Husky AI

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

Executive Summary

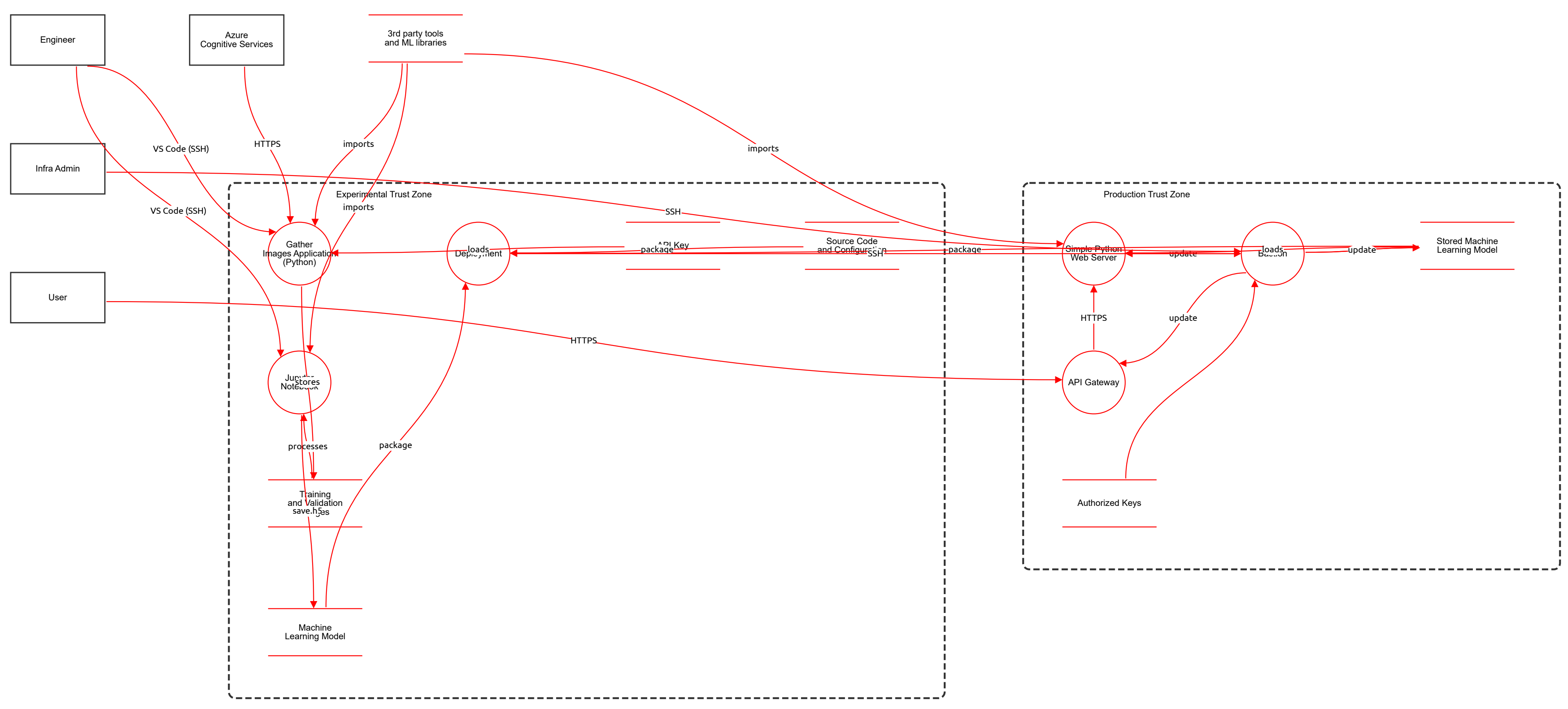
High level system description

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	85
Total Mitigated	0
Total Open	85
Open / Critical Severity	0
Open / High Severity	59
Open / Medium Severity	25
Open / Low Severity	1

Husky AI


Engineer
Azure Cognitive Services
3rd party tools and ML libraries
Infra Admin
User
VS Code (SSH)
HTTPS
imports
imports
VS Code (SSH)
imports
Experimental Trust Zone
Gather Images Application (Python)
Deployment
loads
SSH
API Key
package
Source Code and Configuration
SSH
package
Jupyter Notebook
stores
HTTPS
processes
package
Training and Validation Images
save.h5
Machine Learning Model
Production Trust Zone
Simple Python Web Server
update
Bastion
loads
update
Stored Machine Learning Model
HTTPS
update
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
	Compromised Engineer Credentials Leading to Unauthorized Access	Spoofing	High	Open		The Engineer actor connects to Gather Images Application and Jupyter Notebook via SSH from outside the Experimental Trust Zone. If the engineer's SSH credentials or private keys are compromised, an attacker could impersonate the engineer and gain unauthorized access to development resources, training data, and machine learning models. This is a boundary-crossing ingress threat from an external actor into a trusted zone.	- Enforce SSH key-based authentication with passphrase-protected keys - Implement multi-factor authentication (MFA) for all SSH connections - Use certificate-based SSH authentication with short-lived certificates - Monitor and log all SSH access attempts - Implement IP allowlisting for SSH connections
	Insider Threat - Malicious Engineer Actions	Elevation of Privilege	Medium	Open		The Engineer has direct SSH access to critical development components (Gather Images Application and Jupyter Notebook) within the Experimental Trust Zone. A malicious or compromised engineer could abuse these privileges to exfiltrate training data, inject malicious code into models, or tamper with the ML pipeline. The engineer's position outside the trust boundary with ingress access creates an elevated risk.	- Implement least privilege access controls with RBAC - Enable comprehensive audit logging of all engineer actions - Implement code review and approval workflows - Use session recording for SSH connections - Implement data loss prevention (DLP) controls - Conduct regular access reviews and background checks

Infra Admin (Actor)

Description: Administrator responsible for securing and maintaining production infrastructure.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Compromised Admin Credentials Enabling Production System Takeover	Spoofing	High	Open		The Infrastructure Admin actor connects to the Bastion via SSH from outside the Production Trust Zone. If admin credentials are compromised, an attacker could gain full control over production infrastructure including the API Gateway, Simple Python Web Server, and Stored Machine Learning Model. This represents a critical boundary-crossing ingress threat into the production environment.	- Enforce SSH key-based authentication with hardware security keys (e.g., YubiKey) - Implement multi-factor authentication (MFA) with FIDO2/WebAuthn - Use certificate-based SSH authentication with short-lived certificates (1-hour validity) - Implement privileged access management (PAM) solution - Require just-in-time (JIT) access approval for production access - Monitor and alert on all admin access attempts

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Insider Threat - Malicious Admin Actions in Production	Elevation of Privilege	High	Open		The Infrastructure Admin has privileged SSH access through Bastion to all production components including API Gateway, Simple Python Web Server, and Stored Machine Learning Model. A malicious admin could abuse these privileges to exfiltrate production data, inject backdoors, modify ML models, or cause service disruption. The admin's external position with full production ingress access creates critical risk.	- Implement separation of duties with multiple approval requirements - Enable comprehensive audit logging with immutable log storage - Use session recording and real-time monitoring for all admin actions - Implement anomaly detection for admin behavior - Require peer review for production changes - Conduct regular access reviews and background checks - Implement break-glass procedures with automatic alerting

Azure Cognitive Services (Actor)

Description: External service providing resources for machine learning experimentation.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Malicious or Poisoned Data from External Service	Tampering	Medium	Open		Azure Cognitive Services is an external actor outside all trust boundaries that provides images to the Gather Images Application via HTTPS. If the external service is compromised or serves malicious content, poisoned training data could be injected into the system, leading to model poisoning attacks. This is an ingress flow from an untrusted external source crossing into the Experimental Trust Zone.	- Implement content validation and sanitization for all incoming images - Use image format verification and malware scanning - Implement rate limiting and anomaly detection for data ingestion - Validate image checksums and digital signatures where available - Implement data provenance tracking - Use separate validation datasets from trusted sources - Monitor model performance for signs of poisoning
	Service Availability Dependency Risk	Denial of Service	Low	Open		The Gather Images Application depends on Azure Cognitive Services for image retrieval. If the external service experiences outages, rate limiting, or becomes unavailable, the data gathering process will fail, impacting the ability to train and update models. This represents a dependency on an external actor outside the trust boundary.	- Implement retry logic with exponential backoff - Cache previously retrieved images - Use multiple data sources for redundancy - Implement circuit breaker patterns - Monitor service availability and set up alerts - Maintain offline datasets for critical training scenarios

User (Actor)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Malicious User Input Attacks via API Gateway	Tampering	High	Open		The User actor is external to the Production Trust Zone and sends HTTPS requests to the API Gateway. Malicious users could attempt injection attacks, send malformed requests, or exploit API vulnerabilities to compromise the system. This is a boundary-crossing ingress flow from an untrusted external source into the production environment.	- Implement comprehensive input validation and sanitization - Use Web Application Firewall (WAF) with OWASP ruleset - Implement rate limiting and request throttling - Use API schema validation (OpenAPI/Swagger) - Implement request size limits - Deploy DDoS protection - Use API gateway security features (authentication, authorization) - Implement CAPTCHA for suspicious traffic patterns
	Denial of Service via Resource Exhaustion	Denial of Service	Medium	Open		External users can send unlimited requests to the API Gateway, potentially overwhelming the Simple Python Web Server and causing service disruption. The User is positioned outside the Production Trust Zone with direct ingress access to the API Gateway, creating a DoS attack surface.	- Implement rate limiting per IP address and API key - Use API gateway throttling policies - Deploy DDoS protection services - Implement request queuing and load balancing - Set up auto-scaling for backend services - Monitor resource utilization and set alerts - Implement circuit breakers to protect backend services
	Unauthorized Access Attempts	Spoofing	Medium	Open		External users may attempt to access the API Gateway without proper authentication or authorization. The User actor is outside the Production Trust Zone, and the HTTPS flow represents a boundary-crossing ingress point that must be properly secured to prevent unauthorized access.	- Implement strong authentication mechanisms (OAuth 2.0, JWT) - Require API keys or tokens for all requests - Implement authorization checks at API Gateway level - Use mutual TLS (mTLS) for sensitive operations - Implement account lockout policies - Monitor and alert on failed authentication attempts - Implement IP-based access controls where appropriate

3rd party tools and ML libraries (Store)

Description: External third party tools for the services

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Supply Chain Attack via Compromised Third-Party Libraries	Tampering	High	Open		Third-party tools and ML libraries are external to all trust boundaries and are imported by Gather Images Application, Jupyter Notebook, and Simple Python Web Server. Compromised or malicious libraries could inject backdoors, exfiltrate data, or compromise the ML pipeline. This represents multiple ingress flows from an untrusted external source into both Experimental and Production Trust Zones.	- Implement software composition analysis (SCA) tools - Use dependency scanning and vulnerability management - Pin specific versions of all dependencies - Verify package checksums and signatures - Use private package repositories with vetted libraries - Implement automated security scanning in CI/CD pipeline - Regularly update dependencies and monitor security advisories - Use container image scanning for production deployments
	Malicious Code Execution via Vulnerable Dependencies	Elevation of Privilege	High	Open		Vulnerable third-party libraries imported into Gather Images Application, Jupyter Notebook, and Simple Python Web Server could be exploited to gain elevated privileges, execute arbitrary code, or compromise the system. The external position of this data store with imports crossing into both trust zones creates significant risk.	- Implement continuous vulnerability scanning - Use automated dependency update tools (e.g., Dependabot) - Implement runtime application self-protection (RASP) - Use least privilege execution contexts - Implement sandboxing for untrusted code execution - Monitor for suspicious library behavior - Maintain an inventory of all dependencies with risk assessment

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
	API Key Exposure in Process Memory or Logs	Information Disclosure	High	Open		The Gather Images Application loads API keys from API Key Storage and uses them to authenticate with Azure Cognitive Services. If the process memory is dumped, logs are improperly configured, or debugging is enabled, API keys could be exposed. This process is within the Experimental Trust Zone but handles sensitive credentials.	- Use secure credential management libraries that prevent memory dumps - Implement secrets rotation policies - Avoid logging sensitive data - Use environment variables or secure vaults for credential storage - Implement memory encryption for sensitive data - Disable debug logging in production-like environments - Use short-lived tokens instead of long-term API keys where possible

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Unauthorized Modification of Image Gathering Logic	Tampering	Medium	Open		Engineers with SSH access to the Gather Images Application could modify the code to gather inappropriate images, bypass validation, or exfiltrate API keys. The process receives ingress SSH connections from external engineers crossing into the Experimental Trust Zone.	- Implement code integrity monitoring - Use version control with mandatory code review - Implement file integrity monitoring (FIM) - Use read-only file systems where possible - Implement change management processes - Enable comprehensive audit logging of code changes - Use infrastructure as code (IaC) with approval workflows
	Insufficient Validation of Retrieved Images	Tampering	Medium	Open		The Gather Images Application receives images from external Azure Cognitive Services via HTTPS and stores them in Training and Validation Images. Insufficient validation could allow malicious or corrupted images to enter the training pipeline, leading to model poisoning or system compromise.	- Implement comprehensive image validation (format, size, content) - Use malware scanning for all incoming files - Implement content-based filtering - Validate image metadata and remove EXIF data - Implement checksums and integrity verification - Use sandboxed image processing - Implement anomaly detection for image characteristics

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
	Arbitrary Code Execution via Notebook Interface	Elevation of Privilege	High	Open		Jupyter Notebook allows engineers to execute arbitrary Python code with SSH access from outside the Experimental Trust Zone. Malicious or compromised engineers could execute code to escalate privileges, access sensitive data, or compromise the system. The notebook has access to Training and Validation Images and can save models to Machine Learning Model storage.	- Implement kernel isolation and sandboxing - Use containerization with resource limits - Implement least privilege execution contexts - Enable comprehensive audit logging of all executed code - Implement code review for production-bound notebooks - Use JupyterHub with authentication and authorization - Restrict network access from notebook environment - Implement data access controls at storage level

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Model Poisoning via Malicious Training Code	Tampering	High	Open		Engineers with access to Jupyter Notebook could intentionally or unintentionally inject malicious training logic that creates backdoored models, introduces biases, or reduces model accuracy. The notebook processes training data and saves models that will be deployed to production, creating a critical attack path from experimental to production zones.	- Implement model validation and testing pipelines - Use automated model performance monitoring - Implement model versioning and rollback capabilities - Require peer review for model training code - Implement model explainability and interpretability tools - Use differential privacy techniques - Implement anomaly detection for model behavior - Separate training and deployment environments with approval gates
	Data Exfiltration via Notebook Execution	Information Disclosure	High	Open		Jupyter Notebook has access to Training and Validation Images containing potentially sensitive data. Engineers could use the notebook to exfiltrate training data, API keys, or other sensitive information. The notebook receives ingress SSH connections from external engineers and has access to multiple data stores within the Experimental Trust Zone.	- Implement data loss prevention (DLP) controls - Restrict outbound network access from notebook environment - Monitor and log all data access patterns - Implement data classification and access controls - Use network segmentation to isolate notebook environment - Implement egress filtering and monitoring - Enable session recording for SSH connections - Implement anomaly detection for data access patterns
	Vulnerable Third-Party Library Exploitation	Elevation of Privilege	High	Open		Jupyter Notebook imports third-party ML libraries from external sources. Vulnerable or malicious libraries could be exploited to gain elevated privileges, execute arbitrary code, or compromise the notebook environment. This represents an ingress flow from untrusted external sources into the Experimental Trust Zone.	- Implement software composition analysis (SCA) - Use virtual environments with pinned dependencies - Implement automated vulnerability scanning - Use private package repositories with vetted libraries - Implement runtime monitoring for suspicious library behavior - Use container scanning for notebook images - Regularly update dependencies with security patches - Implement least privilege execution for notebook kernels

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
	Unauthorized Model Deployment to Production	Tampering	High	Open		The Deployment service packages models from Machine Learning Model storage and source code from Source Code and Configuration, then deploys to production via SSH to Bastion. Insufficient access controls could allow unauthorized or malicious model deployments. This process bridges the Experimental and Production Trust Zones, creating a critical control point.	- Implement multi-person approval workflows for deployments - Use automated model validation and testing gates - Implement deployment pipelines with security scanning - Require digital signatures for deployment artifacts - Implement rollback capabilities - Enable comprehensive audit logging of all deployments - Use infrastructure as code (IaC) with version control - Implement canary deployments and gradual rollouts
	Malicious Code Injection During Packaging	Tampering	High	Open		The Deployment service packages source code and configuration from storage within the Experimental Trust Zone. If the deployment process is compromised, malicious code could be injected during packaging before deployment to production. This represents a critical attack path between trust zones.	- Implement code signing and verification - Use immutable build artifacts - Implement build process isolation and sandboxing - Use secure CI/CD pipelines with security scanning - Implement integrity checks for all packaged components - Use container image signing and verification - Enable comprehensive audit logging of build and package processes - Implement supply chain security controls
	Compromised Deployment Credentials	Spoofing	High	Open		The Deployment service uses SSH to connect to Bastion in the Production Trust Zone. If deployment credentials are compromised, attackers could deploy malicious models or code to production. This is a boundary-crossing flow from Experimental to Production zones.	- Use certificate-based SSH authentication with short-lived certificates - Implement secrets management with automatic rotation - Use service accounts with least privilege - Implement just-in-time (JIT) credential provisioning - Monitor and alert on all deployment activities - Use hardware security modules (HSM) for key storage - Implement anomaly detection for deployment patterns

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
	Unauthorized Access to Training Data	Information Disclosure	High	Open		Training and Validation Images storage contains up to 100,000 images with business sensitivity. Despite RBAC controls, misconfigured permissions or compromised credentials could allow unauthorized access. The storage is within the Experimental Trust Zone and is accessed by both Gather Images Application and Jupyter Notebook.	- Implement least privilege access with RBAC and ABAC - Enable encryption at rest with customer-managed keys - Implement access logging and monitoring - Use private endpoints and disable public access - Implement data classification and labeling - Conduct regular access reviews - Use Azure Private Link for secure access - Implement conditional access policies
	Data Tampering in Training Dataset	Tampering	High	Open		Training and Validation Images are written by Gather Images Application and read by Jupyter Notebook. Unauthorized modification of training data could lead to model poisoning, biased models, or compromised ML performance. The storage is encrypted but lacks signing verification.	- Implement versioning and immutability for training data - Use digital signatures or checksums for data integrity - Enable audit logging for all data modifications - Implement access controls with write restrictions - Use append-only storage where appropriate - Implement data validation pipelines - Monitor for unauthorized data changes - Implement backup and recovery procedures
	Data Exfiltration via Compromised Access	Information Disclosure	Medium	Open		Engineers with access to Jupyter Notebook or Gather Images Application could exfiltrate training images. The storage contains business-sensitive data with RBAC controls, but the data is accessible to multiple processes within the Experimental Trust Zone.	- Implement data loss prevention (DLP) controls - Enable comprehensive access logging and monitoring - Implement egress filtering and monitoring - Use Azure Storage Analytics for access patterns - Implement anomaly detection for data access - Restrict network access to storage - Use Azure Private Link - Implement data watermarking for tracking

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
	API Key Theft via Unauthorized Access	Information Disclosure	High	Open		API Key storage contains up to 20 credential-sensitive API keys with RBAC controls. Compromised access could expose keys used to authenticate with Azure Cognitive Services. The storage is within the Experimental Trust Zone and is accessed by Gather Images Application.	- Implement strict RBAC with least privilege - Enable encryption at rest with customer-managed keys - Use Azure Key Vault or equivalent secrets management - Implement access logging and monitoring with alerts - Use private endpoints and disable public access - Implement secrets rotation policies - Use managed identities where possible instead of API keys - Implement conditional access policies - Enable soft delete and purge protection
	Insufficient Secrets Rotation	Information Disclosure	Medium	Open		API keys stored in this storage may not be rotated regularly, increasing the risk of compromise over time. Long-lived credentials provide attackers with extended windows of opportunity if keys are exposed.	- Implement automated secrets rotation policies (e.g., 90-day rotation) - Use short-lived tokens instead of long-term API keys where possible - Implement secrets expiration monitoring and alerts - Use Azure Key Vault rotation features - Implement zero-trust principles with continuous verification - Monitor for use of old or expired credentials
	Privilege Escalation via Key Access	Elevation of Privilege	Medium	Open		If an attacker gains access to API keys in this storage, they could use those keys to access Azure Cognitive Services with the same privileges as the legitimate application, potentially accessing additional data or services beyond the intended scope.	- Implement principle of least privilege for API keys - Use scoped API keys with minimal permissions - Implement API key usage monitoring and anomaly detection - Use separate keys for different environments - Implement rate limiting on API key usage - Monitor for unusual API access patterns - Implement IP allowlisting for API key usage where possible

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
	Model Theft via Unauthorized Access	Information Disclosure	High	Open		Machine Learning Model storage contains up to 5,000 trained models with business sensitivity and ACL controls. Unauthorized access could allow competitors or attackers to steal proprietary ML models. The storage is within the Experimental Trust Zone and receives models from Jupyter Notebook and provides them to Deployment service.	- Implement strict access controls with ACL and RBAC - Enable encryption at rest with customer-managed keys - Implement access logging and monitoring - Use private endpoints and disable public access - Implement model watermarking for tracking - Use Azure Private Link for secure access - Conduct regular access reviews - Implement data classification and labeling
	Model Tampering Before Deployment	Tampering	High	Open		Machine Learning Model storage is written by Jupyter Notebook and read by Deployment service. Unauthorized modification of models before deployment could introduce backdoors, biases, or malicious behavior in production. The storage is encrypted but lacks signing verification.	- Implement model versioning and immutability - Use digital signatures for model integrity verification - Enable audit logging for all model modifications - Implement access controls with write restrictions - Use append-only storage where appropriate - Implement model validation pipelines before deployment - Monitor for unauthorized model changes - Implement backup and recovery procedures
	Model Poisoning via Malicious Model Upload	Tampering	High	Open		Engineers with access to Jupyter Notebook could save malicious or backdoored models to this storage. These poisoned models could then be deployed to production, compromising the entire ML system. This represents a critical attack path from development to production.	- Implement automated model validation and testing - Use model performance benchmarking - Implement model explainability and interpretability checks - Require peer review for model deployments - Implement anomaly detection for model behavior - Use differential privacy techniques - Implement A/B testing before full deployment - Separate model training and deployment with approval gates

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
	Source Code Theft via Unauthorized Access	Information Disclosure	High	Open		Source Code and Configuration storage contains up to 200 files with business sensitivity and RBAC controls. Unauthorized access could expose proprietary algorithms, configurations, and intellectual property. The storage is within the Experimental Trust Zone and is accessed by Deployment service.	- Implement strict RBAC with least privilege - Enable encryption at rest with customer-managed keys - Implement access logging and monitoring - Use private endpoints and disable public access - Implement code obfuscation for sensitive algorithms - Use Azure Private Link for secure access - Conduct regular access reviews - Implement data classification and labeling
	Malicious Code Injection in Configuration	Tampering	High	Open		Source Code and Configuration storage is read by Deployment service for packaging and deployment to production. Unauthorized modification of code or configuration could inject backdoors, vulnerabilities, or malicious logic into production systems. The storage is encrypted but lacks signing verification.	- Implement version control with mandatory code review - Use digital signatures for code integrity verification - Enable audit logging for all modifications - Implement access controls with write restrictions - Use immutable infrastructure patterns - Implement code scanning and security analysis - Monitor for unauthorized changes - Implement backup and recovery procedures - Use infrastructure as code (IaC) with approval workflows
	Secrets Exposure in Configuration Files	Information Disclosure	High	Open		Configuration files in this storage may contain embedded secrets, API keys, or credentials. If access controls are misconfigured or compromised, these secrets could be exposed. The storage contains business-sensitive data with RBAC controls.	- Never store secrets in configuration files - Use Azure Key Vault or secrets management solutions - Implement secrets scanning in CI/CD pipelines - Use environment variables or managed identities - Implement configuration encryption - Conduct regular security audits of configuration files - Use secret detection tools in version control - Implement automated secrets rotation

Simple Python Web Server (Process)

Description: Serves as simple web server

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Model Inference Manipulation via Input Tampering	Tampering	High	Open		Simple Python Web Server receives HTTPS requests from API Gateway and loads models from Stored Machine Learning Model. Malicious users could craft adversarial inputs designed to manipulate model predictions or cause incorrect classifications. The server is within the Production Trust Zone but receives ingress traffic from external users via API Gateway.	- Implement comprehensive input validation and sanitization - Use adversarial training techniques - Implement input anomaly detection - Use rate limiting per user/IP - Implement model robustness testing - Use ensemble models for critical predictions - Implement confidence thresholds for predictions - Monitor for adversarial attack patterns
	Vulnerable Third-Party Library Exploitation	Elevation of Privilege	High	Open		Simple Python Web Server imports third-party tools and ML libraries from external sources. Vulnerable libraries could be exploited to gain elevated privileges, execute arbitrary code, or compromise the production server. This represents an ingress flow from untrusted external sources into the Production Trust Zone.	- Implement software composition analysis (SCA) - Use automated vulnerability scanning - Pin specific versions of all dependencies - Use private package repositories with vetted libraries - Implement runtime application self-protection (RASP) - Use container scanning for production images - Regularly update dependencies with security patches - Implement least privilege execution contexts
	Unauthorized Administrative Access via Bastion	Elevation of Privilege	High	Open		Simple Python Web Server receives update commands from Bastion within the Production Trust Zone. If Bastion is compromised or admin credentials are stolen, attackers could gain full control over the web server, modify its behavior, or exfiltrate data.	- Implement strict access controls for Bastion access - Use certificate-based SSH authentication - Implement multi-factor authentication (MFA) - Enable comprehensive audit logging of all admin actions - Implement session recording - Use just-in-time (JIT) access provisioning - Implement anomaly detection for admin behavior - Use immutable infrastructure with automated deployments

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Model Extraction via Repeated Queries	Information Disclosure	Medium	Open		External users can query the Simple Python Web Server through API Gateway to get model predictions. Attackers could use repeated queries with carefully crafted inputs to extract or reverse-engineer the underlying ML model. The server is within the Production Trust Zone but exposed to external users.	- Implement rate limiting per user/IP - Use query complexity analysis - Implement prediction confidence obfuscation - Monitor for model extraction attack patterns - Use differential privacy techniques - Implement query auditing and anomaly detection - Limit the precision of returned predictions - Implement CAPTCHA for suspicious query patterns
	Denial of Service via Resource Exhaustion	Denial of Service	Medium	Open		Simple Python Web Server processes inference requests from API Gateway. Computationally expensive model inference operations could be exploited to exhaust server resources, causing service disruption. The server receives ingress traffic from external users via API Gateway.	- Implement request queuing and load balancing - Use auto-scaling based on resource utilization - Implement timeout limits for inference operations - Use resource quotas and limits - Implement circuit breakers - Monitor resource utilization and set alerts - Use asynchronous processing for heavy workloads - Implement request prioritization

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
	API Gateway Bypass via Direct Access	Spoofing	High	Open		API Gateway serves as the entry point for external users into the Production Trust Zone. If the Simple Python Web Server is accessible directly without going through the API Gateway, attackers could bypass authentication, authorization, and rate limiting controls. The gateway is positioned at the trust boundary.	- Implement network segmentation to prevent direct access - Use security groups and network ACLs to restrict access - Ensure backend services only accept traffic from API Gateway - Implement mutual TLS (mTLS) between gateway and backend - Use private endpoints for backend services - Monitor for direct access attempts - Implement IP allowlisting at backend services

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Insufficient Authentication and Authorization	Spoofing	High	Open		API Gateway receives HTTPS requests from external users crossing into the Production Trust Zone. Without proper authentication and authorization, unauthorized users could access production services. The gateway is the primary ingress point and must enforce security controls.	- Implement OAuth 2.0 or OpenID Connect authentication - Use API keys or JWT tokens for authorization - Implement role-based access control (RBAC) - Use multi-factor authentication (MFA) for sensitive operations - Implement rate limiting per authenticated user - Monitor and alert on authentication failures - Implement account lockout policies - Use Azure AD or equivalent identity provider
	API Gateway Configuration Tampering	Tampering	High	Open		API Gateway receives update commands from Bastion. If admin access is compromised, attackers could modify gateway configurations to disable security controls, redirect traffic, or expose internal services. The gateway is within the Production Trust Zone and is a critical security control point.	- Implement strict access controls for gateway configuration - Use infrastructure as code (IaC) with version control - Enable audit logging for all configuration changes - Implement change approval workflows - Use immutable infrastructure patterns - Monitor for unauthorized configuration changes - Implement configuration backup and recovery - Use Azure Policy or equivalent for compliance enforcement
	DDoS Attack on API Gateway	Denial of Service	High	Open		API Gateway is the public-facing entry point receiving HTTPS traffic from external users. It is vulnerable to distributed denial of service (DDoS) attacks that could overwhelm the gateway and prevent legitimate users from accessing the service. The gateway sits at the trust boundary between external users and the Production Trust Zone.	- Deploy DDoS protection services (e.g., Azure DDoS Protection) - Implement rate limiting and throttling - Use Web Application Firewall (WAF) - Implement geo-blocking for suspicious regions - Use CDN for traffic distribution - Implement auto-scaling for gateway instances - Monitor traffic patterns and set alerts - Implement CAPTCHA for suspicious traffic
	Insufficient Logging and Monitoring	Repudiation	Medium	Open		API Gateway is the primary ingress point for external traffic into the Production Trust Zone. Without comprehensive logging and monitoring, malicious activities, security incidents, or unauthorized access attempts may go undetected, and attackers could deny their actions.	- Enable comprehensive access logging for all requests - Implement centralized log management (SIEM) - Use immutable log storage - Implement real-time monitoring and alerting - Log authentication and authorization events - Implement log retention policies - Use Azure Monitor or equivalent - Implement anomaly detection on logs - Ensure logs include timestamps, source IPs, and user identities

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
	Bastion Host Compromise Leading to Full Production Access	Elevation of Privilege	High	Open		Bastion provides SSH access from Infrastructure Admin to all production components (API Gateway, Simple Python Web Server, Stored Machine Learning Model). If the Bastion host is compromised, attackers gain full administrative access to the entire Production Trust Zone. The Bastion receives ingress SSH from external admins crossing the trust boundary.	- Harden Bastion host with minimal software and services - Implement host-based intrusion detection (HIDS) - Use immutable infrastructure for Bastion - Implement regular security patching and updates - Use multi-factor authentication (MFA) for access - Enable comprehensive audit logging - Implement session recording - Use just-in-time (JIT) access provisioning - Implement network segmentation - Monitor for suspicious activities
	Unauthorized SSH Key Access	Information Disclosure	High	Open		Bastion loads authorized keys from Authorized Keys storage to authenticate admin SSH connections. If these keys are compromised or improperly managed, unauthorized users could gain administrative access to production systems. The Bastion is within the Production Trust Zone and provides privileged access.	- Use certificate-based SSH authentication with short-lived certificates - Implement strict access controls for key storage - Use hardware security modules (HSM) for key protection - Implement key rotation policies - Enable audit logging for key access - Use Azure Key Vault or equivalent - Implement key usage monitoring - Use passphrase-protected keys - Implement conditional access policies
	Lateral Movement from Bastion to Production Services	Elevation of Privilege	High	Open		Bastion has update access to API Gateway, Simple Python Web Server, and Stored Machine Learning Model within the Production Trust Zone. If Bastion is compromised, attackers could use it as a pivot point for lateral movement to compromise all production services.	- Implement network segmentation with micro-segmentation - Use least privilege access controls - Implement just-in-time (JIT) access provisioning - Enable comprehensive audit logging - Implement anomaly detection for lateral movement - Use zero-trust network architecture - Implement host-based firewalls - Monitor for suspicious network connections - Use separate credentials for each service

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Insufficient Session Monitoring and Recording	Repudiation	Medium	Open		Bastion provides privileged SSH access to production systems. Without comprehensive session recording and monitoring, malicious admin activities could go undetected, and admins could deny their actions. The Bastion is a critical control point for production access.	- Implement session recording for all SSH sessions - Use centralized log management (SIEM) - Enable real-time monitoring and alerting - Implement user behavior analytics (UBA) - Use immutable log storage - Implement log retention policies - Monitor for privilege escalation attempts - Implement anomaly detection for admin behavior - Ensure logs include timestamps, commands, and user identities

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
	SSH Key Theft via Unauthorized Access	Information Disclosure	High	Open		Authorized Keys storage contains up to 100 SSH keys with credential sensitivity and RBAC controls. Compromised access could expose keys used for administrative access to production systems via Bastion. The storage is within the Production Trust Zone and contains highly sensitive credentials.	- Implement strict RBAC with least privilege - Enable encryption at rest with customer-managed keys - Use Azure Key Vault or equivalent secrets management - Implement access logging and monitoring with alerts - Use private endpoints and disable public access - Implement key rotation policies - Use certificate-based authentication instead of long-lived keys - Implement conditional access policies - Enable soft delete and purge protection - Use hardware security modules (HSM) for key protection
	Insufficient Key Rotation	Information Disclosure	Medium	Open		SSH keys stored in Authorized Keys storage may not be rotated regularly, increasing the risk of compromise over time. Long-lived keys provide attackers with extended windows of opportunity if keys are exposed. The storage contains credential-sensitive data.	- Implement automated key rotation policies (e.g., 90-day rotation) - Use certificate-based SSH authentication with short-lived certificates - Implement key expiration monitoring and alerts - Use Azure Key Vault rotation features - Monitor for use of old or expired keys - Implement zero-trust principles with continuous verification - Use just-in-time (JIT) key provisioning

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Privilege Escalation via Key Compromise	Elevation of Privilege	High	Open		If an attacker gains access to SSH keys in Authorized Keys storage, they could use those keys to authenticate to Bastion and gain full administrative access to all production systems. This represents a critical privilege escalation path.	- Implement multi-factor authentication (MFA) in addition to SSH keys - Use certificate-based authentication with hardware tokens - Implement anomaly detection for key usage - Monitor for unusual SSH access patterns - Implement IP allowlisting for SSH access - Use just-in-time (JIT) access provisioning - Implement session monitoring and recording - Use separate keys for different privilege levels

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
	Production Model Theft via Unauthorized Access	Information Disclosure	High	Open		Stored Machine Learning Model contains up to 10 production models with business sensitivity and RBAC controls. Unauthorized access could allow competitors or attackers to steal proprietary production ML models. The storage is within the Production Trust Zone and is loaded by Simple Python Web Server. Note: The storage is not encrypted according to the model.	- Implement strict RBAC with least privilege - Enable encryption at rest with customer-managed keys - Implement access logging and monitoring - Use private endpoints and disable public access - Implement model watermarking for tracking - Use Azure Private Link for secure access - Conduct regular access reviews - Implement data classification and labeling - Enable Azure Storage encryption
	Production Model Tampering	Tampering	High	Open		Stored Machine Learning Model is updated by Bastion and loaded by Simple Python Web Server. Unauthorized modification of production models could introduce backdoors, biases, or malicious behavior affecting all users. The storage is within the Production Trust Zone and is not encrypted or signed.	- Implement model versioning and immutability - Use digital signatures for model integrity verification - Enable audit logging for all model modifications - Implement access controls with write restrictions - Use append-only storage where appropriate - Implement model validation before loading - Monitor for unauthorized model changes - Implement backup and recovery procedures - Enable encryption at rest and in transit

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Lack of Encryption for Production Models	Information Disclosure	High	Open		According to the model, Stored Machine Learning Model storage has isEncrypted set to false. This means production ML models are stored in plaintext, making them vulnerable to theft if storage access controls are bypassed or if physical media is compromised. The storage contains business-sensitive production models.	- Enable encryption at rest immediately with customer-managed keys - Use Azure Storage Service Encryption (SSE) - Implement encryption in transit (TLS 1.2+) - Use Azure Key Vault for key management - Implement key rotation policies - Conduct security audit to ensure encryption is enabled - Implement data classification and protection policies
	Model Rollback Attack	Tampering	Medium	Open		Attackers with access to Stored Machine Learning Model via Bastion could replace current production models with older, vulnerable versions. This could reintroduce previously fixed vulnerabilities or biases. The storage is updated by Bastion within the Production Trust Zone.	- Implement model versioning with immutability - Use digital signatures for model versions - Implement rollback approval workflows - Enable audit logging for model updates - Implement model validation before deployment - Monitor for unauthorized version changes - Use infrastructure as code (IaC) for model deployment - Implement automated model testing pipelines

HTTPS (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Man-in-the-Middle Attack on HTTPS Connection	Tampering	Medium	Open		This flow transfers data from Azure Cognitive Services (external, untrusted) to Gather Images Application (within Experimental Trust Zone) via HTTPS. Despite encryption, improper certificate validation or compromised certificate authorities could allow man-in-the-middle attacks, enabling data tampering or injection of malicious images. This is a boundary-crossing ingress flow from an external source.	- Implement certificate pinning for Azure Cognitive Services - Use mutual TLS (mTLS) authentication - Validate server certificates against trusted CA list - Implement certificate transparency monitoring - Use TLS 1.3 with strong cipher suites - Implement connection integrity checks - Monitor for certificate anomalies

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Data Injection via Compromised External Service	Tampering	Medium	Open		This HTTPS flow brings images from external Azure Cognitive Services into the Experimental Trust Zone. If the external service is compromised, malicious or poisoned images could be injected into the training pipeline, leading to model poisoning. This is a critical ingress point from an untrusted external source.	- Implement comprehensive input validation for all images - Use malware scanning and content verification - Implement image format and metadata validation - Use checksums and integrity verification - Implement anomaly detection for image characteristics - Use multiple data sources for validation - Monitor for suspicious data patterns

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
	Malicious Library Import into Gather Images Application	Tampering	High	Open		This flow imports third-party tools and ML libraries from external sources into Gather Images Application within the Experimental Trust Zone. Compromised or malicious libraries could inject backdoors, exfiltrate data, or compromise the application. This is a boundary-crossing ingress flow from an untrusted external source.	- Implement software composition analysis (SCA) - Use dependency scanning and vulnerability management - Pin specific versions of all dependencies - Verify package checksums and signatures - Use private package repositories with vetted libraries - Implement automated security scanning - Monitor for suspicious library behavior

imports (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Malicious Library Import into Jupyter Notebook	Tampering	High	Open		This flow imports third-party tools and ML libraries from external sources into Jupyter Notebook within the Experimental Trust Zone. Compromised libraries could execute arbitrary code, exfiltrate training data, or inject malicious logic into ML models. This is a boundary-crossing ingress flow from an untrusted external source.	- Implement software composition analysis (SCA) - Use virtual environments with pinned dependencies - Verify package checksums and signatures - Use private package repositories with vetted libraries - Implement runtime monitoring for suspicious behavior - Use container scanning - Implement least privilege execution

VS Code (SSH) (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Compromised SSH Connection to Gather Images Application	Spoofing	High	Open		This flow uses VS Code over SSH from Engineer (external) to Gather Images Application (within Experimental Trust Zone). If SSH credentials are compromised or the connection is not properly secured, attackers could impersonate engineers and gain unauthorized access. This is a boundary-crossing ingress flow with encrypted transport.	- Enforce SSH key-based authentication with passphrase protection - Implement multi-factor authentication (MFA) - Use certificate-based SSH authentication - Monitor and log all SSH connections - Implement IP allowlisting - Use session recording - Implement anomaly detection for SSH access patterns
	Code Injection via SSH Session	Tampering	Medium	Open		Engineers with SSH access to Gather Images Application could inject malicious code, modify application logic, or tamper with the image gathering process. This flow crosses from external Engineer into the Experimental Trust Zone with privileged access.	- Implement code review and approval workflows - Use version control with audit logging - Implement file integrity monitoring (FIM) - Enable comprehensive audit logging of code changes - Use read-only file systems where possible - Implement change management processes - Monitor for suspicious code modifications

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
	Compromised SSH Connection to Jupyter Notebook	Spoofing	High	Open		This flow uses VS Code over SSH from Engineer (external) to Jupyter Notebook (within Experimental Trust Zone). Compromised SSH credentials could allow attackers to impersonate engineers and execute arbitrary code in the notebook environment. This is a boundary-crossing ingress flow with access to training data and models.	- Enforce SSH key-based authentication with passphrase protection - Implement multi-factor authentication (MFA) - Use certificate-based SSH authentication - Monitor and log all SSH connections - Implement IP allowlisting - Use session recording - Implement anomaly detection for SSH access patterns
	Malicious Code Execution via Notebook	Elevation of Privilege	High	Open		Engineers with SSH access to Jupyter Notebook can execute arbitrary Python code with access to training data and the ability to save models. This flow crosses from external Engineer into the Experimental Trust Zone with significant privileges, creating risk of data exfiltration or model poisoning.	- Implement kernel isolation and sandboxing - Use containerization with resource limits - Implement least privilege execution contexts - Enable comprehensive audit logging - Implement code review for production-bound notebooks - Restrict network access from notebook environment - Implement data access controls

stores (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Unvalidated Image Storage Leading to Data Poisoning	Tampering	Medium	Open		This flow stores images from Gather Images Application to Training and Validation Images storage within the Experimental Trust Zone. If images are not properly validated before storage, malicious or corrupted images could poison the training dataset. The flow is internal to the trust zone but handles data from external sources.	- Implement comprehensive image validation before storage - Use malware scanning for all images - Validate image format, size, and metadata - Implement checksums and integrity verification - Use content-based filtering - Implement anomaly detection for image characteristics - Enable versioning for stored images

loads (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	API Key Exposure During Load Operation	Information Disclosure	High	Open		This flow loads API keys from API Key Storage to Gather Images Application within the Experimental Trust Zone. If the load operation is not properly secured, keys could be exposed in memory, logs, or network traffic. The flow handles credential-sensitive data.	- Use secure credential management libraries - Implement memory encryption for sensitive data - Avoid logging sensitive data - Use TLS for all internal communications - Implement secrets rotation - Use short-lived tokens where possible - Monitor for unusual key access patterns

processes (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Unauthorized Data Access During Processing	Information Disclosure	Medium	Open		This flow processes training images from Training and Validation Images storage to Jupyter Notebook within the Experimental Trust Zone. Engineers with notebook access could exfiltrate or misuse training data. The flow handles business-sensitive data.	- Implement data access logging and monitoring - Use data loss prevention (DLP) controls - Restrict network access from notebook environment - Implement egress filtering - Monitor for unusual data access patterns - Implement data classification and access controls - Use anomaly detection for data access

package (Data Flow)

Description: Transfer data from Machine Learning Model to Deployment.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Model Tampering During Packaging	Tampering	High	Open		This flow packages models from Machine Learning Model storage to Deployment service within the Experimental Trust Zone. If the packaging process is compromised, malicious code could be injected into models before deployment to production. This is a critical path from experimental to production.	- Implement model integrity verification with digital signatures - Use checksums for model validation - Enable audit logging for all packaging operations - Implement automated model validation - Use immutable build artifacts - Monitor for unauthorized model modifications - Implement secure packaging pipelines

save.h5 (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Malicious Model Upload from Notebook	Tampering	High	Open		This flow saves trained models from Jupyter Notebook to Machine Learning Model storage within the Experimental Trust Zone. Engineers could save backdoored or poisoned models that will later be deployed to production. This is a critical control point in the ML pipeline.	- Implement automated model validation and testing - Use model performance benchmarking - Require peer review for model saves - Implement model explainability checks - Enable audit logging for all model saves - Implement anomaly detection for model behavior - Use digital signatures for model integrity

package (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Unauthorized Model Access During Deployment	Information Disclosure	Medium	Open		This flow packages models from Stored Machine Learning Model (Production Trust Zone) to Deployment service (Experimental Trust Zone). This represents a flow from production back to experimental zone, which could expose production models if not properly secured. This is an unusual egress flow from production.	- Implement strict access controls for production model access - Use encryption in transit (TLS 1.2+) - Enable audit logging for all model access - Implement least privilege access - Use separate credentials for production access - Monitor for unusual access patterns - Implement data classification and protection

package (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Malicious Code Injection During Packaging	Tampering	High	Open		This flow packages source code and configuration from Source Code and Configuration storage to Deployment service within the Experimental Trust Zone. If the packaging process is compromised, malicious code could be injected before deployment to production. This is a critical path from experimental to production.	- Implement code signing and verification - Use immutable build artifacts - Enable audit logging for all packaging operations - Implement code scanning and security analysis - Use version control with mandatory review - Monitor for unauthorized code changes - Implement secure CI/CD pipelines

HTTPS (Data Flow)

Description: Transfer from User to API Gateway.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Insufficient TLS Configuration	Information Disclosure	Medium	Open		This HTTPS flow from User (external) to API Gateway (Production Trust Zone) is encrypted but may use weak TLS versions or cipher suites. This is a boundary-crossing ingress flow from untrusted external users into production.	- Enforce TLS 1.2 or higher - Use strong cipher suites only - Implement perfect forward secrecy (PFS) - Disable weak protocols (SSLv3, TLS 1.0, TLS 1.1) - Implement HSTS (HTTP Strict Transport Security) - Use certificate pinning where appropriate - Monitor for TLS downgrade attacks
	Replay Attack on API Requests	Tampering	Medium	Open		This HTTPS flow from external User to API Gateway could be vulnerable to replay attacks if requests are not properly protected with nonces, timestamps, or other anti-replay mechanisms. This is a boundary-crossing ingress flow into the Production Trust Zone.	- Implement request nonces or unique identifiers - Use timestamp validation with short validity windows - Implement idempotency keys for critical operations - Use JWT with expiration times - Implement request signing - Monitor for duplicate requests - Implement rate limiting

update (Data Flow)

Description: Transfer data from Bastion to API Gateway.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Unauthorized Configuration Changes via Bastion	Tampering	High	Open		This flow allows Bastion to update API Gateway configuration within the Production Trust Zone. If Bastion is compromised, attackers could disable security controls, redirect traffic, or expose internal services. This is an internal flow within production with high privileges.	- Implement strict access controls for configuration changes - Use infrastructure as code (IaC) with version control - Require multi-person approval for changes - Enable audit logging for all configuration changes - Implement change validation and testing - Use immutable infrastructure patterns - Monitor for unauthorized changes

HTTPS (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Insufficient Internal Authentication	Spoofing	Medium	Open		This HTTPS flow from API Gateway to Simple Python Web Server is internal to the Production Trust Zone. If internal authentication is weak or absent, a compromised component could impersonate the API Gateway. This is an internal flow within the production environment.	- Implement mutual TLS (mTLS) for internal communications - Use service mesh for authentication and authorization - Implement API keys or JWT tokens for internal services - Use network segmentation - Monitor for unusual internal traffic patterns - Implement zero-trust architecture

update (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Unauthorized Application Updates via Bastion	Tampering	High	Open		This flow allows Bastion to update Simple Python Web Server within the Production Trust Zone. If Bastion is compromised, attackers could inject malicious code, modify application logic, or create backdoors. This is an internal flow within production with high privileges.	- Implement strict access controls for updates - Use infrastructure as code (IaC) with version control - Require multi-person approval for updates - Enable audit logging for all updates - Implement code signing and verification - Use immutable infrastructure patterns - Monitor for unauthorized changes

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
	Model Tampering During Load Operation	Tampering	High	Open		This flow loads production models from Stored Machine Learning Model to Simple Python Web Server within the Production Trust Zone. If the load operation is not properly secured, models could be tampered with in transit. The storage is not encrypted according to the model.	- Implement model integrity verification with digital signatures - Use checksums for model validation - Enable encryption in transit (TLS 1.2+) - Implement model validation before loading - Enable audit logging for model loads - Monitor for unauthorized model access - Enable encryption at rest for storage

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Unencrypted Model Storage Access	Information Disclosure	High	Open		This flow loads models from Stored Machine Learning Model which is not encrypted (isEncrypted: false). Models are transmitted and stored in plaintext, making them vulnerable to theft if network or storage access controls are bypassed. This is an internal flow within the Production Trust Zone.	- Enable encryption at rest immediately for model storage - Use encryption in transit (TLS 1.2+) - Implement access controls with least privilege - Enable audit logging for all model access - Use Azure Storage Service Encryption - Implement key management with Azure Key Vault - Monitor for unauthorized access

SSH (Data Flow)

Description: Transfer sensitive data from Deployment Service to Bastion

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Compromised Deployment Credentials	Spoofing	High	Open		This SSH flow from Deployment service (Experimental Trust Zone) to Bastion (Production Trust Zone) crosses trust boundaries. If deployment credentials are compromised, attackers could deploy malicious code to production. This is a critical boundary-crossing flow from experimental to production.	- Use certificate-based SSH authentication with short-lived certificates - Implement secrets management with automatic rotation - Use service accounts with least privilege - Implement just-in-time (JIT) credential provisioning - Monitor and alert on all deployment activities - Use hardware security modules (HSM) for key storage - Implement anomaly detection for deployment patterns
	Unauthorized Production Deployment	Elevation of Privilege	High	Open		This SSH flow allows Deployment service to access Bastion and deploy to production. Without proper controls, unauthorized or malicious deployments could compromise production systems. This is a boundary-crossing flow from Experimental to Production Trust Zone.	- Implement multi-person approval workflows - Use automated validation and testing gates - Require digital signatures for deployments - Enable comprehensive audit logging - Implement deployment pipelines with security scanning - Use canary deployments and gradual rollouts - Implement rollback capabilities - Monitor for unauthorized deployment attempts

update (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Unauthorized Production Model Updates	Tampering	High	Open		This flow allows Bastion to update Stored Machine Learning Model within the Production Trust Zone. If Bastion is compromised, attackers could replace production models with malicious versions. This is an internal flow within production with high privileges affecting critical assets.	- Implement strict access controls for model updates - Require multi-person approval for production model changes - Use digital signatures for model integrity - Enable audit logging for all model updates - Implement model validation before deployment - Use version control with rollback capabilities - Monitor for unauthorized model changes - Implement automated model testing

SSH (Data Flow)

Description: Transfer data from Infrastructure Admin to Bastion.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Compromised Admin SSH Credentials	Spoofing	High	Open		This SSH flow from Infrastructure Admin (external) to Bastion (Production Trust Zone) is the primary ingress point for administrative access. Compromised credentials would grant full production access. This is a boundary-crossing ingress flow with the highest privileges.	- Enforce SSH key-based authentication with hardware security keys - Implement multi-factor authentication (MFA) with FIDO2/WebAuthn - Use certificate-based SSH authentication with short-lived certificates - Implement privileged access management (PAM) - Require just-in-time (JIT) access approval - Monitor and alert on all admin access attempts - Implement IP allowlisting - Use session recording
	SSH Session Hijacking	Spoofing	Medium	Open		This SSH flow from external Infrastructure Admin to Bastion could be vulnerable to session hijacking if not properly secured. An attacker could take over an active admin session to gain production access. This is a boundary-crossing ingress flow with encrypted transport.	- Use SSH protocol version 2 only - Implement session timeout policies - Use strong encryption algorithms - Implement session monitoring and anomaly detection - Use certificate-based authentication - Implement session recording - Monitor for unusual session behavior - Use jump host architecture with additional authentication

update (Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Duplicate Threat - Same as bastion-ml-model	Tampering	High	Open		This flow appears to be a duplicate of bastion-ml-model flow, allowing Bastion to update Stored Machine Learning Model within the Production Trust Zone. The same threats apply: unauthorized model updates, model tampering, and privilege abuse.	- Implement strict access controls for model updates - Require multi-person approval for production model changes - Use digital signatures for model integrity - Enable audit logging for all model updates - Implement model validation before deployment - Use version control with rollback capabilities - Monitor for unauthorized model changes

(Data Flow)

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

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	SSH Key Exposure During Load Operation	Information Disclosure	High	Open		This flow loads authorized SSH keys from Authorized Keys storage to Bastion within the Production Trust Zone. The flow is encrypted but if not properly secured, keys could be exposed during the load operation. This handles credential-sensitive data for production access.	- Use secure credential management libraries - Implement memory encryption for sensitive data - Use TLS 1.2+ for all communications - Avoid logging sensitive data - Implement key rotation policies - Monitor for unusual key access patterns - Use hardware security modules (HSM) for key protection - Enable audit logging for key access

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
	Malicious Library Import into Production Web Server	Tampering	High	Open		This flow imports third-party tools and ML libraries from external sources into Simple Python Web Server within the Production Trust Zone. Compromised libraries could inject backdoors, exfiltrate data, or compromise production services. This is a boundary-crossing ingress flow from untrusted external sources into production.	- Implement software composition analysis (SCA) - Use automated vulnerability scanning - Pin specific versions of all dependencies - Verify package checksums and signatures - Use private package repositories with vetted libraries - Implement runtime application self-protection (RASP) - Use container scanning for production images - Regularly update dependencies with security patches

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Supply Chain Attack on Production Dependencies	Elevation of Privilege	High	Open		This flow brings external third-party libraries into the production Simple Python Web Server. Vulnerable or compromised libraries could be exploited to gain elevated privileges in production. This is a critical boundary-crossing ingress flow into the Production Trust Zone.	- Implement continuous vulnerability scanning - Use automated dependency update tools - Implement runtime monitoring for suspicious behavior - Use least privilege execution contexts - Implement sandboxing for library execution - Monitor for suspicious library behavior - Maintain an inventory of all dependencies with risk assessment - Use container image scanning