Ericsson White Paper
GFTL-24:000268 Uen
February 2024

ERICSSON

MLSecOps: Protecting the AI/ML Lifecycle in telecom

Content

Introduction

Modern telecom systems meet the needs and improve the quality of life of billions of people. They increasingly benefit from artificial intelligence/machine learning (AI/ML) to manage network complexity, analyze vast data, enhance security, and boost efficiency, and performance. AI/ML systems must be secure by design, default, and deployment, safeguarding sensitive data by integrating security practices into all stages of the life cycle. To achieve these goals, MLSecOps extends MLOps for building, deploying, operationalizing, and observing ML-based systems.

The use of AI/ML in telecom products is constantly increasing. It improves efficiency and enhances functionality, while simultaneously introducing new attack surfaces. It is critical for Ericsson and our customers to recognize the security and privacy challenges in AI/ML for telecom and take steps to implement automated security measures into the ML models as early stage.

In this white paper, we discuss security threats and risks, followed by mitigations through well-defined controls, and lastly a suggestion for safeguarding MLOps.

Security challenges in the AI/ML development life cycle

MLOps is about automated, efficient, and reliable ML model and pipeline development, deployment, and execution, but its components and processes might expose several security issues. These include data poisoning, vulnerabilities in open-source ML libraries, and attacks to trained ML models. While some of the security issues are common in every development pipeline, some are very specific to AI/ML. Let us have a closer look at the need for MLSecOps.

- AI/ML Supply chain security: the supply chain includes external sources of data and ML models, software and hardware components, storage and management systems, and communication networks. These components can be open-source or commercial. Known vulnerabilities in them are easy targets for attackers, exposing sensitive data or operations.
- Model training and inference security: AI/ML training involves large amounts of data and complex data handling processes. Lack of proper access control, confidentiality, and integrity mechanisms risks unauthorized data access, disclosure, and tampering.

In addition, an attacker with data access rights might poison the training data to craft targeted ML model responses. Access to interim parameters or the inference environment enable model poisoning and adversarial attacks. This can lead evading certain responses, reduced accuracy, or extraction of valuable data such as the ML model parameters or training data set.

- Data and ML model provenance: Data and ML model provenance means to track the handling of data and ML models in the pipeline. Record keeping should be secure, integrity protected, and traceable. Access and version control of data, ML model, and pipeline parameters, logging, and monitoring are all crucial controls.
- Regulatory requirements: Governments around the world are taking note of these facts and developing standards and regulations for AI systems, especially for critical infrastructure. The European Union's Artificial Intelligence Act, following the EU ethics guidelines for trustworthy AI, is one such effort. It defines a useful framework, breaking down trustworthiness into specific areas, including security and privacy protection [1].

Ensuring effective security and privacy protection within MLOps processes is challenging. Best practices for automating security are needed throughout the entire AI/ML life cycle, from design to testing, deployment, and ongoing monitoring. The solution is Machine Learning Security Operations or MLSecOps

Securing AI/ML systems with MLSecOps

The increasing use of ML models has emphasized the need for enhanced methods to develop, deploy, and manage them, leading to the growing popularity of the MLOps concept. Inspired by DevOps, MLOps aims to align the practices of AI/ML development and operations. By emphasizing automation and monitoring throughout the entire life cycle, MLOps encompasses development, integration, testing, release, deployment, and infrastructure management. [2].

The need for ensuring the security leads further development of MLOps into MLSecOps. MLSecOps places a strong emphasis on integrating security practices within the ML development life cycle. It establishes security as a shared responsibility among ML developers, security practitioners, and operations teams. Embracing this methodology enables early identification and mitigation of security risks, facilitating the development of secure and trustworthy ML models.

Understanding the architecture of MLOps is necessary for ensuring security. While various MLOps frameworks exist, this paper uses a generalized MLOps architecture to represent processes and security procedures. The architecture, illustrated in Figure 1, incorporates an automated continuous integration/continuous delivery (CI/CD) system. It supports the efficient exploration of new techniques in ML model crafting and pipeline preparation and simplifies the processes of building, testing, and deploying new ML components.

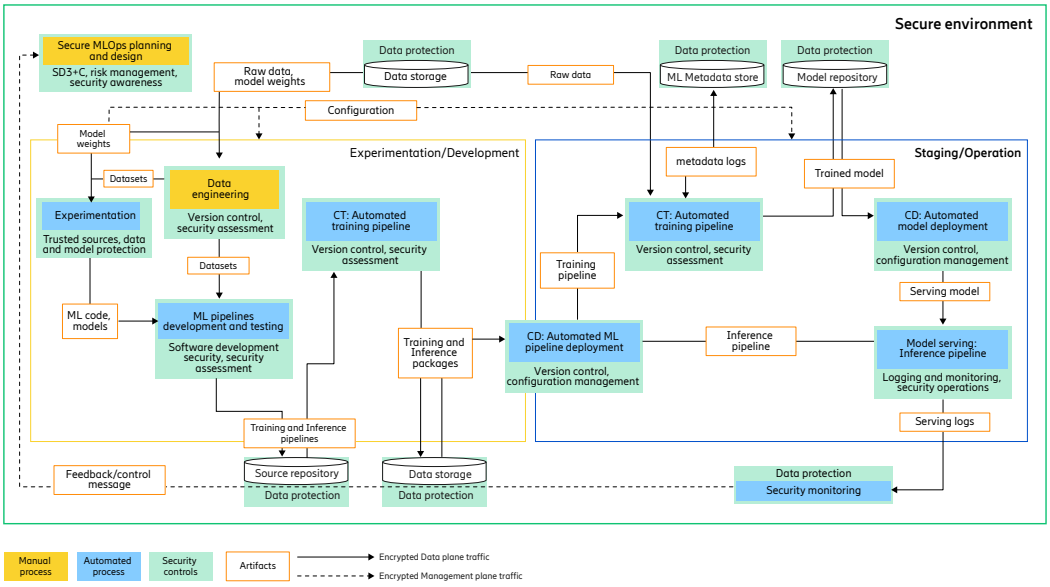


Figure 1: Generalized MLSecOps architecture with security controls

Figure 1 provides an integrated view of the MLSecOps framework, highlighting security controls and illustrating the flow of artifacts through the pipelines. Artifacts—including datasets, ML code, models, and deployment packages—must be protected.

A secure development environment is crucial for MLSecOps. It is necessary to evaluate potential security risks and implement corresponding mitigations regarding tools and development environments in line with the information security management system (ISMS) [3].

Particularly, the following risks related to development tools should be considered:

- Secure tools - Use trusted and approved tools for development
- Patch management - Tools should be updated frequently
- Access to tools should be limited based on role and responsibilities

In certain cases, the organization might not have complete control over the production environment such as when the product is deployed onto an HCP, or into a customer's private cloud. Here, security obligation responsibility should be shared between the company and the environment provider. When using an HCP, evaluating the HCP's security controls should be included as part of the regular risk assessment process.

Below is the description of the specific processes depicted in Figure 1.

Secure MLOps design

Secure MLOps design involves integrating security practices into the ML life cycle, including planning, development, deployment, and operations.

During secure MLOps design:

- Identify the key MLOps principles, components, and roles.
- Gain a comprehensive understanding of the MLOps architecture.
- Define the workflows - the sequence of tasks executed throughout the MLOps process.

Starting by establishing a security baseline, provides a foundation for the AI/ML development life cycle. The baseline takes into account threats to AI/ML systems and includes minimum security controls, best practices, and guidelines. It is the starting point for protecting the AI/ML system and data.

Once the baseline is in place, a security risk assessment (RA) helps identify and prioritize AI/ML risks, allowing for effective risk mitigation strategies through MLOps processes.

Useful tools and references to help with risk assessment include:

- The STRIDE framework, as illustrated in "Modeling Threats to AI-ML Systems Using STRIDE," addresses vulnerabilities and specifies tools [4].
- "Microsoft AI Security Risk Assessment" offers an exhaustive analysis [5].
- MITRE's ATLAS provides data on tactics, techniques, and case studies [6].
- OWASP ML Security Top Ten [7]

Consider a scenario in which an anomaly detection model detects abnormal behavior and then automates solutions. Identifying risks such as data poisoning, model evasion, and Denial-of-Service(DoS) during software design enables the implementation of essential safeguards. To ensure seamless integration, these security components needs to be aligned with MLOps processes. This way, they undergo development and testing concurrently with ML artifacts such as ML models and pipelines.

The secure MLOps design process also requires a secure configuration of the entire MLOps architecture, including specific processes and security controls.

Data Engineering

Data engineering takes raw data as input and produces datasets needed by subsequent processes. There should be implemented security policies and controls for collecting and storing data.

Collected data might include sensitive personal details. The appropriate legal and contractual authority must be in place for processing sensitive data.

- Before acquiring data from untrusted sources, it must be approved by the appropriate line of escalation.
- Before incorporating data into an MLOps environment, it must be validated and authorized by the appropriate line of escalation.

Unwanted or malicious information in the data could impact performance or introduce malicious behavior, as can tampering with stored data. To avoid this and maintain privacy, data must be properly secured at rest.

The following security measures should be implemented:

- Data storage should employ security controls appropriate for the sensitivity of the data.
- If the sensitivity demands encryption, use strong encryption algorithms.
- Stored data should be integrity-protected.
- Data access should be monitored and logged using a formal access control process.
- Implement versioning and formal change control processes.
- Conduct regular vulnerability scans to identify potential poisoning threats using updated tools.
- Perform regular data backups and recovery tests.
- Implement data retention policies on storage lifetime and secure disposal.

Data quality determines model quality, so data quality should be ensured throughout the development life cycle. Similarly, data protection measures must be implemented during development and production. During development, acquired training data should be analyzed for malicious entries. In certain cases such as anomaly detection, it can be challenging as anomalies can be real-world events or could be an outcome of a malicious attack. Therefore, to analyze suspicious data effectively, the involvement of subject-matter experts is crucial.

Experimentation

While conducting experiments, a data scientist performs ML model engineering, selects features, and algorithms or develops new ones, trains the model, and tunes hyperparameters. The inputs consist of model weights and datasets, and the outputs include ML code and ML models.

Incorrect or insecure code can result in availability, integrity, or confidentiality risks. Security practitioners should consider the following requirements:

- Conduct design and research in a secure environment
- Review and approve model selection at an early stage in development before implementing it in production. Track the model throughout
- Document experiments and associated metadata. Implement full traceability of experiments and model training

An ML model trained under ideal conditions might prove fragile when deployed in potentially adversarial environments. Model selection criteria, for example, metrics and testing sets should emulate different kinds of drift and anticipated adversarial conditions. The following measures should be taken:

- Ensure that training, validation, and testing sets adhere to natural temporal dependencies
- Enhance model robustness by augmenting datasets with common corruptions that could reasonably be encountered
- If adversarial examples are a concern, consider adversarial training
- If training uses distributed data, consider federated learning to mitigate privacy concerns

Version control and integrity protection should be used to detect unauthorized changes in the ML model, which helps detect poisoning. Signed and integrity-protected versions enable reversion to a known good state in the event of corruption

When transferring ML models, it is important to safeguard them from unauthorized alterations. A standard approach is to apply a cryptographic hash function over the model. Hashes should be encrypted or transmitted on alternate channels.

Trained models are intellectual property that should be protected, according to the security risk assessment and the relative value of the model. Training scripts and feature engineering codes could have even higher intellectual property value and merit protection. Model, training, feature calculation can be protected by confidential computing, encryption, and obfuscation.

ML model validation procedures should include comprehensive security testing. Regular testing can detect compromised ML models through simple checks, while advanced testing uses a broader range of attacks to identify vulnerabilities. Additionally, custom and threat-based scenarios should be developed for penetration testing purposes.

ML pipeline development and testing

Automated training and inference pipelines are used for continuous training and model serving. The process takes datasets, ML codes, and models from the experimentation phase as inputs and outputs in training and inference pipelines. Creating ML pipelines should follow a software development life cycle (SDLC) like any other software development process.

Pipeline security includes version control, integrity, and confidentiality protection. Pipeline protection is like ML model protection. Both require attention to confidentiality, integrity, access controls, and full life cycle compliance with established policies and regulations.

The security considerations for ML code and parameters used in ML pipelines should be defined and agreed upon. Security measures should be implemented during pipeline development, security testing practices should be aligned for application in pipeline testing.

Consider the following security practices for pipeline development:

- The Software Assurance Maturity Model (SAMM) by OWASP [8] provides a framework for incorporating security activities into software development and maintenance.
- Code review or peer reviews, including those conducted by a software engineer..
- Static Application Security Testing (SAST) examines software security without execution by analyzing either the source or the compiled binary.
- Dynamic Application Security Testing (DAST) assesses the software security in a runtime environment and is used to test pipelines without access to the source.
- Fuzz testing provides invalid input (randomly generated or specifically crafted), and monitors a pipeline for crashes, buffer overflows, or other unexpected results.
- Interface (API) testing involves multiple teams working on different parts of an ML pipeline.
- Misuse or abuse case testing simulates user attempts at manipulating inputs to produce a corrupted ML model.

During continuous training, an automated training pipeline must produce models that behave similarly to those created during experimentation, provided the same inputs are used. Similarly, the inference pipeline should produce results that align with those achieved during experimentation. A reference model, derived from the experimentation phase, should be integrity-protected and signed.

Continuous Integration/Continuous Delivery and Deployment, Continuous Training

In Continuous Integration/Continuous Delivery (CI/CD), and Continuous Training (CT) phases pipelines, model artifacts, and other relevant assets are often transmitted between environments, and should be protected from modification in transit.

When an ML model is embedded in a solution or product, a separate verification for the model's authenticity might not be necessary if the authenticity of the solution inherently validates the model. However, if the ML model is supplied independently, such as during a version update, the authenticity of the model must be verified. The following general aspects of securing CI/CD should be investigated:

- Encrypt and integrity protect artifacts in transit and at rest.
- The target environment should be able to perform authenticity checks of signed artifacts.
- Use version control to prevent updating with an old, potentially vulnerable ML model.

For CI, the inputs are training and inference pipelines, while the outputs are training and inference packages. Key concerns include insecure code, insecure or outdated third-party dependencies (vulnerable to known attacks), build artifacts containing sensitive information, and insecure configurations.

- The build environment should be isolated and securely configured.
- Third-party dependencies should be scanned for vulnerabilities and updated promptly.
- Securely store and transmit build artifacts.

While CD accelerates ML pipeline delivery or deployment, it can introduce security concerns if not managed properly. For automated ML pipeline deployment, the inputs include training and inference packages, and the outputs are the deployed pipelines. For automated model deployment, the inputs are trained ML models and the outputs are serving ML models. The CD pipelines should guarantee the secure delivery and deployment of an ML pipeline or the serving model. The following security concerns require attention:

- Input ML pipelines should be validated, integrity-protected, and signed by the person responsible for their security testing.
- The CD pipeline, delivery, and deployment environment should be securely configured and evaluated regularly.
- The CD pipeline should track the delivery and deployment artifacts and block sensitive ones. Relevant logs should be confidentiality and integrity protected. Delete unnecessary artifacts.

CT refers to regularly re-training an ML model by incorporating new data. It is facilitated by a monitoring component, a feedback loop, and an automated training pipeline that takes the raw data and executes the necessary preprocessing and training steps. CT takes raw data and a training pipeline as inputs to produce a trained model. Metadata logs serve as

both input and output artifacts and should be secured accordingly. Model evaluation and validation are critical components of CT, as they analyze any changes in model quality and security.

Security considerations for CT include:

- Regular security assessments and patching to keep the training pipeline secure
- Data integrity checks to prevent data tampering or injection attacks
- Model evaluation, which assesses changes in model quality, must be conducted in a secure environment

Model Serving

Model serving is the process, when a trained ML model makes inferences in a production environment. Before the model would serve client requests, it must be deployed to the MLOps system. The serving ML model and the inference pipeline are used as inputs for model deployment. Each of these artifacts must be properly secured.

Secure implementation, configuration, and testing of the serving process include:

- A properly configured container and orchestration environment
- Robust access control mechanisms for the inference service.
- Data encryption and access controls. Privacy-enhancing technologies or data de-identification techniques (such as anonymization or pseudonymization) might be employed.
- Model encryption, watermarking, or homomorphic encryption to protect against reverse-engineering, and IP or sensitive data leakage.
- Model inversion and membership inference protections. Model usage should be monitored. When handling direct client requests rather than consuming data streams or processing batches from a single source, it is important to restrict the number of client requests.
- Evasion/adversarial attack protections. The model should be hardened against malicious attacks. Use input validation and sanitization algorithms, limit the number of queries, and implement proper adversarial robustness techniques.

Additionally, regular security audits, least privilege policies, and scalability in security measures can significantly enhance the robustness of the serving process.

Security Monitoring

Securing AI/ML systems is a continuous process that extends beyond development and deployment. Operational procedures to create a controlled and secure environment should be standardized.

Introduce activity monitoring by implementing actionable dashboards, displaying critical metrics such as:

- Model performance indicators
- Usage statistics
- Metadata related to inference requests
- Input and output error tracking

Events can be sorted by significance, facilitating prioritized investigation. Identifiable error types can help categorize and correlate events. This aids in identifying patterns and trends that might not be apparent when looking at events in isolation.

Automated detection and response mechanisms are important. Drift monitoring helps maintain detection capabilities, by helping to detect malicious input, such as adversarial examples. AI/ML-specific monitoring systems should be integrated into the overall dashboard. Simple alerts are effective for regular performance incidents, but advanced security filters and custom responses might be necessary for unexpected inputs or crashes.

Regardless of the specific response, the AI/ML system's issues must be promptly addressed, and best practices and processes continually updated. This includes providing additional training to the relevant personnel.

Suggestions

Considerations before implementing MLSecOps

Existing security posture - Evaluate the AI/ML security practices during development and operation, supplemented by other security protocols, throughout the life cycle.

Team collaboration - Promote robust cross-departmental collaboration, including the engagement of a security practitioner into the MLOps team without impeding the development process. Educate team members on the novelty and specificity of AI/ML threats and countermeasures.

IT Infrastructure assessment - Review the IT infrastructure for its compatibility with MLSecOps, which favors a flexible infrastructure such as a cloud-based system, over traditional rigid ones. It is important to note that MLSecOps is not a standalone solution, but rather an additional layer on top of existing cybersecurity mechanisms, which should be robust and mature. Establishing an Information Security Management System (ISMS) can facilitate MLSecOps.

Setting clear objectives - This involves automating threat detection for AI/ML, developing defense evasion susceptibility, or predicting security incidents. Specific objectives will guide MLSecOps' strategy and help measure its effectiveness.

Steps to implement MLSecOps

AI/ML Security competence - Assign security practitioners with appropriate skills to assist the MLOps team and ensure adherence to security practices throughout the AI/ML development life cycle. They will also promote security awareness and help all team members understand their roles in securing AI/ML.

Integration of security tools - Automate security tooling within the development process. Integrate tools for data analysis, static code analysis, dynamic testing, and dependency checking into the CI/CD pipeline.

Continuous monitoring and improvement - MLSecOps requires regular reviews and improvements. The MLOps process should be monitored, logged, and audited regularly as new threats appear frequently. Security incident handling is valuable for learning about them and enhancing the overall MLSecOps.

Potential obstacles and ways to overcome them

Lack of skilled personnel - MLSecOps calls for a solid understanding of both ML and cybersecurity, including which threats can be mitigated with conventional controls, and which require ML-specific ones. If your organization lacks such expertise, consider recruiting specialists or investing in competence development.

Data privacy concerns - Mitigate privacy concerns by anonymizing data, using differential privacy, and keeping up to date with the latest regulatory advancements.
Constantly evolving threats - AI/ML threats constantly change, potentially making even recently developed ML models susceptible to attacks. The model development and deployment process need to be flexible and undergo regular updates.

Tooling challenges - Integrating the appropriate security tools can be intimidating. Initially, focus on identifying the most crucial ones that can be easily integrated. As the process becomes stable, use more sophisticated tools.

Process overhead leading to resistance to change - Implementing security measures can initially slow down development, leading to resistance. Demonstrate that incorporating security in the early stages of ML preparation will minimize future concerns. Make it clear that AI/ML systems are often opaque and data-intensive. Without a comprehensive approach to protecting the AI/ML development life cycle, potential security and privacy issues could arise later, at a greater overall cost.

Conclusion

As the deployment of AI/ML in telecom networks plays a pivotal role, the need to secure it becomes critical. AI can be a powerful tool for telecom systems, but it must be secure. Ericsson is actively engaged in research and development projects to secure AI, including integrating MLSecOps practices into SDLC. MLSecOps is a comprehensive, highly automated approach that ensures that AI/ML systems are secure by design, default, and deployment. Rather than a standalone model, it is a layer that enhances existing security measures and operational practices.

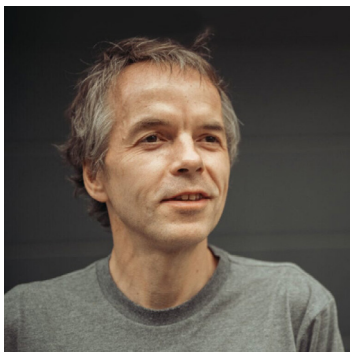
Before diving into MLSecOps, organizations must evaluate their security posture, encourage cross-departmental collaboration, assess their IT infrastructure, and set clear objectives. MLSecOps should not be seen as a replacement for existing security protocols but as a supplement that enhances every stage of AI/ML development.

MLSecOps is a continually evolving approach for integrating security into the AI/ML development process. Ericsson is at the forefront of this initiative, incorporating MLSecOps principles to enhance its product development and operations security. This ensures leveraging the benefits of AI in telecom networks, which is done in a secure, efficient, and sustainable manner.

References

1. Trustworthy AI - What it means for telecom https://www.ericsson.com/en/reports-and-papers/white-papers/trustworthy-ai
2. Machine Learning Operations (MLOps): Overview, Definition, and Architecture, https://ieeexplore.ieee.org/abstract/document/10081336
3. ISO27001, Information security management systems https://www.iso.org/standard/27001
4. https://github.com/LaraMauri/STRIDE-AI
5. https://github.com/Azure/AI-Security-Risk-Assessment/blob/main/AI_Risk_Assessment_v4.1.4.pdf
6. https://atlas.mitre.org/
7. OWASP Machine Learning Security Top Ten, https://owasp.org/www-project-machine-learning-security-top-10/
8. https://owasp.org/www-project-samm/

Authors



Attila Ulbert joined Ericsson in 2015 and he is currently Artificial Intelligence System Manager. In his enthusiastic journey with Ericsson, he lead the development of Ericsson's AI platform, and worked on fundamental AI studies on security, trustworthiness, and industrialization. Attila has a PhD in Informatics from Eötvös Loránd University. He is a marathoner.

Andrey Shorov is a Senior Security Technology Specialist at Product Security, who joined Ericsson in 2019. He works on security technologies for telecommunication networks, including AI/ML security, Zero Trust, and network slicing. Shorov has a Ph.D. in computer science from the St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences and holds ISC2 CISSP and CompTIA Security+ certifications.

Elif Ustundag Soykan joined Ericsson Research in 2018 and worked with security research for four years before moving to her current position in Product Security. Prior to joining Ericsson, she worked several years at The Scientific and Technological Research Council (TUBITAK), National Cryptology Institute in Türkiye, in the security domain. She achieved CISSP (Certified Information Systems Security Professional) certification in 2015. Elif holds Ph.D. and Master's degrees in Computational Science and Engineering from Istanbul Technical University, Türkiye, and a B.S degree in Computer Engineering from Istanbul University, Türkiye.

Göran Hall is an Expert in Network Architecture Evolution (AI/ML) at the CTO office. He joined Ericsson in 1991 to work on development and standardization, primarily within the area of Packet Core network architecture, which has so far includes GPRS, WCDMA, PDC, EPC and 5G Core. He has been chief network architect for the Packet Core domain in his previous assignment, including responsibility for the functional requirements and architecture for the 5G Core network. Hall holds an M.Sc. in Electrical Engineering from Chalmers University of Technology in Gothenburg, Sweden.

Jim Reno is a distinguished Engineer at Ericsson, where he works on security aspects of Artificial Intelligence as applied to telecommunication systems. He has more than 40 years of industry experience in fields including system software (operating systems, networking, system management, and cloud native systems), payment system security, authentication, authorization and identity management.