hackerone

NULLCON

MLOps Under Attack: Threat Modeling Modern AI Systems

Sandeep Singh

March 01, 2025

About Me

- Director, Security Strategy & Operations @ HackerOne

Interests

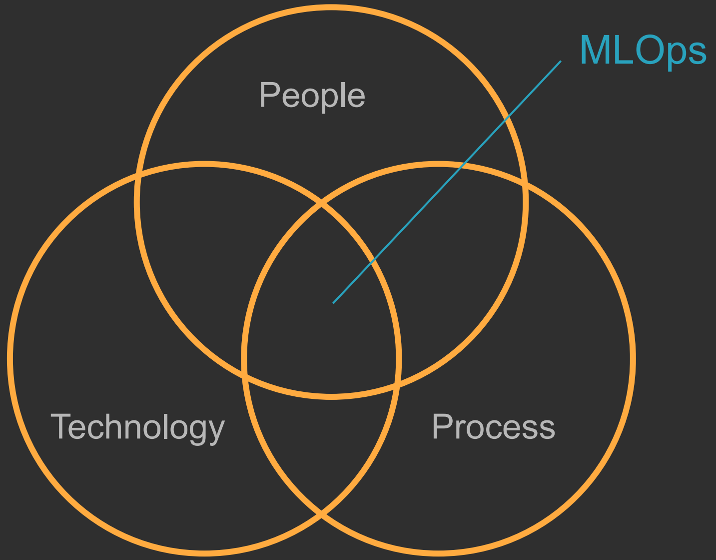
- Appsec, cloud security, vulnerability management, response
- Vulnerability disclosures, coordination, bug bounties

Agenda

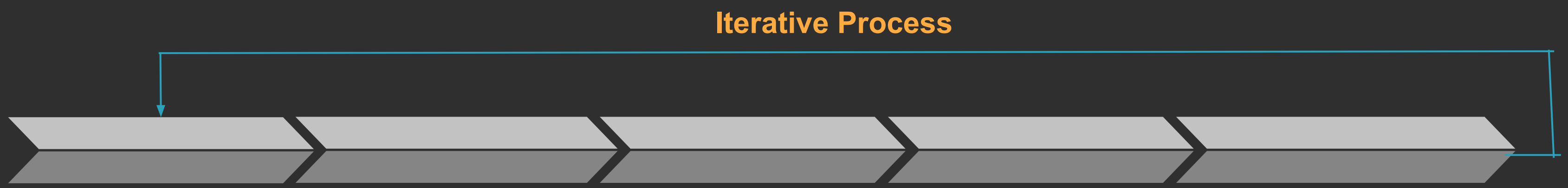
- Overview of ML lifecycle, platforms, and supply chain
- Attack surface and attack scenarios
- Building for security (defensive practices)
- Example Tabletop Scenarios

Understanding MLOps

MLOps is a unified engineering practice and cultural approach that integrates the ML system development (Dev) and ML system operation (Ops).



Understanding MLOps



Plan / Design

Problem framing, collect, clean, process data, feature engineering

Build / Train

Write code to develop model, fine-tune model,

Evaluate / Test

Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model

Deploy

Deploy to production, client-side apps, APIs, etc. Infrastructure management

Manage / Monitor

System wide monitoring and debugging for model reliability and overall health of the system

Understanding MLOps

Pre-Training | Training | Inference

Plan / Design

Problem framing, collect, clean, process data, feature engineering

Build / Train

Write code to develop model, fine-tune model,

Evaluate / Test

Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model

Deploy

Deploy to production, client-side apps, APIs, etc. Infrastructure management

Manage / Monitor

System wide monitoring and debugging for model reliability and overall health of the system

Understanding MLOps

- Jupyter
- Hugging Face
- Git
- Pandas

Pre-Training | Training | Inference

Plan / Design

Problem framing, collect, clean, process data, feature engineering

Build / Train

Write code to develop model, fine-tune model,

Evaluate / Test

Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model

Deploy

Deploy to production, client-side apps, APIs, etc. Infrastructure management

Manage / Monitor

System wide monitoring and debugging for model reliability and overall health of the system

Understanding MLOps

- Jupyter
- Hugging Face
- Git
- Pandas

- PyTorch, TensorFlow
- MLflow
- Kubeflow
- Weights & Biases

Pre-Training	Training		Inference	
Plan / Design	**Build / Train**	**Evaluate / Test**	**Deploy**	**Manage / Monitor**
Problem framing, collect, clean, process data, feature engineering	Write code to develop model, fine-tune model,	Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model	Deploy to production, client-side apps, APIs, etc. Infrastructure management	System wide monitoring and debugging for model reliability and overall health of the system

Understanding MLOps

- Jupyter
- Hugging Face
- Git
- Pandas

- PyTorch, TensorFlow
- MLflow
- Kubeflow
- Weights & Biases

- Alibi
- MLflow
- Weights & Biases
- Fairlearn

Pre-Training	Training	Inference

Plan / Design

Problem framing, collect, clean, process data, feature engineering

Build / Train

Write code to develop model, fine-tune model,

Evaluate / Test

Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model

Deploy

Deploy to production, client-side apps, APIs, etc. Infrastructure management

Manage / Monitor

System wide monitoring and debugging for model reliability and overall health of the system

Understanding MLOps

Plan / Design	Build / Train	Evaluate / Test	Deploy	Manage / Monitor
→ Jupyter → Hugging Face → Git → Pandas	→ PyTorch, TensorFlow → MLflow → Kubeflow → Weights & Biases	→ Alibi → MLflow → Weights & Biases → Fairlearn	→ Docker, Kubernetes → MLFlow → Kubeflow → Seldon Core	
Pre-Training	Training		Inference	
Problem framing, collect, clean, process data, feature engineering	Write code to develop model, fine-tune model,	Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model	Deploy to production, client-side apps, APIs, etc. Infrastructure management	System wide monitoring and debugging for model reliability and overall health of the system

Understanding MLOps

- ➔ Jupyter
- ➔ Hugging Face
- ➔ Git
- ➔ Pandas

- ➔ PyTorch, TensorFlow
- ➔ MLflow
- ➔ Kubeflow
- ➔ Weights & Biases

- ➔ Alibi
- ➔ MLflow
- ➔ Weights & Biases
- ➔ Fairlearn

- ➔ Docker, Kubernetes
- ➔ MLFlow
- ➔ Kubeflow
- ➔ Seldon Core

- ➔ Splunk
- ➔ ELK
- ➔ Prometheus
- ➔ Grafana

Pre-Training | Training | Inference

Plan / Design

Problem framing, collect, clean, process data, feature engineering

Build / Train

Write code to develop model, fine-tune model,

Evaluate / Test

Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model

Deploy

Deploy to production, client-side apps, APIs, etc. Infrastructure management

Manage / Monitor

System wide monitoring and debugging for model reliability and overall health of the system

Understanding MLOps

- Jupyter
- Hugging Face
- Git
- Pandas

- PyTorch, TensorFlow
- MLflow
- Kubeflow
- Weights & Biases

- Alibi
- MLflow
- Weights & Biases
- Fairlearn

- Docker, Kubernetes
- MLFlow
- Kubeflow
- Seldon Core

- Splunk
- ELK
- Prometheus
- Grafana

Cloud Pipelines (Amazon Sagemaker, Azure AI, Google Cloud Vertex AI)

Pre-Training | Training | Inference

Plan / Design

Problem framing, collect, clean, process data, feature engineering

Build / Train

Write code to develop model, fine-tune model,

Evaluate / Test

Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model

Deploy

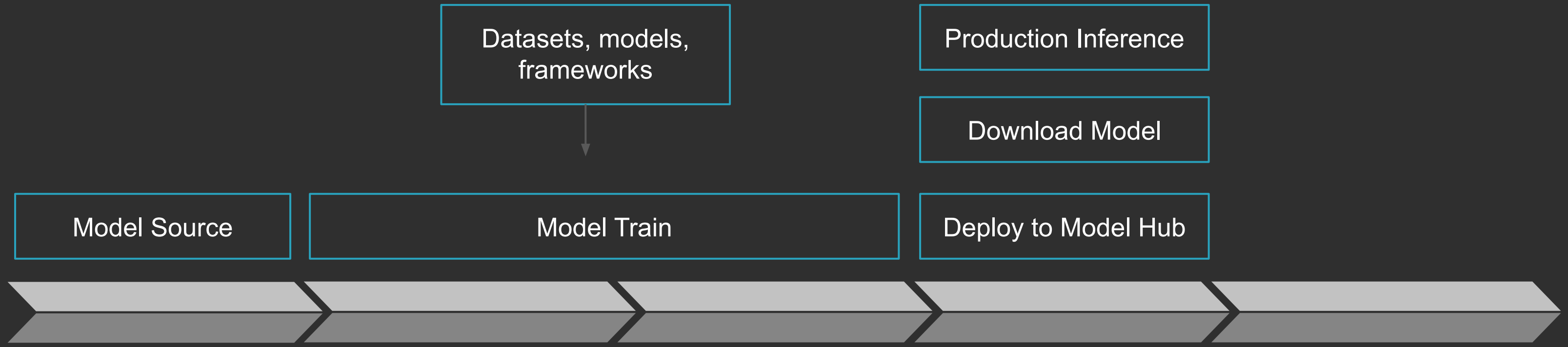
Deploy to production, client-side apps, APIs, etc. Infrastructure management

Manage / Monitor

System wide monitoring and debugging for model reliability and overall health of the system

Understanding MLOps

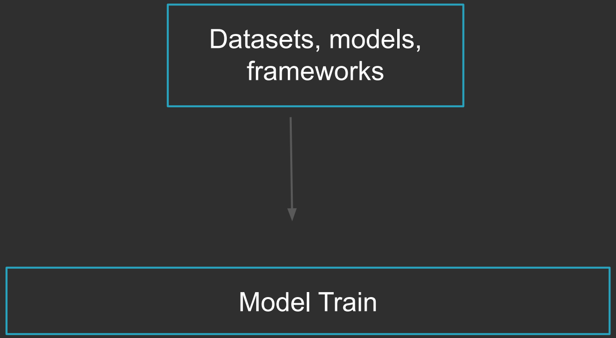
The ML Model Development Lifecycle



Plan / Design	Build / Train	Evaluate / Test	Deploy	Manage / Monitor
Problem framing, collect, clean, process data, feature engineering	Write code to develop model, fine-tune model,	Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model	Deploy to production, client-side apps, APIs, etc. Infrastructure management	System wide monitoring and debugging for model reliability and overall health of the system

Understanding MLOps

The ML Model Development Lifecycle



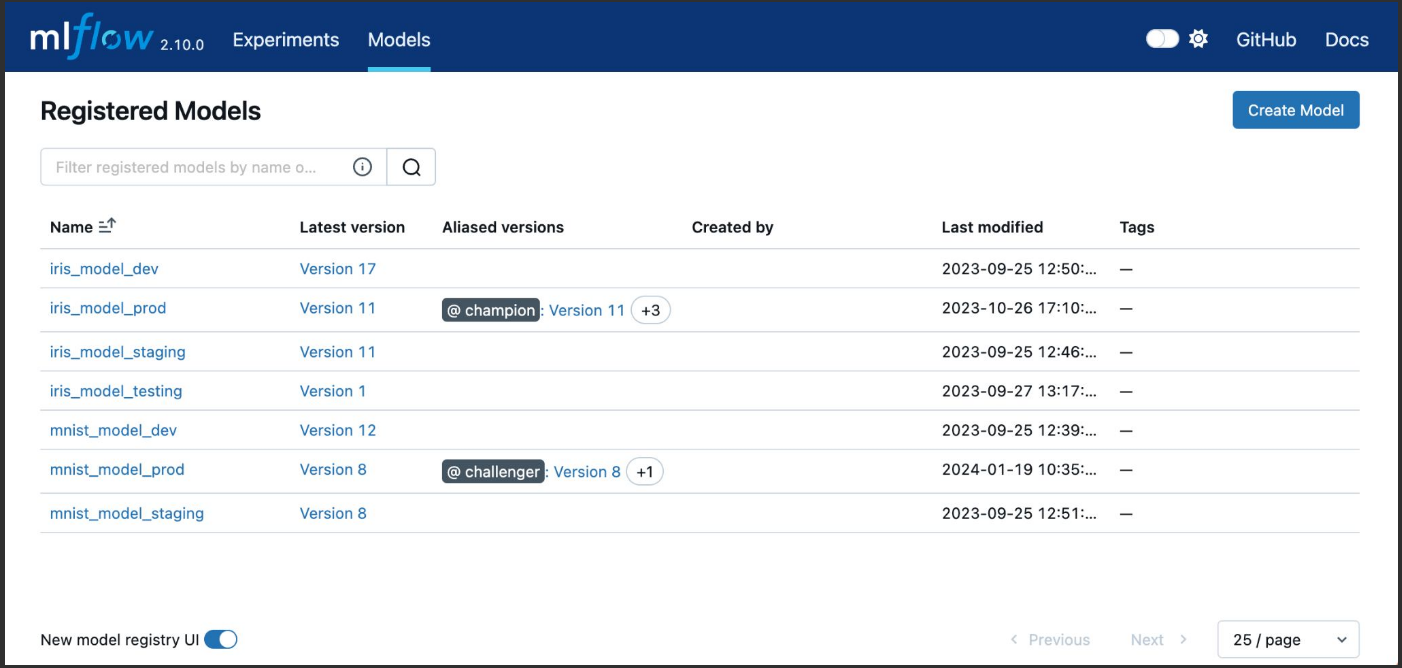

```python
# Load and prepare the data
data = pd.read_csv('customer_data.csv')
X = data.drop('churn', axis=1)  # Features
y = data['churn']  # Target variable

# Handle categorical features
X = pd.get_dummies(X, drop_first=True)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

# Make predictions
y_pred = model.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Model accuracy: {accuracy:.4f}')
print(classification_report(y_test, y_pred))

# Save the model
import joblib
joblib.dump(model, 'churn_prediction_model.pkl')
```

Understanding MLOps

The ML Model Development Lifecycle

Model Registry

Deploy to Model Hub

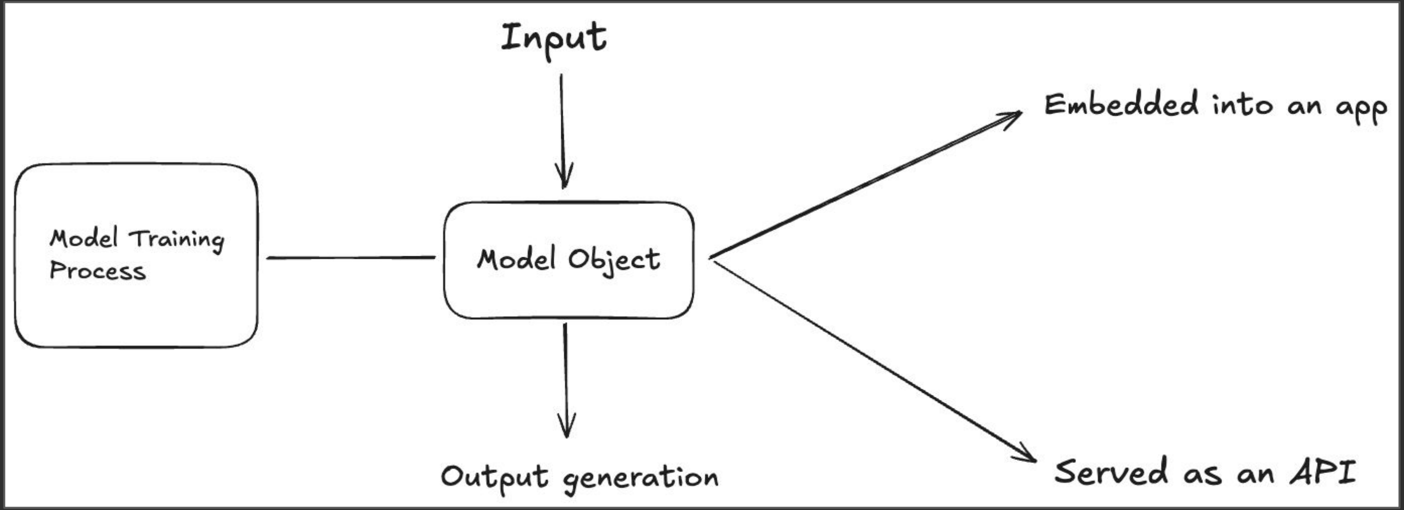


Understanding MLOps

The ML Model Development Lifecycle

Model Serving

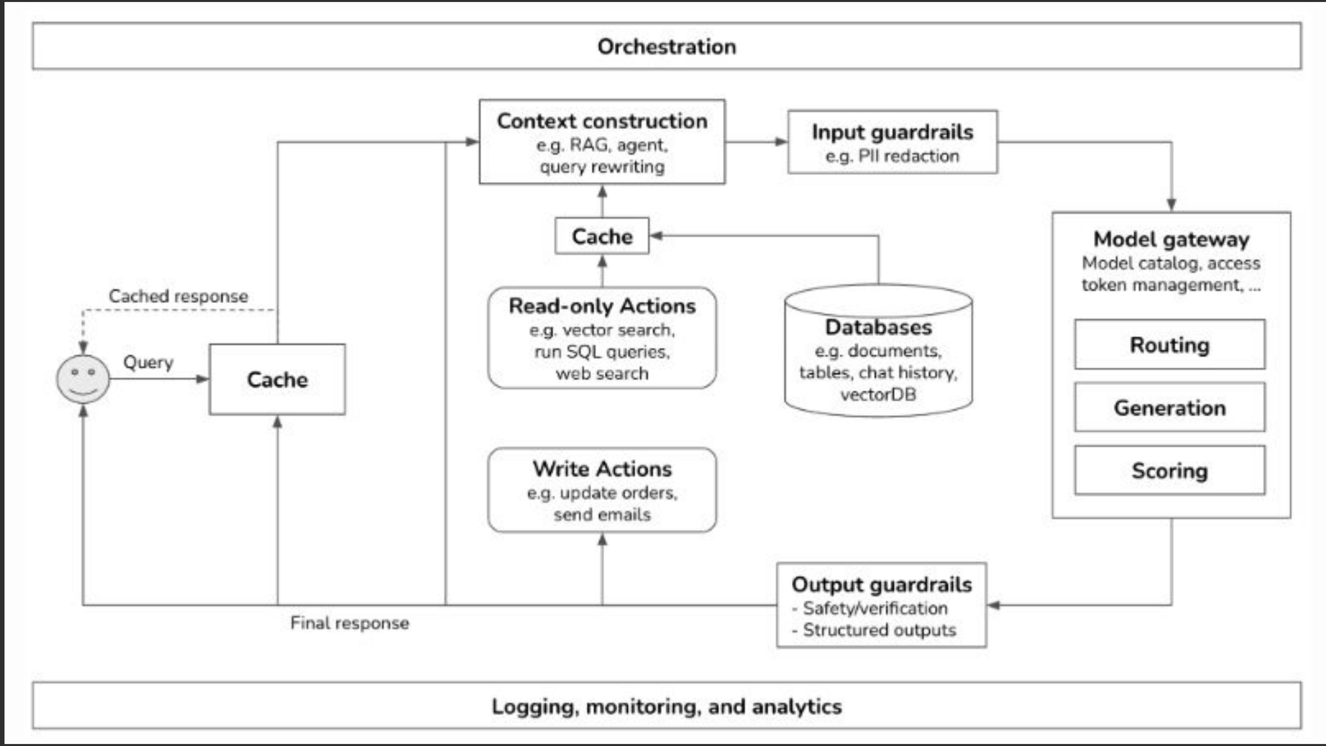
Production Inference



Understanding LLMOps

- Focus on LLM development and managing model in production
- Broad design of entire end-to-end application (front-end, back-end, data engineering, etc.)
- Experimentation on foundation models
- Fine tuning
- Monitoring
- Evaluate generative output

Understanding LLMOps



Example generative AI platform architecture

https://huyenchip.com/2024/07/25/genai-platform.html

Attack Scenarios

- Data poisoning
- Model inversion
- Stealing sensitive data

- Data extraction
- Model extraction
- Backdooring model
- Code execution
- 3rd Party Software Supply Chain

- Model poisoning
- Model extraction
- Container escapes

Plan / Design

Problem framing, collect, clean, process data, feature engineering

Build / Train

Write code to develop model, fine-tune model,

Evaluate / Test

Test for quality, accuracy and performance. Evaluate model performance against set eval criteria. Select model

Deploy

Deploy to production, client-side apps, APIs, etc. Infrastructure management

Manage / Monitor

System wide monitoring and debugging for model reliability and overall health of the system

Attack Scenarios

- Credentials compromise to gain access in the ML pipeline
    - Attackers steal authentication credentials through phishing or exposed secrets, gaining access to inject poisoned data or steal proprietary models.
- Misconfigured access control leads to privilege escalation and lateral movement
    - Attackers exploit overly permissive roles or improperly segmented environments to escalate privileges and move laterally across ML infrastructure.

Attack Scenarios

- Supply Chain attacks through third party libraries, data
  - Malicious code in dependencies or poisoned public datasets compromise model integrity, enabling backdoors or data leakage.
- Vulnerabilities in MLOps platforms
  - Insecure container configurations, deserialization vulnerabilities, or insufficient isolation in ML platforms allow arbitrary code execution or access to sensitive artifacts.

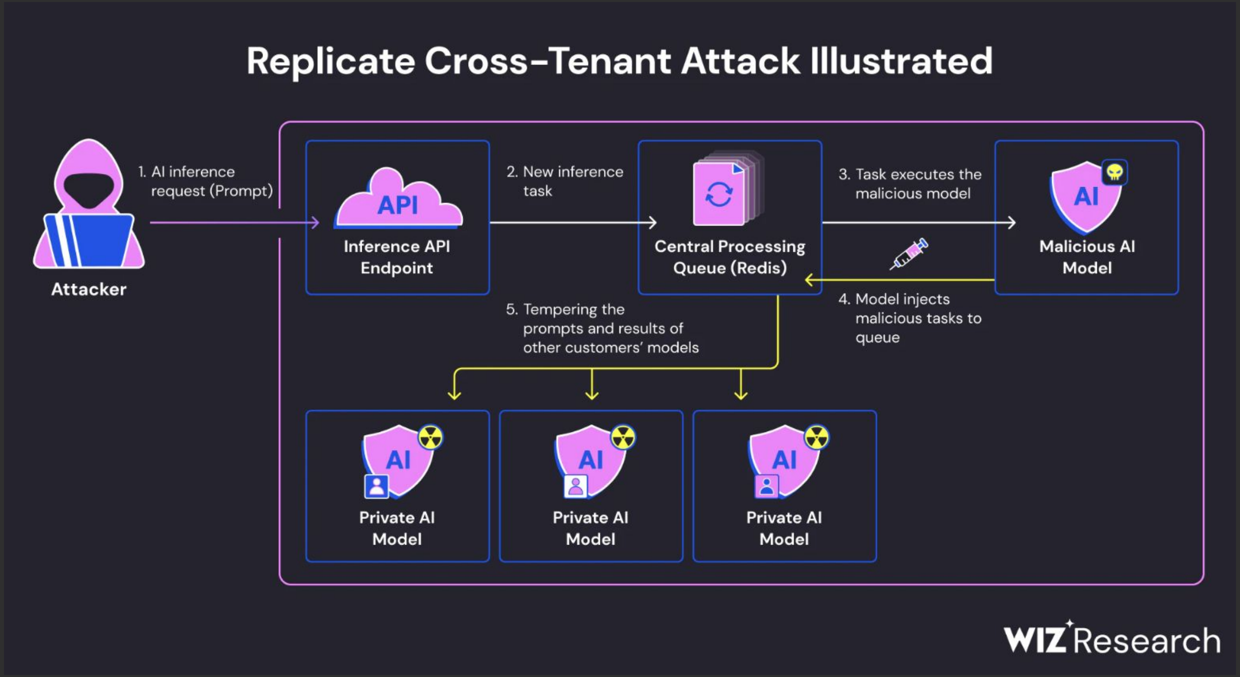
Attack Scenarios

From MLOps to MLOops - Exposing the Attack Surface of Machine Learning Platforms | BlackHat USA 2024 - Security analysis of popular open source ML platforms by JFrog research team

Abusing MLOps Platforms to Compromise ML Models and Enterprise Data Lakes | X-Force Red research on attacks against MLOps platforms after an attacker has obtained valid credential material. Open Source toolkit MLOkit

Attack Scenarios

[Hosting Malicious Models - AI-as-a-Service provider risks | Wiz]



Attack Scenarios

Malicious ML models discovered on Hugging Face platform | Reversing Labs Research Two malicious models containing reverse shell payloads that evaded detection by exploiting limitations in Hugging Face's Picklescan security tool.

Data Scientists Targeted by Malicious Hugging Face ML Models with Silent Backdoor | JFrog Research

Attack Surface

- Authentication and Access Control Vulnerabilities
    - User token/credential stealing via phishing
    - Misconfigured internal network resources
    - Exploitation of misconfigured or overly permissive IAM roles
    - Service account compromise
    - API key exposures in notebooks or code repositories

Attack Surface

- Infrastructure Vulnerabilities
  - Container escape in model training/ model serving environments
  - Resource exhaustion through crafted training jobs
  - Network pivoting through compromised ML instances due to insufficient network segregation

Attack Surface

- CI/CD / Supply Chain
  - Vulnerabilities in third party software components
  - Exploitation of outdated dependencies in ML environments
  - Implementation issues in ML Ops platforms and ML components

- API and Model Inference Vulnerabilities
  - SSRF through model serving endpoints
  - Prompt Injection

CVE Landscape

Notable CVEs

Platform	CVE	CVSS	Vulnerability Type	Details
MLFlow	CVE-2023-6977	7.5 (High)	Path Traversal	Local file inclusion due to path traversal in GitHub repository mlflow/mlflow
	CVE-2023-6018	9.8 (Critical)	OS Command Injection	RCE via/ajax-api/2.0/mlflow/model-versions/create endpoint.
	CVE-2024-0520	9.4 (Critical)	Path Traversal → RCE	Arbitrary file write via HTTP dataset source parsing, fixed in v2.9.0.
Kubeflow	CVE-2023-6570	7.7 (High)	Server-Side Request Forgery	SSRF enabling internal network reconnaissance.
Weights & Biases	CVE-2024-4642	9.1 (Critical)	SSRF via HTTP 302 Redirection	Redirect mishandling allowed access to internal APIs.

Building for Security

Data Protection

- Encrypt training data and model
- Data provenance tracking to ensure integrity throughout the pipeline
  - e.g., - S3 object lock, S3 versioning
- Granular access control to training data stores
- Scan data for PII, PHI, and other sensitive data before training or fine tuning

Building for Security

Example organization wide SCP to prohibit changes to Amazon Sagemaker models inside AWS

```
"Statement": [{
  "Effect": "Deny",
  "Action": [
    "sagemaker:DeleteModel",
    "sagemaker:CreateEndpoint",
    "sagemaker:UpdateEndpointWeightsAndCapacities",
    "sagemaker:DeleteEndpoint",
    "sagemaker:UpdateEndpoint",
    "sagemaker:AddTags",
    "sagemaker:DeleteEndpointConfig",
    "sagemaker:DeleteTags"
  ],
  "Resource": "<Model ARN>"
}]
```

Building for Security

Code and Model

- Signed commits and code reviews for all model development
- Scan container images, and functions
- Scan for dependencies
- SBOM to understand supply chain
- Test model against adversarial examples
- Input validation and sanitization on inference endpoints
- Rate limiting on APIs

Building for Security

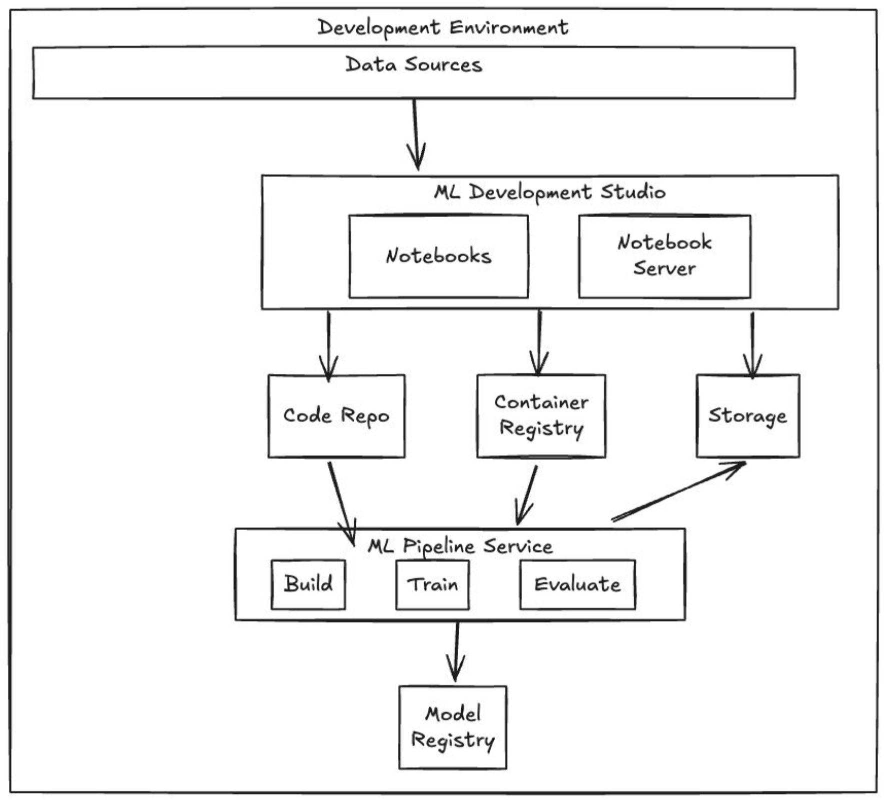
Infrastructure

- IaC with security checks
- Segmentation controls to prevent exfiltration
- Fine grained permissions for cloud pipelines

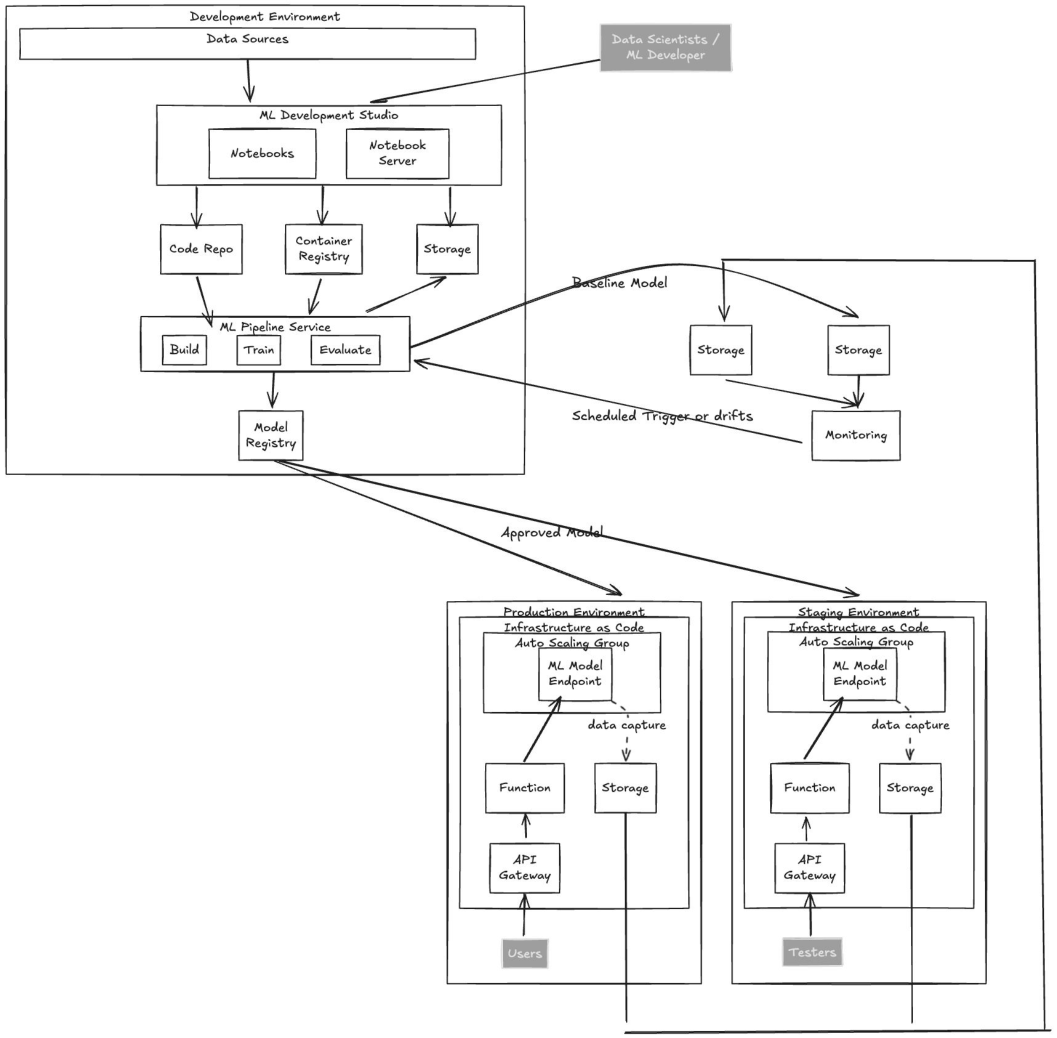
```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Deny",
      "Action": [
        "sagemaker:DeletePipeline"
      ],
      "Resource": [
        "arn:aws:sagemaker:{Region}:{Account}:pipeline/{PipelineName}"
      ]
    }
  ]
}
```

Example organization wide SCP to prevent deletion of SageMaker pipelines

Building for Security

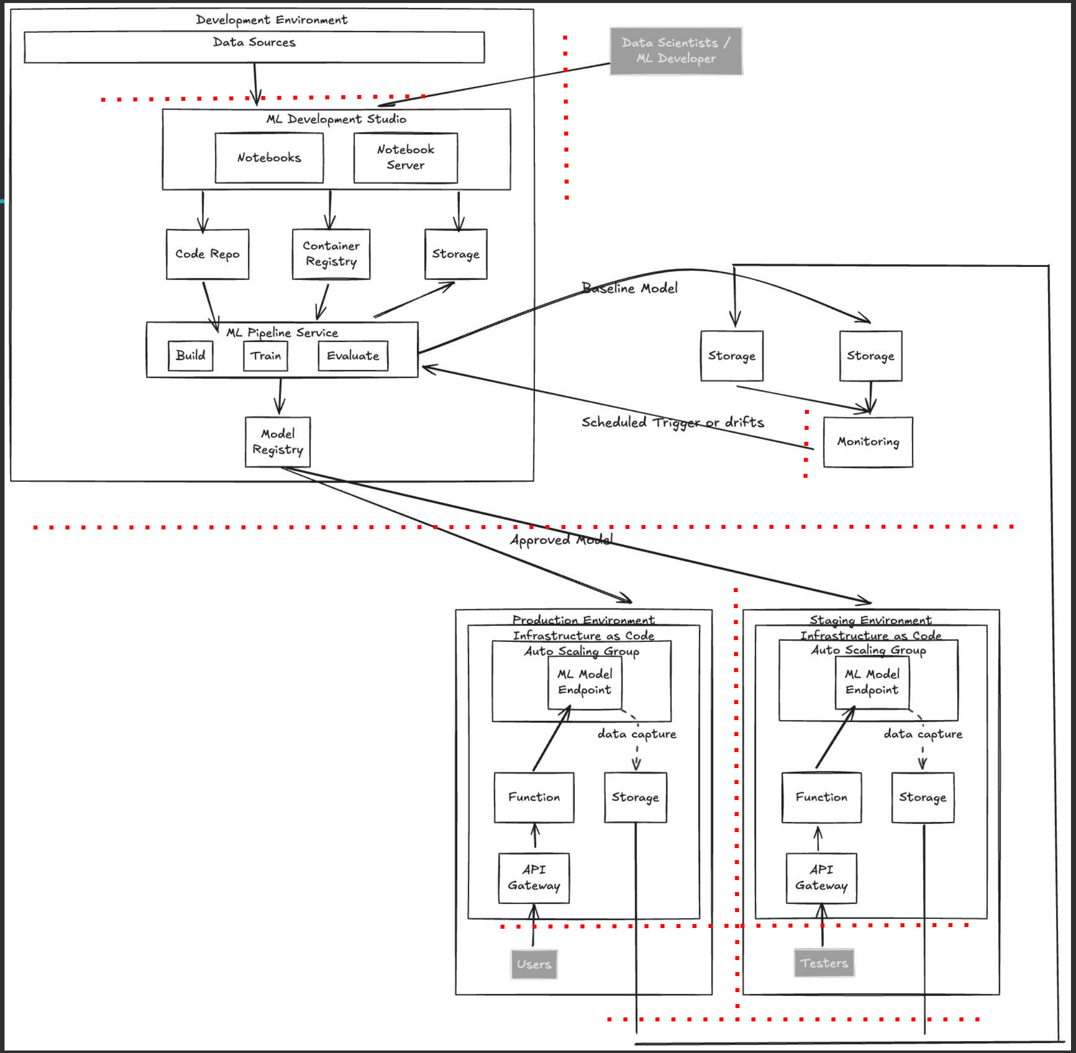


Building for Security



Building for Security

Possible Security Boundaries



Example Tabletop Scenarios

- An authentication bypass vulnerability in your model serving infrastructure allows unauthenticated access to models and protected data used for personalization.

- An attacker has gained access to a notebook server and is attempting to use it to pivot into more sensitive infrastructure components that host production models.

- An internal user has deployed an unauthorized shadow model that mimics your production API but sends data to external servers for unknown purposes.

Example Tabletop Scenarios

- An attacker has achieved container escape on your ML training infrastructure and is accessing the underlying host system to compromise other workloads.

- Your model inference API is experiencing a sophisticated distributed denial of service attack specifically targeting your most computationally expensive models.

Interested in Finding Bugs?

- https://github.com/kubeflow/pipelines/security
- https://github.com/SeldonIO/seldon-core/security
- https://github.com/mlflow/mlflow/security
- https://github.com/aws/amazon-sagemaker-examples/security/policy
- MSRC [Azure AI]
- Google VRP [Vertex AI]
- And More

Reporting Security Issues

Amazon Web Services (AWS) is dedicated to the responsible disclosure of security vulnerabilities.

We kindly ask that you **do not** open a public GitHub issue to report security concerns.

Instead, please submit the issue to the AWS Vulnerability Disclosure Program via HackerOne or send your report via email.

For more details, visit the AWS Vulnerability Reporting Page.

Thank you in advance for collaborating with us to help protect our customers.

Reporting a Vulnerability

When finding a security vulnerability in MLflow, please perform the following actions:

- Open an issue on the MLflow repository. Ensure that you use `[BUG] Security Vulnerability` as the title and vulnerability details in the issue post.
- Send a notification email to `mlflow-oss-maintainers@googlegroups.com` that contains, at a minimum:
  - The link to the filed issue stub.
  - Your GitHub handle.
  - Detailed information about the security vulnerability, evidence that supports the relevance of the finding and any reproducibility instructions for independent confirmation.

This first stage of reporting is to ensure that a rapid validation can occur without wasting the time and effort of a reporter. Future communication and vulnerability resolution will be conducted after validating the veracity of the reported issue.

An MLflow maintainer will, after validating the report:

- Acknowledge the bug during triage
- Mark the issue as `priority/critical-urgent`
- Open a draft GitHub Security Advisory to discuss the vulnerability details in private.

The private Security Advisory will be used to confirm the issue, prepare a fix, and publicly disclose it after the fix has been released.

Report a bug

Vulnerabilities can be reported to security@wandb.com.

- In your email to the Security team, include in the subject line: Bug Bounty Program Disclose: [vulnerability]
- Refer to the Rules of Engagement for additional details for reporting.

Thank You