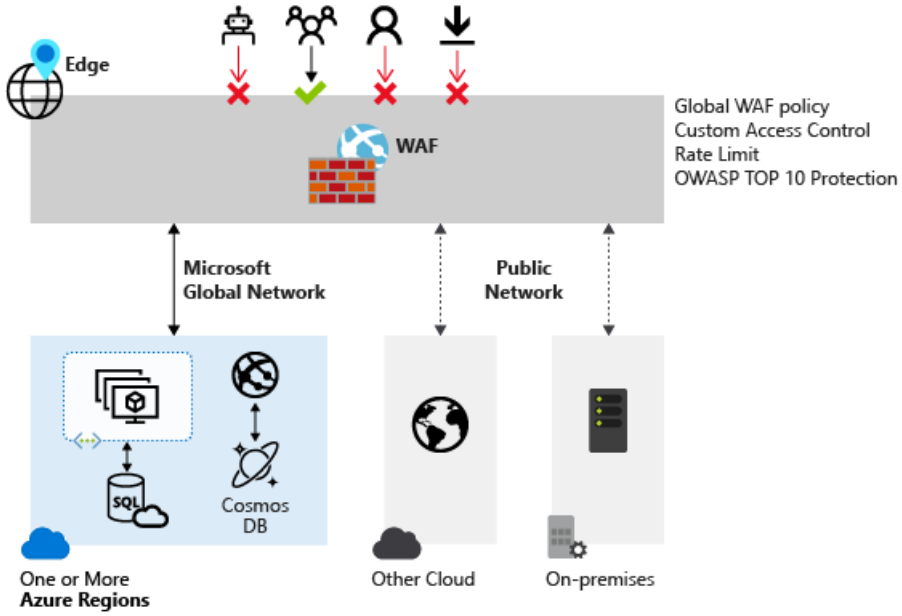
Contents

What is Azure Web Application Firewall?

11/4/2019 • 2 minutes to read • Edit Online

Web Application Firewall (WAF) provides centralized protection of your web applications from common exploits and vulnerabilities. Web applications are increasingly targeted by malicious attacks that exploit commonly known vulnerabilities. SQL injection and cross-site scripting are among the most common attacks.



Preventing such attacks in application code is challenging. It can require rigorous maintenance, patching, and monitoring at multiple layers of the application topology. A centralized web application firewall helps make security management much simpler. A WAF also gives application administrators better assurance of protection against threats and intrusions.

A WAF solution can react to a security threat faster by centrally patching a known vulnerability, instead of securing each individual web application.

WAF can be deployed with Azure Application Gateway and Azure Front Door Service. Currently, WAF has features that are customized for each specific service. For more information about WAF features for each service, see the overview for each service.

Next steps

- For more information about Web Application Firewall on Application Gateway, see Web Application Firewall on Azure Application Gateway.
- For more information about Web Application Firewall on Azure Front Door Service see Web Application Firewall on Azure Front Door Service.

Azure Web Application Firewall on Azure Application Gateway

11/22/2019 • 8 minutes to read • Edit Online

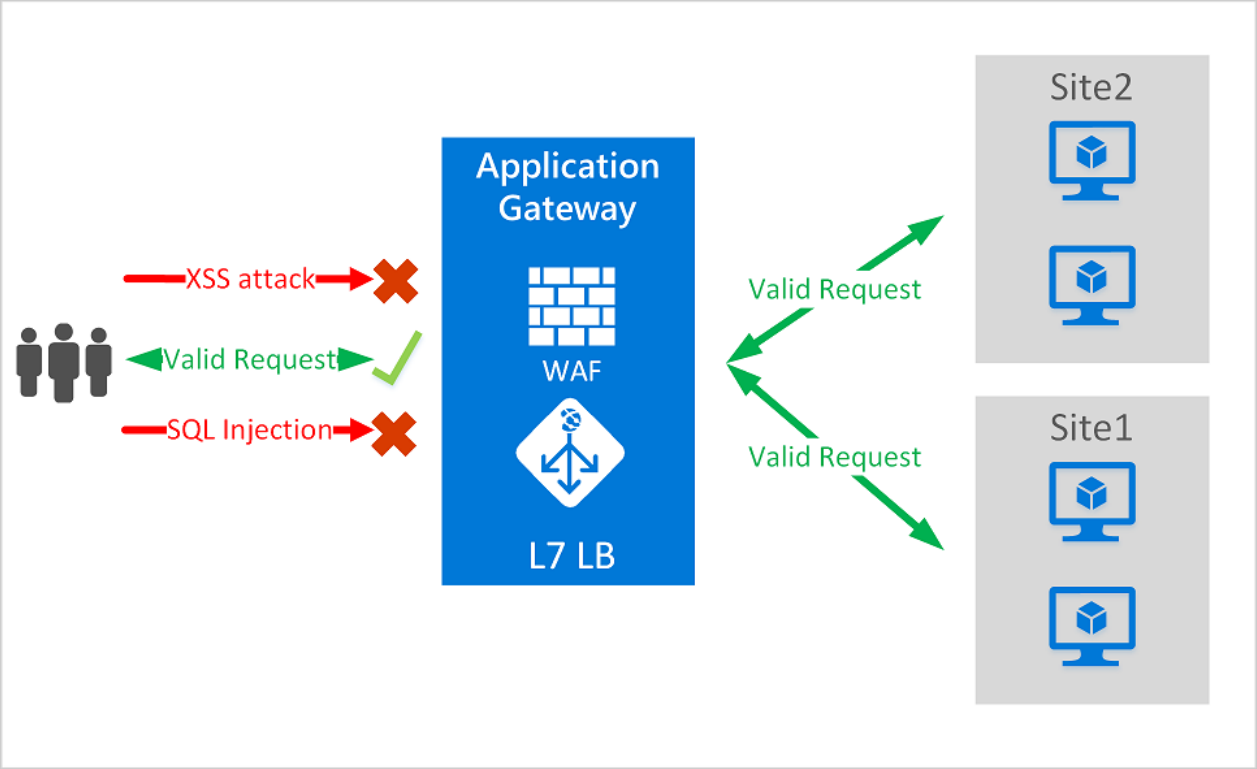
Azure Web Application Firewall (WAF) on Azure Application Gateway provides centralized protection of your web applications from common exploits and vulnerabilities. Web applications are increasingly targeted by malicious attacks that exploit commonly known vulnerabilities. SQL injection and cross-site scripting are among the most common attacks.

WAF on Application Gateway is based on Core Rule Set (CRS) 3.1, 3.0, or 2.2.9 from the Open Web Application Security Project (OWASP). The WAF automatically updates to include protection against new vulnerabilities, with no additional configuration needed.

All of the WAF features listed below exist inside of a WAF Policy. You can create multiple policies, and they can be associated with an Application Gateway, to individual listeners, or to path-based routing rules on an Application Gateway. This way, you can have separate policies for each site behind your Application Gateway if needed. For more information on WAF Policies, see Create a WAF Policy.

NOTE

Per-site and per-URI WAF Policies are in Public Preview. That means this feature is subject to Microsoft's Supplemental Terms of Use. For more information, see Supplemental Terms of Use for Microsoft Azure Previews.



Application Gateway operates as an application delivery controller (ADC). It offers Secure Sockets Layer (SSL) termination, cookie-based session affinity, round-robin load distribution, content-based routing, ability to host multiple websites, and security enhancements.

Application Gateway security enhancements include SSL policy management and end-to-end SSL support. Application security is strengthened by WAF integration into Application Gateway. The combination protects

your web applications against common vulnerabilities. And it provides an easy-to-configure central location to manage.

Benefits

This section describes the core benefits that WAF on Application Gateway provides.

Protection

- Protect your web applications from web vulnerabilities and attacks without modification to back-end code.
- Protect multiple web applications at the same time. An instance of Application Gateway can host of up to 40 websites that are protected by a web application firewall.
- Create custom WAF policies for different sites behind the same WAF
- Protect your web applications from malicious bots with the IP Reputation ruleset (preview)

Monitoring

- Monitor attacks against your web applications by using a real-time WAF log. The log is integrated with Azure Monitor to track WAF alerts and easily monitor trends.
- The Application Gateway WAF is integrated with Azure Security Center. Security Center provides a central view of the security state of all your Azure resources.

Customization

- Customize WAF rules and rule groups to suit your application requirements and eliminate false positives.
- Associate a WAF Policy for each site behind your WAF to allow for site-specific configuration
- Create custom rules to suit the needs of your application

Features

- SQL-injection protection.
- Cross-site scripting protection.
- Protection against other common web attacks,
  such as command injection, HTTP request smuggling, HTTP response splitting, and remote file inclusion.
- Protection against HTTP protocol violations.
- Protection against HTTP protocol anomalies, such as missing host user-agent and accept headers.
- Protection against crawlers and scanners.
- Detection of common application misconfigurations (for example, Apache and IIS).
- Configurable request size limits with lower and upper bounds.
- Exclusion lists let you omit certain request attributes from a WAF evaluation. A common example is Active Directory-inserted tokens that are used for authentication or password fields.
- Create custom rules to suit the specific needs of your applications.
- Geo-filter traffic to allow or block certain countries from gaining access to your applications. (preview)
- Protect your applications from bots with the bot mitigation ruleset. (preview)

WAF Policy

To enable a Web Application Firewall on an Application Gateway, you must create a WAF Policy. This Policy is where all of the managed rules, custom rules, exclusions, and other customizations such as file upload limit exist.

Core rule sets

Application Gateway supports three rule sets: CRS 3.1, CRS 3.0, and CRS 2.2.9. These rules protect your web

applications from malicious activity.

For more information, see Web application firewall CRS rule groups and rules.

Custom rules

Application Gateway also supports custom rules. With custom rules, you can create your own rules, which are evaluated for each request that passes through WAF. These rules hold a higher priority than the rest of the rules in the managed rule sets. If a set of conditions is met, an action is taken to allow or block.

The geomatch operator is now available in public preview for custom rules. Please see geomatch custom rules for more information.

> **NOTE**
> The geomatch operator for custom rules is currently in public preview and is provided with a preview service level agreement. Certain features may not be supported or may have constrained capabilities. See the Supplemental Terms of Use for Microsoft Azure Previews for details.

For more information on custom rules, see Custom Rules for Application Gateway.

Bot Mitigation (preview)

A managed Bot protection rule set can be enabled for your WAF to block or log requests from known malicious IP addresses, alongside the managed ruleset. The IP addresses are sourced from the Microsoft Threat Intelligence feed. Intelligent Security Graph powers Microsoft threat intelligence and is used by multiple services including Azure Security Center.

> **NOTE**
> Bot protection rule set is currently in public preview and is provided with a preview service level agreement. Certain features may not be supported or may have constrained capabilities. See the Supplemental Terms of Use for Microsoft Azure Previews for details.

If Bot Protection is enabled, incoming requests that match Malicious Bot's client IPs are logged in the Firewall log, see more information below. You may access WAF logs from storage account, event hub, or log analytics.

WAF modes

The Application Gateway WAF can be configured to run in the following two modes:

- **Detection mode**: Monitors and logs all threat alerts. You turn on logging diagnostics for Application Gateway in the **Diagnostics** section. You must also make sure that the WAF log is selected and turned on. Web application firewall doesn't block incoming requests when it's operating in Detection mode.
- **Prevention mode**: Blocks intrusions and attacks that the rules detect. The attacker receives a "403 unauthorized access" exception, and the connection is closed. Prevention mode records such attacks in the WAF logs.

> **NOTE**
> It is recommended that you run a newly deployed WAF in Detection mode for a short period of time in a production environment. This provides the opportunity to obtain firewall logs and update any exceptions or custom rules prior to transition to Prevention mode. This can help reduce the occurrence of unexpected blocked traffic.

Anomaly Scoring mode

OWASP has two modes for deciding whether to block traffic: Traditional mode and Anomaly Scoring mode.

In Traditional mode, traffic that matches any rule is considered independently of any other rule matches. This mode is easy to understand. But the lack of information about how many rules match a specific request is a limitation. So, Anomaly Scoring mode was introduced. It's the default for OWASP 3.*x*.

In Anomaly Scoring mode, traffic that matches any rule isn't immediately blocked when the firewall is in Prevention mode. Rules have a certain severity: *Critical*, *Error*, *Warning*, or *Notice*. That severity affects a numeric value for the request, which is called the Anomaly Score. For example, one *Warning* rule match contributes 3 to the score. One *Critical* rule match contributes 5.

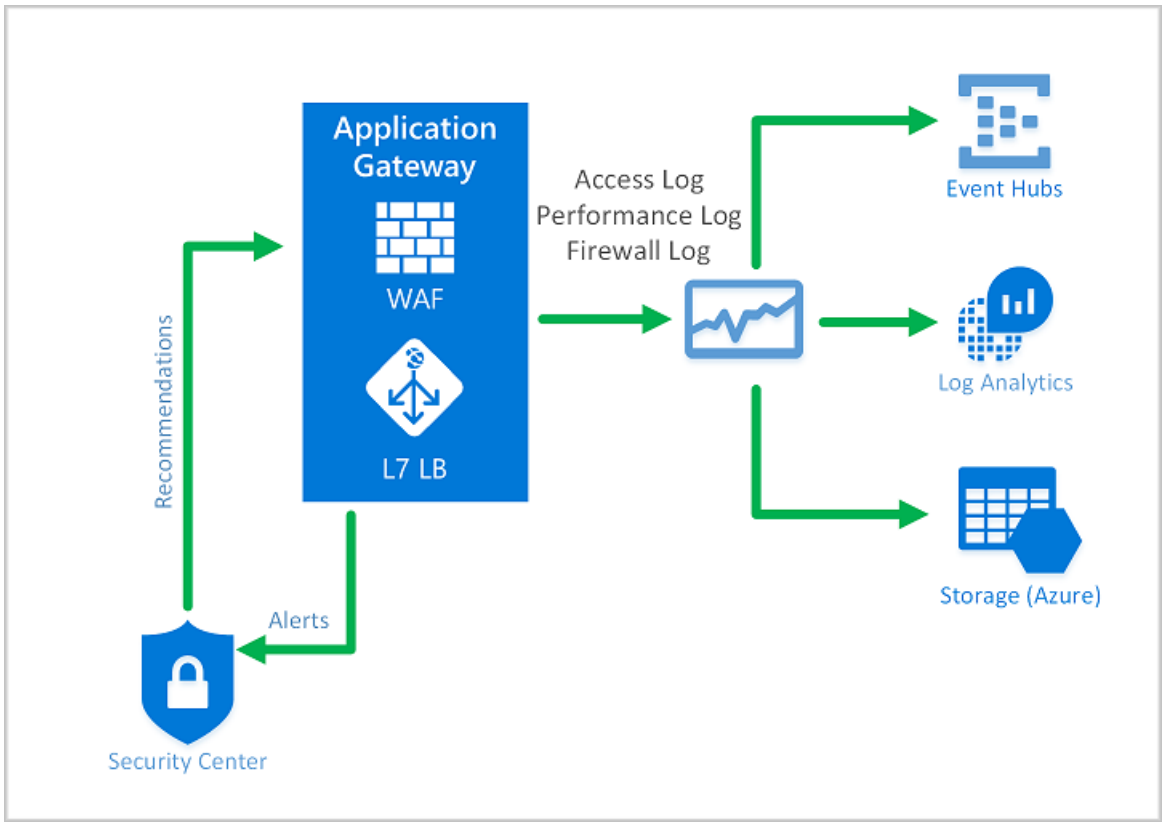
SEVERITY	VALUE
Critical	5
Error	4
Warning	3
Notice	2

There's a threshold of 5 for the Anomaly Score to block traffic. So, a single *Critical* rule match is enough for the Application Gateway WAF to block a request, even in Prevention mode. But one *Warning* rule match only increases the Anomaly Score by 3, which isn't enough by itself to block the traffic.

> **NOTE**
> The message that's logged when a WAF rule matches traffic includes the action value "Blocked." But the traffic is actually only blocked for an Anomaly Score of 5 or higher.

WAF monitoring

Monitoring the health of your application gateway is important. Monitoring the health of your WAF and the applications that it protects are supported by integration with Azure Security Center, Azure Monitor, and Azure Monitor logs.

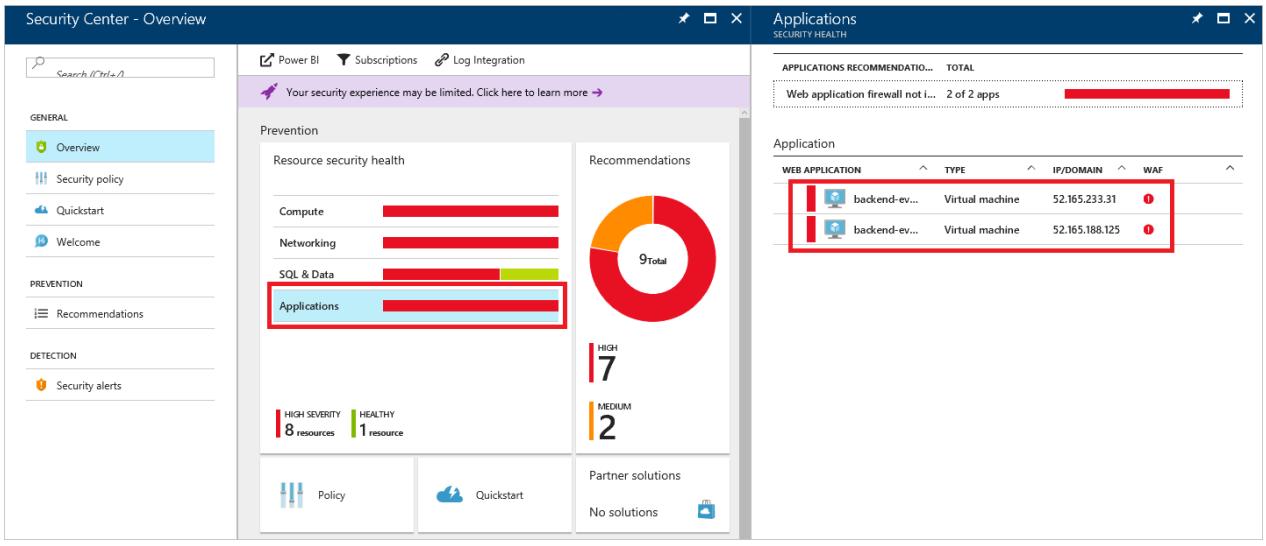


Azure Monitor

Application Gateway logs are integrated with Azure Monitor. This allows you to track diagnostic information, including WAF alerts and logs. You can access this capability on the **Diagnostics** tab in the Application Gateway resource in the portal or directly through Azure Monitor. To learn more about enabling logs, see Application Gateway diagnostics.

Azure Security Center

Security Center helps you prevent, detect, and respond to threats. It provides increased visibility into and control over the security of your Azure resources. Application Gateway is integrated with Security Center. Security Center scans your environment to detect unprotected web applications. It can recommend Application Gateway WAF to protect these vulnerable resources. You create the firewalls directly from Security Center. These WAF instances are integrated with Security Center. They send alerts and health information to Security Center for reporting.

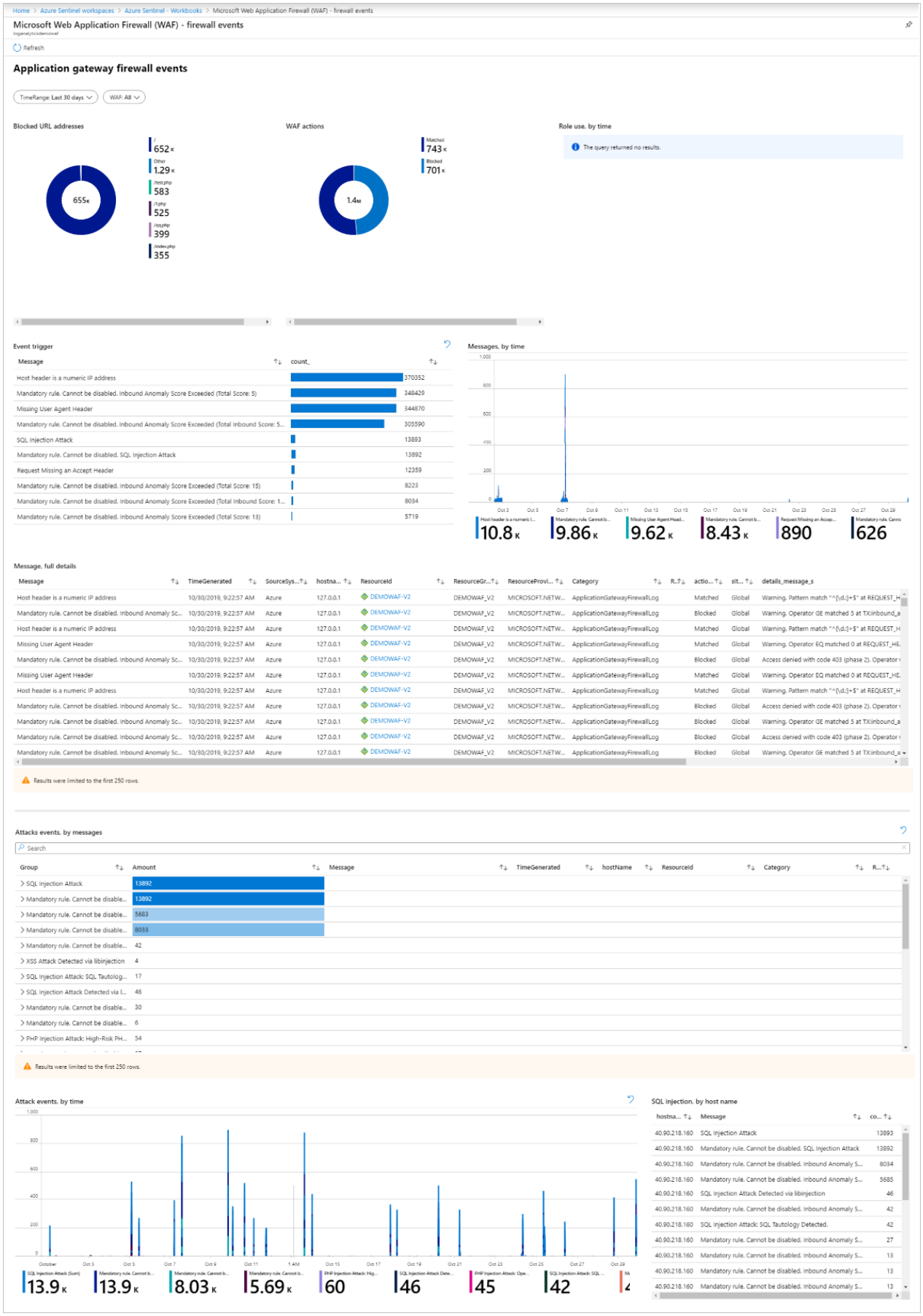


Azure Sentinel

Microsoft Azure Sentinel is a scalable, cloud-native, security information event management (SIEM) and security orchestration automated response (SOAR) solution. Azure Sentinel delivers intelligent security analytics and threat intelligence across the enterprise, providing a single solution for alert detection, threat visibility, proactive

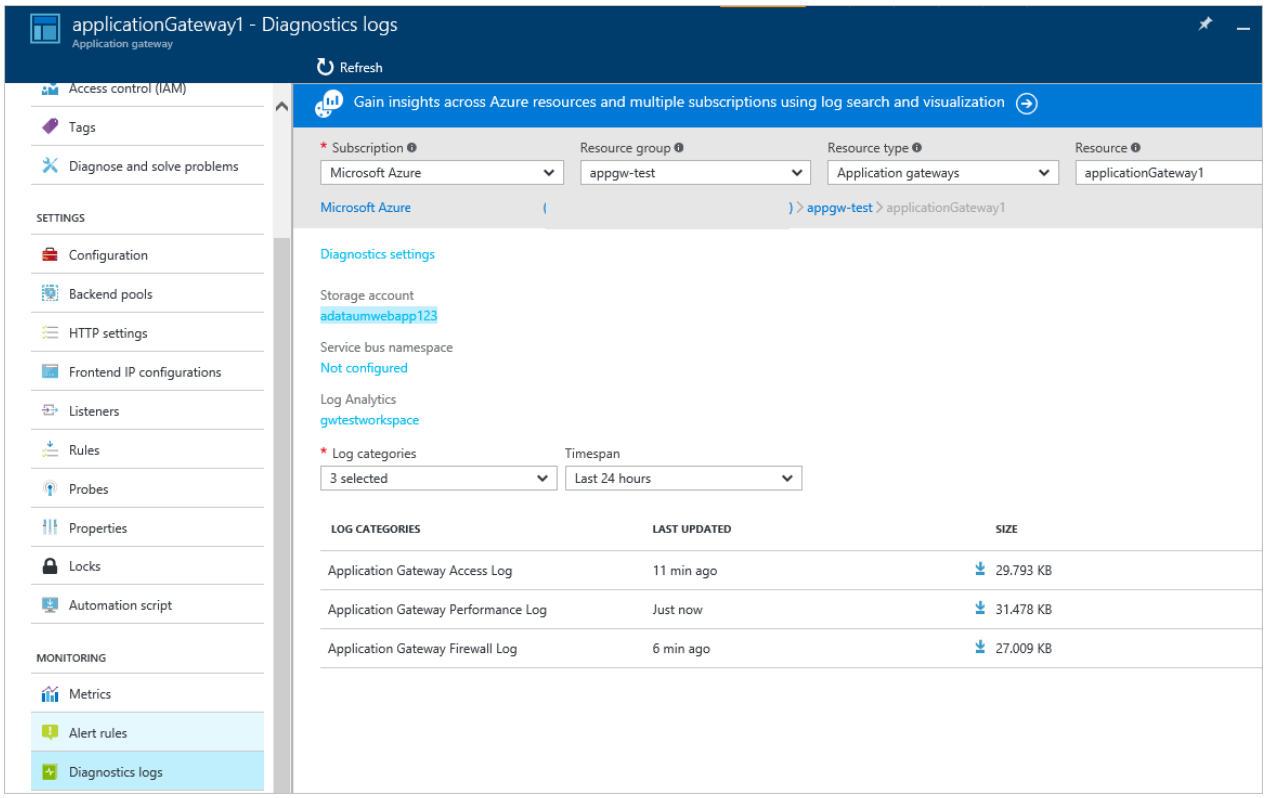
hunting, and threat response.

With the built-in Azure WAF firewall events workbook, you can get an overview of the security events on your WAF. This includes events, matched and blocked rules, and everything else that gets logged in the firewall logs. See more on logging below.



Logging

Application Gateway WAF provides detailed reporting on each threat that it detects. Logging is integrated with Azure Diagnostics logs. Alerts are recorded in the .json format. These logs can be integrated with Azure Monitor logs.



```
{
  "resourceId":
"/SUBSCRIPTIONS/{subscriptionId}/RESOURCEGROUPS/{resourceGroupId}/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATE
WAYS/{appGatewayName}",
  "operationName": "ApplicationGatewayFirewall",
  "time": "2017-03-20T15:52:09.1494499Z",
  "category": "ApplicationGatewayFirewallLog",
  "properties": {
    {
      "instanceId": "ApplicationGatewayRole_IN_0",
      "clientIp": "52.161.109.145",
      "clientPort": "0",
      "requestUri": "/",
      "ruleSetType": "OWASP",
      "ruleSetVersion": "3.0",
      "ruleId": "920350",
      "ruleGroup": "920-PROTOCOL-ENFORCEMENT",
      "message": "Host header is a numeric IP address",
      "action": "Matched",
      "site": "Global",
      "details": {
        "message": "Warning. Pattern match \"^[\\\\d.:]+$\" at REQUEST_HEADERS:Host ....",
        "data": "127.0.0.1",
        "file": "rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf",
        "line": "791"
      },
      "hostname": "127.0.0.1",
      "transactionId": "16861477007022634343"
      "policyId": "/subscriptions/1496a758-b2ff-43ef-b738-
8e9eb5161a86/resourceGroups/drewRG/providers/Microsoft.Network/ApplicationGatewayWebApplicationFirewallPolici
es/globalWafPolicy",
      "policyScope": "Global",
      "policyScopeName": " Global "
    }
  }
}
```


Application Gateway WAF SKU pricing

The pricing models are different for the WAF_v1 and WAF_v2 SKUs. Please see the Application Gateway pricing page to learn more.

Next steps

- Get started by Creating a WAF policy
- Learn more about WAF managed rules
- Learn more about Custom Rules
- Learn about Web Application Firewall on Azure Front Door

What's new in Azure Web Application Firewall?

11/4/2019 • 2 minutes to read • Edit Online

Azure Web Application Firewall is updated on an ongoing basis. To stay up-to-date with the most recent developments, this article provides you with information about:

- The latest releases
- Known issues
- Bug fixes
- Deprecated functionality

New features

FEATURE	DESCRIPTION	DATE ADDED
Bot Mitigation Ruleset (preview)	You can enable a Bot Mitigation ruleset, alongside the CRS ruleset you choose.	November 2019
GeoDB Integration (preview)	Now you can create custom rules restricting traffic by country of origin.	November 2019
WAF per-site/per-URI policy (preview)	WAF-v2 now supports applying a policy to listeners, as well as path-based rules. See Create WAF Policy.	November 2019
WAF custom rules	Application Gateway WAF_v2 now supports creating custom rules. See Application Gateway custom rules.	June 2019
WAF configuration and exclusion list	We've added more options to help you configure your WAF and reduce false positives. See Web application firewall request size limits and exclusion lists for more information.	December 2018

Next steps

For more information about Web Application Firewall on Application Gateway, see Azure Web Application Firewall on Azure Application Gateway.

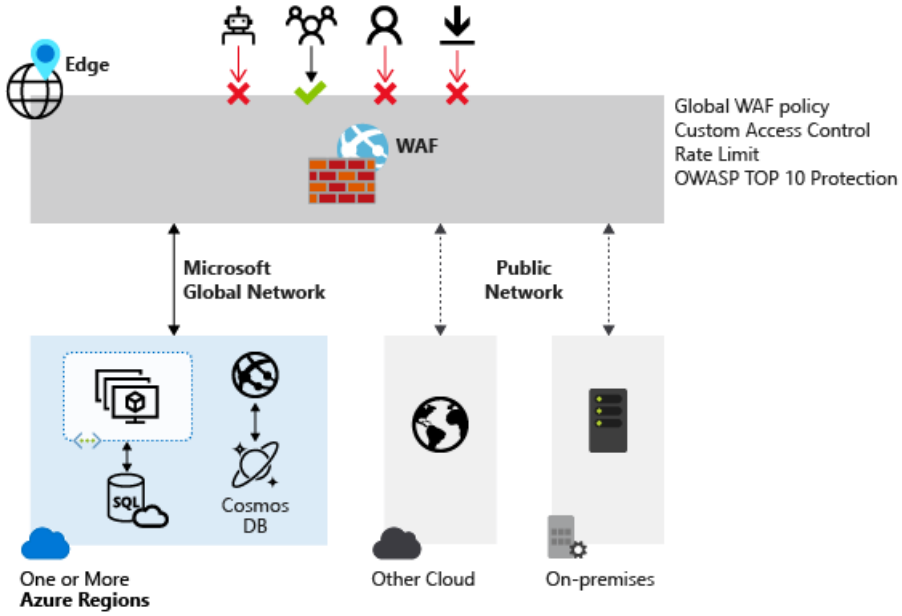
Azure Web Application Firewall on Azure Front Door

2/27/2020 • 6 minutes to read • Edit Online

Azure Web Application Firewall (WAF) on Azure Front Door provides centralized protection for your web applications. WAF defends your web services against common exploits and vulnerabilities. It keeps your service highly available for your users and helps you meet compliance requirements.

WAF on Front Door is a global and centralized solution. It's deployed on Azure network edge locations around the globe. WAF enabled web applications inspect every incoming request delivered by Front Door at the network edge.

WAF prevents malicious attacks close to the attack sources, before they enter your virtual network. You get global protection at scale without sacrificing performance. A WAF policy easily links to any Front Door profile in your subscription. New rules can be deployed within minutes, so you can respond quickly to changing threat patterns.



WAF policy and rules

You can configure a WAF policy and associate that policy to one or more Front Door front-ends for protection. A WAF policy consists of two types of security rules:

- custom rules that are authored by the customer.
- managed rule sets that are a collection of Azure-managed pre-configured set of rules.

When both are present, custom rules are processed before processing the rules in a managed rule set. A rule is made of a match condition, a priority, and an action. Action types supported are: ALLOW, BLOCK, LOG, and REDIRECT. You can create a fully customized policy that meets your specific application protection requirements by combining managed and custom rules.

Rules within a policy are processed in a priority order. Priority is a unique integer that defines the order of rules to process. Smaller integer value denotes a higher priority and those rules are evaluated before rules with a higher integer value. Once a rule is matched, the corresponding action that was defined in the rule is applied to the request. Once such a match is processed, rules with lower priorities aren't processed further.

A web application delivered by Front Door can have only one WAF policy associated with it at a time. However,

you can have a Front Door configuration without any WAF policies associated with it. If a WAF policy is present, it's replicated to all of our edge locations to ensure consistent security policies across the world.

WAF modes

WAF policy can be configured to run in the following two modes:

- **Detection mode:** When run in detection mode, WAF doesn't take any other actions other than monitors and logs the request and its matched WAF rule to WAF logs. You can turn on logging diagnostics for Front Door. When you use the portal, go to the **Diagnostics** section.
- **Prevention mode:** In prevention mode, WAF takes the specified action if a request matches a rule. If a match is found, no further rules with lower priority are evaluated. Any matched requests are also logged in the WAF logs.

WAF actions

WAF customers can choose to run from one of the actions when a request matches a rule's conditions:

- **Allow:** Request passes through the WAF and is forwarded to back-end. No further lower priority rules can block this request.
- **Block:** The request is blocked and WAF sends a response to the client without forwarding the request to the back-end.
- **Log:** Request is logged in the WAF logs and WAF continues evaluating lower priority rules.
- **Redirect:** WAF redirects the request to the specified URI. The URI specified is a policy level setting. Once configured, all requests that match the **Redirect** action will be sent to that URI.

WAF rules

A WAF policy can consist of two types of security rules - custom rules, authored by the customer and managed rulesets, Azure-managed pre-configured set of rules.

Custom authored rules

You can configure custom rules WAF as follows:

- **IP allow list and block list:** You can control access to your web applications based on a list of client IP addresses or IP address ranges. Both IPv4 and IPv6 address types are supported. This list can be configured to either block or allow those requests where the source IP matches an IP in the list.
- **Geographic based access control:** You can control access to your web applications based on the country code that's associated with a client's IP address.
- **HTTP parameters-based access control:** You can base rules on string matches in HTTP/HTTPS request parameters. For example, query strings, POST args, Request URI, Request Header, and Request Body.
- **Request method-based access control:** You based rules on the HTTP request method of the request. For example, GET, PUT, or HEAD.
- **Size constraint:** You can base rules on the lengths of specific parts of a request such as query string, Uri, or request body.
- **Rate limiting rules:** A rate control rule is to limit abnormal high traffic from any client IP. You may configure a threshold on the number of web requests allowed from a client IP during a one-minute duration. This rule is distinct from an IP list-based allow/block custom rule that either allows all or blocks all request from a client IP. Rate limits can be combined with additional match conditions such as HTTP(S) parameter matches for granular rate control.

Azure-managed rule sets

Azure-managed rule sets provide an easy way to deploy protection against a common set of security threats. Since such rulesets are managed by Azure, the rules are updated as needed to protect against new attack signatures. Azure-managed Default Rule Set includes rules against the following threat categories:

- Cross-site scripting
- Java attacks
- Local file inclusion
- PHP injection attacks
- Remote command execution
- Remote file inclusion
- Session fixation
- SQL injection protection
- Protocol attackers

The version number of the Default Rule Set increments when new attack signatures are added to the rule set. Default Rule Set is enabled by default in Detection mode in your WAF policies. You can disable or enable individual rules within the Default Rule Set to meet your application requirements. You can also set specific actions (ALLOW/BLOCK/REDIRECT/LOG) per rule.

Sometimes you may need to omit certain request attributes from a WAF evaluation. A common example is Active Directory-inserted tokens that are used for authentication. You may configure an exclusion list for a managed rule, rule group, or for the entire rule set.

The Default action is to BLOCK. Additionally, custom rules can be configured in the same WAF policy if you wish to bypass any of the pre-configured rules in the Default Rule Set.

Custom rules are always applied before rules in the Default Rule Set are evaluated. If a request matches a custom rule, the corresponding rule action is applied. The request is either blocked or passed through to the back-end. No other custom rules or the rules in the Default Rule Set are processed. You can also remove the Default Rule Set from your WAF policies.

Bot protection rule set (preview)

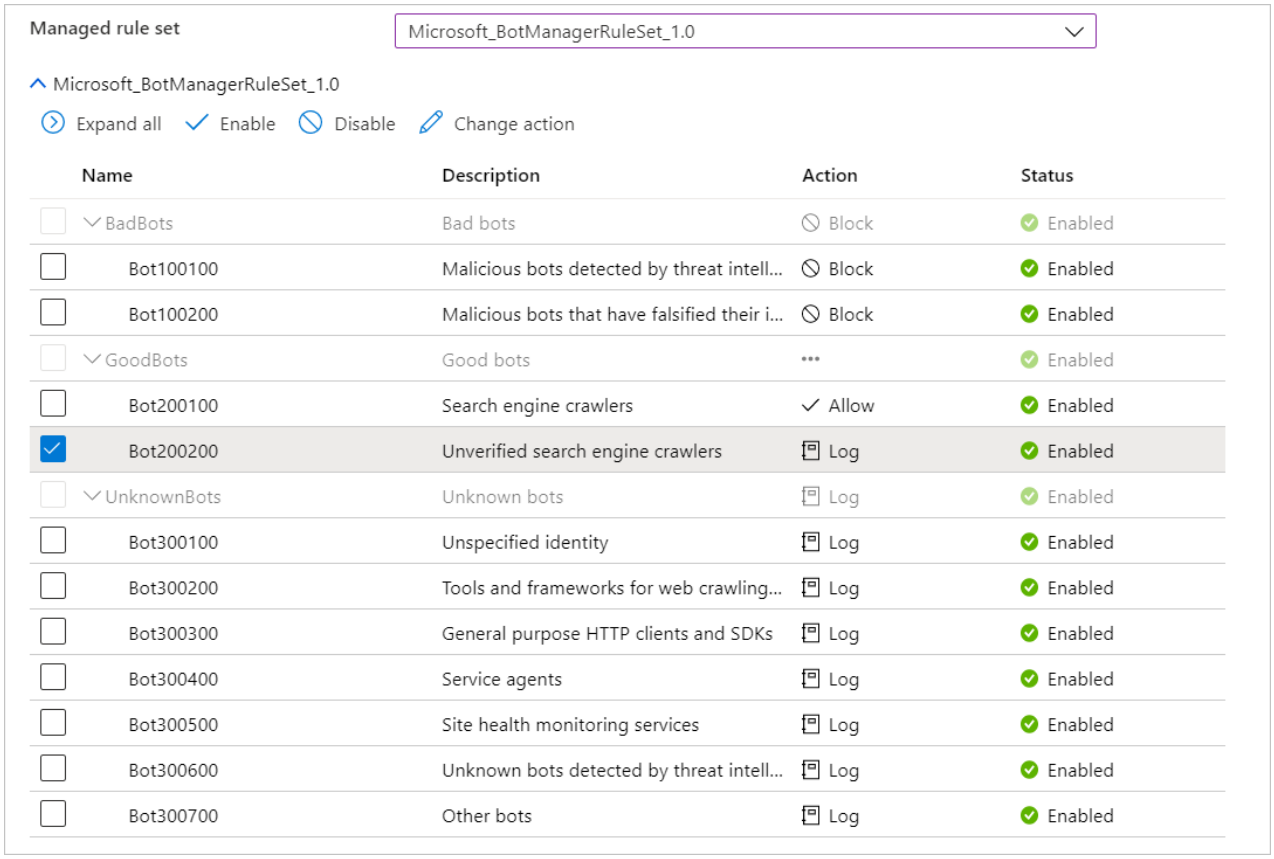
You can enable a managed bot protection rule set to take custom actions on requests from known bot categories.

There are three bot categories supported: Bad, Good, and Unknown. Bot signatures are managed and dynamically updated by the WAF platform.

Bad bots include bots from malicious IP addresses and bots that have falsified their identities. Malicious IP addresses are sourced from the Microsoft Threat Intelligence feed and updated every hour. Intelligent Security Graph powers Microsoft Threat Intelligence and is used by multiple services including Azure Security Center.

Good Bots include validated search engines. Unknown categories include additional bot groups that have identified themselves as bots. For example, market analyzer, feed fetchers and data collection agents.

Unknown bots are classified via published user agents without additional validation. You can set custom actions to block, allow, log, or redirect for different types of bots.



IMPORTANT

The Bot protection rule set is currently in public preview and is provided with a preview service level agreement. Certain features may not be supported or may have constrained capabilities. See the Supplemental Terms of Use for Microsoft Azure Previews for details.

If bot protection is enabled, incoming requests that match bot rules are logged at the FrontdoorWebApplicationFirewallLog log. You may access WAF logs from a storage account, event hub, or log analytics.

Configuration

You can configure and deploy all WAF rule types using the Azure portal, REST APIs, Azure Resource Manager templates, and Azure PowerShell.

Monitoring

Monitoring for WAF at Front Door is integrated with Azure Monitor to track alerts and easily monitor traffic trends.

Next steps

- Learn about Web Application Firewall on Azure Application Gateway

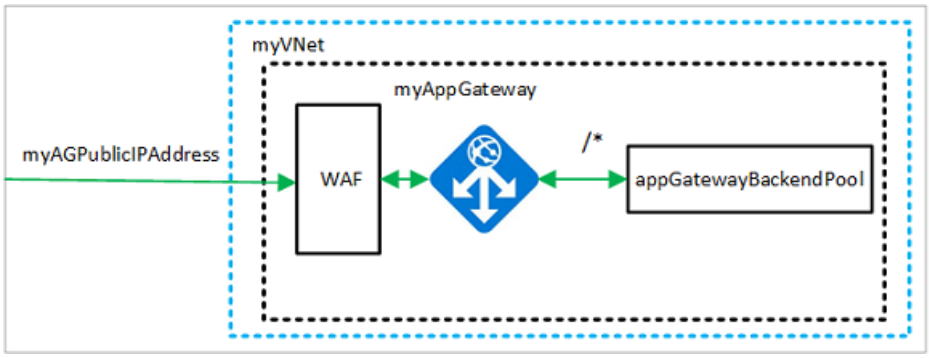
Tutorial: Create an application gateway with a Web Application Firewall using the Azure portal

11/13/2019 • 10 minutes to read • Edit Online

This tutorial shows you how to use the Azure portal to create an Application Gateway with a Web Application Firewall (WAF). The WAF uses OWASP rules to protect your application. These rules include protection against attacks such as SQL injection, cross-site scripting attacks, and session hijacks. After creating the application gateway, you test it to make sure it's working correctly. With Azure Application Gateway, you direct your application web traffic to specific resources by assigning listeners to ports, creating rules, and adding resources to a backend pool. For the sake of simplicity, this tutorial uses a simple setup with a public front-end IP, a basic listener to host a single site on this application gateway, two virtual machines used for the backend pool, and a basic request routing rule.

In this tutorial, you learn how to:

- Create an application gateway with WAF enabled
- Create the virtual machines used as backend servers
- Create a storage account and configure diagnostics
- Test the application gateway



NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

If you don't have an Azure subscription, create a free account before you begin.

Sign in to Azure

Sign in to the Azure portal at https://portal.azure.com.

Create an application gateway

For Azure to communicate between resources, it needs a virtual network. You can either create a new virtual network or use an existing one. In this example, you create a new virtual network. You can create a virtual network at the same time that you create the application gateway. Application Gateway instances are created in separate subnets. You create two subnets in this example: one for the application gateway, and another for the backend

servers.

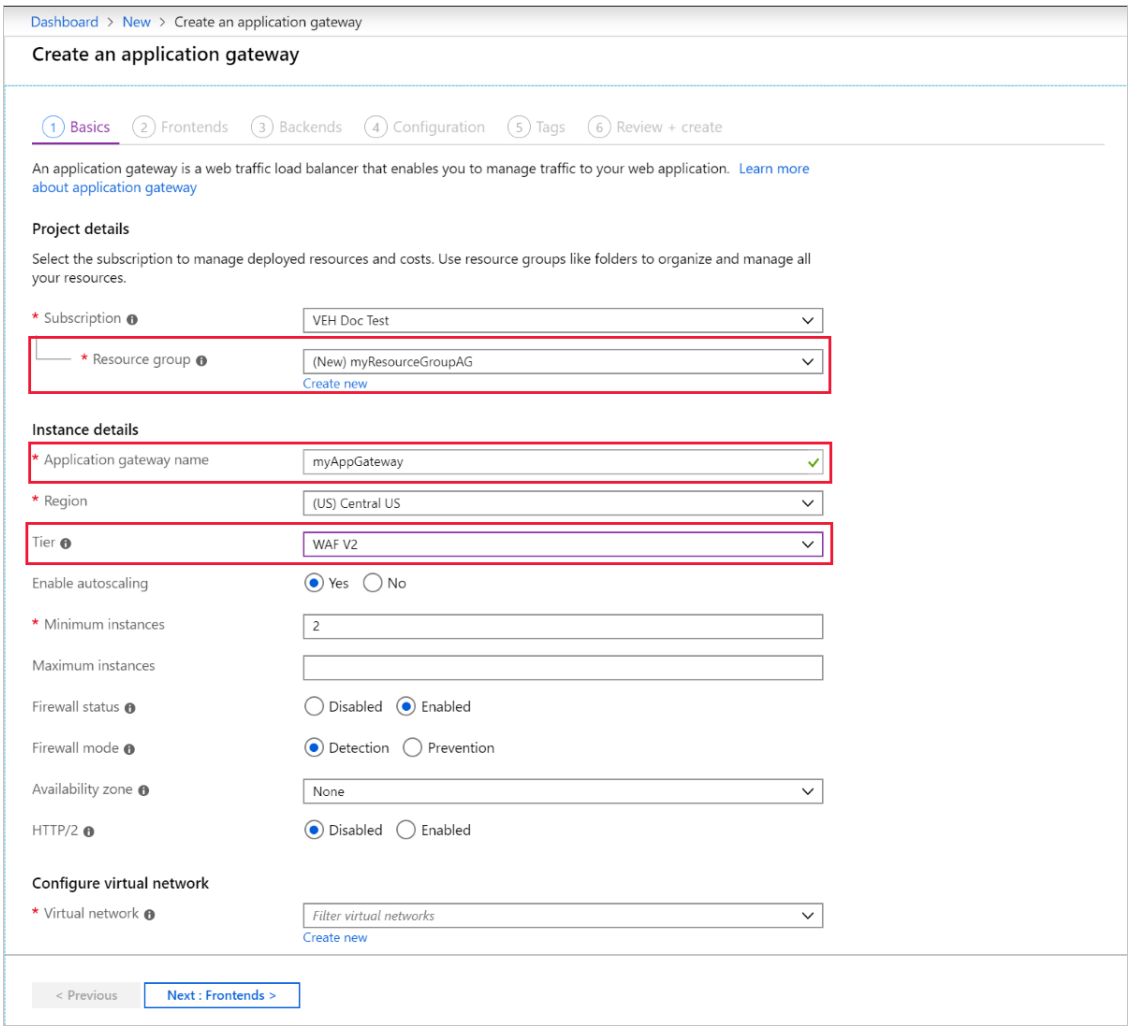
Select **Create a resource** on the left menu of the Azure portal. The **New** window appears.

Select **Networking** and then select **Application Gateway** in the **Featured** list.

Basics tab

1. On the **Basics** tab, enter these values for the following application gateway settings:

   - **Resource group**: Select **myResourceGroupAG** for the resource group. If it doesn't exist, select **Create new** to create it.
   - **Application gateway name**: Enter *myAppGateway* for the name of the application gateway.
   - **Tier**: select **WAF V2**.



2. For Azure to communicate between the resources that you create, it needs a virtual network. You can either create a new virtual network or use an existing one. In this example, you'll create a new virtual network at the same time that you create the application gateway. Application Gateway instances are created in separate subnets. You create two subnets in this example: one for the application gateway, and another for the backend servers.

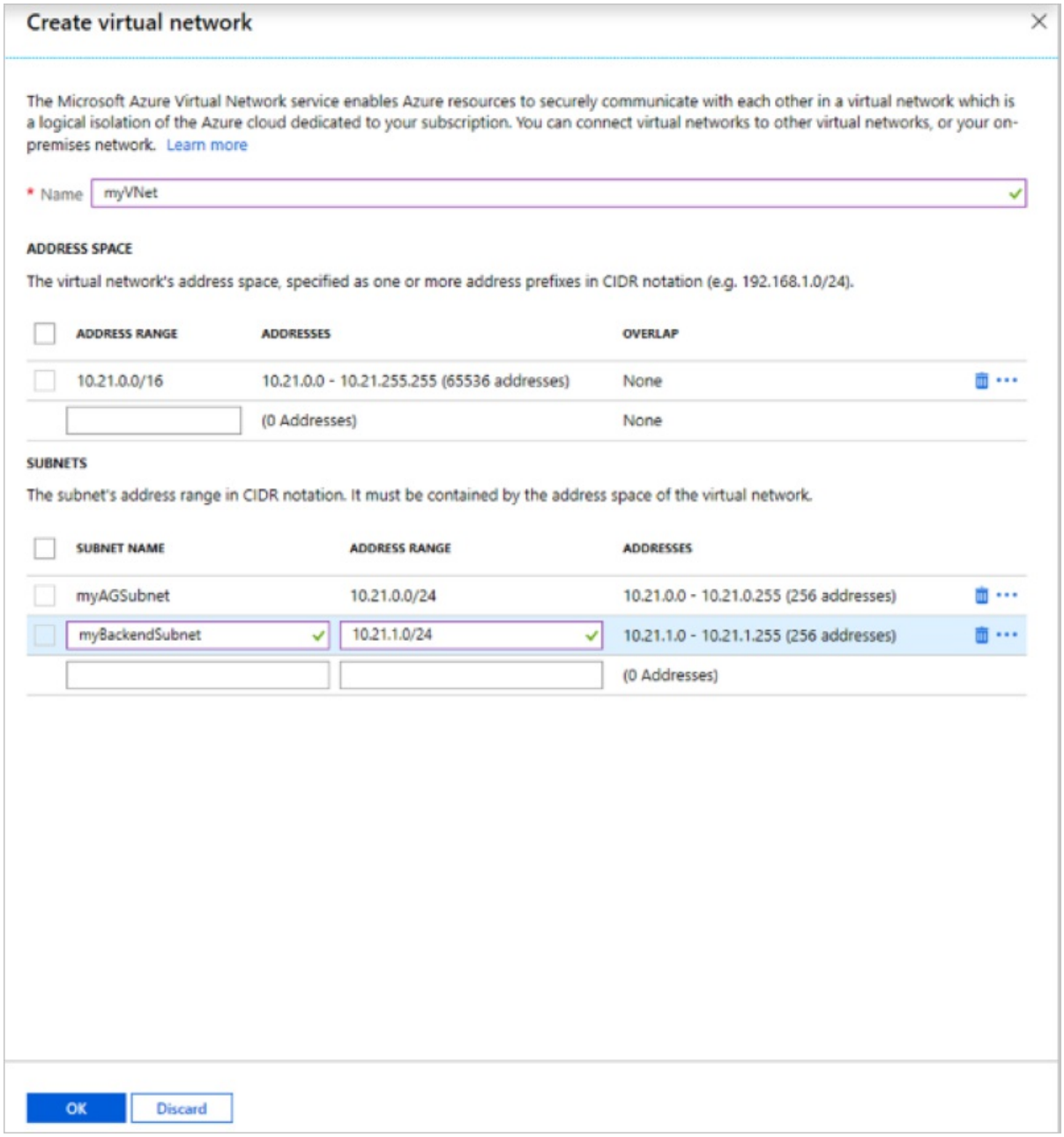
   Under **Configure virtual network**, create a new virtual network by selecting **Create new**. In the **Create virtual network** window that opens, enter the following values to create the virtual network and two subnets:

   - **Name**: Enter *myVNet* for the name of the virtual network.
   - **Subnet name** (Application Gateway subnet): The **Subnets** grid will show a subnet named *Default*. Change the name of this subnet to *myAGSubnet*.

The application gateway subnet can contain only application gateways. No other resources are allowed.

- **Subnet name** (backend server subnet): In the second row of the **Subnets** grid, enter *myBackendSubnet* in the **Subnet name** column.
- **Address range** (backend server subnet): In the second row of the **Subnets** Grid, enter an address range that doesn't overlap with the address range of *myAGSubnet*. For example, if the address range of *myAGSubnet* is 10.0.0.0/24, enter *10.0.1.0/24* for the address range of *myBackendSubnet*.

Select **OK** to close the **Create virtual network** window and save the virtual network settings.



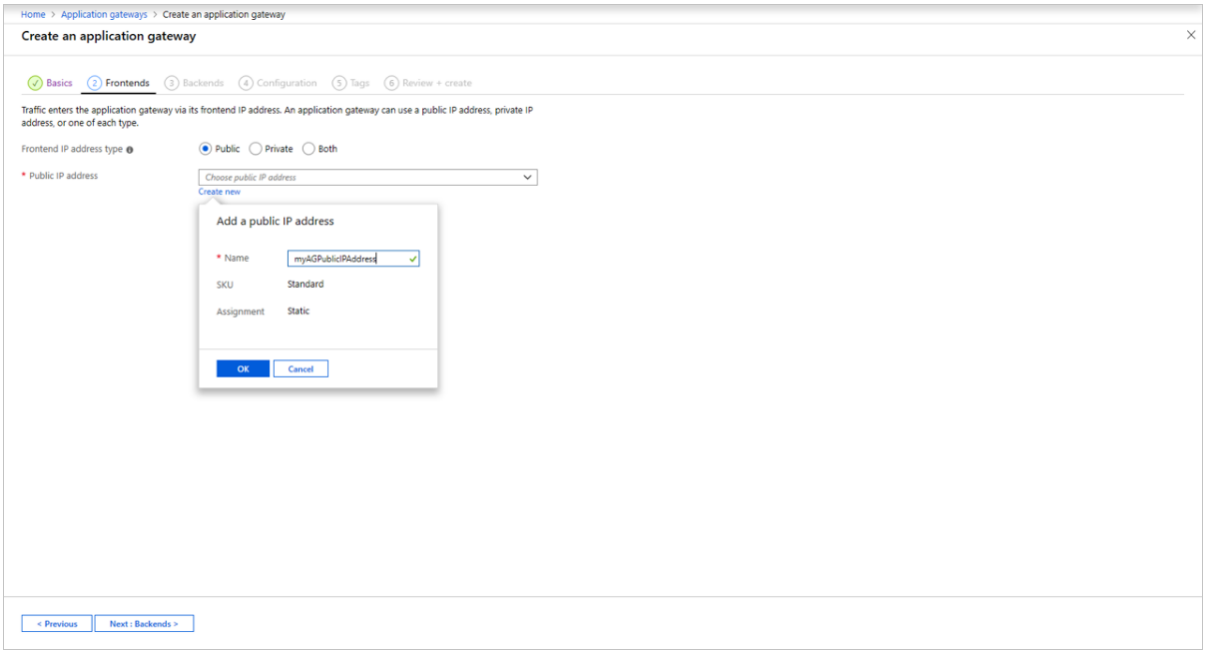
3. On the **Basics** tab, accept the default values for the other settings and then select **Next: Frontends**.

Frontends tab

1. On the **Frontends** tab, verify **Frontend IP address type** is set to **Public**.
   You can configure the Frontend IP to be Public or Private as per your use case. In this example, you'll choose a Public Frontend IP.

> **NOTE**
>
> For the Application Gateway v2 SKU, you can only choose **Public** frontend IP configuration. Private frontend IP configuration is currently not enabled for this v2 SKU.

2. Choose **Create new** for the **Public IP address** and enter *myAGPublicIPAddress* for the public IP address name, and then select **OK**.

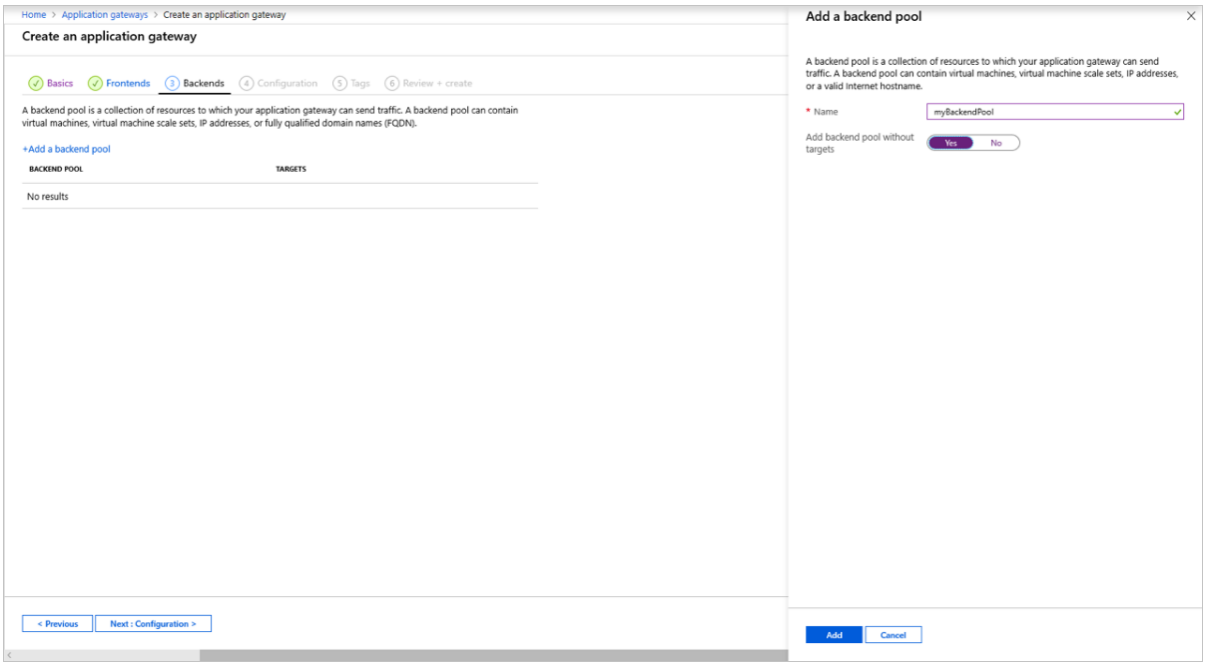


3. Select **Next: Backends**.

Backends tab

The backend pool is used to route requests to the backend servers that serve the request. Backend pools can be composed of NICs, virtual machine scale sets, public IPs, internal IPs, fully qualified domain names (FQDN), and multi-tenant back-ends like Azure App Service. In this example, you'll create an empty backend pool with your application gateway and then add backend targets to the backend pool.

1. On the **Backends** tab, select **+Add a backend pool**.
2. In the **Add a backend pool** window that opens, enter the following values to create an empty backend pool:
   - **Name**: Enter *myBackendPool* for the name of the backend pool.
   - **Add backend pool without targets**: Select **Yes** to create a backend pool with no targets. You'll add backend targets after creating the application gateway.
3. In the **Add a backend pool** window, select **Add** to save the backend pool configuration and return to the **Backends** tab.



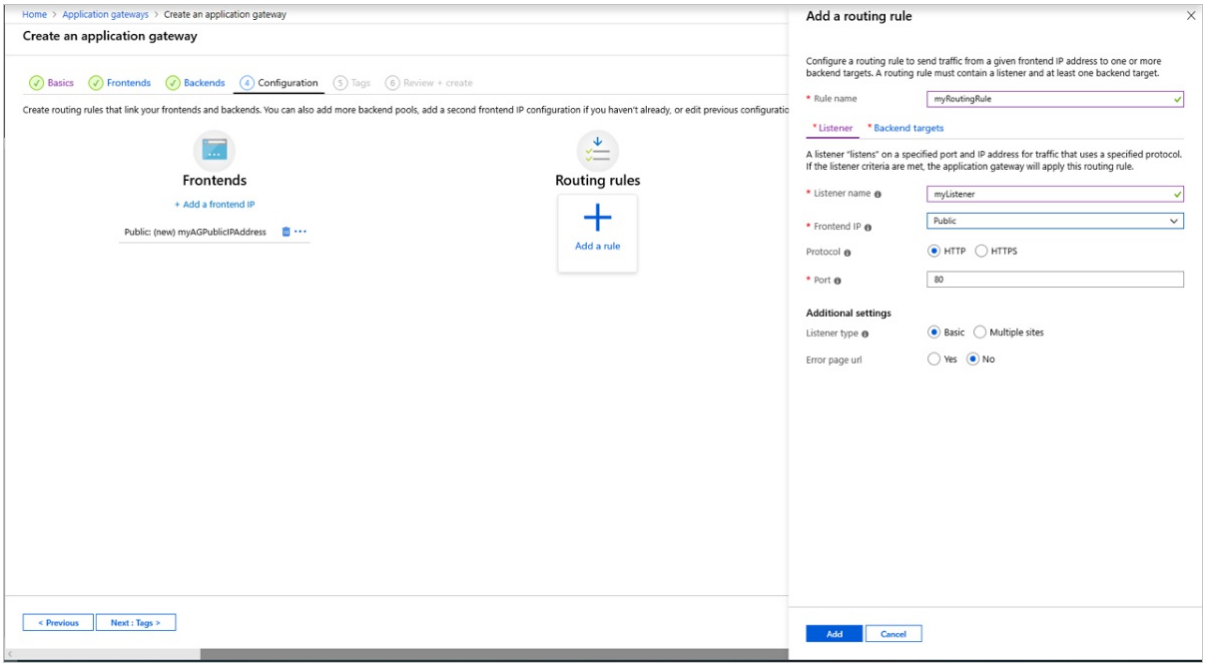
4. On the **Backends** tab, select **Next: Configuration**.

Configuration tab

On the **Configuration** tab, you'll connect the frontend and backend pool you created using a routing rule.

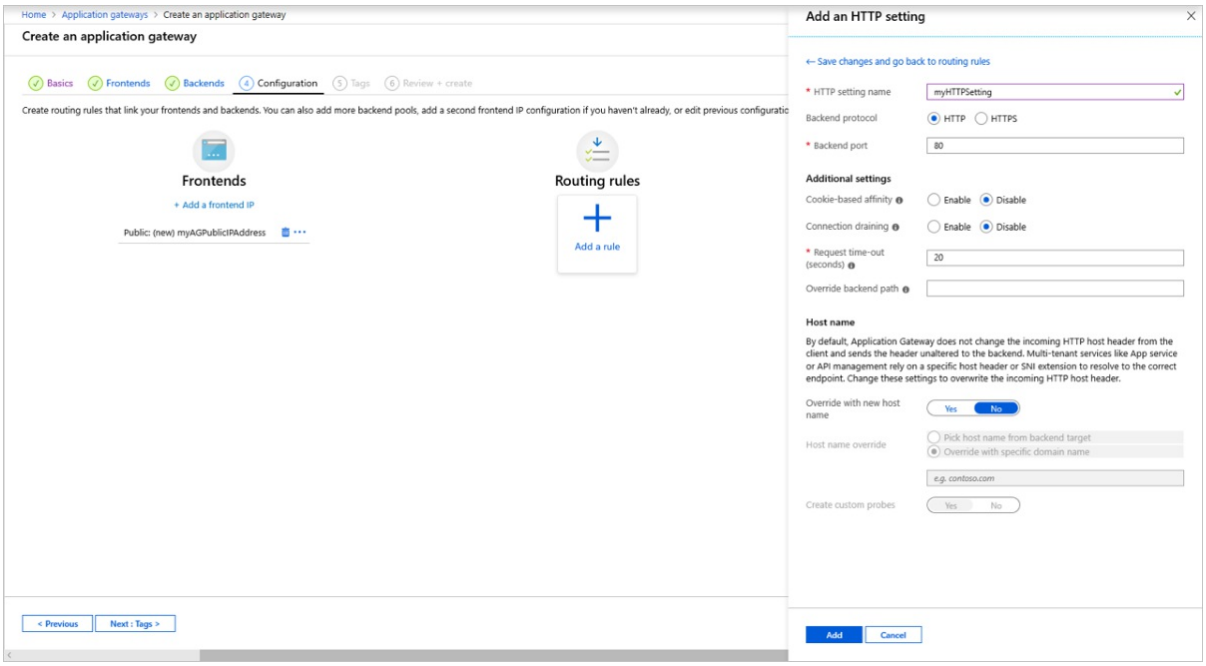
1. Select **Add a rule** in the **Routing rules** column.
2. In the **Add a routing rule** window that opens, enter *myRoutingRule* for the **Rule name**.
3. A routing rule requires a listener. On the **Listener** tab within the **Add a routing rule** window, enter the following values for the listener:
   - **Listener name**: Enter *myListener* for the name of the listener.
   - **Frontend IP**: Select **Public** to choose the public IP you created for the frontend.

     Accept the default values for the other settings on the **Listener** tab, then select the **Backend targets** tab to configure the rest of the routing rule.

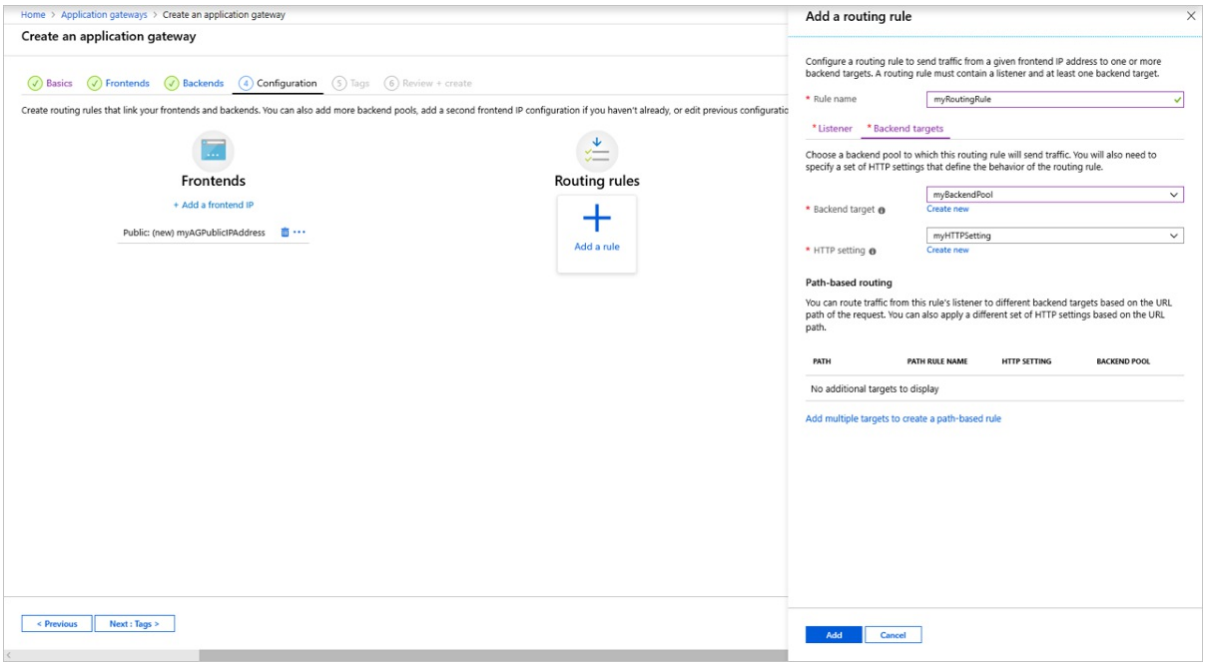


4. On the **Backend targets** tab, select **myBackendPool** for the **Backend target**.

5. For the **HTTP setting**, select **Create new** to create a new HTTP setting. The HTTP setting will determine the behavior of the routing rule. In the **Add an HTTP setting** window that opens, enter *myHTTPSetting* for the **HTTP setting name**. Accept the default values for the other settings in the **Add an HTTP setting** window, then select **Add** to return to the **Add a routing rule** window.



6. On the **Add a routing rule** window, select **Add** to save the routing rule and return to the **Configuration** tab.



7. Select **Next: Tags** and then **Next: Review + create**.

Review + create tab

Review the settings on the **Review + create** tab, and then select **Create** to create the virtual network, the public IP address, and the application gateway. It may take several minutes for Azure to create the application gateway.

Wait until the deployment finishes successfully before moving on to the next section.

Add backend targets

In this example, you'll use virtual machines as the target backend. You can either use existing virtual machines or

create new ones. You'll create two virtual machines that Azure uses as backend servers for the application gateway.

To do this, you'll:

1. Create two new VMs, *myVM* and *myVM2*, to be used as backend servers.
2. Install IIS on the virtual machines to verify that the application gateway was created successfully.
3. Add the backend servers to the backend pool.

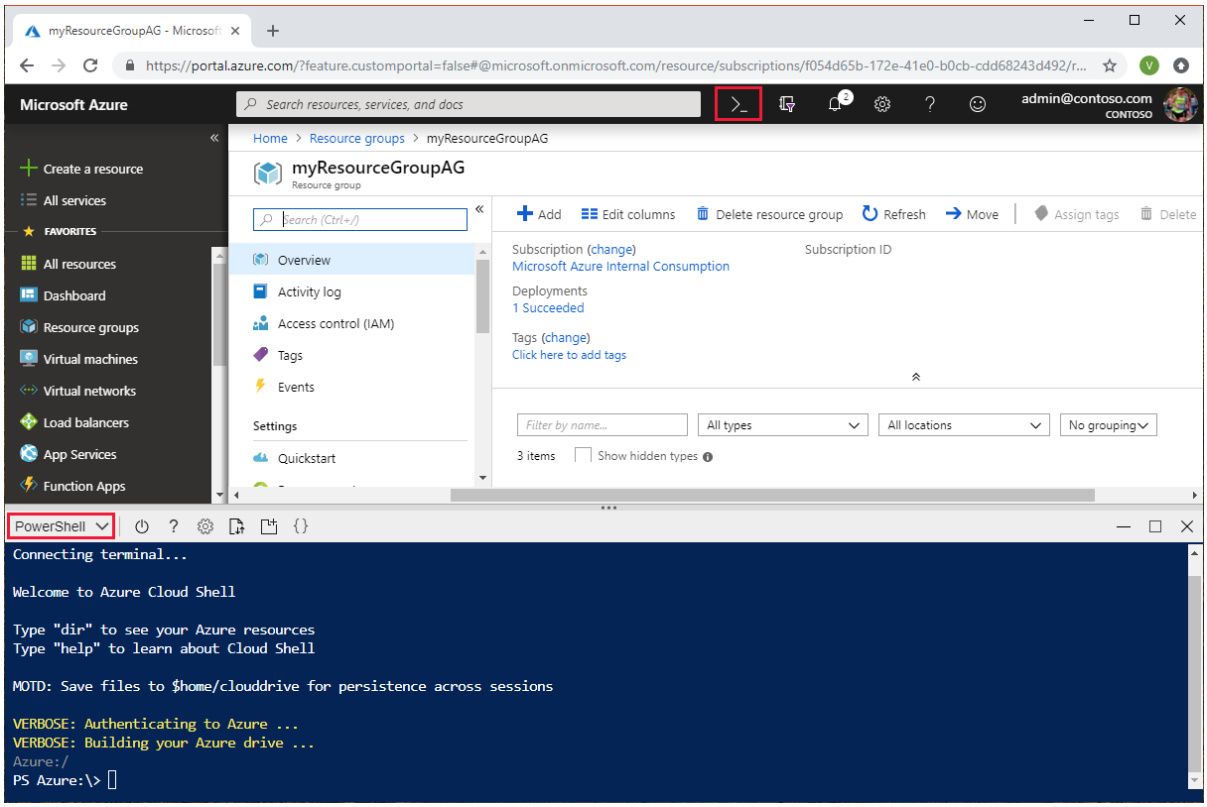
Create a virtual machine

1. On the Azure portal, select **Create a resource**. The **New** window appears.
2. Select **Windows Server 2016 Datacenter** in the **Popular** list. The **Create a virtual machine** page appears.
   Application Gateway can route traffic to any type of virtual machine used in its backend pool. In this example, you use a Windows Server 2016 Datacenter.
3. Enter these values in the **Basics** tab for the following virtual machine settings:
   - **Resource group**: Select **myResourceGroupAG** for the resource group name.
   - **Virtual machine name**: Enter *myVM* for the name of the virtual machine.
   - **Username**: Enter *azureuser* for the administrator user name.
   - **Password**: Enter *Azure123456!* for the administrator password.
4. Accept the other defaults and then select **Next: Disks**.
5. Accept the **Disks** tab defaults and then select **Next: Networking**.
6. On the **Networking** tab, verify that **myVNet** is selected for the **Virtual network** and the **Subnet** is set to **myBackendSubnet**. Accept the other defaults and then select **Next: Management**.
   Application Gateway can communicate with instances outside of the virtual network that it is in, but you need to ensure there's IP connectivity.
7. On the **Management** tab, set **Boot diagnostics** to **Off**. Accept the other defaults and then select **Review + create**.
8. On the **Review + create** tab, review the settings, correct any validation errors, and then select **Create**.
9. Wait for the virtual machine creation to complete before continuing.

Install IIS for testing

In this example, you install IIS on the virtual machines only to verify Azure created the application gateway successfully.

1. Open Azure PowerShell. To do so, select **Cloud Shell** from the top navigation bar of the Azure portal and then select **PowerShell** from the drop-down list.



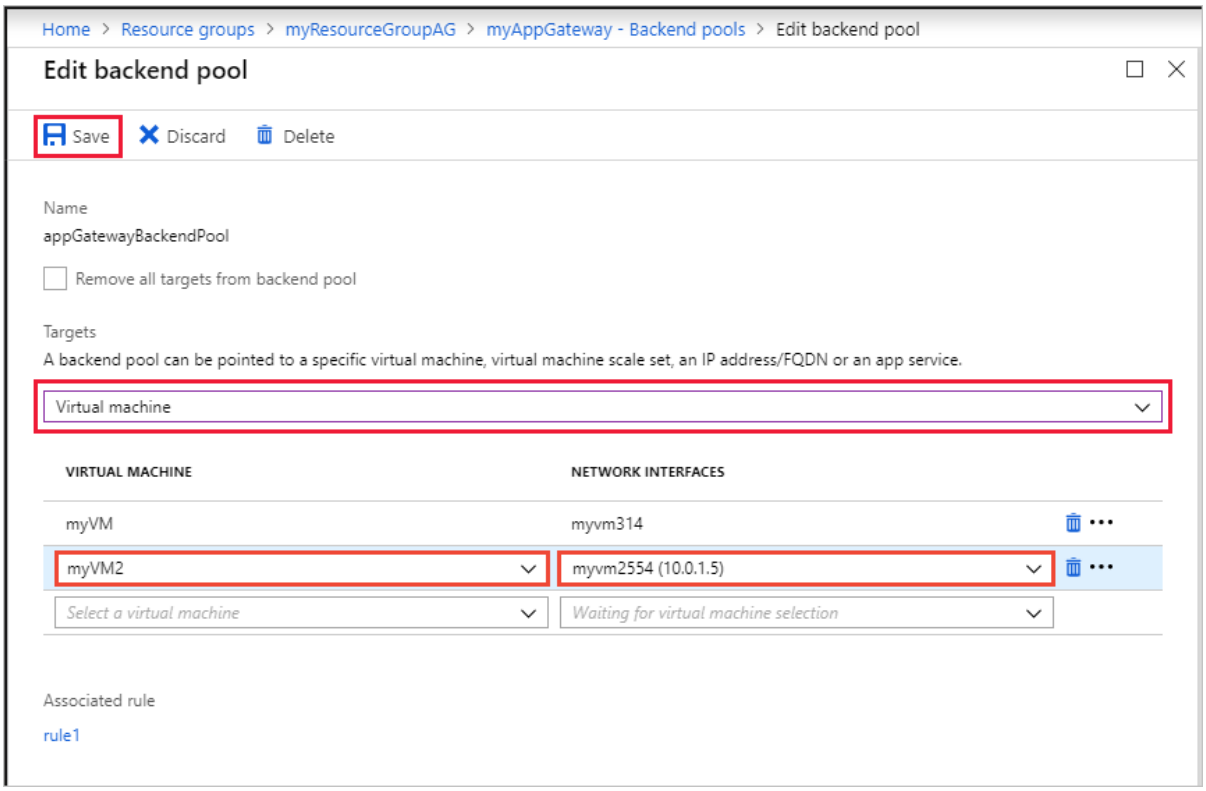
2. Run the following command to install IIS on the virtual machine:

```
Set-AzVMExtension `
  -ResourceGroupName myResourceGroupAG `
  -ExtensionName IIS `
  -VMName myVM `
  -Publisher Microsoft.Compute `
  -ExtensionType CustomScriptExtension `
  -TypeHandlerVersion 1.4 `
  -SettingString '{"commandToExecute":"powershell Add-WindowsFeature Web-Server; powershell Add-Content
-Path \"C:\\inetpub\\wwwroot\\Default.htm\" -Value $($env:computername)"}' `
  -Location EastUS
```

3. Create a second virtual machine and install IIS by using the steps that you previously completed. Use *myVM2* for the virtual machine name and for the **VMName** setting of the **Set-AzVMExtension** cmdlet.

Add backend servers to backend pool

1. Select **All resources**, and then select **myAppGateway**.
2. Select **Backend pools** from the left menu.
3. Select **myBackendPool**.
4. Under **Targets**, select **Virtual machine** from the drop-down list.
5. Under **VIRTUAL MACHINE** and **NETWORK INTERFACES**, select the **myVM** and **myVM2** virtual machines and their associated network interfaces from the drop-down lists.



6. Select **Save**.

7. Wait for the deployment to complete before proceeding to the next step.

Create a storage account and configure diagnostics

Create a storage account

For this article, the application gateway uses a storage account to store data for detection and prevention purposes. You could also use Azure Monitor logs or Event Hub to record data.

1. Select **Create a resource** on the upper left-hand corner of the Azure portal.
2. Select **Storage**, and then select **Storage account**.
3. For *Resource group*, select **myResourceGroupAG** for the resource group.
4. Type *myagstore1* for the name of the storage account.
5. Accept the default values for the other settings and then select **Review + Create**.
6. Review the settings, and then select **Create**.

Configure diagnostics

Configure diagnostics to record data into the ApplicationGatewayAccessLog, ApplicationGatewayPerformanceLog, and ApplicationGatewayFirewallLog logs.

1. In the left-hand menu, select **All resources**, and then select *myAppGateway*.

2. Under Monitoring, select **Diagnostics settings**.

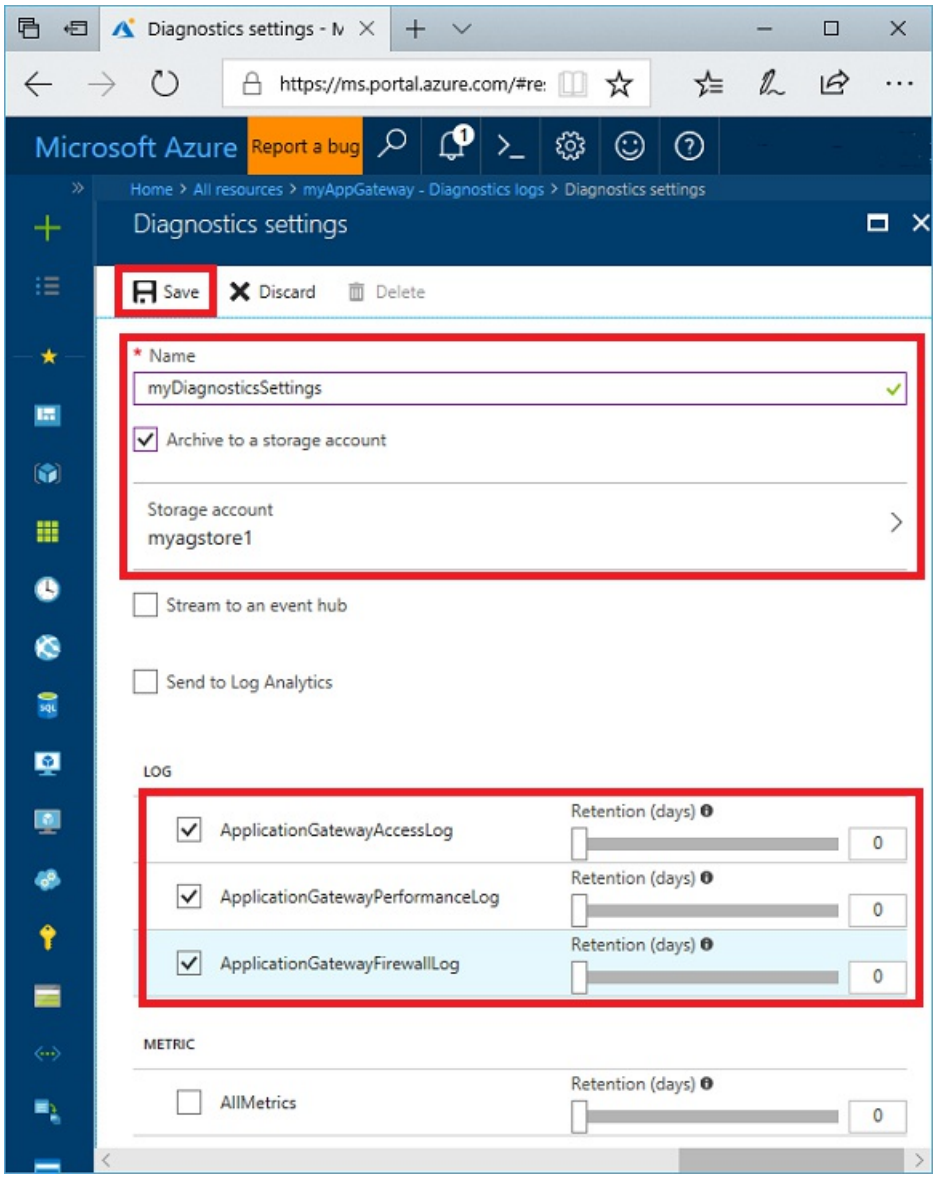
3. select **Add diagnostics setting**.

4. Enter *myDiagnosticsSettings* as the name for the diagnostics settings.

5. Select **Archive to a storage account**, and then select **Configure** to select the *myagstore1* storage account that you previously created, and then select **OK**.

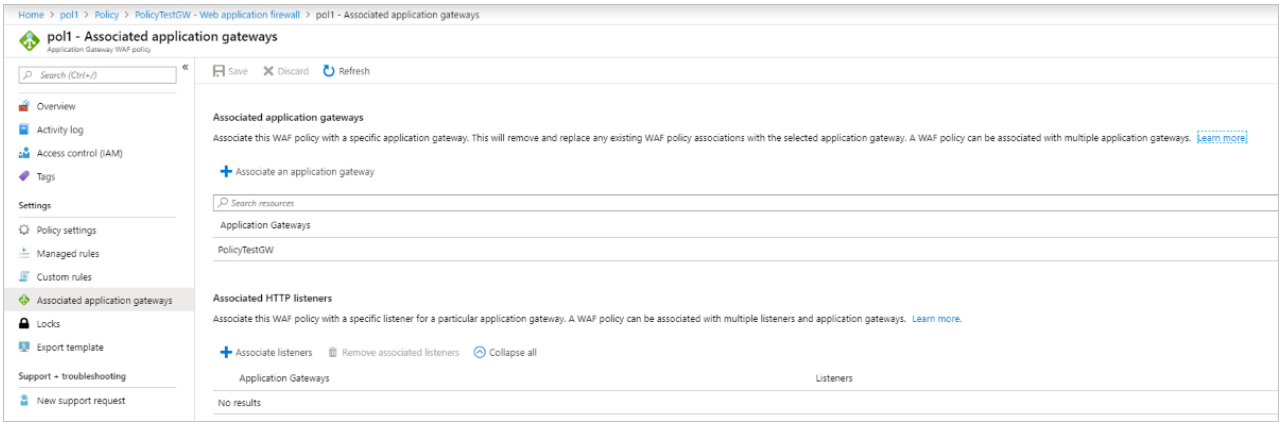
6. Select the application gateway logs to collect and keep.

7. Select **Save**.



Create and link a Web Application Firewall policy

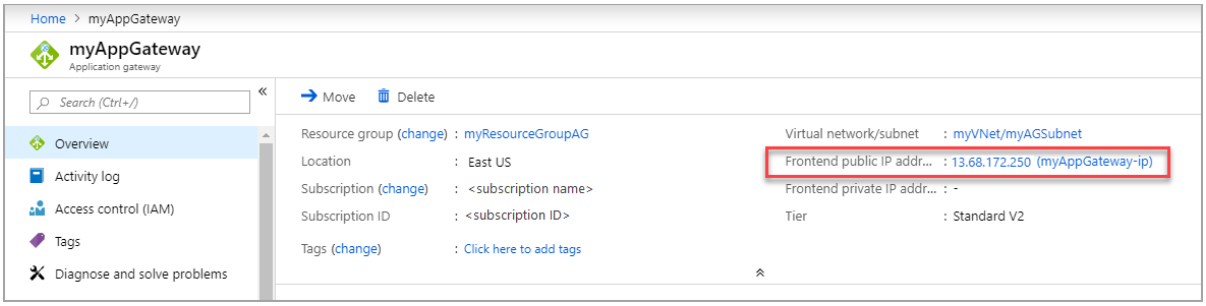
All of the WAF customizations and settings are in a separate object, called a WAF Policy. The policy must be associated with your Application Gateway. To create a WAF Policy, see Create a WAF Policy. Once it's been created, you can then associate the policy to your WAF (or an individual listener) from the WAF Policy in the **Associated Application Gateways** tab.



Test the application gateway

Although IIS isn't required to create the application gateway, you installed it to verify whether Azure successfully created the application gateway. Use IIS to test the application gateway:

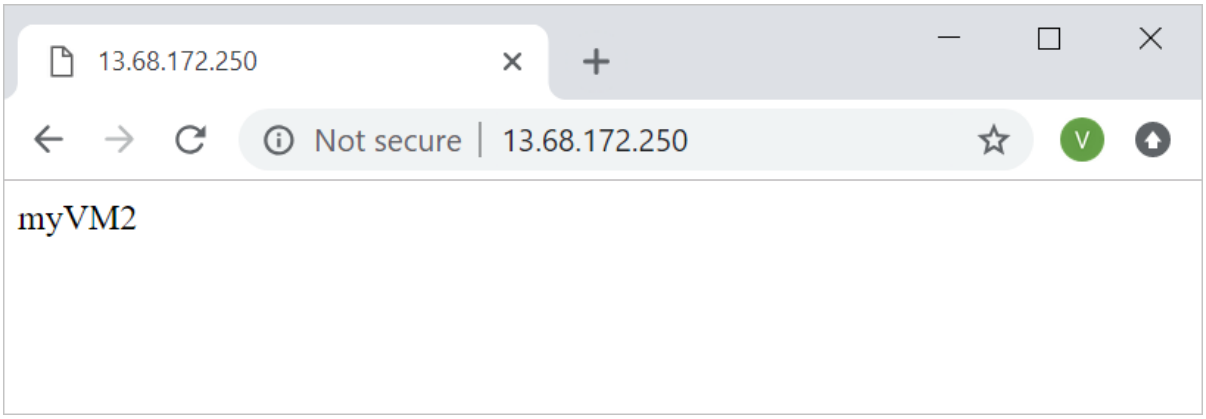
1. Find the public IP address for the application gateway on its **Overview** page.



   Or, you can select **All resources**, enter *myAGPublicIPAddress* in the search box, and then select it in the search results. Azure displays the public IP address on the **Overview** page.

2. Copy the public IP address, and then paste it into the address bar of your browser.

3. Check the response. A valid response verifies that the application gateway was successfully created and it can successfully connect with the backend.



Clean up resources

When you no longer need the resources that you created with the application gateway, remove the resource group. By removing the resource group, you also remove the application gateway and all its related resources.

To remove the resource group:

1. On the left menu of the Azure portal, select **Resource groups**.
2. On the **Resource groups** page, search for **myResourceGroupAG** in the list, then select it.
3. On the **Resource group page**, select **Delete resource group**.
4. Enter *myResourceGroupAG* for **TYPE THE RESOURCE GROUP NAME** and then select **Delete**.

Next steps

Learn more about Web Application Firewall

Tutorial: Create a Web Application Firewall policy on Azure Front Door using the Azure portal

11/19/2019 • 2 minutes to read • Edit Online

This tutorial show you how to create a basic Azure Web Application Firewall (WAF) policy and apply it to a front-end host at Azure Front Door.

In this tutorial, you learn how to:

- Create a WAF policy
- Associate it with a frontend host
- Configure WAF rules

Prerequisites

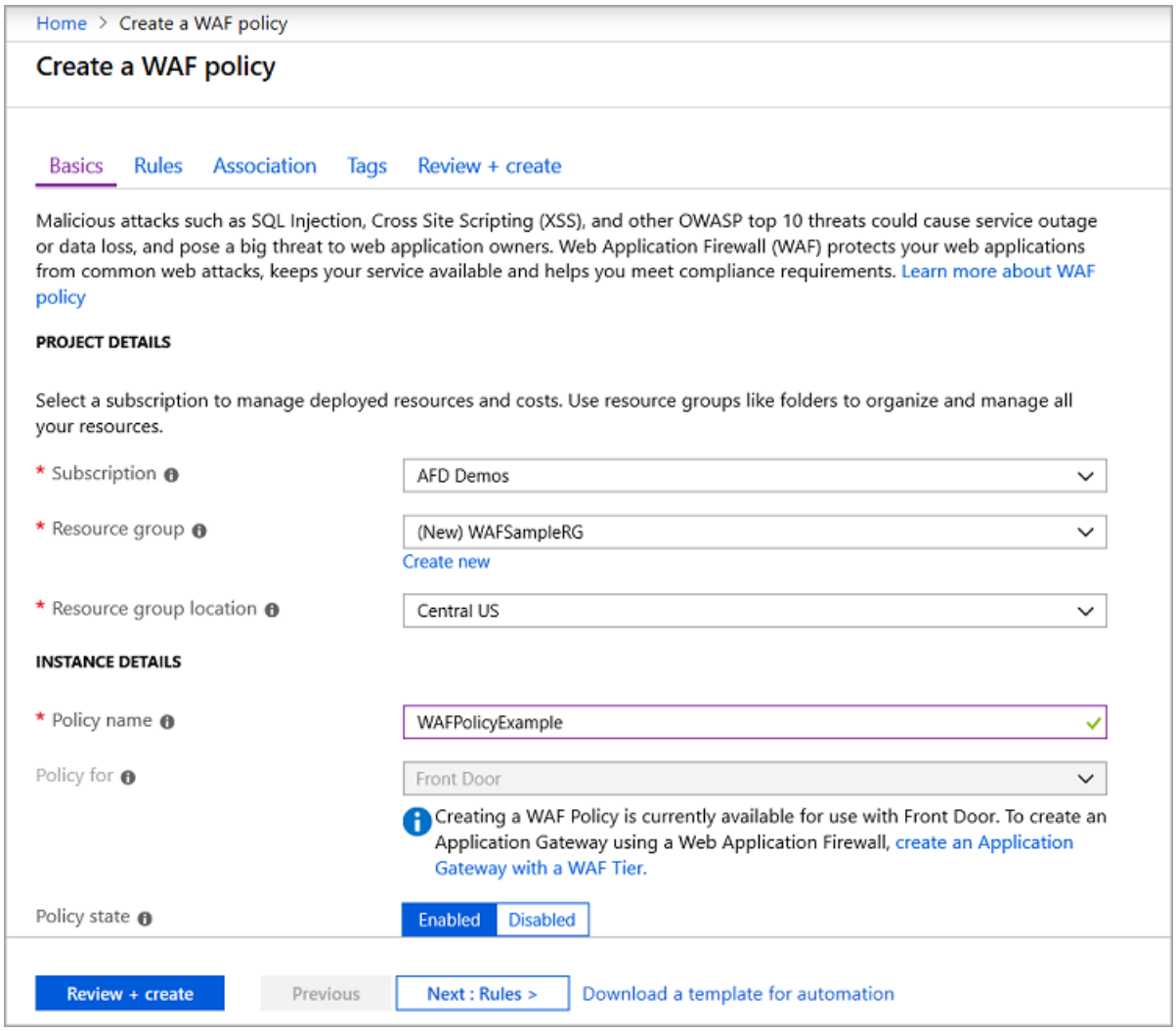
Create a Front Door profile by following the instructions described in Quickstart: Create a Front Door profile.

Create a Web Application Firewall policy

First, create a basic WAF policy with managed Default Rule Set (DRS) by using the portal.

1. On the top left-hand side of the screen, select **Create a resource**>search for **WAF**>select **Web application firewall (Preview)** > select **Create**.
2. In the **Basics** tab of the **Create a WAF policy** page, enter or select the following information, accept the defaults for the remaining settings, and then select **Review + create**:

SETTING	VALUE
Subscription	Select your Front Door subscription name.
Resource group	Select your Front Door resource group name.
Policy name	Enter a unique name for your WAF policy.



3. In the **Association** tab of the **Create a WAF policy** page, select **Add frontend host**, enter the following settings, and then select **Add**:

SETTING	VALUE
Front door	Select your Front Door profile name.
Frontend host	Select the name of your front door host, then select **Add**.

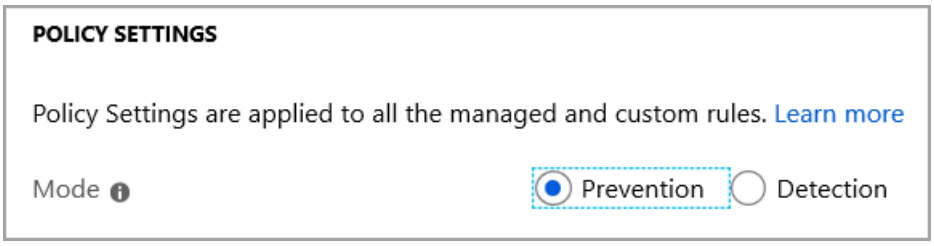
> **NOTE**
> If the frontend host is associated to a WAF policy, it is shown as grayed out. You must first remove the frontend host from the associated policy, and then re-associate the frontend host to a new WAF policy.

4. Select **Review + create**, then select **Create**.

Configure Web Application Firewall rules (optional)

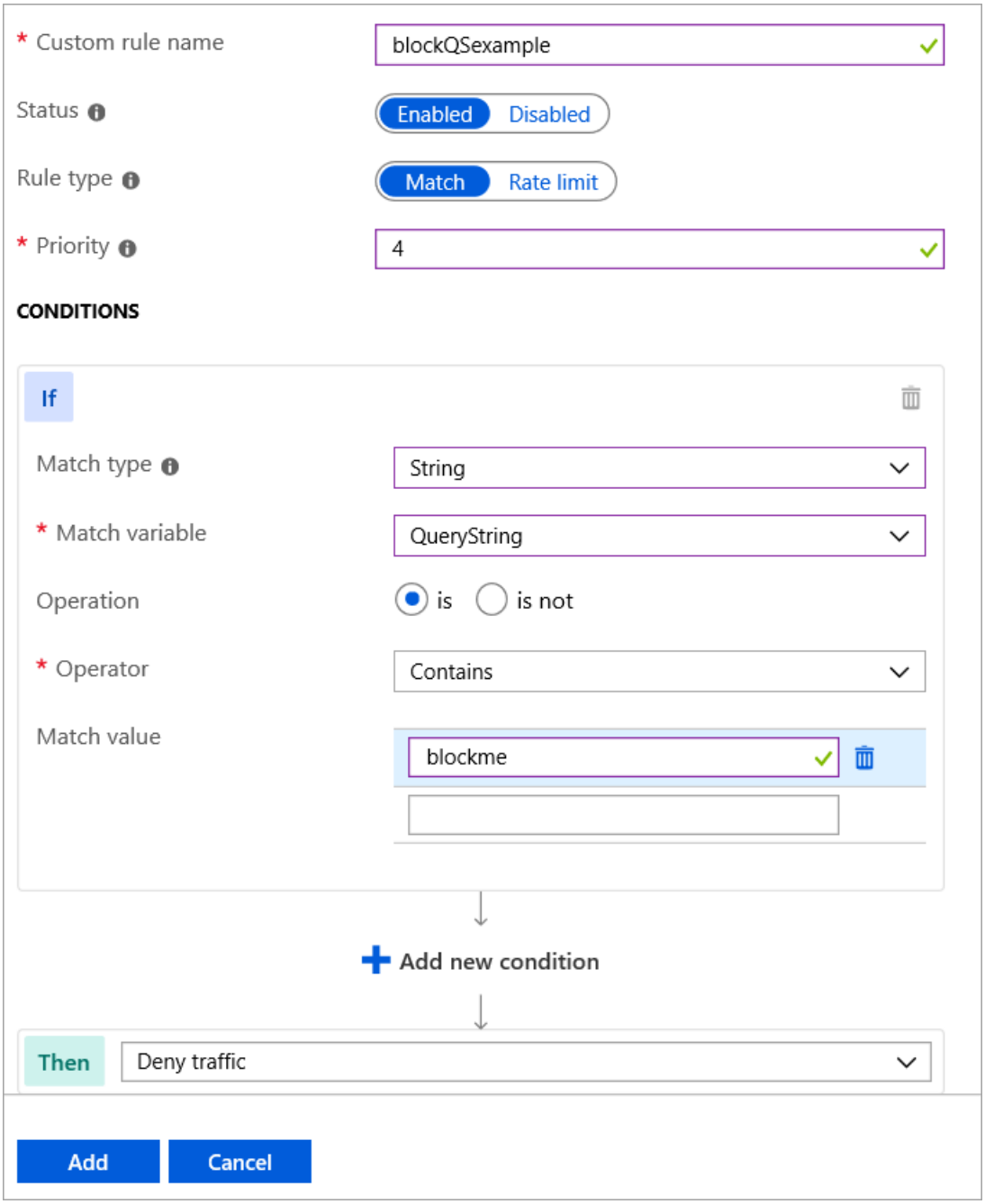
Change mode

When you create a WAF policy, by the default WAF policy is in **Detection** mode. In **Detection** mode, WAF does not block any requests, instead, requests matching the WAF rules are logged at WAF logs. To see WAF in action, you can change the mode settings from **Detection** to **Prevention**. In **Prevention** mode, requests that match rules that are defined in Default Rule Set (DRS) are blocked and logged at WAF logs.



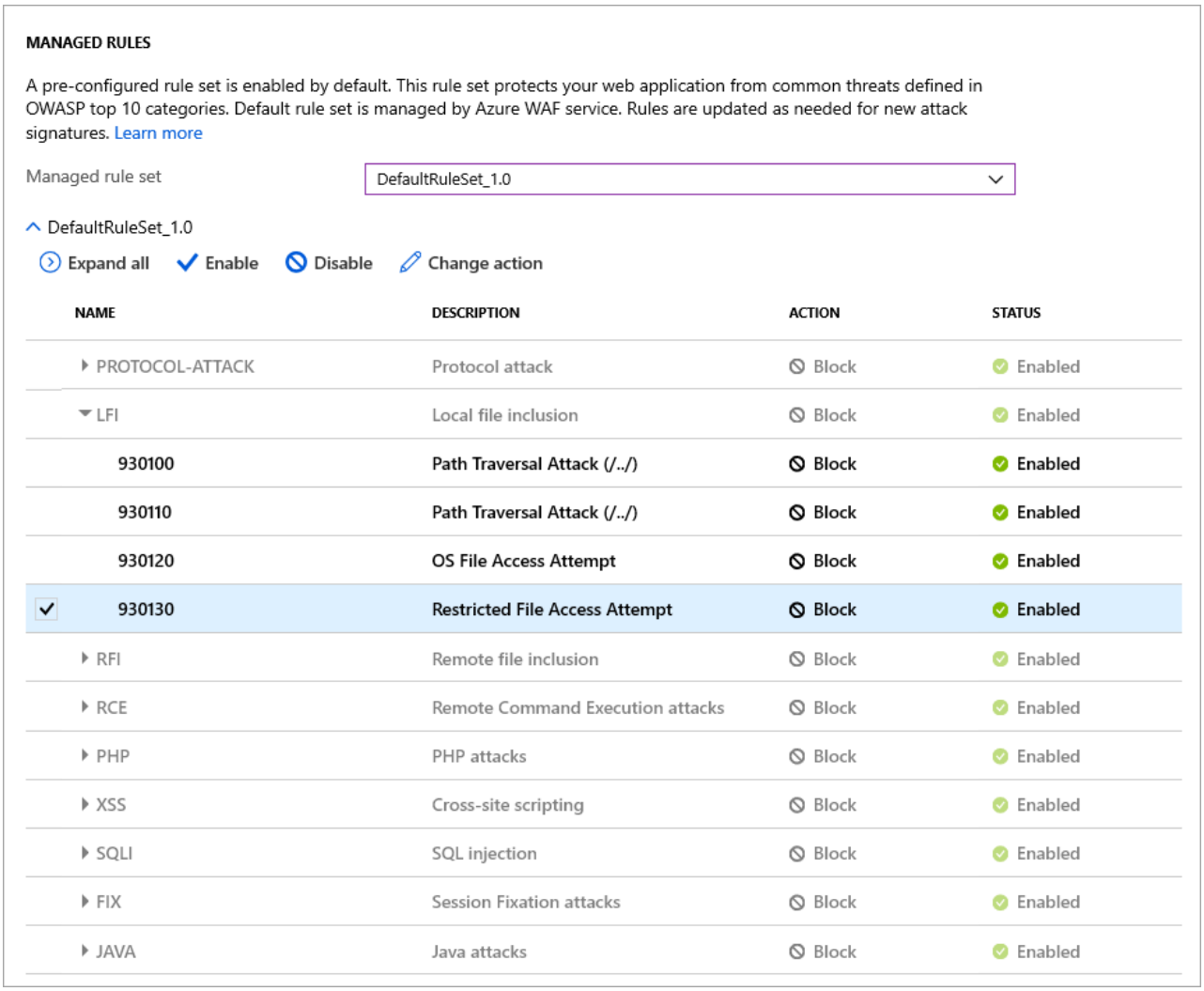
Custom rules

You can create a custom rule by selecting **Add custom rule** under the **Custom rules** section. This launches the custom rule configuration page. Below is an example of configuring a custom rule to block a request if the query string contains **blockme**.



Default Rule Set (DRS)

Azure-managed Default Rule Set is enabled by default. To disable an individual rule within a rule group, expand the rules within that rule group, select the **check box** in front of the rule number, and select **Disable** on the tab above. To change actions types for individual rules within the rule set, select the check box in front of the rule number, and then select the **Change action** tab above.



Next steps

Learn about Azure Web Application Firewall Learn more about Azure Front Door

Azure PowerShell examples for Azure Application Gateway

11/4/2019 • 2 minutes to read • Edit Online

The following table includes links to Azure PowerShell script examples for Azure Application Gateway.

WAF v2 custom rules	Creates an Application Gateway Web Application Firewall v2 with custom rules.

Azure Resource Manager templates for Azure Web Application Firewall

11/4/2019 • 2 minutes to read • Edit Online

The following table includes links to Azure Resource Manager templates for Azure Web Application Firewall.

Application Gateway v2 with Web Application Firewall	Creates an Application Gateway v2 with Web Application Firewall v2.

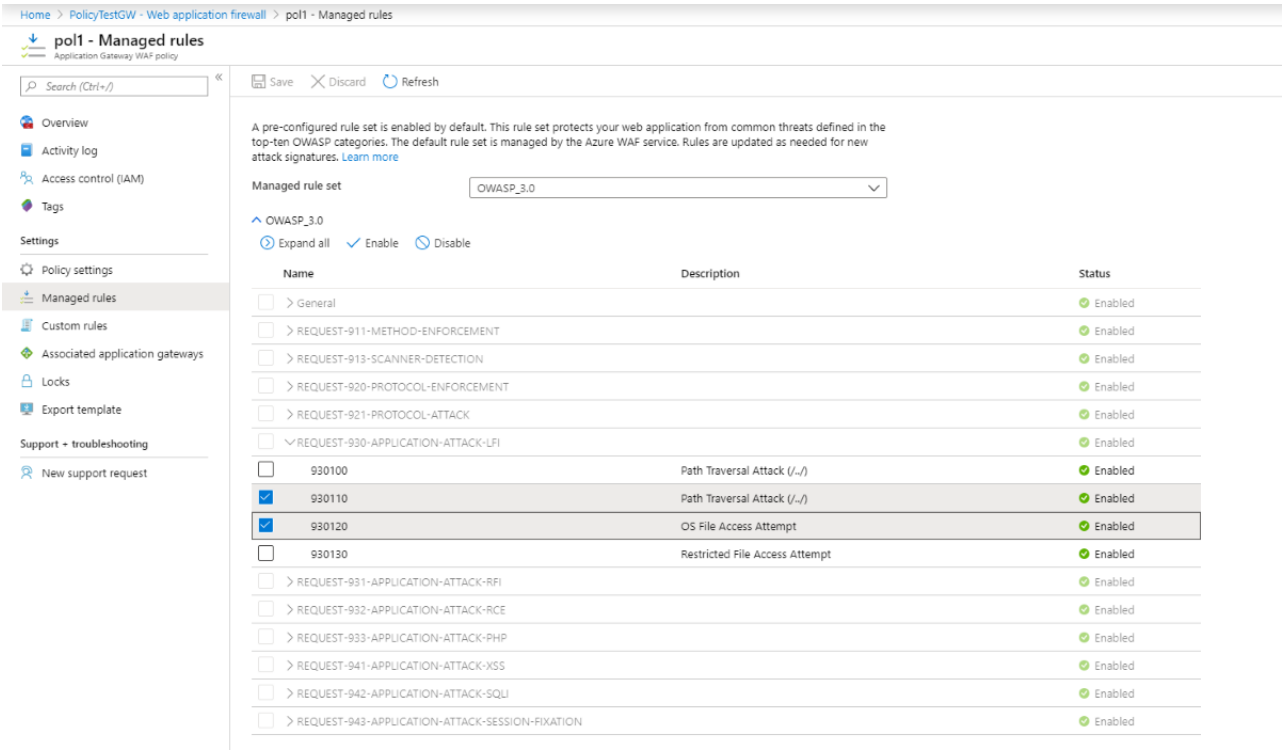
Web Application Firewall CRS rule groups and rules

11/13/2019 • 20 minutes to read • Edit Online

Application Gateway web application firewall (WAF) protects web applications from common vulnerabilities and exploits. This is done through rules that are defined based on the OWASP core rule sets 3.1, 3.0, or 2.2.9. These rules can be disabled on a rule-by-rule basis. This article contains the current rules and rule sets offered.

Core rule sets

The Application Gateway WAF comes pre-configured with CRS 3.0 by default. But you can choose to use CRS 3.1 or CRS 2.2.9 instead. CRS 3.1 offers new rule sets defending against Java infections, an initial set of file upload checks, fixed false positives, and more. CRS 3.0 offers reduced false positives compared with CRS 2.2.9. You can also customize rules to suit your needs.



The WAF protects against the following web vulnerabilities:

- SQL-injection attacks
- Cross-site scripting attacks
- Other common
  attacks, such as command injection, HTTP request smuggling, HTTP response splitting, and remote file inclusion
- HTTP protocol violations
- HTTP protocol anomalies, such as missing host user-agent and accept headers
- Bots, crawlers, and scanners
- Common application misconfigurations (for example, Apache and IIS)

OWASP CRS 3.1

CRS 3.1 includes 13 rule groups, as shown in the following table. Each group contains multiple rules, which can be disabled.

RULE GROUP	DESCRIPTION
General	General group
REQUEST-911-METHOD-ENFORCEMENT	Lock-down methods (PUT, PATCH)
REQUEST-913-SCANNER-DETECTION	Protect against port and environment scanners
REQUEST-920-PROTOCOL-ENFORCEMENT	Protect against protocol and encoding issues
REQUEST-921-PROTOCOL-ATTACK	Protect against header injection, request smuggling, and response splitting
REQUEST-930-APPLICATION-ATTACK-LFI	Protect against file and path attacks
REQUEST-931-APPLICATION-ATTACK-RFI	Protect against remote file inclusion (RFI) attacks
REQUEST-932-APPLICATION-ATTACK-RCE	Protect again remote code execution attacks
REQUEST-933-APPLICATION-ATTACK-PHP	Protect against PHP-injection attacks
REQUEST-941-APPLICATION-ATTACK-XSS	Protect against cross-site scripting attacks
REQUEST-942-APPLICATION-ATTACK-SQLI	Protect against SQL-injection attacks
REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION	Protect against session-fixation attacks
REQUEST-944-APPLICATION-ATTACK-SESSION-JAVA	Protect against JAVA attacks

OWASP CRS 3.0

CRS 3.0 includes 12 rule groups, as shown in the following table. Each group contains multiple rules, which can be disabled.

RULE GROUP	DESCRIPTION
General	General group
REQUEST-911-METHOD-ENFORCEMENT	Lock-down methods (PUT, PATCH)
REQUEST-913-SCANNER-DETECTION	Protect against port and environment scanners
REQUEST-920-PROTOCOL-ENFORCEMENT	Protect against protocol and encoding issues
REQUEST-921-PROTOCOL-ATTACK	Protect against header injection, request smuggling, and response splitting
REQUEST-930-APPLICATION-ATTACK-LFI	Protect against file and path attacks
REQUEST-931-APPLICATION-ATTACK-RFI	Protect against remote file inclusion (RFI) attacks
REQUEST-932-APPLICATION-ATTACK-RCE	Protect again remote code execution attacks

RULE GROUP	DESCRIPTION
REQUEST-933-APPLICATION-ATTACK-PHP	Protect against PHP-injection attacks
REQUEST-941-APPLICATION-ATTACK-XSS	Protect against cross-site scripting attacks
REQUEST-942-APPLICATION-ATTACK-SQLI	Protect against SQL-injection attacks
REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION	Protect against session-fixation attacks

OWASP CRS 2.2.9

CRS 2.2.9 includes 10 rule groups, as shown in the following table. Each group contains multiple rules, which can be disabled.

RULE GROUP	DESCRIPTION
crs_20_protocol_violations	Protect against protocol violations (such as invalid characters or a GET with a request body)
crs_21_protocol_anomalies	Protect against incorrect header information
crs_23_request_limits	Protect against arguments or files that exceed limitations
crs_30_http_policy	Protect against restricted methods, headers, and file types
crs_35_bad_robots	Protect against web crawlers and scanners
crs_40_generic_attacks	Protect against generic attacks (such as session fixation, remote file inclusion, and PHP injection)
crs_41_sql_injection_attacks	Protect against SQL-injection attacks
crs_41_xss_attacks	Protect against cross-site scripting attacks
crs_42_tight_security	Protect against path-traversal attacks
crs_45_trojans	Protect against backdoor trojans

The following rule groups and rules are available when using Web Application Firewall on Application Gateway.

- OWASP 3.1
- OWASP 3.0
- OWASP 2.2.9

Rule sets

General

RULEID	DESCRIPTION
200004	Possible Multipart Unmatched Boundary.

REQUEST-911-METHOD-ENFORCEMENT

RULEID	DESCRIPTION
911100	Method is not allowed by policy

REQUEST-913-SCANNER-DETECTION

RULEID	DESCRIPTION
913100	Found User-Agent associated with security scanner
913101	Found User-Agent associated with scripting/generic HTTP client
913102	Found User-Agent associated with web crawler/bot
913110	Found request header associated with security scanner
913120	Found request filename/argument associated with security scanner

REQUEST-920-PROTOCOL-ENFORCEMENT

RULEID	DESCRIPTION
920100	Invalid HTTP Request Line
920120	Attempted multipart/form-data bypass
920121	Attempted multipart/form-data bypass
920130	Failed to parse request body.
920140	Multipart request body failed strict validation
920160	Content-Length HTTP header is not numeric.
920170	GET or HEAD Request with Body Content.
920171	GET or HEAD Request with Transfer-Encoding.
920180	POST request missing Content-Length Header.
920190	Range = Invalid Last Byte Value.
920200	Range = Too many fields (6 or more)
920201	Range = Too many fields for pdf request (35 or more)
920202	Range = Too many fields for pdf request (6 or more)
920210	Multiple/Conflicting Connection Header Data Found.
920220	URL Encoding Abuse Attack Attempt

RULEID	DESCRIPTION
920230	Multiple URL Encoding Detected
920240	URL Encoding Abuse Attack Attempt
920250	UTF8 Encoding Abuse Attack Attempt
920260	Unicode Full/Half Width Abuse Attack Attempt
920270	Invalid character in request (null character)
920271	Invalid character in request (non printable characters)
920272	Invalid character in request (outside of printable chars below ascii 127)
920273	Invalid character in request (outside of very strict set)
920274	Invalid character in request headers (outside of very strict set)
920280	Request Missing a Host Header
920290	Empty Host Header
920300	Request Missing an Accept Header
920310	Request Has an Empty Accept Header
920311	Request Has an Empty Accept Header
920320	Missing User Agent Header
920330	Empty User Agent Header
920340	Request Containing Content but Missing Content-Type header
920341	Request containing content requires Content-Type header
920350	Host header is a numeric IP address
920360	Argument name too long
920370	Argument value too long
920380	Too many arguments in request
920390	Total arguments size exceeded
920400	Uploaded file size too large

RULEID	DESCRIPTION
920410	Total uploaded files size too large
920420	Request content type is not allowed by policy
920430	HTTP protocol version is not allowed by policy
920440	URL file extension is restricted by policy
920450	HTTP header is restricted by policy (%@{MATCHED_VAR})
920460	Abnormal Escape Characters
920470	Illegal Content-Type header
920480	Restrict charset parameter within the content-type header

REQUEST-921-PROTOCOL-ATTACK

RULEID	DESCRIPTION
921110	HTTP Request Smuggling Attack
921120	HTTP Response Splitting Attack
921130	HTTP Response Splitting Attack
921140	HTTP Header Injection Attack via headers
921150	HTTP Header Injection Attack via payload (CR/LF detected)
921151	HTTP Header Injection Attack via payload (CR/LF detected)
921160	HTTP Header Injection Attack via payload (CR/LF and header-name detected)
921170	HTTP Parameter Pollution
921180	HTTP Parameter Pollution (%{TX.1})

REQUEST-930-APPLICATION-ATTACK-LFI

RULEID	DESCRIPTION
930100	Path Traversal Attack (/../)
930110	Path Traversal Attack (/../)
930120	OS File Access Attempt
930130	Restricted File Access Attempt

REQUEST-931-APPLICATION-ATTACK-RFI

RULEID	DESCRIPTION
931100	Possible Remote File Inclusion (RFI) Attack = URL Parameter using IP Address
931110	Possible Remote File Inclusion (RFI) Attack = Common RFI Vulnerable Parameter Name used w/URL Payload
931120	Possible Remote File Inclusion (RFI) Attack = URL Payload Used w/Trailing Question Mark Character (?)
931130	Possible Remote File Inclusion (RFI) Attack = Off-Domain Reference/Link

REQUEST-932-APPLICATION-ATTACK-RCE

RULEID	DESCRIPTION
932100	Remote Command Execution: Unix Command Injection
932105	Remote Command Execution: Unix Command Injection
932106	Remote Command Execution: Unix Command Injection
932110	Remote Command Execution: Windows Command Injection
932115	Remote Command Execution: Windows Command Injection
932120	Remote Command Execution = Windows PowerShell Command Found
932130	Remote Command Execution = Unix Shell Expression Found
932140	Remote Command Execution = Windows FOR/IF Command Found
932160	Remote Command Execution = Unix Shell Code Found
932170	Remote Command Execution = Shellshock (CVE-2014-6271)
932171	Remote Command Execution = Shellshock (CVE-2014-6271)
932180	Restricted File Upload Attempt
932190	Remote Command Execution: Wildcard bypass technique attempt

REQUEST-933-APPLICATION-ATTACK-PHP

RULEID	DESCRIPTION
933100	PHP Injection Attack = Opening/Closing Tag Found

RULEID	DESCRIPTION
933110	PHP Injection Attack = PHP Script File Upload Found
933111	PHP Injection Attack: PHP Script File Upload Found
933120	PHP Injection Attack = Configuration Directive Found
933130	PHP Injection Attack = Variables Found
933131	PHP Injection Attack: Variables Found
933140	PHP Injection Attack: I/O Stream Found
933150	PHP Injection Attack = High-Risk PHP Function Name Found
933151	PHP Injection Attack: Medium-Risk PHP Function Name Found
933160	PHP Injection Attack = High-Risk PHP Function Call Found
933161	PHP Injection Attack: Low-Value PHP Function Call Found
933170	PHP Injection Attack: Serialized Object Injection
933180	PHP Injection Attack = Variable Function Call Found
933190	PHP Injection Attack: PHP Closing Tag Found

REQUEST-941-APPLICATION-ATTACK-XSS

RULEID	DESCRIPTION
941100	XSS Attack Detected via libinjection
941101	XSS Attack Detected via libinjection
941110	XSS Filter - Category 1 = Script Tag Vector
941130	XSS Filter - Category 3 = Attribute Vector
941140	XSS Filter - Category 4 = Javascript URI Vector
941150	XSS Filter - Category 5 = Disallowed HTML Attributes
941160	NoScript XSS InjectionChecker: HTML Injection
941170	NoScript XSS InjectionChecker: Attribute Injection
941180	Node-Validator Blacklist Keywords
941190	XSS using style sheets

RULEID	DESCRIPTION
941200	XSS using VML frames
941210	XSS using obfuscated Javascript
941220	XSS using obfuscated VB Script
941230	XSS using 'embed' tag
941240	XSS using 'import' or 'implementation' attribute
941250	IE XSS Filters - Attack Detected
941260	XSS using 'meta' tag
941270	XSS using 'link' href
941280	XSS using 'base' tag
941290	XSS using 'applet' tag
941300	XSS using 'object' tag
941310	US-ASCII Malformed Encoding XSS Filter - Attack Detected.
941320	Possible XSS Attack Detected - HTML Tag Handler
941330	IE XSS Filters - Attack Detected.
941340	IE XSS Filters - Attack Detected.
941350	UTF-7 Encoding IE XSS - Attack Detected.

REQUEST-942-APPLICATION-ATTACK-SQLI

RULEID	DESCRIPTION
942100	SQL Injection Attack Detected via libinjection
942110	SQL Injection Attack: Common Injection Testing Detected
942130	SQL Injection Attack: SQL Tautology Detected.
942140	SQL Injection Attack = Common DB Names Detected
942150	SQL Injection Attack
942160	Detects blind sqli tests using sleep() or benchmark().
942170	Detects SQL benchmark and sleep injection attempts including conditional queries

RULEID	DESCRIPTION
942180	Detects basic SQL authentication bypass attempts 1/3
942190	Detects MSSQL code execution and information gathering attempts
942200	Detects MySQL comment-/space-obfuscated injections and backtick termination
942210	Detects chained SQL injection attempts 1/2
942220	Looking for integer overflow attacks, these are taken from skipfish, except 3.0.00738585072
942230	Detects conditional SQL injection attempts
942240	Detects MySQL charset switch and MSSQL DoS attempts
942250	Detects MATCH AGAINST, MERGE and EXECUTE IMMEDIATE injections
942251	Detects HAVING injections
942260	Detects basic SQL authentication bypass attempts 2/3
942270	Looking for basic sql injection. Common attack string for mysql oracle and others
942280	Detects Postgres pg_sleep injection, waitfor delay attacks and database shutdown attempts
942290	Finds basic MongoDB SQL injection attempts
942300	Detects MySQL comments, conditions and ch(a)r injections
942310	Detects chained SQL injection attempts 2/2
942320	Detects MySQL and PostgreSQL stored procedure/function injections
942330	Detects classic SQL injection probings 1/2
942340	Detects basic SQL authentication bypass attempts 3/3
942350	Detects MySQL UDF injection and other data/structure manipulation attempts
942360	Detects concatenated basic SQL injection and SQLLFI attempts
942361	Detects basic SQL injection based on keyword alter or union
942370	Detects classic SQL injection probings 2/2

RULEID	DESCRIPTION
942380	SQL Injection Attack
942390	SQL Injection Attack
942400	SQL Injection Attack
942410	SQL Injection Attack
942420	Restricted SQL Character Anomaly Detection (cookies): # of special characters exceeded (8)
942421	Restricted SQL Character Anomaly Detection (cookies): # of special characters exceeded (3)
942430	Restricted SQL Character Anomaly Detection (args): # of special characters exceeded (12)
942431	Restricted SQL Character Anomaly Detection (args): # of special characters exceeded (6)
942432	Restricted SQL Character Anomaly Detection (args): # of special characters exceeded (2)
942440	SQL Comment Sequence Detected.
942450	SQL Hex Encoding Identified
942460	Meta-Character Anomaly Detection Alert - Repetitive Non-Word Characters
942470	SQL Injection Attack
942480	SQL Injection Attack
942490	Detects classic SQL injection probings 3/3

REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION

RULEID	DESCRIPTION
943100	Possible Session Fixation Attack = Setting Cookie Values in HTML
943110	Possible Session Fixation Attack = SessionID Parameter Name with Off-Domain Referrer
943120	Possible Session Fixation Attack = SessionID Parameter Name with No Referrer

REQUEST-944-APPLICATION-ATTACK-SESSION-JAVA

RULEID	DESCRIPTION
944120	Possible payload execution and remote command execution
944130	Suspicious Java classes
944200	Exploitation of Java deserialization Apache Commons

Next steps

- Customize Web Application Firewall rules using the Azure portal

Custom rules for Web Application Firewall v2 on Azure Application Gateway

2/5/2020 • 5 minutes to read • Edit Online

The Azure Application Gateway Web Application Firewall (WAF) v2 comes with a pre-configured, platform-managed ruleset that offers protection from many different types of attacks. These attacks include cross site scripting, SQL injection, and others. If you're a WAF admin, you may want to write your own rules to augment the core rule set (CRS) rules. Your rules can either block or allow requested traffic based on matching criteria.

Custom rules allow you to create your own rules that are evaluated for each request that passes through the WAF. These rules hold a higher priority than the rest of the rules in the managed rule sets. The custom rules contain a rule name, rule priority, and an array of matching conditions. If these conditions are met, an action is taken (to allow or block).

For example, you can block all requests from an IP address in the range 192.168.5.4/24. In this rule, the operator is *IPMatch*, the matchValues is the IP address range (192.168.5.4/24), and the action is to block the traffic. You also set the rule's name and priority.

Custom rules support using compounding logic to make more advanced rules that address your security needs. For example, (Condition 1 **and** Condition 2) **or** Condition 3). This means that if Condition 1 **and** Condition 2 are met, **or** if Condition 3 is met, the WAF should take the action specified in the custom rule.

Different matching conditions within the same rule are always compounded using **and**. For example, block traffic from a specific IP address, and only if they're using a certain browser.

If you want to **or** two different conditions, the two conditions must be in different rules. For example, block traffic from a specific IP address or block traffic if they're using a specific browser.

> **NOTE**
>
> The maximum number of WAF custom rules is 100. For more information about Application Gateway limits, see Azure subscription and service limits, quotas, and constraints.

Regular expressions are also supported in custom rules, just like in the CRS rulesets. For examples, see Examples 3 and 5 in Create and use custom web application firewall rules.

Allowing vs. blocking

Allowing and blocking traffic is simple with custom rules. For example, you can block all traffic coming from a range of IP addresses. You can make another rule to allow traffic if the request comes from a specific browser.

To allow something, ensure that the `-Action` parameter is set to **Allow**. To block something, ensure that the `-Action` parameter is set to **Block**.

```
$AllowRule = New-AzApplicationGatewayFirewallCustomRule `
   -Name example1 `
   -Priority 2 `
   -RuleType MatchRule `
   -MatchCondition $condition `
   -Action Allow

$BlockRule = New-AzApplicationGatewayFirewallCustomRule `
   -Name example2 `
   -Priority 2 `
   -RuleType MatchRule `
   -MatchCondition $condition `
   -Action Block
```

The previous `$BlockRule` maps to the following custom rule in Azure Resource Manager:

```
"customRules": [
      {
        "name": "blockEvilBot",
        "priority": 2,
        "ruleType": "MatchRule",
        "action": "Block",
        "matchConditions": [
          {
            "matchVariables": [
              {
                "variableName": "RequestHeaders",
                "selector": "User-Agent"
              }
            ],
            "operator": "Contains",
            "negationConditon": false,
            "matchValues": [
              "evilbot"
            ],
            "transforms": [
              "Lowercase"
            ]
          }
        ]
      }
    ],
```

This custom rule contains a name, priority, an action, and the array of matching conditions that must be met for the action to take place. For further explanation of these fields, see the following field descriptions. For example custom rules, see Create and use custom web application firewall rules.

Fields for custom rules

Name [optional]

The name of the rule. It appears in the logs.

Priority [required]

- Determines the rule valuation order. The lower the value, the earlier the evaluation of the rule. The allowable range is from 1-100.
- Must be unique across all custom rules. A rule with priority 40 is evaluated before a rule with priority 80.

Rule type [required]

Currently, must be **MatchRule**.

Match variable [required]

Must be one of the variables:

- RemoteAddr – IP Address/hostname of the remote computer connection
- RequestMethod – HTTP Request method (GET, POST, PUT, DELETE, and so on.)
- QueryString – Variable in the URI
- PostArgs – Arguments sent in the POST body. Custom Rules using this match variable are only applied if the 'Content-Type' header is set to 'application/x-www-form-urlencoded' and 'multipart/form-data'.
- RequestUri – URI of the request
- RequestHeaders – Headers of the request
- RequestBody – This contains the entire request body as a whole. Custom rules using this match variable are only applied if the 'Content-Type' header is set to 'application/x-www-form-urlencoded'.
- RequestCookies – Cookies of the request

Selector [optional]

Describes the field of the matchVariable collection. For example, if the matchVariable is RequestHeaders, the selector could be on the *User-Agent* header.

Operator [required]

Must be one of the following operators:

- IPMatch - only used when Match Variable is *RemoteAddr*
- Equals – input is the same as the MatchValue
- Contains
- LessThan
- GreaterThan
- LessThanOrEqual
- GreaterThanOrEqual
- BeginsWith
- EndsWith
- Regex
- Geomatch (preview)

Negate condition [optional]

Negates the current condition.

Transform [optional]

A list of strings with names of transformations to do before the match is attempted. These can be the following transformations:

- Lowercase
- Trim
- UrlDecode
- UrlEncode
- RemoveNulls
- HtmlEntityDecode

Match values [required]

List of values to match against, which can be thought of as being *OR*'ed. For example, it could be IP addresses or other strings. The value format depends on the previous operator.

Action [required]

- Allow – Authorizes the transaction, skipping all other rules. The specified request is added to the allow list and once matched, the request stops further evaluation and is sent to the backend pool. Rules that are on the allow list aren't evaluated for any further custom rules or managed rules.
- Block – Blocks the transaction based on *SecDefaultAction* (detection/prevention mode). Just like the Allow action, once the request is evaluated and added to the block list, evaluation is stopped and the request is blocked. Any request after that meets the same conditions won't be evaluated and will just be blocked.
- Log – Lets the rule write to the log, but lets the rest of the rules run for evaluation. The other custom rules are evaluated in order of priority, followed by the managed rules.

Geomatch custom rules (preview)

Custom rules let you create tailored rules to suit the exact needs of your applications and security policies. You can restrict access to your web applications by country/region. For more information, see Geomatch custom rules (preview).

Next steps

After you learn about custom rules, create your own custom rules.

Geomatch custom rules (preview)

2/1/2020 • 3 minutes to read • Edit Online

Custom rules allow you to create tailored rules to suit the exact needs of your applications and security policies. Now, you can restrict access to your web applications by country/region. As with all custom rules, this logic can be compounded with other rules to suit the needs of your application.

To create a geo-filtering custom rule, simply select *Geo-location* as the Match Type, and then select the country or countries you want to allow/block from your application. See how to create custom rules in Powershell and more custom rule examples(create-custom-waf-rules.md) for more information.

IMPORTANT

This public preview is provided without a service level agreement and should not be used for production workloads. Certain features may not be supported, may have constrained capabilities, or may not be available in all Azure locations. See the Supplemental Terms of Use for Microsoft Azure Previews for details.

Country codes

If you are using the Geomatch operator, the selectors can be any of the following two-digit country codes.

COUNTRY CODE	COUNTRY NAME
AD	Andorra
AE	United Arab Emirates
AF	Afghanistan
AG	Antigua and Barbuda
AL	Albania
AM	Armenia
AO	Angola
AR	Argentina
AS	American Samoa
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina

COUNTRY CODE	COUNTRY NAME
BB	Barbados
BD	Bangladesh
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BH	Bahrain
BI	Burundi
BJ	Benin
BL	Saint Barthélemy
BN	Brunei Darussalam
BO	Bolivia
BR	Brazil
BS	Bahamas
BT	Bhutan
BW	Botswana
BY	Belarus
BZ	Belize
CA	Canada
CD	Democratic Republic of the Congo
CF	Central African Republic
CH	Switzerland
CI	Cote d'Ivoire
CL	Chile
CM	Cameroon
CN	China

COUNTRY CODE	COUNTRY NAME
CO	Colombia
CR	Costa Rica
CU	Cuba
CV	Cabo Verde
CY	Cyprus
CZ	Czech Republic
DE	Germany
DK	Denmark
DO	Dominican Republic
DZ	Algeria
EC	Ecuador
EE	Estonia
EG	Egypt
ES	Spain
ET	Ethiopia
FI	Finland
FJ	Fiji
FM	Micronesia, Federated States of
FR	France
GB	United Kingdom
GE	Georgia
GF	French Guiana
GH	Ghana
GN	Guinea
GP	Guadeloupe

COUNTRY CODE	COUNTRY NAME
GR	Greece
GT	Guatemala
GY	Guyana
HK	Hong Kong SAR
HN	Honduras
HR	Croatia
HT	Haiti
HU	Hungary
ID	Indonesia
IE	Ireland
IL	Israel
IN	India
IQ	Iraq
IR	Iran, Islamic Republic of
IS	Iceland
IT	Italy
JM	Jamaica
JO	Jordan
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KH	Cambodia
KI	Kiribati
KN	Saint Kitts and Nevis
KP	Korea, Democratic People's Republic of

COUNTRY CODE	COUNTRY NAME
KR	Korea, Republic of
KW	Kuwait
KY	Cayman Islands
KZ	Kazakhstan
LA	Lao People's Democratic Republic
LB	Lebanon
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
LY	Libya
MA	Morocco
MD	Moldova, Republic of
MG	Madagascar
MK	North Macedonia
ML	Mali
MM	Myanmar
MN	Mongolia
MO	Macao SAR
MQ	Martinique
MR	Mauritania
MT	Malta

COUNTRY CODE	COUNTRY NAME
MV	Maldives
MW	Malawi
MX	Mexico
MY	Malaysia
MZ	Mozambique
NA	Namibia
NE	Niger
NG	Nigeria
NI	Nicaragua
NL	Netherlands
NO	Norway
NP	Nepal
NR	Nauru
NZ	New Zealand
OM	Oman
PA	Panama
PE	Peru
PH	Philippines
PK	Pakistan
PL	Poland
PR	Puerto Rico
PT	Portugal
PW	Palau
PY	Paraguay
QA	Qatar

COUNTRY CODE	COUNTRY NAME
RE	Reunion
RO	Romania
RS	Serbia
RU	Russian Federation
RW	Rwanda
SA	Saudi Arabia
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SO	Somalia
SR	Suriname
SS	South Sedan
SV	El Salvador
SY	Syrian Arab Republic
SZ	Swaziland
TC	Turks and Caicos Islands
TG	Togo
TH	Thailand
TN	Tunisia
TR	Turkey
TT	Trinidad and Tobago
TW	Taiwan

COUNTRY CODE	COUNTRY NAME
TZ	Tanzania, United Republic of
UA	Ukraine
UG	Uganda
US	United States
UY	Uruguay
UZ	Uzbekistan
VC	Saint Vincent and the Grenadines
VE	Venezuela
VG	Virgin Islands, British
VI	Virgin Islands, U.S.
VN	Vietnam
ZA	South Africa
ZM	Zambia
ZW	Zimbabwe

Next steps

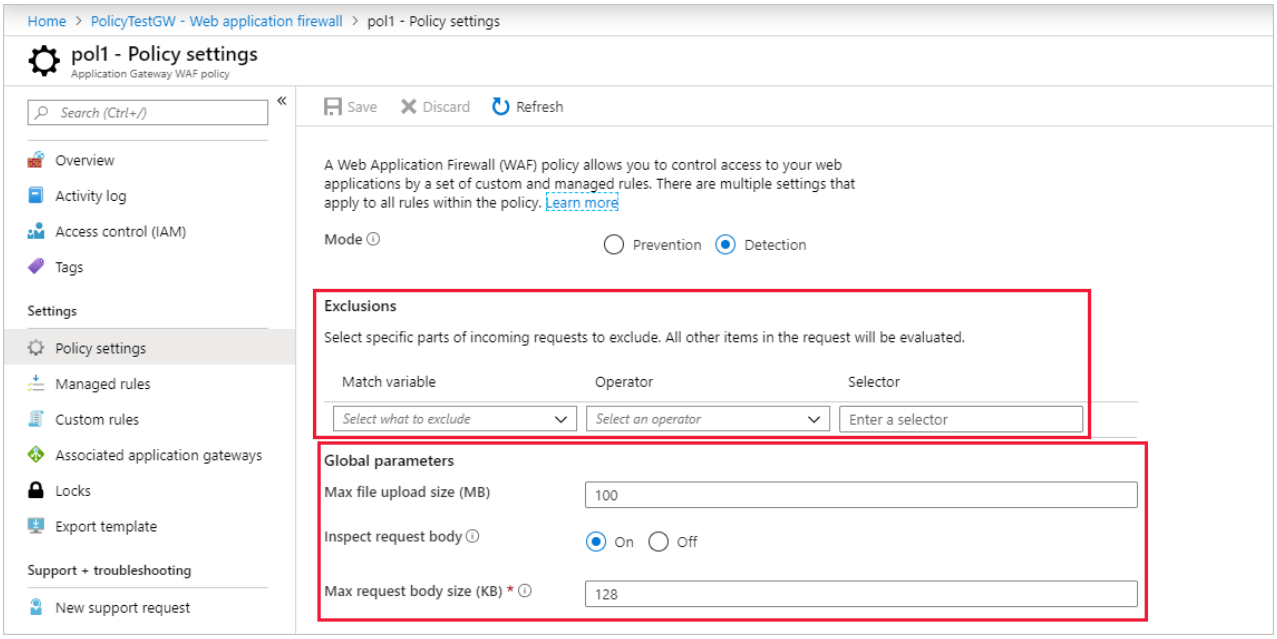
After you learn about custom rules, create your own custom rules.

Web Application Firewall request size limits and exclusion lists

2/21/2020 • 4 minutes to read • Edit Online

The Azure Application Gateway Web Application Firewall (WAF) provides protection for web applications. This article describes WAF request size limits and exclusion lists configuration. These settings are located in the WAF Policy associated to your Application Gateway. To learn more about WAF Policies, see Azure Web Application Firewall on Azure Application Gateway and Create Web Application Firewall policies for Application Gateway

WAF exclusion lists



WAF exclusion lists allow you to omit certain request attributes from a WAF evaluation. A common example is Active Directory inserted tokens that are used for authentication or password fields. Such attributes are prone to contain special characters that may trigger a false positive from the WAF rules. Once an attribute is added to the WAF exclusion list, it isn't considered by any configured and active WAF rule. Exclusion lists are global in scope.

The following attributes can be added to exclusion lists by name. The values of the chosen field aren't evaluated against WAF rules, but their names still are (see Example 1 below, the value of the User-Agent header is excluded from WAF evaluation). The exclusion lists remove inspection of the field's value.

- Request Headers
- Request Cookies
- Request attribute name (args) can be added as an exclusion element, such as:
  - Form field name
  - XML entity
  - JSON entity
  - URL query string args

You can specify an exact request header, body, cookie, or query string attribute match. Or, you can optionally specify partial matches. Exclusion rules are global in scope, and apply to all pages and all rules.

The following are the supported match criteria operators:

- **Equals**: This operator is used for an exact match. As an example, for selecting a header named **bearerToken**, use the equals operator with the selector set as **bearerToken**.
- **Starts with**: This operator matches all fields that start with the specified selector value.
- **Ends with**: This operator matches all request fields that end with the specified selector value.
- **Contains**: This operator matches all request fields that contain the specified selector value.
- **Equals any**: This operator matches all request fields. * will be the selector value.

In all cases matching is case insensitive and regular expression aren't allowed as selectors.

> **NOTE**
> For more information and troubleshooting help, see WAF troubleshooting.

Examples

> **NOTE**
> This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

The following examples demonstrate the use of exclusions.

Example 1

In this example, you want to exclude the user-agent header. The user-agent request header contains a characteristic string that allows the network protocol peers to identify the application type, operating system, software vendor, or software version of the requesting software user agent. For more information, see User-Agent.

There can be any number of reasons to disable evaluating this header. There could be a string that the WAF sees and assumes it's malicious. For example, the classic SQL attack "x=x" in a string. In some cases, this can be legitimate traffic. So you might need to exclude this header from WAF evaluation.

The following Azure PowerShell cmdlet excludes the user-agent header from evaluation:

```
$exclusion1 = New-AzApplicationGatewayFirewallExclusionConfig `
   -MatchVariable "RequestHeaderNames" `
   -SelectorMatchOperator "Equals" `
   -Selector "User-Agent"
```

Example 2

This example excludes the value in the *user* parameter that is passed in the request via the URL. For example, say it's common in your environment for the user field to contain a string that the WAF views as malicious content, so it blocks it. You can exclude the user parameter in this case so that the WAF doesn't evaluate anything in the field.

The following Azure PowerShell cmdlet excludes the user parameter from evaluation:

```
$exclusion2 = New-AzApplicationGatewayFirewallExclusionConfig `
   -MatchVariable "RequestArgNames" `
   -SelectorMatchOperator "StartsWith" `
   -Selector "user"
```

So if the URL `http://www.contoso.com/?user%281%29=fdafdasfda` is passed to the WAF, it won't evaluate the string

fdafdasfda, but it will still evaluate the parameter name **user%281%29**.

WAF request size limits

Web Application Firewall allows you to configure request size limits within lower and upper bounds. The following two size limits configurations are available:

- The maximum request body size field is specified in kilobytes and controls overall request size limit excluding any file uploads. This field can range from 1-KB minimum to 128-KB maximum value. The default value for request body size is 128 KB.
- The file upload limit field is specified in MB and it governs the maximum allowed file upload size. This field can have a minimum value of 1 MB and the following maximums:
  - 100 MB for v1 Medium WAF gateways
  - 500 MB for v1 Large WAF gateways
  - 750 MB for v2 WAF gateways

The default value for file upload limit is 100 MB.

WAF also offers a configurable knob to turn the request body inspection on or off. By default, the request body inspection is enabled. If the request body inspection is turned off, WAF doesn't evaluate the contents of HTTP message body. In such cases, WAF continues to enforce WAF rules on headers, cookies, and URI. If the request body inspection is turned off, then maximum request body size field isn't applicable and can't be set. Turning off the request body inspection allows for messages larger than 128 KB to be sent to WAF, but the message body isn't inspected for vulnerabilities.

Next steps

After you configure your WAF settings, you can learn how to view your WAF logs. For more information, see Application Gateway diagnostics.

Azure Web Application Firewall (WAF) policy overview

2/1/2020 • 3 minutes to read • Edit Online

Web Application Firewall Policies contain all the WAF settings and configurations. This includes exclusions, custom rules, managed rules, and so on. These policies are then associated to an application gateway (global), a listener (per-site), or a path-based rule (per-URI) for them to take effect.

> **NOTE**
>
> Azure Web Application Firewall (WAF) per-site and per-URI policies are in Public Preview.
>
> This public preview is provided without a service-level agreement and shouldn't be used for production workloads. Certain features might not be supported, might have constrained capabilities, or might not be available in all Azure locations. For more information, see Supplemental Terms of Use for Microsoft Azure Previews.

There's no limit on the number of policies you can create. When you create a policy, it must be associated to an application gateway to take effect. It can be associated with any combination of application gateways, listeners, and path-based rules.

Global WAF policy

When you associate a WAF policy globally, every site behind your Application Gateway WAF is protected with the same managed rules, custom rules, exclusions, and any other configured settings.

If you want a single policy to apply to all sites, you can associate the policy with the application gateway. For more information, see Create Web Application Firewall policies for Application Gateway to create and apply a WAF policy using the Azure portal.

Per-site WAF policy

With per-site WAF policies, you can protect multiple sites with differing security needs behind a single WAF by using per-site policies. For example, if there are five sites behind your WAF, you can have five separate WAF policies (one for each listener) to customize the exclusions, custom rules, managed rule sets, and all other WAF settings for each site.

Say your application gateway has a global policy applied to it. Then you apply a different policy to a listener on that application gateway. The listener's policy now takes effect for just that listener. The application gateway's global policy still applies to all other listeners and path-based rules that don't have a specific policy assigned to them.

Per-URI policy

For even more customization down to the URI level, you can associate a WAF policy with a path-based rule. If there are certain pages within a single site that require different policies, you can make changes to the WAF policy that only affect a given URI. This might apply to a payment or sign-in page, or any other URIs that need an even more specific WAF policy than the other sites behind your WAF.

As with per-site WAF policies, more specific policies overrides less specific ones. This means a per-URI policy on a URL path map overrides any per-site or global WAF policy above it.

Example

Say you have three sites: contoso.com, fabrikam.com, and adatum.com all behind the same application gateway. You want a WAF applied to all three sites, but you need added security with adatum.com because that is where customers visit, browse, and purchase products.

You can apply a global policy to the WAF, with some basic settings, exclusions, or custom rules if necessary to stop some false positives from blocking traffic. In this case, there's no need to have global SQL injection rules running because fabrikam.com and contoso.com are static pages with no SQL backend. So you can disable those rules in the global policy.

This global policy is suitable for contoso.com and fabrikam.com, but you need to be more careful with adatum.com where sign in information and payments are handled. You can apply a per-site policy to the adatum listener and leave the SQL rules running. Also assume there's a cookie blocking some traffic, so you can create an exclusion for that cookie to stop the false positive.

The adatum.com/payments URI is where you need to be careful. So apply another policy on that URI and leave all rules enabled, and also remove all exclusions.

In this example, you have a global policy that applies to two sites. You have a per-site policy that applies to one site, and then a per-URI policy that applies to one specific path-based rule. See (insert link here when it exists) how-to create per-site and per-URI policies for the corresponding PowerShell for this example.

Existing WAF configurations

All new Web Application Firewall's WAF settings (custom rules, managed rule set configurations, exclusions, and so on.) exist in a WAF policy. If you have an existing WAF, these settings may still exist in your WAF configuration. For more information about moving to the new WAF policy, see(link: Migrate WAF Config to a WAF Policy).

Next steps

Create per-site and per-URI policies using Azure PowerShell.

Azure Web Application Firewall on Azure Application Gateway bot protection overview

2/4/2020 • 2 minutes to read • Edit Online

Roughly 20% of all Internet traffic comes from bad bots. They do things like scraping, scanning, and looking for vulnerabilities in your web application. When these bots are stopped at the Web Application Firewall (WAF), they can't attack you. They also can't use up your resources and services, such as your backends and other underlying infrastructure.

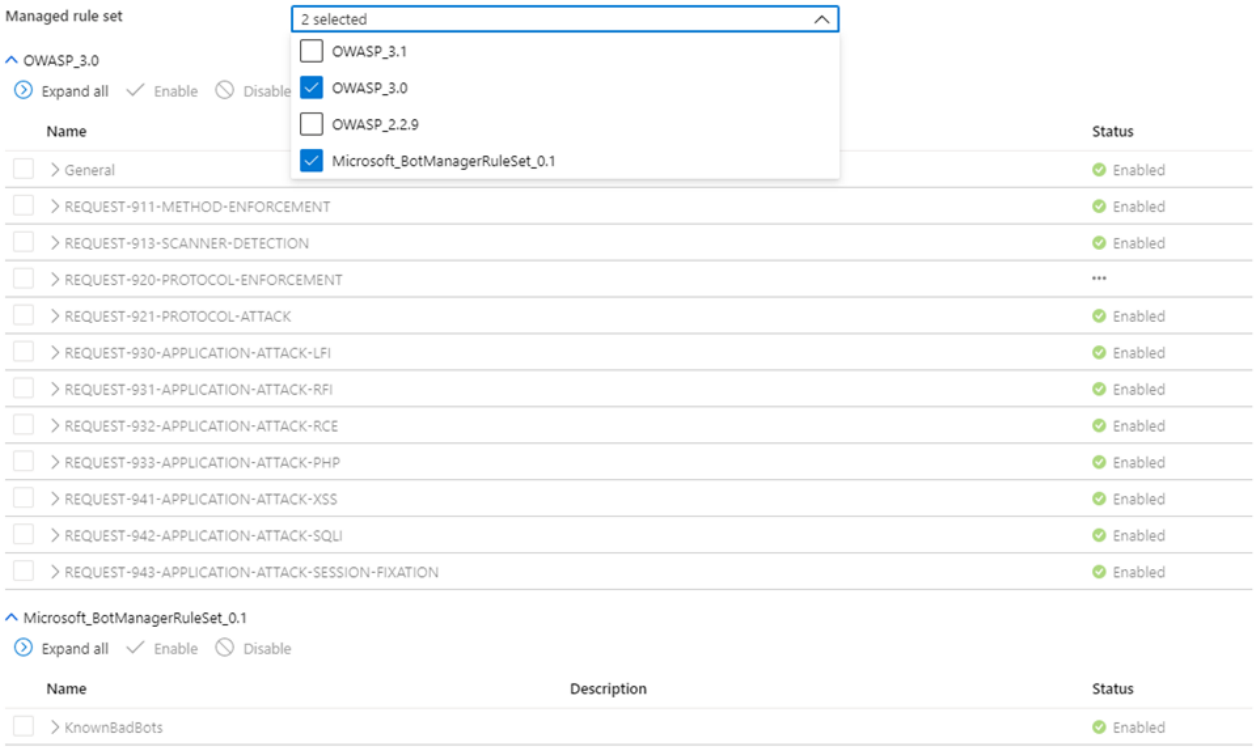
You can enable a managed bot protection rule set for your WAF to block or log requests from known malicious IP addresses. The IP addresses are sourced from the Microsoft Threat Intelligence feed. Intelligent Security Graph powers Microsoft threat intelligence and is used by multiple services including Azure Security Center.

IMPORTANT

The bot protection rule set is currently in public preview and is provided with a preview service level agreement. Certain features may not be supported or may have constrained capabilities. See theSupplemental Terms of Use for Microsoft Azure Previewsfor details.

Use with OWASP rulesets

You can use the Bot Protection ruleset alongside any of the OWASP rulesets (2.2.9, 3.0, and 3.1). Only one OWASP ruleset can be used at any given time. The bot protection ruleset contains an additional rule that appears in its own ruleset. It's titled **Microsoft_BotManagerRuleSet_0.1**, and you can enable or disable it like the other OWASP rules.



Ruleset update

The bot mitigation ruleset list of known bad IP addresses updates multiple times per day from the Microsoft Threat Intelligence feed to stay in sync with the bots. Your web applications are continuously protected even as the bot attack vectors change.

Next steps

- Configure bot protection for Web Application Firewall on Azure Application Gateway (Preview)

Custom rules for Web Application Firewall with Azure Front Door

11/4/2019 • 3 minutes to read • Edit Online

Azure Web Application Firewall (WAF) with Front Door service allows you to control access to your web applications based on the conditions you define. A custom WAF rule consists of a priority number, rule type, match conditions, and an action. There are two types of custom rules: match rules and rate limit rules. A match rule controls access based on a set of matching conditions while a rate limit rule controls access based on matching conditions and the rates of incoming requests. You may disable a custom rule to prevent it from being evaluated, but still keep the configuration.

Priority, match conditions, and action types

You can control access with a custom WAf rule that defines a priority number, a rule type, an array of match conditions, and an action.

- **Priority:** is a unique integer that describes the order of evaluation of WAF rules. Rules with lower priority values are evaluated before rules with higher values. Priority numbers must be unique among all custom rules.
- **Action:** defines how to route a request if a WAF rule is matched. You can choose one of the below actions to apply when a request matches a custom rule.
  - *Allow* - WAF forwards the quest to the back-end, logs an entry in WAF logs and exits.
  - *Block* - Request is blocked, WAF sends response to client without forwarding the request to the back-end. WAF logs an entry in WAF logs.
  - *Log* - WAF logs an entry in WAF logs and continues to evaluate the next rule.
  - *Redirect* - WAF redirects request to a specified URI, logs an entry in WAF logs, and exits.
- **Match condition:** defines a match variable, an operator, and match value. Each rule may contain multiple match conditions. A match condition may be based on geo location, client IP addresses (CIDR), size, or string match. String match can be against a list of match variables.
  - **Match variable:**
    - RequestMethod
    - QueryString
    - PostArgs
    - RequestUri
    - RequestHeader
    - RequestBody
    - Cookies
  - **Operator:**
    - Any: is often used to define default action if no rules are matched. Any is a match all operator.
    - Equal
    - Contains
    - LessThan: size constraint
    - GreaterThan: size constraint

- LessThanOrEqual: size constraint
- GreaterThanOrEqual: size constraint
- BeginsWith
- EndsWith
- Regex

- **Regex** does not support the following operations:
  - Backreferences and capturing subexpressions
  - Arbitrary zero-width assertions
  - Subroutine references and recursive patterns
  - Conditional patterns
  - Backtracking control verbs
  - The \C single-byte directive
  - The \R newline match directive
  - The \K start of match reset directive
  - Callouts and embedded code
  - Atomic grouping and possessive quantifiers
- **Negate [optional]:** You can set the *negate* condition to true if the result of a condition should be negated.
- **Transform [optional]:** A list of strings with names of transformations to do before the match is attempted. These can be the following transformations:
  - Uppercase
  - Lowercase
  - Trim
  - RemoveNulls
  - UrlDecode
  - UrlEncode
- **Match value:** Supported HTTP request method values include:
  - GET
  - POST
  - PUT
  - HEAD
  - DELETE
  - LOCK
  - UNLOCK
  - PROFILE
  - OPTIONS
  - PROPFIND
  - PROPPATCH
  - MKCOL
  - COPY
  - MOVE

Examples

WAF custom rules example based on http parameters

Here is an example that shows the configuration of a custom rule with two match conditions. Requests are from a specified site as defined by referrer, and query string doesn't contain "password".

```
# http rules example
{
  "name": "AllowFromTrustedSites",
  "priority": 1,
  "ruleType": "MatchRule",
  "matchConditions": [
    {
      "matchVariable": "RequestHeader",
      "selector": "Referer",
      "operator": "Equal",
      "negateCondition": false,
      "matchValue": [
        "www.mytrustedsites.com/referpage.html"
      ]
    },
    {
      "matchVariable": "QueryString",
      "operator": "Contains",
      "matchValue": [
        "password"
      ],
      "negateCondition": true
    }
  ],
  "action": "Allow",
  "transforms": []
}
```

An example configuration for blocking "PUT" method is shown as below:

```
# http Request Method custom rules
{
  "name": "BlockPUT",
  "priority": 2,
  "ruleType": "MatchRule",
  "matchConditions": [
    {
      "matchVariable": "RequestMethod",
      "selector": null,
      "operator": "Equal",
      "negateCondition": false,
      "matchValue": [
        "PUT"
      ]
    }
  ],
  "action": "Block",
  "transforms": []
}
```

Size constraint

You may build a custom rule that specifies size constraint on part of an incoming request. For example, below rule blocks a Url that is longer than 100 characters.

```
# http parameters size constraint
{
  "name": "URLOver100",
  "priority": 5,
  "ruleType": "MatchRule",
  "matchConditions": [
    {
      "matchVariable": "RequestUri",
      "selector": null,
      "operator": "GreaterThanOrEqual",
      "negateCondition": false,
      "matchValue": [
        "100"
      ]
    }
  ],
  "action": "Block",
  "transforms": []
}
```

Next steps

- Configure a Web Application Firewall policy using Azure PowerShell
- Learn about web Application Firewall with Front Door
- Learn how to create a Front Door.

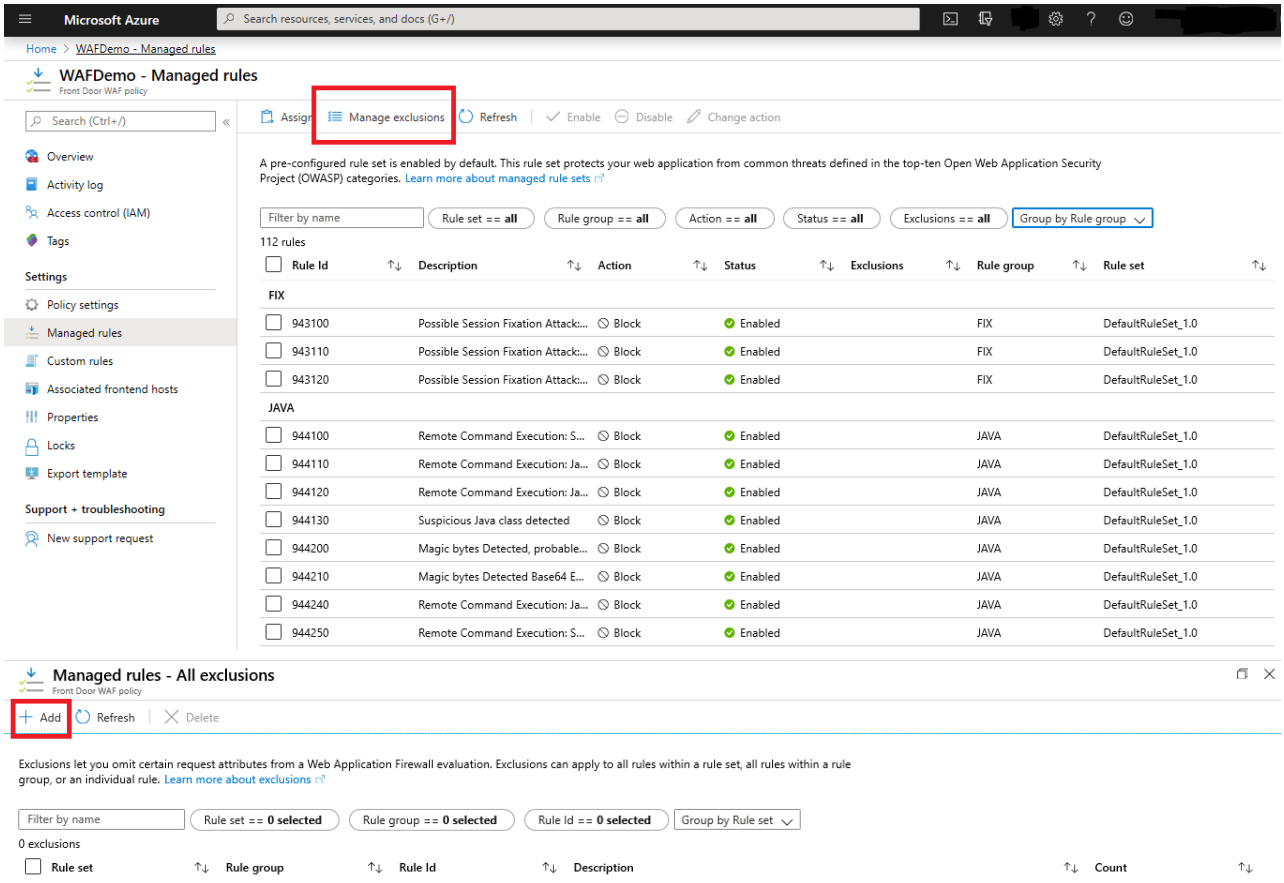
Web Application Firewall (WAF) with Front Door Service exclusion lists

2/27/2020 • 2 minutes to read • Edit Online

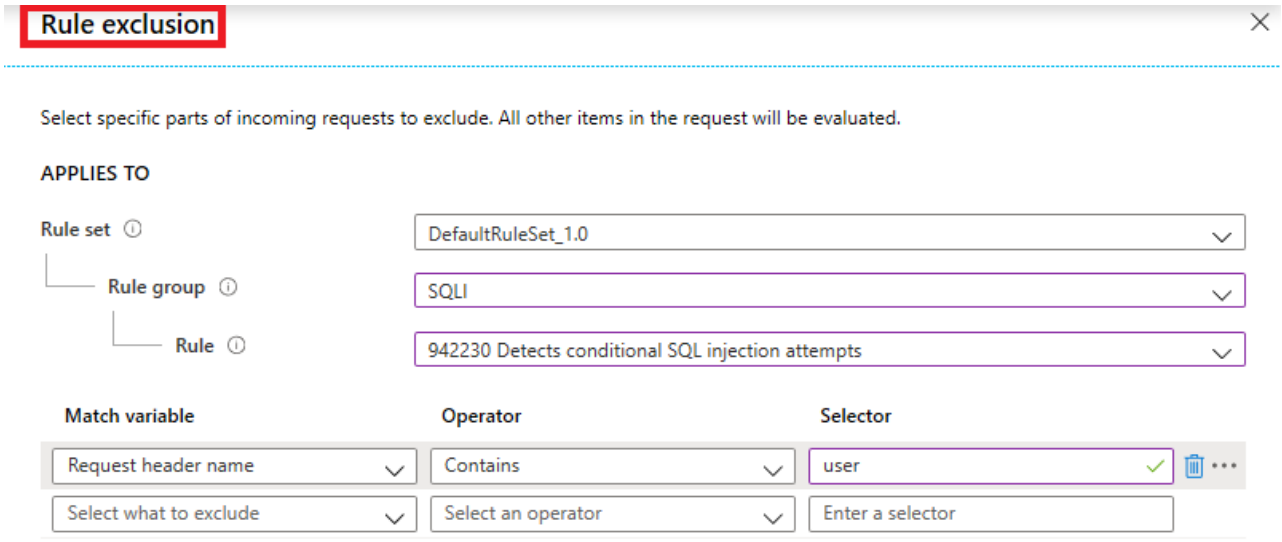
Sometimes Web Application Firewall (WAF) might block a request that you want to allow for your application. For example, Active Directory inserts tokens that are used for authentication. These tokens can contain special characters that may trigger a false positive from the WAF rules. WAF exclusion lists allow you to omit certain request attributes from a WAF evaluation. An exclusion list can be configured using PowerShell, Azure CLI, Rest API, or the Azure portal. The following example shows the Azure portal configuration.

Configure exclusion lists using the Azure portal

Manage exclusions is accessible from WAF portal under **Managed rules**



An example exclusion list:



This example excludes the value in the *user* header field. A valid request may include the *user* field that contains a string that triggers a SQL injection rule. You can exclude the *user* parameter in this case so that the WAF rule doesn't evaluate anything in the field.

The following attributes can be added to exclusion lists by name. The values of the fields you use aren't evaluated against WAF rules, but their names are evaluated. The exclusion lists remove inspection of the field's value.

- Request header name
- Request cookie name
- Query string args name
- Request body post args name

You can specify an exact request header, body, cookie, or query string attribute match. Or, you can optionally specify partial matches. The following operators are the supported match criteria:

- **Equals**: This operator is used for an exact match. For example, to select a header named **bearerToken**, use the equals operator with the selector set as **bearerToken**.
- **Starts with**: This operator matches all fields that start with the specified selector value.
- **Ends with**: This operator matches all request fields that end with the specified selector value.
- **Contains**: This operator matches all request fields that contain the specified selector value.
- **Equals any**: This operator matches all request fields. * is the selector value.

Header and cookie names are case insensitive.

You can apply exclusion list to all rules within the managed rule set, to rules for a specific rule group, or to a single rule as shown in the previous example.

Next steps

After you configure your WAF settings, learn how to view your WAF logs. For more information, see Front Door diagnostics.

Policy settings for Web Application Firewall on Azure Front Door

1/13/2020 • 2 minutes to read • Edit Online

A Web Application Firewall (WAF) policy allows you to control access to your web applications by a set of custom and managed rules. The WAF policy name must be unique. You will receive a validation error if you try to use an existing name. There are multiple policy level settings that apply to all rules specified for that policy as described in this article.

WAF state

A WAF policy for Front Door can be in one of the following two states:

- **Enabled:** When a policy is enabled, WAF is actively inspecting incoming requests and takes corresponding actions according to rule definitions
- **Disabled:** - When a policy is disabled, WAF inspection is paused. Incoming requests will bypass WAF and are sent to back-ends based on Front Door routing.

WAF mode

WAF policy can be configured to run in the following two modes:

- **Detection mode** When run in detection mode, WAF does not take any actions other than monitor and log the request and its matched WAF rule to WAF logs. Turn on logging diagnostics for Front Door (when using portal, this can be achieved by going to the **Diagnostics** section in the Azure portal).
- **Prevention mode** When configured to run in prevention mode, WAF takes the specified action if a request matches a rule. Any matched requests are also logged in the WAF logs.

WAF response for blocked requests

By default, when WAF blocks a request because of a matched rule, it returns a 403 status code with - **The request is blocked** message. A reference string is also returned for logging.

You can define a custom response status code and response message when a request is blocked by WAF. The following custom status codes are supported:

- 200 OK
- 403 Forbidden
- 405 Method not allowed
- 406 Not acceptable
- 429 Too many requests

Custom response status code and response message is a policy level setting. Once it is configured, all blocked requests get the same custom response status and response message.

URI for redirect action

You are required to define a URI to redirect requests to if the **REDIRECT** action is selected for any of the rules contained in a WAF policy. This redirect URI needs to be a valid HTTP(S) site and once configured, all requests

matching rules with a "REDIRECT" action will be redirected to the specified site.

Next steps

- Learn how to define WAF custom responses

What is geo-filtering on a domain for Azure Front Door?

11/4/2019 • 3 minutes to read • Edit Online

By default, Azure Front Door Service responds to user requests regardless of the location of the user making the request. However, in some cases, you may want to restrict access to your web applications by country/region. Web application firewall (WAF) service at Front Door enables you to define a policy using custom access rules for specific path on your endpoint to allow or block access from specified countries/regions.

A WAF policy usually includes a set of custom rules. A rule consists of match conditions, an action, and a priority. In match condition, you define a match variable, operator, and match value. For geo filtering rule, match variable is REMOTE_ADDR, operator is GeoMatch, value is the two letter country code of interest. You may combine a GeoMatch condition and a REQUEST_URI string match condition to create a path-based geo-filtering rule.

You can configure a geo-filtering policy for your Front Door by either using Azure PowerShell or by using our quickstart template.

Country code reference

COUNTRY CODE	COUNTRY NAME
AD	Andorra
AE	United Arab Emirates
AF	Afghanistan
AG	Antigua and Barbuda
AL	Albania
AM	Armenia
AO	Angola
AR	Argentina
AS	American Samoa
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados

COUNTRY CODE	COUNTRY NAME
BD	Bangladesh
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BH	Bahrain
BI	Burundi
BJ	Benin
BL	Saint Barthélemy
BN	Brunei Darussalam
BO	Bolivia
BR	Brazil
BS	Bahamas
BT	Bhutan
BW	Botswana
BY	Belarus
BZ	Belize
CA	Canada
CD	Democratic Republic of the Congo
CF	Central African Republic
CH	Switzerland
CI	Cote d'Ivoire
CL	Chile
CM	Cameroon
CN	China
CO	Colombia

COUNTRY CODE	COUNTRY NAME
CR	Costa Rica
CU	Cuba
CV	Cabo Verde
CY	Cyprus
CZ	Czech Republic
DE	Germany
DK	Denmark
DO	Dominican Republic
DZ	Algeria
EC	Ecuador
EE	Estonia
EG	Egypt
ES	Spain
ET	Ethiopia
FI	Finland
FJ	Fiji
FM	Micronesia, Federated States of
FR	France
GB	United Kingdom
GE	Georgia
GF	French Guiana
GH	Ghana
GN	Guinea
GP	Guadeloupe
GR	Greece

COUNTRY CODE	COUNTRY NAME
GT	Guatemala
GY	Guyana
HK	Hong Kong SAR
HN	Honduras
HR	Croatia
HT	Haiti
HU	Hungary
ID	Indonesia
IE	Ireland
IL	Israel
IN	India
IQ	Iraq
IR	Iran, Islamic Republic of
IS	Iceland
IT	Italy
JM	Jamaica
JO	Jordan
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KH	Cambodia
KI	Kiribati
KN	Saint Kitts and Nevis
KP	Korea, Democratic People's Republic of
KR	Korea, Republic of

COUNTRY CODE	COUNTRY NAME
KW	Kuwait
KY	Cayman Islands
KZ	Kazakhstan
LA	Lao People's Democratic Republic
LB	Lebanon
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
LY	Libya
MA	Morocco
MD	Moldova, Republic of
MG	Madagascar
MK	North Macedonia
ML	Mali
MM	Myanmar
MN	Mongolia
MO	Macao SAR
MQ	Martinique
MR	Mauritania
MT	Malta
MV	Maldives

COUNTRY CODE	COUNTRY NAME
MW	Malawi
MX	Mexico
MY	Malaysia
MZ	Mozambique
NA	Namibia
NE	Niger
NG	Nigeria
NI	Nicaragua
NL	Netherlands
NO	Norway
NP	Nepal
NR	Nauru
NZ	New Zealand
OM	Oman
PA	Panama
PE	Peru
PH	Philippines
PK	Pakistan
PL	Poland
PR	Puerto Rico
PT	Portugal
PW	Palau
PY	Paraguay
QA	Qatar
RE	Reunion

COUNTRY CODE	COUNTRY NAME
RO	Romania
RS	Serbia
RU	Russian Federation
RW	Rwanda
SA	Saudi Arabia
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SO	Somalia
SR	Suriname
SS	South Sedan
SV	El Salvador
SY	Syrian Arab Republic
SZ	Swaziland
TC	Turks and Caicos Islands
TG	Togo
TH	Thailand
TN	Tunisia
TR	Turkey
TT	Trinidad and Tobago
TW	Taiwan
TZ	Tanzania, United Republic of

COUNTRY CODE	COUNTRY NAME
UA	Ukraine
UG	Uganda
US	United States
UY	Uruguay
UZ	Uzbekistan
VC	Saint Vincent and the Grenadines
VE	Venezuela
VG	Virgin Islands, British
VI	Virgin Islands, U.S.
VN	Vietnam
ZA	South Africa
ZM	Zambia
ZW	Zimbabwe

Next steps

- Learn about application layer security with Front Door.
- Learn how to create a Front Door.

Frequently asked questions for Azure Web Application Firewall on Azure Front Door Service

11/4/2019 • 3 minutes to read • Edit Online

This article answers common questions about Azure web application firewall (WAF) features and functionality.

What is Azure WAF?

Azure WAF is a web application firewall that helps protect your web applications from common threats such as SQL injection, cross-site scripting, and other web exploits. You can define a WAF policy consisting of a combination of custom and managed rules to control access to your web applications.

An Azure WAF policy can be applied to web applications hosted on Application Gateway or Azure Front Door services.

What is WAF on Azure Front Door Service?

Azure Front Door is a highly scalable, globally distributed application and content delivery network. Azure WAF, when integrated with Front Door, stops denial-of-service and targeted application attacks at the Azure network edge, close to attack sources before they enter your virtual network, offers protection without sacrificing performance.

Does Azure WAF support HTTPS?

Front Door Service offers SSL offloading. WAF is natively integrated with Front Door and can inspect a request after it's decrypted.

Does Azure WAF support IPv6?

Yes. You can configure IP restriction for IPv4 and IPv6.

How up-to-date are the managed rule sets?

We do our best to keep up with changing threat landscape. Once a new rule is updated, it's added to the Default Rule Set with a new version number.

What is the propagation time if I make a change to my WAF policy?

Deploying a WAF policy globally usually takes about 5 minutes and often completes sooner.

Can WAF policies be different for different regions?

When integrated with Front Door Service, WAF is a global resource. Same configuration applies across all Front Door locations.

How do I limit access to my back-end to be from Front Door only?

You may configure IP Access Control List in your back-end to allow for only Front Door outbound IP address ranges and deny any direct access from Internet. Service tags are supported for you to use on your virtual network. Additionally, you can verify that the X-Forwarded-Host HTTP header field is valid for your web application.

Which Azure WAF options should I choose?

There are two options when applying WAF policies in Azure. WAF with Azure Front Door is a globally distributed, edge security solution. WAF with Application Gateway is a regional, dedicated solution. We recommend you choose a solution based on your overall performance and security requirements. For more information, see Load-balancing with Azure's application delivery suite.

Do you support same WAF features in all integrated platforms?

Currently, ModSec CRS 2.2.9, CRS 3.0, and CRS 3.1 rules are only supported with WAF on Application Gateway. Rate-limiting, geo-filtering, and Azure managed Default Rule Set rules are supported only with WAF on Azure Front Door.

Is DDoS protection integrated with Front Door?

Globally distributed at Azure network edges, Azure Front Door can absorb and geographically isolate large volume attacks. You can create custom WAF policy to automatically block and rate limit http(s) attacks that have known signatures. Further more, you can enable DDoS Protection Standard on the VNet where your back-ends are deployed. Azure DDoS Protection Standard customers receive additional benefits including cost protection, SLA guarantee, and access to experts from DDoS Rapid Response Team for immediate help during an attack.

Why do additional requests above the threshold configured for my rate limit rule get passed to my backend server?

A rate limit rule can limit abnormally high traffic from any client IP address. You may configure a threshold on the number of web requests allowed from a client IP address during a one-minute or five-minute duration. For granular rate control, rate limiting can be combined with additional match conditions such as HTTP(S) parameter matching.

Requests from the same client often arrive at the same Front Door server. In that case, you'll see additional requests above the threshold get blocked immediately.

However, it's possible that requests from the same client may arrive at a different Front Door server that has not refreshed the rate limit counter yet. For example, the client may open a new connection for each request and the threshold is low. In this case, the first request to the new Front Door server would pass the rate limit check. A rate limit threshold is usually set high to defend against denial of service attacks from any client IP address. For a very low threshold, you may see additional requests above the threshold get through.

Next steps

- Learn about Azure Web Application Firewall.
- Learn more about Azure Front Door.

Create Web Application Firewall policies for Application Gateway

2/7/2020 • 4 minutes to read • Edit Online

Associating a WAF policy with listeners allows for multiple sites behind a single WAF to be protected by different policies. For example, if there are five sites behind your WAF, you can have five separate WAF policies (one for each listener) to customize the exclusions, custom rules, and managed rulesets for one site without effecting the other four. If you want a single policy to apply to all sites, you can just associate the policy with the Application Gateway, rather than the individual listeners, to make it apply globally. Policies can also be applied to a path-based routing rule.

You can make as many policies as you want. Once you create a policy, it must be associated to an Application Gateway to go into effect, but it can be associated with any combination of Application Gateways and listeners.

If your Application Gateway has a policy applied, and then you apply a different policy to a listener on that Application Gateway, the listener's policy will take effect, but just for the listener(s) that they're assigned to. The Application Gateway policy still applies to all other listeners that don't have a specific policy assigned to them.

> **NOTE**
>
> Per-site and per-URI WAF Policies are in Public Preview. That means this feature is subject to Microsoft's Supplemental Terms of Use. For more information, see Supplemental Terms of Use for Microsoft Azure Previews.

All new Web Application Firewall's WAF settings (custom rules, managed rulset configurations, exclusions, etc.) live inside of a WAF Policy. If you have an existing WAF, these settings may still exist in your WAF config. For steps on how to move to the new WAF Policy, see Migrate your WAF Config to a WAF Policy later in this article.

Create a policy

First, create a basic WAF policy with a managed Default Rule Set (DRS) using the Azure portal.

1. On the upper left side of the portal, select **Create a resource**. Search for **WAF**, select **Web Application Firewall**, then select **Create**.
2. On **Create a WAF policy** page, **Basics** tab, enter or select the following information, accept the defaults for the remaining settings, and then select **Review + create**:

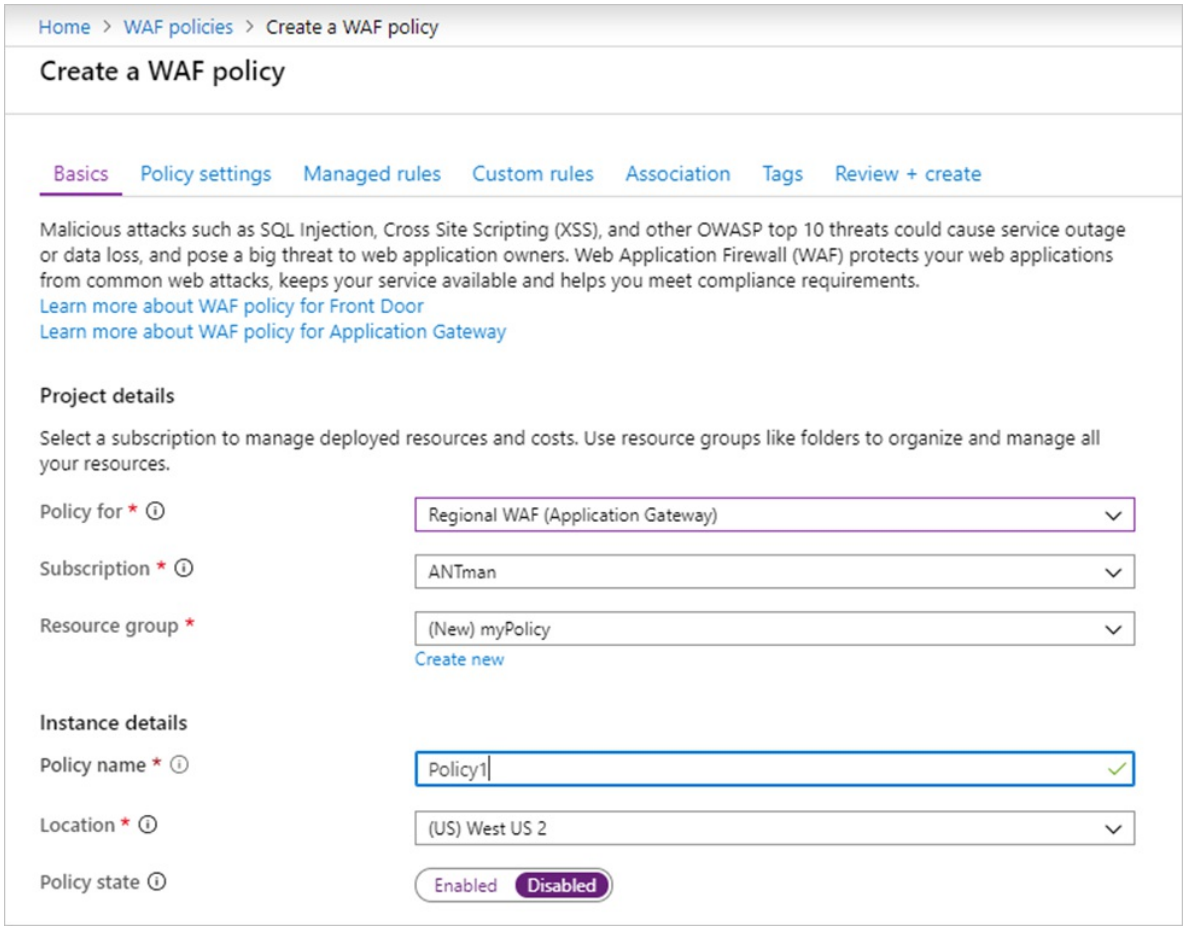
SETTING	VALUE
Policy for	Regional WAF (Application Gateway)
Subscription	Select your subscription name
Resource group	Select your resource group
Policy name	Type a unique name for your WAF policy.

3. On the **Association** tab, enter one of the following settings, then select **Add**:

SETTING	VALUE
Associate Application Gateway	Select your Application Gateway profile name.
Associate Listeners	Select the name of your Application Gateway Listener, then select **Add**.

> **NOTE**
> If you assign a policy to your Application Gateway (or listener) that already has a policy in place, the original policy is overwritten and replaced by the new policy.

4. Select **Review + create**, then select **Create**.

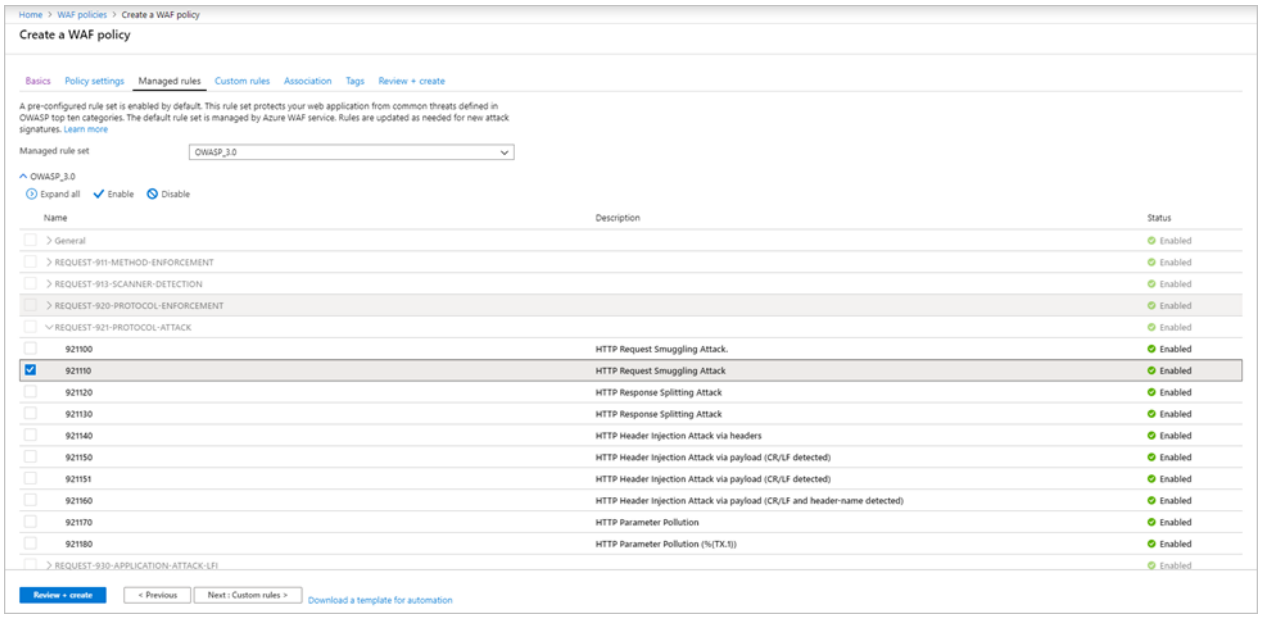


Configure WAF rules (optional)

When you create a WAF policy, by default it is in *Detection* mode. In Detection mode, WAF doesn't block any requests. Instead, the matching WAF rules are logged in the WAF logs. To see WAF in action, you can change the mode settings to *Prevention*. In Prevention mode, matching rules defined in the CRS Ruleset you selected are blocked and/or logged in the WAF logs.

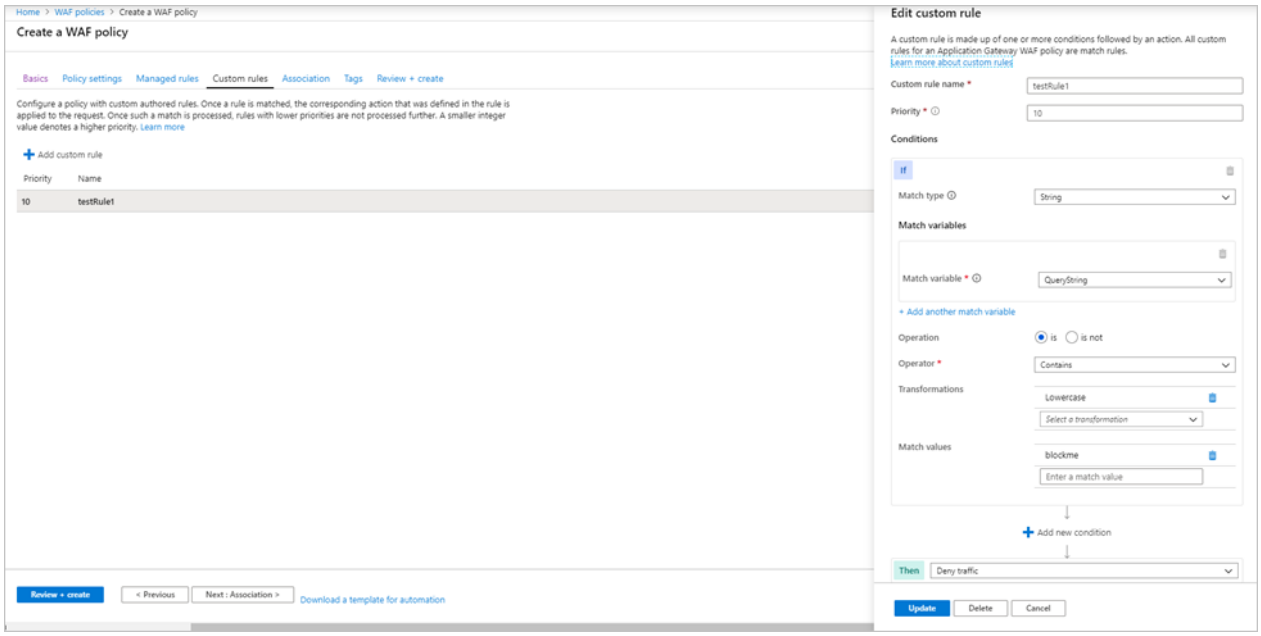
Managed rules

Azure-managed OWASP rules are enabled by default. To disable an individual rule within a rule group, expand the rules within that rule group, select the check box in front of the rule number, and select **Disable** on the tab above.



Custom rules

To create a custom rule, select **Add custom rule** under the **Custom rules** tab. This opens the custom rule configuration page. The following screenshot shows an example custom rule configured to block a request if the query string contains the text *blockme*.

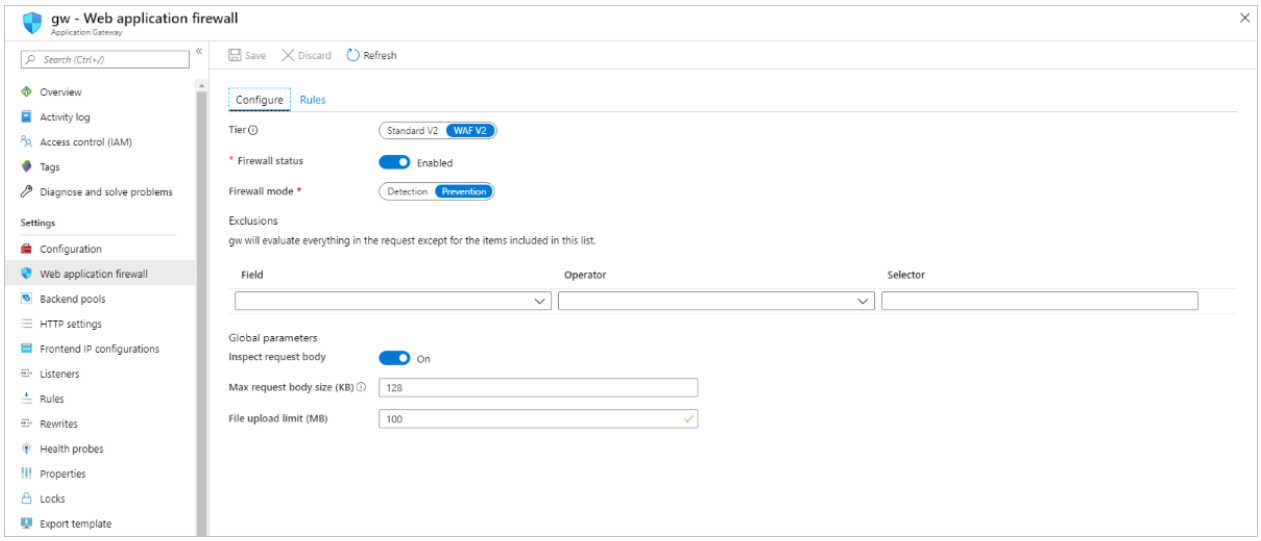


Migrate your WAF Config to a WAF Policy

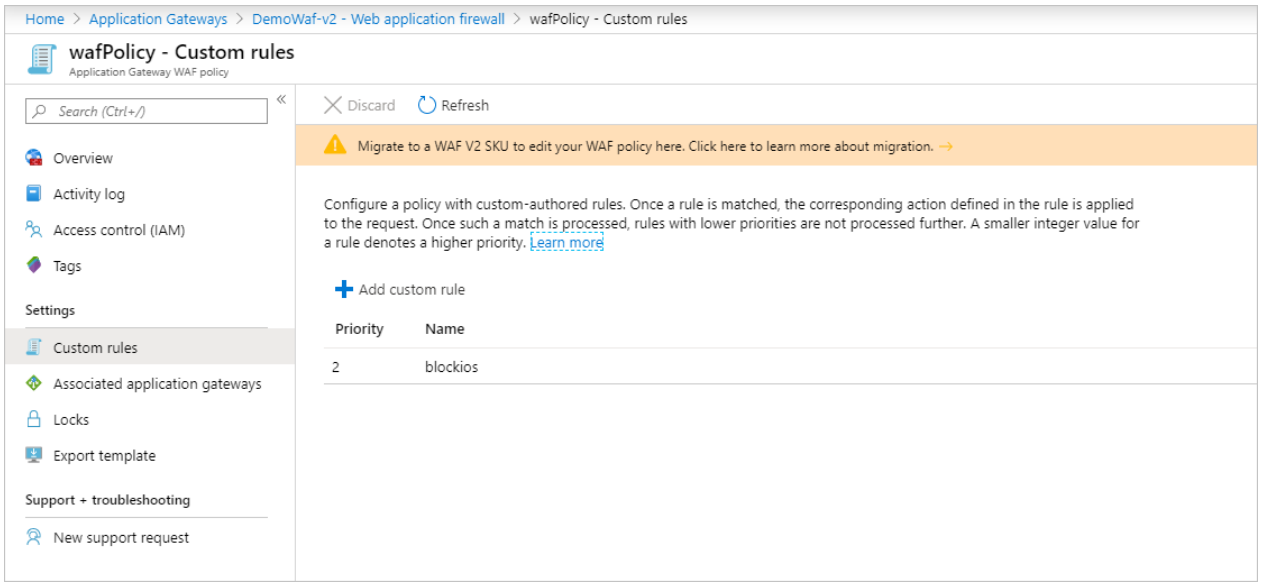
If you have an existing WAF, you may have noticed some changes in the portal. First you need to identify what kind of Policy you've enabled on your WAF. There are three potential states:

- No WAF Policy
- Custom Rules only Policy
- WAF Policy

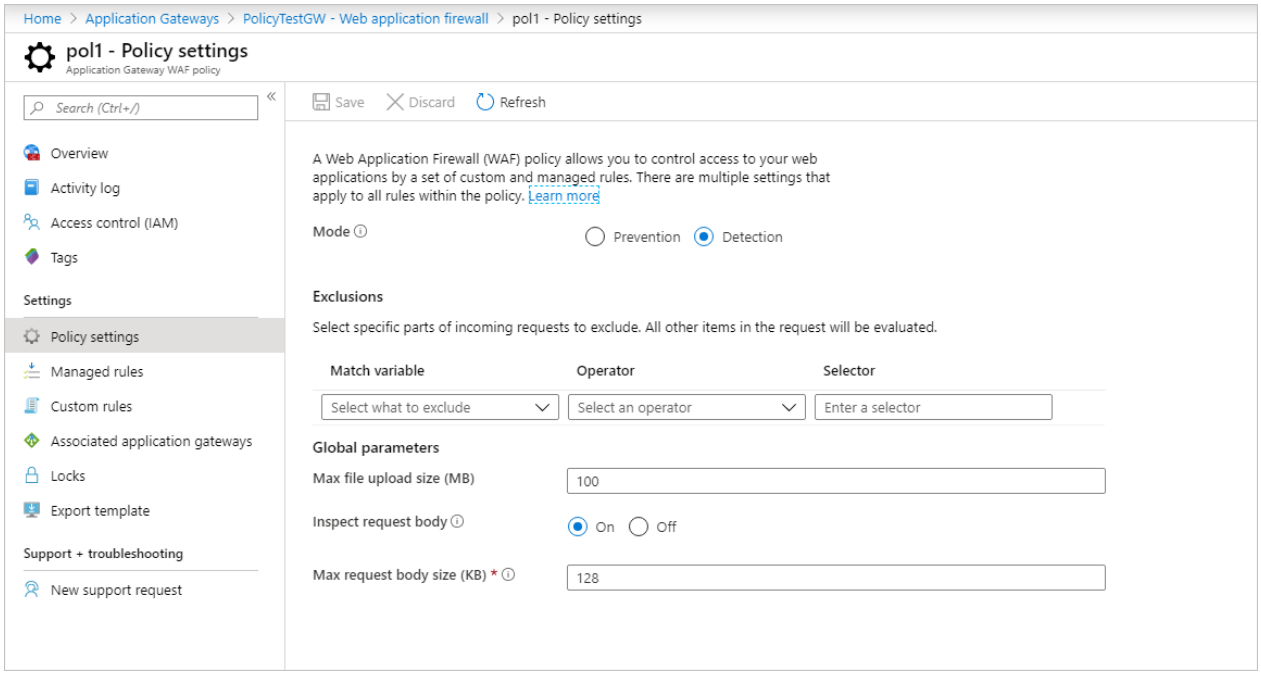
You can tell which state your WAF is in by looking at it in the portal. If the WAF settings are visible and can be changed from within the Application Gateway view, your WAF is in state 1.



If you select **Web Application Firewall** and it shows you an associated policy, the WAF is in state 2 or state 3. After navigating to the policy, if it shows **only** custom rules, and Associated Application Gateways, then it's a Custom Rules only Policy.



If it also shows Policy Settings and Managed Rules, then it's a full Web Application Firewall policy.



Migrate to WAF Policy

If you have a Custom Rules only WAF Policy, then you may want to move to the new WAF Policy. Going forward, the firewall policy will support WAF policy settings, managed rulesets, exclusions, and disabled rule-groups. Essentially, all the WAF configurations that were previously done inside the Application Gateway are now done through the WAF Policy.

Edits to the custom rule only WAF policy are disabled. To edit any WAF settings such as disabling rules, adding exclusions, etc. you have to migrate to a new top-level firewall policy resource.

To do so, create a *Web Application Firewall Policy* and associate it to your Application Gateway(s) and listener(s) of choice. This new Policy **must** be exactly the same as the current WAF config, meaning every custom rule, exclusion, disabled rule, etc. must be copied into the new Policy you are creating. Once you have a Policy associated with your Application Gateway, then you can continue to make changes to your WAF rules and settings. You can also do this with Azure PowerShell. For more information, see Associate a WAF policy with an existing Application Gateway.

Optionally, you can use a migration script to migrate to a WAF policy. For more information, see Migrate Web Application Firewall policies using Azure PowerShell.

Next steps

Learn more about Web Application Firewall CRS rule groups and rules.

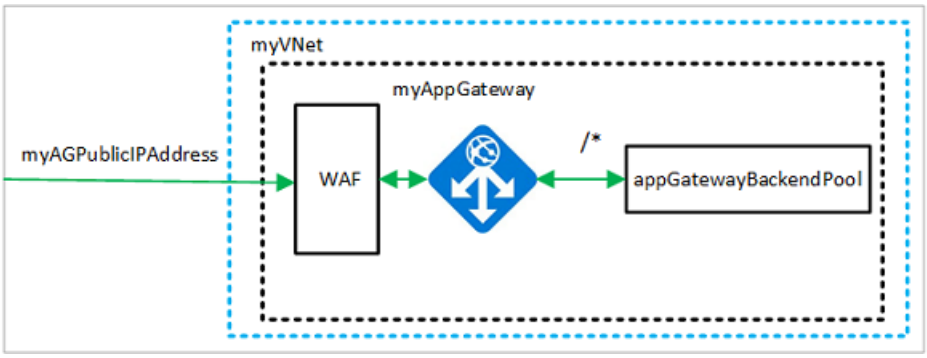
Enable Web Application Firewall using Azure PowerShell

11/13/2019 • 6 minutes to read • Edit Online

You can restrict traffic on an application gateway with a Web Application Firewall (WAF). The WAF uses OWASP rules to protect your application. These rules include protection against attacks such as SQL injection, cross-site scripting attacks, and session hijacks.

In this article, you learn how to:

- Set up the network
- Create an application gateway with WAF enabled
- Create a virtual machine scale set
- Create a storage account and configure diagnostics



If you prefer, you can complete this article using the Azure portal or the Azure CLI.

If you don't have an Azure subscription, create a free account before you begin.

> **NOTE**
> This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	

OPTION	EXAMPLE/LINK
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl**+**Shift**+**V** on Windows and Linux or by selecting **Cmd**+**Shift**+**V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the PowerShell locally, this article requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the version. If you need to upgrade, see Install Azure PowerShell module. If you're running PowerShell locally, you also need to run `Login-AzAccount` to create a connection with Azure.

Create a resource group

A resource group is a logical container into which Azure resources are deployed and managed. Create an Azure resource group using New-AzResourceGroup.

```
New-AzResourceGroup -Name myResourceGroupAG -Location eastus
```

Create network resources

Create the subnet configurations named *myBackendSubnet* and *myAGSubnet* using New-AzVirtualNetworkSubnetConfig. Create the virtual network named *myVNet* using New-AzVirtualNetwork with the subnet configurations. And finally, create the public IP address named *myAGPublicIPAddress* using New-AzPublicIpAddress. These resources are used to provide network connectivity to the application gateway and its associated resources.

```
$backendSubnetConfig = New-AzVirtualNetworkSubnetConfig `
  -Name myBackendSubnet `
  -AddressPrefix 10.0.1.0/24

$agSubnetConfig = New-AzVirtualNetworkSubnetConfig `
  -Name myAGSubnet `
  -AddressPrefix 10.0.2.0/24

$vnet = New-AzVirtualNetwork `
  -ResourceGroupName myResourceGroupAG `
  -Location eastus `
  -Name myVNet `
  -AddressPrefix 10.0.0.0/16 `
  -Subnet $backendSubnetConfig, $agSubnetConfig

$pip = New-AzPublicIpAddress `
  -ResourceGroupName myResourceGroupAG `
  -Location eastus `
  -Name myAGPublicIPAddress `
  -AllocationMethod Static `
  -Sku Standard
```

Create an application gateway

In this section, you create resources that support the application gateway, and then finally create it and a WAF. The resources that you create include:

- *IP configurations and frontend port* - Associates the subnet that you previously created to the application gateway and assigns a port to use to access it.
- *Default pool* - All application gateways must have at least one backend pool of servers.
- *Default listener and rule* - The default listener listens for traffic on the port that was assigned and the default rule sends traffic to the default pool.

Create the IP configurations and frontend port

Associate *myAGSubnet* that you previously created to the application gateway using New-AzApplicationGatewayIPConfiguration. Assign *myAGPublicIPAddress* to the application gateway using New-AzApplicationGatewayFrontendIPConfig.

```
$vnet = Get-AzVirtualNetwork `
  -ResourceGroupName myResourceGroupAG `
  -Name myVNet

$subnet=$vnet.Subnets[1]

$gipconfig = New-AzApplicationGatewayIPConfiguration `
  -Name myAGIPConfig `
  -Subnet $subnet

$fipconfig = New-AzApplicationGatewayFrontendIPConfig `
  -Name myAGFrontendIPConfig `
  -PublicIPAddress $pip

$frontendport = New-AzApplicationGatewayFrontendPort `
  -Name myFrontendPort `
  -Port 80
```

Create the backend pool and settings

Create the backend pool named *appGatewayBackendPool* for the application gateway using New-AzApplicationGatewayBackendAddressPool. Configure the settings for the backend address pools using New-

AzApplicationGatewayBackendHttpSettings.

```
$defaultPool = New-AzApplicationGatewayBackendAddressPool `
  -Name appGatewayBackendPool

$poolSettings = New-AzApplicationGatewayBackendHttpSettings `
  -Name myPoolSettings `
  -Port 80 `
  -Protocol Http `
  -CookieBasedAffinity Enabled `
  -RequestTimeout 120
```

Create the default listener and rule

A listener is required to enable the application gateway to route traffic appropriately to the backend address pools. In this example, you create a basic listener that listens for traffic at the root URL.

Create a listener named *mydefaultListener* using New-AzApplicationGatewayHttpListener with the frontend configuration and frontend port that you previously created. A rule is required for the listener to know which backend pool to use for incoming traffic. Create a basic rule named *rule1* using New-AzApplicationGatewayRequestRoutingRule.

```
$defaultlistener = New-AzApplicationGatewayHttpListener `
  -Name mydefaultListener `
  -Protocol Http `
  -FrontendIPConfiguration $fipconfig `
  -FrontendPort $frontendport

$frontendRule = New-AzApplicationGatewayRequestRoutingRule `
  -Name rule1 `
  -RuleType Basic `
  -HttpListener $defaultlistener `
  -BackendAddressPool $defaultPool `
  -BackendHttpSettings $poolSettings
```

Create the application gateway with the WAF

Now that you created the necessary supporting resources, specify parameters for the application gateway using New-AzApplicationGatewaySku. Specify the Firewall Policy using New-AzApplicationGatewayFirewallPolicy. And then create the application gateway named *myAppGateway* using New-AzApplicationGateway.

```
$sku = New-AzApplicationGatewaySku `
  -Name WAF_v2 `
  -Tier WAF_v2 `
  -Capacity 2

$policySetting = New-AzApplicationGatewayFirewallPolicySetting -Mode Prevention -State Enabled -
MaxRequestBodySizeInKb 100 -MaxFileUploadInMb 256

$wafPolicy = New-AzApplicationGatewayFirewallPolicy -Name wafpolicyNew -ResourceGroup $rgname -Location
$location -PolicySetting $PolicySetting

$appgw = New-AzApplicationGateway `
  -Name myAppGateway `
  -ResourceGroupName myResourceGroupAG `
  -Location eastus `
  -BackendAddressPools $defaultPool `
  -BackendHttpSettingsCollection $poolSettings `
  -FrontendIpConfigurations $fipconfig `
  -GatewayIpConfigurations $gipconfig `
  -FrontendPorts $frontendport `
  -HttpListeners $defaultlistener `
  -RequestRoutingRules $frontendRule `
  -Sku $sku `
  -FirewallPolicy $wafPolicy
```

Create a virtual machine scale set

In this example, you create a virtual machine scale set to provide servers for the backend pool in the application gateway. You assign the scale set to the backend pool when you configure the IP settings.

```
$vnet = Get-AzVirtualNetwork `
  -ResourceGroupName myResourceGroupAG `
  -Name myVNet

$appgw = Get-AzApplicationGateway `
  -ResourceGroupName myResourceGroupAG `
  -Name myAppGateway

$backendPool = Get-AzApplicationGatewayBackendAddressPool `
  -Name appGatewayBackendPool `
  -ApplicationGateway $appgw

$ipConfig = New-AzVmssIpConfig `
  -Name myVmssIPConfig `
  -SubnetId $vnet.Subnets[1].Id `
  -ApplicationGatewayBackendAddressPoolsId $backendPool.Id

$vmssConfig = New-AzVmssConfig `
  -Location eastus `
  -SkuCapacity 2 `
  -SkuName Standard_DS2 `
  -UpgradePolicyMode Automatic

Set-AzVmssStorageProfile $vmssConfig `
  -ImageReferencePublisher MicrosoftWindowsServer `
  -ImageReferenceOffer WindowsServer `
  -ImageReferenceSku 2016-Datacenter `
  -ImageReferenceVersion latest `
  -OsDiskCreateOption FromImage

Set-AzVmssOsProfile $vmssConfig `
  -AdminUsername azureuser `
  -AdminPassword "Azure123456!" `
  -ComputerNamePrefix myvmss

Add-AzVmssNetworkInterfaceConfiguration `
  -VirtualMachineScaleSet $vmssConfig `
  -Name myVmssNetConfig `
  -Primary $true `
  -IPConfiguration $ipConfig

New-AzVmss `
  -ResourceGroupName myResourceGroupAG `
  -Name myvmss `
  -VirtualMachineScaleSet $vmssConfig
```

Install IIS

```
$publicSettings = @{ "fileUris" = (,"https://raw.githubusercontent.com/Azure/azure-docs-powershell-
samples/master/application-gateway/iis/appgatewayurl.ps1");
  "commandToExecute" = "powershell -ExecutionPolicy Unrestricted -File appgatewayurl.ps1" }

$vmss = Get-AzVmss -ResourceGroupName myResourceGroupAG -VMScaleSetName myvmss

Add-AzVmssExtension -VirtualMachineScaleSet $vmss `
  -Name "customScript" `
  -Publisher "Microsoft.Compute" `
  -Type "CustomScriptExtension" `
  -TypeHandlerVersion 1.8 `
  -Setting $publicSettings

Update-AzVmss `
  -ResourceGroupName myResourceGroupAG `
  -Name myvmss `
  -VirtualMachineScaleSet $vmss
```

Create a storage account and configure diagnostics

In this article, the application gateway uses a storage account to store data for detection and prevention purposes. You could also use Azure Monitor logs or Event Hub to record data.

Create the storage account

Create a storage account named *myagstore1* using New-AzStorageAccount.

```
$storageAccount = New-AzStorageAccount `
  -ResourceGroupName myResourceGroupAG `
  -Name myagstore1 `
  -Location eastus `
  -SkuName "Standard_LRS"
```

Configure diagnostics

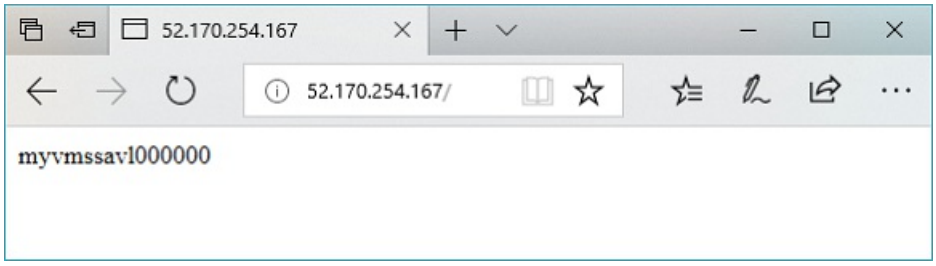
Configure diagnostics to record data into the ApplicationGatewayAccessLog, ApplicationGatewayPerformanceLog, and ApplicationGatewayFirewallLog logs using Set-AzDiagnosticSetting.

```
$appgw = Get-AzApplicationGateway `
  -ResourceGroupName myResourceGroupAG `
  -Name myAppGateway

$store = Get-AzStorageAccount `
  -ResourceGroupName myResourceGroupAG `
  -Name myagstore1

Set-AzDiagnosticSetting `
  -ResourceId $appgw.Id `
  -StorageAccountId $store.Id `
  -Categories ApplicationGatewayAccessLog, ApplicationGatewayPerformanceLog, ApplicationGatewayFirewallLog `
  -Enabled $true `
  -RetentionEnabled $true `
  -RetentionInDays 30
```

Test the application gateway

You can use Get-AzPublicIPAddress to get the public IP address of the application gateway. Copy the public IP address, and then paste it into the address bar of your browser.

```
Get-AzPublicIPAddress -ResourceGroupName myResourceGroupAG -Name myAGPublicIPAddress
```



Clean up resources

When no longer needed, remove the resource group, application gateway, and all related resources using Remove-AzResourceGroup.

```
Remove-AzResourceGroup -Name myResourceGroupAG
```

Next steps

Customize web application firewall rules

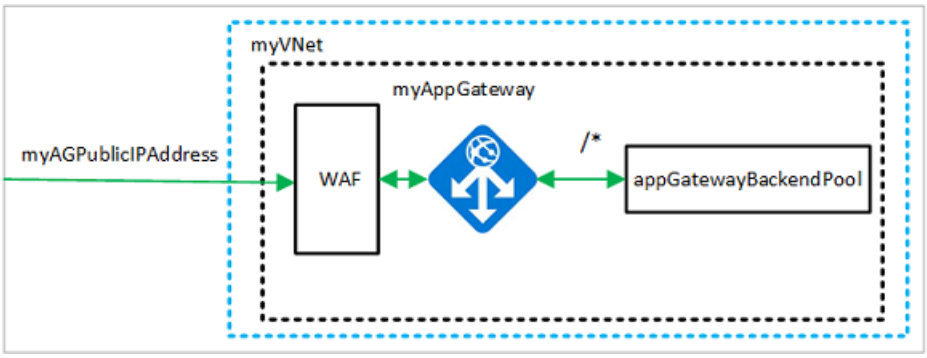
Enable Web Application Firewall using the Azure CLI

11/4/2019 • 5 minutes to read • Edit Online

You can restrict traffic on an application gateway with a Web Application Firewall (WAF). The WAF uses OWASP rules to protect your application. These rules include protection against attacks such as SQL injection, cross-site scripting attacks, and session hijacks.

In this article, you learn how to:

- Set up the network
- Create an application gateway with WAF enabled
- Create a virtual machine scale set
- Create a storage account and configure diagnostics



If you prefer, you can complete this procedure using Azure PowerShell.

If you don't have an Azure subscription, create a free account before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.

2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl**+**Shift**+**V** on Windows and Linux or by selecting **Cmd**+**Shift**+**V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this article requires you to run the Azure CLI version 2.0.4 or later. To find the version, run `az --version`. If you need to install or upgrade, see Install Azure CLI.

Create a resource group

A resource group is a logical container into which Azure resources are deployed and managed. Create an Azure resource group named *myResourceGroupAG* with az group create.

```
az group create --name myResourceGroupAG --location eastus
```

Create network resources

The virtual network and subnets are used to provide network connectivity to the application gateway and its associated resources. Create a virtual network named *myVNet* and a subnet named *myAGSubnet*. then create a public IP address named *myAGPublicIPAddress*.

```
az network vnet create \
  --name myVNet \
  --resource-group myResourceGroupAG \
  --location eastus \
  --address-prefix 10.0.0.0/16 \
  --subnet-name myBackendSubnet \
  --subnet-prefix 10.0.1.0/24

az network vnet subnet create \
  --name myAGSubnet \
  --resource-group myResourceGroupAG \
  --vnet-name myVNet \
  --address-prefix 10.0.2.0/24

az network public-ip create \
  --resource-group myResourceGroupAG \
  --name myAGPublicIPAddress \
  --allocation-method Static \
  --sku Standard
```

Create an application gateway with a WAF

You can use az network application-gateway create to create the application gateway named *myAppGateway*. When you create an application gateway using the Azure CLI, you specify configuration information, such as capacity, sku, and HTTP settings. The application gateway is assigned to *myAGSubnet* and *myAGPublicIPAddress*.

```
az network application-gateway create \
  --name myAppGateway \
  --location eastus \
  --resource-group myResourceGroupAG \
  --vnet-name myVNet \
  --subnet myAGSubnet \
  --capacity 2 \
  --sku WAF_v2 \
  --http-settings-cookie-based-affinity Disabled \
  --frontend-port 80 \
  --http-settings-port 80 \
  --http-settings-protocol Http \
  --public-ip-address myAGPublicIPAddress

az network application-gateway waf-config set \
  --enabled true \
  --gateway-name myAppGateway \
  --resource-group myResourceGroupAG \
  --firewall-mode Detection \
  --rule-set-version 3.0
```

It may take several minutes for the application gateway to be created. After the application gateway is created, you can see these new features of it:

- *appGatewayBackendPool* - An application gateway must have at least one backend address pool.
- *appGatewayBackendHttpSettings* - Specifies that port 80 and an HTTP protocol is used for communication.
- *appGatewayHttpListener* - The default listener associated with *appGatewayBackendPool*.
- *appGatewayFrontendIP* - Assigns *myAGPublicIPAddress* to *appGatewayHttpListener*.
- *rule1* - The default routing rule that is associated with *appGatewayHttpListener*.

Create a virtual machine scale set

In this example, you create a virtual machine scale set that provides two servers for the backend pool in the application gateway. The virtual machines in the scale set are associated with the *myBackendSubnet* subnet. To create the scale set, you can use az vmss create.

```
az vmss create \
  --name myvmss \
  --resource-group myResourceGroupAG \
  --image UbuntuLTS \
  --admin-username azureuser \
  --admin-password Azure123456! \
  --instance-count 2 \
  --vnet-name myVNet \
  --subnet myBackendSubnet \
  --vm-sku Standard_DS2 \
  --upgrade-policy-mode Automatic \
  --app-gateway myAppGateway \
  --backend-pool-name appGatewayBackendPool
```

Install NGINX

```
az vmss extension set \
  --publisher Microsoft.Azure.Extensions \
  --version 2.0 \
  --name CustomScript \
  --resource-group myResourceGroupAG \
  --vmss-name myvmss \
  --settings '{ "fileUris": ["https://raw.githubusercontent.com/Azure/azure-docs-powershell-
samples/master/application-gateway/iis/install_nginx.sh"],"commandToExecute": "./install_nginx.sh" }'
```

Create a storage account and configure diagnostics

In this article, the application gateway uses a storage account to store data for detection and prevention purposes. You could also use Azure Monitor logs or Event Hub to record data.

Create a storage account

Create a storage account named *myagstore1* with az storage account create.

```
az storage account create \
  --name myagstore1 \
  --resource-group myResourceGroupAG \
  --location eastus \
  --sku Standard_LRS \
  --encryption blob
```

Configure diagnostics

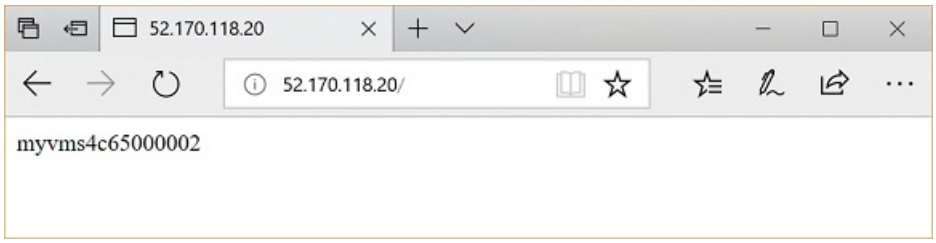
Configure diagnostics to record data into the ApplicationGatewayAccessLog, ApplicationGatewayPerformanceLog, and ApplicationGatewayFirewallLog logs. Replace `<subscriptionId>` with your subscription identifier and then configure diagnostics with az monitor diagnostic-settings create.

```
appgwid=$(az network application-gateway show --name myAppGateway --resource-group myResourceGroupAG --query
id -o tsv)

storeid=$(az storage account show --name myagstore1 --resource-group myResourceGroupAG --query id -o tsv)

az monitor diagnostic-settings create --name appgwdiag --resource $appgwid \
  --logs '[ { "category": "ApplicationGatewayAccessLog", "enabled": true, "retentionPolicy": { "days": 30,
"enabled": true } }, { "category": "ApplicationGatewayPerformanceLog", "enabled": true, "retentionPolicy": {
"days": 30, "enabled": true } }, { "category": "ApplicationGatewayFirewallLog", "enabled": true,
"retentionPolicy": { "days": 30, "enabled": true } } ]' \
  --storage-account $storeid
```

Test the application gateway

To get the public IP address of the application gateway, use az network public-ip show. Copy the public IP address, and then paste it into the address bar of your browser.

```
az network public-ip show \
  --resource-group myResourceGroupAG \
  --name myAGPublicIPAddress \
  --query [ipAddress] \
  --output tsv
```



Clean up resources

When no longer needed, remove the resource group, application gateway, and all related resources.

```
az group delete --name myResourceGroupAG
```

Next steps

Customize web application firewall rules

Configure per-site WAF policies using Azure PowerShell

2/18/2020 • 9 minutes to read • Edit Online

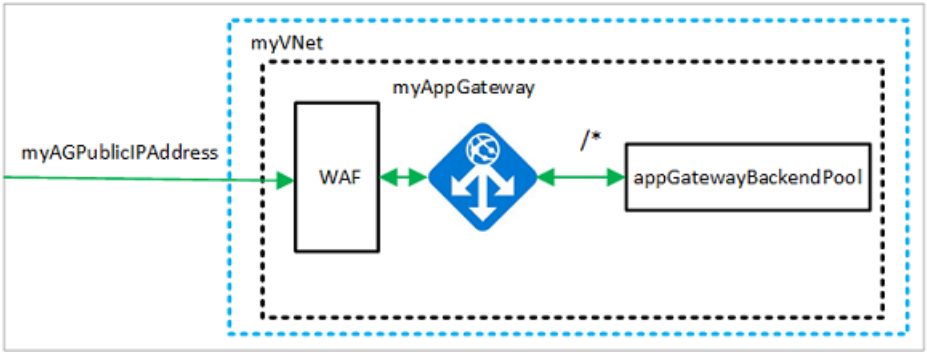
Web Application Firewall (WAF) settings are contained in WAF policies, and to change your WAF configuration you modify the WAF policy.

When associated with your Application Gateway, the policies and all the settings are reflected globally. So, if you have five sites behind your WAF, all five sites are protected by the same WAF Policy. This is great if you need the same security settings for every site. But you can also apply WAF policies to individual listeners to allow for site-specific WAF configuration.

By applying WAF policies to a listener, you can configure WAF settings for individual sites without the changes affecting every site. The most specific policy takes precedent. If there's a global policy, and a per-site policy (a WAF policy associated with a listener), then the per-site policy overrides the global WAF policy for that listener. Other listeners without their own policies will only be affected by the global WAF policy.

In this article, you learn how to:

- Set up the network
- Create a WAF policy
- Create an application gateway with WAF enabled
- Apply the WAF policy globally, per-site, and per-URI
- Create a virtual machine scale set
- Create a storage account and configure diagnostics
- Test the application gateway



If you don't have an Azure subscription, create a free account before you begin.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can

use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl**+**Shift**+**V** on Windows and Linux or by selecting **Cmd**+**Shift**+**V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the PowerShell locally, this article requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the version. If you need to upgrade, see Install Azure PowerShell module. If you're running PowerShell locally, you also need to run `Login-AzAccount` to create a connection with Azure.

Create a resource group

A resource group is a logical container into which Azure resources are deployed and managed. Create an Azure resource group using New-AzResourceGroup.

```
$rgname = New-AzResourceGroup -Name myResourceGroupAG -Location eastus
```

Create network resources

Create the subnet configurations named *myBackendSubnet* and *myAGSubnet* using New-AzVirtualNetworkSubnetConfig. Create the virtual network named *myVNet* using New-AzVirtualNetwork with the subnet configurations. And finally, create the public IP address named *myAGPublicIPAddress* using New-AzPublicIpAddress. These resources are used to provide network connectivity to the application gateway and its associated resources.

```
$backendSubnetConfig = New-AzVirtualNetworkSubnetConfig `
  -Name myBackendSubnet `
  -AddressPrefix 10.0.1.0/24

$agSubnetConfig = New-AzVirtualNetworkSubnetConfig `
  -Name myAGSubnet `
  -AddressPrefix 10.0.2.0/24

$vnet = New-AzVirtualNetwork `
  -ResourceGroupName myResourceGroupAG `
  -Location eastus `
  -Name myVNet `
  -AddressPrefix 10.0.0.0/16 `
  -Subnet $backendSubnetConfig, $agSubnetConfig

$pip = New-AzPublicIpAddress `
  -ResourceGroupName myResourceGroupAG `
  -Location eastus `
  -Name myAGPublicIPAddress `
  -AllocationMethod Static `
  -Sku Standard
```

Create an application gateway

In this section, you create resources that support the application gateway, and then finally create it and a WAF. The resources that you create include:

- *IP configurations and frontend port* - Associates the subnet that you previously created to the application gateway and assigns a port to use to access it.
- *Default pool* - All application gateways must have at least one backend pool of servers.
- *Default listener and rule* - The default listener listens for traffic on the port that was assigned and the default rule sends traffic to the default pool.

Create the IP configurations and frontend port

Associate *myAGSubnet* that you previously created to the application gateway using New-AzApplicationGatewayIPConfiguration. Assign *myAGPublicIPAddress* to the application gateway using New-AzApplicationGatewayFrontendIPConfig.

```
$vnet = Get-AzVirtualNetwork `
  -ResourceGroupName myResourceGroupAG `
  -Name myVNet

$subnet=$vnet.Subnets[1]

$gipconfig = New-AzApplicationGatewayIPConfiguration `
  -Name myAGIPConfig `
  -Subnet $subnet

$fipconfig = New-AzApplicationGatewayFrontendIPConfig `
  -Name myAGFrontendIPConfig `
  -PublicIPAddress $pip

$frontendport80 = New-AzApplicationGatewayFrontendPort `
  -Name myFrontendPort `
  -Port 80

$frontendport8080 = New-AzApplicationGatewayFrontendPort `
  -Name myFrontendPort `
  -Port 8080
```

Create the backend pool and settings

Create the backend pool named *appGatewayBackendPool* for the application gateway using New-AzApplicationGatewayBackendAddressPool. Configure the settings for the backend address pools using New-AzApplicationGatewayBackendHttpSettings.

```
$defaultPool = New-AzApplicationGatewayBackendAddressPool `
  -Name appGatewayBackendPool

$poolSettings = New-AzApplicationGatewayBackendHttpSettings `
  -Name myPoolSettings `
  -Port 80 `
  -Protocol Http `
  -CookieBasedAffinity Enabled `
  -RequestTimeout 120
```

Create two WAF policies

Create two WAF policies, one global and one per-site, and add custom rules.

The per-site policy restricts the file upload limit to 5 MB. Everything else is the same.

```
$variable = New-AzApplicationGatewayFirewallMatchVariable -VariableName RequestUri
$condition = New-AzApplicationGatewayFirewallCondition -MatchVariable $variable -Operator Contains -MatchValue
"globalAllow"
$rule = New-AzApplicationGatewayFirewallCustomRule -Name globalAllow -Priority 5 -RuleType MatchRule -
MatchCondition $condition -Action Allow

$variable1 = New-AzApplicationGatewayFirewallMatchVariable -VariableName RequestUri
$condition1 = New-AzApplicationGatewayFirewallCondition -MatchVariable $variable1 -Operator Contains -
MatchValue "globalBlock"
$rule1 = New-AzApplicationGatewayFirewallCustomRule -Name globalAllow -Priority 10 -RuleType MatchRule -
MatchCondition $condition1 -Action Block

$variable2 = New-AzApplicationGatewayFirewallMatchVariable -VariableName RequestUri
$condition2 = New-AzApplicationGatewayFirewallCondition -MatchVariable $variable2 -Operator Contains -
MatchValue "siteAllow"
$rule2 = New-AzApplicationGatewayFirewallCustomRule -Name globalAllow -Priority 5 -RuleType MatchRule -
MatchCondition $condition2 -Action Allow

$variable3 = New-AzApplicationGatewayFirewallMatchVariable -VariableName RequestUri
$condition3 = New-AzApplicationGatewayFirewallCondition -MatchVariable $variable3 -Operator Contains -
MatchValue "siteBlock"
$rule3 = New-AzApplicationGatewayFirewallCustomRule -Name globalAllow -Priority 10 -RuleType MatchRule -
MatchCondition $condition3 -Action Block

$variable4 = New-AzApplicationGatewayFirewallMatchVariable -VariableName RequestUri
$condition4 = New-AzApplicationGatewayFirewallCondition -MatchVariable $variable4 -Operator Contains -
MatchValue "URIAllow"
$rule4 = New-AzApplicationGatewayFirewallCustomRule -Name globalAllow -Priority 5 -RuleType MatchRule -
MatchCondition $condition4 -Action Allow

$variable5 = New-AzApplicationGatewayFirewallMatchVariable -VariableName RequestUri
$condition5 = New-AzApplicationGatewayFirewallCondition -MatchVariable $variable5 -Operator Contains -
MatchValue "URIBlock"
$rule5 = New-AzApplicationGatewayFirewallCustomRule -Name globalAllow -Priority 10 -RuleType MatchRule -
MatchCondition $condition5 -Action Block

$policySettingGlobal = New-AzApplicationGatewayFirewallPolicySetting `
  -Mode Prevention `
  -State Enabled `
  -MaxRequestBodySizeInKb 100 `
  -MaxFileUploadInMb 256

$wafPolicyGlobal = New-AzApplicationGatewayFirewallPolicy `
  -Name wafpolicyGlobal `
  -ResourceGroup myResourceGroupAG `
  -Location eastus `
  -PolicySetting $PolicySettingGlobal `
  -CustomRule $rule, $rule1

$policySettingSite = New-AzApplicationGatewayFirewallPolicySetting `
  -Mode Prevention `
  -State Enabled `
  -MaxRequestBodySizeInKb 100 `
  -MaxFileUploadInMb 5

$wafPolicySite = New-AzApplicationGatewayFirewallPolicy `
  -Name wafpolicySite `
  -ResourceGroup myResourceGroupAG `
  -Location eastus `
  -PolicySetting $PolicySettingSite `
  -CustomRule $rule2, $rule3
```

Create the default listener and rule

A listener is required to enable the application gateway to route traffic appropriately to the backend address pools. In this example, you create a basic listener that listens for traffic at the root URL.

Create a listener named *mydefaultListener* using New-AzApplicationGatewayHttpListener with the frontend configuration and frontend port that you previously created. A rule is required for the listener to know which backend pool to use for incoming traffic. Create a basic rule named *rule1* using New-AzApplicationGatewayRequestRoutingRule.

```
$globalListener = New-AzApplicationGatewayHttpListener `
  -Name mydefaultListener `
  -Protocol Http `
  -FrontendIPConfiguration $fipconfig `
  -FrontendPort $frontendport80

$frontendRule = New-AzApplicationGatewayRequestRoutingRule `
  -Name rule1 `
  -RuleType Basic `
  -HttpListener $globallistener `
  -BackendAddressPool $defaultPool `
  -BackendHttpSettings $poolSettings

$siteListener = New-AzApplicationGatewayHttpListener `
  -Name mydefaultListener `
  -Protocol Http `
  -FrontendIPConfiguration $fipconfig `
  -FrontendPort $frontendport8080 `
  -FirewallPolicy $wafPolicySite

$frontendRuleSite = New-AzApplicationGatewayRequestRoutingRule `
  -Name rule2 `
  -RuleType Basic `
  -HttpListener $siteListener `
  -BackendAddressPool $defaultPool `
  -BackendHttpSettings $poolSettings
```

Create the application gateway with the WAF

Now that you created the necessary supporting resources, specify parameters for the application gateway using New-AzApplicationGatewaySku. Specify the Firewall Policy using New-AzApplicationGatewayFirewallPolicy. And then create the application gateway named *myAppGateway* using New-AzApplicationGateway.

```
$sku = New-AzApplicationGatewaySku `
  -Name WAF_v2 `
  -Tier WAF_v2 `
  -Capacity 2

$appgw = New-AzApplicationGateway `
  -Name myAppGateway `
  -ResourceGroupName myResourceGroupAG `
  -Location eastus `
  -BackendAddressPools $defaultPool `
  -BackendHttpSettingsCollection $poolSettings `
  -FrontendIpConfigurations $fipconfig `
  -GatewayIpConfigurations $gipconfig `
  -FrontendPorts $frontendport80 `
  -HttpListeners $globallistener `
  -RequestRoutingRules $frontendRule `
  -Sku $sku `
  -FirewallPolicy $wafPolicyGlobal
```

Apply a per-URI policy

To apply a per-URI policy, simply create a new policy and apply it to the path rule config.

```
$policySettingURI = New-AzApplicationGatewayFirewallPolicySetting `
  -Mode Prevention `
  -State Enabled `
  -MaxRequestBodySizeInKb 100 `
  -MaxFileUploadInMb 5

$wafPolicyURI = New-AzApplicationGatewayFirewallPolicy `
  -Name wafpolicySite `
  -ResourceGroup myResourceGroupAG `
  -Location eastus `
  -PolicySetting $PolicySettingURI `
  -CustomRule $rule4, $rule5

$Gateway = Get-AzApplicationGateway -Name "myAppGateway"

$PathRuleConfig = New-AzApplicationGatewayPathRuleConfig -Name "base" `
  -Paths "/base" `
  -BackendAddressPool $defaultPool `
  -BackendHttpSettings $poolSettings `
  -FirewallPolicy $wafPolicyURI

$PathRuleConfig1 = New-AzApplicationGatewayPathRuleConfig `
  -Name "base" -Paths "/test" `
  -BackendAddressPool $defaultPool `
  -BackendHttpSettings $poolSettings

$URLPathMap = New-AzApplicationGatewayUrlPathMapConfig -Name "PathMap" `
  -PathRules $PathRuleConfig, $PathRuleConfig1 `
  -DefaultBackendAddressPoolId $defaultPool.Id `
  -DefaultBackendHttpSettingsId poolSettings.Id

Add-AzApplicationGatewayRequestRoutingRule -ApplicationGateway $AppGw `
  -Name "RequestRoutingRule" `
  -RuleType PathBasedRouting `
  -HttpListener $siteListener `
  -UrlPathMapId $URLPathMap.Id
```

Create a virtual machine scale set

In this example, you create a virtual machine scale set to provide servers for the backend pool in the application gateway. You assign the scale set to the backend pool when you configure the IP settings.

```
$vnet = Get-AzVirtualNetwork `
  -ResourceGroupName myResourceGroupAG `
  -Name myVNet

$appgw = Get-AzApplicationGateway `
  -ResourceGroupName myResourceGroupAG `
  -Name myAppGateway

$backendPool = Get-AzApplicationGatewayBackendAddressPool `
  -Name defaultPool `
  -ApplicationGateway $appgw

$ipConfig = New-AzVmssIpConfig `
  -Name myVmssIPConfig `
  -SubnetId $vnet.Subnets[1].Id `
  -ApplicationGatewayBackendAddressPoolsId $backendPool.Id

$vmssConfig = New-AzVmssConfig `
  -Location eastus `
  -SkuCapacity 2 `
  -SkuName Standard_DS2 `
  -UpgradePolicyMode Automatic

Set-AzVmssStorageProfile $vmssConfig `
  -ImageReferencePublisher MicrosoftWindowsServer `
  -ImageReferenceOffer WindowsServer `
  -ImageReferenceSku 2016-Datacenter `
  -ImageReferenceVersion latest `
  -OsDiskCreateOption FromImage

Set-AzVmssOsProfile $vmssConfig `
  -AdminUsername azureuser `
  -AdminPassword "Azure123456!" `
  -ComputerNamePrefix myvmss

Add-AzVmssNetworkInterfaceConfiguration `
  -VirtualMachineScaleSet $vmssConfig `
  -Name myVmssNetConfig `
  -Primary $true `
  -IPConfiguration $ipConfig

New-AzVmss `
  -ResourceGroupName myResourceGroupAG `
  -Name myvmss `
  -VirtualMachineScaleSet $vmssConfig
```

Install IIS

```
$publicSettings = @{ "fileUris" = (,"https://raw.githubusercontent.com/Azure/azure-docs-powershell-
samples/master/application-gateway/iis/appgatewayurl.ps1");
  "commandToExecute" = "powershell -ExecutionPolicy Unrestricted -File appgatewayurl.ps1" }

$vmss = Get-AzVmss -ResourceGroupName myResourceGroupAG -VMScaleSetName myvmss

Add-AzVmssExtension -VirtualMachineScaleSet $vmss `
  -Name "customScript" `
  -Publisher "Microsoft.Compute" `
  -Type "CustomScriptExtension" `
  -TypeHandlerVersion 1.8 `
  -Setting $publicSettings

Update-AzVmss `
  -ResourceGroupName myResourceGroupAG `
  -Name myvmss `
  -VirtualMachineScaleSet $vmss
```

Create a storage account and configure diagnostics

In this article, the application gateway uses a storage account to store data for detection and prevention purposes. You could also use Azure Monitor logs or Event Hub to record data.

Create the storage account

Create a storage account named *myagstore1* using New-AzStorageAccount.

```
$storageAccount = New-AzStorageAccount `
  -ResourceGroupName myResourceGroupAG `
  -Name myagstore1 `
  -Location eastus `
  -SkuName "Standard_LRS"
```

Configure diagnostics

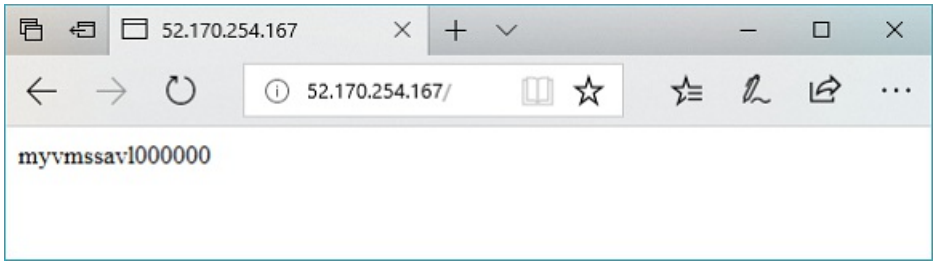
Configure diagnostics to record data into the ApplicationGatewayAccessLog, ApplicationGatewayPerformanceLog, and ApplicationGatewayFirewallLog logs using Set-AzDiagnosticSetting.

```
$appgw = Get-AzApplicationGateway `
  -ResourceGroupName myResourceGroupAG `
  -Name myAppGateway

$store = Get-AzStorageAccount `
  -ResourceGroupName myResourceGroupAG `
  -Name myagstore1

Set-AzDiagnosticSetting `
  -ResourceId $appgw.Id `
  -StorageAccountId $store.Id `
  -Categories ApplicationGatewayAccessLog, ApplicationGatewayPerformanceLog, ApplicationGatewayFirewallLog `
  -Enabled $true `
  -RetentionEnabled $true `
  -RetentionInDays 30
```

Test the application gateway

You can use Get-AzPublicIPAddress to get the public IP address of the application gateway. Then use this IP address to curl against (replace the 1.1.1.1 shown below).

```
Get-AzPublicIPAddress -ResourceGroupName myResourceGroupAG -Name myAGPublicIPAddress

#should be blocked
curl 1.1.1.1/globalBlock
curl 1.1.1.1/?1=1

#should be allowed
curl 1.1.1.1/globalAllow?1=1

#should be blocked
curl 1.1.1.1:8080/siteBlock
curl 1.1.1.1/?1=1

#should be allowed
curl 1.1.1.1:8080/siteAllow?1=1

#should be blocked
curl 1.1.1.1/URIBlock
curl 1.1.1.1/?1=1

#should be allowed
curl 1.1.1.1/URIAllow?1=1
```



Clean up resources

When no longer needed, remove the resource group, application gateway, and all related resources using Remove-AzResourceGroup.

```
Remove-AzResourceGroup -Name myResourceGroupAG
```

Next steps

Customize web application firewall rules

Migrate Web Application Firewall policies using Azure PowerShell

11/18/2019 • 4 minutes to read • Edit Online

This script makes it easy to transition from a WAF config or a custom rules-only WAF policy to a full WAF policy. You may see a warning in the portal that says *migrate to WAF policy*, or you may want the new public preview WAF features such as Geomatch custom rules, per-site and per-URI WAF policy, or the bot mitigation ruleset. To use any of these features, you need a full WAF policy associated to your application gateway.

For more information about creating a new WAF policy, see Create Web Application Firewall policies for Application Gateway. For information about migrating, see Migrate to WAF policy.

To migrate to WAF policy using the migration script

Use the following steps to run the migration script:

1. Open the following cloud shell window, or open one from within the portal.
2. Copy the script into the cloud shell window and run it.
3. The script asks for Subscription ID, Resource Group name, the name of the Application Gateway that the WAF config is associated with, and the name of the new WAF policy that to create. Once you enter these inputs, the script runs and creates your new WAF policy
4. Associate the new WAF policy with your application gateway. Go to the WAF policy in the portal and select the **Associated Application Gateways** tab. Select **Associate an Application Gateway** and then select the Application Gateway to associate the WAF policy to.

```
<#PSScriptInfo
.DESCRIPTION
Will be used to migrate to the application-gateway to a top level waf policy experience.

.VERSION 1.0

.GUID b6fedd43-ebd0-41ed-9847-4f1c1c43be22

.AUTHOR Venkat.Krishnan

.PARAMETER subscriptionId
Subscription Id of where the resources are present.
.PARAMETER resourceGroupName
Resource-group where the resources are present.
.PARAMETER applicationGatewayName
Application-Gateway name
.PARAMETER wafPolicyName
Name of the web application firewall policy

.EXAMPLE
./migrateToWafPolicy.ps1 -subscriptionId  <your-subscription-id> -applicationGatewayName <your-appgw-name> -
resourceGroupName <your-resource-group-name> -wafPolicyName <new-waf-policy-name>
#>

param(
 [Parameter(Mandatory=$true)]
 [string] $subscriptionId,
 [Parameter(Mandatory=$true)]
 [string] $resourceGroupName,
 [Parameter(Mandatory=$true)]
 [string] $applicationGatewayName,
```

```
 [string] $applicationGatewayName,
 [Parameter(Mandatory=$true)]
 [string] $wafPolicyName
)

function ValidateInput ($appgwName, $resourceGroupName) {
 # Obtain the application-gateway
 $appgw = Get-AzApplicationGateway -Name $applicationGatewayName -ResourceGroupName $resourceGroupName
 if (-not $appgw) {
  Write-Error "ApplicationGateway: $applicationGatewayName is not present in ResourceGroup:
$resourceGroupName"
  return $false
 }

 # Check if already have a global firewall policy
 if ($appgw.FirewallPolicy) {
  $fp = Get-AzResource -ResourceId $appgw.FirewallPolicy.Id
  if ($fp.PolicySettings) {
   Write-Error "ApplicationGateway: $applicationGatewayName already has a global firewall policy: $fp.Name.
Please use portal for changing the policy."
   return $false
  }
 }

 if ($appgw.WebApplicationFirewallConfiguration) {
  # Throw an error, since ruleGroup disabled case can't be migrated now.
  if ($appgw.WebApplicationFirewallConfiguration.DisabledRuleGroups) {
   foreach ($disabled in $appgw.WebApplicationFirewallConfiguration.DisabledRuleGroups) {
    if ($disabled.Rules.Count -eq 0) {
     $ruleGroupName = $disabled.RuleGroupName
                    Write-Error "The ruleGroup '$ruleGroupName' is disabled. Currently we can't migrate to a
firewall policy when an entire ruleGroup is disabled. This feature will be delivered shortly. To continue,
kindly ensure the entire rulegroups are not disabled. "
                    return $false
    }
   }
  }

  # Throw an error when exclusion entry with 'EqualsAny' operator is present
  if ($appgw.WebApplicationFirewallConfiguration.Exclusions) {
   foreach ($excl in $appgw.WebApplicationFirewallConfiguration.Exclusions) {
    if ($null -ne $excl.MatchVariable -and $null -eq $excl.SelectorMatchOperator -and $null -eq
$excl.Selector) {
     Write-Error " You have an exclusion entry(s) with the 'Equals any' operator. Currently we can't migrate
to a firewall policy with 'Equals Any' operator. This feature will be delivered shortly. To continue, kindly
ensure exclusion entries with 'Equals Any' operator is not present. "
     return $false
    }
   }
  }
 }

 if ($appgw.Sku.Name -ne "WAF_v2" -or $appgw.Sku.Tier -ne "WAF_v2") {
  Write-Error " Cannot associate a firewall policy to application gateway :$applicationGatewayName since the
Sku is not on WAF_v2"
  return $false
 }

 return $true
}

function Login() {
 $context = Get-AzContext
 if ($null -eq $context -or $null -eq $context.Account) {
  Login-AzAccount
 }
}

function createNewTopLevelWafPolicy ($subscriptionId, $resourceGroupName, $applicationGatewayName,
$wafPolicyName) {
```

```
$wafPolicyName) {
 Select-AzSubscription -Subscription $subscriptionId
 $retVal = ValidateInput -appgwName $applicationGatewayName -resourceGroupName $resourceGroupName
 if (!$retVal) {
  return
 }

 $appgw = Get-AzApplicationGateway -Name $applicationGatewayName -ResourceGroupName $resourceGroupName

 # Get the managedRule and PolicySettings
 $managedRule = New-AzApplicationGatewayFirewallPolicyManagedRule
 $policySetting = New-AzApplicationGatewayFirewallPolicySetting
    if ($appgw.WebApplicationFirewallConfiguration) {
  $ruleGroupOverrides = [System.Collections.ArrayList]@()
  if ($appgw.WebApplicationFirewallConfiguration.DisabledRuleGroups) {
   foreach ($disabled in $appgw.WebApplicationFirewallConfiguration.DisabledRuleGroups) {
    $rules = [System.Collections.ArrayList]@()
    if ($disabled.Rules.Count -gt 0) {
     foreach ($rule in $disabled.Rules) {
      $ruleOverride = New-AzApplicationGatewayFirewallPolicyManagedRuleOverride -RuleId $rule
      $_ = $rules.Add($ruleOverride)
     }
    }

    $ruleGroupOverride = New-AzApplicationGatewayFirewallPolicyManagedRuleGroupOverride -RuleGroupName
$disabled.RuleGroupName -Rule $rules
    $_ = $ruleGroupOverrides.Add($ruleGroupOverride)
   }
  }

  $managedRuleSet = New-AzApplicationGatewayFirewallPolicyManagedRuleSet -RuleSetType
$appgw.WebApplicationFirewallConfiguration.RuleSetType -RuleSetVersion
$appgw.WebApplicationFirewallConfiguration.RuleSetVersion
        if ($ruleGroupOverrides.Count -ne 0) {
   $managedRuleSet = New-AzApplicationGatewayFirewallPolicyManagedRuleSet -RuleSetType
$appgw.WebApplicationFirewallConfiguration.RuleSetType -RuleSetVersion
$appgw.WebApplicationFirewallConfiguration.RuleSetVersion -RuleGroupOverride $ruleGroupOverrides
  }

  $exclusions = [System.Collections.ArrayList]@()
  if ($appgw.WebApplicationFirewallConfiguration.Exclusions) {
   foreach ($excl in $appgw.WebApplicationFirewallConfiguration.Exclusions) {
               if ($excl.MatchVariable -and $excl.SelectorMatchOperator -and $excl.Selector) {
        $exclusionEntry = New-AzApplicationGatewayFirewallPolicyExclusion -MatchVariable  $excl.MatchVariable
-SelectorMatchOperator $excl.SelectorMatchOperator -Selector $excl.Selector
                    $_ = $exclusions.Add($exclusionEntry)
               }
   }
  }

  $managedRule = New-AzApplicationGatewayFirewallPolicyManagedRule -ManagedRuleSet $managedRuleSet
  $exclCount = $exclusions.Count
        if ($exclCount -ne 0) {
   $managedRule = New-AzApplicationGatewayFirewallPolicyManagedRule -ManagedRuleSet $managedRuleSet -Exclusion
$exclusions
  }


  $policySetting = New-AzApplicationGatewayFirewallPolicySetting -MaxFileUploadInMb
$appgw.WebApplicationFirewallConfiguration.FileUploadLimitInMb -MaxRequestBodySizeInKb
$appgw.WebApplicationFirewallConfiguration.MaxRequestBodySizeInKb -Mode Detection -State Disabled
  if ($appgw.WebApplicationFirewallConfiguration.FirewallMode -eq "Prevention") {
   $policySetting.Mode = "Prevention"
  }

  if ($appgw.WebApplicationFirewallConfiguration.Enabled) {
   $policySetting.State = "Enabled"
  }
```

```
  $policySetting.RequestBodyCheck = $appgw.WebApplicationFirewallConfiguration.RequestBodyCheck;
 }

 if ($appgw.FirewallPolicy) {
  $customRulePolicyId = $appgw.FirewallPolicy.Id
  $rg = Get-AzResourceGroup -Id $customRulePolicyId
  $crPolicyName = $customRulePolicyId.Substring($customRulePolicyId.LastIndexOf("/") + 1)
  $customRulePolicy = Get-AzApplicationGatewayFirewallPolicy -ResourceGroupName $rg.ResourceGroupName -Name
$crPolicyName
  $wafPolicy = New-AzApplicationGatewayFirewallPolicy -ResourceGroupName $rg.ResourceGroupName -Name
$wafPolicyName -CustomRule $customRulePolicy.CustomRules -ManagedRule $managedRule -PolicySetting
$policySetting -Location $appgw.Location
 } else {
  $wafPolicy = New-AzApplicationGatewayFirewallPolicy -Name $wafPolicyName -ResourceGroupName
$resourceGroupName -PolicySetting $policySetting -ManagedRule $managedRule -Location $appgw.Location
 }

 if (!$wafPolicy) {
  return
 }

    $appgw.FirewallPolicy = $wafPolicy
 $appgw = Set-AzApplicationGateway -ApplicationGateway $appgw
 Write-Host " firewallPolicy: $wafPolicyName has been created/updated successfully and applied to
applicationGateway: $applicationGatewayName!"
    return $wafPolicy
}

function Main() {
 Login
 $policy = createNewTopLevelWafPolicy -subscriptionId $subscriptionId -resourceGroupName $resourceGroupName -
applicationGatewayName $applicationGatewayName -wafPolicyName $wafPolicyName
 return $policy
}

Main
```

Next steps

Learn more about Web Application Firewall CRS rule groups and rules.

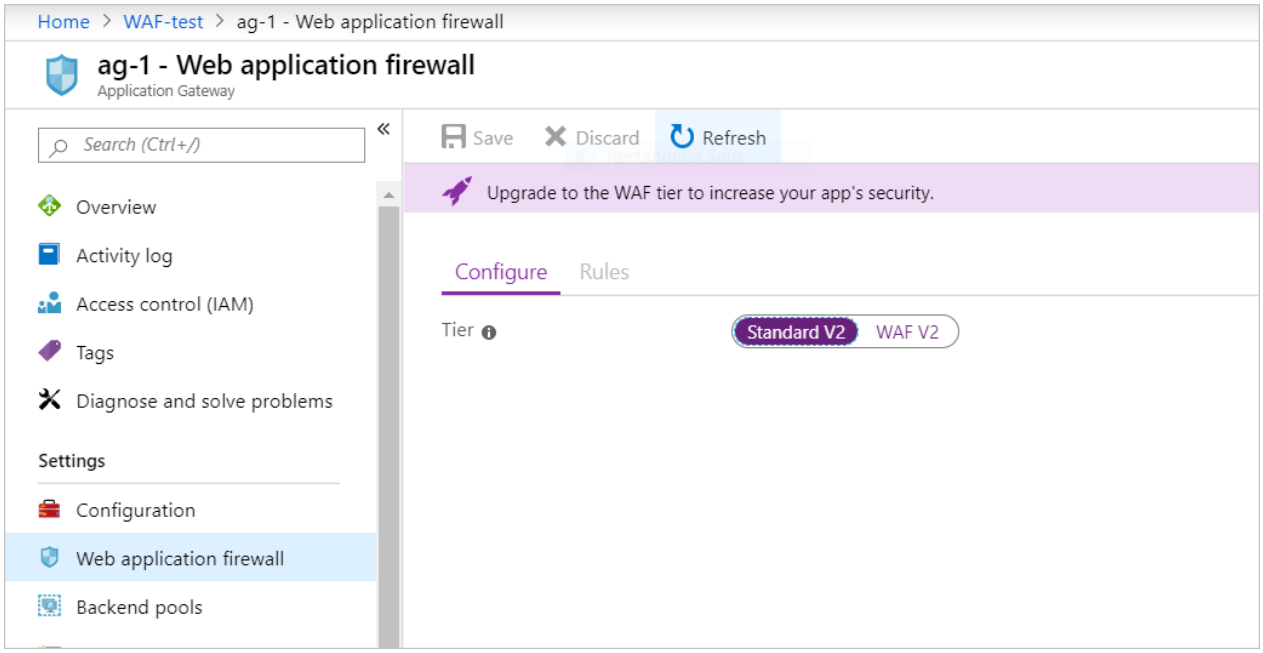
Customize Web Application Firewall rules using the Azure portal

11/13/2019 • 2 minutes to read • Edit Online

The Azure Application Gateway Web Application Firewall (WAF) provides protection for web applications. These protections are provided by the Open Web Application Security Project (OWASP) Core Rule Set (CRS). Some rules can cause false positives and block real traffic. For this reason, Application Gateway provides the capability to customize rule groups and rules. For more information on the specific rule groups and rules, see List of Web Application Firewall CRS rule groups and rules.

NOTE

If your application gateway is not using the WAF tier, the option to upgrade the application gateway to the WAF tier appears in the right pane.



View rule groups and rules

To view rule groups and rules

1. Browse to the application gateway, and then select **Web application firewall**.
2. Select your **WAF Policy**.
3. Select **Managed Rules**.

   This view shows a table on the page of all the rule groups provided with the chosen rule set. All of the rule's check boxes are selected.

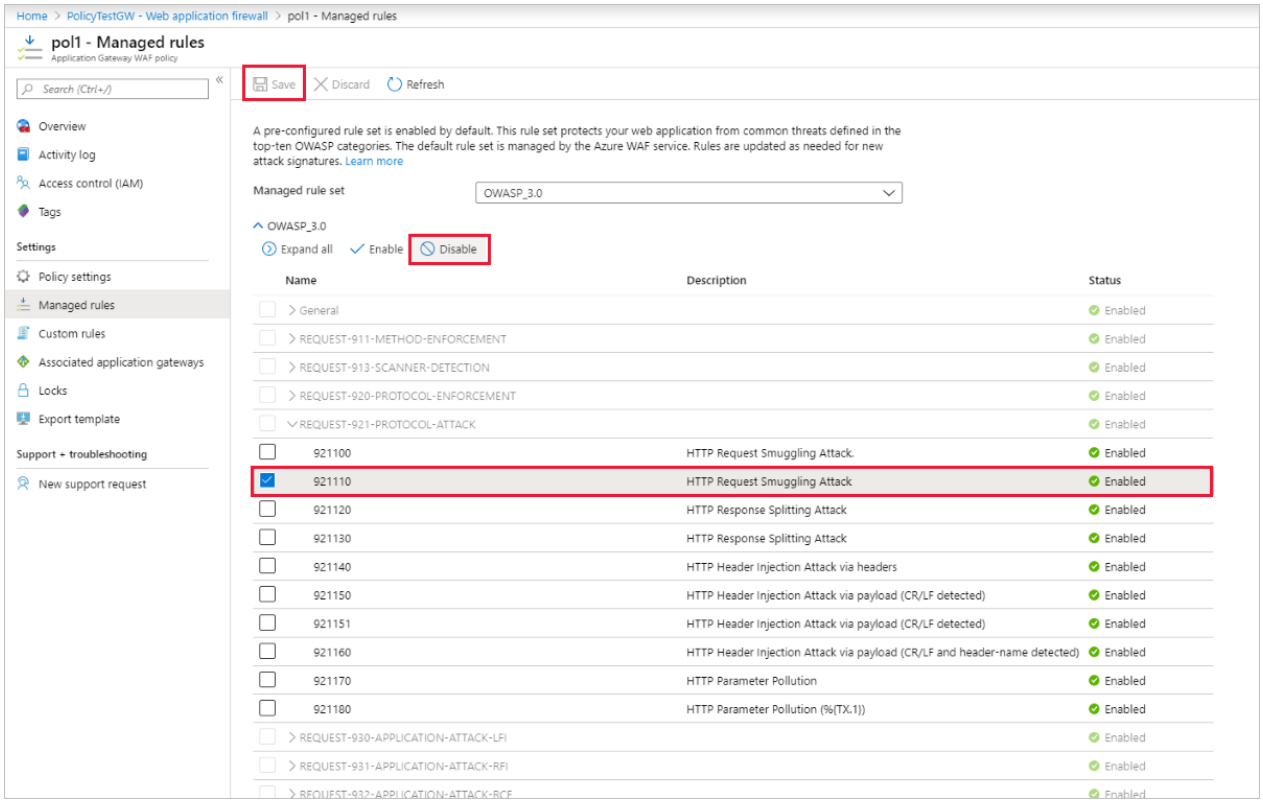
Disable rule groups and rules

> **IMPORTANT**
> Use caution when disabling any rule groups or rules. This may expose you to increased security risks.

When you're disabling rules, you can disable an entire rule group or specific rules under one or more rule groups.

To disable rule groups or specific rules

1. Search for the rules or rule groups that you want to disable.
2. Select the check boxes for the rules that you want to disable.
3. Select the action at the top of the page (enable/disable) for the selected rules.
4. Select **Save**.



Mandatory rules

The following list contains conditions that cause the WAF to block the request while in Prevention Mode. In Detection Mode, they're logged as exceptions.

These can't be configured or disabled:

- Failure to parse the request body results in the request being blocked, unless body inspection is turned off (XML, JSON, form data)
- Request body (with no files) data length is larger than the configured limit
- Request body (including files) is larger than the limit
- An internal error happened in the WAF engine

CRS 3.x specific:

- Inbound anomaly score exceeded threshold

Next steps

After you configure your disabled rules, you can learn how to view your WAF logs. For more information, see

Application Gateway diagnostics.

Customize Web Application Firewall rules using PowerShell

11/13/2019 • 2 minutes to read • Edit Online

The Azure Application Gateway Web Application Firewall (WAF) provides protection for web applications. These protections are provided by the Open Web Application Security Project (OWASP) Core Rule Set (CRS). Some rules can cause false positives and block real traffic. For this reason, Application Gateway provides the capability to customize rule groups and rules. For more information on the specific rule groups and rules, see List of Web Application Firewall CRS Rule groups and rules.

View rule groups and rules

The following code examples show how to view rules and rule groups that are configurable on a WAF-enabled application gateway.

View rule groups

The following example shows how to view rule groups:

```
Get-AzApplicationGatewayAvailableWafRuleSets
```

The following output is a truncated response from the preceding example:

```
OWASP (Ver. 3.0):

    General:
        Description:

        Rules:
            RuleId     Description
            ------     -----------
            200004     Possible Multipart Unmatched Boundary.

    REQUEST-911-METHOD-ENFORCEMENT:
        Description:

        Rules:
            RuleId     Description
            ------     -----------
            911011     Rule 911011
            911012     Rule 911012
            911100     Method is not allowed by policy
            911013     Rule 911013
            911014     Rule 911014
            911015     Rule 911015
            911016     Rule 911016
            911017     Rule 911017
            911018     Rule 911018

    REQUEST-913-SCANNER-DETECTION:
        Description:

        Rules:
            RuleId     Description
            ------     -----------
            913011     Rule 913011
            913012     Rule 913012
            913100     Found User-Agent associated with security scanner
            913110     Found request header associated with security scanner
            913120     Found request filename/argument associated with security scanner
            913013     Rule 913013
            913014     Rule 913014
            913101     Found User-Agent associated with scripting/generic HTTP client
            913102     Found User-Agent associated with web crawler/bot
            913015     Rule 913015
            913016     Rule 913016
            913017     Rule 913017
            913018     Rule 913018

            ...        ...
```

Disable rules

The following example disables rules `911011` and `911012` on an application gateway:

```
$disabledrules=New-AzApplicationGatewayFirewallDisabledRuleGroupConfig -RuleGroupName REQUEST-911-METHOD-
ENFORCEMENT -Rules 911011,911012
Set-AzApplicationGatewayWebApplicationFirewallConfiguration -ApplicationGateway $gw -Enabled $true -
FirewallMode Detection -RuleSetVersion 3.0 -RuleSetType OWASP -DisabledRuleGroups $disabledrules
Set-AzApplicationGateway -ApplicationGateway $gw
```

Mandatory rules

The following list contains conditions that cause the WAF to block the request while in Prevention Mode (in Detection Mode they are logged as exceptions). These can't be configured or disabled:

- Failure to parse the request body results in the request being blocked, unless body inspection is turned off (XML, JSON, form data)
- Request body (with no files) data length is larger than the configured limit
- Request body (including files) is larger than the limit
- An internal error happened in the WAF engine

CRS 3.x specific:

- Inbound anomaly score exceeded threshold

Next steps

After you configure your disabled rules, you can learn how to view your WAF logs. For more information, see Application Gateway Diagnostics.

Customize Web Application Firewall rules using the Azure CLI

11/13/2019 • 2 minutes to read • Edit Online

The Azure Application Gateway Web Application Firewall (WAF) provides protection for web applications. These protections are provided by the Open Web Application Security Project (OWASP) Core Rule Set (CRS). Some rules can cause false positives and block real traffic. For this reason, Application Gateway provides the capability to customize rule groups and rules. For more information on the specific rule groups and rules, see List of Web Application Firewall CRS rule groups and rules.

View rule groups and rules

The following code examples show how to view rules and rule groups that are configurable.

View rule groups

The following example shows how to view the rule groups:

```
az network application-gateway waf-config list-rule-sets --type OWASP
```

The following output is a truncated response from the preceding example:

```
[
  {
    "id":
"/subscriptions//resourceGroups//providers/Microsoft.Network/applicationGatewayAvailableWafRuleSets/",
    "location": null,
    "name": "OWASP_3.0",
    "provisioningState": "Succeeded",
    "resourceGroup": "",
    "ruleGroups": [
      {
        "description": "",
        "ruleGroupName": "REQUEST-910-IP-REPUTATION",
        "rules": null
      },
      ...
    ],
    "ruleSetType": "OWASP",
    "ruleSetVersion": "3.0",
    "tags": null,
    "type": "Microsoft.Network/applicationGatewayAvailableWafRuleSets"
  },
  {
    "id":
"/subscriptions//resourceGroups//providers/Microsoft.Network/applicationGatewayAvailableWafRuleSets/",
    "location": null,
    "name": "OWASP_2.2.9",
    "provisioningState": "Succeeded",
    "resourceGroup": "",
   "ruleGroups": [
      {
        "description": "",
        "ruleGroupName": "crs_20_protocol_violations",
        "rules": null
      },
      ...
    ],
    "ruleSetType": "OWASP",
    "ruleSetVersion": "2.2.9",
    "tags": null,
    "type": "Microsoft.Network/applicationGatewayAvailableWafRuleSets"
  }
]
```


View rules in a rule group

The following example shows how to view rules in a specified rule group:

```
az network application-gateway waf-config list-rule-sets --group "REQUEST-910-IP-REPUTATION"
```

The following output is a truncated response from the preceding example:

```
[
  {
    "id":
"/subscriptions//resourceGroups//providers/Microsoft.Network/applicationGatewayAvailableWafRuleSets/",
    "location": null,
    "name": "OWASP_3.0",
    "provisioningState": "Succeeded",
    "resourceGroup": "",
    "ruleGroups": [
      {
        "description": "",
        "ruleGroupName": "REQUEST-910-IP-REPUTATION",
        "rules": [
          {
            "description": "Rule 910011",
            "ruleId": 910011
          },
          ...
        ]
      }
    ],
    "ruleSetType": "OWASP",
    "ruleSetVersion": "3.0",
    "tags": null,
    "type": "Microsoft.Network/applicationGatewayAvailableWafRuleSets"
  }
]
```


Disable rules

The following example disables rules `910018` and `910017` on an application gateway:

```
az network application-gateway waf-config set --resource-group AdatumAppGatewayRG --gateway-name
AdatumAppGateway --enabled true --rule-set-version 3.0 --disabled-rules 910018 910017
```

Mandatory rules

The following list contains conditions that cause the WAF to block the request while in Prevention Mode (in Detection Mode they are logged as exceptions). These can't be configured or disabled:

- Failure to parse the request body results in the request being blocked, unless body inspection is turned off (XML, JSON, form data)
- Request body (with no files) data length is larger than the configured limit
- Request body (including files) is larger than the limit
- An internal error happened in the WAF engine

CRS 3.x specific:

- Inbound anomaly score exceeded threshold

Next steps

After you configure your disabled rules, you can learn how to view your WAF logs. For more information, see Application Gateway diagnostics.

Configure Web Application Firewall v2 on Application Gateway with a custom rule using Azure PowerShell

2/19/2020 • 2 minutes to read • Edit Online

Custom rules allow you to create your own rules evaluated for each request that passes through the Web Application Firewall (WAF) v2. These rules hold a higher priority than the rest of the rules in the managed rule sets. The custom rules have an action (to allow or block), a match condition, and an operator to allow full customization.

This article creates an Application Gateway WAF v2 that uses a custom rule. The custom rule blocks traffic if the request header contains User-Agent *evilbot*.

To see more custom rule examples, see Create and use custom web application firewall rules

If you want run the Azure PowerShell in this article in one continuous script that you can copy, paste, and run, see Azure Application Gateway PowerShell samples.

Prerequisites

Azure PowerShell module

If you choose to install and use Azure PowerShell locally, this script requires the Azure PowerShell module version 2.1.0 or later.

1. To find the version, run `Get-Module -ListAvailable Az`. If you need to upgrade, see Install Azure PowerShell module.
2. To create a connection with Azure, run `Connect-AzAccount`.

If you don't have an Azure subscription, create a free account before you begin.

Example script

Set up variables

```
$rgname = "CustomRulesTest"

$location = "East US"

$appgwName = "WAFCustomRules"
```

Create a resource group

```
$resourceGroup = New-AzResourceGroup -Name $rgname -Location $location
```

Create a VNet

```
$sub1 = New-AzVirtualNetworkSubnetConfig -Name "appgwSubnet" -AddressPrefix "10.0.0.0/24"

$sub2 = New-AzVirtualNetworkSubnetConfig -Name "backendSubnet" -AddressPrefix "10.0.1.0/24"

$vnet = New-AzvirtualNetwork -Name "Vnet1" -ResourceGroupName $rgname -Location $location `
  -AddressPrefix "10.0.0.0/16" -Subnet @($sub1, $sub2)
```

Create a Static Public VIP

```
$publicip = New-AzPublicIpAddress -ResourceGroupName $rgname -name "AppGwIP" `
  -location $location -AllocationMethod Static -Sku Standard
```

Create pool and frontend port

```
$gwSubnet = Get-AzVirtualNetworkSubnetConfig -Name "appgwSubnet" -VirtualNetwork $vnet

$gipconfig = New-AzApplicationGatewayIPConfiguration -Name "AppGwIpConfig" -Subnet $gwSubnet

$fipconfig01 = New-AzApplicationGatewayFrontendIPConfig -Name "fipconfig" -PublicIPAddress $publicip

$pool = New-AzApplicationGatewayBackendAddressPool -Name "pool1" `
  -BackendIPAddresses testbackend1.westus.cloudapp.azure.com, testbackend2.westus.cloudapp.azure.com

$fp01 = New-AzApplicationGatewayFrontendPort -Name "port1" -Port 80
```

Create a listener, http setting, rule, and autoscale

```
$listener01 = New-AzApplicationGatewayHttpListener -Name "listener1" -Protocol Http `
  -FrontendIPConfiguration $fipconfig01 -FrontendPort $fp01

$poolSetting01 = New-AzApplicationGatewayBackendHttpSettings -Name "setting1" -Port 80 `
  -Protocol Http -CookieBasedAffinity Disabled

$rule01 = New-AzApplicationGatewayRequestRoutingRule -Name "rule1" -RuleType basic `
  -BackendHttpSettings $poolSetting01 -HttpListener $listener01 -BackendAddressPool $pool

$autoscaleConfig = New-AzApplicationGatewayAutoscaleConfiguration -MinCapacity 3

$sku = New-AzApplicationGatewaySku -Name WAF_v2 -Tier WAF_v2
```

Create two custom rules and apply it to WAF policy

```
# Create WAF config
$wafConfig = New-AzApplicationGatewayWebApplicationFirewallConfiguration -Enabled $true -FirewallMode
"Prevention" -RuleSetType "OWASP" -RuleSetVersion "3.0"
# Create a User-Agent header custom rule
$variable = New-AzApplicationGatewayFirewallMatchVariable -VariableName RequestHeaders -Selector User-Agent
$condition = New-AzApplicationGatewayFirewallCondition -MatchVariable $variable -Operator Contains -MatchValue
"evilbot" -Transform Lowercase -NegationCondition $False
$rule = New-AzApplicationGatewayFirewallCustomRule -Name blockEvilBot -Priority 2 -RuleType MatchRule -
MatchCondition $condition -Action Block

# Create a geo-match custom rule
$var2 = New-AzApplicationGatewayFirewallMatchVariable -VariableName RequestUri
$condition2 = New-AzApplicationGatewayFirewallCondition -MatchVariable $var2 -Operator GeoMatch -MatchValue
"US"  -NegationCondition $False
$rule2 = New-AzApplicationGatewayFirewallCustomRule -Name allowUS -Priority 14 -RuleType MatchRule -
MatchCondition $condition2 -Action Allow

# Create a firewall policy
$wafPolicy = New-AzApplicationGatewayFirewallPolicy -Name wafpolicyNew -ResourceGroup $rgname -Location
$location -CustomRule $rule,$rule2
```

Create the Application Gateway

```
$appgw = New-AzApplicationGateway -Name $appgwName -ResourceGroupName $rgname `
  -Location $location -BackendAddressPools $pool `
  -BackendHttpSettingsCollection  $poolSetting01 `
  -GatewayIpConfigurations $gipconfig -FrontendIpConfigurations $fipconfig01 `
  -FrontendPorts $fp01 -HttpListeners $listener01 `
  -RequestRoutingRules $rule01 -Sku $sku -AutoscaleConfiguration $autoscaleConfig `
  -WebApplicationFirewallConfig $wafConfig `
  -FirewallPolicy $wafPolicy
```

Next steps

Learn more about Web Application Firewall on Application Gateway

Create and use Web Application Firewall v2 custom rules on Application Gateway

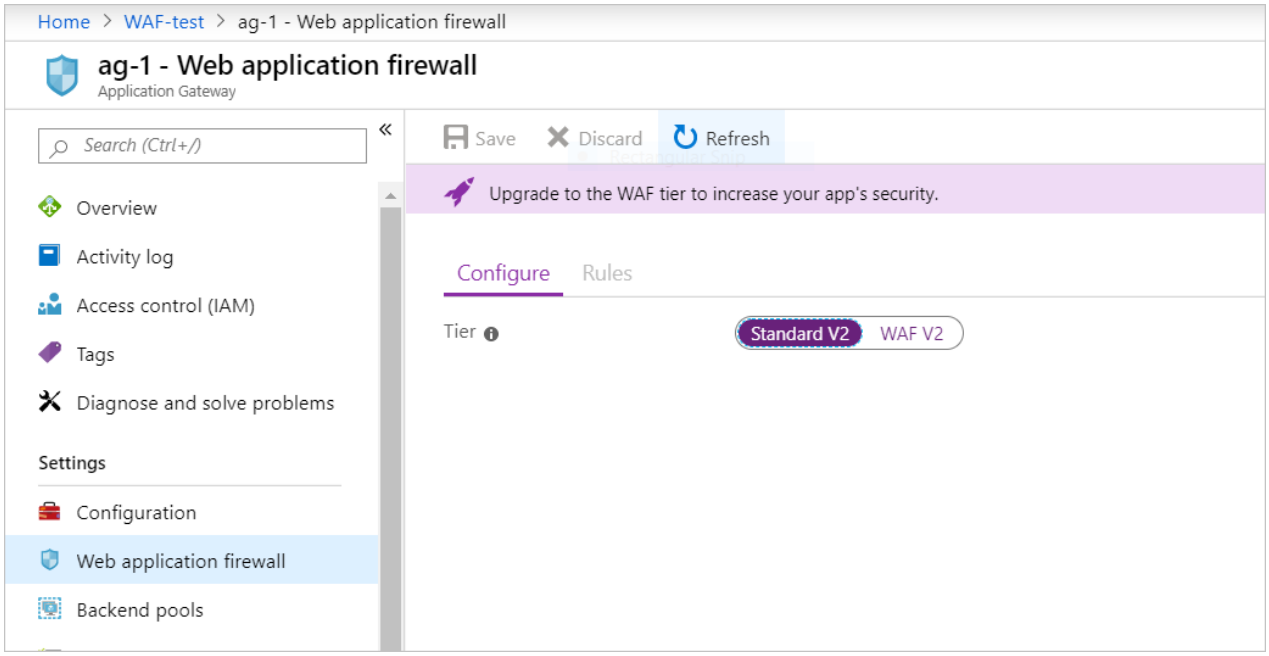
2/14/2020 • 6 minutes to read • Edit Online

The Web Application Firewall (WAF) v2 on Azure Application Gateway provides protection for web applications. This protection is provided by the Open Web Application Security Project (OWASP) Core Rule Set (CRS). In some cases, you may need to create your own custom rules to meet your specific needs. For more information about WAF custom rules, see Custom web application firewall rules overview.

This article shows you some example custom rules that you can create and use with your v2 WAF. To learn how to deploy a WAF with a custom rule using Azure PowerShell, see Configure Web Application Firewall custom rules using Azure PowerShell.

NOTE

If your application gateway is not using the WAF tier, the option to upgrade the application gateway to the WAF tier appears in the right pane.



Example 1

You know there's a bot named *evilbot* that you want to block from crawling your website. In this case, you'll block on the User-Agent *evilbot* in the request headers.

Logic: p

```
$variable = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RequestHeaders `
   -Selector User-Agent

$condition = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable `
   -Operator Contains `
   -MatchValue "evilbot" `
   -Transform Lowercase `
   -NegationCondition $False

$rule = New-AzApplicationGatewayFirewallCustomRule `
   -Name blockEvilBot `
   -Priority 2 `
   -RuleType MatchRule `
   -MatchCondition $condition `
   -Action Block
```

And here is the corresponding JSON:

```
{
  "customRules": [
    {
      "name": "blockEvilBot",
      "ruleType": "MatchRule",
      "priority": 2,
      "action": "Block",
      "matchConditions": [
        {
          "matchVariable": "RequestHeaders",
          "operator": "User-Agent",
          "matchValues": [
            "evilbot"
          ]
        }
      ]
    }
  ]
}
```

To see a WAF deployed using this custom rule, see Configure a Web Application Firewall custom rule using Azure PowerShell.

Example 1a

You can accomplish the same thing using a regular expression:

```
$variable = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RequestHeaders `
   -Selector User-Agent

$condition = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable `
   -Operator Regex `
   -MatchValue "evilbot" `
   -Transform Lowercase `
   -NegationCondition $False

$rule = New-AzApplicationGatewayFirewallCustomRule `
   -Name blockEvilBot `
   -Priority 2 `
   -RuleType MatchRule `
   -MatchCondition $condition `
   -Action Block
```

And the corresponding JSON:

```
{
  "customRules": [
    {
      "name": "blockEvilBot",
      "ruleType": "MatchRule",
      "priority": 2,
      "action": "Block",
      "matchConditions": [
        {
          "matchVariable": "RequestHeaders",
          "operator": "User-Agent",
          "matchValues": [
            "evilbot"
          ]
        }
      ]
    }
  ]
}
```

Example 2

You want to allow traffic from the US using the GeoMatch operator:

```
$variable = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RemoteAddr `

$condition = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable `
   -Operator GeoMatch `
   -MatchValue "US" `
   -Transform Lowercase `
   -NegationCondition $False

$rule = New-AzApplicationGatewayFirewallCustomRule `
   -Name "allowUS" `
   -Priority 2 `
   -RuleType MatchRule `
   -MatchCondition $condition `
   -Action Allow
```

And the corresponding JSON:

```
{
  "customRules": [
    {
      "name": "allowUS",
      "ruleType": "MatchRule",
      "priority": 2,
      "action": "Allow",
      "matchConditions": [
        {
          "matchVariable": "RemoteAddr",
          "operator": "GeoMatch",
          "matchValues": [
            "US"
          ]
        }
      ]
    }
  ]
}
```

Example 3

You want to block all requests from IP addresses in the range 198.168.5.0/24.

In this example, you'll block all traffic that comes from an IP addresses range. The name of the rule is *myrule1* and the priority is set to 10.

Logic: p

```
$variable1 = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RemoteAddr

$condition1 = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable1 `
   -Operator IPMatch `
   -MatchValue "192.168.5.0/24" `
   -NegationCondition $False

$rule = New-AzApplicationGatewayFirewallCustomRule `
   -Name myrule1 `
   -Priority 10 `
   -RuleType MatchRule `
   -MatchCondition $condition1 `
   -Action Block
```

Here's the corresponding JSON:

```
{
  "customRules": [
    {
      "name": "myrule1",
      "ruleType": "MatchRule",
      "priority": 10,
      "action": "Block",
      "matchConditions": [
        {
          "matchVariable": "RemoteAddr",
          "operator": "IPMatch",
          "matchValues": [
            "192.168.5.0/24"
          ]
        }
      ]
    }
  ]
}
```

Corresponding CRS rule: `SecRule REMOTE_ADDR "@ipMatch 192.168.5.0/24" "id:7001,deny"`

Example 4

For this example, you want to block User-Agent *evilbot*, and traffic in the range 192.168.5.0/24. To accomplish this, you can create two separate match conditions, and put them both in the same rule. This ensures that if both *evilbot* in the User-Agent header **and** IP addresses from the range 192.168.5.0/24 are matched, then the request is blocked.

Logic: p **and** q

```
$variable1 = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RemoteAddr

 $variable2 = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RequestHeaders `
   -Selector User-Agent

$condition1 = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable1 `
   -Operator IPMatch `
   -MatchValue "192.168.5.0/24" `
   -NegationCondition $False

$condition2 = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable2 `
   -Operator Contains `
   -MatchValue "evilbot" `
   -Transform Lowercase `
   -NegationCondition $False

 $rule = New-AzApplicationGatewayFirewallCustomRule `
   -Name myrule `
   -Priority 10 `
   -RuleType MatchRule `
   -MatchCondition $condition1, $condition2 `
   -Action Block
```

Here's the corresponding JSON:

```
{

    "customRules": [
      {
        "name": "myrule",
        "ruleType": "MatchRule",
        "priority": 10,
        "action": "block",
        "matchConditions": [
            {
              "matchVariable": "RemoteAddr",
              "operator": "IPMatch",
              "negateCondition": false,
              "matchValues": [
                "192.168.5.0/24"
              ]
            },
            {
              "matchVariable": "RequestHeaders",
              "selector": "User-Agent",
              "operator": "Contains",
              "transforms": [
                "Lowercase"
              ],
              "matchValues": [
                "evilbot"
              ]
            }
        ]
      }
    ]
  }
```

Example 5

For this example, you want to block if the request is either outside of the IP address range *192.168.5.0/24*, or the user agent string isn't *chrome* (meaning the user isn't using the Chrome browser). Since this logic uses **or**, the two conditions are in separate rules as seen in the following example. *myrule1* and *myrule2* both need to match to block the traffic.

Logic: **not** (p **and** q) = **not** p **or not** q.

```
$variable1 = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RemoteAddr

$variable2 = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RequestHeaders `
   -Selector User-Agent

$condition1 = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable1 `
   -Operator IPMatch `
   -MatchValue "192.168.5.0/24" `
   -NegationCondition $True

$condition2 = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable2 `
   -Operator Contains `
   -MatchValue "chrome" `
   -Transform Lowercase `
   -NegationCondition $True

$rule1 = New-AzApplicationGatewayFirewallCustomRule `
   -Name myrule1 `
   -Priority 10 `
   -RuleType MatchRule `
   -MatchCondition $condition1 `
   -Action Block

$rule2 = New-AzApplicationGatewayFirewallCustomRule `
   -Name myrule2 `
   -Priority 20 `
   -RuleType MatchRule `
   -MatchCondition $condition2 `
   -Action Block
```

And the corresponding JSON:

```
{
    "customRules": [
      {
        "name": "myrule1",
        "ruleType": "MatchRule",
        "priority": 10,
        "action": "block",
        "matchConditions": [
          {
            "matchVariable": "RequestHeaders",
            "operator": "IPMatch",
            "negateCondition": true,
            "matchValues": [
              "192.168.5.0/24"
            ]
          }
        ]
      },
      {
        "name": "myrule2",
        "ruleType": "MatchRule",
        "priority": 20,
        "action": "block",
        "matchConditions": [
          {
            "matchVariable": "RequestHeaders",
            "selector": "User-Agent",
            "operator": "Contains",
            "negateCondition": true,
            "transforms": [
              "Lowercase"
            ],
            "matchValues": [
              "chrome"
            ]
          }
        ]
      }
    ]
  }
```

Example 6

You want to block custom SQLI. Since the logic used here is **or**, and all the values are in the *RequestUri*, all of the *MatchValues* can be in a comma-separated list.

Logic: p **or** q **or** r

```
$variable1 = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RequestUri
$condition1 = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable1 `
   -Operator Contains `
   -MatchValue "1=1", "drop tables", "'—" `
   -NegationCondition $False

$rule1 = New-AzApplicationGatewayFirewallCustomRule `
   -Name myrule4 `
   -Priority 10 `
   -RuleType MatchRule `
   -MatchCondition $condition1 `
   -Action Block
```

Corresponding JSON:

```
{
  "customRules": [
    {
      "name": "myrule4",
      "ruleType": "MatchRule",
      “priority”: 10
      "action": "block",
      "matchConditions": [
        {
          "matchVariable": "RequestUri",
          "operator": "Contains",
          "matchValues": [
            "1=1",
            "drop tables",
            "'--"
          ]
        }
      ]
    }
  ]
}
```

Alternative Azure PowerShell:

```
$variable1 = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RequestUri
$condition1 = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable1 `
   -Operator Contains `
   -MatchValue "1=1" `
   -NegationCondition $False

$rule1 = New-AzApplicationGatewayFirewallCustomRule `
   -Name myrule1 `
   -Priority 10 `
   -RuleType MatchRule `
   -MatchCondition $condition1 `
-Action Block

$variable2 = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RequestUri

$condition2 = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable2 `
   -Operator Contains `
   -MatchValue "drop tables" `
   -NegationCondition $False

$rule2 = New-AzApplicationGatewayFirewallCustomRule `
   -Name myrule2 `
   -Priority 20 `
   -RuleType MatchRule `
   -MatchCondition $condition2 `
   -Action Block

$variable3 = New-AzApplicationGatewayFirewallMatchVariable `
   -VariableName RequestUri

$condition3 = New-AzApplicationGatewayFirewallCondition `
   -MatchVariable $variable3 `
   -Operator Contains `
   -MatchValue "’—" `
   -NegationCondition $False

$rule3 = New-AzApplicationGatewayFirewallCustomRule `
   -Name myrule3 `
   -Priority 30 `
   -RuleType MatchRule `
   -MatchCondition $condition3 `
   -Action Block
```

Corresponding JSON:

```
{
  "customRules": [
    {
      "name": "myrule1",
      "ruleType": "MatchRule",
      "priority": 10,
      "action": "block",
      "matchConditions": [
        {
          "matchVariable": "RequestUri",
          "operator": "Contains",
          "matchValues": [
            "1=1"
          ]
        }
      ]
    },
    {
      "name": "myrule2",
      "ruleType": "MatchRule",
      "priority": 20,
      "action": "block",
      "matchConditions": [
        {
          "matchVariable": "RequestUri",
          "operator": "Contains",
          "transforms": [
            "Lowercase"
          ],
          "matchValues": [
            "drop tables"
          ]
        }
      ]
    },
    {
      "name": "myrule3",
      "ruleType": "MatchRule",
      "priority": 30,
      "action": "block",
      "matchConditions": [
        {
          "matchVariable": "RequestUri",
          "operator": "Contains",
          "matchValues": [
            "'--"
          ]
        }
      ]
    }
  ]
}
```

Next steps

After you create your custom rules, you can learn how to view your WAF logs. For more information, see Application Gateway diagnostics.

Configure bot protection for Web Application Firewall on Azure Application Gateway (Preview)

11/4/2019 • 2 minutes to read • Edit Online

This article shows you how to configure a bot protection rule in Azure Web Application Firewall (WAF) for Application Gateway using the Azure portal.

You can enable a managed bot protection rule set for your WAF to block or log requests from known malicious IP addresses. The IP addresses are sourced from the Microsoft Threat Intelligence feed. Intelligent Security Graph powers Microsoft threat intelligence and is used by multiple services including Azure Security Center.

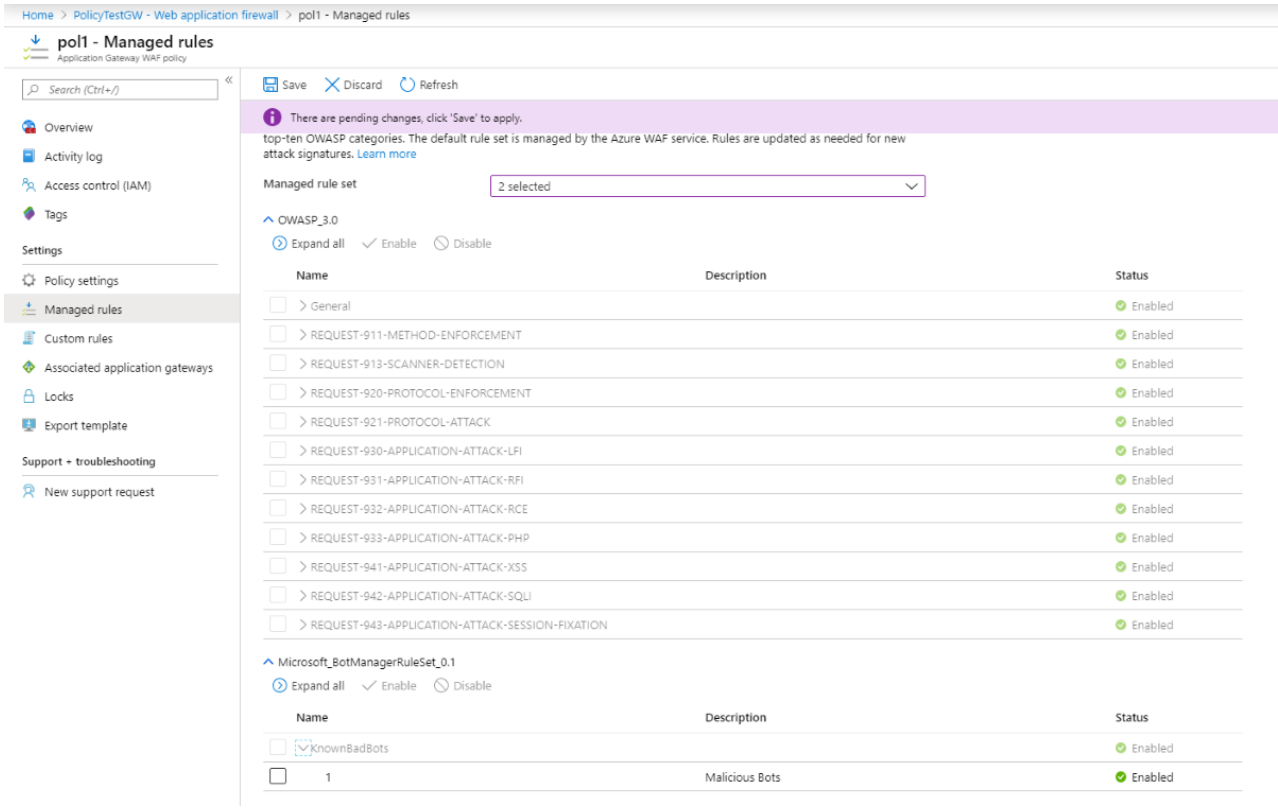
> **NOTE**
> The bot protection rule set is currently in public preview and is provided with a preview service level agreement. Certain features may not be supported or may have constrained capabilities. See theSupplemental Terms of Use for Microsoft Azure Previewsfor details.

Prerequisites

Create a basic WAF policy for Application Gateway by following the instructions described inCreate Web Application Firewall policies for Application Gateway.

Enable bot protection rule set

1. In the **Basic** policy page that you created previously, under **Settings**, select **Rules**.
2. In the details page, under the**Manage rules**section, from the drop-down menu, select the check box for the bot Protection rule, and then select **Save**.



Next steps

For more information about custom rules, see Custom rules for Web Application Firewall v2 on Azure Application Gateway.

Associate a WAF policy with an existing Application Gateway

11/14/2019 • 2 minutes to read • Edit Online

You can use Azure PowerShell to create a WAF Policy, but you might already have an Application Gateway and just want to associate a WAF Policy to it. In this article, you do just that; you create a WAF Policy and associate it to an already existing Application Gateway.

1. Get your Application Gateway and Firewall Policy. If you don't have an existing Firewall Policy, see step 2.

```
Connect-AzAccount
Select-AzSubscription -Subscription "<sub id>"`

#Get Application Gateway and existing policy object or create new`
$gw = Get-AzApplicationGateway -Name <Waf v2> -ResourceGroupName <RG name>`
$policy = Get-AzApplicationGatewayFirewallPolicy -Name <policy name> -ResourceGroupName <RG name>`
```

2. (Optional) Create a Firewall Policy.

```
New-AzApplicationGatewayFirewallPolicy -Name <policy name> -ResourceGroupName <RG name>'
$policy = Get-AzApplicationGatewayFirewallPolicy -Name <policy name> -ResourceGroupName <RG name>`
```

> **NOTE**
> If you are creating this WAF Policy to transition from a WAF Config to a WAF Policy, then the Policy needs to be an exact copy of your old Config. This means that every exclusion, custom rule, disabled rule group, etc. needs to be the exact same as it is in the WAF Config.

3. (Optional) You can configure the WAF policy to suit your needs. This includes custom rules, disabling rules/rule groups, exclusions, setting file upload limits, etc. If you skip this step, all defaults will be selected.

4. Save the policy, and attach it to your Application Gateway.

```
#Save the policy itself
Set-AzApplicationGatewayFirewallPolicy -InputObject $policy`

#Attach the policy to an Application Gateway
$gw.FirewallPolicy = $policy`

#Save the Application Gateway
Set-AzApplicationGateway -ApplicationGateway $gw`
```

Next steps

Learn about Custom Rules.

Diagnostic logs for Azure Web Application Firewall

1/14/2020 • 11 minutes to read • Edit Online

You can monitor Web Application Firewall resources using logs. You can save performance, access, and other data or consume it from a resource for monitoring purposes.

> **NOTE**
> This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Diagnostic logs

You can use different types of logs in Azure to manage and troubleshoot application gateways. You can access some of these logs through the portal. All logs can be extracted from Azure Blob storage and viewed in different tools, such as Azure Monitor logs, Excel, and Power BI. You can learn more about the different types of logs from the following list:

- **Activity log**: You can use Azure activity logs (formerly known as operational logs and audit logs) to view all operations that are submitted to your Azure subscription, and their status. Activity log entries are collected by default, and you can view them in the Azure portal.
- **Access log**: You can use this log to view Application Gateway access patterns and analyze important information. This includes the caller's IP, requested URL, response latency, return code, and bytes in and out. An access log is collected every 300 seconds. This log contains one record per instance of Application Gateway. The Application Gateway instance is identified by the instanceId property.
- **Performance log**: You can use this log to view how Application Gateway instances are performing. This log captures performance information for each instance, including total requests served, throughput in bytes, total requests served, failed request count, and healthy and unhealthy back-end instance count. A performance log is collected every 60 seconds. The Performance log is available only for the v1 SKU. For the v2 SKU, use Metrics for performance data.
- **Firewall log**: You can use this log to view the requests that are logged through either detection or prevention mode of an application gateway that is configured with the web application firewall.

> **NOTE**
> Logs are available only for resources deployed in the Azure Resource Manager deployment model. You cannot use logs for resources in the classic deployment model. For a better understanding of the two models, see the Understanding Resource Manager deployment and classic deployment article.

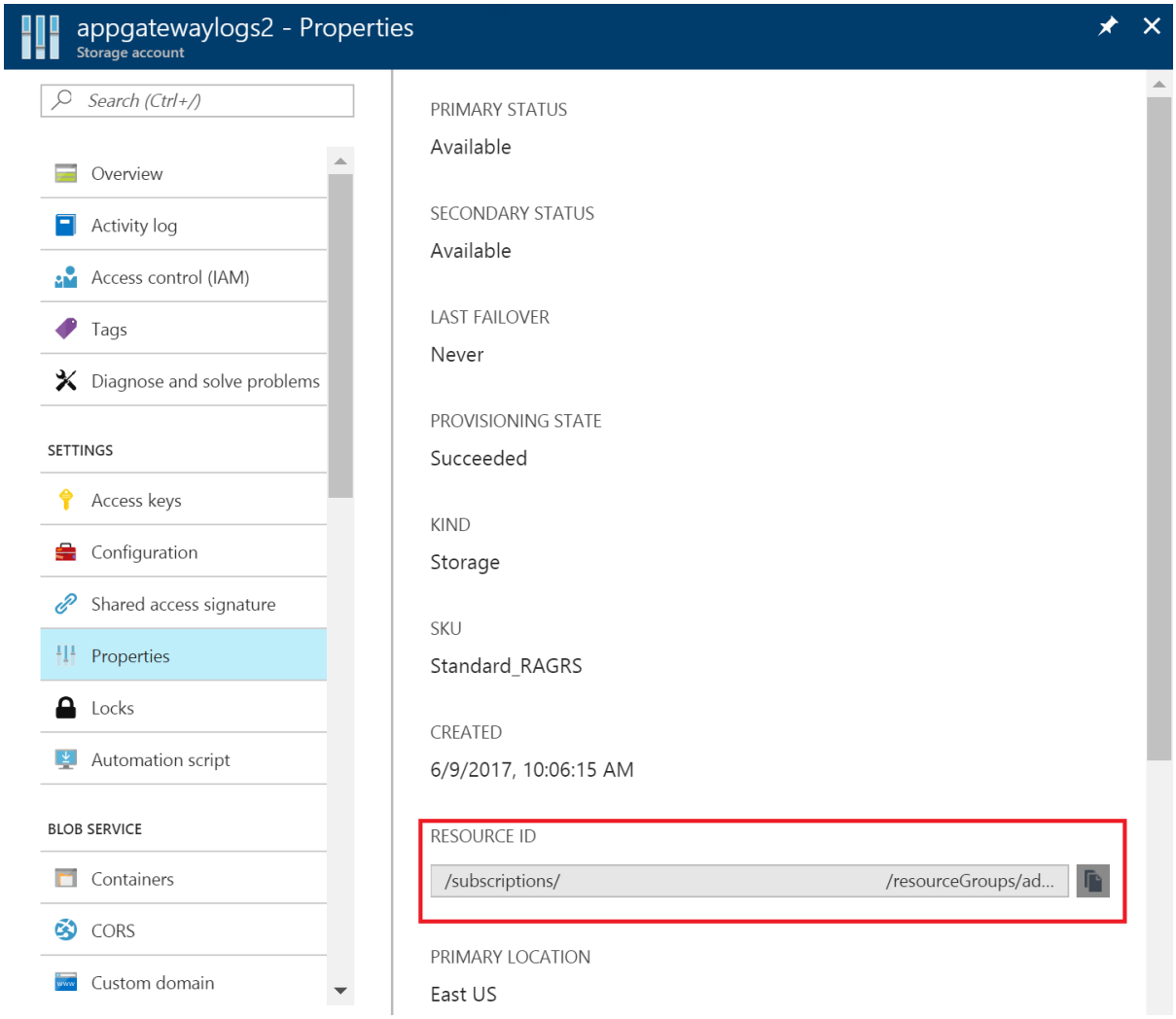
You have three options for storing your logs:

- **Storage account**: Storage accounts are best used for logs when logs are stored for a longer duration and reviewed when needed.
- **Event hubs**: Event hubs are a great option for integrating with other security information and event management (SIEM) tools to get alerts on your resources.
- **Azure Monitor logs**: Azure Monitor logs is best used for general real-time monitoring of your application or looking at trends.

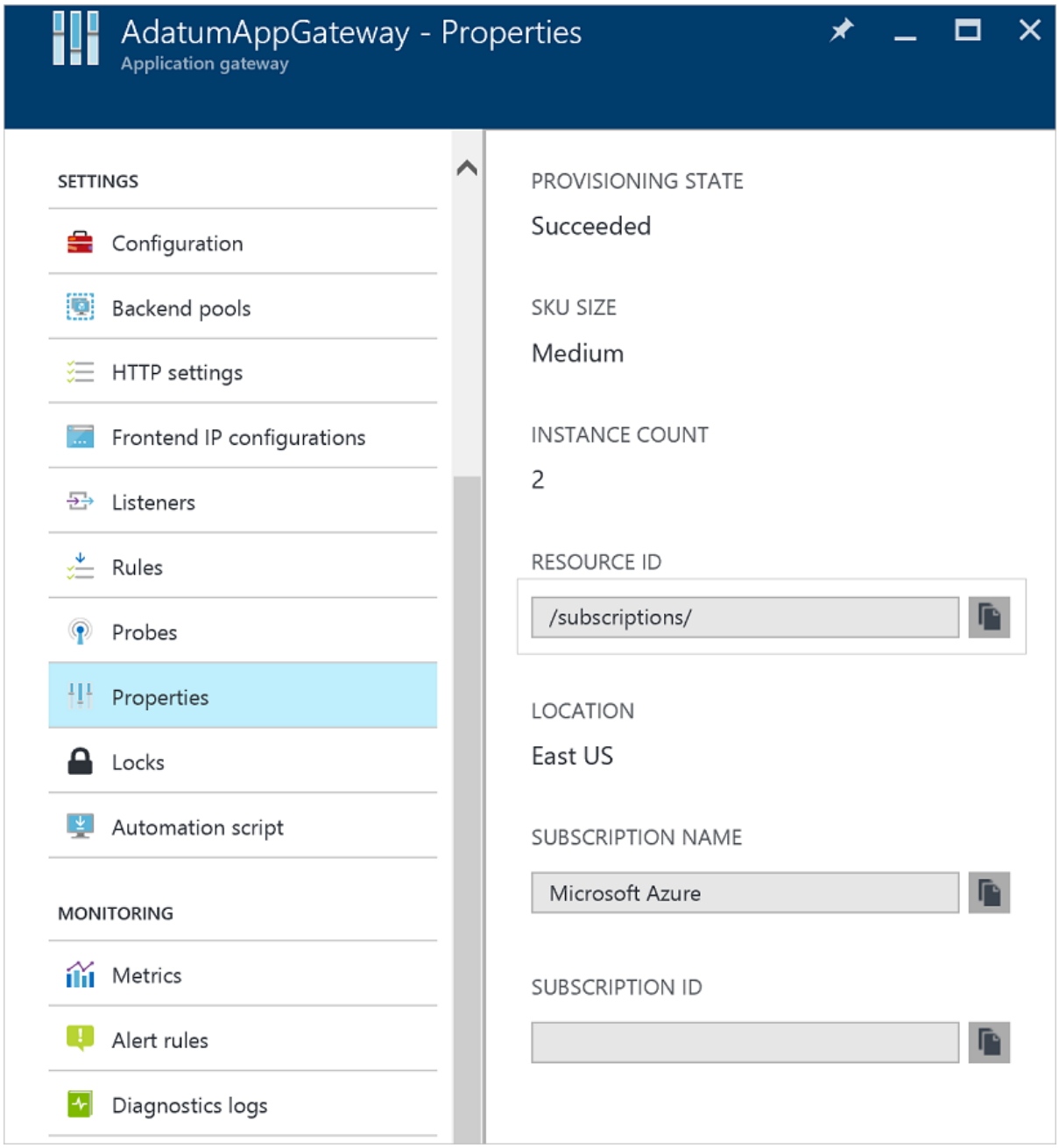
Enable logging through PowerShell

Activity logging is automatically enabled for every Resource Manager resource. You must enable access and performance logging to start collecting the data available through those logs. To enable logging, use the following steps:

1. Note your storage account's resource ID, where the log data is stored. This value is of the form: /subscriptions/<subscriptionId>/resourceGroups/<resource group name>/providers/Microsoft.Storage/storageAccounts/<storage account name>. You can use any storage account in your subscription. You can use the Azure portal to find this information.



2. Note your application gateway's resource ID for which logging is enabled. This value is of the form: /subscriptions/<subscriptionId>/resourceGroups/<resource group name>/providers/Microsoft.Network/applicationGateways/<application gateway name>. You can use the portal to find this information.



3. Enable diagnostic logging by using the following PowerShell cmdlet:

```
Set-AzDiagnosticSetting  -ResourceId /subscriptions/<subscriptionId>/resourceGroups/<resource group
name>/providers/Microsoft.Network/applicationGateways/<application gateway name> -StorageAccountId
/subscriptions/<subscriptionId>/resourceGroups/<resource group
name>/providers/Microsoft.Storage/storageAccounts/<storage account name> -Enabled $true
```

TIP

Activity logs do not require a separate storage account. The use of storage for access and performance logging incurs service charges.

Enable logging through the Azure portal

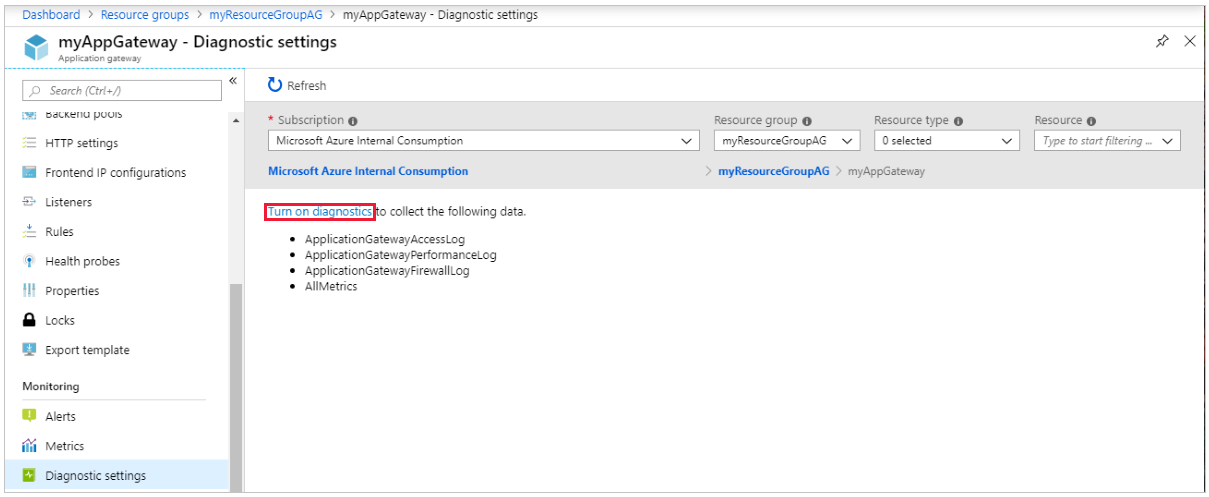
1. In the Azure portal, find your resource and select **Diagnostic settings**.

   For Application Gateway, three logs are available:

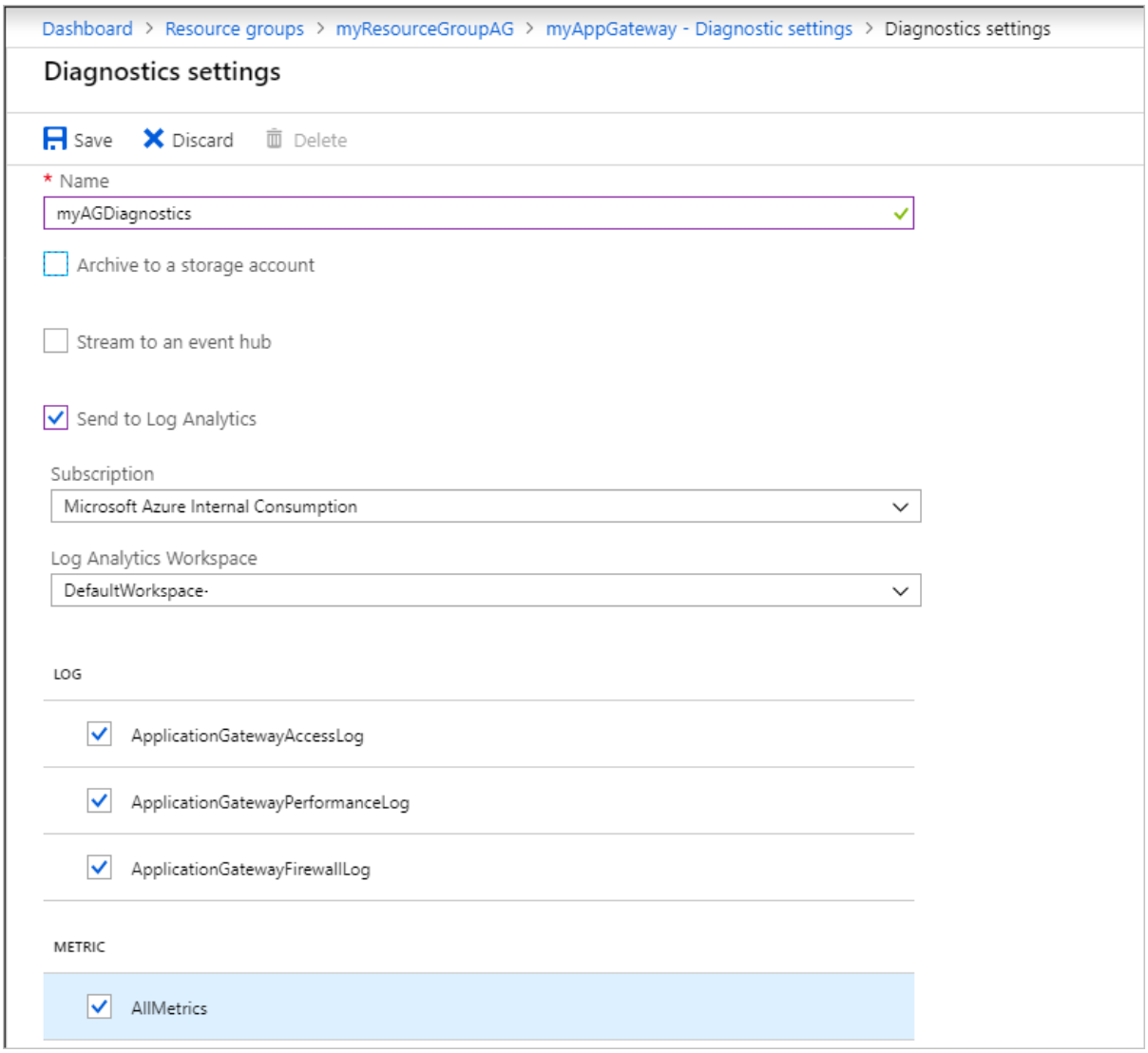
   - Access log
   - Performance log

- Firewall log

2. To start collecting data, select **Turn on diagnostics**.



3. The **Diagnostics settings** page provides the settings for the diagnostic logs. In this example, Log Analytics stores the logs. You can also use event hubs and a storage account to save the diagnostic logs.



4. Type a name for the settings, confirm the settings, and select **Save**.

Activity log

Azure generates the activity log by default. The logs are preserved for 90 days in the Azure event logs store. Learn more about these logs by reading the View events and activity log article.

Access log

The access log is generated only if you've enabled it on each Application Gateway instance, as detailed in the preceding steps. The data is stored in the storage account that you specified when you enabled the logging. Each access of Application Gateway is logged in JSON format, as shown in the following example for v1:

VALUE	DESCRIPTION
instanceId	Application Gateway instance that served the request.
clientIP	Originating IP for the request.
clientPort	Originating port for the request.
httpMethod	HTTP method used by the request.
requestUri	URI of the received request.
RequestQuery	**Server-Routed**: Back-end pool instance that was sent the request. **X-AzureApplicationGateway-LOG-ID**: Correlation ID used for the request. It can be used to troubleshoot traffic issues on the back-end servers. **SERVER-STATUS**: HTTP response code that Application Gateway received from the back end.
UserAgent	User agent from the HTTP request header.
httpStatus	HTTP status code returned to the client from Application Gateway.
httpVersion	HTTP version of the request.
receivedBytes	Size of packet received, in bytes.
sentBytes	Size of packet sent, in bytes.
timeTaken	Length of time (in milliseconds) that it takes for a request to be processed and its response to be sent. This is calculated as the interval from the time when Application Gateway receives the first byte of an HTTP request to the time when the response send operation finishes. It's important to note that the Time-Taken field usually includes the time that the request and response packets are traveling over the network.
sslEnabled	Whether communication to the back-end pools used SSL. Valid values are on and off.
host	The hostname with which the request has been sent to the backend server. If backend hostname is being overridden, this name will reflect that.
originalHost	The hostname with which the request was received by the Application Gateway from the client.

```
{
    "resourceId":
"/SUBSCRIPTIONS/{subscriptionId}/RESOURCEGROUPS/PEERINGTEST/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/{ap
plicationGatewayName}",
    "operationName": "ApplicationGatewayAccess",
    "time": "2017-04-26T19:27:38Z",
    "category": "ApplicationGatewayAccessLog",
    "properties": {
        "instanceId": "ApplicationGatewayRole_IN_0",
        "clientIP": "191.96.249.97",
        "clientPort": 46886,
        "httpMethod": "GET",
        "requestUri": "/phpmyadmin/scripts/setup.php",
        "requestQuery": "X-AzureApplicationGateway-CACHE-HIT=0&SERVER-ROUTED=10.4.0.4&X-
AzureApplicationGateway-LOG-ID=874f1f0f-6807-41c9-b7bc-f3cfa74aa0b1&SERVER-STATUS=404",
        "userAgent": "-",
        "httpStatus": 404,
        "httpVersion": "HTTP/1.0",
        "receivedBytes": 65,
        "sentBytes": 553,
        "timeTaken": 205,
        "sslEnabled": "off",
        "host": "www.contoso.com",
        "originalHost": "www.contoso.com"
    }
}
```


For Application Gateway and WAF v2, the logs show a little more information:

VALUE	DESCRIPTION
instanceId	Application Gateway instance that served the request.
clientIP	Originating IP for the request.
clientPort	Originating port for the request.
httpMethod	HTTP method used by the request.
requestUri	URI of the received request.
UserAgent	User agent from the HTTP request header.
httpStatus	HTTP status code returned to the client from Application Gateway.
httpVersion	HTTP version of the request.
receivedBytes	Size of packet received, in bytes.
sentBytes	Size of packet sent, in bytes.
timeTaken	Length of time (in milliseconds) that it takes for a request to be processed and its response to be sent. This is calculated as the interval from the time when Application Gateway receives the first byte of an HTTP request to the time when the response send operation finishes. It's important to note that the Time-Taken field usually includes the time that the request and response packets are traveling over the network.

VALUE	DESCRIPTION
sslEnabled	Whether communication to the back-end pools used SSL. Valid values are on and off.
sslCipher	Cipher suite being used for SSL communication (if SSL is enabled).
sslProtocol	SSL protocol being used (if SSL is enabled).
serverRouted	The backend server that application gateway routes the request to.
serverStatus	HTTP status code of the backend server.
serverResponseLatency	Latency of the response from the backend server.
host	Address listed in the host header of the request.

```
{
    "resourceId":
"/SUBSCRIPTIONS/{subscriptionId}/RESOURCEGROUPS/PEERINGTEST/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/{ap
plicationGatewayName}",
    "operationName": "ApplicationGatewayAccess",
    "time": "2017-04-26T19:27:38Z",
    "category": "ApplicationGatewayAccessLog",
    "properties": {
        "instanceId": "appgw_1",
        "clientIP": "191.96.249.97",
        "clientPort": 46886,
        "httpMethod": "GET",
        "requestUri": "/phpmyadmin/scripts/setup.php",
        "userAgent": "-",
        "httpStatus": 404,
        "httpVersion": "HTTP/1.0",
        "receivedBytes": 65,
        "sentBytes": 553,
        "timeTaken": 205,
        "sslEnabled": "off",
        "sslCipher": "",
        "sslProtocol": "",
        "serverRouted": "104.41.114.59:80",
        "serverStatus": "200",
        "serverResponseLatency": "0.023",
        "host": "www.contoso.com",
    }
}
```

Performance log

The performance log is generated only if you have enabled it on each Application Gateway instance, as detailed in the preceding steps. The data is stored in the storage account that you specified when you enabled the logging. The

performance log data is generated in 1-minute intervals. It is available only for the v1 SKU. For the v2 SKU, use Metrics for performance data. The following data is logged:

VALUE	DESCRIPTION
instanceId	Application Gateway instance for which performance data is being generated. For a multiple-instance application gateway, there is one row per instance.
healthyHostCount	Number of healthy hosts in the back-end pool.
unHealthyHostCount	Number of unhealthy hosts in the back-end pool.
requestCount	Number of requests served.
latency	Average latency (in milliseconds) of requests from the instance to the back end that serves the requests.
failedRequestCount	Number of failed requests.
throughput	Average throughput since the last log, measured in bytes per second.

```
{
    "resourceId":
"/SUBSCRIPTIONS/{subscriptionId}/RESOURCEGROUPS/{resourceGroupName}/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATE
WAYS/{applicationGatewayName}",
    "operationName": "ApplicationGatewayPerformance",
    "time": "2016-04-09T00:00:00Z",
    "category": "ApplicationGatewayPerformanceLog",
    "properties":
    {
        "instanceId":"ApplicationGatewayRole_IN_1",
        "healthyHostCount":"4",
        "unHealthyHostCount":"0",
        "requestCount":"185",
        "latency":"0",
        "failedRequestCount":"0",
        "throughput":"119427"
    }
}
```

> **NOTE**
>
> Latency is calculated from the time when the first byte of the HTTP request is received to the time when the last byte of the HTTP response is sent. It's the sum of the Application Gateway processing time plus the network cost to the back end, plus the time that the back end takes to process the request.

Firewall log

The firewall log is generated only if you have enabled it for each application gateway, as detailed in the preceding steps. This log also requires that the web application firewall is configured on an application gateway. The data is stored in the storage account that you specified when you enabled the logging. The following data is logged:

VALUE	DESCRIPTION

VALUE	DESCRIPTION
instanceId	Application Gateway instance for which firewall data is being generated. For a multiple-instance application gateway, there is one row per instance.
clientIp	Originating IP for the request.
clientPort	Originating port for the request.
requestUri	URL of the received request.
ruleSetType	Rule set type. The available value is OWASP.
ruleSetVersion	Rule set version used. Available values are 2.2.9 and 3.0.
ruleId	Rule ID of the triggering event.
message	User-friendly message for the triggering event. More details are provided in the details section.
action	Action taken on the request. Available values are Blocked and Allowed.
site	Site for which the log was generated. Currently, only Global is listed because rules are global.
details	Details of the triggering event.
details.message	Description of the rule.
details.data	Specific data found in request that matched the rule.
details.file	Configuration file that contained the rule.
details.line	Line number in the configuration file that triggered the event.
hostname	Hostname or IP address of the Application Gateway.
transactionId	Unique ID for a given transaction which helps group multiple rule violations that occurred within the same request.
policyId	Unique ID of the Firewall Policy associated with the Application Gateway, Listener, or Path.
policyScope	The location of the policy - values can be "Global", "Listener", or "Location".
policyScopeName	The name of the object where the policy is applied.

```
{
  "resourceId":
"/SUBSCRIPTIONS/{subscriptionId}/RESOURCEGROUPS/{resourceGroupName}/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATE
WAYS/{applicationGatewayName}",
  "operationName": "ApplicationGatewayFirewall",
  "time": "2017-03-20T15:52:09.1494499Z",
  "category": "ApplicationGatewayFirewallLog",
  "properties": {
      "instanceId": "ApplicationGatewayRole_IN_0",
      "clientIp": "52.161.109.147",
      "clientPort": "0",
      "requestUri": "/",
      "ruleSetType": "OWASP",
      "ruleSetVersion": "3.0",
      "ruleId": "920350",
      "ruleGroup": "920-PROTOCOL-ENFORCEMENT",
      "message": "Host header is a numeric IP address",
      "action": "Matched",
      "site": "Global",
      "details": {
        "message": "Warning. Pattern match \"^[\\\\d.:]+$\" at REQUEST_HEADERS:Host ....",
        "data": "127.0.0.1",
        "file": "rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf",
        "line": "791"
      },
      "hostname": "127.0.0.1",
      "transactionId": "16861477007022634343",
      "policyId": "/subscriptions/1496a758-b2ff-43ef-b738-
8e9eb5161a86/resourceGroups/drewRG/providers/Microsoft.Network/ApplicationGatewayWebApplicationFirewallPolicies
/perListener",
      "policyScope": "Listener",
      "policyScopeName": "httpListener1"
    }
  }
}
```


View and analyze the activity log

You can view and analyze activity log data by using any of the following methods:

- **Azure tools**: Retrieve information from the activity log through Azure PowerShell, the Azure CLI, the Azure REST API, or the Azure portal. Step-by-step instructions for each method are detailed in the Activity operations with Resource Manager article.
- **Power BI**: If you don't already have a Power BI account, you can try it for free. By using the Power BI template apps, you can analyze your data.

View and analyze the access, performance, and firewall logs

Azure Monitor logs can collect the counter and event log files from your Blob storage account. It includes visualizations and powerful search capabilities to analyze your logs.

You can also connect to your storage account and retrieve the JSON log entries for access and performance logs. After you download the JSON files, you can convert them to CSV and view them in Excel, Power BI, or any other data-visualization tool.

> **TIP**
>
> If you are familiar with Visual Studio and basic concepts of changing values for constants and variables in C#, you can use the log converter tools available from GitHub.

Analyzing Access logs through GoAccess

We have published a Resource Manager template that installs and runs the popular GoAccess log analyzer for Application Gateway Access Logs. GoAccess provides valuable HTTP traffic statistics such as Unique Visitors, Requested Files, Hosts, Operating Systems, Browsers, HTTP Status codes and more. For more details, please see the Readme file in the Resource Manager template folder in GitHub.

Next steps

- Visualize counter and event logs by using Azure Monitor logs.
- Visualize your Azure activity log with Power BI blog post.
- View and analyze Azure activity logs in Power BI and more blog post.

Troubleshoot Web Application Firewall (WAF) for Azure Application Gateway

11/13/2019 • 10 minutes to read • Edit Online

There are a few things you can do if requests that should pass through your Web Application Firewall (WAF) are blocked.

First, ensure you've read the WAF overview and the WAF configuration documents. Also, make sure you've enabled WAF monitoring These articles explain how the WAF functions, how the WAF rule sets work, and how to access WAF logs.

Understanding WAF logs

The purpose of WAF logs is to show every request that is matched or blocked by the WAF. It is a ledger of all evaluated requests that are matched or blocked. If you notice that the WAF blocks a request that it shouldn't (a false positive), you can do a few things. First, narrow down, and find the specific request. Look through the logs to find the specific URI, timestamp, or transaction ID of the request. When you find the associated log entries, you can begin to act on the false positives.

For example, say you have a legitimate traffic containing the string *1=1* that you want to pass through your WAF. If you try the request, the WAF blocks traffic that contains your *1=1* string in any parameter or field. This is a string often associated with a SQL injection attack. You can look through the logs and see the timestamp of the request and the rules that blocked/matched.

In the following example, you can see that four rules are triggered during the same request (using the TransactionId field). The first one says it matched because the user used a numeric/IP URL for the request, which increases the anomaly score by three since it's a warning. The next rule that matched is 942130, which is the one you're looking for. You can see the *1=1* in the `details.data` field. This further increases the anomaly score by three again, as it's also a warning. Generally, every rule that has the action **Matched** increases the anomaly score, and at this point the anomaly score would be six. For more information, see Anomaly scoring mode.

The final two log entries show the request was blocked because the anomaly score was high enough. These entries have a different action than the other two. They show they actually *blocked* the request. These rules are mandatory and can't be disabled. They shouldn't be thought of as rules, but more as core infrastructure of the WAF internals.


```
{
    "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
    "operationName": "ApplicationGatewayFirewall",
    "category": "ApplicationGatewayFirewallLog",
    "properties": {
        "instanceId": "appgw_3",
        "clientIp": "167.220.2.139",
        "clientPort": "",
        "requestUri": "\/",
        "ruleSetType": "OWASP_CRS",
        "ruleSetVersion": "3.0.0",
        "ruleId": "920350",
        "message": "Host header is a numeric IP address",
        "action": "Matched",
        "site": "Global",
        "details": {
            "message": "Warning. Pattern match \\\"^[\\\\\\\\\\d.:]+$\\\" at REQUEST_HEADERS:Host. ",
            "data": "40.90.218.160",
```

```
            "file": "rules\/REQUEST-920-PROTOCOL-ENFORCEMENT.conf\\\"",
            "line": "791"
        },
        "hostname": "vm000003",
        "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
    }
}
{
    "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
    "operationName": "ApplicationGatewayFirewall",
    "category": "ApplicationGatewayFirewallLog",
    "properties": {
        "instanceId": "appgw_3",
        "clientIp": "167.220.2.139",
        "clientPort": "",
        "requestUri": "\/",
        "ruleSetType": "OWASP_CRS",
        "ruleSetVersion": "3.0.0",
        "ruleId": "942130",
        "message": "SQL Injection Attack: SQL Tautology Detected.",
        "action": "Matched",
        "site": "Global",
        "details": {
            "message": "Warning. Pattern match \\\"(?i:([\\\\\\\\\\s'\\\\\\\"`\\\\\\\\\(\\\\\\\\)]*?)
([\\\\\\\\d\\\\\\\\w]++)([\\\\\\\\\s'\\\\\\\"`\\\\\\\\\(\\\\\\\\)]*?)(?:(?:=|\\u003c=\\u003e|r?
like|sounds\\\\\\\\s+like|regexp)([\\\\\\\\\s'\\\\\\\"`\\\\\\\\\(\\\\\\\\)]*?)\\\\\\\\2|
(?:!=|\\u003c=|\\u003e=|\\u003c\\u003e|\\u003c|\\u003e|\\\\\\\\^|is\\\\\\\\s+not|not\\\\\\\\s+like|not\\\\\\\\s
+regexp)([\\\\\\\\\s'\\\\\\\"`\\\\\\\\\(\\\\\\\\)]*?)(?!\\\\\\\\2)([\\\\\\\\d\\\\\\\w]+)))\\\" at ARGS:text1. ",
            "data": "Matched Data: 1=1 found within ARGS:text1: 1=1",
            "file": "rules\/REQUEST-942-APPLICATION-ATTACK-SQLI.conf\\\"",
            "line": "554"
        },
        "hostname": "vm000003",
        "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
    }
}
{
    "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
    "operationName": "ApplicationGatewayFirewall",
    "category": "ApplicationGatewayFirewallLog",
    "properties": {
        "instanceId": "appgw_3",
        "clientIp": "167.220.2.139",
        "clientPort": "",
        "requestUri": "\/",
        "ruleSetType": "",
        "ruleSetVersion": "",
        "ruleId": "0",
        "message": "Mandatory rule. Cannot be disabled. Inbound Anomaly Score Exceeded (Total Score: 8)",
        "action": "Blocked",
        "site": "Global",
        "details": {
            "message": "Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. ",
            "data": "",
            "file": "rules\/REQUEST-949-BLOCKING-EVALUATION.conf\\\"",
            "line": "57"
        },
        "hostname": "vm000003",
        "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
    }
}
{
    "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
    "operationName": "ApplicationGatewayFirewall",
    "category": "ApplicationGatewayFirewallLog",
    "properties": {
```

```
        "instanceId": "appgw_3",
        "clientIp": "167.220.2.139",
        "clientPort": "",
        "requestUri": "\/",
        "ruleSetType": "",
        "ruleSetVersion": "",
        "ruleId": "0",
        "message": "Mandatory rule. Cannot be disabled. Inbound Anomaly Score Exceeded (Total Inbound Score: 8
- SQLI=5,XSS=0,RFI=0,LFI=0,RCE=0,PHPI=0,HTTP=0,SESS=0): SQL Injection Attack: SQL Tautology Detected.",
        "action": "Blocked",
        "site": "Global",
        "details": {
            "message": "Warning. Operator GE matched 5 at TX:inbound_anomaly_score. ",
            "data": "",
            "file": "rules\/RESPONSE-980-CORRELATION.conf\\\"",
            "line": "73"
        },
        "hostname": "vm000003",
        "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
    }
}
```


Fixing false positives

With this information, and the knowledge that rule 942130 is the one that matched the *1=1* string, you can do a few things to stop this from blocking your traffic:

- Use an Exclusion List

  See WAF configuration for more information about exclusion lists.

- Disable the rule.

Using an exclusion list

To make an informed decision about handling a false positive, it's important to familiarize yourself with the technologies your application uses. For example, say there isn't a SQL server in your technology stack, and you are getting false positives related to those rules. Disabling those rules doesn't necessarily weaken your security.

One benefit of using an exclusion list is that only a specific part of a request is being disabled. However, this means that a specific exclusion is applicable to all traffic passing through your WAF because it is a global setting. For example, this could lead to an issue if *1=1* is a valid request in the body for a certain app, but not for others. Another benefit is that you can choose between body, headers, and cookies to be excluded if a certain condition is met, as opposed to excluding the whole request.

Occasionally, there are cases where specific parameters get passed into the WAF in a manner that may not be intuitive. For example, there is a token that gets passed when authenticating using Azure Active Directory. This token, *__RequestVerificationToken*, usually get passed in as a Request Cookie. However, in some cases where cookies are disabled, this token is also passed as a request attribute or "arg". If this happens, you need to ensure that *__RequestVerificationToken* is added to the exclusion list as a **Request attribute name** as well.

FIELD	OPERATOR	SELECTOR
Request cookie name	Equals	__RequestVerificationToken
Request attribute name	Equals	__RequestVerificationToken

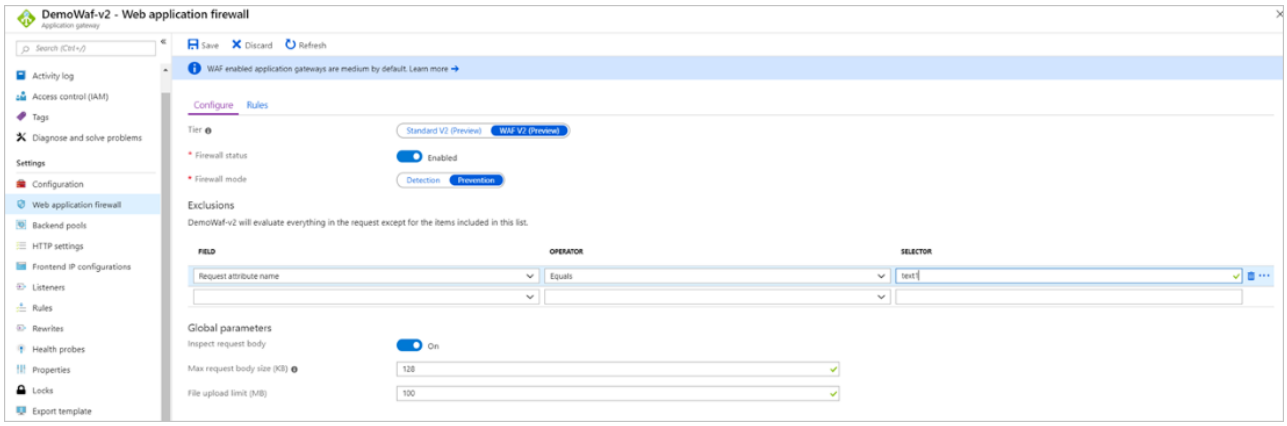
Field = Request attribute name

Operator = Equals

Selector = __RequestVerificationToken

In this example, you want to exclude the **Request attribute name** that equals *text1*. This is apparent because you

can see the attribute name in the firewall logs: **data: Matched Data: 1=1 found within ARGS:text1: 1=1**. The attribute is **text1**. You can also find this attribute name a few other ways, see Finding request attribute names.

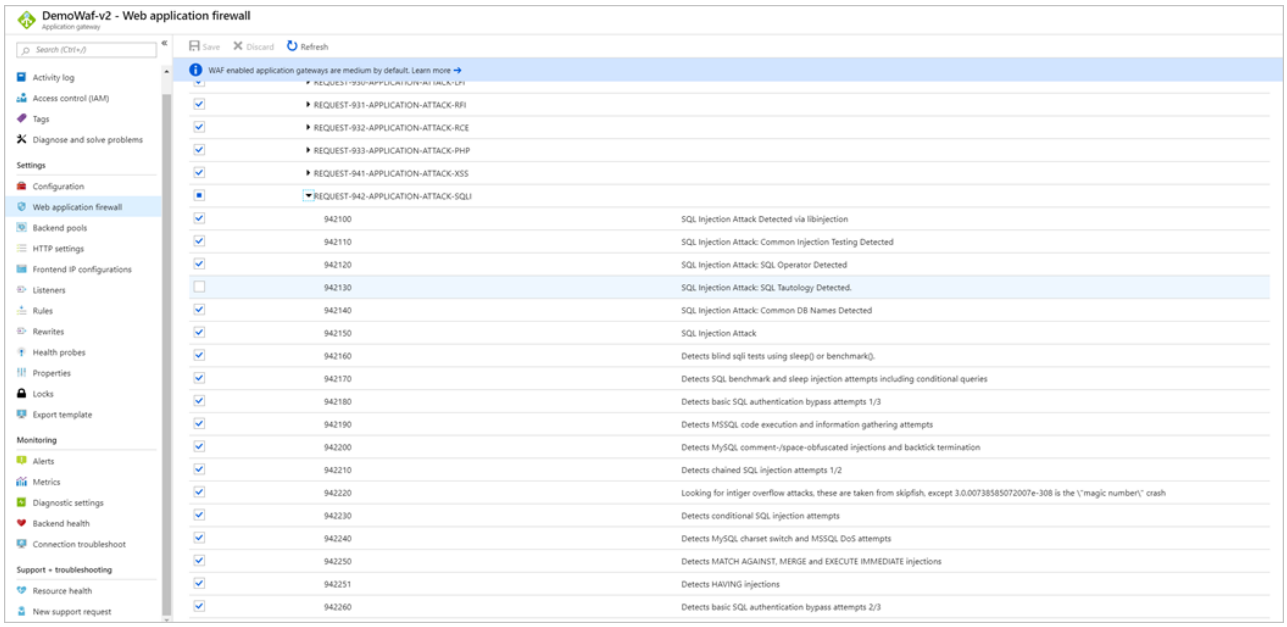


Disabling rules

Another way to get around a false positive is to disable the rule that matched on the input the WAF thought was malicious. Since you've parsed the WAF logs and have narrowed the rule down to 942130, you can disable it in the Azure portal. See Customize web application firewall rules through the Azure portal.

One benefit of disabling a rule is that if you know all traffic that contains a certain condition that will normally be blocked is valid traffic, you can disable that rule for the entire WAF. However, if it's only valid traffic in a specific use case, you open up a vulnerability by disabling that rule for the entire WAF since it is a global setting.

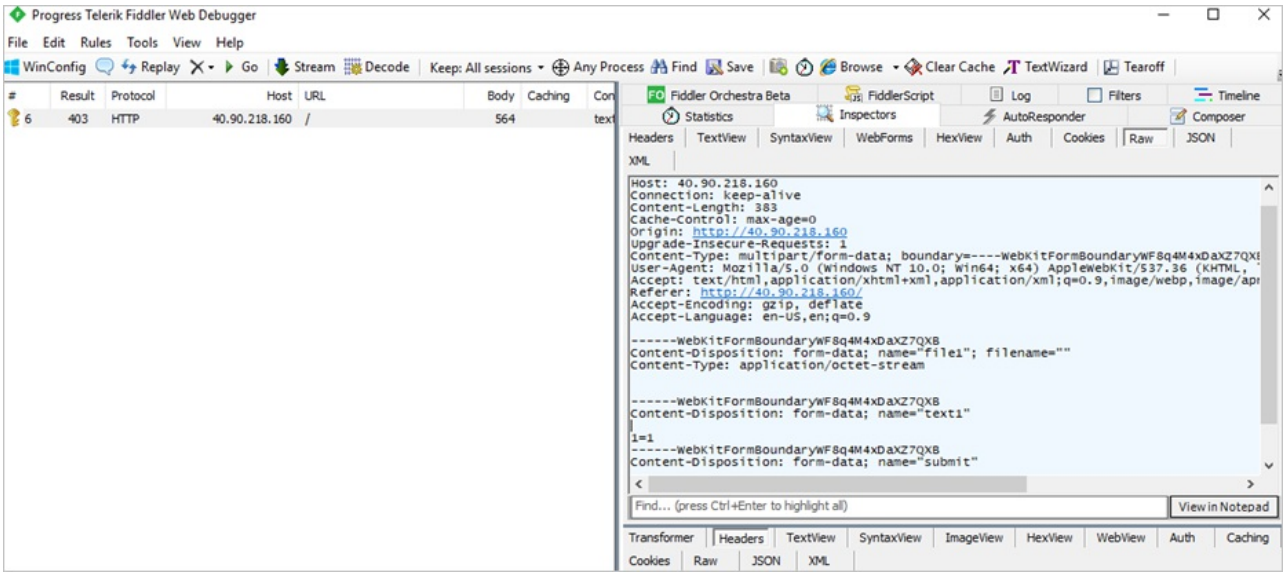
If you want to use Azure PowerShell, see Customize web application firewall rules through PowerShell. If you want to use Azure CLI, see Customize web application firewall rules through the Azure CLI.



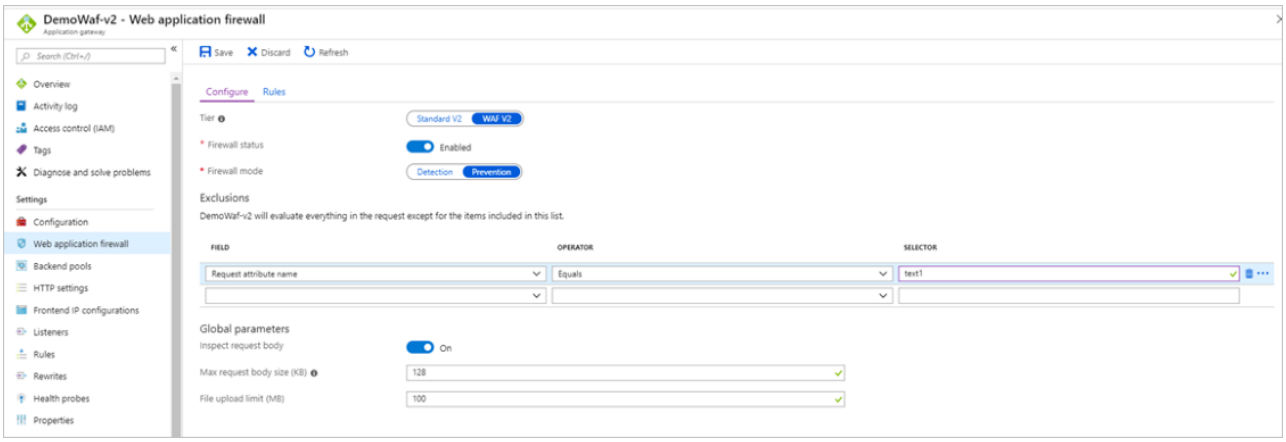
Finding request attribute names

With the help of Fiddler, you inspect individual requests and determine what specific fields of a web page are called. This can help to exclude certain fields from inspection using Exclusion Lists.

In this example, you can see that the field where the *1=1* string was entered is called **text1**.



This is a field you can exclude. To learn more about exclusion lists, See Web application firewall request size limits and exclusion lists. You can exclude the evaluation in this case by configuring the following exclusion:



You can also examine the firewall logs to get the information to see what you need to add to the exclusion list. To enable logging, see Back-end health, diagnostic logs, and metrics for Application Gateway.

Examine the firewall log and view the PT1H.json file for the hour that the request you want to inspect occurred.

In this example, you can see that you have four rules with the same TransactionID, and that they all occurred at the exact same time:


```
- {
-     "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
-     "operationName": "ApplicationGatewayFirewall",
-     "category": "ApplicationGatewayFirewallLog",
-     "properties": {
-         "instanceId": "appgw_3",
-         "clientIp": "167.220.2.139",
-         "clientPort": "",
-         "requestUri": "\/",
-         "ruleSetType": "OWASP_CRS",
-         "ruleSetVersion": "3.0.0",
-         "ruleId": "920350",
-         "message": "Host header is a numeric IP address",
-         "action": "Matched",
-         "site": "Global",
-         "details": {
-             "message": "Warning. Pattern match \\\"^[\\\\\\\\\d.:]+$\\\" at REQUEST_HEADERS:Host. ",
-             "data": "40.90.218.160",
-             "file": "rules\/REQUEST-920-PROTOCOL-ENFORCEMENT.conf\\\"",
-             "line": "791"
```

```
-         },
-         "hostname": "vm000003",
-         "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
-     }
- }
- {
-     "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
-     "operationName": "ApplicationGatewayFirewall",
-     "category": "ApplicationGatewayFirewallLog",
-     "properties": {
-         "instanceId": "appgw_3",
-         "clientIp": "167.220.2.139",
-         "clientPort": "",
-         "requestUri": "\/",
-         "ruleSetType": "OWASP_CRS",
-         "ruleSetVersion": "3.0.0",
-         "ruleId": "942130",
-         "message": "SQL Injection Attack: SQL Tautology Detected.",
-         "action": "Matched",
-         "site": "Global",
-         "details": {
-             "message": "Warning. Pattern match \\\"(?i:([\\\\\\\\\s'\\\\\\\"`\\\\\\\\(\\\\\\\\)]*?)
([\\\\\\\\d\\\\\\\\w]++)([\\\\\\\\s'\\\\\\\"`\\\\\\\\(\\\\\\\\)]*?)(?:(?:=|\\u003c=\\u003e|r?
like|sounds\\\\\\\\s+like|regexp)([\\\\\\\\s'\\\\\\\"`\\\\\\\\(\\\\\\\\)]*?)\\\\\\\\2|
(?:!=|\\u003c=|\\u003e=|\\u003c\\u003e|\\u003c|\\u003e|\\\\\\\\^|is\\\\\\\\s+not|not\\\\\\\\s+like|not\\\\\\\\s
+regexp)([\\\\\\\\s'\\\\\\\"`\\\\\\\\(\\\\\\\\)]*?)(?!\\\\\\\\2)([\\\\\\\\d\\\\\\\\w]+)))\\\" at ARGS:text1. ",
-             "data": "Matched Data: 1=1 found within ARGS:text1: 1=1",
-             "file": "rules\/REQUEST-942-APPLICATION-ATTACK-SQLI.conf\\\"",
-             "line": "554"
-         },
-         "hostname": "vm000003",
-         "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
-     }
- }
- {
-     "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
-     "operationName": "ApplicationGatewayFirewall",
-     "category": "ApplicationGatewayFirewallLog",
-     "properties": {
-         "instanceId": "appgw_3",
-         "clientIp": "167.220.2.139",
-         "clientPort": "",
-         "requestUri": "\/",
-         "ruleSetType": "",
-         "ruleSetVersion": "",
-         "ruleId": "0",
-         "message": "Mandatory rule. Cannot be disabled. Inbound Anomaly Score Exceeded (Total Score: 8)",
-         "action": "Blocked",
-         "site": "Global",
-         "details": {
-             "message": "Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. ",
-             "data": "",
-             "file": "rules\/REQUEST-949-BLOCKING-EVALUATION.conf\\\"",
-             "line": "57"
-         },
-         "hostname": "vm000003",
-         "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
-     }
- }
- {
-     "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
-     "operationName": "ApplicationGatewayFirewall",
-     "category": "ApplicationGatewayFirewallLog",
-     "properties": {
-         "instanceId": "appgw_3",
-         "clientIp": "167.220.2.139",
```

```
-         "clientIp": "167.220.2.139",
-         "clientPort": "",
-         "requestUri": "\/",
-         "ruleSetType": "",
-         "ruleSetVersion": "",
-         "ruleId": "0",
-         "message": "Mandatory rule. Cannot be disabled. Inbound Anomaly Score Exceeded (Total Inbound Score:
8 - SQLI=5,XSS=0,RFI=0,LFI=0,RCE=0,PHPI=0,HTTP=0,SESS=0): SQL Injection Attack: SQL Tautology Detected.",
-         "action": "Blocked",
-         "site": "Global",
-         "details": {
-             "message": "Warning. Operator GE matched 5 at TX:inbound_anomaly_score. ",
-             "data": "",
-             "file": "rules\/RESPONSE-980-CORRELATION.conf\\\"",
-             "line": "73"
-         },
-         "hostname": "vm000003",
-         "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
-     }
- }
```

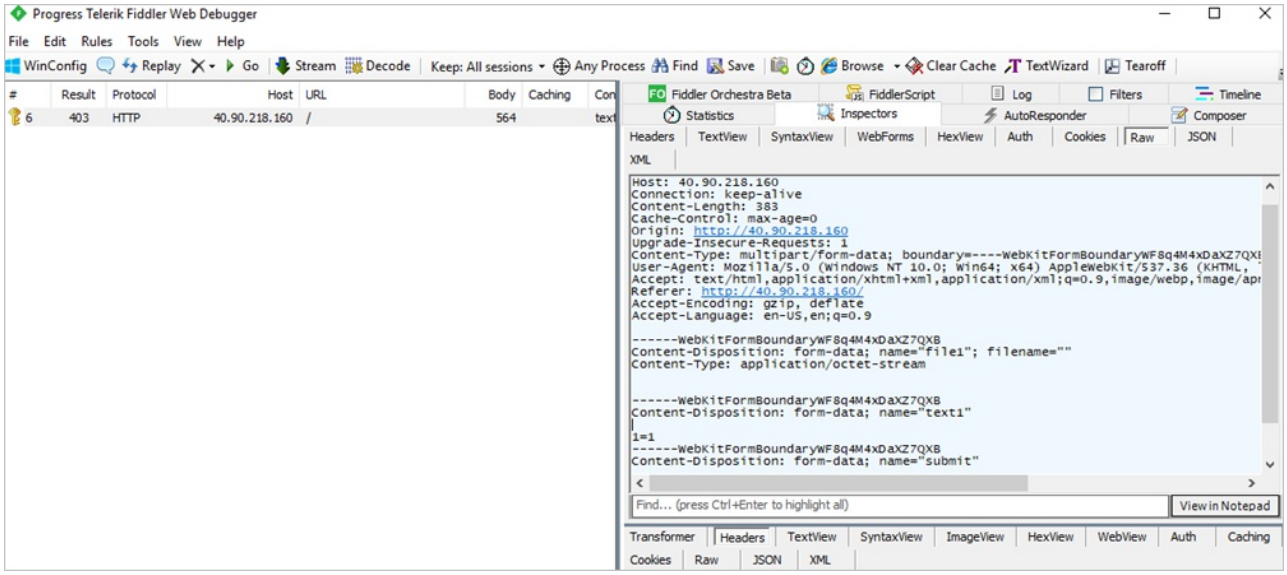

With your knowledge of how the CRS rule sets work, and that the CRS ruleset 3.0 works with an anomaly scoring system (see Web Application Firewall for Azure Application Gateway) you know that the bottom two rules with the **action: Blocked** property are blocking based on the total anomaly score. The rules to focus on are the top two.

The first entry is logged because the user used a numeric IP address to navigate to the Application Gateway, which can be ignored in this case.

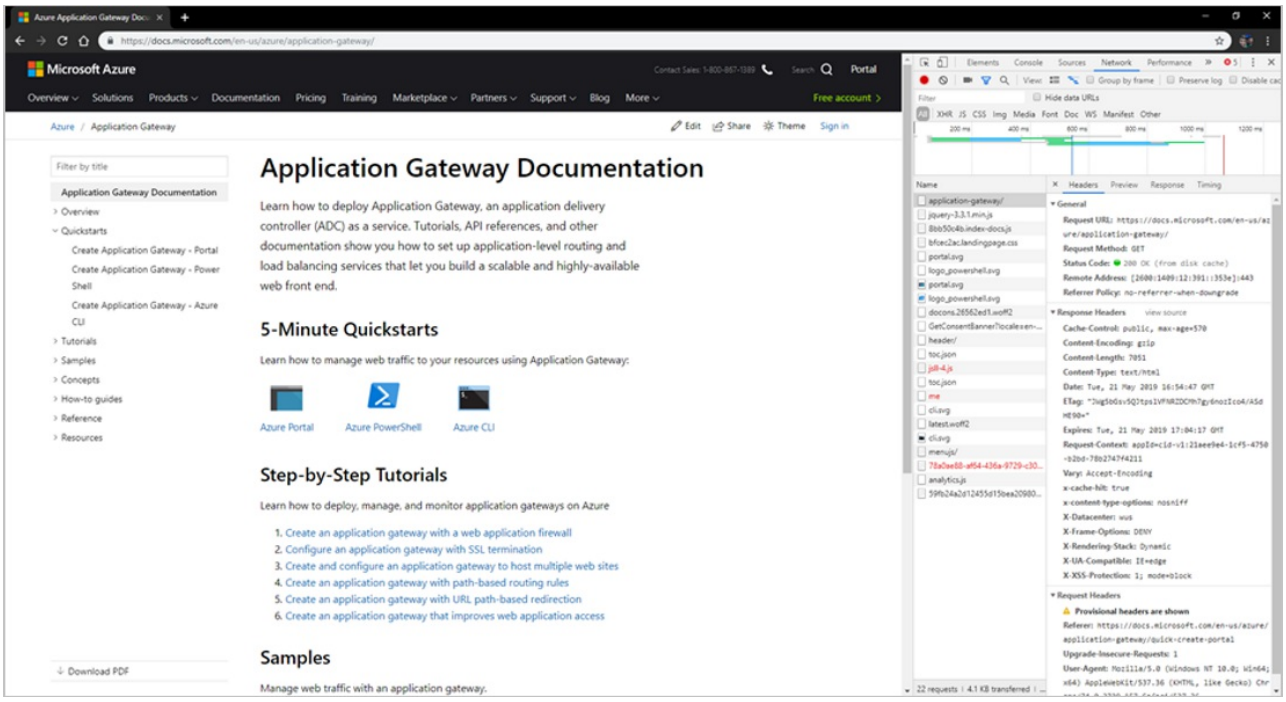
The second one (rule 942130) is the interesting one. You can see in the details that it matched a pattern (1=1), and the field is named **text1**. Follow the same previous steps to exclude the **Request Attribute Name** that **equals 1=1**.

Finding request header names

Fiddler is a useful tool once again to find request header names. In the following screenshot, you can see the headers for this GET request, which include *Content-Type*, *User-Agent*, and so on.



Another way to view request and response headers is to look inside the developer tools of Chrome. You can press F12 or right-click -> **Inspect** -> **Developer Tools**, and select the **Network** tab. Load a web page, and click the request you want to inspect.



Finding request cookie names

If the request contains cookies, the **Cookies** tab can be selected to view them in Fiddler.

Restrict global parameters to eliminate false positives

- Disable request body inspection

  By setting **Inspect request body** to off, the request bodies of all traffic will not be evaluated by your WAF. This may be useful if you know that the request bodies aren't malicious to your application.

  By disabling this option, only the request body is not inspected. The headers and cookies remain inspected, unless individual ones are excluded using the exclusion list functionality.

- File size limits

  By limiting the file size for your WAF, you're limiting the possibility of an attack happening to your web servers. By allowing large files to be uploaded, the risk of your backend being overwhelmed increases. Limiting the file size to a normal use case for your application is just another way to prevent attacks.

  > **NOTE**
  > If you know that your app will never need any file upload above a given size, you can restrict that by setting a limit.

Firewall Metrics (WAF_v1 only)

For v1 Web Application Firewalls, the following metrics are now available in the portal:

1. Web Application Firewall Blocked Request Count The number of requests that were blocked
2. Web Application Firewall Blocked Rule Count All rules that were matched **and** the request was blocked
3. Web Application Firewall Total Rule Distribution All rules that were matched during evaluation

To enable metrics, select the **Metrics** tab in the portal, and select one of the three metrics.

Next steps

See How to configure web application firewall on Application Gateway.

Configure a Web Application Firewall policy using Azure PowerShell

11/19/2019 • 2 minutes to read • Edit Online

Azure Web Application Firewall (WAF) policy defines inspections required when a request arrives at Front Door. This article shows how to configure a WAF policy that consists of some custom rules and with Azure-managed Default Rule Set enabled.

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

Before you begin to set up a rate limit policy, set up your PowerShell environment and create a Front Door profile.

Set up your PowerShell environment

Azure PowerShell provides a set of cmdlets that use the Azure Resource Manager model for managing your Azure resources.

You can install Azure PowerShell on your local machine and use it in any PowerShell session. Follow the instructions on the page, to sign in with your Azure credentials, and install Az PowerShell module.

Sign in to Azure

```
Connect-AzAccount
```

Before install Front Door module, make sure you have the current version of PowerShellGet installed. Run below command and reopen PowerShell.

```
Install-Module PowerShellGet -Force -AllowClobber
```

Install Az.FrontDoor module

```
Install-Module -Name Az.FrontDoor
```

Create a Front Door profile

Create a Front Door profile by following the instructions described in Quickstart: Create a Front Door profile

Custom rule based on http parameters

The following example shows how to configure a custom rule with two match conditions using New-AzFrontDoorWafMatchConditionObject. Requests are from a specified site as defined by referrer, and query string doesn't contain "password".

```
$referer = New-AzFrontDoorWafMatchConditionObject -MatchVariable RequestHeader -OperatorProperty Equal -
Selector "Referer" -MatchValue "www.mytrustedsites.com/referpage.html"
$password = New-AzFrontDoorWafMatchConditionObject -MatchVariable QueryString -OperatorProperty Contains -
MatchValue "password"
$AllowFromTrustedSites = New-AzFrontDoorWafCustomRuleObject -Name "AllowFromTrustedSites" -RuleType MatchRule
-MatchCondition $referer,$password -Action Allow -Priority 1
```

Custom rule based on http request method

Create a rule blocking "PUT" method using New-AzFrontDoorWafCustomRuleObject as follows:

```
$put = New-AzFrontDoorWafMatchConditionObject -MatchVariable RequestMethod -OperatorProperty Equal -MatchValue
PUT
$BlockPUT = New-AzFrontDoorWafCustomRuleObject -Name "BlockPUT" -RuleType MatchRule -MatchCondition $put -
Action Block -Priority 2
```

Create a custom rule based on size constraint

The following example creates a rule blocking requests with Url that is longer than 100 characters using Azure PowerShell:

```
$url = New-AzFrontDoorWafMatchConditionObject -MatchVariable RequestUri -OperatorProperty GreaterThanOrEqual -
MatchValue 100
$URLOver100 = New-AzFrontDoorWafCustomRuleObject -Name "URLOver100" -RuleType MatchRule -MatchCondition $url -
Action Block -Priority 3
```

Add managed Default Rule Set

The following example creates a managed Default Rule Set using Azure PowerShell:

```
$managedRules =  New-AzFrontDoorWafManagedRuleObject -Type DefaultRuleSet -Version 1.0
```

Configure a security policy

Find the name of the resource group that contains the Front Door profile using `Get-AzResourceGroup`. Next, configure a security policy with created rules in the previous steps using New-AzFrontDoorWafPolicy in the specified resource group that contains the Front Door profile.

```
$myWAFPolicy=New-AzFrontDoorWafPolicy -Name $policyName -ResourceGroupName $resourceGroupName -Customrule
$AllowFromTrustedSites,$BlockPUT,$URLOver100 -ManagedRule $managedRules -EnabledState Enabled -Mode Prevention
```

Link policy to a Front Door front-end host

Link the security policy object to an existing Front Door front-end host and update Front Door properties. First, retrieve the Front Door object using Get-AzFrontDoor. Next, set the front-end *WebApplicationFirewallPolicyLink* property to the *resourceId* of the "$myWAFPolicy$" created in the previous step using Set-AzFrontDoor.

The below example uses the Resource Group name *myResourceGroupFD1* with the assumption that you've created the Front Door profile using instructions provided in the Quickstart: Create a Front Door article. Also, in the below example, replace $frontDoorName with the name of your Front Door profile.

```
$FrontDoorObjectExample = Get-AzFrontDoor `
  -ResourceGroupName myResourceGroupFD1 `
  -Name $frontDoorName
$FrontDoorObjectExample[0].FrontendEndpoints[0].WebApplicationFirewallPolicyLink = $myWAFPolicy.Id
Set-AzFrontDoor -InputObject $FrontDoorObjectExample[0]
```

NOTE

You only need to set *WebApplicationFirewallPolicyLink* property once to link a security policy to a Front Door front-end. Subsequent policy updates are automatically applied to the front-end.

Next steps

- Learn more about Front Door
- Learn more about WAF with Front Door

Configure bot protection for Web Application Firewall (Preview)

2/1/2020 • 2 minutes to read • Edit Online

This article shows you how to configure bot protection rule in Azure Web Application Firewall (WAF) for Front Door by using Azure portal. Bot protection rule can also be configured using CLI, Azure PowerShell, or Azure Resource Manager template.

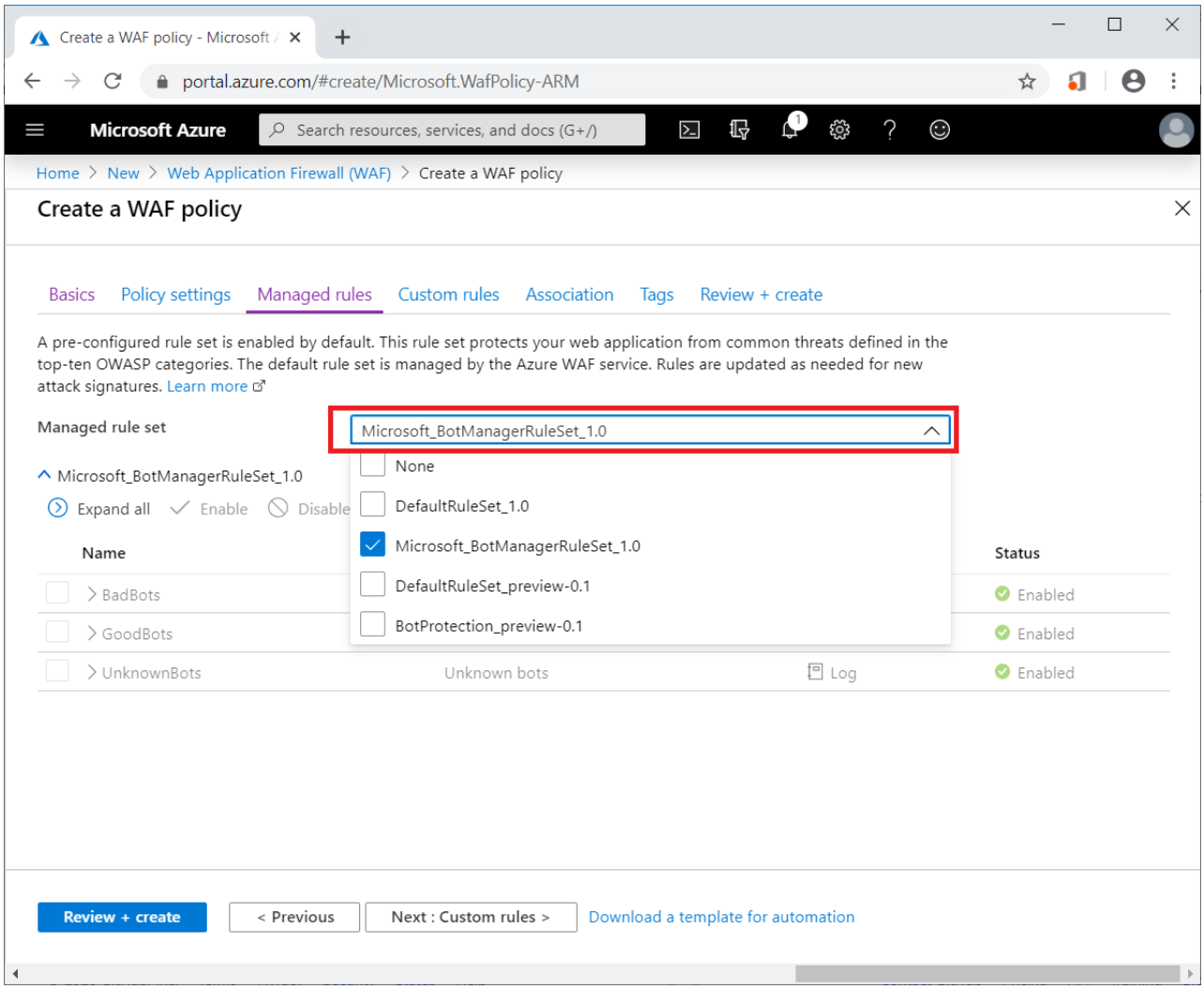
> **IMPORTANT**
>
> Bot protection rule set is currently in public preview and is provided with a preview service level agreement. Certain features may not be supported or may have constrained capabilities. See the Supplemental Terms of Use for Microsoft Azure Previews for details.

Prerequisites

Create a basic WAF policy for Front Door by following the instructions described in Create a WAF policy for Azure Front Door by using the Azure portal.

Enable bot protection rule set

In the **Managed Rules** page when creating a Web Application Firewall policy, first find **Managed rule set** section, select the check box in front of the rule **Microsoft_BotManager_1.0** from the drop-down menu, and then select **Review + Create**.



Next steps

- Learn how to monitor WAF.

Configure a custom response for Azure Web Application Firewall

11/19/2019 • 2 minutes to read • Edit Online

By default, when Azure Web Application Firewall (WAF) with Azure Front Door blocks a request because of a matched rule, it returns a 403 status code with **The request is blocked** message. This article describes how to configure a custom response status code and response message when a request is blocked by WAF.

Set up your PowerShell environment

Azure PowerShell provides a set of cmdlets that use the Azure Resource Manager model for managing your Azure resources.

You can install Azure PowerShell on your local machine and use it in any PowerShell session. Follow the instructions on the page, to sign in with your Azure credentials, and install the Az PowerShell module.

Connect to Azure with an interactive dialog for sign-in

```
Connect-AzAccount
Install-Module -Name Az
```

Make sure you have the current version of PowerShellGet installed. Run below command and reopen PowerShell.

```
Install-Module PowerShellGet -Force -AllowClobber
```

Install Az.FrontDoor module

```
Install-Module -Name Az.FrontDoor
```

Create a resource group

In Azure, you allocate related resources to a resource group. In this example, you create a resource group by using New-AzResourceGroup.

```
New-AzResourceGroup -Name myResourceGroupWAF
```

Create a new WAF policy with custom response

Below is an example of creating a new WAF policy with custom response status code set to 405 and message to **You are blocked.** using New-AzFrontDoorWafPolicy.

```
# WAF policy setting
New-AzFrontDoorWafPolicy `
-Name myWAFPolicy `
-ResourceGroupName myResourceGroupWAF `
-EnabledState enabled `
-Mode Detection `
-CustomBlockResponseStatusCode 405 `
-CustomBlockResponseBody "<html><head><title>You are blocked.</title></head><body></body></html>"
```

Modify custom response code or response body settings of an existing WAF policy, using Update-AzFrontDoorFireWallPolicy.

```
# modify WAF response code
Update-AzFrontDoorFireWallPolicy `
-Name myWAFPolicy `
-ResourceGroupName myResourceGroupWAF `
-EnabledState enabled `
-Mode Detection `
-CustomBlockResponseStatusCode 403
```

```
# modify WAF response body
Update-AzFrontDoorFireWallPolicy `
-Name myWAFPolicy `
-ResourceGroupName myResourceGroupWAF `
-CustomBlockResponseBody "<html><head><title> Forbidden</title></head><body></body></html>"
```

Next steps

- Learn more about Web Application Firewall with Azure Front Door

Configure an IP restriction rule with a Web Application Firewall for Azure Front Door Service

1/30/2020 • 5 minutes to read • Edit Online

This article shows you how to configure IP restriction rules in a Web Application Firewall (WAF) for Azure Front Door Service by using the Azure CLI, Azure PowerShell, or an Azure Resource Manager template.

An IP address–based access control rule is a custom WAF rule that lets you control access to your web applications. It does this by specifying a list of IP addresses or IP address ranges in Classless Inter-Domain Routing (CIDR) format.

By default, your web application is accessible from the internet. If you want to limit access to clients from a list of known IP addresses or IP address ranges, you may create an IP matching rule that contains the list of IP addresses as matching values and sets operator to "Not" (negate is true) and the action to **Block**. After an IP restriction rule is applied, requests that originate from addresses outside this allowed list receive a 403 Forbidden response.

Configure a WAF policy with the Azure CLI

Prerequisites

Before you begin to configure an IP restriction policy, set up your CLI environment and create an Azure Front Door Service profile.

Set up the Azure CLI environment

1. Install the Azure CLI, or use Azure Cloud Shell. Azure Cloud Shell is a free Bash shell that you can run directly within the Azure portal. It has the Azure CLI preinstalled and configured to use with your account. Select the **Try it** button in the CLI commands that follow, and then sign in to your Azure account in the Cloud Shell session that opens. After the session starts, enter `az extension add --name front-door` to add the Azure Front Door Service extension.
2. If you're using the CLI locally in Bash, sign in to Azure by using `az login`.

Create an Azure Front Door Service profile

Create an Azure Front Door Service profile by following the instructions described in Quickstart: Create a Front Door for a highly available global web application.

Create a WAF policy

Create a WAF policy by using the az network front-door waf-policy create command. In the example that follows, replace the policy name *IPAllowPolicyExampleCLI* with a unique policy name.

```
az network front-door waf-policy create \
  --resource-group <resource-group-name> \
  --subscription <subscription ID> \
  --name IPAllowPolicyExampleCLI
```

Add a custom IP access control rule

Use the az network front-door waf-policy custom-rule create command to add a custom IP access control rule for the WAF policy you just created.

In the following examples:

- Replace *IPAllowPolicyExampleCLI* with your unique policy created earlier.

- Replace *ip-address-range-1*, *ip-address-range-2* with your own range.

First, create an IP allow rule for the policy created from the previous step. Note **--defer** is required because a rule must have a match condition to be added in the next step.

```
az network front-door waf-policy rule create \
  --name IPAllowListRule \
  --priority 1 \
  --rule-type MatchRule \
  --action Block \
  --resource-group <resource-group-name> \
  --policy-name IPAllowPolicyExampleCLI --defer
```

Next, add match condition to the rule:

```
az network front-door waf-policy rule match-condition add \
--match-variable RemoteAddr \
--operator IPMatch \
--values "ip-address-range-1" "ip-address-range-2" \
--negate true \
--name IPAllowListRule \
  --resource-group <resource-group-name> \
  --policy-name IPAllowPolicyExampleCLI
```

Find the ID of a WAF policy

Find a WAF policy's ID by using the az network front-door waf-policy show command. Replace *IPAllowPolicyExampleCLI* in the following example with your unique policy that you created earlier.

```
az network front-door  waf-policy show \
  --resource-group <resource-group-name> \
  --name IPAllowPolicyExampleCLI
```

Link a WAF policy to an Azure Front Door Service front-end host

Set the Azure Front Door Service *WebApplicationFirewallPolicyLink* ID to the policy ID by using the az network front-door update command. Replace *IPAllowPolicyExampleCLI* with your unique policy that you created earlier.

```
az network front-door update \
  --set FrontendEndpoints[0].WebApplicationFirewallPolicyLink.id=/subscriptions/<subscription
ID>/resourcegroups/resource-group-
name/providers/Microsoft.Network/frontdoorwebapplicationfirewallpolicies/IPAllowPolicyExampleCLI \
  --name <frontdoor-name>
  --resource-group <resource-group-name>
```

In this example, the WAF policy is applied to **FrontendEndpoints[0]**. You can link the WAF policy to any of your front ends.

> **NOTE**
> You need to set the **WebApplicationFirewallPolicyLink** property only once to link a WAF policy to an Azure Front Door Service front end. Subsequent policy updates are automatically applied to the front end.

Configure a WAF policy with Azure PowerShell

Prerequisites

Before you begin to configure an IP restriction policy, set up your PowerShell environment and create an Azure Front Door Service profile.

Set up your PowerShell environment

Azure PowerShell provides a set of cmdlets that use the Azure Resource Manager model for managing Azure resources.

You can install Azure PowerShell on your local machine and use it in any PowerShell session. Follow the instructions on the page to sign in to PowerShell by using your Azure credentials, and then install the Az module.

1. Connect to Azure by using the following command, and then use an interactive dialog to sign in.

   ```
   Connect-AzAccount
   ```

2. Before you install an Azure Front Door Service module, make sure you have the current version of the PowerShellGet module installed. Run the following command, and then reopen PowerShell.

   ```
   Install-Module PowerShellGet -Force -AllowClobber
   ```

3. Install the Az.FrontDoor module by using the following command.

   ```
   Install-Module -Name Az.FrontDoor
   ```

Create an Azure Front Door Service profile

Create an Azure Front Door Service profile by following the instructions described in Quickstart: Create a Front Door for a highly available global web application.

Define an IP match condition

Use the New-AzFrontDoorWafMatchConditionObject command to define an IP match condition. In the following example, replace *ip-address-range-1*, *ip-address-range-2* with your own range.

```
$IPMatchCondition = New-AzFrontDoorWafMatchConditionObject `
-MatchVariable  RemoteAddr `
-OperatorProperty IPMatch `
-MatchValue "ip-address-range-1", "ip-address-range-2"
-NegateCondition 1
```

Create a custom IP allow rule

Use the New-AzFrontDoorWafCustomRuleObject command to define an action and set a priority. In the following example, requests not from client IPs that match the list will be blocked.

```
$IPAllowRule = New-AzFrontDoorWafCustomRuleObject `
-Name "IPAllowRule" `
-RuleType MatchRule `
-MatchCondition $IPMatchCondition `
-Action Block -Priority 1
```

Configure a WAF policy

Find the name of the resource group that contains the Azure Front Door Service profile by using `Get-AzResourceGroup`. Next, configure a WAF policy with the IP rule by using New-AzFrontDoorWafPolicy.

```
$IPAllowPolicyExamplePS = New-AzFrontDoorWafPolicy `
  -Name "IPRestrictionExamplePS" `
  -resourceGroupName <resource-group-name> `
  -Customrule $IPAllowRule`
  -Mode Prevention `
  -EnabledState Enabled
```

Link a WAF policy to an Azure Front Door Service front-end host

Link a WAF policy object to an existing front-end host and update Azure Front Door Service properties. First, retrieve the Azure Front Door Service object by using Get-AzFrontDoor. Next, set the **WebApplicationFirewallPolicyLink** property to the resource ID of *$IPAllowPolicyExamplePS*, created in the previous step, by using the Set-AzFrontDoor command.

```
$FrontDoorObjectExample = Get-AzFrontDoor `
  -ResourceGroupName <resource-group-name> `
  -Name $frontDoorName
$FrontDoorObjectExample[0].FrontendEndpoints[0].WebApplicationFirewallPolicyLink = $IPBlockPolicy.Id
Set-AzFrontDoor -InputObject $FrontDoorObjectExample[0]
```

> **NOTE**
>
> In this example, the WAF policy is applied to **FrontendEndpoints[0]**. You can link a WAF policy to any of your front ends. You need to set the **WebApplicationFirewallPolicyLink** property only once to link a WAF policy to an Azure Front Door Service front end. Subsequent policy updates are automatically applied to the front end.

Configure a WAF policy with a Resource Manager template

To view the template that creates an Azure Front Door Service policy and a WAF policy with custom IP restriction rules, go to GitHub.

Next steps

- Learn how to create an Azure Front Door Service profile.

Configure a Web Application Firewall rate limit rule using Azure PowerShell

2/26/2020 • 2 minutes to read • Edit Online

The Azure Web Application Firewall (WAF) rate limit rule for Azure Front Door controls the number of requests allowed from clients during a one-minute duration. This article shows how to configure a WAF rate limit rule that controls the number of requests allowed from clients to a web application that contains */promo* in the URL using Azure PowerShell.

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

Before you begin to set up a rate limit policy, set up your PowerShell environment and create a Front Door profile.

Set up your PowerShell environment

Azure PowerShell provides a set of cmdlets that use the Azure Resource Manager model for managing your Azure resources.

You can install Azure PowerShell on your local machine and use it in any PowerShell session. Follow the instructions on the page, to sign in with your Azure credentials, and install Az PowerShell module.

Connect to Azure with an interactive dialog for sign in

```
Connect-AzAccount
```

Before install Front Door module, make sure the current version of PowerShellGet is installed. Run the following command and reopen PowerShell.

```
Install-Module PowerShellGet -Force -AllowClobber
```

Install Az.FrontDoor module

```
Install-Module -Name Az.FrontDoor
```

Create a Front Door profile

Create a Front Door profile by following the instructions described in Quickstart: Create a Front Door profile

Define url match conditions

Define a URL match condition (URL contains /promo) using New-AzFrontDoorWafMatchConditionObject. The following example matches */promo* as the value of the *RequestUri* variable:

```
$promoMatchCondition = New-AzFrontDoorWafMatchConditionObject `
  -MatchVariable RequestUri `
  -OperatorProperty Contains `
  -MatchValue "/promo"
```

Create a custom rate limit rule

Set a rate limit using New-AzFrontDoorWafCustomRuleObject. In the following example, the limit is set to 1000. Requests from any client to the promo page exceeding 1000 during one minute are blocked until the next minute starts.

```
$promoRateLimitRule = New-AzFrontDoorWafCustomRuleObject `
  -Name "rateLimitRule" `
  -RuleType RateLimitRule `
  -MatchCondition $promoMatchCondition `
  -RateLimitThreshold 1000 `
  -Action Block -Priority 1
```

Configure a security policy

Find the name of the resource group that contains the Front Door profile using `Get-AzureRmResourceGroup`. Next, configure a security policy with a custom rate limit rule using New-AzFrontDoorWafPolicy in the specified resource group that contains the Front Door profile.

The below example uses the Resource Group name *myResourceGroupFD1* with the assumption that you've created the Front Door profile using instructions provided in the Quickstart: Create a Front Door article.

using New-AzFrontDoorWafPolicy.

```
$ratePolicy = New-AzFrontDoorWafPolicy `
  -Name "RateLimitPolicyExamplePS" `
  -resourceGroupName myResourceGroupFD1 `
  -Customrule $promoRateLimitRule `
  -Mode Prevention `
  -EnabledState Enabled
```

Link policy to a Front Door front-end host

Link the security policy object to an existing Front Door front-end host and update Front Door properties. First retrieve the Front Door object using Get-AzFrontDoor command. Next, set the front-end *WebApplicationFirewallPolicyLink* property to the *resourceId* of the "$ratePolicy" created in the previous step using Set-AzFrontDoor command.

The below example uses the Resource Group name *myResourceGroupFD1* with the assumption that you've created the Front Door profile using instructions provided in the Quickstart: Create a Front Door article. Also, in the below example, replace $frontDoorName with the name of your Front Door profile.

```
$FrontDoorObjectExample = Get-AzFrontDoor `
  -ResourceGroupName myResourceGroupFD1 `
  -Name $frontDoorName
$FrontDoorObjectExample[0].FrontendEndpoints[0].WebApplicationFirewallPolicyLink = $ratePolicy.Id
Set-AzFrontDoor -InputObject $FrontDoorObjectExample[0]
```

NOTE

You only need to set *WebApplicationFirewallPolicyLink* property once to link a security policy to a Front Door front-end. Subsequent policy updates are automatically applied to the front-end.

Next steps

- Learn more about Front Door.

Set up a geo-filtering WAF policy for your Front Door

11/4/2019 • 2 minutes to read • Edit Online

This tutorial shows how to use Azure PowerShell to create a sample geo-filtering policy and associate the policy with your existing Front Door frontend host. This sample geo-filtering policy will block requests from all other countries/regions except United States.

If you don't have an Azure subscription, create a free account now.

Prerequisites

Before you begin to set up a geo-filter policy, set up your PowerShell environment and create a Front Door profile.

Set up your PowerShell environment

Azure PowerShell provides a set of cmdlets that use the Azure Resource Manager model for managing your Azure resources.

You can install Azure PowerShell on your local machine and use it in any PowerShell session. Follow the instructions on the page, to sign in with your Azure credentials, and install the Az PowerShell module.

Connect to Azure with an interactive dialog for sign in

```
Install-Module -Name Az
Connect-AzAccount
```

Make sure you have the current version of PowerShellGet installed. Run below command and reopen PowerShell.

```
Install-Module PowerShellGet -Force -AllowClobber
```

Install Az.FrontDoor module

```
Install-Module -Name Az.FrontDoor
```

Create a Front Door profile

Create a Front Door profile by following the instructions described in Quickstart: Create a Front Door profile.

Define geo-filtering match condition

Create a sample match condition that selects requests not coming from "US" using New-AzFrontDoorWafMatchConditionObject on parameters when creating a match condition. Two letter country codes to country mapping are provided in What is geo-filtering on a domain for Azure Front Door?.

```
$nonUSGeoMatchCondition = New-AzFrontDoorWafMatchConditionObject `
-MatchVariable RemoteAddr `
-OperatorProperty GeoMatch `
-NegateCondition $true `
-MatchValue "US"
```

Add geo-filtering match condition to a rule with Action and Priority

Create a CustomRule object `nonUSBlockRule` based on the match condition, an Action, and a Priority using New-AzFrontDoorWafCustomRuleObject. A CustomRule can have multiple MatchCondition. In this example, Action is set to Block and Priority to 1, the highest priority.

```
$nonUSBlockRule = New-AzFrontDoorWafCustomRuleObject `
-Name "geoFilterRule" `
-RuleType MatchRule `
-MatchCondition $nonUSGeoMatchCondition `
-Action Block `
-Priority 1
```

Add rules to a policy

Find the name of the resource group that contains the Front Door profile using `Get-AzResourceGroup`. Next, create a `geoPolicy` policy object containing `nonUSBlockRule` using New-AzFrontDoorWafPolicy in the specified resource group that contains the Front Door profile. You must provide a unique name for the geo policy.

The following example uses the Resource Group name *myResourceGroupFD1* with the assumption that you've created the Front Door profile using instructions provided in the Quickstart: Create a Front Door article. In the below example, replace the policy name *geoPolicyAllowUSOnly* with a unique policy name.

```
$geoPolicy = New-AzFrontDoorWafPolicy `
-Name "geoPolicyAllowUSOnly" `
-resourceGroupName myResourceGroupFD1 `
-Customrule $nonUSBlockRule  `
-Mode Prevention `
-EnabledState Enabled
```

Link WAF policy to a Front Door frontend host

Link the WAF policy object to the existing Front Door frontend host and update Front Door properties.

To do so, first retrieve your Front Door object using Get-AzFrontDoor.

```
$geoFrontDoorObjectExample = Get-AzFrontDoor -ResourceGroupName myResourceGroupFD1
$geoFrontDoorObjectExample[0].FrontendEndpoints[0].WebApplicationFirewallPolicyLink = $geoPolicy.Id
```

Next, set the frontend WebApplicationFirewallPolicyLink property to the resourceId of the `geoPolicy` using Set-AzFrontDoor.

```
Set-AzFrontDoor -InputObject $geoFrontDoorObjectExample[0]
```

> **NOTE**
> You only need to set WebApplicationFirewallPolicyLink property once to link a WAF policy to a Front Door frontend host. Subsequent policy updates are automatically applied to the frontend host.

Next steps

- Learn about Azure web application firewall.

- Learn how to create a Front Door.

Azure Web Application Firewall monitoring and logging

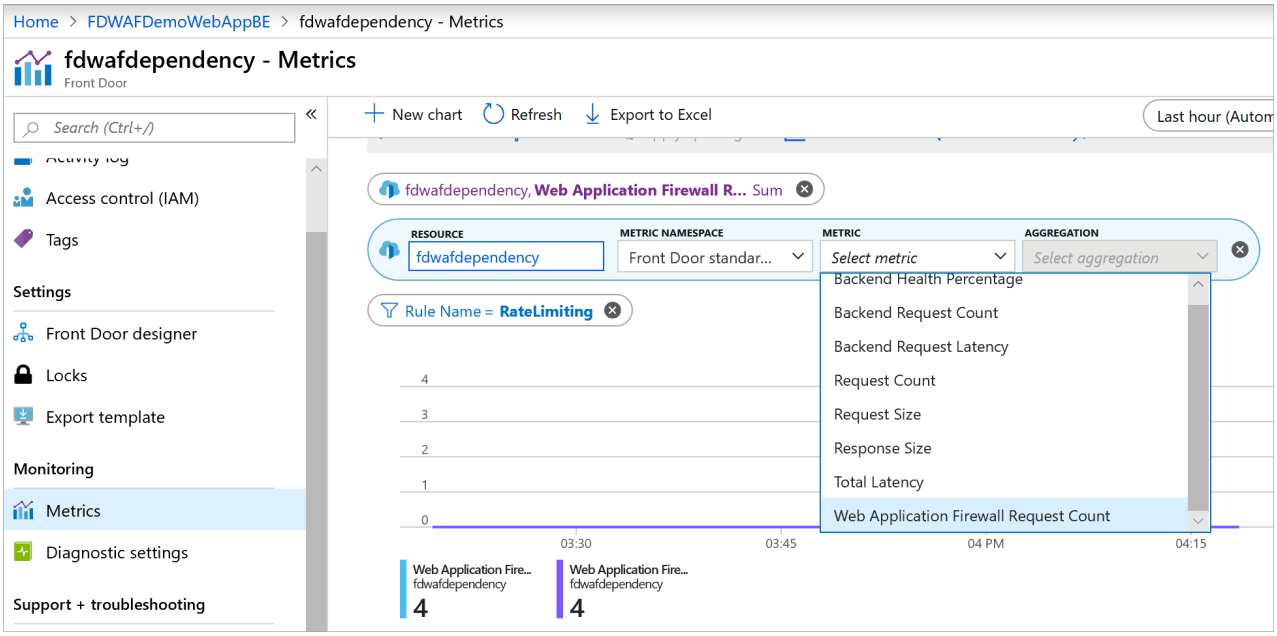
2/1/2020 • 2 minutes to read • Edit Online

Azure Web Application Firewall (WAF) monitoring and logging are provided through logging and integration with Azure Monitor and Azure Monitor logs.

Azure Monitor

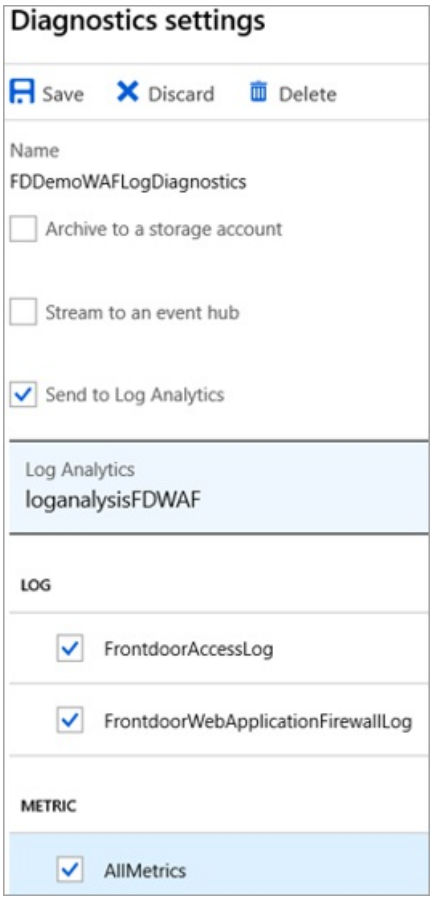
WAF with FrontDoor log is integrated with Azure Monitor. Azure Monitor allows you to track diagnostic information including WAF alerts and logs. You can configure WAF monitoring within the Front Door resource in the portal under the **Diagnostics** tab or through the Azure Monitor service directly.

From Azure portal, go to Front Door resource type. From **Monitoring**/**Metrics** tab on the left, you can add **WebApplicationFirewallRequestCount** to track number of requests that match WAF rules. Custom filters can be created based on action types and rule names.



Logs and diagnostics

WAF with Front Door provides detailed reporting on each threat it detects. Logging is integrated with Azure Diagnostics logs and alerts are recorded in a json format. These logs can be integrated with Azure Monitor logs.



FrontdoorAccessLog logs all requests that are forwarded to customer back-ends.
FrontdoorWebApplicationFirewallLog logs any request that matches a WAF rule.

The following example query obtains WAF logs on blocked requests:

```
AzureDiagnostics
| where ResourceType == "FRONTDOORS" and Category == "FrontdoorWebApplicationFirewallLog"
| where action_s == "Block"
```

Here is an example of a logged request in WAF log:

```
{
    "PreciseTimeStamp": "2020-01-25T00:11:19.3866091Z",
    "time": "2020-01-25T00:11:19.3866091Z",
    "category": "FrontdoorWebApplicationFirewallLog",
    "operationName": "Microsoft.Network/FrontDoor/WebApplicationFirewallLog/Write",
    "properties": {
        "clientIP": "xx.xx.xxx.xxx",
        "socketIP": "xx.xx.xxx.xxx",
        "requestUri": "https://wafdemofrontdoorwebapp.azurefd.net:443/?q=../../x",
        "ruleName": "Microsoft_DefaultRuleSet-1.1-LFI-930100",
        "policy": "WafDemoCustomPolicy",
        "action": "Block",
        "host": "wafdemofrontdoorwebapp.azurefd.net",
        "refString":
"0p4crXgAAAABgMq5aIpu0T6AUfCYOroltV1NURURHRTA2MTMANjMxNTAwZDAtOTRiNS00YzIwLTljY2YtNjFhNzMyOWQyYTgy",
        "policyMode": "prevention"
    }
}
```

The following example query obtains AccessLogs entries:

```
AzureDiagnostics
| where ResourceType == "FRONTDOORS" and Category == "FrontdoorAccessLog"
```

Here is an example of a logged request in Access log:


```
{
    "PreciseTimeStamp": "2020-01-25T00:11:12.0160150Z",
    "time": "2020-01-25T00:11:12.0160150Z",
    "category": "FrontdoorAccessLog",
    "operationName": "Microsoft.Network/FrontDoor/AccessLog/Write",
    "properties": {
        "trackingReference":
"0n4crXgAAAACnRKbdALbyToAqNfSHssDvV1NURURHRTA2MTMANjMxNTAwZDAtOTRiNS00YzIwLTljY2YtNjFhNzMyOWQyYTgy",
        "httpMethod": "GET",
        "httpVersion": "2.0",
        "requestUri": "https://wafdemofrontdoorwebapp.azurefd.net:443/",
        "requestBytes": "710",
        "responseBytes": "3116",
        "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/81.0.4017.0 Safari/537.36 Edg/81.0.389.2",
        "clientIp": "xx.xx.xxx.xxx",
        "timeTaken": "0.598",
        "securityProtocol": "TLS 1.2",
        "routingRuleName": "WAFdemoWebAppRouting",
        "backendHostname": "wafdemouksouth.azurewebsites.net:443",
        "sentToOriginShield": false,
        "httpStatusCode": "200",
        "httpStatusDetails": "200"
    }
}
```


Next steps

- Learn more about Front Door.

Troubleshoot Web Application Firewall (WAF) for Azure Application Gateway

11/13/2019 • 10 minutes to read • Edit Online

There are a few things you can do if requests that should pass through your Web Application Firewall (WAF) are blocked.

First, ensure you've read the WAF overview and the WAF configuration documents. Also, make sure you've enabled WAF monitoring These articles explain how the WAF functions, how the WAF rule sets work, and how to access WAF logs.

Understanding WAF logs

The purpose of WAF logs is to show every request that is matched or blocked by the WAF. It is a ledger of all evaluated requests that are matched or blocked. If you notice that the WAF blocks a request that it shouldn't (a false positive), you can do a few things. First, narrow down, and find the specific request. Look through the logs to find the specific URI, timestamp, or transaction ID of the request. When you find the associated log entries, you can begin to act on the false positives.

For example, say you have a legitimate traffic containing the string *1=1* that you want to pass through your WAF. If you try the request, the WAF blocks traffic that contains your *1=1* string in any parameter or field. This is a string often associated with a SQL injection attack. You can look through the logs and see the timestamp of the request and the rules that blocked/matched.

In the following example, you can see that four rules are triggered during the same request (using the TransactionId field). The first one says it matched because the user used a numeric/IP URL for the request, which increases the anomaly score by three since it's a warning. The next rule that matched is 942130, which is the one you're looking for. You can see the *1=1* in the `details.data` field. This further increases the anomaly score by three again, as it's also a warning. Generally, every rule that has the action **Matched** increases the anomaly score, and at this point the anomaly score would be six. For more information, see Anomaly scoring mode.

The final two log entries show the request was blocked because the anomaly score was high enough. These entries have a different action than the other two. They show they actually *blocked* the request. These rules are mandatory and can't be disabled. They shouldn't be thought of as rules, but more as core infrastructure of the WAF internals.


```
{
    "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
    "operationName": "ApplicationGatewayFirewall",
    "category": "ApplicationGatewayFirewallLog",
    "properties": {
        "instanceId": "appgw_3",
        "clientIp": "167.220.2.139",
        "clientPort": "",
        "requestUri": "\/",
        "ruleSetType": "OWASP_CRS",
        "ruleSetVersion": "3.0.0",
        "ruleId": "920350",
        "message": "Host header is a numeric IP address",
        "action": "Matched",
        "site": "Global",
        "details": {
            "message": "Warning. Pattern match \\\"^[\\\\\\\\\\d.:]+$\\\" at REQUEST_HEADERS:Host. ",
            "data": "40.90.218.160",
```

```
            "file": "rules\/REQUEST-920-PROTOCOL-ENFORCEMENT.conf\\\"",
            "line": "791"
        },
        "hostname": "vm000003",
        "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
    }
}
{
    "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
    "operationName": "ApplicationGatewayFirewall",
    "category": "ApplicationGatewayFirewallLog",
    "properties": {
        "instanceId": "appgw_3",
        "clientIp": "167.220.2.139",
        "clientPort": "",
        "requestUri": "\/",
        "ruleSetType": "OWASP_CRS",
        "ruleSetVersion": "3.0.0",
        "ruleId": "942130",
        "message": "SQL Injection Attack: SQL Tautology Detected.",
        "action": "Matched",
        "site": "Global",
        "details": {
            "message": "Warning. Pattern match \\\"(?i:([\\\\\\\\\\s'\\\\\\\\\"`\\\\\\\\\\(\\\\\\\\\\)]*?)
([\\\\\\\\\\d\\\\\\\\\\w]++)([\\\\\\\\\\s'\\\\\\\\\"`\\\\\\\\\\(\\\\\\\\\\)]*?)(?:(?:=|\\u003c=\\u003e|r?
like|sounds\\\\\\\\\\s+like|regexp)([\\\\\\\\\\s'\\\\\\\\\"`\\\\\\\\\\(\\\\\\\\\\)]*?)\\\\\\\\\\2|
(?:!=|\\u003c=|\\u003e=|\\u003c\\u003e|\\u003c|\\u003e|\\\\\\\\\\^|is\\\\\\\\\\s+not|not\\\\\\\\\\s+like|not\\\\\\\\\\
s+regexp)([\\\\\\\\\\s'\\\\\\\\\"`\\\\\\\\\\(\\\\\\\\\\)]*?)(?!\\\\\\\\\\2)([\\\\\\\\\\d\\\\\\\\\\w]+)))\\\\\" at ARGS:text1.
",
            "data": "Matched Data: 1=1 found within ARGS:text1: 1=1",
            "file": "rules\/REQUEST-942-APPLICATION-ATTACK-SQLI.conf\\\"",
            "line": "554"
        },
        "hostname": "vm000003",
        "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
    }
}
{
    "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
    "operationName": "ApplicationGatewayFirewall",
    "category": "ApplicationGatewayFirewallLog",
    "properties": {
        "instanceId": "appgw_3",
        "clientIp": "167.220.2.139",
        "clientPort": "",
        "requestUri": "\/",
        "ruleSetType": "",
        "ruleSetVersion": "",
        "ruleId": "0",
        "message": "Mandatory rule. Cannot be disabled. Inbound Anomaly Score Exceeded (Total Score: 8)",
        "action": "Blocked",
        "site": "Global",
        "details": {
            "message": "Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. ",
            "data": "",
            "file": "rules\/REQUEST-949-BLOCKING-EVALUATION.conf\\\"",
            "line": "57"
        },
        "hostname": "vm000003",
        "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
    }
}
{
    "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
    "operationName": "ApplicationGatewayFirewall",
    "category": "ApplicationGatewayFirewallLog",
```

```
    "properties": {
        "instanceId": "appgw_3",
        "clientIp": "167.220.2.139",
        "clientPort": "",
        "requestUri": "\/",
        "ruleSetType": "",
        "ruleSetVersion": "",
        "ruleId": "0",
        "message": "Mandatory rule. Cannot be disabled. Inbound Anomaly Score Exceeded (Total Inbound Score: 8
- SQLI=5,XSS=0,RFI=0,LFI=0,RCE=0,PHPI=0,HTTP=0,SESS=0): SQL Injection Attack: SQL Tautology Detected.",
        "action": "Blocked",
        "site": "Global",
        "details": {
            "message": "Warning. Operator GE matched 5 at TX:inbound_anomaly_score. ",
            "data": "",
            "file": "rules\/RESPONSE-980-CORRELATION.conf\\\"",
            "line": "73"
        },
        "hostname": "vm000003",
        "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
    }
}
```


Fixing false positives

With this information, and the knowledge that rule 942130 is the one that matched the *1=1* string, you can do a few things to stop this from blocking your traffic:

- Use an Exclusion List

  See WAF configuration for more information about exclusion lists.

- Disable the rule.

Using an exclusion list

To make an informed decision about handling a false positive, it's important to familiarize yourself with the technologies your application uses. For example, say there isn't a SQL server in your technology stack, and you are getting false positives related to those rules. Disabling those rules doesn't necessarily weaken your security.

One benefit of using an exclusion list is that only a specific part of a request is being disabled. However, this means that a specific exclusion is applicable to all traffic passing through your WAF because it is a global setting. For example, this could lead to an issue if *1=1* is a valid request in the body for a certain app, but not for others. Another benefit is that you can choose between body, headers, and cookies to be excluded if a certain condition is met, as opposed to excluding the whole request.

Occasionally, there are cases where specific parameters get passed into the WAF in a manner that may not be intuitive. For example, there is a token that gets passed when authenticating using Azure Active Directory. This token, *__RequestVerificationToken*, usually get passed in as a Request Cookie. However, in some cases where cookies are disabled, this token is also passed as a request attribute or "arg". If this happens, you need to ensure that *__RequestVerificationToken* is added to the exclusion list as a **Request attribute name** as well.

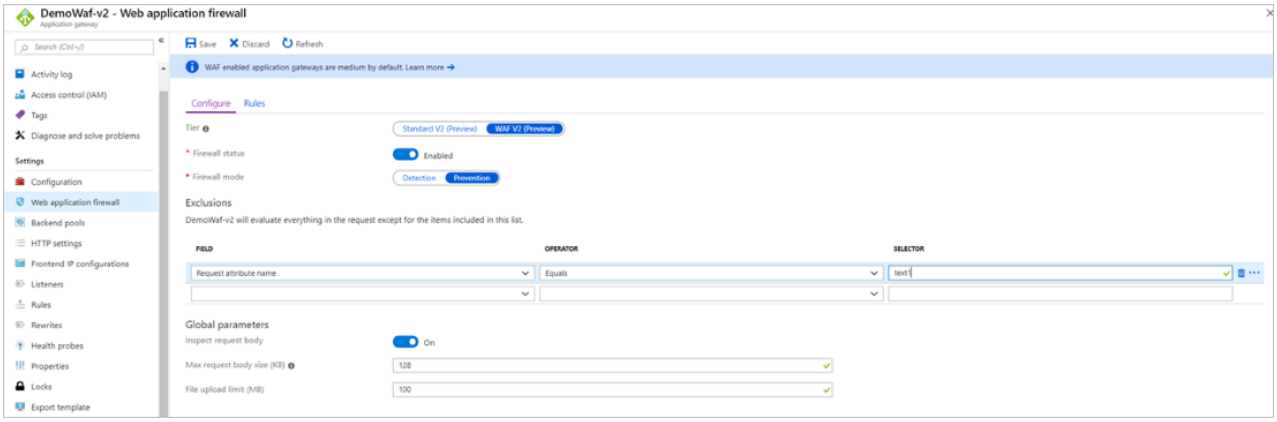
FIELD	OPERATOR	SELECTOR
Request cookie name	Equals	__RequestVerificationToken
Request attribute name	Equals	__RequestVerificationToken

Field = Request attribute name

Operator = Equals

Selector = __RequestVerificationToken

In this example, you want to exclude the **Request attribute name** that equals *text1*. This is apparent because you can see the attribute name in the firewall logs: **data: Matched Data: 1=1 found within ARGS:text1: 1=1**. The attribute is **text1**. You can also find this attribute name a few other ways, see Finding request attribute names.

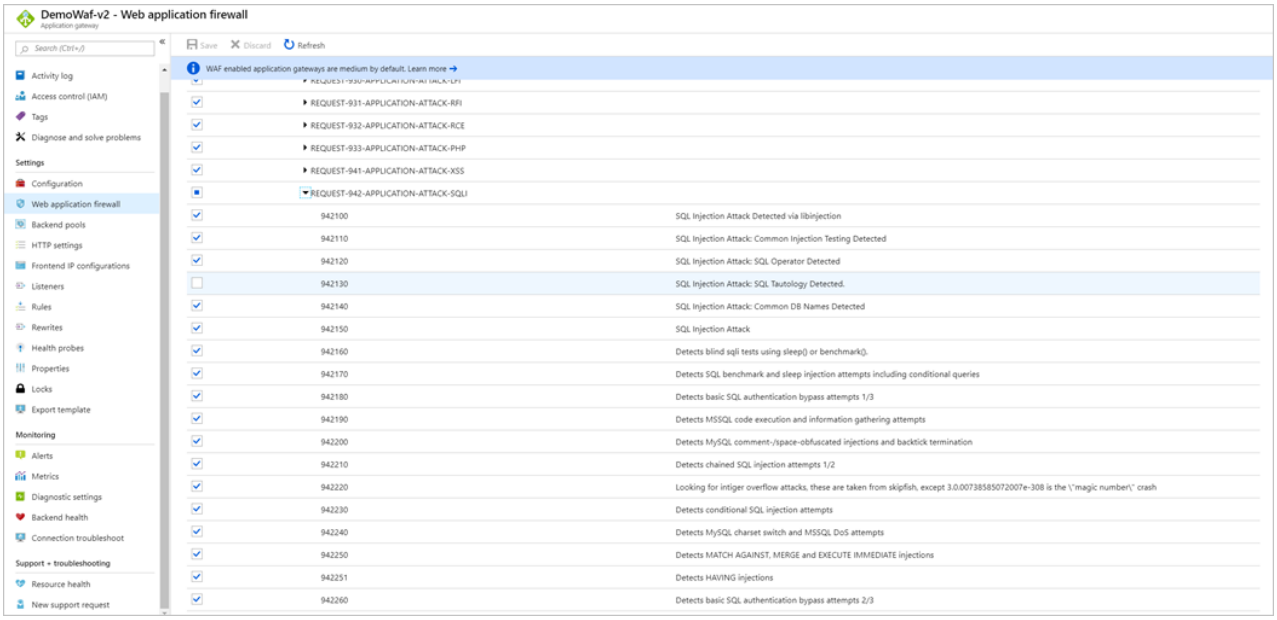


Disabling rules

Another way to get around a false positive is to disable the rule that matched on the input the WAF thought was malicious. Since you've parsed the WAF logs and have narrowed the rule down to 942130, you can disable it in the Azure portal. See Customize web application firewall rules through the Azure portal.

One benefit of disabling a rule is that if you know all traffic that contains a certain condition that will normally be blocked is valid traffic, you can disable that rule for the entire WAF. However, if it's only valid traffic in a specific use case, you open up a vulnerability by disabling that rule for the entire WAF since it is a global setting.

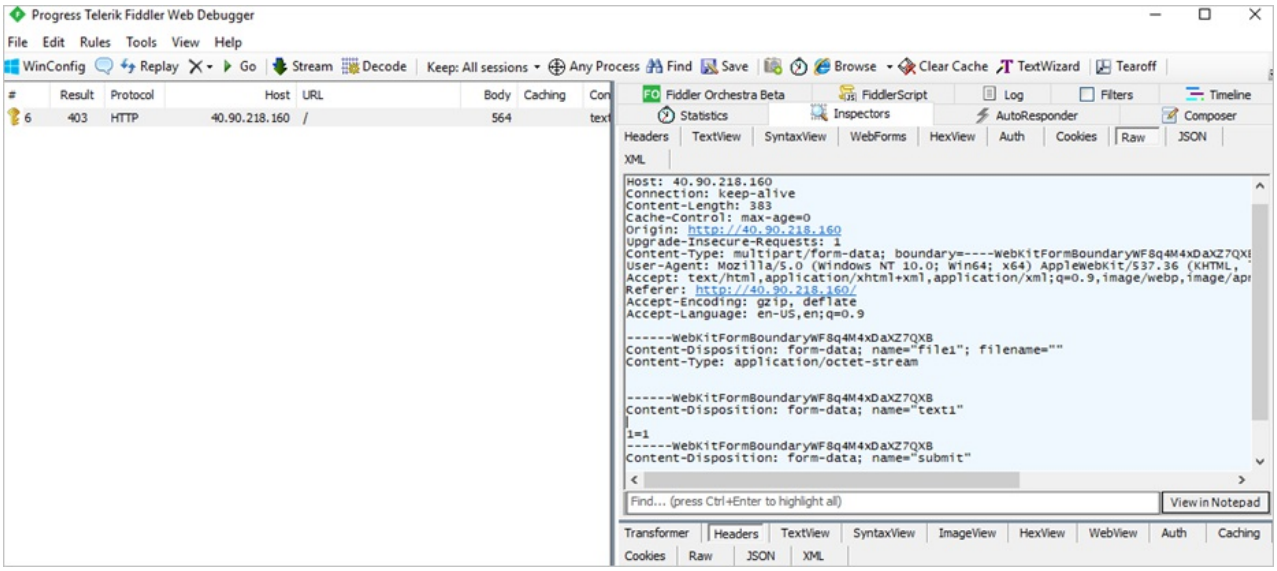
If you want to use Azure PowerShell, see Customize web application firewall rules through PowerShell. If you want to use Azure CLI, see Customize web application firewall rules through the Azure CLI.



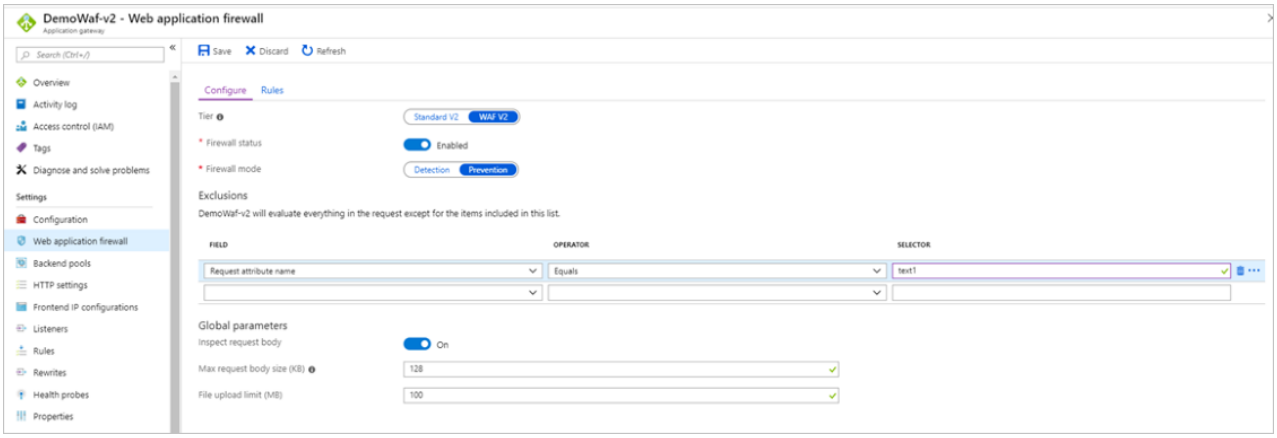
Finding request attribute names

With the help of Fiddler, you inspect individual requests and determine what specific fields of a web page are called. This can help to exclude certain fields from inspection using Exclusion Lists.

In this example, you can see that the field where the *1=1* string was entered is called **text1**.



This is a field you can exclude. To learn more about exclusion lists, See Web application firewall request size limits and exclusion lists. You can exclude the evaluation in this case by configuring the following exclusion:



You can also examine the firewall logs to get the information to see what you need to add to the exclusion list. To enable logging, see Back-end health, diagnostic logs, and metrics for Application Gateway.

Examine the firewall log and view the PT1H.json file for the hour that the request you want to inspect occurred.

In this example, you can see that you have four rules with the same TransactionID, and that they all occurred at the exact same time:


```
- {
-     "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
-     "operationName": "ApplicationGatewayFirewall",
-     "category": "ApplicationGatewayFirewallLog",
-     "properties": {
-         "instanceId": "appgw_3",
-         "clientIp": "167.220.2.139",
-         "clientPort": "",
-         "requestUri": "\/",
-         "ruleSetType": "OWASP_CRS",
-         "ruleSetVersion": "3.0.0",
-         "ruleId": "920350",
-         "message": "Host header is a numeric IP address",
-         "action": "Matched",
-         "site": "Global",
-         "details": {
-             "message": "Warning. Pattern match \\\"^[\\\\\\\\\\d.:]+$\\\" at REQUEST_HEADERS:Host. ",
-             "data": "40.90.218.160",
-             "file": "rules\/REQUEST-920-PROTOCOL-ENFORCEMENT.conf\\\"",
-             "line": "791"
```

```
-         },
-         "hostname": "vm000003",
-         "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
-     }
- }
- {
-     "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
-     "operationName": "ApplicationGatewayFirewall",
-     "category": "ApplicationGatewayFirewallLog",
-     "properties": {
-         "instanceId": "appgw_3",
-         "clientIp": "167.220.2.139",
-         "clientPort": "",
-         "requestUri": "\/",
-         "ruleSetType": "OWASP_CRS",
-         "ruleSetVersion": "3.0.0",
-         "ruleId": "942130",
-         "message": "SQL Injection Attack: SQL Tautology Detected.",
-         "action": "Matched",
-         "site": "Global",
-         "details": {
-             "message": "Warning. Pattern match \\\"(?i:([\\\\\\\\\s'\\\\\\\"`\\\\\\\\(\\\\\\\\)]*?)
([\\\\\\\\d\\\\\\\\w]++)([\\\\\\\\s'\\\\\\\"`\\\\\\\\(\\\\\\\\)]*?)(?:(?:=|\\u003c=\\u003e|r?
like|sounds\\\\\\\\s+like|regexp)([\\\\\\\\s'\\\\\\\"`\\\\\\\\(\\\\\\\\)]*?)\\\\\\\\2|
(?:!=|\\u003c=|\\u003e=|\\u003c\\u003e|\\u003c|\\u003e|\\\\\\\\^|is\\\\\\\\s+not|not\\\\\\\\s+like|not\\\\\\\\
s+regexp)([\\\\\\\\s'\\\\\\\"`\\\\\\\\(\\\\\\\\)]*?)(?!\\\\\\\\2)([\\\\\\\\d\\\\\\\\w]+)))\\\" at ARGS:text1.
",
-             "data": "Matched Data: 1=1 found within ARGS:text1: 1=1",
-             "file": "rules\/REQUEST-942-APPLICATION-ATTACK-SQLI.conf\\\"",
-             "line": "554"
-         },
-         "hostname": "vm000003",
-         "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
-     }
- }
- {
-     "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
-     "operationName": "ApplicationGatewayFirewall",
-     "category": "ApplicationGatewayFirewallLog",
-     "properties": {
-         "instanceId": "appgw_3",
-         "clientIp": "167.220.2.139",
-         "clientPort": "",
-         "requestUri": "\/",
-         "ruleSetType": "",
-         "ruleSetVersion": "",
-         "ruleId": "0",
-         "message": "Mandatory rule. Cannot be disabled. Inbound Anomaly Score Exceeded (Total Score: 8)",
-         "action": "Blocked",
-         "site": "Global",
-         "details": {
-             "message": "Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. ",
-             "data": "",
-             "file": "rules\/REQUEST-949-BLOCKING-EVALUATION.conf\\\"",
-             "line": "57"
-         },
-         "hostname": "vm000003",
-         "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
-     }
- }
- {
-     "resourceId": "/SUBSCRIPTIONS/A6F44B25-259E-4AF5-888A-
386FED92C11B/RESOURCEGROUPS/DEMOWAF_V2/PROVIDERS/MICROSOFT.NETWORK/APPLICATIONGATEWAYS/DEMOWAF-V2",
-     "operationName": "ApplicationGatewayFirewall",
-     "category": "ApplicationGatewayFirewallLog",
-     "properties": {
-         "instanceId": "appgw_3",
```

```
-         "clientIp": "167.220.2.139",
-         "clientPort": "",
-         "requestUri": "\/",
-         "ruleSetType": "",
-         "ruleSetVersion": "",
-         "ruleId": "0",
-         "message": "Mandatory rule. Cannot be disabled. Inbound Anomaly Score Exceeded (Total Inbound Score:
8 - SQLI=5,XSS=0,RFI=0,LFI=0,RCE=0,PHPI=0,HTTP=0,SESS=0): SQL Injection Attack: SQL Tautology Detected.",
-         "action": "Blocked",
-         "site": "Global",
-         "details": {
-             "message": "Warning. Operator GE matched 5 at TX:inbound_anomaly_score. ",
-             "data": "",
-             "file": "rules\/RESPONSE-980-CORRELATION.conf\\\"",
-             "line": "73"
-         },
-         "hostname": "vm000003",
-         "transactionId": "AcAcAcAcAKH@AcAcAcAcAyAt"
-     }
- }
```

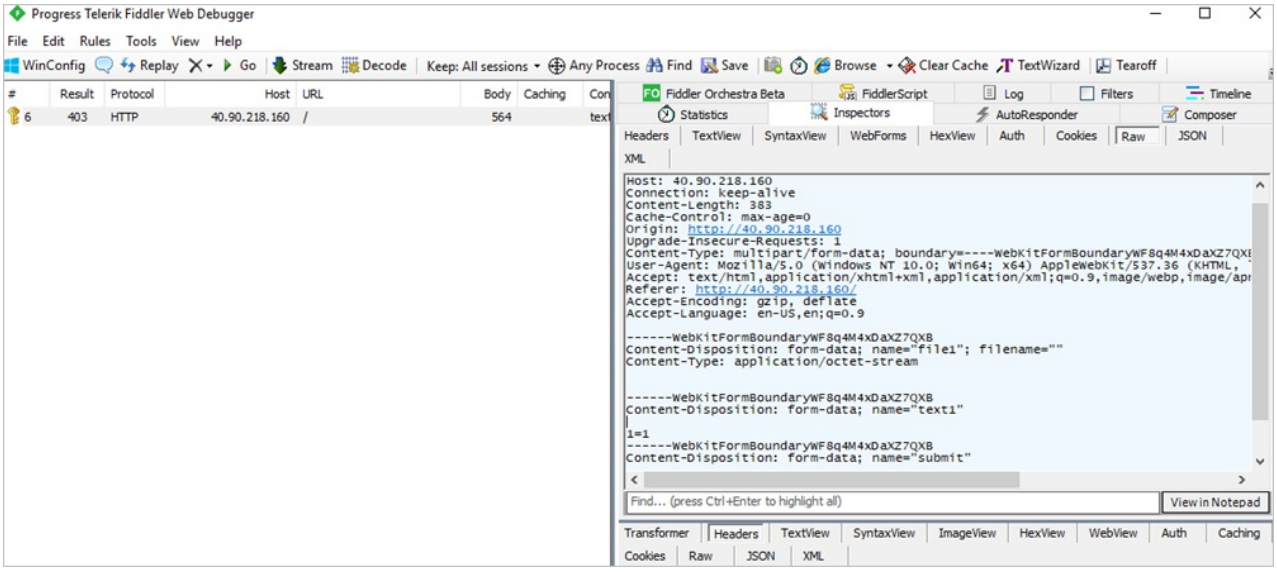
With your knowledge of how the CRS rule sets work, and that the CRS ruleset 3.0 works with an anomaly scoring system (see Web Application Firewall for Azure Application Gateway) you know that the bottom two rules with the **action: Blocked** property are blocking based on the total anomaly score. The rules to focus on are the top two.

The first entry is logged because the user used a numeric IP address to navigate to the Application Gateway, which can be ignored in this case.

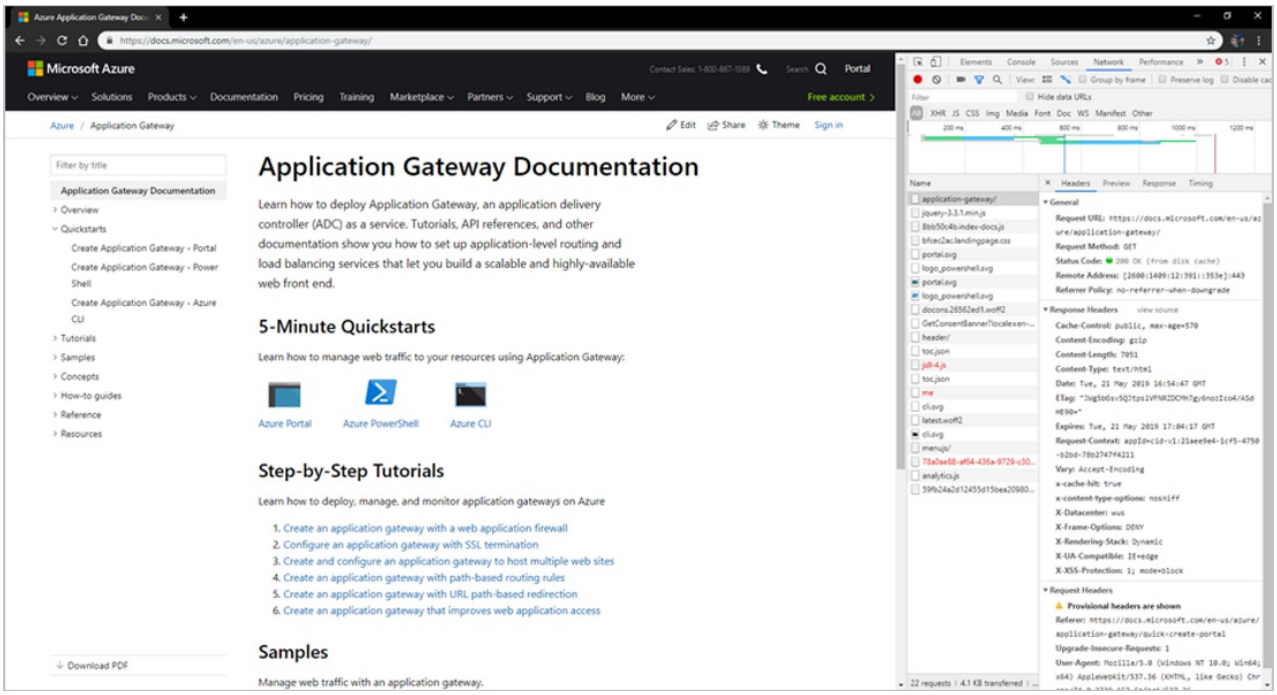
The second one (rule 942130) is the interesting one. You can see in the details that it matched a pattern (1=1), and the field is named **text1**. Follow the same previous steps to exclude the **Request Attribute Name** that **equals 1=1**.

Finding request header names

Fiddler is a useful tool once again to find request header names. In the following screenshot, you can see the headers for this GET request, which include *Content-Type*, *User-Agent*, and so on.



Another way to view request and response headers is to look inside the developer tools of Chrome. You can press F12 or right-click -> **Inspect** -> **Developer Tools**, and select the **Network** tab. Load a web page, and click the request you want to inspect.



Finding request cookie names

If the request contains cookies, the **Cookies** tab can be selected to view them in Fiddler.

Restrict global parameters to eliminate false positives

- Disable request body inspection

  By setting **Inspect request body** to off, the request bodies of all traffic will not be evaluated by your WAF. This may be useful if you know that the request bodies aren't malicious to your application.

  By disabling this option, only the request body is not inspected. The headers and cookies remain inspected, unless individual ones are excluded using the exclusion list functionality.

- File size limits

  By limiting the file size for your WAF, you're limiting the possibility of an attack happening to your web servers. By allowing large files to be uploaded, the risk of your backend being overwhelmed increases. Limiting the file size to a normal use case for your application is just another way to prevent attacks.

  > **NOTE**
  > If you know that your app will never need any file upload above a given size, you can restrict that by setting a limit.

Firewall Metrics (WAF_v1 only)

For v1 Web Application Firewalls, the following metrics are now available in the portal:

1. Web Application Firewall Blocked Request Count The number of requests that were blocked
2. Web Application Firewall Blocked Rule Count All rules that were matched **and** the request was blocked
3. Web Application Firewall Total Rule Distribution All rules that were matched during evaluation

To enable metrics, select the **Metrics** tab in the portal, and select one of the three metrics.

Next steps

See How to configure web application firewall on Application Gateway.

Use Log Analytics to examine Application Gateway Web Application Firewall Logs

11/4/2019 • 2 minutes to read • Edit Online

Once your Application Gateway WAF is operational, you can enable logs to inspect what is happening with each request. Firewall logs give insight to what the WAF is evaluating, matching, and blocking. With Log Analytics, you can examine the data inside the firewall logs to give even more insights. For more information about creating a Log Analytics workspace, see Create a Log Analytics workspace in the Azure portal. For more information about log queries, see Overview of log queries in Azure Monitor.

Import WAF logs

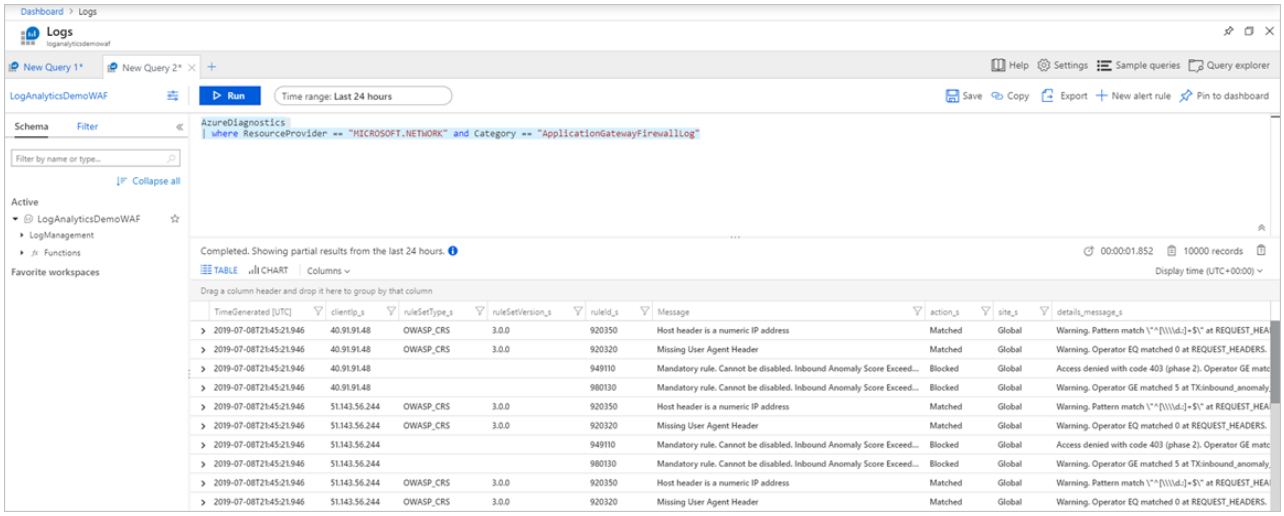
To import your firewall logs into Log Analytics, see Back-end health, diagnostic logs, and metrics for Application Gateway. When you have the firewall logs in your Log Analytics workspace, you can view data, write queries, create visualizations, and add them to your portal dashboard.

Explore data with examples

To view the raw data in the firewall log, you can run the following query:

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.NETWORK" and Category == "ApplicationGatewayFirewallLog"
```

This will look similar to the following query:



You can drill down into the data, and plot graphs or create visualizations from here. See the following queries as a starting point:

Matched/Blocked requests by IP

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.NETWORK" and Category == "ApplicationGatewayFirewallLog"
| summarize count() by clientIp_s, bin(TimeGenerated, 1m)
| render timechart
```

Matched/Blocked requests by URI

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.NETWORK" and Category == "ApplicationGatewayFirewallLog"
| summarize count() by requestUri_s, bin(TimeGenerated, 1m)
| render timechart
```

Top matched rules

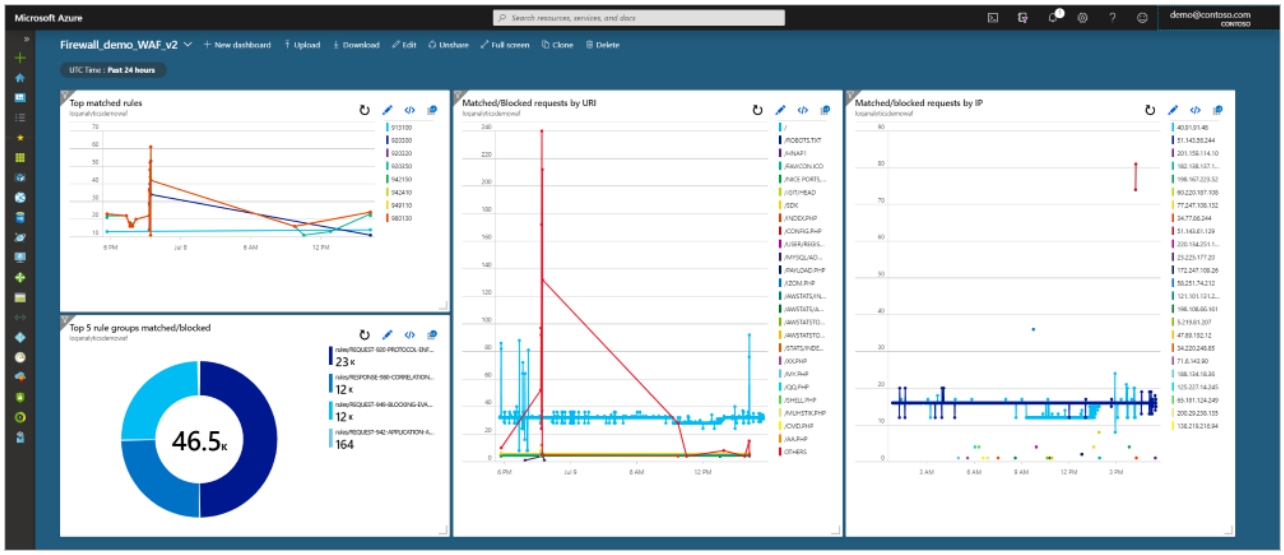
```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.NETWORK" and Category == "ApplicationGatewayFirewallLog"
| summarize count() by ruleId_s, bin(TimeGenerated, 1m)
| where count_ > 10
| render timechart
```

Top five matched rule groups

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.NETWORK" and Category == "ApplicationGatewayFirewallLog"
| summarize Count=count() by details_file_s, action_s
| top 5 by Count desc
| render piechart
```

Add to your dashboard

Once you create a query, you can add it to your dashboard. Select the **Pin to dashboard** in the top right of the log analytics workspace. With the previous four queries pinned to an example dashboard, this is the data you can see at a glance:



Next steps

Back-end health, diagnostic logs, and metrics for Application Gateway

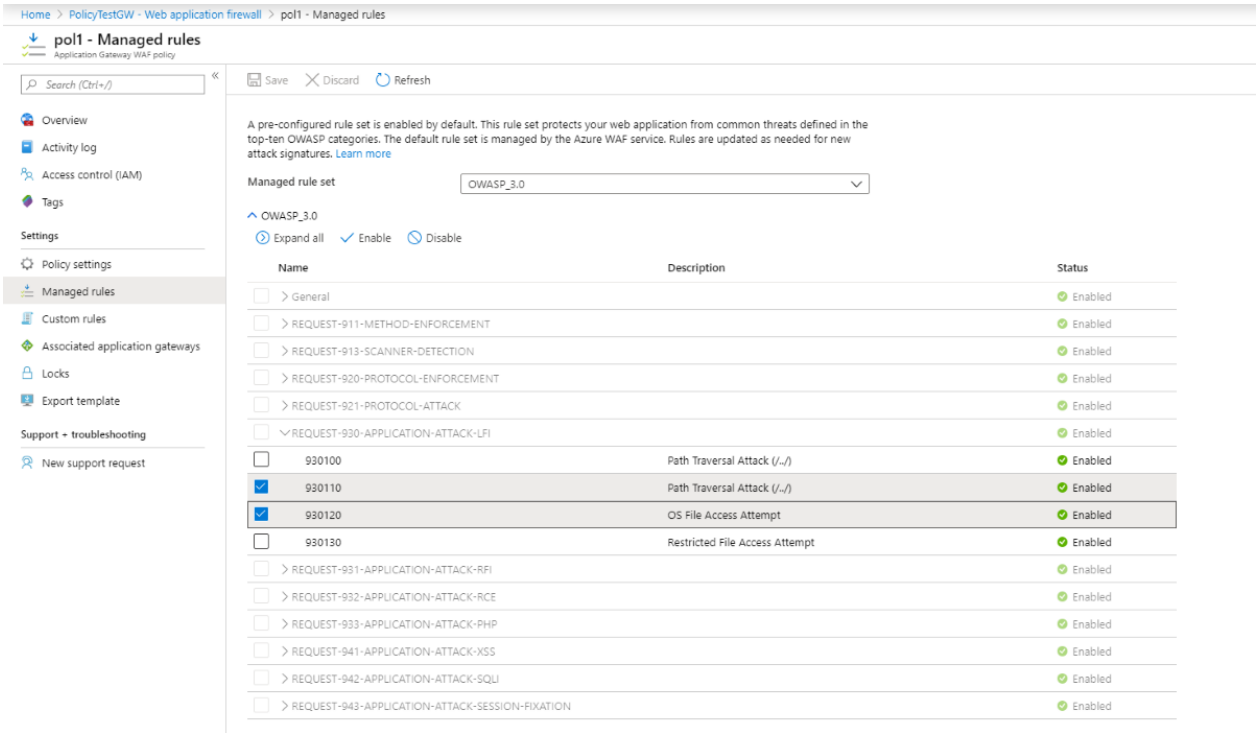
Web Application Firewall CRS rule groups and rules

11/13/2019 • 20 minutes to read • Edit Online

Application Gateway web application firewall (WAF) protects web applications from common vulnerabilities and exploits. This is done through rules that are defined based on the OWASP core rule sets 3.1, 3.0, or 2.2.9. These rules can be disabled on a rule-by-rule basis. This article contains the current rules and rule sets offered.

Core rule sets

The Application Gateway WAF comes pre-configured with CRS 3.0 by default. But you can choose to use CRS 3.1 or CRS 2.2.9 instead. CRS 3.1 offers new rule sets defending against Java infections, an initial set of file upload checks, fixed false positives, and more. CRS 3.0 offers reduced false positives compared with CRS 2.2.9. You can also customize rules to suit your needs.



The WAF protects against the following web vulnerabilities:

- SQL-injection attacks
- Cross-site scripting attacks
- Other common
  attacks, such as command injection, HTTP request smuggling, HTTP response splitting, and remote file inclusion
- HTTP protocol violations
- HTTP protocol anomalies, such as missing host user-agent and accept headers
- Bots, crawlers, and scanners
- Common application misconfigurations (for example, Apache and IIS)

OWASP CRS 3.1

CRS 3.1 includes 13 rule groups, as shown in the following table. Each group contains multiple rules, which can be disabled.

RULE GROUP	DESCRIPTION
General	General group
REQUEST-911-METHOD-ENFORCEMENT	Lock-down methods (PUT, PATCH)
REQUEST-913-SCANNER-DETECTION	Protect against port and environment scanners
REQUEST-920-PROTOCOL-ENFORCEMENT	Protect against protocol and encoding issues
REQUEST-921-PROTOCOL-ATTACK	Protect against header injection, request smuggling, and response splitting
REQUEST-930-APPLICATION-ATTACK-LFI	Protect against file and path attacks
REQUEST-931-APPLICATION-ATTACK-RFI	Protect against remote file inclusion (RFI) attacks
REQUEST-932-APPLICATION-ATTACK-RCE	Protect again remote code execution attacks
REQUEST-933-APPLICATION-ATTACK-PHP	Protect against PHP-injection attacks
REQUEST-941-APPLICATION-ATTACK-XSS	Protect against cross-site scripting attacks
REQUEST-942-APPLICATION-ATTACK-SQLI	Protect against SQL-injection attacks
REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION	Protect against session-fixation attacks
REQUEST-944-APPLICATION-ATTACK-SESSION-JAVA	Protect against JAVA attacks

OWASP CRS 3.0

CRS 3.0 includes 12 rule groups, as shown in the following table. Each group contains multiple rules, which can be disabled.

RULE GROUP	DESCRIPTION
General	General group
REQUEST-911-METHOD-ENFORCEMENT	Lock-down methods (PUT, PATCH)
REQUEST-913-SCANNER-DETECTION	Protect against port and environment scanners
REQUEST-920-PROTOCOL-ENFORCEMENT	Protect against protocol and encoding issues
REQUEST-921-PROTOCOL-ATTACK	Protect against header injection, request smuggling, and response splitting
REQUEST-930-APPLICATION-ATTACK-LFI	Protect against file and path attacks
REQUEST-931-APPLICATION-ATTACK-RFI	Protect against remote file inclusion (RFI) attacks
REQUEST-932-APPLICATION-ATTACK-RCE	Protect again remote code execution attacks

RULE GROUP	DESCRIPTION
REQUEST-933-APPLICATION-ATTACK-PHP	Protect against PHP-injection attacks
REQUEST-941-APPLICATION-ATTACK-XSS	Protect against cross-site scripting attacks
REQUEST-942-APPLICATION-ATTACK-SQLI	Protect against SQL-injection attacks
REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION	Protect against session-fixation attacks

OWASP CRS 2.2.9

CRS 2.2.9 includes 10 rule groups, as shown in the following table. Each group contains multiple rules, which can be disabled.

RULE GROUP	DESCRIPTION
crs_20_protocol_violations	Protect against protocol violations (such as invalid characters or a GET with a request body)
crs_21_protocol_anomalies	Protect against incorrect header information
crs_23_request_limits	Protect against arguments or files that exceed limitations
crs_30_http_policy	Protect against restricted methods, headers, and file types
crs_35_bad_robots	Protect against web crawlers and scanners
crs_40_generic_attacks	Protect against generic attacks (such as session fixation, remote file inclusion, and PHP injection)
crs_41_sql_injection_attacks	Protect against SQL-injection attacks
crs_41_xss_attacks	Protect against cross-site scripting attacks
crs_42_tight_security	Protect against path-traversal attacks
crs_45_trojans	Protect against backdoor trojans

The following rule groups and rules are available when using Web Application Firewall on Application Gateway.

- OWASP 3.1
- OWASP 3.0
- OWASP 2.2.9

Rule sets

General

RULEID	DESCRIPTION
200004	Possible Multipart Unmatched Boundary.

REQUEST-911-METHOD-ENFORCEMENT

RULEID	DESCRIPTION
911100	Method is not allowed by policy

REQUEST-913-SCANNER-DETECTION

RULEID	DESCRIPTION
913100	Found User-Agent associated with security scanner
913101	Found User-Agent associated with scripting/generic HTTP client
913102	Found User-Agent associated with web crawler/bot
913110	Found request header associated with security scanner
913120	Found request filename/argument associated with security scanner

REQUEST-920-PROTOCOL-ENFORCEMENT

RULEID	DESCRIPTION
920100	Invalid HTTP Request Line
920120	Attempted multipart/form-data bypass
920121	Attempted multipart/form-data bypass
920130	Failed to parse request body.
920140	Multipart request body failed strict validation
920160	Content-Length HTTP header is not numeric.
920170	GET or HEAD Request with Body Content.
920171	GET or HEAD Request with Transfer-Encoding.
920180	POST request missing Content-Length Header.
920190	Range = Invalid Last Byte Value.
920200	Range = Too many fields (6 or more)
920201	Range = Too many fields for pdf request (35 or more)
920202	Range = Too many fields for pdf request (6 or more)
920210	Multiple/Conflicting Connection Header Data Found.

RULEID	DESCRIPTION
920220	URL Encoding Abuse Attack Attempt
920230	Multiple URL Encoding Detected
920240	URL Encoding Abuse Attack Attempt
920250	UTF8 Encoding Abuse Attack Attempt
920260	Unicode Full/Half Width Abuse Attack Attempt
920270	Invalid character in request (null character)
920271	Invalid character in request (non printable characters)
920272	Invalid character in request (outside of printable chars below ascii 127)
920273	Invalid character in request (outside of very strict set)
920274	Invalid character in request headers (outside of very strict set)
920280	Request Missing a Host Header
920290	Empty Host Header
920300	Request Missing an Accept Header
920310	Request Has an Empty Accept Header
920311	Request Has an Empty Accept Header
920320	Missing User Agent Header
920330	Empty User Agent Header
920340	Request Containing Content but Missing Content-Type header
920341	Request containing content requires Content-Type header
920350	Host header is a numeric IP address
920360	Argument name too long
920370	Argument value too long
920380	Too many arguments in request
920390	Total arguments size exceeded

RULEID	DESCRIPTION
920400	Uploaded file size too large
920410	Total uploaded files size too large
920420	Request content type is not allowed by policy
920430	HTTP protocol version is not allowed by policy
920440	URL file extension is restricted by policy
920450	HTTP header is restricted by policy (%@{MATCHED_VAR})
920460	Abnormal Escape Characters
920470	Illegal Content-Type header
920480	Restrict charset parameter within the content-type header

REQUEST-921-PROTOCOL-ATTACK

RULEID	DESCRIPTION
921110	HTTP Request Smuggling Attack
921120	HTTP Response Splitting Attack
921130	HTTP Response Splitting Attack
921140	HTTP Header Injection Attack via headers
921150	HTTP Header Injection Attack via payload (CR/LF detected)
921151	HTTP Header Injection Attack via payload (CR/LF detected)
921160	HTTP Header Injection Attack via payload (CR/LF and header-name detected)
921170	HTTP Parameter Pollution
921180	HTTP Parameter Pollution (%{TX.1})

REQUEST-930-APPLICATION-ATTACK-LFI

RULEID	DESCRIPTION
930100	Path Traversal Attack (/../)
930110	Path Traversal Attack (/../)
930120	OS File Access Attempt

RULEID	DESCRIPTION
930130	Restricted File Access Attempt

REQUEST-931-APPLICATION-ATTACK-RFI

RULEID	DESCRIPTION
931100	Possible Remote File Inclusion (RFI) Attack = URL Parameter using IP Address
931110	Possible Remote File Inclusion (RFI) Attack = Common RFI Vulnerable Parameter Name used w/URL Payload
931120	Possible Remote File Inclusion (RFI) Attack = URL Payload Used w/Trailing Question Mark Character (?)
931130	Possible Remote File Inclusion (RFI) Attack = Off-Domain Reference/Link

REQUEST-932-APPLICATION-ATTACK-RCE

RULEID	DESCRIPTION
932100	Remote Command Execution: Unix Command Injection
932105	Remote Command Execution: Unix Command Injection
932106	Remote Command Execution: Unix Command Injection
932110	Remote Command Execution: Windows Command Injection
932115	Remote Command Execution: Windows Command Injection
932120	Remote Command Execution = Windows PowerShell Command Found
932130	Remote Command Execution = Unix Shell Expression Found
932140	Remote Command Execution = Windows FOR/IF Command Found
932160	Remote Command Execution = Unix Shell Code Found
932170	Remote Command Execution = Shellshock (CVE-2014-6271)
932171	Remote Command Execution = Shellshock (CVE-2014-6271)
932180	Restricted File Upload Attempt
932190	Remote Command Execution: Wildcard bypass technique attempt

REQUEST-933-APPLICATION-ATTACK-PHP

RULEID	DESCRIPTION
933100	PHP Injection Attack = Opening/Closing Tag Found
933110	PHP Injection Attack = PHP Script File Upload Found
933111	PHP Injection Attack: PHP Script File Upload Found
933120	PHP Injection Attack = Configuration Directive Found
933130	PHP Injection Attack = Variables Found
933131	PHP Injection Attack: Variables Found
933140	PHP Injection Attack: I/O Stream Found
933150	PHP Injection Attack = High-Risk PHP Function Name Found
933151	PHP Injection Attack: Medium-Risk PHP Function Name Found
933160	PHP Injection Attack = High-Risk PHP Function Call Found
933161	PHP Injection Attack: Low-Value PHP Function Call Found
933170	PHP Injection Attack: Serialized Object Injection
933180	PHP Injection Attack = Variable Function Call Found
933190	PHP Injection Attack: PHP Closing Tag Found

REQUEST-941-APPLICATION-ATTACK-XSS

RULEID	DESCRIPTION
941100	XSS Attack Detected via libinjection
941101	XSS Attack Detected via libinjection
941110	XSS Filter - Category 1 = Script Tag Vector
941130	XSS Filter - Category 3 = Attribute Vector
941140	XSS Filter - Category 4 = Javascript URI Vector
941150	XSS Filter - Category 5 = Disallowed HTML Attributes
941160	NoScript XSS InjectionChecker: HTML Injection
941170	NoScript XSS InjectionChecker: Attribute Injection
941180	Node-Validator Blacklist Keywords

RULEID	DESCRIPTION
941190	XSS using style sheets
941200	XSS using VML frames
941210	XSS using obfuscated Javascript
941220	XSS using obfuscated VB Script
941230	XSS using 'embed' tag
941240	XSS using 'import' or 'implementation' attribute
941250	IE XSS Filters - Attack Detected
941260	XSS using 'meta' tag
941270	XSS using 'link' href
941280	XSS using 'base' tag
941290	XSS using 'applet' tag
941300	XSS using 'object' tag
941310	US-ASCII Malformed Encoding XSS Filter - Attack Detected.
941320	Possible XSS Attack Detected - HTML Tag Handler
941330	IE XSS Filters - Attack Detected.
941340	IE XSS Filters - Attack Detected.
941350	UTF-7 Encoding IE XSS - Attack Detected.

REQUEST-942-APPLICATION-ATTACK-SQLI

RULEID	DESCRIPTION
942100	SQL Injection Attack Detected via libinjection
942110	SQL Injection Attack: Common Injection Testing Detected
942130	SQL Injection Attack: SQL Tautology Detected.
942140	SQL Injection Attack = Common DB Names Detected
942150	SQL Injection Attack
942160	Detects blind sqli tests using sleep() or benchmark().

RULEID	DESCRIPTION
942170	Detects SQL benchmark and sleep injection attempts including conditional queries
942180	Detects basic SQL authentication bypass attempts 1/3
942190	Detects MSSQL code execution and information gathering attempts
942200	Detects MySQL comment-/space-obfuscated injections and backtick termination
942210	Detects chained SQL injection attempts 1/2
942220	Looking for integer overflow attacks, these are taken from skipfish, except 3.0.00738585072
942230	Detects conditional SQL injection attempts
942240	Detects MySQL charset switch and MSSQL DoS attempts
942250	Detects MATCH AGAINST, MERGE and EXECUTE IMMEDIATE injections
942251	Detects HAVING injections
942260	Detects basic SQL authentication bypass attempts 2/3
942270	Looking for basic sql injection. Common attack string for mysql oracle and others
942280	Detects Postgres pg_sleep injection, waitfor delay attacks and database shutdown attempts
942290	Finds basic MongoDB SQL injection attempts
942300	Detects MySQL comments, conditions and ch(a)r injections
942310	Detects chained SQL injection attempts 2/2
942320	Detects MySQL and PostgreSQL stored procedure/function injections
942330	Detects classic SQL injection probings 1/2
942340	Detects basic SQL authentication bypass attempts 3/3
942350	Detects MySQL UDF injection and other data/structure manipulation attempts
942360	Detects concatenated basic SQL injection and SQLLFI attempts

RULEID	DESCRIPTION
942361	Detects basic SQL injection based on keyword alter or union
942370	Detects classic SQL injection probings 2/2
942380	SQL Injection Attack
942390	SQL Injection Attack
942400	SQL Injection Attack
942410	SQL Injection Attack
942420	Restricted SQL Character Anomaly Detection (cookies): # of special characters exceeded (8)
942421	Restricted SQL Character Anomaly Detection (cookies): # of special characters exceeded (3)
942430	Restricted SQL Character Anomaly Detection (args): # of special characters exceeded (12)
942431	Restricted SQL Character Anomaly Detection (args): # of special characters exceeded (6)
942432	Restricted SQL Character Anomaly Detection (args): # of special characters exceeded (2)
942440	SQL Comment Sequence Detected.
942450	SQL Hex Encoding Identified
942460	Meta-Character Anomaly Detection Alert - Repetitive Non-Word Characters
942470	SQL Injection Attack
942480	SQL Injection Attack
942490	Detects classic SQL injection probings 3/3

REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION

RULEID	DESCRIPTION
943100	Possible Session Fixation Attack = Setting Cookie Values in HTML
943110	Possible Session Fixation Attack = SessionID Parameter Name with Off-Domain Referrer
943120	Possible Session Fixation Attack = SessionID Parameter Name with No Referrer

REQUEST-944-APPLICATION-ATTACK-SESSION-JAVA

RULEID	DESCRIPTION
944120	Possible payload execution and remote command execution
944130	Suspicious Java classes
944200	Exploitation of Java deserialization Apache Commons

Next steps

- Customize Web Application Firewall rules using the Azure portal

Understand the structure and syntax of Azure Resource Manager templates

2/26/2020 • 13 minutes to read • Edit Online

This article describes the structure of an Azure Resource Manager template. It presents the different sections of a template and the properties that are available in those sections.

This article is intended for users who have some familiarity with Resource Manager templates. It provides detailed information about the structure of the template. For a step-by-step tutorial that guides you through the process of creating a template, see Tutorial: Create and deploy your first Azure Resource Manager template.

Template format

In its simplest structure, a template has the following elements:

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

ELEMENT NAME	REQUIRED	DESCRIPTION

$schema	Yes	Location of the JSON schema file that describes the version of the template language.
		
		For resource group deployments, use:
		`https://schema.management.azure.com/schemas/2`
		`01-01/deploymentTemplate.json#`
		
		For subscription deployments, use:
		`https://schema.management.azure.com/schemas/2`
		`05-01/subscriptionDeploymentTemplate.json#`

contentVersion	Yes	Version of the template (such as 1.0.0.0). You can provide any value for this element. Use this value to document significant changes in your template. When deploying resources using the template, this value can be used to make sure that the right template is being used.

ELEMENT NAME	REQUIRED	DESCRIPTION
apiProfile	No	An API version that serves as a collection of API versions for resource types. Use this value to avoid having to specify API versions for each resource in the template. When you specify an API profile version and don't specify an API version for the resource type, Resource Manager uses the API version for that resource type that is defined in the profile. The API profile property is especially helpful when deploying a template to different environments, such as Azure Stack and global Azure. Use the API profile version to make sure your template automatically uses versions that are supported in both environments. For a list of the current API profile versions and the resources API versions defined in the profile, see API Profile. For more information, see Track versions using API profiles.
parameters	No	Values that are provided when deployment is executed to customize resource deployment.
variables	No	Values that are used as JSON fragments in the template to simplify template language expressions.
functions	No	User-defined functions that are available within the template.
resources	Yes	Resource types that are deployed or updated in a resource group or subscription.
outputs	No	Values that are returned after deployment.

Each element has properties you can set. This article describes the sections of the template in greater detail.

Parameters

In the parameters section of the template, you specify which values you can input when deploying the resources. You're limited to 256 parameters in a template. You can reduce the number of parameters by using objects that contain multiple properties.

The available properties for a parameter are:

```
"parameters": {
  "<parameter-name>" : {
    "type" : "<type-of-parameter-value>",
    "defaultValue": "<default-value-of-parameter>",
    "allowedValues": [ "<array-of-allowed-values>" ],
    "minValue": <minimum-value-for-int>,
    "maxValue": <maximum-value-for-int>,
    "minLength": <minimum-length-for-string-or-array>,
    "maxLength": <maximum-length-for-string-or-array-parameters>,
    "metadata": {
      "description": "<description-of-the parameter>"
    }
  }
}
```

ELEMENT NAME	REQUIRED	DESCRIPTION
parameter-name	Yes	Name of the parameter. Must be a valid JavaScript identifier.
type	Yes	Type of the parameter value. The allowed types and values are **string**, **securestring**, **int**, **bool**, **object**, **secureObject**, and **array**. See Data types.
defaultValue	No	Default value for the parameter, if no value is provided for the parameter.
allowedValues	No	Array of allowed values for the parameter to make sure that the right value is provided.
minValue	No	The minimum value for int type parameters, this value is inclusive.
maxValue	No	The maximum value for int type parameters, this value is inclusive.
minLength	No	The minimum length for string, secure string, and array type parameters, this value is inclusive.
maxLength	No	The maximum length for string, secure string, and array type parameters, this value is inclusive.
description	No	Description of the parameter that is displayed to users through the portal. For more information, see Comments in templates.

For examples of how to use parameters, see Parameters in Azure Resource Manager templates.

Data types

For integers passed as inline parameters, the range of values may be limited by the SDK or command-line tool you use for deployment. For example, when using PowerShell to deploy a template, integer types can range from -2147483648 to 2147483647. To avoid this limitation, specify large integer values in a parameter file. Resource types apply their own limits for integer properties.

When specifying boolean and integer values in your template, don't surround the value with quotation marks. Start and end string values with double quotation marks.

Objects start with a left brace and end with a right brace. Arrays start with a left bracket and end with a right bracket.

Secure strings and secure objects can't be read after resource deployment.

For samples of formatting data types, see Parameter type formats.

Variables

In the variables section, you construct values that can be used throughout your template. You don't need to define variables, but they often simplify your template by reducing complex expressions.

The following example shows the available options for defining a variable:

```
"variables": {
  "<variable-name>": "<variable-value>",
  "<variable-name>": {
    <variable-complex-type-value>
  },
  "<variable-object-name>": {
    "copy": [
      {
        "name": "<name-of-array-property>",
        "count": <number-of-iterations>,
        "input": <object-or-value-to-repeat>
      }
    ]
  },
  "copy": [
    {
      "name": "<variable-array-name>",
      "count": <number-of-iterations>,
      "input": <object-or-value-to-repeat>
    }
  ]
}
```

For information about using `copy` to create several values for a variable, see Variable iteration.

For examples of how to use variables, see Variables in Azure Resource Manager template.

Functions

Within your template, you can create your own functions. These functions are available for use in your template. Typically, you define complicated expressions that you don't want to repeat throughout your template. You create the user-defined functions from expressions and functions that are supported in templates.

When defining a user function, there are some restrictions:

- The function can't access variables.
- The function can only use parameters that are defined in the function. When you use the parameters function within a user-defined function, you're restricted to the parameters for that function.
- The function can't call other user-defined functions.
- The function can't use the reference function.
- Parameters for the function can't have default values.

```
"functions": [
  {
    "namespace": "<namespace-for-functions>",
    "members": {
      "<function-name>": {
        "parameters": [
          {
            "name": "<parameter-name>",
            "type": "<type-of-parameter-value>"
          }
        ],
        "output": {
          "type": "<type-of-output-value>",
          "value": "<function-return-value>"
        }
      }
    }
  }
],
```

ELEMENT NAME	REQUIRED	DESCRIPTION
namespace	Yes	Namespace for the custom functions. Use to avoid naming conflicts with template functions.
function-name	Yes	Name of the custom function. When calling the function, combine the function name with the namespace. For example, to call a function named uniqueName in the namespace contoso, use `"[contoso.uniqueName()]"`.
parameter-name	No	Name of the parameter to be used within the custom function.
parameter-value	No	Type of the parameter value. The allowed types and values are **string**, **securestring**, **int**, **bool**, **object**, **secureObject**, and **array**.
output-type	Yes	Type of the output value. Output values support the same types as function input parameters.
output-value	Yes	Template language expression that is evaluated and returned from the function.

For examples of how to use custom functions, see User-defined functions in Azure Resource Manager template.

Resources

In the resources section, you define the resources that are deployed or updated.

You define resources with the following structure:

```
"resources": [
  {
      "condition": "<true-to-deploy-this-resource>",
      "type": "<resource-provider-namespace/resource-type-name>",
      "apiVersion": "<api-version-of-resource>",
      "name": "<name-of-the-resource>",
      "comments": "<your-reference-notes>",
      "location": "<location-of-resource>",
      "dependsOn": [
          "<array-of-related-resource-names>"
      ],
      "tags": {
          "<tag-name1>": "<tag-value1>",
          "<tag-name2>": "<tag-value2>"
      },
      "sku": {
          "name": "<sku-name>",
          "tier": "<sku-tier>",
          "size": "<sku-size>",
          "family": "<sku-family>",
          "capacity": <sku-capacity>
      },
      "kind": "<type-of-resource>",
      "copy": {
          "name": "<name-of-copy-loop>",
          "count": <number-of-iterations>,
          "mode": "<serial-or-parallel>",
          "batchSize": <number-to-deploy-serially>
      },
      "plan": {
          "name": "<plan-name>",
          "promotionCode": "<plan-promotion-code>",
          "publisher": "<plan-publisher>",
          "product": "<plan-product>",
          "version": "<plan-version>"
      },
      "properties": {
          "<settings-for-the-resource>",
          "copy": [
              {
                  "name": ,
                  "count": ,
                  "input": {}
              }
          ]
      },
      "resources": [
          "<array-of-child-resources>"
      ]
  }
]
```

ELEMENT NAME	REQUIRED	DESCRIPTION
condition	No	Boolean value that indicates whether the resource will be provisioned during this deployment. When `true`, the resource is created during deployment. When `false`, the resource is skipped for this deployment. See condition.

ELEMENT NAME	REQUIRED	DESCRIPTION
type	Yes	Type of the resource. This value is a combination of the namespace of the resource provider and the resource type (such as **Microsoft.Storage/storageAccounts**). To determine available values, see template reference. For a child resource, the format of the type depends on whether it's nested within the parent resource or defined outside of the parent resource. See Set name and type for child resources.
apiVersion	Yes	Version of the REST API to use for creating the resource. To determine available values, see template reference.
name	Yes	Name of the resource. The name must follow URI component restrictions defined in RFC3986. Azure services that expose the resource name to outside parties validate the name to make sure it isn't an attempt to spoof another identity. For a child resource, the format of the name depends on whether it's nested within the parent resource or defined outside of the parent resource. See Set name and type for child resources.
comments	No	Your notes for documenting the resources in your template. For more information, see Comments in templates.
location	Varies	Supported geo-locations of the provided resource. You can select any of the available locations, but typically it makes sense to pick one that is close to your users. Usually, it also makes sense to place resources that interact with each other in the same region. Most resource types require a location, but some types (such as a role assignment) don't require a location. See Set resource location.
dependsOn	No	Resources that must be deployed before this resource is deployed. Resource Manager evaluates the dependencies between resources and deploys them in the correct order. When resources aren't dependent on each other, they're deployed in parallel. The value can be a comma-separated list of a resource names or resource unique identifiers. Only list resources that are deployed in this template. Resources that aren't defined in this template must already exist. Avoid adding unnecessary dependencies as they can slow your deployment and create circular dependencies. For guidance on setting dependencies, see Defining dependencies in Azure Resource Manager templates.

ELEMENT NAME	REQUIRED	DESCRIPTION
tags	No	Tags that are associated with the resource. Apply tags to logically organize resources across your subscription.
sku	No	Some resources allow values that define the SKU to deploy. For example, you can specify the type of redundancy for a storage account.
kind	No	Some resources allow a value that defines the type of resource you deploy. For example, you can specify the type of Cosmos DB to create.
copy	No	If more than one instance is needed, the number of resources to create. The default mode is parallel. Specify serial mode when you don't want all or the resources to deploy at the same time. For more information, see Create several instances of resources in Azure Resource Manager.
plan	No	Some resources allow values that define the plan to deploy. For example, you can specify the marketplace image for a virtual machine.
properties	No	Resource-specific configuration settings. The values for the properties are the same as the values you provide in the request body for the REST API operation (PUT method) to create the resource. You can also specify a copy array to create several instances of a property. To determine available values, see template reference.
resources	No	Child resources that depend on the resource being defined. Only provide resource types that are permitted by the schema of the parent resource. Dependency on the parent resource isn't implied. You must explicitly define that dependency. See Set name and type for child resources.

Outputs

In the Outputs section, you specify values that are returned from deployment. Typically, you return values from resources that were deployed.

The following example shows the structure of an output definition:

```
"outputs": {
  "<output-name>": {
    "condition": "<boolean-value-whether-to-output-value>",
    "type": "<type-of-output-value>",
    "value": "<output-value-expression>",
    "copy": {
      "count": <number-of-iterations>,
      "input": <values-for-the-variable>
    }
  }
}
```

ELEMENT NAME	REQUIRED	DESCRIPTION
output-name	Yes	Name of the output value. Must be a valid JavaScript identifier.
condition	No	Boolean value that indicates whether this output value is returned. When `true`, the value is included in the output for the deployment. When `false`, the output value is skipped for this deployment. When not specified, the default value is `true`.
type	Yes	Type of the output value. Output values support the same types as template input parameters. If you specify **securestring** for the output type, the value isn't displayed in the deployment history and can't be retrieved from another template. To use a secret value in more than one template, store the secret in a Key Vault and reference the secret in the parameter file. For more information, see Use Azure Key Vault to pass secure parameter value during deployment.
value	No	Template language expression that is evaluated and returned as output value. Specify either **value** or **copy**.
copy	No	Used to return more than one value for an output. Specify **value** or **copy**. For more information, see Output iteration in Azure Resource Manager templates.

For examples of how to use outputs, see Outputs in Azure Resource Manager template.

Comments and metadata

You have a few options for adding comments and metadata to your template.

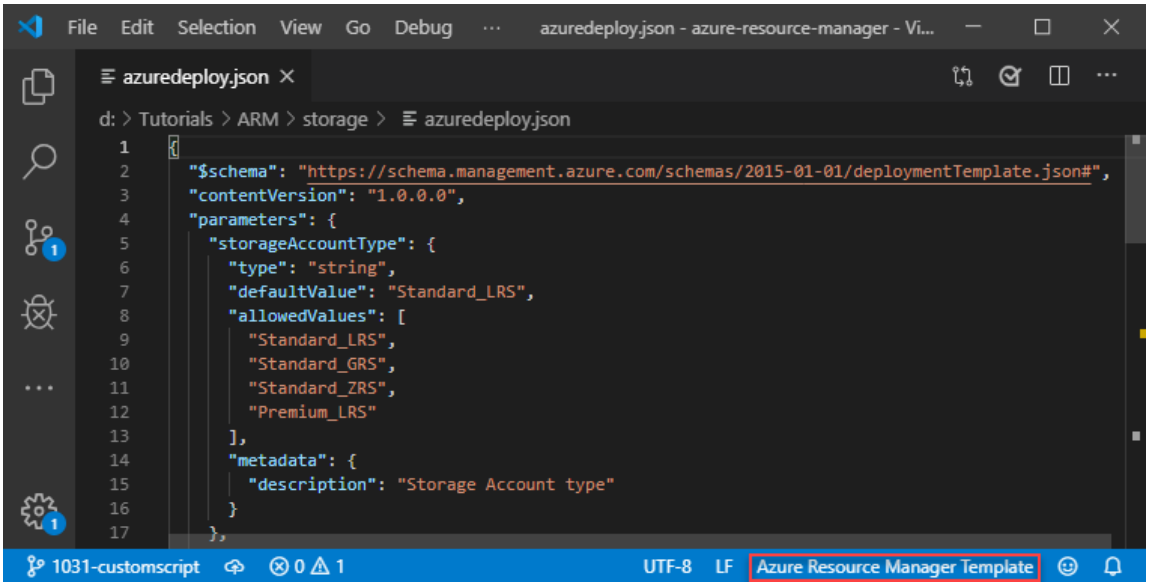
Comments

For inline comments, you can use either `//` or `/* ... */` but this syntax doesn't work with all tools. You can't use the portal template editor to work on templates with inline comments. If you add this style of comment, be sure the tools you use support inline JSON comments.

> **NOTE**
> To deploy templates with comments by using Azure CLI, you must use the `--handle-extended-json-format` switch.

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
  "name": "[variables('vmName')]", // to customize name, change it in variables
  "location": "[parameters('location')]", //defaults to resource group location
  "dependsOn": [ /* storage account and network interface must be deployed first */
    "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
  ],
```

In Visual Studio Code, the Azure Resource Manager Tools extension can automatically detect Resource Manager template and change the language mode accordingly. If you see **Azure Resource Manager Template** at the bottom-right corner of VS Code, you can use the inline comments. The inline comments are no longer marked as invalid.



Metadata

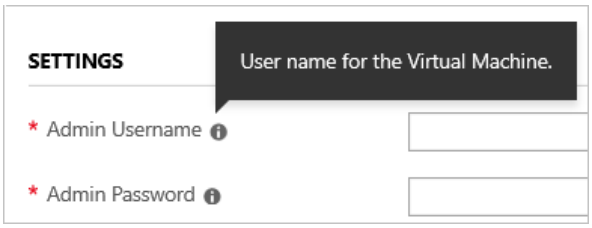
You can add a `metadata` object almost anywhere in your template. Resource Manager ignores the object, but your JSON editor may warn you that the property isn't valid. In the object, define the properties you need.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "comments": "This template was developed for demonstration purposes.",
    "author": "Example Name"
  },
```

For **parameters**, add a `metadata` object with a `description` property.

```
"parameters": {
  "adminUsername": {
    "type": "string",
    "metadata": {
      "description": "User name for the Virtual Machine."
    }
  },
```

When deploying the template through the portal, the text you provide in the description is automatically used as a tip for that parameter.



For **resources**, add a `comments` element or a metadata object. The following example shows both a comments element and a metadata object.

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2018-07-01",
    "name": "[concat('storage', uniqueString(resourceGroup().id))]",
    "comments": "Storage account used to store VM disks",
    "location": "[parameters('location')]",
    "metadata": {
      "comments": "These tags are needed for policy compliance."
    },
    "tags": {
      "Dept": "[parameters('deptName')]",
      "Environment": "[parameters('environment')]"
    },
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "properties": {}
  }
]
```

For **outputs**, add a metadata object to the output value.

```
"outputs": {
  "hostname": {
    "type": "string",
    "value": "[reference(variables('publicIPAddressName')).dnsSettings.fqdn]",
    "metadata": {
      "comments": "Return the fully qualified domain name"
    }
  },
```

You can't add a metadata object to user-defined functions.

Multi-line strings

You can break a string into multiple lines. For example, see the location property and one of the comments in the following JSON example.

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
  "name": "[variables('vmName')]", // to customize name, change it in variables
  "location": "[
    parameters('location')
    ]", //defaults to resource group location
  /*
    storage account and network interface
    must be deployed first
  */
  "dependsOn": [
    "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
  ],
```

To deploy templates with multi-line strings by using Azure CLI, you must use the `--handle-extended-json-format` switch.

Next steps

- To view complete templates for many different types of solutions, see the Azure Quickstart Templates.
- For details about the functions you can use from within a template, see Azure Resource Manager Template Functions.
- To combine several templates during deployment, see Using linked templates with Azure Resource Manager.
- For recommendations about creating templates, see Azure Resource Manager template best practices.
- For recommendations on creating Resource Manager templates that you can use across all Azure environments and Azure Stack, see Develop Azure Resource Manager templates for cloud consistency.