PRODUCT NAME

Azure Firewall Premium 20Q3

NDA Private Preview | Participant Guide

v1.2

Contents

Overview

Azure Firewall Standard has been generally available since September 2018. It is a cloud native firewall-as-a-service offering which enables customers to centrally govern and log all their traffic flows using a DevOps approach. The service supports both application and network level filtering rules and is integrated with the Microsoft Threat Intelligence feed for filtering known malicious IP addresses and domains. Azure Firewall is highly available with built-in auto scaling.

While Azure Firewall Standard has been a good fit for many customer scenarios, it is lacking key next generation firewall capabilities that are required for highly sensitive and regulated environments. We are excited to introduce the new Azure Firewall Premium private preview as our first step towards closing these gaps.

The private preview edition is offered to selected customers who are early adopters by nature and willing to provide feedback to allow the Azure Firewall team to provide a better service in the coming public preview and GA releases.

This document is intended to serve these customers as an introduction and overview of the new offered capabilities as well as actual operational guidance.

All private preview functionality is available via Azure Resource Manager templates and REST API only. This document includes some template examples to ease the use of the deployment and configuration. Azure portal UI will be added in public preview and enhanced for GA.

How to use this document most effectively

The document structure consists of first high-level features description and then general information about private preview participation. The last part is a drill down to the detailed deployment instructions and functionality setting with examples.

If this is your first time working with Azure Firewall, then go throughout the entire document step by step.

If you are already familiar with Azure firewall and have hands-on experience, you can skip to the template deployment part to get started.

If this is your first time participating in a private preview release, read the participation section to get familiar with how this process works.

Scope and Plans Ahead

This private preview includes outbound TLS inspection, IDPS (Intrusion Detection & Prevention System) and URL filtering. Traffic filtering based on Web Categorization will be included in the public preview.

Azure Firewall Premium Release content

Azure Firewall Premium is planned to be released in 3 release cycles:

1. **Private Preview** – This is the current release which include TLS Inspection, IDPS and URL Filtering. Available for a selected set of customers free of charge with no SLA (note: your testing environment except for the firewall is charged at the GA price)
2. **Public Preview** –Planned for Q4 CY2020 and includes traffic filtering based on Web Categories in addition to the private preview content. The public preview will be available to all customers with a 50% discount and no SLA (note: your testing environment except for the firewall is charged at the GA price).
3. **General Availability (GA)** – This is the final release with all the following content targeted for production deployments with full SLA Tentative ETA is Q1 CY2021.

TLS Inspection

Transport Layer Security (TLS), previously known as Secure Sockets Layer (SSL), is the standard security technology for establishing an encrypted link between a client and a server. This link ensures that all data passed between the client and server remain private and encrypted.

Azure Firewall Premium decrypts the outbound traffic, performs the required value-added security services and then encrypt the traffic which is sent to the original destination.

There are several advantages of doing TLS Inspection on Azure Firewall:

1. Enhanced Visibility: logs and metrics are available for all decrypted traffic.
2. URL Filtering: TLS Inspection is an enabler for advanced security services, such as URL Filtering. An URL, as opposed to a FQDN, is not accessible to the FW when traffic is encrypted. URLs provide stricter outbound traffic filtering for domains that are common for different customers (for example, OneDrive.live.com).
3. IDPS: while some detections can be done for encrypted traffic, TLS inspection is important to utilize the best of IDPS.
4. Threat Intelligence based filtering works better for shared domains when URLs are visible (for example, CDN)

While customers are interested in having TLS inspection for both outbound and inbound scenarios, this Azure Firewall Premium release only focuses on outbound and internal scenarios. Customers who are interested in inbound TLS inspection, will be able to chain Azure Firewall with Application Gateway WAF for that purpose. For more information, see Azure Virtual Network security.

IDPS

A network intrusion detection and prevention system (IDPS) allows you to monitor network activities for malicious activity, log information about this activity, report it, and optionally attempt to block it.

Azure Firewall Premium provides signature-based IDPS to allow rapid detection of attacks by looking for specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware.

IDPS allow you to detect attacks in all ports and protocols for non-encrypted traffic. However, when HTTPS traffic needs to be inspected, Azure Firewall can utilize its TLS inspection capability to decrypt the traffic and better detect malicious activities.

IDPS Bypass List allow you not filter traffic to any of the IP addresses, ranges, and subnets specified in the bypass list.

In Public Preview IDPS will never filter traffic for Office365 or Azure PaaS ranges as defined by a Service Tag.

URL Filtering

URL filtering extends Azure Firewall's FQDN filtering capability to consider an entire URL (for example www.contoso.com/a/c instead of www.contoso.com).

URL Filtering can be applied both on HTTP and HTTPS traffic. When HTTPS traffic needs to be inspected, Azure Firewall can utilize its TLS inspection capability to decrypt the traffic and extract the target URL to validate whether access is permitted. TLS inspection requires opt-in at the application rule level. Once enabled, you can use URLs for filtering with HTTPS.

Web Categories

Traffic filtering based on Web Categorization will be included in Azure Firewall Premium Public Preview, see the following *Release Plan Milestones* for more details on target dates.

Web Categories lets administrators allow or deny user access to website categories such as gambling websites, social media websites and others. Web categories will also be included in Azure Firewall Standard, but it will be more finetuned in Azure Firewall Premium. As opposed to Web Categories capability in Standard SKU which matches the category based on an FQDN, the Premium SKU matches the category according to the entire URL for both HTTP and HTTPS traffic.

For example, if Azure Firewall intercepts an HTTPS request for www.google.com/news, the following categorization is expected:

- Firewall Standard – Only the FQDN part will be examined, so www.google.com will be categorized as "Search Engine".
- Firewall Premium – the complete URL will be examined, so www.google.com/news will be categorized as "News".

Release Plan Milestones

The full list of planned features with their tentative delivery dates, can be found in the following table.

Feature Name	Release State	Tentative Target Date
TLS Inspection	Public Preview	Q4 2020
IDPS	Public Preview	Q4 2020
URL Filtering	Public Preview	Q4 2020
Web Categorization	Public Preview	Q4 2020
All Premium Features	GA	Q1 2021

Private Preview Participation

Customers who are willing evaluate and provide feedback for this private release can apply using this form. Once approved and enrolled, the provided subscription IDs will be enabled for the preview.

Note – Customer approval for private preview does not mean all its subscriptions are opt-in by default to the service but only the subscription IDs that were enrolled by Microsoft.

Supported Regions

The following regions are supporting Azure Firewall Premium Private Preview edition:

1. US East
2. US East 2
3. US Central
4. US West
5. US West 2
6. North Central US
7. South Central US
8. West Central US
9. Canada Central
10. Canada East
11. Brazil South
12. Europe North
13. Europe West
14. UK West
15. UK South
16. France South
17. France Central
18. East Asia
19. Southeast Asia
20. Australia Central

21. Australia Central 2
22. Australia East
23. Australia Southeast
24. Japan East
25. Japan West
26. Korea Central
27. Korea South
28. UAE Central
29. West India
30. South Africa North

Additional regions are planned to be added upon Public Preview and GA release.

Private Preview Pricing

All Private Preview premium features are free of charge. However, your testing environment (for example, VMs, Public IP addresses, etc.) will be charged at GA rates.

- Public Preview – 50% discount on Azure list price
- GA – according to Azure list price

Support & Feedback

You may get support and/or provide feedback in many ways, as noted below:

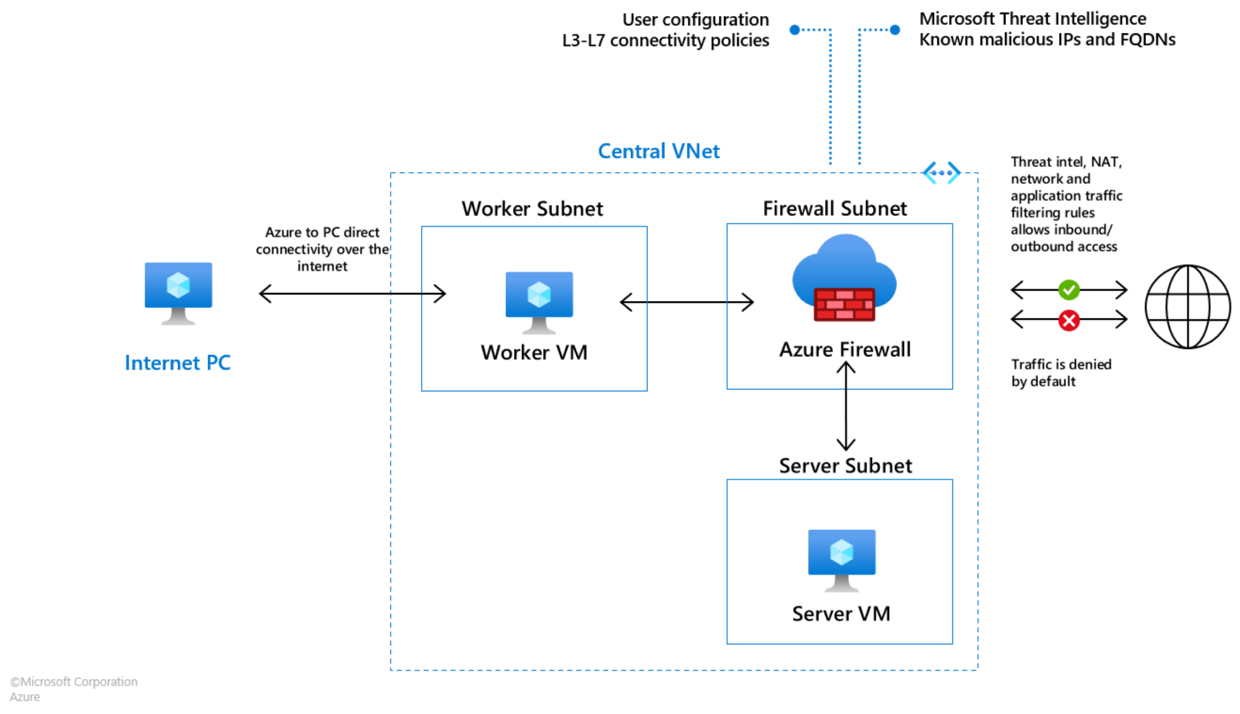
- Use our feedback form to give us your insight and suggestions to make this solution better
- Send an email directly to the Azure Firewall Premium Private Preview Support DL: fwpremiumpreview@microsoft.com
- Leverage the Azure Firewall & Firewall Manager Yammer forum for feedback and discussion related to Azure Firewall Premium.
- If you are in our ongoing preview program, you may leverage the Network Security Private Preview Channel
    - If you are not in our ongoing program and would like to join, go to: www.aka.ms/SecurityPrP

We are looking forward for your feedback.

Suggested Topologies

Azure Firewall requires a customer VNET and a dedicated subnet for its deployment. The protected resources can either live in a different subnet in the same VNET or more commonly, in another VNET which is peered with Azure Firewall VNET, following the reference Azure Hub & Spokes model architecture. For this preview purposes, it is recommended to start with a single VNET architecture to simplify the initial flow. Once it's deployed and working, it is possible to peer additional VNETs to the Azure Firewall VNET for testing purposes as shown in the **Topology 2 - Spoke to Spoke** diagram.

Topology 1 - Central VNet


User configuration
L3-L7 connectivity policies
Microsoft Threat Intelligence
Known malicious IPs and FQDNs
Central VNet
Worker Subnet
Firewall Subnet
Azure to PC direct connectivity over the internet
Internet PC
Worker VM
Azure Firewall
Threat intel, NAT, network and application traffic filtering rules allows inbound/ outbound access
Traffic is denied by default
Server Subnet
Server VM
©Microsoft Corporation
Azure

In this topology, a worker VM serves as a client which initiate HTTP/S requests to the Internet as well as towards Server VM.

The Server VM image contains a pre-installed NGINX that is fully capable to answer both HTTP/S requests via these FQDNs: http://server.2020-private-preview.com/ https://server.2020-private-preview.com/ . The web server response for both HTTP and HTTPS is "HelloWorld". This webserver is available only from within the testing perimeter e.g. Worker VM.

When simulating malicious request generation from a worker VM, you should target these requests toward NGINX installed on the Server VM rather than an Internet-based commercial web site.

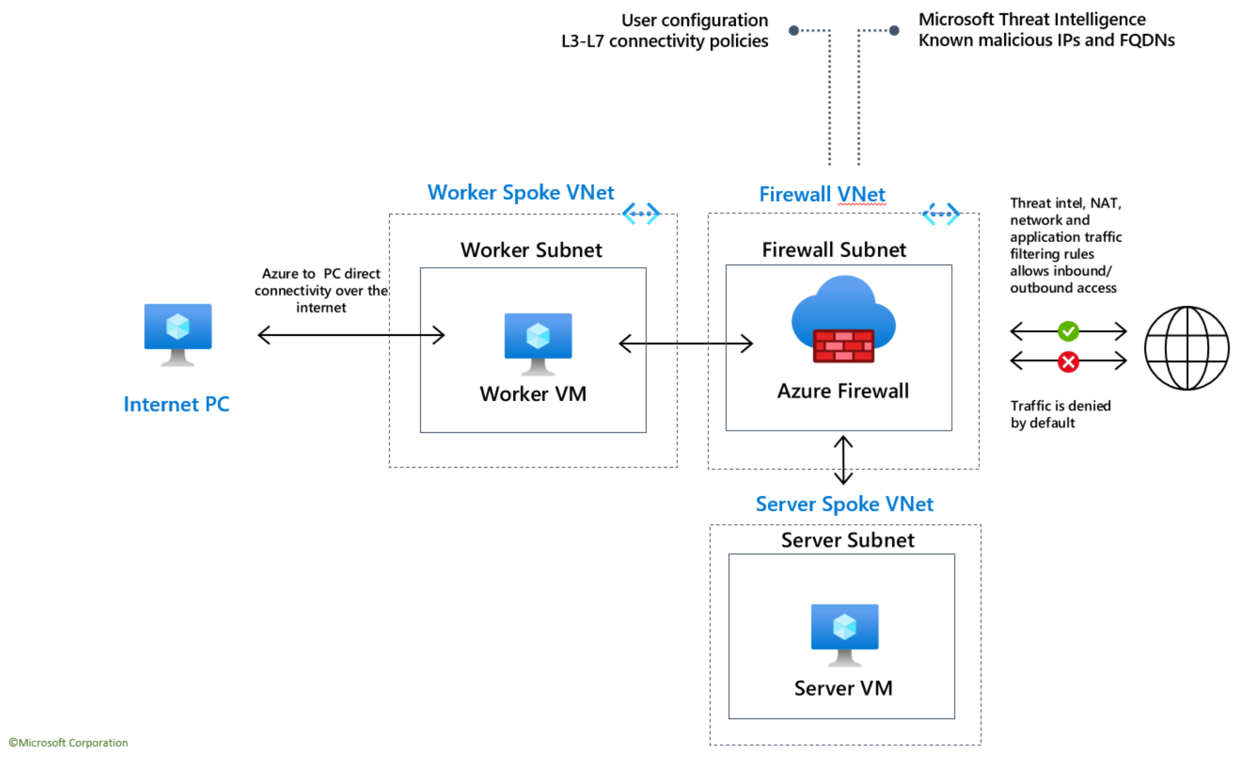
To establish remote connectivity from your computer to the worker VM, RDP can be used with standard TCP port 3389. The Worker VM has a public IP exposed to the Internet and TCP port 3389 is open for incoming requests.

To avoid malicious connectivity attempts to the Worker VM public IP address, connectivity is allowed only from a pre-defined source IP address or range. This can be easily configured as the public IP address of your on-premises computer.

The Worker subnet has two routing rules in its routing table:

- All outgoing traffic is routed to Azure Firewall
- Traffic destined to the on-premises public IP address is routed directly and not through Azure Firewall.

Topology 2 – Spoke to Spoke


User configuration
L3-L7 connectivity policies
Microsoft Threat Intelligence
Known malicious IPs and FQDNs
Worker Spoke VNet
Firewall VNet
Worker Subnet
Firewall Subnet
Threat intel, NAT, network and application traffic filtering rules allows inbound/ outbound access
Azure to PC direct connectivity over the internet
Internet PC
Worker VM
Azure Firewall
Traffic is denied by default
Server Spoke VNet
Server Subnet
Server VM
©Microsoft Corporation

Deployment and Configuration

Prerequisites

To successfully deploy and configure this premium release of Azure Firewall, you need to have a valid subscription that is registered to Azure Firewall Private Preview by Microsoft. For more info please see the previous *Private Preview Participation* section.

Azure Firewall Templates

In this private preview release, the entire service configuration and deployment can be done only via ARM (Azure Resource Manager) templates.

The template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax, which lets you state what you intend to deploy without having to write the sequence of programming commands to create it. In the template, you specify the resources to deploy and the properties for those resources. More info about ARM templates is available here.

To allow quick deployment of this Azure Firewall Premium Private Preview edition, a set of samples templates are available for download here.

The following template samples files are available in this shared folder:

1. FirewallTopology1.template.json – This is the main template file which include the deployment definition of this premium firewall preview in single VNet topology
2. FirewallTopology2.template.json – This is the main template file which include the deployment definition of this premium firewall preview in spoke to spoke topology
3. FirewallPolicy.template.json – This is Azure Firewall Policy configuration parameters definition
4. CACertificate.template.json – This is a template for easy deployment of customer CA certificate in Key vault.

Important Note – For simplicity, in this guide Topology 1 with a single VNet is referenced throughout the entire document. However, the same instructions are also valid for the Topology 2 template with spoke to spoke VNets.

The Worker VM can be accessed via the Internet on RDP TCP port 3389 with authentication details specified in the FirewallTopology1.template.json file and only from a single public IP address as specified in the parameter *remoteAccessAddressPrefix*.

This template can be used as-is or modified to align with your exact topology requirements. Editing can be done via a json file editor or via the Azure portal as explained in the next section.

Template deployment

To successfully deploy your template, validate that your resource group is created with the enrolled subscription and in a private preview supported region.

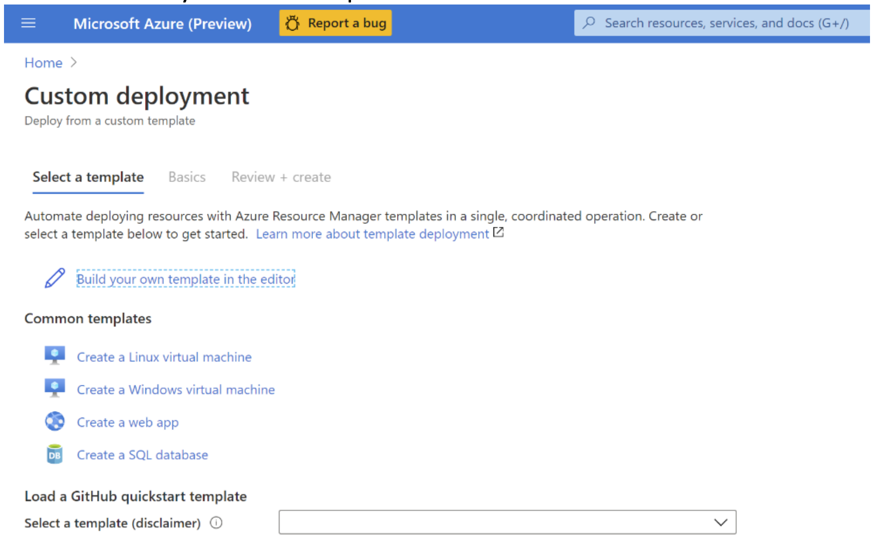
This guide provides instructions for template deployment in the two following ways :

a. Via Azure Portal
b. Via Azure Cloud Shell

Template deployment via Azure Portal

The following steps are required:

1. Go to Azure Portal Custom Deployment https://portal.azure.com/#create/Microsoft.Template
2. Select "Build your own template in the editor"



Microsoft Azure (Preview)
Report a bug
Search resources, services, and docs (G+/)
Home
Select a template
Basics
Review + create
Build your own template in the editor
Create a Linux virtual machine
Create a Windows virtual machine
Create a web app
Create a SQL database
Select a template (disclaimer)

Custom deployment

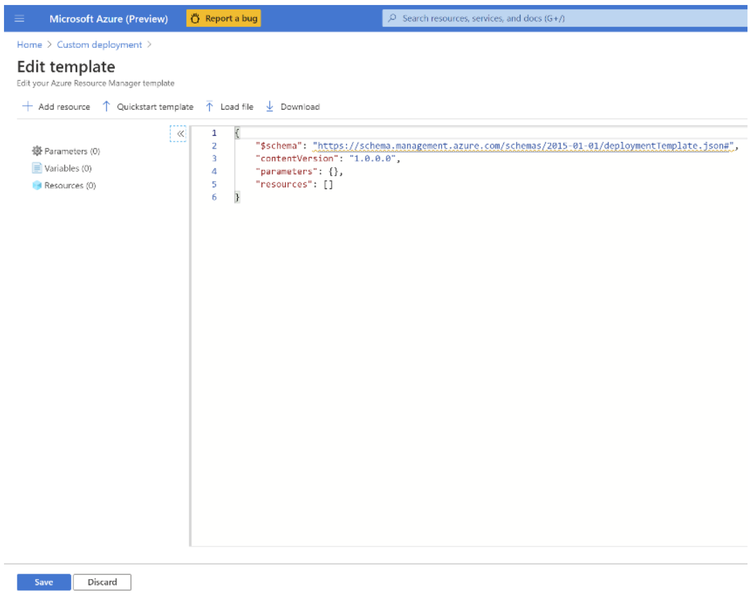
Deploy from a custom template

Automate deploying resources with Azure Resource Manager templates in a single, coordinated operation. Create or select a template below to get started. Learn more about template deployment

Common templates

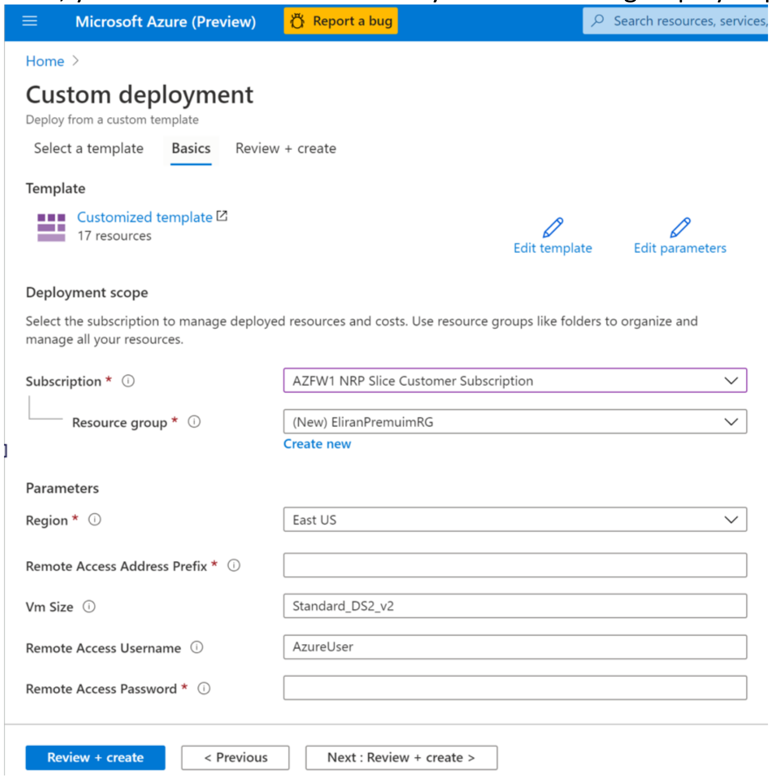
Load a GitHub quickstart template

3. Select "Load File"


Microsoft Azure (Preview)
Report a bug
Search resources, services, and docs (G+/)
Home > Custom deployment >
Edit template
Edit your Azure Resource Manager template
Add resource
Quickstart template
Load file
Download
Parameters (0)
Variables (0)
Resources (0)
Save
Discard

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {},
    "resources": []
}
```

4. In the file browser window select "FirewallTopology1.template.json" and then click save
5. Now, you can view and modified any of the following displayed parameters:



Subscription *	AZFW1 NRP Slice Customer Subscription
Resource group *	(New) EliranPremuimRG
Region *	East US
Remote Access Address Prefix *	
Vm Size	Standard_DS2_v2
Remote Access Username	AzureUser
Remote Access Password *	

6. Select your enrolled subscription ID and make sure the following 2 parameters are modified correctly:
    a. remoteAccessPassword- Set the password to allow remote access to Worker Server.
    b. remoteAccessAddressPrefix- Set your own computer public IP address, this will allow you to get access to Worker VM from the Internet. To determine your PC IP address, you can use multiple services such as https://www.whatismyip.com/. The IP address shall be in CIDR notation a.b.c.d/32. You can change your computer public IP address at any time by modifying the WorkerNSG and WorkerRoute objects in the template.
7. Finally, select **Review + Create** to finalize the deployment process.

Template deployment via Azure Cloud Shell

The following steps are required:

1. Edit the parameters json file and modify the following fields:
    a. remoteAccessPassword- Set the password to allow remote access to Worker Server.
    b. remoteAccessAddressPrefix- Set your own computer public IP address. This will allow you to get access to Worker VM from the Internet. To determine your computer IP address, you can use multiple services such as https://www.whatismyip.com/. IP address will be in CIDR notation a.b.c.d/32. You can change your computer public IP address at any time via by modifying WorkerNSG and WorkerRoute objects.
2. From Azure Portal main screen, press the Cloud Shell icon
3. When asked to select Bash or PowerShell, select Bash.
4. Once the Bash shell prompt is displayed, run the following commands:
    a. Set your subscription ID
    ```
    >az account set --subscription <your enrolled subscription ID>
    ```
    b. Create new resource group named “FirewallPremiumRG”
    ```
    >az group create --location <preferred region name> --name FirewallPremiumRG
    ```
    c. Deploy the various resources into FirewallPremiumRG
    ```
    >az deployment group create -g FirewallPremiumRG --template-file FirewallTopology1.template.json
    ```

Policy & Rules Processing

Although Azure Firewall supports both traditional rules (as part of the firewall JSON) and firewall policies, all Premium features can only be configured via Firewall Policy. For more information about Firewall Policy and its usage see Azure Firewall Manager policy overview.

Rule processing logic is the same as Azure Firewall standard. For more info see Azure Firewall rule processing logic.

TLS Inspection and URL Filtering Settings

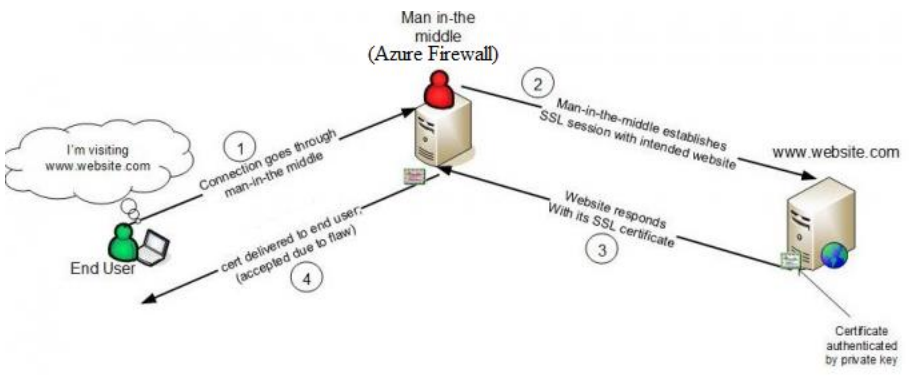
Certificates

There are 3 types of certificates that will be used in our deployment setup:

1. Intermediate CA Certificate (AKA CA Certificate) – Certificate Authority (CA) is an organization that is trusted to sign digital certificates. CA verifies identity and legitimacy of company or individual that requested a certificate and if the verification is successful, CA issues signed certificate. When server presents certificate to client (for example, your web browser) during SSL/TLS handshake, client will attempt to verify signature against a list of ‘known good’ signers. Web browsers normally come with lists of CAs that they will implicitly trust to identify hosts. If the authority is not in the list, as with some sites that sign their own certificates, the browser will alert the user that the certificate is not signed by a recognized authority and ask the user if they wish to continue communications with unverified site.
2. Server Certificate (AKA Website certificate) – A certificate associated with to specific domain name. If a website has a valid certificate, it means that a certificate authority has taken steps to verify that the web address actually belongs to that organization. When you type a URL or follow a link to a secure website, your browser will check the certificate for the following characteristics:
    - The website address matches the address on the certificate.
    - The certificate is signed by a certificate authority that the browser recognizes as a "trusted" authority.
3. Root CA Certificate (AKA root certificate) - A certificate authority can issue multiple certificates in the form of a tree structure. A root certificate is the top-most certificate of the tree.

As shown in the below diagram, Azure Premuim FW serves as man-in-the-middle and auto-generate in step 4 a server certificate for www.website.com. This certificate is generated using the CA certificate provided by the customer.

The end user must have a Root CA certificate installed in his client to ensure the server certificate is signed by trusted CA.


Man in-the middle
(Azure Firewall)
I'm visiting www.website.com
End User
1
Connection goes through man-in-the middle
2
Man-in-the-middle establishes SSL session with intended website
www.website.com
3
Website responds With its SSL certificate
4
cert delivered to end user (accepted due to flaw)
Certificate authenticated by private key

In Appendix A you can find more details on how to create your own Intermediate and Root CA certificates, in case you don't want to use your organization's CA certificates.

Azure Key Vault

Azure Key Vault is a platform-managed secret store that you can use to safeguard secrets, keys, and TLS/SSL certificates. Azure Firewall Premium supports integration with Key Vault for server certificates that are attached to Firewall Policy.

The Key Vault integration model for TLS Inspection allows you to provide a reference to an existing Key Vault certificate or secret at any given time.

After Azure Firewall is configured to use Key Vault certificates, its instances retrieve the certificate from Key Vault and install them locally for TLS Inspection. The instances also poll Key Vault at 24-hour intervals to retrieve a renewed version of the certificate, if it exists. If an updated certificate is found, the TLS/SSL certificate currently associated with the Firewall Policy is automatically rotated. For security reasons, customer CA certificate is kept in Azure Firewall memory for a short while.

Firewall Policy integration with Key Vault requires a three-step configuration process:

1. **Create a user-assigned managed identity**

   You create or reuse an existing user-assigned managed identity, which Azure Firewall uses to retrieve certificates from Key Vault on your behalf. For more information, see What is managed identities for Azure resources?. This step creates a new identity in the Azure Active Directory tenant. The identity is trusted by the subscription that's used to create the identity.

2. **Configure your key vault**

   You then need to **import** an existing certificate with its key pair into your key vault.

   Alternatively, you can also use a key vault **secret** that's stored as a password-less, base-64 encoded PFX file. A PFX file is a digital certificate containing both private key and public key.

   We recommend using a **CA certificate import** because it allows you to configure an alert based on certificate expiration date.

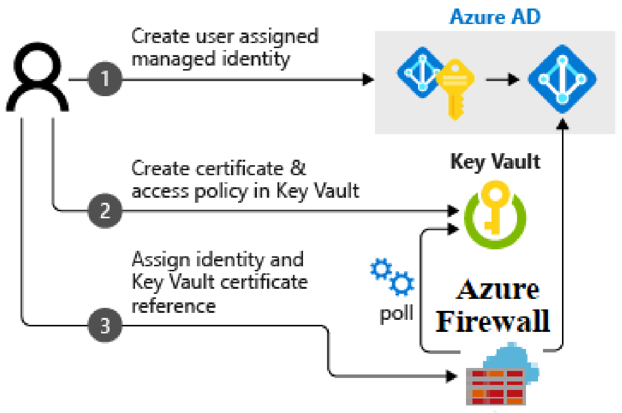
   After you've imported a certificate or a secret, you need to define access policies in the key vault to allow the identity to be granted *get* access to the certificate/secret.

   The certificate will be used by Azure Firewall to enable the outbound TLS inspection.

The provided CA certificate needs to be trusted by your workload. as explained here.

3. **Configure the Firewall Policy**

   After you complete the two preceding steps, you can set up or modify an existing Firewall policy to use the user-assigned managed identity.



Azure AD
Create user assigned managed identity
1
Create certificate & access policy in Key Vault
2
Key Vault
Assign identity and Key Vault certificate reference
3
poll
Azure Firewall

To setup successfully TLS Inspection, the following steps are required:

1. You can utilize the CACertificate template to place a CA certificate in the pre-deployed Key Vault and Managed Identity which were part of the Firewall Template. To do so, you can edit the policy template (json file) and modify the following field:
   a. caCertificate – Customer's owned CA certificate with its key pair.
      i. The embedded certificate should be a PFX file containing a single key pair with no password.
      ii. The file should be encoded in base64. See Appendix A for more details on how to create CA certificate.
   b. Deploy the key vault and certificate via the CACertificate template. Deployment can be done via Azure portal or Cloud Shell as follow:
      ```
      >az deployment group create -g FirewallPremiumRG --template-file CACertificate.template.json
      ```
2. Alternatively, you can manually place its CA certificate in Azure key vault as explained previously, or utilize an existing key vault, and then edit the policy template (json file) and modify the following three fields to allow Azure Firewall to access your key vault and retrieve the CA certificate:
   a. identity.userAssignedIdentities – Replace with your own managed identity
   b. keyVaultSecretId – your owned key vault's secret ID (Secret ID implicitly contain key vault name embedded within it)
3. Now deployment of the revised policy can be done via Azure portal or Cloud Shell as follow:
   a. Deploy the policy rules from the policy template.
      ```
      >az deployment group create -g FirewallPremiumRG --template-file FirewallPolicy.template.json
      ```

The following are the various TLS Inspection configuration parameters as they are shown in the Firewall Policy template file:

- "userAssignedIdentities" parameter to set customer's Managed Identity

```
"identity": {
    "type": "UserAssigned",
    "userAssignedIdentities": {"identityResourceId": {}}
}
```

- "keyVaultSecretId" parameter to set KeyVault secret ID

```
"transportSecurity": {
```

```
    "certificateAuthority": {
        "name": "cacert",
        "keyVaultSecretId": "secret-resource-id"
    }
}
```

- "terminateTLS" and "TargetUrls" parameters allow the customer to set TLS Inspection per Application rule and for specified target URLs

```
"rules": [
    {
        "ruleType": "ApplicationRule",
        "name": "AllowWeb",
        "protocols": [
            {
                "protocolType": "Https",
                "port": 443
            }
        ],
        "targetUrls": [
             "*google.com/maps*",
            "www.microsoft.com/en-us/surface/devices/surface-duo"
        ],
        "sourceAddresses": [
            "*"
        ],
        "terminateTLS": true
    }
]
```

In the previous application rule example, Azure Firewall will perform TLS Inspection only to traffic destined to *google.com/maps and www.microsoft.com/.../surface-duo on port 443. Additional rules can be concatenated as required.

Note – While TLS Inspection is supported only for HTTPS on port 443, URL filtering is also supported with HTTP protocol on any given port.

These Policy parameters can be modified as required and then redeployed on the existing Firewall Premium service.

Important – To successfully perform TLS Inspection with Azure Firewall, you must install its root CA certificate on the Worker VM. This will allow the browser running on Worker VM to trust the intermediate certificate that is generated by the Firewall and shared with TLS client. For more information, see the Testing section.

IDPS Settings

The IDPS service can be applied both on HTTP as well as encrypted HTTPS traffic. Once HTTPS traffic needs to be inspected, Azure Firewall can utilize its TLS Inspection capability to decrypt the traffic and reveal the malicious activity out of it. Therefore, to apply IDPS on HTTPS, a specific application rule must be defined with TLS Inspection mode set to enabled, as shown in the previous example (for example: "terminateTLS": true).

The following example shows the various IDPS configuration parameters as they are shown in the Firewall Policy template file:

- "basePolicy" parameter determine the parent policy that this policy is derived from. Remember that IDPS settings on child policy must always be sticker then IDPS setting in parent policy.

```
"properties": {
    "basePolicy": {
        "id": "base-policy-resource-id"
    }
}
```

- “mode” – IDPS mode may contain any of the following values “Alert”, “Deny” or “Off”. If the parent policy is configured for “Alert” mode, the child policy must be stricter, so it can only be configured in “Deny” mode. “Off” mode means IDPS is disabled, “Alert” means that once malicious traffic is identified the Firewall will add an entry to the log and “Deny” means Firewall will block any identified malicious traffic.

```
"intrusionDetection": {
    "mode": "Alert"
}
```

- “ signatureOverrides” parameter provide the granularity of mode setting as per the signature. Signature “mode” values are the same as IDPS “mode” values and “ID” is the identified of the specific signature that mode should be apply on. In the following example, any traffic match to signature number 2024897 will be blocked and any traffic match to signature number 2024898 will create a new entry in Firewall log.

```
"signatureOverrides": [
    {
        "id": "2024897",
        "mode": "deny"
    },
    {
        "id": "2024898",
        "mode": "Alert"
    }
]
```

- “bypassTrafficSettings” is a set of attributes that allow you to define specific traffic pattern criteria for which IDPS will not be applied. In case of inherited policies, these settings are only allowed in the parent policy. In the following example, any TCP traffic sent from 10.0.10.10 or 10.0.10.11 to 1.1.1.1:80 will bypass IDPS service.

```
"bypassTrafficSettings": [
    {
        "name": "MyRule",
        "protocol": "TCP",
        "sourceAddresses": [
            "10.0.10.10",
            "10.0.10.11"
        ],
        "destinationAddresses": [
            "1.1.1.1"
        ],
        "destinationPorts": [
            "80"
        ]
    }
]
```

Another example of “bypassTrafficSettings” utilizing the usage of IP groups:

```
"bypassTrafficSettings": [
    {
        "name": "MyBypassRule",
        "protocol": "ANY",
        "sourceIpGroups": [
            "ip-group-resource-id"
        ],
        "destinationIpGroups": [
            "ip-group-resource-id"
        ],
        "destinationPorts": [
            "*"
        ]
    }
]
```

In the below log example, a TCP request from 10.0.100.5:52919 to 169.254.169.254:80 resulted in a match with signature number 2024897 , as a result this request has been blocked by the Firewall.

```
TCP request from 10.0.100.5:52919 to 169.254.169.254:80. Action: Deny. SignatureID: 2024897. IDS: ET USER_AGENTS Go HTTP Client User-Agent. Priority: 3. Classification: Misc activity
```

In case of a false positive scenario, where a legitimate request has been blocked by the Firewall due to a signature match, you can use the signature ID from the log and add a new signature Settings rule to bypass this signature (for example, mode=”off”).

Another practical use case to utilize signature setting might be when IDPS mode is set to “Alert”, but there are one or more specific signatures that you want to block its associated traffic. In this case you might want to add new signature rules with mode=”deny”.

Azure REST API

Another interface to access and control Azure Firewall is through its REST API.

API can be used in addition or in parallel to ARM template usage as explained previously.

Any operation that can be done using an ARM template can be also achieved using Azure REST API.

A comprehensive description of how to create and update Firewall policy through REST API can be found here.

For your convenience, the following example is a detailed Azure API JSON structure for the new elements added in this premium preview.

TLS Inspection and URL Filtering

New identity definition was added to the firewall policy.

```
"FirewallPolicy": {
   "properties": {
         "x-ms-client-flatten": true,
         "$ref": "#/definitions/FirewallPolicyPropertiesFormat",
         "description": "Properties of the firewall policy."
     },
     "identity": {
       "$ref": "./network.json#/definitions/ManagedServiceIdentity",
       "description": "The identity of the firewall policy."
     },
      ...
   },
    …
}
```

A new transportSecurity definition was added to the FirewallPolicyPropertiesFormat

```
 "FirewallPolicyPropertiesFormat": {
      "properties": { … },
       "transportSecurity": {
         "description": "TLS Configuration definition.",
         "$ref": "#/definitions/FirewallPolicyTransportSecurity"
       },
         …
 },
"FirewallPolicyTransportSecurity": {
      "properties": {
        "certificateAuthority": {
          "$ref": "#/definitions/FirewallPolicyCertificateAuthority",
          "description": "The CA used for intermediate CA generation."
        },
"FirewallPolicyTrustedRootCertificate": {
      "properties": {
        "properties": {
          "x-ms-client-flatten": true,
          "$ref": "#/definitions/FirewallPolicyTrustedRootCertificatePropertiesFormat",
          "description": "Properties of the trusted root authorities."
        },
        "name": {
          "type": "string",
          "description": "Name of the trusted root certificate that is unique within a firewall policy."
        }
      },
      "description": "Trusted Root certificates of a firewall policy."
    }
 "FirewallPolicyCertificateAuthority": {
      "properties": {
        "properties": {
          "x-ms-client-flatten": true,
          "$ref": "#/definitions/FirewallPolicyCertificateAuthorityPropertiesFormat",
          "description": "Properties of the certificate authority."
        },
        "name": {
          "type": "string",
          "description": "Name of the CA certificate."
        }
      },
      "description": "Trusted Root certificates properties for tls."
}
"FirewallPolicyCertificateAuthorityPropertiesFormat": {
      "properties": {
        "keyVaultSecretId": {
          "type": "string",
          "description": "Secret Id of 'Secret' or 'Certificate' object stored in KeyVault."
        }
      },
      "description": "Trusted Root certificates properties for tls."
    },
```

The application rule definition was updated with **targetUrls, targetFqdns** and **terminateTLS**

```
 "ApplicationRule": {
       "targetFqdns": {
         "type": "array",
         "description": "List of FQDNs for this rule.",
         "items": {
           "type": "string"
         }
       },
       "targetUrls": {
         "type": "array",
         "description": "List of Urls for this rule condition.",
         "items": {
           "type": "string"
         }
       },
       "terminateTLS": {
         "type": "boolean",
         "description": "Terminate TLS connections for this rule."
       }
      }
      ….
   }
```

IDPS

```
"IntrusionDetection": {
    "Mode": {
        "$ref": "#/definitions/FirewallPolicyIntrusionSystemModeOptions",
        "description": "The general operation mode for Intrusion system."
    },
    "Configuration": {
        "SignatureOverride": {
           "type": "array",
           "description": "List of specific signature modes.",
           "items": {
                "$ref": "#/definitions/FirewallPolicyIntrusionSystemRuleSettings"
            }
        },
        "BypassTrafficSettings": {
            "type": "array",
            "description": "List of rules for traffic to bypass.",
            "items": {
                "$ref": "#/definitions/FirewallPolicyIntrusionSystemBypassTrafficSettings"
            }
        }
    }
  }
"FirewallPolicyIntrusionSystemModeOptions": {
   "type": "string",
   "enum": [
    "Off",
    "Alert",
    "Deny"
   ]
}
"FirewallPolicyIntrusionSystemRuleSettings": {
    "SignatureId": {
        "type": "string",
        "description": "The rule sid."
    },
    "Mode": {
        "$ref": "#/definitions/FirewallPolicyIntrusionSystemModeOptions",
        "description": "The signature mode."
    }
  }

"FirewallPolicyIntrusionSystemBypassTrafficSettings": {
    "Name": {
        "type": "string",
        "description": "Name of the ignored traffic rule."
    },
   "Description": {
      "type": "string",
      "description": "Description of the ignored traffic rule."
    },
    "Protocol": {
        "type": "string",
        "description": "Possible intrusion system ignored traffic protocols.",
        "enum": [
            "TCP",
            "UDP",
            "ICMP",
            "ANY"
        ]
    } ,
    "SourceAddresses": {
        "type": "array",
        "description": "List of source IP addresses or ranges for this rule.",
        "items": {
            "type": "string"
        }
    },
    "DestinationAddresses": {
        "type": "array",
        "description": "List of destination IP addresses or ranges for this rule.",
        "items": {
            "type": "string"
        }
    },
   "DestinationPorts": {
       "type": "array",
        "description": "List of destination ports or ranges.",
        "items": {
             "type": "string"
        }
     },
    "SourceIpGroups": {
      "type": "array",
       "description": "List of source IpGroups for this rule.",
       "items": {
            "type": "string"
        }
    },
    "DestinationIpGroups": {
           "type": "array",
          "description": "List of destination IpGroups for this rule.",
         "items": {
               "type": "string"
         }
   }
}
```

Uninstallation Flow

This uninstallation flow can be followed in one of the following two ways:

1. Removing the service including all configurations:
   Clean up firewall resources is very easy and straight forward. Just delete the FirewallPremiumRG resource group to delete all firewall-related resources.

   Please also delete the key vault that was created, its name should have the prefix “fw-premium-“ followed by your resource group id number.

2. Removing the service, but saving its configuration:
   If you wish to keep your entire service configuration, you should export the required resources into a template, so you can easily deploy them at any time.
   Review this article to understand how to export your resources.
   Remember to delete your resources once they are no longer needed.

Testing

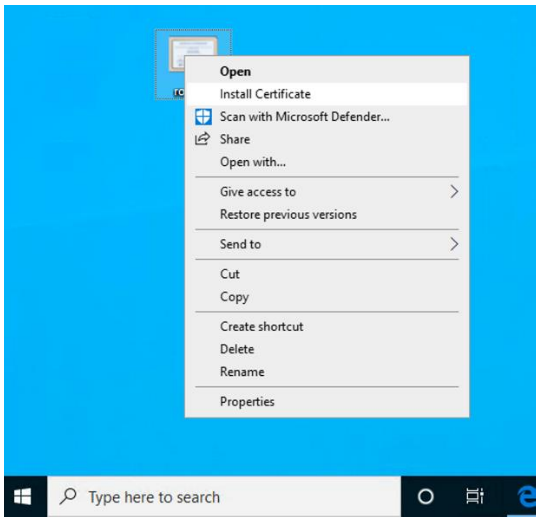
Microsoft recommends using the suggested Topology 2 (spoke to spoke) with the provided template samples for a complete look-a-like environment that will best mimic a real-life deployment.

Installing Root CA Certificate in Worker VM

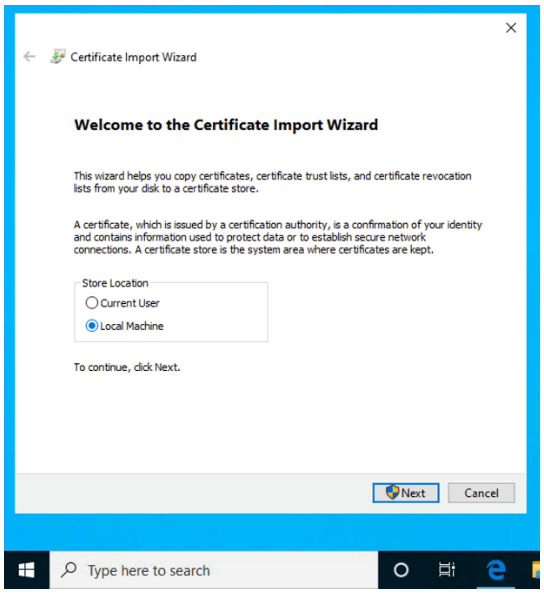
To successfully perform TLS Inspection on Azure Firewall, you must install its root CA certificate on the Worker VM. This will allow the browser running on the Worker VM to trust the intermediate certificate that is generated by the Firewall and shared with TLS client.

To do so, follow these steps:

1. Place the root CA certificate on any folder on the Worker VM.
2. Right-click on the certificate file and choose “Install Certificate”



3. Choose “Local Machine” for store location and select **Next**.


Certificate Import Wizard

Welcome to the Certificate Import Wizard

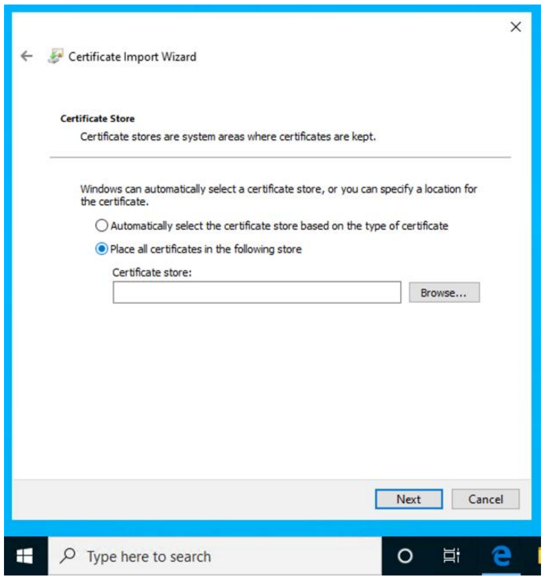
This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

Store Location
( ) Current User
(•) Local Machine

To continue, click Next.

4. Choose certificate store as shown below and then select **Next**.

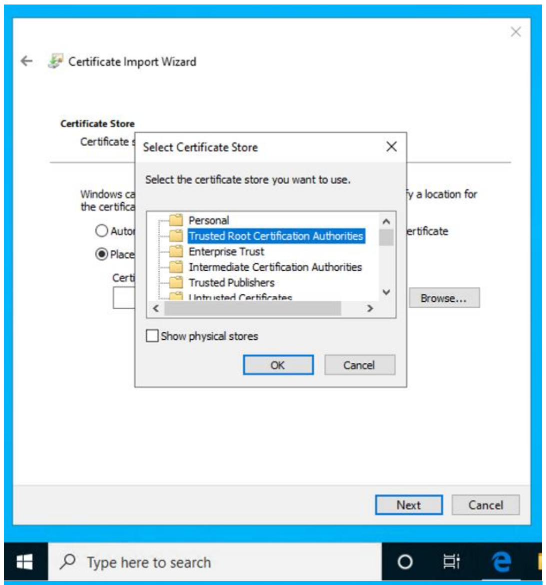

Certificate Import Wizard

Certificate Store
Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

( ) Automatically select the certificate store based on the type of certificate
(•) Place all certificates in the following store

Certificate store:

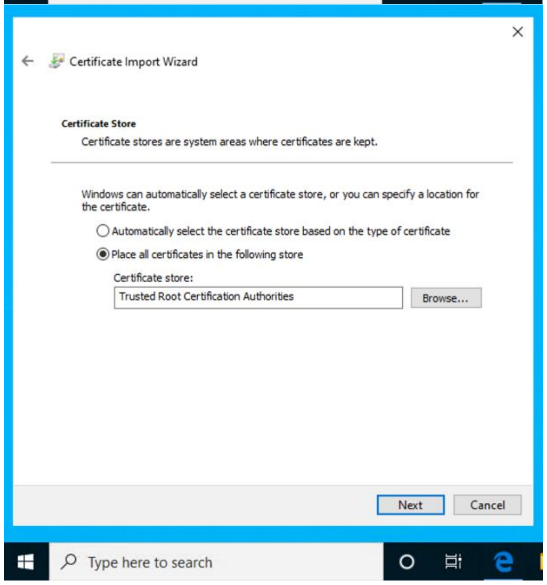

Certificate Import Wizard

Select Certificate Store

Select the certificate store you want to use.

- Personal
- Trusted Root Certification Authorities
- Enterprise Trust
- Intermediate Certification Authorities
- Trusted Publishers
- Untrusted Certificates

☐ Show physical stores


Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

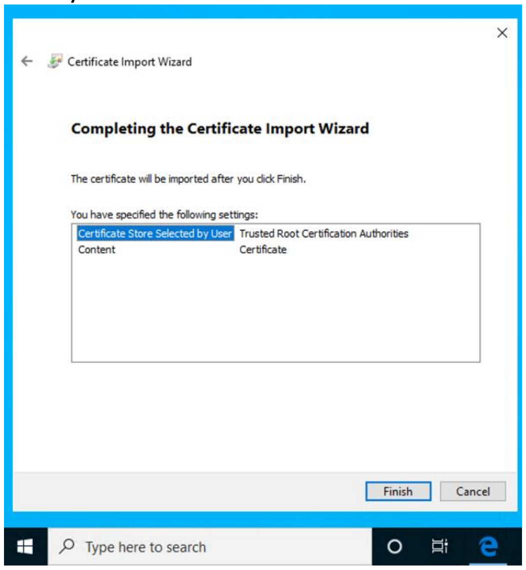
○ Automatically select the certificate store based on the type of certificate

● Place all certificates in the following store

Certificate store:

Trusted Root Certification Authorities

5. Finally select **Finish.**


Certificate Import Wizard

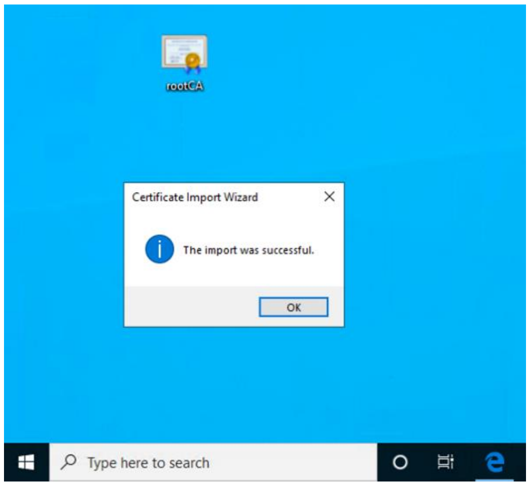
Completing the Certificate Import Wizard

The certificate will be imported after you click Finish.

You have specified the following settings:

Certificate Store Selected by User	Trusted Root Certification Authorities
Content	Certificate

6. You should get an import status message as shown below:



Recommended testing tools

For IDPS testing, Microsoft recommend the following two approaches to simulate malicious traffic:

1. CURL – CURL allows you to control the various HTTP headers and simulate malicious requests. For example, the following curl command will create a match with IDPS signature for malicious user agent:
   >curl -A "BlackSun" http://www.my-example.com
2. Penetration Testing Framework – You can use any selected framework to simulate various types of malicious request. Metasploit is an example of such vendor that offer both open source and licensed frameworks.

Suggested testing use cases

It is recommended to test the following main use cases:

1. TLS Inspection with URL Filtering:
   a. Create a new application rule with terminateTLS=enabled and allowed targetURLs= www.nytimes.com/section/world
   b. Open a browser on the Worker VM and go to https://www.nytimes.com/section/world and validate that the HTML response is displayed as expected in the browser
   c. In Azure portal, you can view the entire URL in the Monitoring logs

   ```
   {"msg":"HTTPS request from 10.0.10.10:56907 to www.nytimes.com:443. Url: www.nytimes.com/section/world. Action: Allow. Rule Collection: PolicyRules_AllowApplication. Rule: AllowWeb"}
   ```

   d. Some HTML pages may look incomplete since they are referring to other URLs which are denied. To solve this issue, the following approach should be taken:
      i. If the HTML page contain links to other domains, then you need to add these domains to a new application rule with allow access to these FQDNs.
      ii. If the HTML page contain links to sub URLs then you can modify the rule and add Asterix to the URL, for example: targetURLs=www.nytimes.com/section/world* ;
      Alternatively , you can add a new URL to the rule, for example: targetURLs=www.nytimes.com/section/world, www.nytimes.com/section/world/*

2. IDPS for plain text HTTP:
   a. In the policy settings, set "intrusionDetection" mode to 'Alert'
   b. To simulate a malicious request, open a Windows command prompt by typing 'cmd' in the search bar.
   c. Type the following command:

   ```
   C:\Users\AzureUser>curl -A "BlackSun" http://server.2020-private-preview.com
   ```

   d. As a result, the following HTTP response shall be displayed on your screen:

   ```
   <html>
       <body>
           Hello World
       </body>
   ```

```
</html>
```

e. Go to Monitor logs in Azure Portal and identify the below alert message:

```
{"msg":"TCP request from 10.0.100.5:16036 to 10.0.20.10:80. Action: Alert. Rule: 2008983. IDS: ET
USER_AGENTS Suspicious User Agent (BlackSun). Priority: 1. Classification: A Network Trojan was
detected"}
```

f. Set 'signatureSettings' to deny any traffic that match signature 2008983

g. Type the following command again:
```
C:\Users\AzureUser>curl -A "BlackSun" http://server.2020-private-preview.com
```

h. Since the HTTP request is now blocked by Azure Firewall, you should expect to see the following CLI output after connection timeout expires:
```
read tcp 10.0.100.5:55734->10.0.20.10:80: read: connection reset by peer
```

i. Go to Monitor logs in Azure Portal and identify the message for the blocked request.

j. Now we would like to bypass IDPS function, so set 'bypassTrafficSettings' to include 'sourceAddress'= 10.0.10.10 which is workerVM private IP address.

k. Type the following command from the Worker VM CLI:
```
C:\Users\AzureUser>curl -A "BlackSun" http://server.2020-private-preview.com
```

l. You should expect to get a "Hello World" HTTP response and no log alert.

3. IDPS for HTTPS Traffic
   a. To allow IDPS in HTTPS a specific application rule can be configured with terminateTLS=enabled and targetURLs= server.2020-private-preview.com
   b. You can repeat the previous test using HTTPS and you should expect the same results. The following command can be used:
   ```
   C:\Users\AzureUser>curl -A "BlackSun" https://server.2020-private-preview.com
   ```

Note: The Server VM image contains a pre-installed NGINX that is fully capable to answer both HTTP/S requests via these FQDNs: http://server.2020-private-preview.com/ https://server.2020-private-preview.com/ . The web server response for both HTTP and HTTPS is "HelloWorld". This webserver is available only from within the testing perimeter e.g. Worker VM.

Monitoring

You can monitor Azure Firewall using firewall logs and you can also use activity logs to audit operations on Azure Firewall resources. Detailed and comprehensive article can be found here.

To view Azure Firewall logs, you should enable first diagnostic logging through the Azure portal.

It can take a few minutes for the data to appear in your logs after you complete this procedure to turn on diagnostic logging.

Logs

The following new logs are available as part of this private preview release:

1. TLS Inspection with URL Filtering: This is a log example of HTTPS request from 10.0.10.10:52344 to allowed URL www.google.com/maps
   ```
   {"msg":"HTTPS request from 10.0.10.10:52344 to www.google.com:443. Url: www.google.com/maps. Action: Allow. Rule
   Collection: PolicyRules_AllowApplication. Rule: AllowWeb"}
   ```

2. IDPS signature match: This is a log example of a TCP request from10.0.100.5:52919 to 169.254.169.254:80 resulted in a match with signature number 2024897, as a result this request has been blocked by the Firewall.

```
TCP request from 10.0.100.5:52919 to 169.254.169.254:80. Action: Deny. SignatureID: 2024897. IDS: ET USER_AGENTS Go HTTP Client User-Agent. Priority: 3. Classification: Misc activity
```

Metrics

New Metrics for Firewall Premium will be available either in public preview or GA release.

Known Issues & Limitations

The following are known issues/limitations in Azure Firewall Premium Private Preview Edition.

Issue	Description	Mitigation
TLS Inspection supported only on HTTPS standard port	TLS Inspection supports HTTPS/443 only	None. Other ports/protocols to be added by public preview
ESNI support for FQDN resolution in HTTPS	Encrypted SNI is not supported in HTTPS handshake	Today only Firefox supports ESNI through custom configuration. Suggested Mitigation is to use another browser.
TLS 1.3	TLS 1.3 connection from the client are not supported. Terminated connection to remote server will use TLS 1.3 where applicable.	None. SSL client will automatically negotiate TLS 1.2. Public preview will support client connection with TLS 1.3.
TLS 1.0 & 1.1	TLS 1.0 & 1.1 are being deprecated and won't be supported. TLS 1.0 & 1.1 versions of TLS/Secure Sockets Layer (SSL) have been found to be vulnerable and while they still currently work to allow backwards compatibility, they are not recommended.	Shift to TLS 1.2
PaaS Inbound Traffic	The TLS inspection will not be able to generate valid certificates as internet clients do not trust the customers CA.	None
PaaS Outgoing Traffic	As the PaaS underlying software is out of reach, they will not have the trusted CA installed on them. Hence, any connectivity to the internet will fail with certificate trust violation.	None
Untrusted Certificates	Users may connect from time to time to servers with untrusted certificates. Firewall will drop the connection as if the server terminated the connection	None
Client Certificates (TLS)	Client certificates are used to build a mutual identity trust between the client and the server. Client certificates are used during a TLS negotiation. Azure firewall re-negotiates a connection with the server and has no access to the private key of the client certificates.	TLS inspection can be disabled on list of specified FQDNs.

QUIC/HTTP3	QUIC is the new major version of HTTP. It is a UDP based protocol over 80 (PLAN) and 443 (SSL). FQDN/URL/TLS inspection will not be supported.	Configure passing UDP 80/443 as network rules.
Untrusted customer signed certificates	Customer signed certificates are not trusted by the Firewall once received from Intranet based web server	Planned to be fixed in public preview
WebSocket	WebSockets are not supported in this private preview	Will be supported in Public Preview
Wrong Source and Destination IP in Alerts for IDPS with TLS inspection	When enabling TLS inspection and IDPS issue new alert, the displayed source/destination IP is wrong (internal IP is displayed instead of the original IP)	Will be supported in Public Preview
Wrong Source IP in Alerts for IDPS for HTTP (**with**out **TLS inspection**)	When plain text HTTP traffic is in use and IDPS issue new alert and destination is public IP, the displayed source IP is wrong (internal IP is displayed instead of the original IP)	Microsoft recommend not to send malicious traffic to public sites; therefore, an internal web site is available on ServerVM for this purpose, hence source IP will be displayed correctly in the log alerts. Will be supported in Public Preview
Forced Tunneling with IDPS	Forced tunneling is not supported with IDPS in alert mode.	Will be supported in Public Preview
HTTP URL Filtering	URL Filtering is not supporting HTTP but only HTTPS	Will be supported in Public Preview
Certificate Propagation	After CA certificate is applied on a firewall, it may take between 5-10 minutes for the certificate to take effect.	Planned to be fixed in public preview

Appendix A – Creating your own CA Certificates

The following linux bash script will allow you to easily create your own CA certificate and root certificate to be used in this private preview evaluation.

First, please create in your linux system an empty directory and place these two files cert.sh and openssl.cnf.

cert.sh file:

```
#!/bin/bash


# Create root CA

openssl req -x509 -new -nodes -newkey rsa:4096 -keyout rootCA.key -sha256 -days 1024 -out rootCA.crt -subj "/C=US/ST=US/O=Self Signed/CN=Self Signed Root CA" -config openssl.cnf -extensions rootCA_ext


# Create intermediate CA request

openssl req -new -nodes -newkey rsa:4096 -keyout interCA.key -sha256 -out interCA.csr -subj "/C=US/ST=US/O=Self Signed/CN=Self Signed Intermediate CA"


# Sign on the intermediate CA

openssl x509 -req -in interCA.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out interCA.crt -days 1024  -extfile openssl.cnf -extensions interCA_ext


# Export the intermediate CA into PFX

openssl pkcs12 -export -out interCA.pfx -inkey interCA.key -in interCA.crt -password "pass:"


# Convert the PFX into base64

cat interCA.pfx | base64 > interCA.pfx.base64


echo ""

echo "================"

echo "Successfully generated root and intermediate CA certificates"

echo "   - rootCA.crt/rootCA.key - Root CA public certificate and private key"

echo "   - interCA.crt/interCA.key - Intermediate CA public certificate and private key"

echo "   - interCA.pfx.base64 - Intermediate CA pkcs12 package to be consumed by CACertificate template"

echo "================"
```

openssl.cnf file:

```
[ req ]
default_bits        = 4096
distinguished_name  = req_distinguished_name
string_mask         = utf8only
default_md          = sha512

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
stateOrProvinceName             = State or Province Name
localityName                    = Locality Name
0.organizationName              = Organization Name
organizationalUnitName          = Organizational Unit Name
commonName                      = Common Name
emailAddress                    = Email Address

[ rootCA_ext ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ interCA_ext ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:1
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

Then, execute the cert.sh bash script , you should be able to see the following example output:

```
./cert.sh

Generating a 4096 bit RSA private key

................................................++

..........................................++

writing new private key to 'rootCA.key'

-----

Generating a 4096 bit RSA private key

............................++

.......++

writing new private key to 'interCA.key'

-----

Signature ok

subject=/C=US/ST=US/O=Self Signed/CN=Self Signed Intermediate CA

Getting CA Private Key


================

Successfully generated root and intermediate CA certificates

  - rootCA.crt/rootCA.key - Root CA public certificate and private key

  - interCA.crt/interCA.key - Intermediate CA public certificate and private key

  - interCA.pfx.base64 - Intermediate CA pkcs12 package to be consumed by CACertificate template

================
```

interCA.pfx.base content should be used as an input parameter for the CAcertificate template – This is actually a PFX of CA certificate without a password encoded in base64.

rootCA.crt is root CA certificate which should be installed in Worker VM as explained here.

Thank you! Your participation is a vital part of our Cloud + AI Security product development process.

Microsoft