Each active response must be configured in its own separate <active-response> section.

<command>

  <name>host-deny</name>

  <executable>host-deny</executable>

  <timeout\_allowed>yes</timeout\_allowed>

</command>

* <name>: Sets a name for the command. In this case, host-deny.
* <executable>: Specifies the active response script or executable that must run upon a trigger. You don't need to specify the file name extension unless you have multiple scripts sharing the same name. In this case, it’s the host-deny executable.
* <timeout\_allowed>: Allows a timeout after a period of time. Setting this value to yes reverts the action after a period of time.

<!-- For Windows systems -->

<command>

  <name>win\_route-null</name>

  <executable>route-null.exe</executable>

  <timeout\_allowed>yes</timeout\_allowed>

</command>

1. Restart the Wazuh manager to apply all the changes made:
2. $ sudo systemctl restart wazuh-manager

**Default active response scripts**

Linux endpoints

/var/ossec/active-response/bin

Windows endpoints

C:\Program Files (x86)\ossec-agent\active-response\bin  directory

**Using custom active response scripts**

**Linux/Unix**

1. Add your custom active response script or executable to the /var/ossec/active-response/bin directory on Linux/Unix endpoints.
2. Change the script permissions and ownership as shown below:

$ sudo chmod 750 /var/ossec/active-response/bin/<CUSTOM\_SCRIPT>

$ sudo chown root:wazuh /var/ossec/active-response/bin/<CUSTOM\_SCRIPT>

**Windows**

1. Add your custom active response script or executable to the C:\Program Files (x86)\ossec-agent\active-response\bin directory on Windows endpoints.

In order to use Active Response with a custom script you need to bear in mind some things:

* Your custom script must be in a specific folder (/var/ossec/active-response/bin) \*
* When the scripts are in that folder, you don' t have to specify the path or the shell, you just write the name of the script.  
  After all that you will need to restart wazuh (/var/ossec/bin/ossec-control restart) and test it.

  <command>

    <name>test-command</name>

    <executable>response.sh</executable> \*

    <timeout\_allowed>no</timeout\_allowed>

  </command>

  <active-response>

    <command>test-command</command>

    <location>local</location>

    <level>1</level>

  </active-response>

<active-response>

<command>host-deny</command>

<location>local</location>

<level>7</level>

<timeout>600</timeout>

</active-response>

* <command>: Specifies the command to configure. This is the command name defined in the previous step.
* <location>: Specifies where the command must execute. The options are:
  + local: It runs the script on the monitored endpoint that generated the alert.
  + server: It runs the script on the Wazuh server.
  + defined-agent: It runs the script on a predefined agent. Use the <agent\_id> tag to specify the ID of the Wazuh agent that must run the script regardless of where the event occurred.

Firewall Rules Windows

# Define the IP address, port number, and protocol

$IPToBlock = "*IP address*”

$PortToBlock = *port#*

$ProtocolToBlock = "TCP"

# Create a new firewall rule to block incoming traffic

New-NetFirewallRule -DisplayName "Block-IP-Rule" -Direction Inbound -LocalPort $PortToBlock -Protocol $ProtocolToBlock -RemoteAddress $IPToBlock -Action Block

Linux Blocking access to file

#!/bin/bash

# Define the path to the critical file or directory

FILE\_OR\_DIR="/path/to/your/critical/file\_or\_directory"

# Set restrictive permissions to block access

chmod 700 "$FILE\_OR\_DIR"

# Optionally, set the file or directory to be owned by a specific user and group

# chown username:groupname "$FILE\_OR\_DIR"

# Log the action

echo "$(date) - Restricted access to $FILE\_OR\_DIR" >> /var/log/custom\_file\_protection.log

Powershell

# Define the path to the critical file or directory

$FilePath = "C:\Path\To\Your\Critical\FileOrDirectory"

# Define the account for which access will be denied (replace with an actual username or group)

$AccountToDeny = "DOMAIN\UserOrGroupName"

# Set restrictive permissions to block access

$acl = Get-Acl -Path $FilePath

$rule = New-Object System.Security.AccessControl.FileSystemAccessRule($AccountToDeny, "FullControl", "Deny")

$acl.AddAccessRule($rule)

Set-Acl -Path $FilePath -AclObject $acl

# Log the action

Add-Content -Path "C:\Path\To\Your\Log\File.log" -Value "$(Get-Date) - Restricted access to $FilePath"

* Get-Acl: Retrieves the Access Control List (ACL) of the specified file or directory.
* New-Object: Creates a new access rule to deny access ("FullControl") for the specified user or group.
* AddAccessRule: Adds the new access rule to the ACL.
* Set-Acl: Applies the modified ACL to the file or directory.
* Add-Content: Logs the action to a custom log file.

Replace "C:\Path\To\Your\Critical\FileOrDirectory" with the actual path to your critical file or directory. Also, replace "DOMAIN\UserOrGroupName" with the appropriate username or group that you want to deny access.