**The Challenge – Private Endpoints for Azure Storage**

Currently, Azure Storage services (Blob, File, Table, Queue, etc.) offer only public IP endpoints for device and client connectivity.  While all communication with Azure Storage requires an encrypted TLS/SSL channel, there are customers who prefer device communication with storage services to occur

over a private connection.

There are several important use cases where Azure Storage would benefit from offering a private

endpoint to devices and clients:

* Private traffic though ExpressRoute (e.g., factory devices with secure private IPs that use MPLS for Cloud connectivity)
* Private traffic through a VPN (e.g., remote sensors that use P2S for high security)
* Devices requiring internal DNS resolution of a PaaS endpoint

**The Solution – Azure Firewall as a Private Azure Storage Gateway**

Azure Firewall is a managed, cloud-based network security service which provides fully stateful firewall inspection with built-in high-availability and unrestricted cloud scalability.

Its applicability to solving this private gateway challenge is as follows:

* Azure Firewall is used as an HA scale-out tier that provides a private IP endpoint for Azure Storage clients and devices.
* Azure Firewall can provide a single endpoint for multiple storage accounts while providing granular control with full auditing capabilities
* Azure Firewall can leverage service endpoints to prevent storage accessibility from any other network while allowing accessibility to on-prem resources.

**The Essential Architecture**

Virtual Network (VNet) Service Endpoints extends your virtual network private address space, and

the identity of your VNet, to multi-tenant Azure PaaS services over a special NAT tunnel in the

Azure fabric.  Service Endpoints allow you to secure your critical Azure service resources to only

your virtual networks. Traffic from your VNet to the Azure PaaS service always remains on the

Microsoft Azure backbone network.

The key here is the ability to deploy Azure Storage with Service Endpoints, while *also* making these

private PaaS endpoints available over an IPSEC VPN or the ExpressRoute private peering. The current limitation of Service Endpoints is that it is not accessible to on-premise resources.  Using service endpoints with Azure Firewall gives on-premise devices a private endpoint to hit coupled with a stateful security device providing centralized logging.

The following diagram illustrates on-premise to cloud communication architecture secured via the

Azure Firewall hosted in Microsoft Azure.



**Lab Components**

Resource group for HUB VNet and a minimum of two subnets:

* GatewaySubnet (contains Azure ExpressRoute Gateway, /27 min, /26 recommended)
* AzureFirewallSubnet (contains Azure Firewall and will scale based on subnet size, minimum /26)
* Service Endpoints enabled for Microsoft Storage
* VNET Peering to Department 1 VNET

Resource group for Department 1 VNet and a minimum of one subnet:

* Server Subnet
* Test VM (used for testing access to Dept 1 storage account)
* VNET Peering to HUB VNET

On-prem connectivity and test host:

* ExpressRoute private peering or IPSEC VPN
* Test host (used for testing access to On-Prem storage account)

Storage accounts with access limited to AzureFirewallSubnet:

* On-Prem Blob Storage account with a single container
* Department 1 Blob Storage account with a single container

**DNS Manipulation**

When the storage accounts are created, a blob service endpoint FQDN will be automatically generated based upon the name of the resource. For example, when we create a blob storage account named department1 and we look at the properties of that resource we will see a Primary Blob Service Endpoint FQDN of <https://department1.blob.core.windows.net> as seen below.



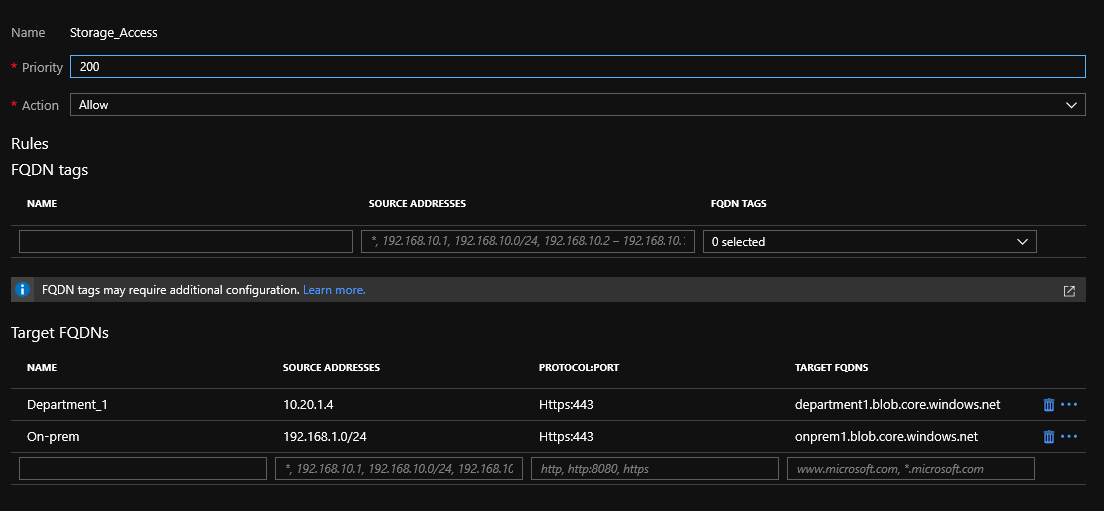
As this FQDN will naturally resolve to a public IP address, we need to modify DNS entries in order to redirect traffic to the private endpoint of the Azure Firewall. We can do this with a single host by modifying the local hosts file on the machine or, for multiple hosts, by creating an authoritative record for this FQDN within our local domain.

We must ensure that we do not modify the FQDN as this would break TLS and give users a certificate warning. Future support for importing of custom certificates is coming to Azure Firewall however, this functionality does not exist today.

**Network Configuration**

Azure Firewall:

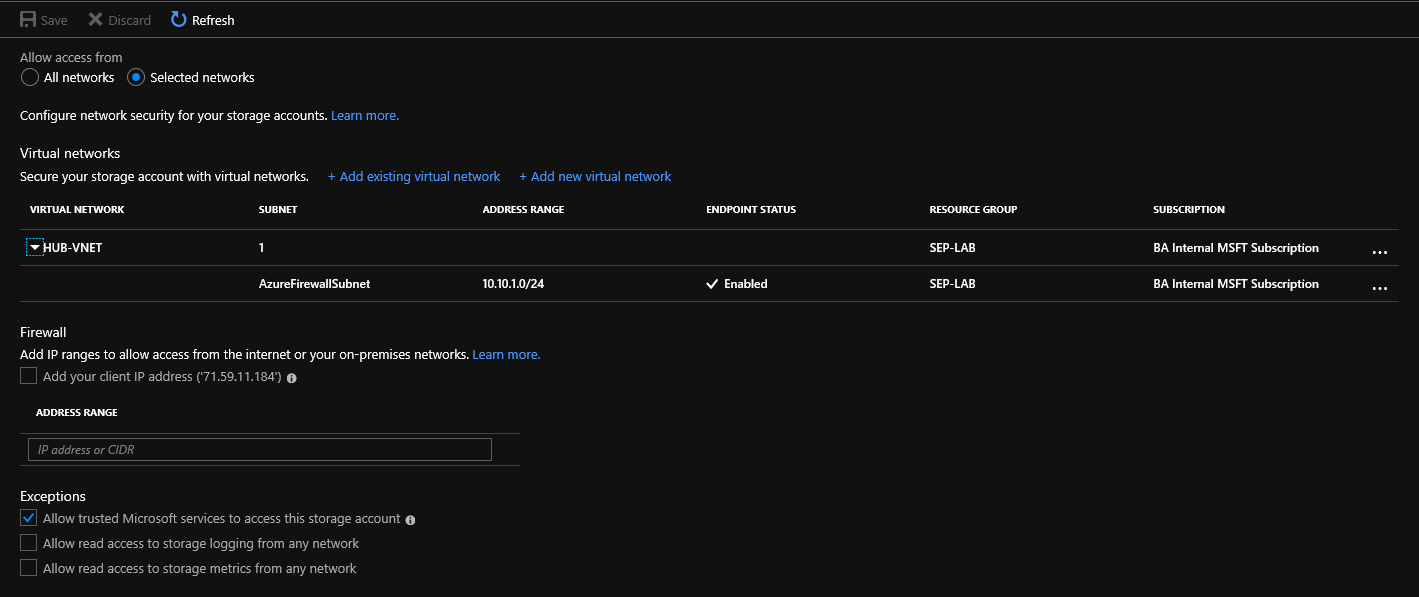
The Azure Firewall Application Rule Collection is configured to allow access from appropriate resources to their respective storage account FQDNs as seen below.



As the Azure Firewall is a default deny device, any traffic destined to these endpoints which is not explicitly defined in these rulesets will be dropped. This ensures that no unapproved endpoints obtain access to the storage accounts as well as prevents data exfiltration between storage accounts.

Azure Storage:

To ensure the storage accounts can only be accessible via the Azure Firewall, we enabled service endpoints on the AzureFirewallSubnet and locked to the storage account down to only this subnet. To enforce this, we only allow access from the AzureFirewallSubnet within the Firewall and virtual networks blade of each storage account as seen below.



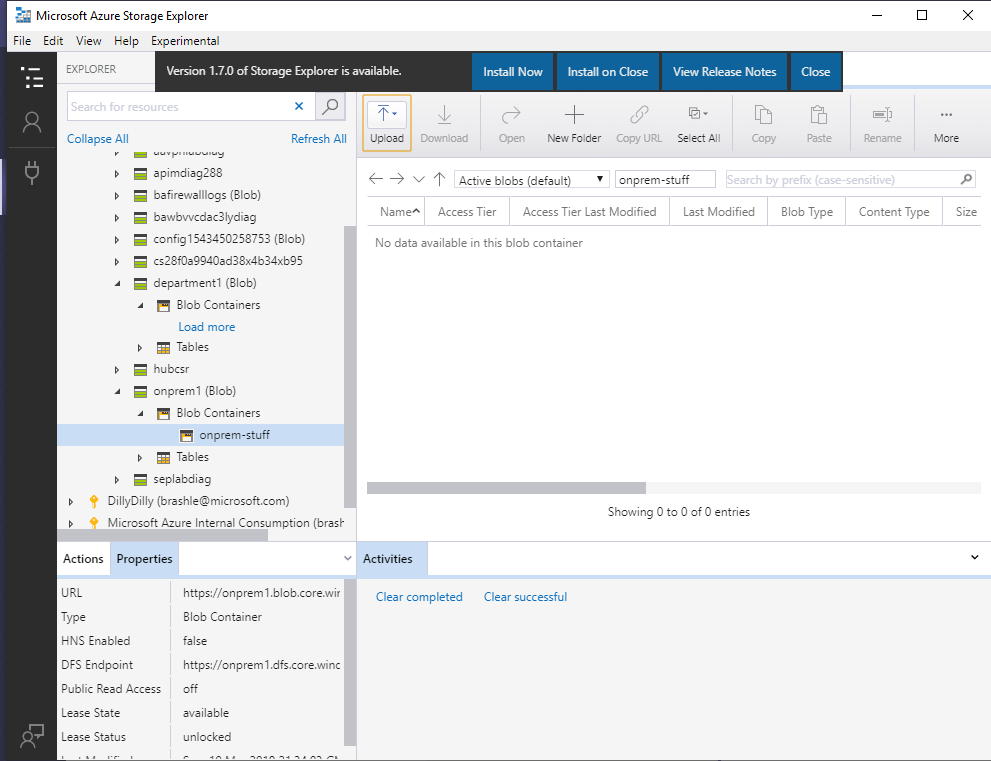
Endpoint Devices:

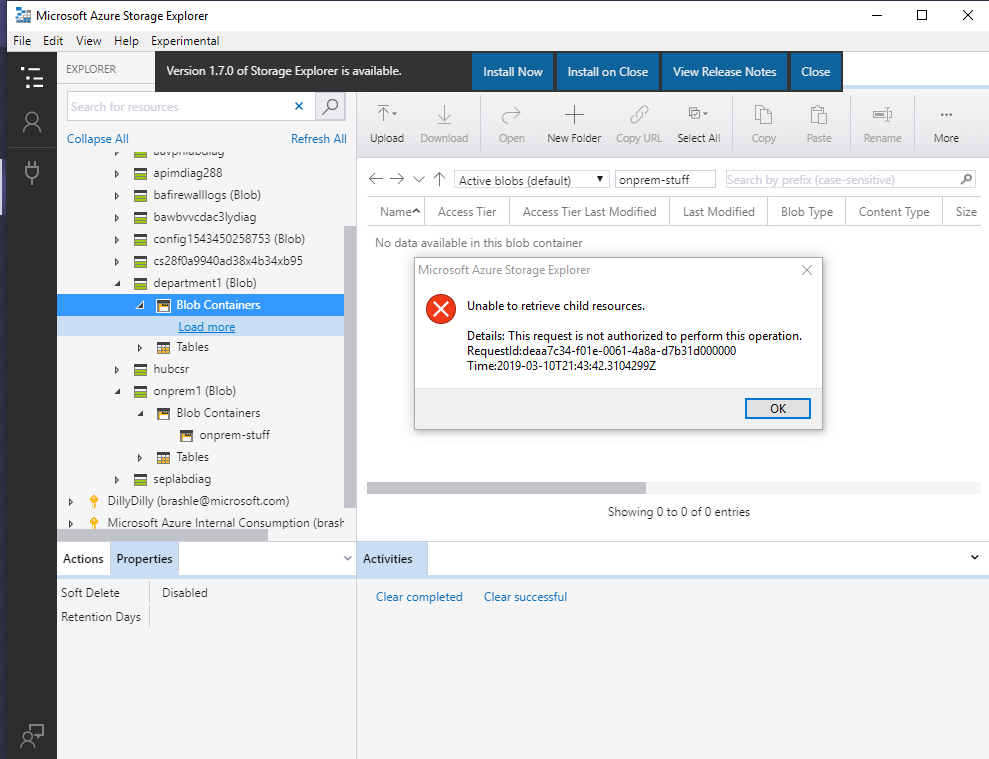
As mentioned in the DNS Manipulation section of this document, we have to ensure traffic destined to each endpoint’s respective storage account is sent to the Azure Firewall. For the Department 1 host within Azure we modified the hosts file locally. For on-premise devices, we created an authoritative record within DNS as modifying the hosts file on all machines was impractical.

**Testing and Validation**

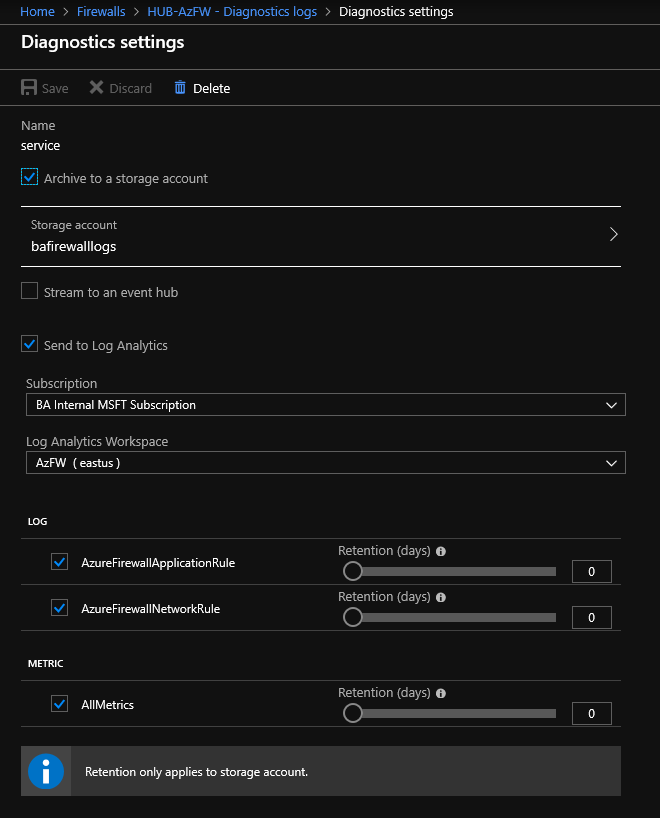
Testing Connectivity:

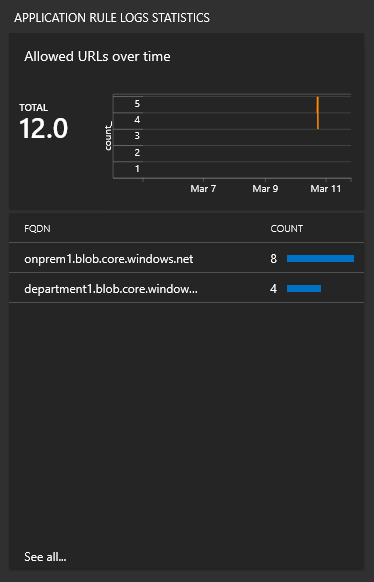
To validate this setup, we installed Azure Storage Explorer (<https://azure.microsoft.com/en-us/features/storage-explorer/>) on both the Department 1 and on-prem hosts and attempted to access the storage accounts. As displayed in the screenshots below, the on-prem host was able to reach its respective storage account while receiving an error message when accessing the department 1 storage account and vice versa.





In addition to the verification of access using Azure Storage Explorer, we also deployed Log Analytics and configured the Azure Firewall to log traffic to this for additional verification. We can see the traffic logs for these attempts here.





Step by Step configuration of this lab can be found here: <https://github.com/Microsoft/ServiceEndpoints_with_AzureFirewall>

This lab can be deployed via a template here: TBD