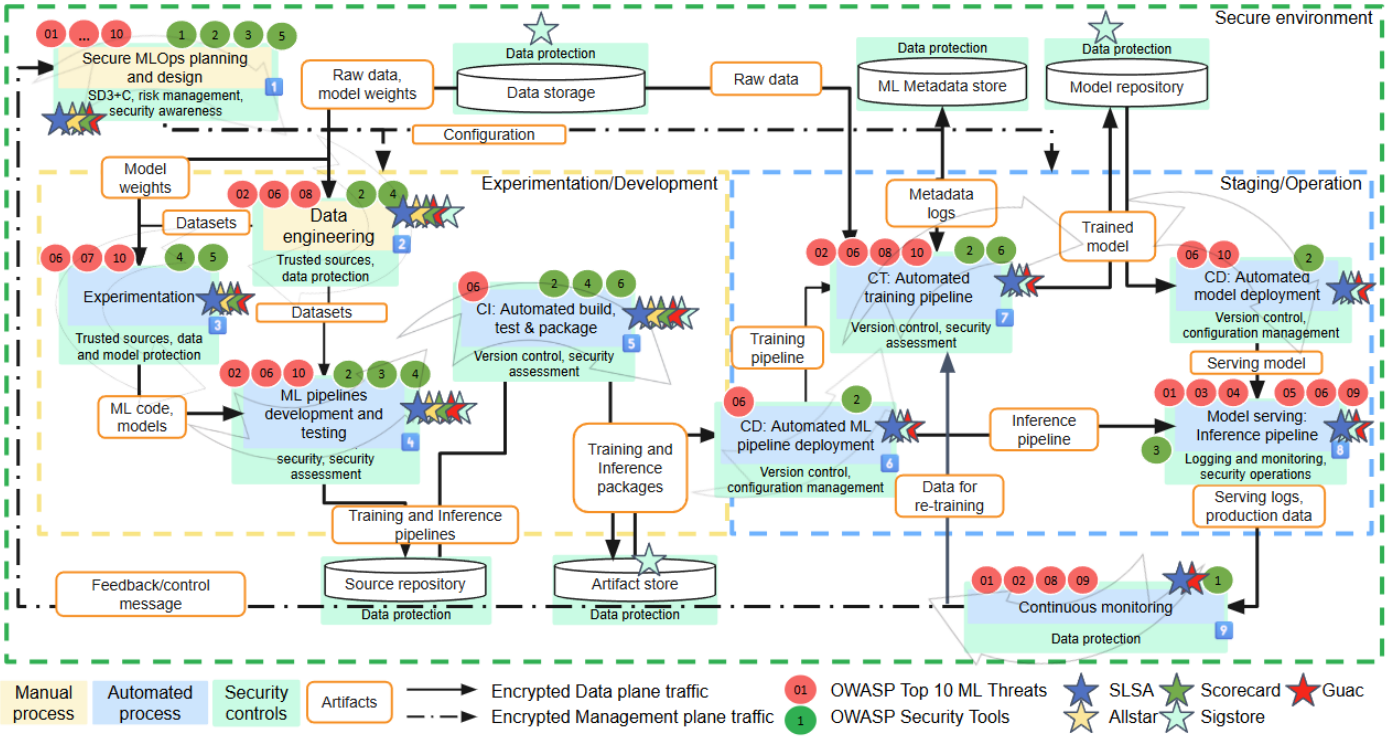

図9：セキュリティ対策とツールの MLSecOps ステージへの対応

Dependency-CheckやDependency-Trackといったツールは、ライブラリのスキャンや推論 API のセキュリティ確保に部分的には役立ちますが、モデルのウェイト、学習データの来歴、敵対的耐性といったML特有の成果物への対応には不十分です。CycloneDXのような仕様やThreat Dragonのようなツールは、SBOM生成や脅威モデリングの強固な基盤を提供しますが、MLワークフロー、データパイプライン、モデルライフサイクルを正確に表現するためには、ターゲットを絞った拡張が必要です。同様に、SAMMは有用な成熟度フレームワークを提供しますが、継続的な学習、モデル再学習のリスク、推論時のセキュリティについてはカバーしていません。

私たちの分析は、重要な洞察を強調しています。OWASPおよびOpenSSFツールは依然として重要ですが、MLSecOpsは、AIシステムの包括的な保護を確保するために、特にモデルの透明性、データの整合性、再現性に関する既存の機能の拡張を要求する新しい要件を導入します。

OWASPやOpenSSFのツールに加えて、オープンソースコミュニティ全体には、より広範なセキュリティソリューションのエコシステムが存在します。これには、データ検証、モデルの説明可能性、敵対的耐性、ランタイム動作監視のためのツールが含まれます。awesome-MLSecOpsのようなリポジトリは、これらのツールのコミュニティによってキュレーションされた一覧を提供しています。ツールの範囲はさまざまですが、多くはMLSecOpsライフサイクルにおいて重要な防御的役割を果たしています。

MLSecOps ライフサイクルステージにおける脅威を緩和するためのセキュリティ対策とツール

次の表は、サポートされる特定のMLSecOpsライフサイクル ステージに沿って、確立されたツールと新たに登場しているツールを統合して示したものです。この視点は、MLパイプラインの様々なステージで生じるセキュリティ上の懸念に対処するための実務者向けのガイドとなることを目的としています。一部のツールは複数のライフサイクルステージに適用されるため、重複して適用される場合があります。

	MLSecOps Stage	セキュリティ対策と運用方法	ツール：（網羅性なし）
1	セキュアなMLOpsの計画と設計	脅威モデリング、セキュア設計パターン	OpenSSF: Scorecard, Allstar, SLSA, GUAC OWASP: Threat Dragon, CycloneDX, SAMM, Threat Modeling Cheat Sheet Open Source Community: SPDX, Syft, Adversarial ML Threat Matrix
2	データエンジニアリング	データの検証、バージョニング、保護。異常検知、データリネージ追跡	OpenSSF: Sigstore (model signing), Scorecard, Allstar, SLSA, GUAC OWASP: CycloneDX, Dependency-Check Open Source Community: Deequ, Great Expectations, Data Version Control DVC, ARX Data Anonymization, YData Profiling
3	実験	サプライチェーンセキュリティ、モデルのバージョン管理、汚染されたデータの検出	OpenSSF: Scorecard, Allstar, SLSA, GUAC OWASP: Dependency-Check, Threat Modeling Cheat Sheet Open Source Community: MLFlow, DVC, ART (Adversarial Robustness Toolbox), NB Defense
4	MLパイプラインの開発とテスト	再現性、セキュアなアーティファクト検証、パイプラインにおけるCIテスト	OpenSSF: Sigstore (model signing), Scorecard, Allstar, SLSA, GUAC OWASP: CycloneDX, Dependency-Check, SAMM Open Source Community: MLRun, AFL++, Giskard
5	継続的インテグレーション（CI）	静的/動的解析、ポリシー適用、依存関係のスキャン	OpenSSF: Sigstore (model signing), Scorecard, Allstar, SLSA, GUAC OWASP: CycloneDX, Dependency-Check, Dependency-Track Open Source Community: ModelScan, Grype
6	継続的デプロイメント（CD）	セキュアな配備の自動化、モデル成果物のチェック、セキュアなソースからのパッケージのインストール	OpenSSF: Sigstore (model signing), SLSA, GUAC OWASP: CycloneDX Open Source Community: Jenkins, ArgoCD, Bandit
7	継続的トレーニング（CT）	継続的なデータ検証、ドリフト検出、モデルのバージョン管理、フィードバックデータの継続的な認証	OpenSSF: Sigstore (model signing), SLSA, GUAC OWASP: CycloneDX, Dependency-Track Open Source Community: Whylogs, TensorFlow Privacy, Evidently

	MLSecOps Stage	セキュリティ対策と運用方法	ツール: (網羅性なし)
8	モデル提供 (推論パイプライン)	入力検証、アクセス制御、 モデル透かし、出力フィルタリング	OpenSSF: Sigstore (model signing), SLSA OWASP: SAMM Open Source Community: Garak, Seldon Core, ProtectAI/LLM-guard, TextAttack, Foolbox
9	継続的モニタリング	ドリフト検出、異常検知、 アラート、敵対的監視	OpenSSF: SLSA, GUAC OWASP: Threat Dragon Open Source Community: Evidently, WhyLogs

表4: MLSecOpsのステージとトップ10の脅威の概要

表4 は、計画とデータ エンジニアリング、展開、継続的な監視まで、MLSecOpsがMLOpsライフサイクルの各ステージにわたってセキュリティの認識とツールをどのように拡張するかを示しています。各ステージをターゲットプラクティスと関連ツールに合わせることで、MLシステムの整合性、機密性、可用性を脅かすリスクを積極的に特定、軽減、対応できます。ここに挙げたツールはすべてを網羅しているわけではありませんが、MLパイプライン全体にわたる多層防御を可能にする、オープンソースおよびエンタープライズ対応ソリューションの、成長を続けるエコシステムを表しています。この分野が成熟するにつれて、組織はセキュリティ体制を進化させ続け、従来のソフトウェアセキュリティ原則を適応させ、ML駆動型システムの特有の変化に対応する必要があります。

セキュアなMLOpsの設計

セキュアなMLOps設計には、計画、開発、導入、運用を含むMLライフサイクルにセキュリティプラクティスを統合することが含まれます。

セキュアなMLOps設計では、以下の点に留意します。

- MLOpsの主要な原則、コンポーネント、および役割を特定すること
- MLOpsアーキテクチャを包括的に理解すること
- ワークフロー(MLOpsプロセス全体で実行されるタスクのシーケンス)を定義すること

セキュリティ対策の設計

セキュリティベースラインを確立することは、AI/ML開発ライフサイクルの基盤となります。ベースラインは、AI/MLシステムへの脅威を考慮し、最低限のセキュリティ統制、ベストプラクティス、ガイドラインを網羅しています。これは、AI/MLシステムとデータを保護するための出発点となります。

ベースラインが確立されると、セキュリティ リスク評価 (RA) によってAI/MLリスクが特定され、優先順位が付けられ、MLOpsプロセスを通じて効果的なリスク軽減戦略が可能になります。

リスク評価に役立つツールと、参考となる資料は以下のとおりです。

- “Modeling Threats to AI-ML Systems Using STRIDE,” にある STRIDEフレームワークは、脆弱性に対処し、ツールを規定しています。
- “Microsoft AI Security Risk Assessment” は、 包括的な分析を提供しています。
- MITRE’s ATLASは、戦術、手法、ケーススタディに関するデータを提供しています。
- OWASP ML Security Top Ten.

- National Institute of Science and Technology (NIST) AI Risk management Framework (RMF) は、リスク管理に関するガイダンスを提供しています。
- ISO/IEC 23894: Information technology — Artificial intelligence — Guidance on risk management.

異常検知モデルが異常な動作を検知し、解決策を自動化するシナリオを考えてみましょう。ソフトウェア設計時にデータポイズニング、モデル回避、サービス拒否（DoS）などのリスクを特定することで、重要な安全対策を実装できます。シームレスな統合を実現するには、これらのセキュリティコンポーネントをMLOpsプロセスと連携させる必要があります。これにより、MLモデルやパイプラインなどのML 成果物と並行して開発とテストが行われます。

安全なMLOps設計プロセスでは、特定のプロセスやセキュリティ制御を含む、MLOpsアーキテクチャ全体の安全な構成も必要です。

設計ツール

MLOpsライフサイクルの計画や設計ステージにセキュリティを統合するには、概念的なフレームワークだけでなく、安全なアーキテクチャの定義とリスク評価をサポートする実用的なツールも必要です。OWASPやより広範なMLSecOpsエコシステムが提供するオープンソースツールは、このステージで特に役立ちます。

- Threat Dragon: 安全な設計ステージにおいて、チームはMLOpsアーキテクチャを視覚的にモデル化し、潜在的なセキュリティ脅威を特定し、開発ライフサイクルの早いステージで緩和戦略を文書化できます。これは、正式な脅威モデリングを初めて行うチームにとって特に役立ちます。
- CycloneDX, SPDX,Syft: SBOMを生成することで、これらの仕様とツールはオープンソース パッケージ、データセット、モデル成果物への可視性を提供し、AIサプライ チェーンに関連するリスクの軽減に役立ちます。(例 : ML06: AI サプライ チェーン攻撃)
- Adversarial ML Threat Matrix: 戦術ガイドとマッピングシステムは、チームがデータ汚染、モデルの盗難、メンバーシップ推論などの脅威を特定し、それらの軽減策を設計レベルに組み込むことに役立ちます。

ML専用の設計ツールが存在しないと、死角が残ります。例えば、事前学習済みモデルやサードパーティから取得したデータセットの整合性検証が計画に含まれていない可能性があり、これはサプライチェーンの隙となります。侵害されたモデルがベースラインとして使用される場合 (ML07: 転移学習攻撃)、対策が計画されていないバックドアが本番環境に残る可能性があります。同様に、モデル成果物に対する強力なアクセス制御の設計が不足していると、モデルの盗難や改ざんが発生する可能性があります。基本的に、OWASP ML Top 10 を考慮した事前のセキュリティアーキテクチャがなければ、下流の制御は事後対応となる場合や不十分となる可能性があります。設計ステージで、OWASP、OpenSSF、その他のオープンソースコントリビューターが提供する強力なオープンソースツールの基盤の恩恵を受けられますが、よりMLに特化した自動化ソリューションが必要です。オープンソースコミュニティには、既存ツールの拡張や、MLシステムの独自のアーキテクチャコンポーネントに合わせたセキュリティバイデザインを実現する新しいツールを開発する機会があります。この分野への投資は、MLSecOps導入の初期ステージを大幅に強化するでしょう。

データエンジニアリングライフサイクルステージ

データエンジニアリングは、生データを入力として受け取り、後続のプロセスに必要なデータセットを生成します。特に多様なソースからデータを集約する場合は、データの取得や検証、保存にセキュリティポリシーと制御を適用する必要があります。

- 収集されたデータには、機密性の高い個人情報が含まれる可能性があります。機密データの処理には、適切な法的権限と契約上の権限が必要です。
- 実際には、データパイプラインは複数のプロバイダーやWebクロールからコンテンツを取り込む可能性があり、それぞれ信頼性、対象範囲、リスクが異なります。したがって、データバケットには信頼レベルを明示的に割り当て、アクセス制御は機密性と検証ステータスを反映させる必要があります。該当する場合、低い信頼レベルから取得されたデータは、適切にエスカレー

ションし精査される必要があります。より大規模で多様なデータセットが使用されるため、この原則を適用することは困難であり、将来のデータセキュリティの革新の余地を残します。実装は、各組織における高リスクユースケースと見なされる機密データセットに対するシステムのリスクに依存します。

データ内の不要な情報や悪意のある情報は、パフォーマンスに影響を与えたり、悪意のある動作を引き起こしたりする可能性があります。また、保存データの改ざんも同様です。こうした事態を回避し、プライバシーを維持するには、保存中のデータを適切に保護する必要があります。

データエンジニアリングのセキュリティ対策

以下のセキュリティ対策を実装する必要があります。

- データ保存には、データの機密性に応じたセキュリティ管理を採用する必要があります。
- 機密性のために暗号化が必要な場合は、強力な暗号化アルゴリズムを使用します。
- 保存データの整合性は保護される必要があります。
- 信頼性の異なるバケットには個別のアクセス制御を使用し、データクリーニングジョブによって信頼性の高いデータセットと信頼性の低いデータセットが誤って混ざり合わないようにします。
- データ アクセスは、正式なアクセス制御プロセスを使用して監視および記録する必要があります。
- バージョン管理と正式な変更管理プロセスを実装します。
- データ処理パイプラインでは、パイプラインとデータセットを個別に改善できるように、元のデータを削除したり置き換えてはなりません。
- 割り当てられた信頼レベルでデータ ソースにラベルを付け、パイプライン全体の系統を追跡します。
- 定期的に脆弱性スキャンを実施し、更新されたツールを使用して潜在的なデータ汚染の脅威を特定します。
- 定期的にデータのバックアップとリカバリテストを実施します。
- 保存期間と安全な廃棄に関するデータ保持ポリシーを実施します。

データ品質はモデル品質に大きく影響するため、開発ライフサイクル全体を通じてデータ品質を最大化する必要があります。同様に、開発中および生産中にデータ保護対策を実施する必要があります。開発中は、取得したトレーニングデータを信頼性、異常、隠れた操作の有無について継続的に評価する必要があります。異常検知など、特定のケースでは、異常が現実世界の出来事である場合もあれば、悪意のある攻撃の結果である場合もあるため、分析が困難な場合があります。したがって、疑わしいデータを効果的に分析するには、専門家の関与が不可欠です。

データエンジニアリングツール

MLOpsライフサイクル内での安全なデータエンジニアリングの実践をサポートするために、オープンソース コミュニティのさまざまなツールが、データの整合性、プライバシー、アクセス、および工程証跡に対する制御の強化に役立ちます。

- Great Expectations と Deequ: データセットの検証とプロファイリングに広く使用されている 2 つのツール。取り込み時のデータ品質の確保、スキーマの異常の検出、ビジネスルールの適用を支援し、汚染されたデータや不正な形式のデータに対する最前線の防御線となります。
- DVC (Data Version Control): 堅牢なデータセットのバージョン管理をサポートし、トレーサビリティとロールバック機能を実現します。アクセス制御されたストレージや CIパイプラインと統合でき、正式な変更管理および安全な保持ポリシーに準拠します。
- ARX Data Anonymization Tool: プライバシー強化技術に関連して必要となります。このツールは、機密性の高いデータセットを扱う際に不可欠な、構造化データの匿名化と画像 / 動画の編集を可能にします。
- YData Profiling: 視覚的なプロファイリングレポートを備えた自動探索的データ分析を提供し、本番環境への統合前に、欠落データ、外れ値、潜在的な危険信号を特定するのに役立ちます。

実験

データサイエンティストは、実験を行う際に、MLモデルのエンジニアリングを行い、特徴量やアルゴリズムを選択または新規開発し、モデルをトレーニングし、ハイパーパラメータを調整します。入力はモデルのウェイトとデータセットで構成され、出力にはMLコードとMLモデルが含まれます。

不適切または安全でないコードは、可用性、整合性、または機密性のリスクにつながる可能性があります。セキュリティの実務者は、以下の要件を考慮する必要があります。

- 安全な環境で設計と調査を実施する
- 開発の初期ステージでモデルの選択をレビューし、承認してから本番環境に導入する。モデル全体を追跡する
- 実験と関連する結果を文書化する。実験とモデルトレーニングの完全なトレーサビリティを実現する

理想的な条件下で学習された機械学習モデルは、潜在的に敵対的な環境にデプロイされると脆弱になる可能性があります。指標とテストセットは、さまざまな種類のドリフトと想定される敵対的状況をエミュレートする必要があります。

実験のセキュリティ対策

以下の対策を講じる必要があります。

- トレーニング、検証、テストセットが自然な時間的依存関係に準拠していることを確認する
- データセットに、一般的に発生する可能性のある破損を組み込むことで、モデルの堅牢性を高める
- 敵対的サンプルが懸念される場合は、敵対的トレーニングを検討する
- トレーニングに分散データを使用する場合は、プライバシーに関する懸念を軽減するために、連合学習（フェデレーテッドラーニング）を検討する

バージョン管理と整合性保護を使用して、MLモデル内の不正な変更を検出し、ポイズニングの検出に役立てる必要があります。署名され整合性が保護されたバージョンは、破損が発生した場合でも、既知の正常な状態に戻すことを可能にします。

MLモデルを転送する際には、不正な変更から保護することが重要です。標準的なアプローチとしては、モデルに暗号ハッシュ関数を適用することが挙げられます。ハッシュは暗号化するか、別のチャネルで転送する必要があります。

学習済みモデルは知的財産であり、透明性、セキュリティリスク評価が求められ、モデルの相対的な価値に応じて保護されるべきです。学習スクリプトや特徴量エンジニアリングコードは、さらに高い知的財産価値を持つ可能性があり、保護する価値があります。モデル、学習、特徴量計算は、機密コンピューティング、暗号化、難読化によって保護できます。

MLモデルの検証手順には、包括的なセキュリティテストを含める必要があります。通常のテストでは、簡単なチェックで侵害された MLモデルを検出できますが、高度なテストでは、より幅広い攻撃により脆弱性を特定します。さらに、ペネトレーションテストのために、カスタムシナリオや脅威ベースのシナリオを開発する必要があります。

実験ツール

実験中のセキュリティは、再現性、トレーサビリティ、堅牢性テスト、敵対的防御をサポートするツールを統合することで強化できます。

以下のオープンソースツールは、チームがこのステージで安全なプラクティスを適用するのに役立ちます。

- MLflow: 機械学習ライフサイクル管理において最も広く利用されているプラットフォームの一つです。実験の追跡、モデルのバージョン管理、成果物のログ記録をサポートしており、これらを組み合わせることで、トレーニング活動とモデルの系統の完全なトレーサビリティを実現します。このトレーサビリティは、監査可能性とモデル成果物の改ざん検出に不可欠です。
- DVC: データ、モデル、MLパイプラインにGitのようなバージョン管理機能を追加します。データセットやトレーニングスクリプトへの不正な変更を検出するのに役立ちます。
- Adversarial Robustness Toolbox (ART): 機械学習モデルの堅牢性をテストするための攻撃と防御のスイートを提供します。セキュリティの実務者は、回避

攻撃、ポイズニング攻撃、推論攻撃に対するモデルの脆弱性を評価し、防御強化のための敵対的トレーニングをサポートします。

- モデルの透明性（Sigstore 経由）: 機械学習モデルのデジタル署名と検証を可能にし、整合性と真正性を確保します。このツールにより、チームは重要な開発ステージ（トレーニング、デプロイ、再トレーニングなど）でモデルに暗号署名を行い、改ざんやサプライチェーン攻撃から保護することができます。

MLパイプラインの開発とテスト

自動化されたトレーニングおよび推論パイプラインは、継続的なトレーニングとモデル提供に使用されます。このプロセスでは、実験ステージのデータセット、MLコード、モデルをトレーニングおよび推論パイプラインの入力と出力として受け取ります。MLパイプラインの作成は、他のソフトウェア開発プロセスと同様に、ソフトウェア開発ライフサイクル（SDLC）に従う必要があります。

パイプラインのセキュリティには、バージョン管理、整合性、機密性の保護が含まれます。パイプラインの保護はMLモデルの保護に似ています。どちらも、機密性、整合性、アクセス制御、そして確立されたポリシーと規制へのライフサイクル全体のコンプライアンスに注意を払う必要があります。

MLパイプラインで使用されるMLコードとパラメータに関するセキュリティ上の考慮事項を定義し、合意する必要があります。セキュリティ対策はパイプラインの開発中に実装する必要があり、セキュリティテストのプラクティスはパイプラインのテストに適用できるように調整する必要があります。

セキュリティ対策の開発とテスト

パイプライン開発においては、以下のセキュリティ対策を検討してください。

- OWASPのソフトウェア保証成熟度モデル (SAMM) は、ソフトウェア開発と保守にセキュリティ活動を組み込むためのフレームワークを提供します。
- コードレビューまたはピアレビュー（ソフトウェアエンジニアが実施するものを含む）。
- 静的アプリケーションセキュリティテスト (SAST) は、ソースまたはコンパイルされたバイナリを分析することで、実行せずにソフトウェアのセキュリティを検査します。
- 動的アプリケーションセキュリティテスト (DAST) は、実行を通じてランタイム環境のソフトウェア セキュリティを評価します。
- ファジングテストは、実行時に無効な入力 (ランダムに生成されたもの、または特別に作成されたもの) を提供し、パイプラインのクラッシュ、バッファオーバーフロー、その他予期しない結果を監視します。
- インターフェース (API) テストには、MLパイプラインのさまざまな部分で作業する複数のチームが関与します。
- 誤用または乱用ケースのテストでは、ユーザーが入力を操作して破損したMLモデルを作成する試みをシミュレートします。

継続的な学習において、自動学習パイプラインは、同じ入力が使用されている限り、実験中に作成されたモデルと同様に動作するモデルを生成する必要があります。同様に、推論パイプラインは、実験中に得られた結果と一致する結果を生成する必要があります。実験ステージから派生した参照モデルは、整合性が保護され、署名される必要があります。

開発とテストツール

MLパイプラインを保護するには、パイプラインの開発とテスト中に活用できるさまざまなオープンソースツールを使用した、従来のソフトウェアエンジニアリングと同様のアプローチが必要です。

- MLRun: バージョン管理、再現性、監査可能性を考慮したMLパイプラインの構築とテストをサポートする堅牢なMLOpsオーケストレーションフレームワークです。モデルのトレーニングと推論ワークフローを標準化された方法でパッケージ化し、ログに記録し、検証することができます。
- OWASP Dependency-Check, SPDX, OWASP CycloneDX: これらのツールを使用すると、パイプライン開発者はパイプラインステージで使用されるライブラリとコンポーネントをスキャンして既知の脆弱性

を検出できます。これらのツールは、複数のオープンソース依存関係に依存するMLパイプラインを監査するためのSBOMも生成します。

- AFL++: パイプライン内のMLサービスに適応して、不正な入力を使用してAPIと中間ステージをストレステストし、モデルや出力を破損する可能性のあるランタイム脆弱性を明らかにするファジングツールです。
- Giskard: 堅牢性、バイアス、プライバシー漏洩テストを含む機械学習モデルの自動テストの実装を支援します。パイプラインに統合すると、継続的な再トレーニングサイクル全体にわたってモデルの動作の回帰テストが可能になります。

これらのツールを組み合わせることで、開発チームはパイプラインコードの静的テストからランタイムコンポーネントの動的検証まで、MLパイプラインのライフサイクル全体にセキュリティを組み込むことができます。しかし、注目すべきギャップの一つは、静的コードスキャナーがMLコンテキストを理解していないことです。一般的な問題（eval()の使用や危険なファイル権限設定など）は検出できるかもしれませんが、モデルトレーニングコードに入力の正規化が欠けている場合や評価スクリプトが不十分な場合は警告されません。また、これらのツールは安全でないモデルシリアル化方法などのエラーを検出できませんが、一般的なSASTルールセットでは、MLコンテキストにおけるこのような脆弱性をフラグ付けできない可能性があります。この問題に対処することで、MLOpsと従来のアプリケーションセキュリティプラクティスとの間のギャップはさらに埋められるでしょう。

継続的インテグレーション/継続的デリバリー・デプロイ、継続的トレーニング

継続的インテグレーション/継続的デリバリー・デプロイ (CI/CD) と継続的トレーニング (CT) の各ステージでは、パイプライン、モデル成果物、その他の関連資産が環境間で転送されることが多く、転送中の変更から保護する必要があります。

MLモデルがソリューションまたは製品に組み込まれている場合、ソリューションの信頼性がモデルの信頼性を本質的に検証するのであれば、モデルの信頼性を個別に検証する必要はないかもしれません。しかし、バージョン更新時など、MLモデルが独立して提供される場合は、モデルの信頼性を検証する必要があります。
CI/CD のセキュリティ保護については、次の一般的な側面を調査する必要があります。

- 転送中および保存中の成果物を暗号化し、整合性を保護する。
- ターゲット環境では、署名された成果物の真正性チェックを実行できる必要がある。
- バージョン管理を使用して、古くて潜在的に脆弱なMLモデルによる更新を防止する。

CIセキュリティ対策

CIの場合、入力は学習および推論パイプラインであり、出力は学習および推論パッケージです。主な懸念事項としては、安全でないコード、安全でないまたは古いサードパーティ依存関係（既知の攻撃に対して脆弱）、機密情報を含むビルド成果物、安全でない構成などが挙げられます。CIでは、以下の対策を検討できます。

- ビルド環境は分離され、安全に構成されている必要がある。
- サードパーティの依存関係の脆弱性をスキャンし、速やかに更新する必要がある。
- ビルド成果物を安全に保存および転送する必要がある。

CDセキュリティ対策

CDはMLパイプラインの配信やデプロイメントを高速化しますが、適切に管理されていない場合、セキュリティ上の懸念が生じる可能性があります。自動MLパイプラインのデプロイメントでは、入力にはトレーニングパッケージと推論パッケージが含まれ、出力はデプロイされたパイプラインです。自動化されたモデルデプロイメントでは、入力はトレーニング済みのMLモデル、出力は提供MLモデルです。CDパイプラインは、MLパイプラインまたは提供モデルの安全な配信とデプロイメントを保証する必要があります。以下のセキュリティ上の懸念事項に注意が必要です。

- 入力MLパイプラインは検証され、整合性が確保され、セキュリティテストの責任者によって署名されている必要があります。

- CDパイプライン、デリバリー、およびデプロイメント環境は安全に構成され、定期的に評価される必要があります。
- CDパイプラインはデリバリーおよびデプロイメントのアーティファクトを追跡し、機密性の高いものはブロックする必要があります。関連するログは機密性と整合性が保護されている必要があります。不要なアーティファクトは削除してください。

CTセキュリティ対策

CTとは、新しいデータを取り込み、機械学習モデルを定期的に再学習することを指します。これは、監視コンポーネント、フィードバックループ、そして生データを取得して必要な前処理と学習手順を実行する自動学習パイプラインによって実現されます。CTは、生データと学習パイプラインを入力として受け取り、学習済みモデルを生成します。メタデータログは入力と出力の両方のアーティファクトとして機能するため、適切に保護する必要があります。モデルの評価と検証は、モデルの品質とセキュリティの変化を分析するため、CTの重要な構成要素です。CTのセキュリティに関する考慮事項は次のとおりです。

- トレーニングパイプラインのセキュリティを維持するための定期的なセキュリティ評価とパッチ適用
- データの改ざんやインジェクション攻撃を防ぐためのデータ整合性チェック
- モデル品質の変化を評価するモデル評価は、 安全な環境で実施する必要があること

CI/CD/CT ツール

CI/CDおよび継続的トレーニング (CT) ワークフローのセキュリティを確保するには、成果物の信頼性、パイプラインの分離、依存関係の衛生、安全なメタデータ管理に関する強力な保証を適用する必要があります。OWASPおよびMLOpsコミュニティのいくつかのオープンソースツールは、配信および再トレーニングのライフサイクル全体を通じてこれらのプラクティスを実施するのに役立ちます。

- OWASP Dependency-Track SPDX および OWASP CycloneDX のコンテキストでは、これらのツールは、トレーニング環境や推論環境に組み込む前にMLの依存関係を検証するのに役立ちます。
- Argo CD は、Kubernetes 環境向けの宣言型GitOps継続的デリバリーツールです。バージョン管理された信頼できる情報源を維持し、ポリシーベースのロールアウトを適用することで、デプロイメントパイプラインのセキュリティを確保し、構成のドリフトを追跡して再現性を確保します。
- Evidently と WhyLogs は CTパイプラインで価値があり、モデルの検証と品質の監視を可能にします。これらの出力は再トレーニングトリガーにフィードされ、データの変更が再トレーニングを必要とするか、データポイズニングの懸念があるかを判断するのに役立ちます。
- Sigstore は、開発者が暗号署名を使用して ML モデルや関連メタデータなどのソフトウェア成果物に署名および検証できるようにする一連のツールとインフラストラクチャを提供します。

モデル提供

モデル提供とは、訓練済みのMLモデルが本番環境で推論を行うプロセスのことです。モデルはクライアントのリクエストに対応する前に、MLOpsシステムにデプロイされている必要があります。この段階で、モデルと推論サービスの両方に新たなセキュリティ課題が生じます。それは例えば、悪用、リバースエンジニアリング、敵対的操作からの保護などです。安全かつ信頼性の高い運用を確保するため、提供中のMLモデルと推論パイプラインは適切に保護されなければなりません。なお、MLモデルが外部クライアントに提供されるため、提供組織の管理外で実行されるケースもあることに注意する必要があります。

モデル提供のセキュリティ対策

モデル提供プロセスの安全な実装、設定、テストには以下が含まれます。

- 適切に設定されたコンテナとオーケストレーション環境
- 推論サービスに対する堅牢なアクセスコントロール機構
- データの暗号化とプライバシー保護技術、または非識別化技術（匿名化や仮名化）

- リバースエンジニアリング、および知的財産（IP）または機微情報の漏洩を防止するためのモデル暗号化、ウォーターマーキング、準同型暗号化
- モデルインバージョン攻撃とメンバーシップ推論攻撃からの保護。モデルの利用状況を監視する必要がある。単一ソースからのデータストリーム利用やバッチ処理ではなくクライアントからのリクエストを直接処理する場合には、クライアントからのリクエスト数を制限することが重要
- 回避攻撃 / 敵対攻撃からの保護。モデルは悪意のある攻撃に対して堅牢である必要がある。入力の検証とサニタイズアルゴリズムを使用し、クエリ数を制限し、敵対攻撃に対する適切な堅牢化技術を利用すること。

加えて定期的なセキュリティ監査、最小権限の原則、セキュリティ対策のスケーラビリティ確保は、提供プロセスの堅牢さを顕著に改善することができます。

モデル提供ツール

MLOpsとMLSecOpsから来たいくつかのオープンソースツールとライブラリは、安全なMLモデル提供の助けになります。

- Garak: プロンプトインジェクション、入力操作、出力による情報漏洩といったモデル脆弱性を評価するための「レッドチーム」向けツール。これらはLLMの提供に強い関連があり、デプロイ前・デプロイ後のセキュリティテストとして実施することができる。
- Seldon Core: Kubernetes インフラで機械学習モデルをデプロイおよびオーケストレートするためのツール。TensorFlow、PyTorch、scikit-learn といった様々なフレームワークで作成されたモデルに対応しており、それらモデルをスケーラブルなコンテナサービスとしてラッピングすることで、開発から本番環境への移行を単純化できる。
- TextAttack と Foolbox: モデルが回避攻撃に対して脆弱かどうかを評価することに利用できる。
これらのツールはしばしばデプロイ前のテストとして認識されていますが、推論サービス実行時の防御メカニズムの設定や検証にも有用です。

セキュリティモニタリング

AI/MLをセキュアに保つことは、開発とデプロイを超えた継続的なプロセスです。
制御されたセキュアな環境を構築する手順は、標準化する必要があります。

セキュリティ対策のセキュリティモニタリング

実用的なダッシュボードや重要な指標の表示を実装して活動のモニタリングを導入すべきです。重要な指標には例えば以下があります。

- モデルのパフォーマンス指標
- 利用統計
- 推論リクエストに関連したメタデータ
- 入出力のエラー記録

イベントは重要度でソートされていると、調査の優先度が特定できます。エラーの種類が特定できていると、イベントのカテゴリ分けと整理に役立ちます。これはイベントを個々に単独で見ても明らかにならないパターンや傾向を特定するのに役立ちます。

自動的な検知と対応の仕組みは重要です。ドリフトをモニタすることで検出の精度を維持することができ、敵対攻撃など悪意のある入力の検出にも役立ちます。AI/MLに特化したモニタリングシステムは、全体的なダッシュボードと統合されているべきです。単純なアラートは通常のパフォーマンス異常に対しては有効ですが、予期せぬ入力やクラッシュに対しては高度なセキュリティフィルターとカスタマイズされた対応が必要になります。

具体的な対応方法に関わらず、AI/MLシステムの問題へは迅速に対処し、ベストプラクティスは継続的にアップデートしなければなりません。これは担当者への継続的なトレーニングの提供も含みます。

セキュリティモニタリングツール

AI/MLシステムのセキュリティモニタリングには、一般的な監視フレームワークとML特化型の仕組みとの統合が必要です。これらのツールはパフォーマンス低下、敵対的行動、異常な利用パターンをリアルタイムに検出する必要があります。MLSecOpsと監視の領域で開発されたオープンソースツールが、堅牢なモニタリングの仕組みを作るのに役立ちます。

- Evidently: モデル品質、データドリフト、ターゲットドリフト、特徴量の重要度をリアルタイムにモニタする機能を提供する。ダッシュボードへの統合とアラートの仕組みを備えており、これは運用監視とセキュリティモニタリングの双方のユースケースで有用。
- WhyLogs: MLアプリケーションのための軽量かつスケーラブルなロギングシステム。データセットとモデル出力の統計的プロファイルを記録し、アノマリ検出と潜在的に有害な入力へのフラグ付けを本番環境のパイプラインで行うことができる。

チャレンジと推奨項目：チャレンジ

MLSecOpsの導入はAIとMLのセキュアなライフサイクルに重要な進化をもたらします。しかしながら、この取組に乗り出す組織はしばしば、AI/ML技術に内在する複雑さ、その動的な性質、および不透明さに起因する本質的な課題に直面します。これらの問題に対処するためには、専門的な能力、戦略的アプローチ、それにAI/ML向けに特化したセキュリティのプラクティスを思慮深く統合していくことが必要になります。さらにこの領域の多くの技術は発展途上のいまだ未成熟な状態にあり、包括的なセキュリティ対策が不足しているため、より厳格な評価が必要になります。これらの課題を理解し、的を絞って推奨事項を実装していくことは、堅牢でセキュアなAI/MLシステムを構築し利用することを望む組織にとっ

チャレンジ1：DevSecOpsとMLSecOpsとで異なる脅威

DevSecOpsはインジェクション脆弱性、不適切な設定、ソフトウェアの脆弱性といった良く知られたリスクに対処する一方で、MLSecOpsはMLの考え方に特化した脅威に対応しなくてはなりません。それらには、訓練時に悪意のある入力でモデルの振る舞いを変化させるデータポイズニング、微小な擾乱に対するモデルの感度を悪用する敵対的入力、モデルそれ自身が高い価値の資産である場合にはモデルの盗用や改ざんが含まれます。モデル反転攻撃やメンバーシップ推論攻撃といったプライバシーデータを標的とした攻撃は、モデルの学習パラメータを利用して機微な学習データを明らかにします。さらに、サービス拒否攻撃から許可されていないアップデートによる完全性の喪失まで、デプロイされたモデルはAPIを通じて悪用される危険にさらされています。これらのアタックベクタは従来のソフトウェアリスクと明確に対応付けられていないため、これらからの防御はMLライフサイクル向けに明示的に設計されたセキュリティ対策が必要となります。多くの例では確実な防御策が存在しないため確率的な防御に留まることになり、何回も試行することができる攻撃者に対する防御を難しくしています。

チャレンジ2：継続的トレーニングの複雑さ

機械学習のワークフローは反復的であり、データドリブンであり、デプロイ後も動的な運用が残るといった点が従来のソフトウェアシステムと異なります。それぞれの再訓練サイクルによりモデルの挙動が変化することと、システムが新たなセキュリティリスクにさらされることは、継続的な訓練により特異的にもたらされる複雑さです。MLのパイプラインは、データ取り込み、特徴量エンジニアリング、トレーニングオーケストレーション、デプロイメントといった相互に接続された複数のコンポーネントにまたがっており、これらの端から端まで全てを確実に保証することは困難です。この複雑さに頻繁なアップデートと複数のチームによる協働が加わり、開発とメンテナンスの境界をあいまいにしているため、初期段階にシフトしたセキュアで、自動化され、再現可能なワークフローが必要となります。MLシステムの完全性と信頼性を維持するためには、組織は全ての反復的訓練サイクルを脆弱性が導入される可能性のあるポイントとして扱い、ライフサイクル全体にわたるセキュリティ対策を組み込む必要があります。

チャレンジ3：ML モデルの不透明性と解釈可能性を管理する

論理が透明で追跡可能な従来のソフトウェアシステムと異なり、ほとんどの機械学習モデル、特にそれがディープラーニングに基づいている場合は、人間が全てを解釈することはできない不透明なシステムとして動作します。これらの複雑な内部表現は、ある判断がどのようになされたのかを理解、調査、説明することを困難にしています。このような透明性の欠如は、モデルに埋め込まれた悪意ある動作が検出されないかもしれないという、セキュリティに対する重大な課題をもたらします。モデルの推論プロセスを明確に把握できないため、セキュリティチームは敵対的テストや説明可能性ツールなどによる間接的な検証手法に頼らざるを得ません。しかし現状の説明可能性技術では、隠されたバックドアや微妙な操作を明らかにする能力には限界があります。高い安全性を求められる用途や規制のある環境では、この不透明性が認証とリスク評価を更に複雑化します。その結果、解釈可能性の改善は透明性だけの問題ではなく、信頼の確立、有効な脅

威の検知、セキュアなデプロイをサポートする上でMLシステムにとって本質的な要求となります。

安全性が重要な環境でこのリスクを低減する一つの方法は、信頼できる孤立環境（エンクレーブ）の中で、改ざん防止対策が施された環境で評価が完了したことを証明する、モデル、データセット、評価スコアの3要素の組を保証する証明書（アテステーション）を生成することです。コントロールプレーンのツールがこのデータを解釈して、あるデータセットを使用した場合のスコアが指定された閾値より高いモデルのみを本番環境へ投入することを許すことができます（データセットは入力シナリオの大半をカバーしていると想定しています）。この手法以外では、人間が嘘をついたり、特定のモデルがテストデータセットで実際より良い性能を出せると誤って主張できます（または人間が直接テストデータセットを扱う場合にそれらが外部に漏洩する可能性もあります）。評価がTrusted Execution Environment（TEE）で行われるならば、そのスコアは改ざん防止されたアテステーションであり、TEEの外側からデータセットにアクセスすることはできません。この手法はモデルを透明化するわけではありませんが、リスクを減らすことができます。

チャレンジ4：頻繁な再訓練によるセキュリティリスク

静的なソフトウェアリリースとは対照的に、機械学習システムは変化するデータや運用環境に適応するため頻繁に再訓練が行われます。これは対応力を高める一方で、各反復ごとにセキュリティリスクが生じます。再訓練サイクルでは改ざんされたデータが入力される可能性があり、モデルの完全性を徐々に低下させるスロー・ポイズニング攻撃の可能性が高まります。頻繁なペースのアップデートを行うためには人間によるセキュリティレビューでは限界があり、自動化された検証、モデルへのバージョン付与、ロールバックの機構が必要となります。系統の追跡と堅牢な評価手順がなければ、再訓練によって脆弱性が本番環境に拡散する可能性があります。従い頻繁なアップデートは継続的なセキュリティへの取組みを必要としており、全ての反復において、モデルの完全性・性能・回復力が基準を満たすことを保証するための訓練パイプラインに統合された自動チェック機構が必要となります。

チャレンジ5：モデルの工程証跡（Provenance）と再現性（Reproducibility）

機械学習システムはその複雑なパイプラインにも関わらず、頻繁な再訓練と大規模なデータセットのアップデートにより高速に進歩します。その結果、モデルの履歴を全て詳細に追跡することは困難です。モデルが頻繁に再訓練またはアップデートされる場合、パラメータ、アーキテクチャ、それぞれのバージョンで使用された学習データを含む変化を時系列で追跡できる実行的なバージョン管理を実装することは困難です。その後の訓練で使用されるデータのいくつかは以前にモデルが実行した結果に基づいており、修正が困難なドリフトを引き起こします。

再現性もまた課題の一つです。モデルの挙動が再現できることは、デバッグや性能の改善、セキュリティ対策といった観点で重要です。しかしモデルと開発環境が動的であるという特性により、再現性の維持は容易ではありません。実際、パイプラインは全く同じモデルを生成することはほとんどなく、同じモデルで繰り返し推論を行った場合でもいくらか異なった結果が出力されます。基準点となる設定、訓練データ、ビルド環境などのスナップショットをどのように取得するかなど、ポリシーの構築と実践は非常に重要です。MLプロジェクトは多くのライブラリやフレームワークに依存しており、これらも時々刻々と変化していきます。「だいたい再現している」ことを検証できる柔軟なテストがしばしば必要とされます。これらの依存性を整理し、確実に同じバージョンが使用されることを保証することが、再現性には不可欠です。

チャレンジ6：リスクアセスメント実行の困難さ

AIのリスク管理は、測定の難しさ、評価指標のトレードオフ、優先度付け、組織としての課題、リスク許容度の定義といった様々な理由により大変困難です。AIシステムに関連するセキュリティ、安全性、倫理に渡る測定指標の定義は、その振る舞い、影響度、運用の文脈が全て複雑で予測できないかもしれないことを理解して行う必要があります。矛盾する目的により複雑さが増大する指標間のトレードオフも存在します。正確さと公平さ、セキュリティと説明可能性性、または透明性と効率などの最適化により指標間のバランスをとるには、慎重な検討と優先度付けが求められます。

推奨事項

これまでに述べた課題に対処するためには、組織は成熟した Machine Learning Security Openrations（MLSecOps）への実践的なステップを踏む必要があります。以下、具体的な推奨事項を記します。

- **構造化されたセキュリティ成熟度モデルの採用**
  OWASP SAMM、 NIST AI RMFといった確立されたフレームワークを用いて、現在のセキュリティに対する状況を最低基準として設定し、MLライフサイクルに存在するギャップを特定すること。安全なデータ収集、モデル検証、モデル廃棄のポリシーを確立することにより、AIプロジェクトに「セキュア・バイ・デザイン」を組み込むこと。
- **MLパイプライン全体にわたるセキュリティ対策の自動化**
  データ検証の有効性、モデルの堅牢性、ドリフトを自動的にチェックする機構を CI/CD/CTパイプラインに統合すること。モデル等成果物のスキャン、署名の検証、 MLに特化した性能指標を監視するツールを使用し、手動での監視を減らし、問題を早期に検出すること。
- **再現性とバージョン整合性の強制**
  データセット、モデル、学習環境に対して包括的なバージョン管理を維持すること。 モデルのレジストリ、 署名済みのリファレンスモデル、 再現可能なコンテナ化されたワークフローによって、 再学習からデプロイにまたがるトレーサビリティと整合性を確かなものにすること。
- **既存のセキュリティ体制**
  開発時と運用時に AI/MLセキュリティ対策について評価を行い、 他のセキュリティプロトコルによってライフサイクル全体を改善すること。
- **チーム間の協働**
  例えば開発プロセスを妨げることなくセキュリティ担当者をMLOpsチームに入れるなど、 強固な組織横断的活動を推進すること。 AI/MLに対する脅威と対策の独自性と特殊性について、 チームメンバーを教育すること。
- **IT インフラの評価**
  ITインフラについて、従来型の固定的な形態を超えた柔軟性を求める MLSecOpsに適合しているかどうかレビューを行うこと。MLSecOpsは単独で存在するソリューションではなく、むしろ既存のサイバーセキュリティ対策機構の上に乗る付加的なレイヤーであり、その機構は堅牢かつ成熟したものである必要がある点に注意すること。情報セキュリティマネジメントシステム（ISMS）を確立することは、MLSecOpsを促進することでもある。
- **明確な目標の設定**
  これには AI/MLに対する脅威の自動的な検知、回避攻撃に対する防御能力の向上開発、またはセキュリティインシデントの予見が含まれる。具体的な目標によりMLSecOpsの戦略が推進され、効果の測定に寄与する。

MLSecOpsの実装ステップ

- **AI/MLセキュリティを強化する**
  MLOpsチームを支援するための適切なスキルを持つセキュリティ担当者をアサインし、AI/ML開発ライフサイクル全体にわたりセキュリティプラクティスの遵守を確実にすること。セキュリティ担当者はセキュリティ意識の向上と、各チームメンバーがAI/MLをセキュアに保つための役割を理解することも支援する。
- **セキュリティツールの統合**
  開発プロセスで使われるセキュリティツールを自動化すること。データ分析、静的コード解析、動的テスト、依存性チェックのためのツールをCI/CDパイプラインに統合する。
- **継続的モニタと改善**
  MLSecOpsには日常的なレビューと改善が不可欠である。現在は新たな脅威が頻繁に現れている状況であり、MLOpsのプロセスを日常的にモニターし、ログを記録し、監査すること。セキュリティインシデント対応はそれらを理解することとMLSecOps全体を改善するために有用である。

予想される障害とそれらの克服

- **スキルを持つ人材の不足**
  MLSecOpsは、MLとサイバーセキュリティ両面への深い理解を要求する。どの脅威が従来の対策により緩和でき、どの脅威はML特有の対策が必要か区別する必要がある。そのような専門性を持つ人材が組織に存在

推奨事項

しない場合は、スペシャリストを雇用するか、能力開発への投資を検討する必要がある。

- **データプライバシーへの懸念**
  データの匿名化、差分プライバシーの利用、最新の規制への追従により、プライバシー問題への懸念を軽減すること。
- **進化し続ける脅威**
  AI/MLへの脅威は常に変化し続けており、最近開発されたMLモデルであっても脅威の影響を受ける可能性がある。モデル開発とデプロイのプロセスは柔軟で、日常的にアップデートされる必要がある。
- **ツール導入への課題**
  適切なセキュリティツールの統合は（開発者やプロセスにとって）負担になり得る。初期段階では、統合しやすい最も重要なツールの特定に注力すること。プロセスが安定したら、その後より高度なツールを使用するようにする。
- **プロセス負荷による変化への抵抗**
  セキュリティ対策を導入した時点では開発が鈍化する可能性があるため、抵抗を招く。ML開発の初期段階でのセキュリティ対策が後の懸念を最小化できることを示すこと。AI/MLシステムはしばしば 不透明でデータ集約的であることを明確にすること。AI/MLの開発ライフサイクルを保護するための包括的な取り組みがなければ、セキュリティとプライバシーに関する潜在的問題があとあと噴出し、トータルでは高いコストがかかる恐れがある。

結論

機械学習が重要なシステムとビジネスワークフローに深く統合されるにつれ、その運用基盤であるMLOpsは新たなリスクへ対処するため積極的なセキュリティとの統合が求められています。このペーパーはMLSecOpsを、計画からデプロイ、モニター、継続的な学習といったデータエンジニアリングの全てのステージにセキュリティを埋め込むことで、DevSecOpsのプラクティスの上に築かれたMLOpsライフサイクルを自然に拡張するものとして紹介しました。

OWASPとOpenSSF の提供する既存のツールが確固たる基盤を提供している一方で、MLSecOpsには現行技術への適応とML固有の要求のための新たな技術開発の両方が必要です。

また、ここでは新たなペルソナ群を導入しました。これらは現状の ML ライフサイクルの多様なセキュリティ担当者を代表するものです。これらのペルソナは、AI/ML全体の中で役割がどう変化していくか、新しいツールとコミュニティ主導の取り組みが実際的かつ利用可能な方法でそれぞれのペルソナをどう支援して現在のワークフローに対応していけるかといったことを反映しています。

MLSecOpsに飛び込む前に、組織はそれ自身のセキュリティに対する姿勢を評価し、組織横断的な協働を促進し、ITインフラを評価し、明確な目標を設定する必要があります。MLSecOpsは既存のセキュリティ対策に取って代わるものではなく、AI/ML開発の全ての段階でそれらを強化し補完するものと捉える必要があります。

MLSecOpsは AI/MLの開発プロセスにセキュリティを統合するための継続的な改善アプローチです。セキュリティが協働による取組であることが理解できれば、性能とスケーラビリティが高いだけではなく、設計として高い安全性と信頼性を備えたAIシステムを構築することができるでしょう。

参考文献

- CNCF end user technology radar provides insights into DevSecOps | CNCF
- DOD Enterprise DevSecOps Reference Design: CNCF Kubernetes
- DevSecOps Days Washington DC 2021
- About – Open Source Security Foundation
- GitHub - ossf/tac: Technical Advisory Council
- GitHub - ossf/ai-ml-security: Working Group on Artificial Intelligence and Machine Learning (AI/ML) Security
- Secure Software Development Education 2024 Survey
- ML4Devs: MLOps Machine Learning LIfe Cycle
- MLSecOps: Protecting AI/ML Lifecycle in telecom - Ericsson
- toolbelt/personas/README.md at main · ossf/toolbelt · GitHub
- Sachiko the Solutions Architect
- Alison the AIML Engineer
- Timmy the Test Engineer
- Guinevere the Security Governance Lead
- Pang the Product Security Engineer
- Chinmay the Cloud Platform Admin
- Ophelia the IT Infrastructure Engineer
- Daniel the Data Scientist
- Dibby the Data Engineer
- Grear the Data Governance Analyst
- Machine Learning Operations (MLOps): Overview, Definition, and Architecture
- OWASP Machine Learning Security Top Ten | OWASP Foundation
- Sigstore
- sigstore/model-transparency: Supply chain security for ML
- ossf/scorecard: OpenSSF Scorecard - Security health metrics for Open Source
- ossf/allstar: GitHub App to set and enforce security policies
- SLSA • Supply-chain Levels for Software Artifacts
- guacsec/guac: GUAC aggregates software security metadata into a high fidelity graph database
- OWASP/www-project-threat-dragon: OWASP Foundation Threat Dragon Project Web Repository
- CycloneDX/specification: OWASP CycloneDX is a full-stack Bill of Materials (BOM) standard that provides advanced supply chain capabilities for cyber risk reduction. SBOM, SaaSBOM, HBOM, AI/ML-BOM, CBOM, OBOM, MBOM, VDR, and VEX
- OWASP SAMM | OWASP Foundation
- OWASP Dependency-Check | OWASP Foundation
- Threat Modeling - OWASP Cheat Sheet Series
- OWASP Dependency-Track | OWASP Foundation
- RiccardoBiosas/awesome-MLSecOps: A curated list of MLSecOps tools, articles and other resources on security applied to Machine Learning and MLOps systems

参考文献

- SPDX – Linux Foundation Projects Site
- anchore/syft: CLI tool and library for generating a Software Bill of Materials from container images and filesystems
- awslabs/deequ: Deequ is a library built on top of Apache Spark for defining “unit tests for data”, which measure data quality in large datasets
- Great Expectations: have confidence in your data, no matter what • Great Expectations
- Data Version Control · DVC
- ARX – Data Anonymization Tool – A comprehensive software for privacy-preserving microdata publishing
- ydataai/ydata-profiling: 1 Line of code data quality profiling & exploratory data analysis for Pandas and Spark DataFrames
- MLflow
- Trusted-AI/adversarial-robustness-toolbox: Adversarial Robustness Toolbox (ART) - Python Library for Machine Learning Security - Evasion, Poisoning, Extraction, Inference - Red and Blue Teams
- protectai/nbdefense: Secure Jupyter Notebooks and Experimentation Environment
- Open Source MLOps Orchestration | MLRun
- AFLplusplus/AFLplusplus: The fuzzer afl++ is afl with community patches, qemu 5.1 upgrade, collision-free coverage, enhanced laf-intel & redqueen, AFLfast++ power schedules, MOpt mutators, unicorn_mode, and a lot more!
- protectai/modelscan: Protection against Model Serialization Attacks
- anchore/grype: A vulnerability scanner for container images and filesystems
- Jenkins
- Argo CD - Declarative GitOps CD for Kubernetes
- whylogs: the open standard for data logging | WhyLabs
- tensorflow/privacy: Library for training machine learning models with privacy for training data
- evidentlyai/evidently: Evidently is an open source ML and LLM observability framework. Evaluate, test, and monitor any AI-powered system or data pipeline. From tabular data to Gen AI. 100+ metrics
- NVIDIA/garak: the LLM vulnerability scanner
- SeldonIO/seldon-core: An MLOps framework to package, deploy, monitor and manage thousands of production machine learning models
- protectai/llm-guard: The Security Toolkit for LLM Interactions
- QData/TextAttack: TextAttack is a Python framework for adversarial attacks, data augmentation, and model training in NLP https://textattack.readthedocs.io/en/master/
- bethgelab/foolbox: A Python toolbox to create adversarial examples that fool neural networks in PyTorch, TensorFlow, and JAX
- Modeling Threats to AI-ML Systems Using STRIDE
- AI Risk Assessment for ML Engineers | Microsoft Learn
- MITRE ATLAS ™
- AI Risk Management Framework | NIST
- ISO/IEC 23894:2023 - AI — Guidance on risk management

著者

このガイドの基礎となった原論文を寄贈してくださったエリクソン社に深く感謝します。このバージョンは、原著者のAndrey Shorov氏および Elif Ustundag Soykan氏による顕著かつ新たな貢献、ならびにデル・テクノロジーズのセキュリティ研究者であるSarah Evans氏およびBahaulddin Shammary氏による多大な貢献により、さらに充実したものとなりました。

またレビュワーとして、Eddie Knight氏、Christopher Robinson氏、David A Wheeler氏、Mihai Maruseac氏、Rob Moffat氏、およびその他の OpenSSFと AI/ML WGメンバーから、論文の品質と明快さを向上する貴重なフィードバックを提供していただいたことにも感謝の意を表します。

この資料は、OpenSSF 発行の whitepaper「Visualizing Secure MLOps: A Practical Guide for Building Robus AI/ML Pipeline Security」を、OpenSSF Japan Chapter の有志メンバーで日本語に翻訳したものです。

日本語版翻訳協力：
OpenSSF Japan Chapter 翻訳チーム
- 清海 佑太（本田技研工業）
- 下沢 拓（日立製作所）
- 余保 束（ルネサスエレクトロニクス）
- 池田 宗広（サイバートラスト）

Thank you! Join us..

openssf.org/getinvolved