| **Threat Vector** | **Python Tool** | **Purpose** |
| --- | --- | --- |
| **Unauthorized Access** | boto3 (AWS), google-cloud-storage (GCP), azure-identity, paramiko | Check access permissions, user roles, and storage policies |
| **Encryption Risks** | cryptography, ssl, boto3, google-cloud-storage, azure-blob-storage | Ensure data is encrypted at rest and in transit |
| **Model Integrity** | hashlib, pickle, boto3, os | Compute and verify hashes to detect model tampering or corruption |
| **Data Poisoning** | pandas, scikit-learn, numpy, scipy | Detect anomalous feature values that may indicate data poisoning |
| **Model Theft** | requests, API logging systems, Flask, FastAPI | Monitor API usage for abnormal patterns and limit model API exposure |
| **Feature Leakage** | pandas, pydeequ, sklearn | Inspect features for sensitive data and apply data sanitization |
| **API Abuse** | requests, Flask, FastAPI | Log and monitor API traffic for abuse or excessive access patterns |
| **Version Mismatch** | MLflow, DVC, boto3, os | Track and version models and data to prevent version mismatch issues |
| **DDoS** | psutil, aws-cloudwatch, gcp-monitoring | Monitor and mitigate denial-of-service attacks on storage or APIs |

**Python Tools to Check for Threat Vectors**

Here are the Python tools you can use to detect and mitigate these threat vectors during feature and model storage:

**Threat Vectors in the Dev Environment for Supervised ML Algorithms**

1. **Data Poisoning**:
   * An attacker may inject malicious data into the training set to influence model behavior.
   * **Impact**: Model performance degradation, incorrect predictions.
   * **Tool**: scikit-learn, anomaly detection algorithms.
2. **Model Tampering**:
   * Unauthorized modifications to models in the catalog, leading to tampered predictions or behavior.
   * **Impact**: Compromised models producing incorrect outputs.
   * **Tool**: hashlib, integrity checks (hashes).
3. **Unauthorized Access**:
   * Unauthorized access to the catalog may lead to data leakage or model theft.
   * **Impact**: Confidentiality breaches.
   * **Tool**: boto3, google-cloud-storage, access control checks.
4. **Unencrypted Model/Feature Storage**:
   * Lack of encryption for stored models and features could lead to exposure in the event of unauthorized access.
   * **Impact**: Confidential data or models are exposed.
   * **Tool**: cryptography, for encryption of data at rest.
5. **Model Theft**:
   * An attacker may steal and reuse proprietary models, leading to loss of competitive advantage.
   * **Impact**: Intellectual property theft.
   * **Tool**: boto3, os, checking for public access to stored models.
6. **Versioning Issues**:
   * Poor version control may lead to deployment of outdated or unapproved models.
   * **Impact**: Outdated or incorrect models being used.
   * **Tool**: MLflow, DVC, versioning checks.
7. **Sensitive Data Exposure**:
   * Feature data may inadvertently contain sensitive information.
   * **Impact**: Breach of privacy or legal compliance (GDPR, CCPA).
   * **Tool**: pandas, pydeequ for feature quality and sensitivity analysis.
8. **Adversarial Attacks**:
   * Attackers may create specific input data designed to fool the model during inference.
   * **Impact**: Incorrect predictions in a production environment.
   * **Tool**: Adversarial robustness tools like cleverhans.
9. **Weak API Security**:
   * Models served through APIs may be exposed to abuse if security is insufficient.
   * **Impact**: Unauthorized access or denial of service.
   * **Tool**: Flask, FastAPI, monitoring API usage patterns.
10. **Python Tools to Identify Threat Vectors**

| **Threat Vector** | **Python Tool** | **Purpose** |
| --- | --- | --- |
| **Data Poisoning** | scikit-learn, pandas, numpy | Detect anomalies in the dataset |
| **Model Tampering** | hashlib, os | Check model integrity using hash comparison |
| **Unauthorized Access** | boto3, google-cloud-storage, azure-identity | Check access control permissions |
| **Unencrypted Model Storage** | cryptography, boto3, os | Ensure encryption of models and data at rest |
| **Model Theft** | boto3, os | Monitor for public access to stored models |
| **Versioning Issues** | MLflow, DVC, os | Track model versions and ensure proper version control |
| **Sensitive Data Exposure** | pandas, pydeequ | Analyze features for sensitive information |
| **Adversarial Attacks** | cleverhans, pytorch, tensorflow | Detect adversarial inputs and enhance model robustness |
| **Weak API Security** | Flask, FastAPI, requests | Monitor and secure API endpoints used for serving models |

**Key Features of the Code:**

1. **Data Poisoning Check**: Uses IsolationForest to detect anomalies in the dataset that may indicate data poisoning.
2. **Model Integrity Check**: Uses hashlib to generate a hash of the model and compares it with the saved hash to ensure model integrity.
3. **Model Encryption**: Serializes and encrypts the trained model using the cryptography library to ensure it is securely stored.
4. **Unauthorized Access Check**: Uses boto3 to check for unauthorized (public) access to an S3 bucket where models might be stored.
5. **Logging**: Logs all checks, including data poisoning alerts, model integrity verification, and access control issues.

**Python Tools Used:**

* **scikit-learn**: For training models and anomaly detection (data poisoning).
* **cryptography**: For encrypting the serialized models before storage.
* **hashlib**: For generating and verifying model integrity using SHA-256 hashes.
* **boto3**: For access control verification in cloud storage (AWS S3).
* **logging**: To log potential threat alerts and warnings.