Husky AI

Owner:
Reviewer:
Contributors: Imported from TM-BOM
Date Generated: Tue Oct 07 2025

Executive Summary

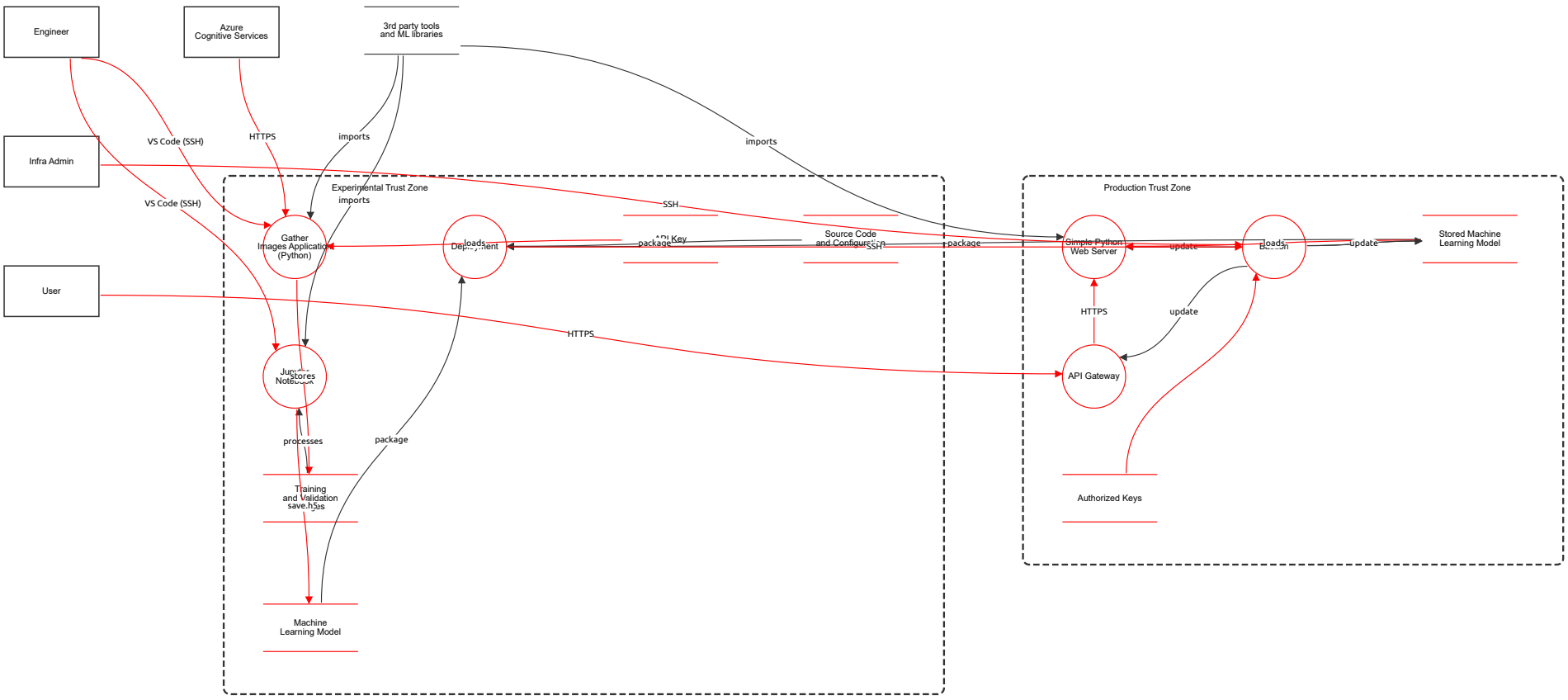
High level system description

A machine learning system to classify Huskies vs dogs. HuskyAI is a machine learning system designed to classify images and distinguish between huskies and non-huskies. It integrates secure data handling practices with a robust convolutional neural network (CNN) for image recognition. Secure Image Retrieval: HuskyAI uses TLS to securely fetch images from Azure Cognitive Services, ensuring encryption during data transmission and validating the server's authenticity to prevent man-in-the-middle attacks. Data Storage and Access Controls: Azure Blob Storage is used to store datasets, with public access fully blocked. Access is controlled using Role-Based Access Control (rbac) and Attribute-Based Access Control (ABAC) to enforce granular, identity-based permissions. Jupyter Notebooks, which host model development and experimentation, are also secured with rbac and ABAC, preventing unauthorized public access. Developer Authentication: Developers access the system through SSH keys protected by passphrases. This adds an additional layer of security, reducing the likelihood of unauthorized access even if keys are exposed. Model and Dataset Dataset Composition: The dataset comprises approximately 1,300 husky images and 3,000 non-husky images sourced via Bing's image search. Data undergoes manual cleansing and is split into training and validation sets to enhance model performance. Model Design: HuskyAI employs a CNN with: Convolutional layers for feature extraction. Max-pooling layers for dimensionality reduction. Dropout layers to prevent overfitting. Dense layers for final classification. The model is trained with the Adam optimizer and a learning rate of 0.0005, optimized for accuracy and computational efficiency. Security Considerations rbac and ABAC controls across storage and development environments ensure sensitive data and configurations are protected. TLS ensures secure communication channels, preventing eavesdropping or data interception during image retrieval. Applications HuskyAI is tailored for accurate image classification and can be adapted for other domains requiring precise visual differentiation, with a focus on maintaining strong security postures. HuskyAI combines state-of-the-art machine learning techniques with stringent security controls, including secure communications, robust access management, and encrypted developer authentication, to deliver a reliable and secure image classification system.

Summary

Total Threats	42
Total Mitigated	0
Total Open	42
Open / Critical Severity	0
Open / High Severity	25
Open / Medium Severity	15
Open / Low Severity	2

Husky AI


Engineer
Azure Cognitive Services
3rd party tools and ML libraries
Infra Admin
User
VS Code (SSH)
HTTPS
imports
Experimental Trust Zone
Gather Images Application (Python)
Deployment
SSH
API Key
package
Source Code and Configuration
Jupyter Notebook
stores
processes
Training and Validation save.h5as
Machine Learning Model
loads
HTTPS
Production Trust Zone
Simple Python Web Server
update
Stored Machine Learning Model
API Gateway
Authorized Keys

Husky AI

Engineer (Actor)

Description: A Data Engineer responsible for building, training, and deploying machine learning models.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Infra Admin (Actor)

Description: Administrator responsible for securing and maintaining production infrastructure.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Azure Cognitive Services (Actor)

Description: External service providing resources for machine learning experimentation.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

User (Actor)

Description: External user interacting with the HuskyAI system via the API Gateway.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

3rd party tools and ML libraries (Store)

Description: External third party tools for the services

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Gather Images Application (Python) (Process)

Description: This is a Python-based application responsible for gathering images from external sources, specifically Azure Cognitive Services, and storing them in the designated Training and Validation Images storage.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing of Azure Cognitive Services	Spoofing	Medium	Open		The Gather Images Application receives data from Azure Cognitive Services via HTTPS. While encrypted, there is no explicit authentication mechanism described to verify the identity of the external service, which could allow an attacker to spoof the service.	Implement certificate pinning or service-specific API key authentication to validate the identity of Azure Cognitive Services.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering of Image Data in Transit	Tampering	Low	Open		The flow from Azure Cognitive Services to the Gather Images Application is encrypted (HTTPS), reducing tampering risk. However, the application itself could be targeted to modify images before storage.	Ensure integrity checks (e.g., hashing) on downloaded images and validate data integrity before processing and storage.
	Information Disclosure via Engineer Access	Information Disclosure	Medium	Open		The Engineer accesses the Gather Images Application via VS Code (SSH). If SSH keys are compromised, sensitive application data or operations could be exposed.	Enforce strong passphrase protection for SSH keys, use multi-factor authentication, and regularly rotate keys.

Jupyter Notebook (Process)

Description: A Jupyter Notebook environment that processes the images stored in Training and Validation Images, executes code using external ML libraries, and provides a UI for engineers to interact with and manipulate data, allowing for iterative model development. It can save trained machine learning models to Machine Learning Model storage.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering of ML Models by Engineer	Tampering	High	Open		The Jupyter Notebook is accessed by the Engineer via VS Code (SSH) and can save models to storage. An authenticated but malicious insider could tamper with model training code or data.	Implement code review processes, version control for notebooks, and audit trails for model changes.
	Information Disclosure via Third-Party Libraries	Information Disclosure	Medium	Open		Jupyter Notebook imports third-party tools and ML libraries. Malicious or vulnerable libraries could leak sensitive training data or model information.	Use dependency scanning, pin library versions, and restrict outbound network access from the notebook environment.
	Denial of Service via Resource Exhaustion	Denial of Service	Medium	Open		The Jupyter Notebook processes large image datasets. An attacker or misconfiguration could cause resource exhaustion, disrupting model training.	Implement resource quotas, monitoring, and auto-scaling for the Jupyter environment.

Deployment (Process)

Description: Handles the deployment of the machine learning model by packaging the model and all necessary source code and configuration stored in Source Code and Configuration. It receives the final model from Jupyter Notebook and prepares it for deployment to the production environment.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering of Deployment Packages	Tampering	High	Open		The Deployment Service packages models and code from multiple sources. An attacker could tamper with these inputs or the deployment process itself, introducing malicious code into production.	Use cryptographic signing for deployment artifacts, verify checksums, and implement secure CI/CD pipelines with access controls.
	Elevation of Privilege via Bastion Access	Elevation of Privilege	High	Open		The Deployment Service communicates with the Bastion via SSH. If the Bastion is compromised, an attacker could gain elevated access to the deployment process and production resources.	Enforce least privilege for Bastion access, use jump host auditing, and segment network access to the Deployment Service.

Training and Validation Images (Store)

Description: Contains images used for training and validation of machine learning models.
Data set: Training and Validation Images
Contains images used for training and validation of machine learning models.
Record count maximum of 100000 with data sensitivity of biz and access control methods of rbac

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure of Training Data	Information Disclosure	High	Open		The Training and Validation Images store contains sensitive image data. While encrypted at rest, unauthorized access via misconfigured RBAC/ABAC could lead to data leakage.	Regularly audit access controls, encrypt data with customer-managed keys, and monitor access logs for anomalies.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering of Training Dataset	Tampering	High	Open		An attacker with write access to the image store could poison the training dataset, compromising model integrity.	Implement immutable storage for baseline datasets, use versioning, and checksum validation for data integrity.

API Key (Store)

Description: Stores API keys for secure access to external services.
Data set: API Keys
Stores API keys for secure access to external services.
Record count maximum of 20 with data sensitivity of cred and access control methods of rbac

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure of API Keys	Information Disclosure	High	Open		The API Key storage holds credentials for external services. Unauthorized access could lead to key leakage and abuse of connected services.	Use a dedicated secrets management service, encrypt keys at rest and in transit, and regularly rotate keys.
	Spoofing via Stolen API Keys	Spoofing	High	Open		If API keys are leaked, an attacker could spoof the Gather Images Application to Azure Cognitive Services or other external services.	Implement strict IP whitelisting and usage quotas on API keys, and monitor for anomalous usage.

Machine Learning Model (Store)

Description: Contains the machine learning models in serialized format.
Data set: Bastion Logs
Contains trained machine learning models in serialized format for production use.
Record count maximum of 5000 with data sensitivity of biz and access control methods of acl

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering of Machine Learning Models	Tampering	High	Open		The Machine Learning Model store contains serialized models. Unauthorized modifications could introduce backdoors or bias into production models.	Use model signing, version control, and access audits to prevent unauthorized changes.
	Information Disclosure of Model Weights	Information Disclosure	Medium	Open		ML models may contain sensitive information about the training data. Unauthorized access could lead to intellectual property theft or data inference attacks.	Encrypt models at rest, control access with RBAC, and consider model obfuscation techniques.

Source Code and Configuration (Store)

Description: Stores source code and configuration files for deployment and production setup.
Data set: Source Code and Configuration
Stores source code and configuration files for deployment and production setup.
Record count maximum of 200 with data sensitivity of biz and access control methods of rbac

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure of Source Code	Information Disclosure	High	Open		The Source Code and Configuration store contains proprietary code and settings. A breach could expose intellectual property and system configuration details.	Use encrypted repositories, access controls, and regular security scans for code storage.
	Tampering of Deployment Configuration	Tampering	High	Open		Malicious changes to configuration files could alter deployment behavior, leading to security weaknesses or system compromise.	Implement GitOps practices, code review, and integrity checks for configuration changes.

Simple Python Web Server (Process)

Description: Serves as simple web server

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Denial of Service via Public API	Denial of Service	Medium	Open		The Simple Python Web Server is exposed to users via the API Gateway. It could be targeted with high-volume requests to exhaust resources.	Implement rate limiting, DDoS protection, and auto-scaling in the production zone.
	Tampering of Web Server Logic	Tampering	High	Open		The web server loads models and code from storage. An attacker with access to the Bastion or deployment process could modify server logic.	Use immutable deployments, integrity checks on loaded artifacts, and runtime protection.
	Information Disclosure via Model Inference	Information Disclosure	Medium	Open		The web server processes user inputs for model inference. Improper input handling could leak model details or training data patterns.	Sanitize inputs, implement output filtering, and monitor for anomalous query patterns.

API Gateway (Process)

Description: Serves as the entry point for external users to interact with the production environment via HTTPS. It routes user requests to the Simple Python Web Server and ensures secure communication. The API Gateway enforces request validation and manages APIs exposed to the public while ensuring access control to internal services.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing of User Requests	Spoofing	Medium	Open		The API Gateway accepts HTTPS requests from external users. Without strong authentication, attackers could spoof user identities to access services.	Implement API keys, OAuth, or client certificate authentication for all incoming requests.
	Tampering of API Traffic	Tampering	Medium	Open		While HTTPS is used, the API Gateway itself could be compromised to modify requests or responses between users and the web server.	Use WAF, input validation, and secure API gateway configurations to prevent tampering.
	Denial of Service at Entry Point	Denial of Service	High	Open		As the public entry point, the API Gateway is vulnerable to DDoS attacks that could make the service unavailable.	Deploy DDoS protection services, rate limiting, and traffic shaping at the gateway.

Bastion (Process)

Description: A secure access management component for administrative functions. It provides controlled SSH access for the Infrastructure Admin to internal production resources, such as the Stored Machine Learning Model and Simple Python Web Server.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Elevation of Privilege via SSH Compromise	Elevation of Privilege	High	Open		The Bastion provides SSH access for admins and deployment. If compromised, it grants extensive access to production resources, including model storage and web services.	Enforce MFA for SSH, use jump host isolation, and regularly audit access logs.
	Spoofing of Admin Connections	Spoofing	High	Open		An attacker could spoof the Infrastructure Admin's connection to the Bastion if SSH keys are stolen or weak credentials are used.	Require strong passphrases for SSH keys, use certificate-based authentication, and monitor for unusual login patterns.
	Tampering of Production Resources	Tampering	High	Open		The Bastion can update the API Gateway, Web Server, and ML models. A malicious insider or compromised bastion could tamper with production systems.	Implement change control, multi-person approval for critical changes, and immutable infrastructure where possible.

Authorized Keys (Store)

Description: Contains SSH keys used for securing administrative access.
Data set: Authorized Keys
Contains SSH keys used for securing administrative access.
Record count maximum of 100 with data sensitivity of cred and access control methods of rbac

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure of SSH Keys	Information Disclosure	High	Open		The Authorized Keys store contains SSH keys for administrative access. Unauthorized disclosure could lead to full system compromise.	Use a dedicated secrets manager, encrypt keys at rest, and enforce regular key rotation.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing via Stolen SSH Keys	Spoofing	High	Open		Compromised SSH keys could allow an attacker to spoof authorized administrators and gain access to the Bastion and production systems.	Enforce key expiration, use hardware security modules (HSMs) for key storage, and monitor for unauthorized key usage.

Stored Machine Learning Model (Store)

Description: Contains storage for machine learning models in serialized format.
Data set: Stored Machine Learning Models
Contains trained machine learning models in serialized format for production use.
Record count maximum of 10 with data sensitivity of biz and access control methods of rbac

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering of Production Models	Tampering	High	Open		The Stored Machine Learning Model in production is not encrypted at rest (isEncrypted: false). An attacker with storage access could tamper with models, affecting all predictions.	Enable encryption at rest for production model storage, use access controls, and implement model integrity checks.
	Information Disclosure of Model Artifacts	Information Disclosure	Medium	Open		Production models may contain sensitive information. Unencrypted storage increases the risk of intellectual property theft if the storage is breached.	Encrypt all model files at rest and in transit, and restrict access using network security groups and RBAC.

HTTPS (Data Flow)

Description: Transfer data from Azure Cognitive Services to Gather Images Application in Python.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via Man-in-the-Middle	Information Disclosure	Low	Open		The HTTPS flow from Azure Cognitive Services to Gather Images Application is encrypted, but misconfigured TLS or certificate validation could allow eavesdropping.	Enforce TLS 1.2+, validate server certificates, and use certificate pinning for critical external services.

imports (Data Flow)

Description: Transfer data from Third Party tools and ML libraries to Gather Images Application in Python.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

imports (Data Flow)

Description: Transfer data from Third Party tools and ML libraries to Jupyter Notebook.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

VS Code (SSH) (Data Flow)

Description: Transfer data from Engineer to Gather Images Application in Python.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing of Engineer Identity	Spoofing	Medium	Open		The VS Code (SSH) flow from Engineer to Gather Images Application relies on SSH keys. If keys are compromised, an attacker could spoof the engineer's identity.	Enforce MFA for SSH, use strong passphrases, and regularly rotate SSH keys.

VS Code (SSH) (Data Flow)

Description: Transfer code and ML models from Engineer locally to Jupyter Notebook.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Repudiation of Model Changes	Repudiation	Medium	Open		The Engineer can transfer code and models to Jupyter Notebook via SSH. Without proper auditing, the engineer could deny making harmful changes.	Implement detailed audit logs for all SSH sessions and model modifications, with non-repudiation mechanisms.

stores (Data Flow)

Description: Transfer images from Gather Images Application to Training and Validation Images.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering of Stored Images	Tampering	Medium	Open		The 'stores' flow from Gather Images Application to Training and Validation Images is unencrypted (isEncrypted: false). An attacker on the network could tamper with images during storage.	Use encrypted connections (e.g., HTTPS) for all data transfers to storage, and validate data integrity upon write/read.

loads (Data Flow)

Description: API Key Storage to Gather Images Application in Python.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure of API Keys in Transit	Information Disclosure	High	Open		The 'loads' flow from API Key Storage to Gather Images Application is unencrypted (isEncrypted: false). API keys could be intercepted if transmitted over an insecure channel.	Ensure all secrets are transmitted over encrypted channels (e.g., TLS) and use secure protocols for secrets retrieval.

processes (Data Flow)

Description: Load from Training and Validation Images to Jupyter Notebook.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

package (Data Flow)

Description: Transfer data from Machine Learning Model to Deployment.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

save.h5 (Data Flow)

Description: Transfer final model from Jupyter Notebook to Machine Learning Model.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering of Model Serialization	Tampering	High	Open		The 'save.h5' flow from Jupyter Notebook to Machine Learning Model store is unencrypted. An attacker could intercept and modify the model during serialization or transfer.	Encrypt model files during transfer, use secure protocols, and verify checksums upon storage.

package (Data Flow)

Description: Transfer from Machine Learning Model Blob to Deployment Service.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

package (Data Flow)

Description: Transfer data from Source Code and Configuration to Deployment.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

HTTPS (Data Flow)

Description: Transfer from User to API Gateway.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing of External Users	Spoofing	Medium	Open		The HTTPS flow from User to API Gateway lacks described authentication. Attackers could spoof user identities to access services.	Implement strong user authentication (e.g., OAuth 2.0) and rate limiting to prevent abuse.
	Denial of Service from External Users	Denial of Service	High	Open		The public-facing API Gateway is vulnerable to DoS attacks from malicious users overwhelming the service with requests.	Deploy DDoS protection, rate limiting, and auto-scaling to mitigate impact.

update (Data Flow)

Description: Transfer data from Bastion to API Gateway.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

HTTPS (Data Flow)

Description: Transfer data from API Gateway to Simple Python Web Server.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via Internal API	Information Disclosure	Medium	Open		The HTTPS flow from API Gateway to Simple Python Web Server is unencrypted (isEncrypted: false). Sensitive data or model inferences could be intercepted on the internal network.	Encrypt all internal traffic using TLS or VPNs to protect against eavesdropping.

update (Data Flow)

Description: Transfer data from Bastion to Simple Python Web Server.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

loads (Data Flow)

Description: Transfer sensitive data from Stored Machine Learning Model to Simple Python Web Server.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering of Model Loading	Tampering	High	Open		The 'loads' flow from Stored Machine Learning Model to Simple Python Web Server is unencrypted. An attacker could tamper with the model as it is loaded into the web server.	Use encrypted storage and secure transfer protocols, and verify model integrity checksums upon loading.

SSH (Data Flow)

Description: Transfer sensitive data from Deployment Service to Bastion

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Elevation of Privilege via Deployment Channel	Elevation of Privilege	High	Open		The SSH flow from Deployment Service to Bastion provides a pathway for privilege escalation if the deployment process is compromised.	Restrict SSH access to the Bastion, use network segmentation, and monitor for unusual deployment activities.

update (Data Flow)

Description: Transfer sensitive data from Bastion to Stored Machine Learning Model.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

SSH (Data Flow)

Description: Transfer data from Infrastructure Admin to Bastion.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing of Infrastructure Admin	Spoofing	High	Open		The SSH flow from Infrastructure Admin to Bastion relies on key-based authentication. Stolen keys could allow an attacker to spoof the admin and gain full system access.	Enforce MFA, use jump hosts with strict access controls, and regularly audit administrative access.

update (Data Flow)

Description: Transfer sensitive data from Bastion to Stored Machine Learning Model.

Number	Title	Type	Severity	Status	Score	Description	Mitigations

(Data Flow)

Description: Transfer sensitive data from Authorized Keys Storage to Bastion.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure of SSH Keys in Transit	Information Disclosure	High	Open		The flow from Authorized Keys Storage to Bastion is encrypted (SSH), but if the storage or retrieval process is insecure, keys could be leaked.	Use a secure secrets management service with encryption in transit and at rest, and audit all access to keys.

imports (Data Flow)

Description: Transfer data from Third Party tools and ML libraries to Simple Python Web Server.

Number	Title	Type	Severity	Status	Score	Description	Mitigations