AZURE KEY VAULT SECURE DEPLOYMENT: DEMONSTRATION, TESTING, AND TROUBLESHOOTING DOCUMENT

This document outlines the validation tests used to demonstrate the secure deployment of the Azure Key Vault solution, focusing on the Principle of Least Privilege (PoLP), and summarizes the troubleshooting steps taken for the SSH Key validation.

I. PROJECT SECURITY CONTEXT AND ARCHITECTURE

The solution utilizes a two-module architecture, which provides Separation of Concerns by separating Key Vault creation (Management Plane) from secret and access policy management (Data Plane). This structure implements robust security principles:

1. **Principle of Least Privilege (PoLP):** Granular Access Policy grants only the essential "Get" access to the principal (Managed Identity), strictly avoiding unnecessary permissions like List or Delete.
2. **Secret Zero:** Access is granted using a system-assigned **Managed Identity**'s `principal_id`, eliminating the need to handle or store traditional credentials in application code.
3. **Data Resilience:** The Azure Key Vault Module configured **Soft Delete** with a 7-day retention period, protecting against immediate permanent deletion of the vault or its secrets.

II. AZURE KEY VAULT SECURITY DEMONSTRATION (LEAST PRIVILEGE VALIDATION)

The core security requirement to apply least privilege principles was explicitly confirmed by multiple tests. These demonstrations prove the rigorous enforcement of the access policy applied to the Virtual Machine's Managed Identity.

1. Data Plane Access Validation (Least Privilege Enforcement)

These tests confirm the Managed Identity has the required "Get" permission while denying all discovery and destructive permissions. This validation confirms the architectural decision to use Managed Identity.

Test Goal	Command/Action	Expected Result	Principle Proven

Positive Test (Retrieve Secret)	`az keyvault secret show --name db-password --vault-name $KV_NAME --query value -o tsv`	SUCCESS	Validates "Get" access and confirms the **Secret Zero principle** is functional. The identity is working to retrieve the secret (`db-password`).
Negative Test (Deny Listing)	`az keyvault secret list --vault-name $KV_NAME`	**FAILURE (403 Forbidden)**	Proves **Data Plane Segmentation** by denying unauthorized discovery. The VM cannot discover or map other secrets (denies List permission).
Negative Test (Deny Destruction)	`az keyvault secret delete --name db-password --vault-name $KV_NAME`	**FAILURE (403 Forbidden)**	Confirms the VM cannot tamper with or delete the secret (denies Delete permission), mitigating risk if the application process is compromised.

2. Management Plane Boundary Checks

These commands ensure the Managed Identity has minimal rights (Reader Role) for resource discovery but strictly cannot compromise the underlying infrastructure (Resource Group).

Test Goal	Command/Action	Expected Result	Principle Proven
Validate Minimal Reader Role	`az keyvault show --name $KV_NAME --query id`	SUCCESS: Returns the Key Vault ID	Confirms the Managed Identity has the minimal required **Reader Role** (Management Plane), allowing resource URI discovery.
Deny Resource Group Write (IAM Boundary Check)	Attempt to update a tag on the Resource Group by the Private VM	**FAILURE (403 Forbidden)**	Proves the VM **does NOT have Contributor/Owner rights** over the Resource Group, successfully enforcing the strict IAM boundary.

III. SSH KEY VALIDATION: TROUBLESHOOTING AND STRUCTURAL SUCCESS

The requirement to automatically generate SSH keys was met. The primary test goal was to achieve a successful SSH connection from the Local Machine to the Private VM via the Bastion Host.

1. Functional Failure Analysis (The Network Block) 🚫

The attempt to establish a functional SSH connection failed repeatedly due to a firewall issue in Azure.

Error Encountered	Cause/Context	Resolution Attempt
`ssh: connect to host ... port 22: Connection timed out`	**Network Security Group (NSG) Block:** A firewall rule in Azure was blocking inbound TCP Port 22 traffic from the specific Public IP to the Bastion Host.	The NSG was successfully created/edited to **Allow access** from the local public IP.
`Identity file ... not accessible: No such file or directory.`	**User Error:** The command was repeatedly run from the remote VM/Cloud Shell instead of the correct Local Machine where the private key file resides.	N/A

Conclusion of Functional Failure: The functional validation of the key was **blocked by a networking flaw (NSG)**, and not by a failure in the Key Vault or VM security configuration.

2. Structural Success: Proof of Key Generation ✅

Despite the network failure preventing functional validation, the SSH Key Generation requirement was structurally and definitively met.

- **Structural Proof (CLI):** The Azure CLI was used to extract the public key data directly from the VM's configuration.
- **Command Used:** `az vm show --resource-group rg-terraform-securelab --name private-vm --query "osProfile.linuxConfiguration.ssh.publicKeys.keyData" -o tsv`.
- **Result Confirmed:** A valid public key string was returned (`ssh-ed25519...`).
- **Structural Location:** The public key data is stored in the VM's `osProfile` settings, which Azure uses to populate the VM's `~/.ssh/authorized_keys` file, fulfilling the structural requirement.