Ayub Roti
Technical Solutions Engineer, Serianu Limited.
The Content Herein is automatically generated

Title: Network Segregation in AWS

Content Courtesy of
https://blog.rackspace.com/network-segregation-aws-2
There is an ongoing paradigm shift in how organizations address security (http://blog.rackspace.com/wp-admin/edit.php?tag=fanatical-support-for-aws) as they move their workloads from traditional data centers to AWS.

Traditional IT security has relied heavily on a strong network segmentation approach as the main security strategy to secure and protect an organization's resources.

Typically, the security governance structure of an organization has been siloed into distinct teams, each responsible for a different aspect of security: a firewall team, a router/switching team, a server team and the application team. Three of those four teams have been based on the fact that there are different physical devices (servers, firewalls, switches, etc.) managed by the different teams, with the application team assuming everything has been properly configured to meet security requirements.

This has led to unnecessary manual processes to ensure adequate coordination between the teams during deployments and change requests, all of which lead to long delays and are prone to human error.

As organizations move workloads from traditional data centers to AWS, the organization's IT and security professionals have had to completely change the way they address security. AWS has virtual constructs that are analogous to the traditional firewalls, routers, switches, servers, etc.; however, they are not separate physical devices. They are all constructs created, configured and terminated through RESTful APIs.

This potentially allows one team to control the firewall rules, the routing, the server configuration and the mode through which applications can be deployed via the AWS APIs. And since everything is controlled via the APIs, automation can be used to remove manual processes, thus speeding all deployments and reducing human error.

Traditional Network Security and Segregation


It's been common to illustrate traditional network security as a strong castle, with large, secure walls, and possibly a large moat full of water, all to keep the resources inside (application, data, etc.) safe from the outside world (the Internet). The only way to enter or leave this protected environment is through the front gates, which are closely guarded to prevent unwanted people from entering and protected resources from leaving it.

This approach relies heavily on a strong perimeter, with tight controls at the entrance gates to successfully protect internal resources. The inside of the castle is further segmented into isolated areas, with additional internal walls and moats, to further segment and protect the important resources.

This analogy is not far from how traditional network security has been implemented for the past 20-plus years. There are redundant external firewalls (the castle's external walls and moat) which only allow defined traffic to ingress and egress the environment from the untrusted networks (the Internet) via specific firewall rules that map to business requirements. The environments are further segregated by deploying internal firewalls, or by relying on ACLs and internal routers to further protect sensitive resources (e.g. application servers, database servers, etc.).

The below example of a logical network diagram illustrates these approaches. It is important to note that the approach relies on physical appliances and devices to obtain the required level of isolation and segregation.

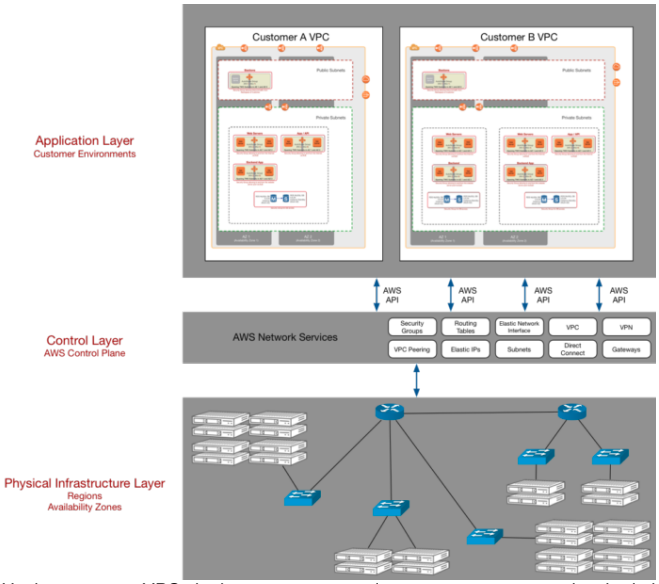
AWS Network Security and Segregation

The rest of this blog post will focus on the AWS security constructs that enable organizations to create the necessary security segregations. These constructs are not defined manually by manipulating physical devices that are controlled by different silos, but via AWS APIs. This allows organizations to collapse silos based on device ownership and facilitates needed automation to remove manual

processes, thus speeding all deployments and reducing human error.

Software Defined Networking

Before diving into AWS Network security, it's important to talk about SDN or Software Defined Networking. In simple terms, SDN is an architectural approach that allows the management of networks through an abstraction that decouples the system and makes decisions about where traffic is sent (control plane) from the underlying systems that forward traffic to the selected destination (data plane). This decoupling enables the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.



Having separate VPCs in the same account does not separate a top-level administrative control boundary for each VPC, thus, the security benefit is reduced (same account controls both VPCs and its resources).

There are operational reasons to have separate VPCs in the same AWS account. For example, having application workloads in a "workload" VPC that is peered to a "shared resources" VPC to support the workloads (e.g. active directory).

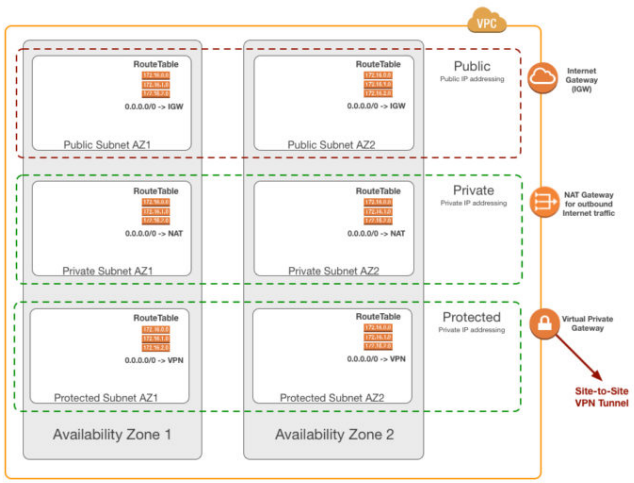
Subnets

After creating a VPC, the next logical construct is the subnet. Subnets in AWS are sub-networks within a VPC and are analogous to the subnets that are defined in traditional VLANs via 801.1q tagging. One can add one or more subnets in each availability zone; however, each subnet must reside exclusively within one AZ and cannot span AZs. AWS has three types of subnets:

1. Public – external subnets that have public IP addresses associated to servers and can be accessible from the Internet. They are analogous to traditional DMZ Networks.

2. Private – internal subnets that have only private IP addresses associated to server and are not accessible from the internet. They are able to access the Internet via NAT.

3. Protected – internal subnets that have only private IP addresses associated to the resources and are not accessible from the internet. They are NOT able to access the Internet.



It is important to ensure each type of subnet can span multiple AZs to achieve resiliency. Thus, if an environment has servers that have public facing servers and internal servers, at least two public subnets (in separate AZs) and two private subnets (in separate AZs) need to be created to support

public facing servers and internal servers in a high availability configuration.

Security Groups

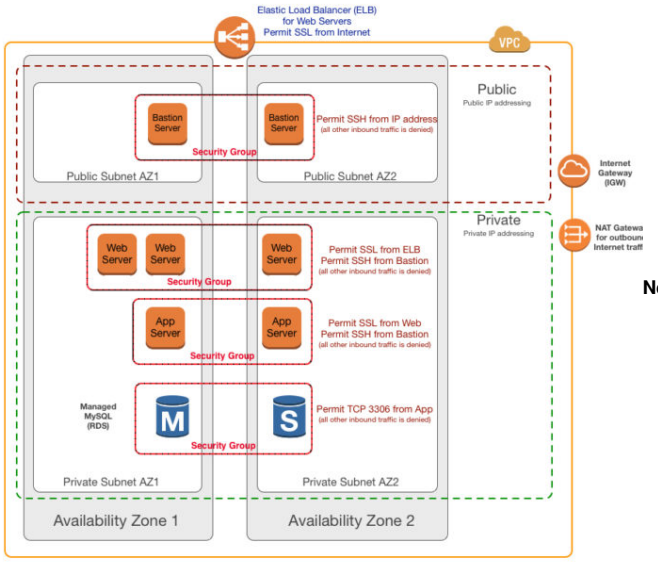
Everything up to this point has been focused on laying the necessary network components in a secure and highly-available manner. And up to this point, the AWS components have very closely corresponded to the traditional network approaches. Our discussion on security groups will deviate with this correlation.

As describes previously discussed, an in-line appliance has been used for all network-based firewall capabilities in traditional networks topologies. AWS security groups are not confined to single security appliances but span the entire VPC and are virtual constructs that actually tie into each hypervisor and network component of the AWS Physical Infrastructure layer.

A security group is a virtual stateful firewall for servers (ec2 instances) that enables the control of inbound and outbound traffic. When a security group is created to “protect” a group of servers in a VPC, the control-plane would create the necessary configurations to ensure all the instances, regardless of what AZ, or subnet, or hypervisors the servers reside to ensure the security group’s policy are adhered to.

It is important to note that security groups require explicit rules to permit all inbound traffic. Outbound traffic is open by default, but it too can be restricted.

Given that security groups create “containers for servers” with specific security profiles, security groups can be used as an isolation mechanism. Thus, instead of creating separate subnets for web servers, application servers and database servers, the security group can be leveraged to create the necessary isolation between “tiers”, and drastically simplify the required infrastructure. The diagram below depicts the isolation achieved by security groups.



Network Access

Lists (NACLs)

AWS also offers network ACLs with rules similar to your security groups. NACLs act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level. The following summarizes the basic differences between security groups and network ACLs:

Security Group

- Operates at the instance level (first layer of defense)
- Supports allow rules only
- Is statefull: Return traffic is automatically allowed, regardless of any rules
- We evaluate all rules before deciding whether to allow traffic
- Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on

Network ACL

- Operates at the subnet level
- Supports allow rules and deny rules
- Is stateless: Return traffic must be explicitly allowed by rules
- We process rules in number order when deciding whether to allow traffic
- Automatically applies to all instances in the subnets it’s associated with (backup layer of defense, so you don’t have to rely on someone specifying the security group)

As a general best practice, Rackspace advises customers to use security groups as their primary method of segmenting and securing workloads within AWS. While NACLs are typically more familiar to networking engineers, they often introduce complexity into AWS architectures.

Security groups provide more granular control, are stateful (therefore more intelligent in allowing appropriate traffic) and apply only to the instance level. By using NACLs as well as security groups,

one must consider all traffic in a stateless context (specifying inbound and outbound ports, including any ephemeral ports used by a given application) and these rules are applied at a subnet level; the “blast radius” or potential for impact when a NACL is incorrect or changed is significantly higher, without providing any tangible benefit over the use of a security group.

Rackspace and AWS recommend avoiding NACLs due to potential conflicts with security groups and performance degradation.

Conclusion

AWS has created a rich set of services, all of which are controlled via the control plane, which allows organizations to create secure and segmented networks that align with the needed security posture. Given that all the network constructs are accessible via the AWS APIs, the need for siloed teams is no longer relevant.

Previous governance policies and processes that relied on silos change to meet the demand for a cohesive multi-discipline team that can do all the work. Given that everything is accomplished via APIs, automation and auditing enable fast provisioning while adhering to the new governance policies and processes.

The following AWS network constructs allow for the needed segmentation:

Security Groups should be leveraged as the principle method for isolating and compartmentalizing workloads. Avoid creating subnets or separate VPC (with peering) for such traditional reasons.

Subnets should be used as a segmentation approach for workloads that have different routing requirements (e.g. Public vs. Private vs. Protected Subnets).

VPCs should be used as a segmentation approach (assuming VPC-Peering) when workloads that are controlled by separate business units in separate AWS accounts are dependent on each other and need the network connectivity without traversing the public network.

Additionally, there are operational reasons to have separate VPCs with peering in the same AWS account (e.g. application workloads in VPC peer to another VPC to access aentralized resources.

Title: Security - Amazon Virtual Private Cloud

Content Courtesy of
https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html

Security

Amazon Virtual Private Cloud provides features that you can use to increase and monitor the security for your virtual private cloud (VPC):

- Security groups â Act as a firewall for associated Amazon EC2 instances, controlling both inbound and outbound traffic at the instance level
- Network access control lists (ACLs) â Act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level
- Flow logs â Capture information about the IP traffic going to and from network interfaces in your VPC

When you launch an instance in a VPC, you can associate one or more security groups that you've created. Each instance in your VPC could belong to a different set of security groups. If you don't specify a security group when you launch an instance, the instance automatically belongs to the default security group for the VPC. For more information about security groups, see Security Groups for Your VPC (VPC_SecurityGroups.html)

You can secure your VPC instances using only security groups; however, you can add network ACLs as an additional layer of defense. For more information, see Network ACLs (vpc-network-acls.html).

You can monitor the accepted and rejected IP traffic going to and from your instances by creating a flow log for a VPC, subnet, or individual network interface. Flow log data is published to CloudWatch Logs, and can help you diagnose overly restrictive or overly permissive security group and network ACL rules. For more information, see VPC Flow Logs (flow-logs.html).

You can use AWS Identity and Access Management to control who in your organization has permission to create and manage security groups, network ACLs and flow logs. For example, you can give only your network administrators that permission, but not personnel who only need to launch instances. For more information, see Controlling Access to Amazon VPC Resources (VPC_IAM.html).

Amazon security groups and network ACLs don't filter traffic to or from link-local addresses (`169.254.0.0/16`) or AWS-reserved IPv4 addressesâthese are the first four IPv4 addresses of the subnet (including the Amazon DNS server address for the VPC). Similarly, flow logs do not capture IP traffic to or from these addresses. These addresses support the services: Domain Name Services (DNS), Dynamic Host Configuration Protocol (DHCP), Amazon EC2 instance metadata, Key Management Server (KMSâlicense management for Windows instances), and routing in the subnet. You can implement additional firewall solutions in your instances to block network communication with link-local addresses.

Comparison of Security Groups and Network ACLs

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

Security Group	Network ACL
Operates at the instance level	Operates at the subnet level
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
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (therefore, an additional layer of defense if the security group rules are too permissive)

The following diagram illustrates the layers of security provided by security groups and network ACLs. For example, traffic from an Internet gateway is routed to the appropriate subnet using the routes in the routing table. The rules of the network ACL associated with the subnet control which traffic is allowed to the subnet. The rules of the security group associated with an instance control which traffic is allowed to the instance.

Title: AWS Security Best Practices

Content Courtesy of
https://aws.amazon.com/whitepapers/aws-security-best-practices/
This whitepaper provides security best practices that will help you define your Information Security Management System (ISMS) and build a set of security policies and processes for your organization so you can protect your data and assets in the AWS Cloud. It also provides an overview of different security topics such as identifying, categorizing and protecting your assets on AWS, managing access to AWS resources using accounts, users and groups and suggesting ways you can secure your data, your operating systems and applications and overall infrastructure in the cloud. **Download Whitepaper (//d1.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf)**

Title: How can I automate VPC network segmentation using AWS Transit Gateway and Aviatrix? | Aviatrix Answers

Content Courtesy of
https://www.aviatrix.com/answers/how-can-i-automate-vpc-network-segmentation-using-aws-transit-gateway-and-aviatrix/

How can I automate VPC network segmentation using AWS Transit Gateway (TGW) and Aviatrix?

Network segmentation is a highly effective strategy to limit the impact of network intrusion. This article delves into ways to simplify network segmentation and how Aviatrix Network Security Domains along with AWS Transit Gateway (TGW) provide a solution for secure network traffic in multi-VPC environments.

Cloud Network Segmentation: why and how?

First, let’s look at why network segmentation is important to network security. Going back to traditional data center networks, segmentation was primarily done via a DMZ or perimeter which had firewall devices. The issue here was that once a bad actor got into the network, they gained wide reach throughout the “private” network.

Segmentation approaches called for partitions of a numerous smaller networks, reducing the reachability for intruders, minimizing the damage they could do. This type of segmentation involves developing and enforcing a set of rules to manage the traffic between segments. The best practice was to find a manageable level to which you could segment your network. Go too fine-grained it’ll be near impossible to manage, go too coarse and your blast radius may be too large for one or more segments. Beyond the practical considerations, the best practices include looking at grouping networks by their sensitivity level, for example production networks vs. test networks, are of different levels usually due to live customer data being in the production systems.

Up until the concept of domains was introduced by AWS at reinvent 2018, network segmentation in AWS was mainly achieved by using Accounts, VPCs, Security Groups, and Access Control Lists (ACLs). Whereas most cloud engineers have found ACLs to be impractical to manage, Security Groups are used widely to provide instance level security within a VPC. At a VPC level, peering and

transit hubs have implemented connectivity but with inconsistent approaches to segmentation. As the number of VPC networks and cloud security concerns has grown with the mass adoption of public clouds, the routing between VPCs has become increasingly complex and a bigger and bigger problem which needs a solution. There simply has not existed an easy way to group VPCs together.

AWS Transit Gateway (TGW) route domains are a concept which simplify the segmentation of groups of VPCs and VPNs. It is a centralized approach to achieving network segmentation, reducing unauthorized lateral movement.

Benefits of Aviatrix Security Domains

Aviatrix Security Domains was introduced in conjunction with AWS Transit Gateway (TGW) to solve the problem of VPC level segmentation. Security Domains are an instantiation of the AWS Transit Gateway (TGW) route domain concept. It turns a concept into an object, ideal for network segmentation. A Security Domain is an AVX controller-enforced network of member VPCs attached to the same route table. Member VPCs have connectivity to each other. VPCs outside of the domain cannot connect. This dramatically simplifies the approach to securing large VPC environments. As a centralized system, the AVX Controller spans multi-region and on-prem environments.

Along with Security Domains, we've introduced Connection Policies.

Connection Policies are AVX controller-enforced cross-Security Domain connectivity. Again leveraging the innovation of AWS, this uses AWS Transit Gateway (TGW) route table propagation. Now Cloud and Security teams can have a common framework, with visibility and control of what is connected and why. Connection Policies are human-readable and again, centrally deployed. Audit and compliance reporting at the VPC level are now more readily available. Lastly, defining VPC level security with a few centralized rules is far simpler to conceptualize and far easier to implement.

Leading us to the Zero Trust Cloud Architecture

Combining Security Domains, Connection Policies, and AWS Transit Gateway (TGW) produces a simple yet powerful architecture for security and compliance of VPC networks. A Zero Trust Cloud Architecture helps to:

- Ensure the VPC network meets security best practices
- Limit lateral movement during a breach
- Minimize blast radius due to misconfigurations
- Avoid delays due to tedious, manual configuration

Summing it up

Aviatrix believes that Security Domains and Connection Policies for network segmentation are important keys to securing cloud-based environments.

Click here to download and print this 2-page PDF Transit Gateway and Zero Trust Architecture quick reference guide (/documents/Aviatrix-Transit-Gateway-Reference-Card.pdf).

Title: 6.1 Understanding Security Zoning and Network Segmentation - AWS Cloud Security [Video]

Content Courtesy of
https://www.oreilly.com/library/view/aws-cloud-security/9780135174784/ACS1_03_06_01.html

Title: AWS Reference Architecture - Segmentation Firewall for Single AZ VPCs | Barracuda Campus

Content Courtesy of
https://campus.barracuda.com/product/cloudgenfirewall/doc/54263936/aws-reference-architecture-segmentation-firewall-for-single-az-vpcs/

A NextGen Firewall F with multiple network interfaces can be used as a segmentation firewall for your private subnets in the VPC. Traffic passing between the private subnets is routed through the firewall, where you can apply security policies and visualize traffic in real time between the subnets. To be able to route the traffic over the firewall, the standard route for internal VPC traffic must be circumvented. By default, all traffic within the VPC is routed over the default gateway. This route cannot be overridden by other more specific routes, nor can it be changed to use the firewall as the gateway instead. Using a combination of a firewall instance with multiple network interfaces and

adding a route on the client instances allows you to use the F-Series Firewall as a segmentation firewall in AWS.

Use a segmentation firewall to enforce access policies and monitor traffic passing between the subnets. When compared with an AWS native solution, a NextGen Firewall is vastly superior regarding the depth at which both traffic can be inspected and security policies applied. In addition, NextGen Admin also provides real-time traffic visibility, and the Firewall Live and History pages allow quick, fine-grained access to all the traffic currently passing through the firewall.

For the firewall, select the instance type according to the number of network interfaces. The number of network interfaces is the number of private subnets plus one for the public subnet. At least three network interfaces are required. The instance type must support at least three network interfaces: one for the public subnet and two for the private subnets.

segmentation.png

Use Cases for a Multi-NIC Segmentation Firewall

A NextGen Firewall Segmentation is deployed like an internal firewall for applications moved to AWS using lift-and-shift migrations.

Limitations

- All resources must be in a single Availability Zone.
- The number of private subnets is limited by the number of network interfaces supported by the instance type. So if the firewall supports three network interfaces, two private subnets can be connected. The primary network interface is used for external connectivity.
- A route must be added to the client instances in the private subnets. The default route over the gateway in the subnet bypasses the firewall. This can be stopped via Security Groups.
- Cannot be deployed as a High Availability Cluster.
- Connecting to subnets in other Availability Zones requires use of source NAT on the matching access rule.

Deploying a Segmentation Firewall via CloudFormation Template

It is recommended to deploy the Segmentation Firewall via a CloudFormation template. The template deploys one firewall per public subnet that is automatically joined into the High Availability Cluster. The route table associated with the private subnets is configured to use the active firewall as the outbound gateway. This template only deploys the AWS infrastructure. The NextGen Firewall must be configured manually.

1. Create an IAM role for the firewall cluster. For step-by-step instructions, see How to Create an IAM Role for an F-Series Firewall in AWS (https://confluence.campus.cuda-inc.com/techlib/display/NGFv71/How+to+Create+an+IAM+Role+for+an+F-Series+Firewall+in+AWS).
2. Download the **NGF_Segmentation.json** template and parameter file from the Barracuda Network GitHub account: https://github.com/barracudanetworks/ngf-aws-templates (https://github.com/barracudanetworks/ngf-aws-templates).
3. (optional) Download the Network Diagram containing the IP addresses (/resources/attachments/image/54263936/segmentation_topology.png?v=2).
4. Accept the Software Terms for the **Barracuda NextGen Firewall PAYG** or **BYOL** image in the AWS Marketplace.
5. Create a parameter template file containing your parameters values.
6. Deploy the template via AWS CLI or AWS console.

```
aws cloudformation create-stack  --stack-name  "YOUR_STACK_NAME" --template-body YOUR_S3_BUCKET/NGF_Segmentation.json --parameter YOUR_S3_B
```

During deployment, the following resources are created by the template:

- One VPC with one public and two private subnets in the same AZ.
- One Barracuda NextGen Firewall with three ENIs.

For step-by-step instructions on how to deploy a CloudFormation template, see How to Deploy an F-Series Firewall in AWS via CloudFormation Template (/doc/53676264/).

(Alternative) Deploying a Segmentation Firewall via AWS Console

Complete the following configuration steps to deploy the NextGen Firewall F as a segmentation firewall. For more detailed descriptions, follow the links for step-by-step instructions.

For step-by-step instruction, see How to Deploy an F-Series Firewall in AWS via Web Portal (/doc/53248279/).

Adding Additional Network Interfaces for Each Private Subnet

The firewall must have a network interface in each private subnet. Create an AWS elastic network interface (ENI) for each private subnet in your VPC. The private IP address must be set explicitly to be able to configure the network interface statically. Also, disable the source/destination check for each interface to be able to process traffic with a destination address not matching the private IP of the network interface. Before attaching the ENIs to the firewall, shut the firewall instance down.

Attach the network interfaces. After starting the firewall, configure the new network interfaces and add the required direct attached routes and virtual server IP addresses.

segmentation_4.png

For step-by-step instructions, see How to Add AWS Elastic Network Interfaces to a Firewall Instance

(/doc/54263910/).

Route Table for Private Subnets

For each private subnet, a dedicated AWS route table handles all traffic with destinations outside the VPC. Associate the subnet with the route table and create a default route with the network interface of the firewall in this subnet as the target.

segmentation_5.png

For step-by-step instructions, see How to Configure AWS Route Tables for Firewalls with Multiple Network Interfaces (/doc/54264258/).

Deploying Instances to Use the Firewall as Default Gateway

It is not currently possible to configure the AWS route table to send traffic between two subnets through the firewall instance. By default, each route table includes a static route for the VPC pointing to the AWS gateway of the subnet. This route cannot be overridden by a more specific route, nor can it be deleted. To send traffic via the firewall, add a route directly on the instance. The route can be added either manually after the instance has been deployed, or automatically in the **User data** section.

AWS Console (Linux Instances Only)

Add the routes to **User data** field of the **Advanced Details** section.

segmentation_client_userdata_01.png

CloudFormation (Linux Instances Only)

Add the definition for the routes in the **UserData** section of the CloudFormation template. If multiple private subnets are used, more than one route may be required.

```
"UserData": { "Fn::Base64": { "Fn::Join": [ "", [ "#!/bin/bash\n\n", "/sbin/route add -net 10.100.1.0/16 gw 10.100.2
```

Manually (Linux Instances Only)

Log into the instance via SSH, and with root privileges enter:

segmentation_client_userdata_02.png

The route is now in the route table. Enter `route -n` to list the routes:

segmentation_client_userdata_03.png

Firewall Service Configuration

Now that the routing and setup in AWS is complete, access rules must be configured to apply your security policies to the traffic passing between the VPC subnets:

- **Network objects** – Create network objects for the VPC, for each subnet, and for individual instances. For more information, see Network Objects (/doc/53248547/).
- **Access rules** – By default, all connections are blocked. Create access rules for each service the instances are allowed to access. Use the **FIREWALL > Live** and **FIREWALL > History** pages to verify which rule matches and which traffic is blocked. For more information, see Live Page (/doc/53248322/) and History Page (/doc/53248395/).

Access rules allowing the backend instances access to the Internet must use the **Dynamic NAT** connection objects to rewrite the source IP of the packets to the IP address of the firewall.

Title: Segmentation, Isolation and Accreditation… oh my! - Trend Micro

Content Courtesy of
https://www.trendmicro.com/aws/cloud-security-segmentation-isolation-and-accreditation/
This post originally appeared at http://blog.trendmicro.com/cloud-security-segmentation-isolation-and-accreditation-oh-my/ (http://blog.trendmicro.com/cloud-security-segmentation-isolation-and-accreditation-oh-my/)

There was a time, not too long ago, where simple perimeter firewalls were the state of the art. It was too complex and costly to implement proper segmentation for servers.

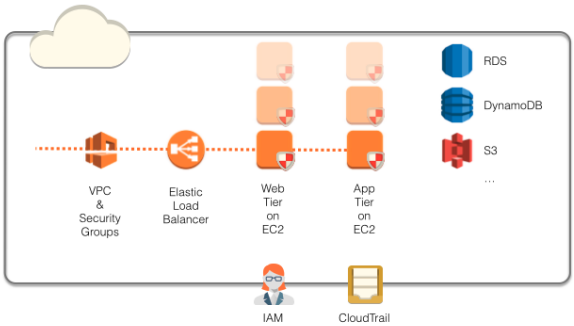
All that's changed with the advent of software-defined networking and advances in virtualization and host-based software. Toto, I've got a feeling we're not in Kansas anymore! We can now affordably apply segmentation and isolation best practices for security as discussed in this Gartner paper (https://resources.trendmicro.com/Gartner-NL-AWS-Security-Best-Practices.html).

Where do I start?

Amazon Web Services provides a powerful and user-friendly means of applying basic network ACLs (Access Control Lists). ACLs are just a fancy name for controlling which network ports instances can speak to each other and the Internet on.

By default AWS operates in deny all, meaning no ports are open unless you specifically request them. Ideally you want to open the minimum of ports needed, and only to the resources that need them. For instance you could open a Web Server to port 80/443 for all of the Internet, but you shouldn't open its RDP port to the Internet…or at all if you can help it.

Like the wizard, AWS has a clever trick up its sleeve to make this easier. Say I wanted to protect a 3-tier application with web servers, application servers and back end data services:



(https://www.trendmicro.com/aws/wp-content/uploads/2015/11/Cloud-Security-blog-5-1.png)

AWS allows you to define Security Groups, which are like templates for multiple instances of the same type. To make it easier to keep the rule set small, you can link Security Groups together. For example, allowing only traffic over 443 to the App tier from instances in the Web Server tier. Sounds simple—and it is—but it's also incredibly powerful.

If you combine that with VPC (Virtual Private Clouds), you can apply rich segmentation and isolation policies for a wide variety of architectures. There are plenty of samples in the architectures (https://aws.amazon.com/architecture/) section on the AWS site to help you get to the Emerald City.

That's enough, right?

Not quite. As we discussed in our last post (http://blog.trendmicro.com/cloud-security-to-patch-or-not-to-patch/), it often isn't enough to simply segment and open only the required ports. Threats like Shellshock, Heartbleed and others occur over these legitimate ports. To keep these winged monkeys out of your instances, you have to look deep into the packets of data from the network. A software based intrusion prevention system ensures that the traffic to and from your instances is free from malicious intent.

Using an IPS running on your EC2 instances—like Deep Security (//)—means that your security will scale automatically with your workload. This consistency reduces your operational burden with a nice clean scaling model.

In addition to the protection of an IPS, many compliance frameworks require detailed logs of all network activity. While VPC's have flow logs, they do not provide sufficient detail for passing compliance audits like PCI-DSS. You may want to consider a 3rd party firewall if you need detailed logging and accreditation. While providing excellent security, AWS Security Groups have not yet been certified by ICSA Labs.

Tying it together…

Before you go far down the segmentation and isolation **red-bricked road**, make sure you have a plan for how your network security will be monitored and audited.

In an earlier post, another best practice highlighted the importance of ongoing monitoring (http://blog.trendmicro.com/cloud-security-you-cant-protect-what-you-cant-see/). You also need to watch for intentional or unintentional configuration changes and constantly reassess your security architecture.

Interested in learning more? Read the Gartner paper (https://resources.trendmicro.com/Gartner-NL-AWS-Security-Best-Practices.html) on best practices for securing AWS workloads.

If you have questions or comments, please post them below or follow me on Twitter: @justin_foster (https://twitter.com/justin_foster).

Title: AWS Security Architecture - Overview

Content Courtesy of
https://www.slideshare.net/ignorantia/aws-security-architecture-overview

Title: 101 AWS Security Tips & Quotes, Part 3: Best Practices for Using Security Groups in AWS - Security Boulevard

Content Courtesy of
https://securityboulevard.com/2018/07/101-aws-security-tips-quotes-part-3-best-practices-for-using-security-groups-in-aws/

Here's the third blog post in our 4-part series of AWS Security Tips and Quotes, which is designed to help you evolve and strengthen your organization's security, building on a proactive, comprehensive security strategy.

So far we've covered:

Today the spotlight falls on *Best Practices for Using Security Groups in AWS*, (and in the final installment, Part 4, we'll deal with *AWS Security Best Practices)*.

Best Practices for Using Security Groups in AWS

1. Security groups are the central component of AWS firewalls.

"Amazon offers a virtual firewall facility for filtering the traffic that crosses your cloud network segment; but the way that AWS firewalls are managed differs slightly from the approach used by traditional firewalls. The central component of AWS firewalls is the "security group," which is essentially what other firewall vendors would call a policy (i.e., a collection of rules). However, there are key differences between security groups and traditional firewall policies that need to be understood.

"First, in AWS, there is no 'action' in the rule stating whether the traffic is allowed or dropped. This is because all rules in AWS are positive and always allow the specified traffic — unlike traditional firewall rules.

"Second, AWS rules let you specify the traffic source, or the traffic destination — but not both on the same rule. For Inbound rules, there is a source that states where the traffic comes from, but no destination telling it where to go. For Outbound rules, it the other way around: you can specify the destination but not the source. The reason for this is that the AWS security group always sets the unspecified side (source or destination, as the case may be) as the instance to which the security group is applied.

"AWS is flexible in how it allows you to apply these rules. Single security groups can be applied to multiple instances, in the same way that you can apply a traditional security policy to multiple firewalls. AWS also allows you to do the reverse: apply multiple security groups to a single instance, meaning that the instance inherits the rules from all the security groups that are associated with it. This is one of the unique features of the Amazon offering, allowing you to create security groups for specific functions or operating systems, and then mix and match them to suit your business' needs."

— Avishai Wool, *A Guide to AWS Security* (http://www.infosecisland.com/blogview/24642-A-Guide-to-AWS-Security.html)**, Infosec Island; Twitter: @InfosecIsland** (https://twitter.com/infosecisland)

2. Enable and configure AWS PVC Flow Logs.

"Enable AWS VPC Flow Logs for your VPC or Subnet or ENI level. AWS VPC flow logs can be configured to capture both accept and reject entries flowing through the ENI and Security groups of the EC2, ELB, and some additional services. These VPC Flow log entries can be scanned to detect attack patterns, alert abnormal activities and information flow inside the VPC, and provide valuable insights to the SOC/MS team operations."

— Harish Ganesan, *27 Best Practice Tips on Amazon Web Services Security Groups* (https://8kmiles.com/blog/27-best-practice-tips-on-amazon-web-services-security-groups/)**, 8K Miles; Twitter: @8KMiles** (https://twitter.com/8KMiles)

3. Use current managed policies.

"Earlier, only Identity-based (IAM) inline policies were available (http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html).

Managed policies came later. So not all old AWS cloud best practices that existed during the inline policies era might hold good in the present day. So, it is recommended that you use managed policies that is available now. With managed policies you can manage permissions from a central place rather than having it attached directly to users. It also enables you to properly categorize policies and reuse them. Updating permissions also becomes easier when a single *managed policy* is attached to multiple users. Plus, in managed policies you can add up to 10 managed policies to a user, role, or group. The size of each managed policy, however, cannot exceed 5,120 characters."

— Jayashree Hegde, *AWS Cloud Security Think Tank: 5 Reasons Why You Should Question Your Old Practices* (https://www.botmetric.com/blog/aws-cloud-security-question-old-practices/)**,**

Botmetric; Twitter: @BotmetricHQ (https://twitter.com/BotmetricHQ)

4. Look at all the security groups associated with each instance for a complete picture of what touches regulated data.

“AWS is clearly set up to enable you to look at each security group in turn and review it, so that you can look over the rules, compare them to your corporate policies and regulatory requirements to see if they match. As such, this is an easy and resource-efficient option.

However, it comes with two caveats. First, you don’t have any context if you look at an individual security group on its own. You don’t even know if each security group has any instances associated with it — it might be ‘free floating’ — which is perfectly possible in AWS.

“Second, AWS security groups can be mixed and matched, so you can have multiple security groups associated with individual instances or servers. To truly understand your security posture, you need to look at all of the security groups that are associated with each instance. Otherwise you will only have a partial view of what traffic is allowed into or out of instances that touch regulated data.”

— Avishai Wool, *Tips for auditing your AWS security policies, the right way* (https://blog.algosec.com/2017/08/tips-auditing-aws-security-policies-right-way.html)**, AlgoSec; Twitter: @AlgoSec** (https://twitter.com/AlgoSec)

5. Categorize your security groups.

“It is good to categorize your security groups. For example all DB and web server connection ports in one security group, all the media connection ports under one security group, all the third-party connection ports in one security group, and all the office-specific ports under one security group. Categorization helps you manage your different sets of connections efficiently so you do not mess up your security groups whenever you need to change some things for a specific port.”

— Chandan Mishra, *10 Tips to secure your EC2 instance on AWS using Security Groups* (https://www.linkedin.com/pulse/10-tips-secure-your-ec2-instance-aws-using-security-groups-mishra/)**, LinkedIn**

6. Minimize the number of discrete security groups.

“Account compromise can come from a variety of sources, one of which is misconfiguration of a security group. By minimizing the number of discrete security groups, enterprises can reduce the risk of misconfiguring an account.”

— Sekhar Sarukkai, *Locking-in the Cloud: Seven Best Practices for AWS* (https://blog.cloudsecurityalliance.org/2017/07/06/locking-cloud-seven-best-practices-aws/)**, Cloud Security Alliance; Twitter: @cloudsa** (https://twitter.com/cloudsa)

7. Use proper naming conventions for your security groups.

“Have proper naming conventions for the Amazon Web Services security group. The naming convention should follow enterprise standards. For example it can follow the notation: ‘AWS Region+ Environment Code+ OS Type+ Tier+ Application Code’

Security Group Name – EU-P-LWA001

AWS Region (2 char) = EU, VA, CA etc

Environment Code (1 Char) = P-Production , Q-QA, T-testing, D-Development etc

OS Type (1 Char)= L -Linux, W-Windows etc

Tier (1 Char)= W-Web, A-App, C-Cache, D-DB etc

Application Code (4 Chars) = A001

“We have been using Amazon Web Services from 2008 and found over the years managing the security groups in multiple environments is itself a huge task. Proper naming conventions from beginning is a simple practice, but will make your AWS journey manageable.”

— Harish Ganesan, *AWS Security Groups – 27 Best Practice Tips* (https://www.linkedin.com/pulse/aws-security-groups-27-best-practice-tips-harish-ganesan/)**, LinkedIn**

8. Attach policies to groups, rather than individual users.

“As a best practice (http://docs.aws.amazon.com/IAM/latest/UserGuide/IAMBestPractices.html#use-groups-for-permissions), attach policies to groups instead of to individual users. If an individual user has a policy, make sure you understand why that user needs the policy.”

— *AWS Security Audit Guidelines* (https://docs.aws.amazon.com/general/latest/gr/aws-security-audit-guide.html#aws-security-audit-review-policy-tips)**, AWS; Twitter: @awscloud** (https://twitter.com/awscloud)

9. Don’t open ports for 0.0.0.0/0 unless specifically required.

“Allowing incoming access by opening up ports for 0.0.0.0/0 in security groups is the most common mistake made by professionals when provisioning resources. Users end up opening their cloud networks and exposing their cloud resources and data to outside threats.”

— Eyal Posener, *Amazon Security Groups – 5 Important Best Practices for Your To-Do List* (https://www.stratoscale.com/blog/compute/aws-security-groups-5-best-practices/)**, Stratoscale; Twitter: @Stratoscale** (https://twitter.com/stratoscale)

10. Grant access from specific Amazon EC2 security groups or particular IP addresses for any security group rule.

“Amazon Relational Database Service (Amazon RDS) has security group configurations which are examined explicitly, and a warning is released if a rule for security group grants or is probable to grant excessive access to your database. For any security group rule, it is recommended that access from only certain Amazon Elastic Compute Cloud (Amazon EC2) security groups or from a particular IP address should be granted.”

— Monali Ghosh, *AWS Security Audit and Best Practices* (https://www.centilytics.com/aws-security-audit-best-practices/)**, Centilytics; Twitter: @centilytics** (https://twitter.com/centilytics)

11. Create a default security group for new instances.

“When you create a new instance through Salt, it requires that you specify a default security group. In order to avoid any security breach, create a default group that only allows SSH.”

— *AWS Security groups best practices* (https://trackit.io/aws-security-groups-best-practices/)**, TrackIt; Twitter: @TrackItCloud** (https://twitter.com/TrackItCloud)

12. Restrict access to EC2 security groups.

“This will prevent DoS, brute-force, and man-in-the-middle attacks. Additionally, assign your identity and access management (IAM) policies and permissions to specific roles/groups instead of specific users.”

— Sekhar Sarukkai, *10 ways to secure sensitive information on AWS* (https://www.cyberscoop.com/skyhigh-networks-sekhar-sarukkai-aws-op-ed/)**, CyberScoop; Twitter: @CyberScoopNews** (https://twitter.com/CyberScoopNews)

13. Create individual IAM users, then assign them to groups, and then attach policies to groups.

“You should always create individual IAM **users**, add them to IAM **groups** and attach IAM **policies** to these groups. Policies define group’s **permissions** allowing or denying access to AWS resources. When attached to groups (rather than users) they make it easy to fine-tune a policy to affect a group and all relevant users at once.”

— Evgeny Goldin, *AWS IAM Best Practices* (https://minops.com/blog/2015/01/aws-iam-best-practices/)**, MinOps; Twitter: @MinOpsInc** (https://twitter.com/MinOpsInc)

14. Set up security groups around access points rather than specific AWS resources.

“Where possible, set up security groups focused around access points rather than specific AWS resources. In other words, create a security group for the IP addresses associated with Company Branch A, Company Branch B, etc. rather than a security group specifying which IP addresses can access EC2_A, EC2_B, etc. That way, you only have to update your traffic rules in one place (for example, if the IP address of Company Branch A changed, you only have to update the Company Branch A security group instead of searching every EC2_A, EC2_B, etc. security group and updating the traffic rules in every place where Company Branch A’s info appears).”

— Julia Stefan, *Best Practices, Tips & Tricks for Managing Your Security Architecture* (http://diamondstream.com/aws-data-security-basics-part-3-putting/)**, DiamondStream; Twitter: @DiamondStream1** (https://twitter.com/DiamondStream1)

15. Be cautious when using multiple security groups.

“**Use caution with multiple security groups.** You can apply multiple security groups to a single EC2 instance or apply a single security group to multiple EC2 instances. System administrators often make changes to the state of the ports; however, when multiple security groups are applied to one instance, there is a higher chance of overlapping security rules. For example, security group A opened port 80 to the entire Internet. Meanwhile, security group B opened port 80 to one IP address. If you assign these two security groups to an EC2 instance and modify either, issues may occur. If you remove port 80 rules from security group A, security group B still has port 80 open. It’s easier to find these mistakes when there is a small number of EC2 instance or security groups. A larger number makes finding mistakes more difficult.”

— Ruihai Fang, *Stand Your Cloud: A Series on Securing AWS* (https://www.bishopfox.com/blog/2015/02/stand-cloud-series-securing-aws/)**, Bishop Fox; Twitter: @bishopfox** (https://twitter.com/bishopfox)

16. Use IAM to control permissions for creating and managing security groups, network ACLs, and flow logs.

“You can use AWS Identity and Access Management to control who in your organization has permission to create and manage security groups, network ACLs, and flow logs. For example, you can give only your network administrators that permission, but not personnel who only need to launch instances. For more information, see Controlling Access to Amazon VPC Resources (https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_IAM.html).

“Amazon security groups and network ACLs don’t filter traffic to or from link-local addresses (169.254.0.0/16) or AWS-reserved IPv4 addresses — these are the first four IPv4 addresses of the subnet (including the Amazon DNS server address for the VPC). Similarly, flow logs do not capture IP

traffic to or from these addresses. These addresses support the services: Domain Name Services (DNS), Dynamic Host Configuration Protocol (DHCP), Amazon EC2 instance metadata, Key Management Server (KMS — license management for Windows instances), and routing in the subnet. You can implement additional firewall solutions in your instances to block network communication with link-local addresses."

— *Security* (https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Security.html#VPC_Security_Comparison)**, AWS; Twitter: @awscloud** (https://twitter.com/awscloud)

17. Create a security group for databases opening at port 3306, but don't allow access to the internet at large.

"You'll want to create an AWS security group for databases which opens port 3306, but don't allow access to the internet at large. Only to your AWS defined webserver security group. You may also decide to use a single box and security group which allows port 22 (ssh) from the internet at large. All ssh connections will then come in through that box, and internal security groups (database & webserver groups) should only allow port 22 connections from that security group.

"When you setup replication, you'll be creating users and granting privileges. You'll need to grant to the wildcard '%' hostname designation as your internal and external IPs will change each time a server dies. This is safe since you expose your database server port 3306 *only* to other AWS security groups, and no internet hosts.

"You may also decide to use an encrypted filesystem for your database mount point, your database backups, and/or your entire filesystem. Be particularly careful of your most sensitive data. If compliance requirements dictate, choose to store very sensitive data outside of the cloud and secure network connections to incorporate it into application pages."

— Sean Hull, *Deploying MySQL on Amazon EC2 – 8 Best Practices* (https://www.iheavy.com/2011/02/21/deploying-mysql-on-amazon-ec2-best-practices/)**, Scalable Startups; Twitter: @hullsean** (https://twitter.com/@hullsean)

18. Keep your naming conventions under wraps.

"For security in depth, make sure your Amazon Web Services security groups naming convention is not self explanatory and also make sure your naming standards stay internal. Example: AWS security group named UbuntuWebCRMProd is self explanatory for hackers that it is a Production CRM web tier running on ubuntu OS. Have an automated program detecting AWS security groups with Regex Pattern scanning of AWS SG assets periodically for information revealing names and alert the SOC/Managed service teams."

— Harish Ganesan, *27 Best Practice Tips on Amazon Web Services Security Groups* (http://harish11g.blogspot.com/2015/06/best-practices-tips-on-amazon-web-services-security-groups-aws-security-managed-services.html)**, Cloud, Big Data, and Mobile; Twitter: @harish11g** (https://twitter.com/harish11g)

19. Close unnecessary system ports.

"EC2 instances can be secured with 'Security Groups.' This is a basic firewall that allows you to open and block network access to your EC2 server.

"Security Groups let you limit inbound and outbound connections for specified protocols (UDP and TCP) for common system services (HTTP, DNS, IMAP, POP, MYSQL, etc.) limited by IP ranges, your IP, or anywhere.

"On Linux Servers, some of the most common services are SSH, HTTP, HTTPS, and MySQL. Let's see how we can secure access to those services using EC2 Security Groups.

"From the AWS Management console, we can add and edit Security Group filtering rules. Let's see how to do it:

1. Login to AWS Management Console.
2. Click on your left menu: Security Groups.
3. Click on your instance security group.
4. Click on Edit to add new rules or customize existing ones."

— Chris Ueland, *How to secure your EC2 virtual servers* (https://www.scalescale.com/tips/aws/secure-ec2-virtual-servers/)**, ScaleScale; Twitter: @scalescalehq** (https://twitter.com/scalescalehq)

20. Regularly revisit your security groups to optimize rules.

"Many organizations begin their cloud journey to AWS by moving a few applications to demonstrate the power and flexibility of AWS. This initial application architecture includes building security groups that control the network ports, protocols, and IP addresses that govern access and traffic to their AWS Virtual Private Cloud (https://aws.amazon.com/vpc/) (VPC). When the architecture process is complete and an application is fully functional, some organizations forget to revisit their security groups to optimize rules and help ensure the appropriate level of governance and compliance. Not optimizing security groups can create less-than-optimal security, with ports open that may not be needed or source IP ranges set that are broader than required."

— Guy Denney, *How to Visualize and Refine Your Network's Security by Adding Security Group IDs to Your VPC Flow Logs* (https://aws.amazon.com/blogs/security/how-to-visualize-and-

refine-your-networks-security-by-adding-security-group-ids-to-your-vpc-flow-logs/)**, AWS; Twitter: @awscloud** (https://twitter.com/awscloud)

21. Periodically audit security groups to identify those not following the established naming conventions.

"Periodically detect, alert, or delete AWS Security groups not following the organization naming standards strictly. Also have an automated program doing this as part of your SOC/Managed service operations. Once you have this stricter control implemented, then things will fall in line automatically."

— Harish Ganesan, *27 Best Practice Tips on Amazon Web Services Security Groups* (https://8kmiles.com/blog/27-best-practice-tips-on-amazon-web-services-security-groups/)**, 8K Miles; Twitter: @8KMiles** (https://twitter.com/8KMiles)

*** This is a Security Bloggers Network syndicated blog from Blog – Threat Stack (https://www.threatstack.com) authored by Bob Allin (https://securityboulevard.com/author/bob-allin/). Read the original post at: https://www.threatstack.com/blog/101-aws-security-tips-quotes-part-3-best-practices-for-using-security-groups-in-aws (https://www.threatstack.com/blog/101-aws-security-tips-quotes-part-3-best-practices-for-using-security-groups-in-aws)

Title: Security Policy Orchestration for Amazon Web Services (AWS)

Content Courtesy of
https://www.tufin.com/supported-devices-and-platforms/aws
Cloud adoption is not only commonplace but increasing faster than anticipated. According to a 2018 ESG white paper titled "Network Security Operations Transformation: Embracing Automation, Cloud Computing and DevOps," enterprises are increasing their use of the public cloud 15% faster than forecasted 2 years ago. While these new platforms offer clear benefits to the enterprise such as increased business agility and reduced costs, they can compromise network security by increasing the attack surface and exposing the business to cyber threats.

Tufin Orchestration Suite (/tufin-orchestration-suite) provides centralized management with end-to-end, policy-based change automation of Amazon VPCs, Security Groups and Instances alongside on-premises data centers and other cloud platforms — for full visibility across the enterprise using a single console. With Tufin, organizations can seamlessly extend network security management to critical business applications deployed on AWS, while ensuring the enterprise is fully secure and compliant.

References