Smart Configuration of AWS Security Group Using PowerShell

Sivaprasad Padisetty (May 2014)

*Securing the infrastructure is critical for any company’s success. Absolute security does not exist and hence it is important to follow security best practices and implementing redundant layers of security aka defense in depth. Likewise security is never complete, one has to continuously monitor and enhance security infrastructure. Keep up with security trends!*

*Firewalls is an effective mechanism to mitigate security threats. Firewalls limit access to specific type of network traffic and allow traffic from valid sources only. It is important to restrict traffic only from valid source IP addresses, this will substantially prune security attack surface. You should restrict internal traffic just like external (e.g.) Allow DB connections only from middle tier and bastion hosts. This will prevent lateral movement if one of the server is compromised. Firewalls can come in layers. The last layer of defense is the OS firewall layer running inside the instance. AWS offers two additional firewall layers (i.e.) Security Groups and Network ACLs.*

# Security Group Overview

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

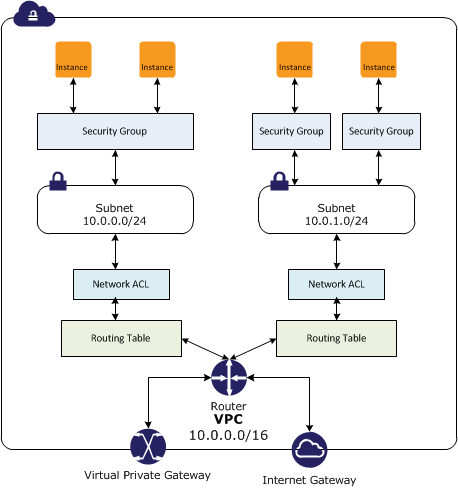
For each security group, you add rules that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic. This section describes the basics things you need to know about security groups for your VPC and their rules. Learn more [here](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html).

# Network ACLs

A network access control list (ACL) is an optional layer of security that acts as a firewall for controlling traffic in and out of a subnet. Network ACLs are collection of stateless rules, processed in order, supports both allow and deny rules. Learn more [here](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_ACLs.html).

# Security Groups and Network ACLs

* Security groups—Act as a firewall for associated Amazon EC2 instances, controlling both inbound and outbound traffic at the instance level
* Network access control lists (ACLs)—Act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level



The following table summarizes the basic differences between security groups and network ACLs.

| **Security Group** | **Network ACL** |
| --- | --- |
| Operates at the instance level (first layer of defense) | Operates at the subnet level (second layer of defense) |
| Supports allow rules only | Supports allow rules and deny rules |
| Is stateful: Return traffic is automatically allowed, regardless of any rules | Is stateless: Return traffic must be explicitly allowed by rules |
| All rules are processed before deciding whether to allow traffic | Rules are processed in number order when deciding whether to allow traffic |

# PowerShell Automation of Security Groups

As discussed before Security Groups are a powerful mechanism to specify what traffic to allow and from what IP addresses. The source IP addresses range is defined in the CIDR notation. Security Groups are flexible in that they can have one or more rules and each rule can specify one or more IP address ranges.

One Nifty trick to only allow traffic from your computer is to use the service <http://checkip.amazonaws.com>. This endpoint returns your public IP address as seen by AWS. Configuring the Security Group only to allow traffic from this source is the most security it can get. This will likely work for the test scenarios but for production you might have to open up depending on the scenario. (e.g.) Front end webserver has to be opened up globally with CIDR “0.0.0.0/0”

The following script can be used as a starting point for your customization, it illustrates the basic concepts. By default it enables PowerShell, HTTP, ICMP and RDP and restricts the source IP to checkup (i.e.) to your public IP address. This script is idempotent in the sense that it can be run any number of times. If the config is different it will be updated.

# Creates or updates the security group

# Default it enables RDP, PowerShell, HTTP and ICMP.

# Define appropriate switch to disable specific protocol

# If SourceIPRange is not defined, it configures based on http://checkip.amazonaws.com

function Update-WinEC2FireWallSource

{

param (

$SecurityGroupName = 'sg\_rdp\_ps\_http\_icmp',

$Region=$defaultRegion,

$SourceIPRange = $null,

[switch] $NoRDP,

[switch] $NoPS,

[switch] $NoHTTP,

[switch] $NoICMP

)

trap {break }

$ErrorActionPreference = 'Stop'

Set-DefaultAWSRegion $defaultRegion

if ($SourceIPRange -eq $null)

{

$bytes = (Invoke-WebRequest 'http://checkip.amazonaws.com/').Content

$SourceIPRange = @([System.Text.Encoding]::Ascii.GetString($bytes).Trim() + "/32")

}

else

{

$SourceIPRange = @($SourceIPRange) #Make it an array, if not already

}

$sg = Get-EC2SecurityGroup | ? { $\_.GroupName -eq $SecurityGroupName}

if ($sg -eq $null)

{

#Create the firewall security group

$groupid = New-EC2SecurityGroup $SecurityGroupName -Description "Enables rdp, ps, http and icmp"

}

else

{

foreach ($ipPermission in $sg.IpPermissions)

{

$delete = $true # will be set to false if we find exact match

if ($ipPermission.IpProtocol -eq 'tcp' -and

$ipPermission.FromPort -eq 3389 -and $ipPermission.ToPort -eq 3389)

{

if (-not $NoRDP)

{

$delete = $false

$NoRDP = $true # Already defined don't have to create it again.

}

}

if ($ipPermission.IpProtocol -eq 'tcp' -and

$ipPermission.FromPort -eq 5985 -and $ipPermission.ToPort -eq 5985)

{

if (-not $NoPS)

{

$delete = $false

$NoPS = $true # Already defined don't have to create it again.

}

}

if ($ipPermission.IpProtocol -eq 'tcp' -and

$ipPermission.FromPort -eq 80 -and $ipPermission.ToPort -eq 80)

{

if (-not $NoHTTP)

{

$delete = $false

$NoHTTP = $true # Already defined don't have to create it again.

}

}

if ($ipPermission.IpProtocol -eq 'icmp' -and

$ipPermission.FromPort -eq -1 -and $ipPermission.ToPort -eq -1)

{

if (-not $NoICMP)

{

$delete = $false

$NoICMP = $true # Already defined don't have to create it again.

}

}

$update = $false

if ($ipPermission.IpRanges.Count -ne $SourceIPRange.Count)

{

$update = $true

}

else

{

foreach ($sourceIP in $SourceIPRange)

{

if (-not $ipPermission.IpRanges.Contains($sourceIP))

{

$update = $true

break

}

}

}

if ($delete -or $update)

{

Revoke-EC2SecurityGroupIngress -GroupName $SecurityGroupName `

-IpPermissions $ipPermission

if (-not $delete)

{

$ipPermission.IpRanges = $SourceIPRange

Grant-EC2SecurityGroupIngress -GroupName $SecurityGroupName `

-IpPermissions $ipPermission

}

}

}

}

if (-not $NoRDP)

{

Grant-EC2SecurityGroupIngress -GroupName $SecurityGroupName -IpPermissions `

@{IpProtocol = 'tcp'; FromPort = 3389; ToPort = 3389; IpRanges = $SourceIPRange}

}

if (-not $NoPS)

{

Grant-EC2SecurityGroupIngress -GroupName $SecurityGroupName -IpPermissions `

@{IpProtocol = 'tcp'; FromPort = 5985; ToPort = 5986; IpRanges = $SourceIPRange}

}

if (-not $NoHTTP)

{

Grant-EC2SecurityGroupIngress -GroupName $SecurityGroupName -IpPermissions `

@{IpProtocol = 'tcp'; FromPort = 80; ToPort = 80; IpRanges = $SourceIPRange}

}

if (-not $NoICMP)

{

Grant-EC2SecurityGroupIngress -GroupName $SecurityGroupName -IpPermissions `

@{IpProtocol = 'icmp'; FromPort = -1; ToPort = -1; IpRanges = $SourceIPRange}

}

Write-Host "Updated $SecurityGroupName IpRange to $SourceIPRange"

}

Sample usage of the script:

PS C:\temp> Update-WinEC2FireWallSource

Updated sg\_rdp\_ps\_http\_icmp IpRange to 174.21.164.151/32

PS C:\temp> Update-WinEC2FireWallSource -NoHTTP -NoICMP

Updated sg\_rdp\_ps\_http\_icmp IpRange to 174.21.164.151/32

PS C:\temp> Update-WinEC2FireWallSource -NoHTTP -NoICMP -SourceIPRange "10.0.0.0/8","172.31.0.0/16"

Updated sg\_rdp\_ps\_http\_icmp IpRange to 10.0.0.0/8 172.31.0.0/16

# References

1. [Security Groups](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html)
2. [Network ACL](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_ACLs.html)

You can find the code under “AWS” folder at <https://github.com/padisetty/Samples>.

Explore & Enjoy!

/Siva