GIGAOM

REPORT

Zero Trust Cloud Security Provider Aporeto: Product Profile and Evaluation

Stronger Security, Simpler Operations, and Better ROI

GOIBENRODRIGUEZ

TOPICS: CLOUD CONTAINERS SECURITY AND RISK

SPONSORED BY Aporeto

GIGAOM

Zero Trust Cloud Security Provider Aporeto: Product Profile and Evaluation

Stronger Security, Simpler Operations, and Better ROI

TABLE OF CONTENTS

1. Summary

Microservices approaches such as Kubernetes are changing the way people think about applications, bringing the dual benefits of massive scalability and modularity. Containers abstract the applications away from the systems and network infrastructure. As a result, goes the theory, application developers can create software without having to request network configuration or other operational changes. However, while this idea of masking what goes on "under the bonnet" is good, it can also be a source of risk. Not the least, for example, is that Kubernetes allocates services to server nodes dynamically. This leaves network and security engineers with a limited set of choices: for example, either restrict Kubernetes clusters to only run within a security-controlled subnetwork (which, of course, undermines the very principle of the distributed microservices architecture), or face the need to open up network firewalls to allow clusters to communicate, undermining security and losing visibility on network activity. Considered in isolation, neither option is particularly attractive. Given an already-challenging network environment, with multiple application types (each with different connectivity needs) and permissions systems, constantly changing endpoints, equipment refresh cycles, fault resolution, and new security vulnerabilities emerging all the time, the result creates a new set of problems to be solved. Engineers have only limited time, and such compromises can have knock-on effects on other systems, leading to inefficiency, cost, and frustration.

Micro-segmentation approaches, such as Aporeto, enable application-specific security controls to be allocated while keeping networking and security professionals assured of policy definition and enforcement. This creates a middle ground between an "anything goes" approach and having a fully locked-down environment, allowing application developers to define and control the ways their application elements communicate while working within predefined security stipulations.

In this report, we provide a comprehensive independent review of the Aporeto solution for network and identity management in a multi-cloud deployment. We review the practicalities of deploying Aporeto to deliver a stronger security architecture for Kubernetes container microservice applications running across distributed networks anywhere. We also evaluate the impact on the IT operations team of running Aporeto versus maintaining legacy security practices. The return on investment for an identity-based security solution becomes clear as we progress through the following series of tests.

The GigaOm Multi-Cloud Test Lab environments used in this report include Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure although the findings are relevant for other clouds and on-premises solutions such as VMware, Packet, IBM Softlayer, etc.

2. Legacy Network Security Controls

Passwords and firewall access control lists (ACLs) are fairly static, broad legacy tools used to protect an organization's important assets. With modern dynamic workloads and the mobile BYOD user base, a more flexible method to control access is needed. Ephemeral containerized cloud-based workloads and their consumers dynamically change IP addresses; workloads scale and move between regions. Attempting to secure multi-cloud microservice-based applications is a series of compromises between isolating and segmenting various workloads versus the speed of change driven by the demand for agility.

For example, patching a legacy application running on a few compute servers is typically done manually by the IT staff once a month with an accepted impact. However, cloud-based Kubernetes workloads would need to be patched daily with new compute nodes being spun up and down automatically with different IP addresses. ACLs on firewalls simply can't keep up with the rate of change that is an inherent part of microservices. It becomes the unacceptable choice between:

1. **Compromising security by opening up a firewall rule** – allowing policies that open entire network segments exposes them to attack. This goes against the principle of least privilege (POLP).
2. **Increasing cost and complexity –** by deploying multiple Kubernetes clusters with different networks for each security classification of data. This is the typical approach taken by customers with security-sensitive workloads today.

Simply sticking with legacy methods to deploy applications on dedicated compute machines is not recommended. Some organizations may survive a short while without adopting newer technologies, but eventually, their ability to compete will be limited by their legacy infrastructure. Protecting against today's advanced security threats with a legacy architecture will become increasingly difficult, if not all but impossible.

Our IT environments must evolve to stay current with business needs and to protect against ever-evolving threats.

3. Aporeto Zero Trust Cloud Security Overview

Aporeto offers a no-compromise solution by applying identity-based access control mechanisms to application workloads in the cloud or on-premises. Labels or tags are associated with the source and destination end-points. These tags can be coarse (such as "@cloud:gcp") or highly granular (for example, "@cloud:gcp:zone=europe-west2-a"). Multiple labels function as the identity of an application or service. These can be auto-generated and are guaranteed by way of a root-of-trust service, such as a certification authority. Labels can also be used to trigger activities, for example, to cause additional monitoring of traffic between specifically labeled end-points. At runtime, network traffic is permitted by way of Aporeto Enforcer agents such that only correctly labeled traffic is delivered to the corresponding end-point. Given that Aporeto agents can run anywhere in a private or public cloud, the result is a perimeter-agnostic and easily configurable capability that enables traffic to pass only where it is authorized. It also enables portability. As such, security rules follow the application should it be migrated or extended from a private to a public cloud environment. This capability runs within network layer 3; however, the result is configuration and enforcement at the workload level. In DevOps environments, this enables policy-as-code infrastructure, whereby application engineers define and enforce security rules rather than waiting for these to be set up by the security or operations teams. Compliance reporting is also provided with the Aporeto solution. Aporeto offers API's to integrate with DevOps automation and orchestration, security testing, logging, and monitoring tools.

This robust security model enforces control over communications between distributed network resources. There are three key components in the deployment of an Aporeto Zero Trust Protection Solution:

1. Aporeto Security Orchestrator
2. Aporeto CLI (APOCTL)
3. Aporeto Enforcer Agent

Aporeto Security Orchestrator

The Security Orchestrator is the heart of the Aporeto solution. This set of management and control-plane services can be deployed on any compliant Kubernetes cluster for a customer-managed environment either on-premises in a datacenter or in the cloud. It is also offered as a subscription service that is hosted and maintained by Aporeto.

Orchestrator is available in multiple geographies and provides a single pane of glass for policy management, analytics, and operations. Features include:

- RESTful application programmatic interface (API)
- HTTPS web console user interface (UI). See Figure 1.

- Online local web-based access to the documentation repository
- APIs can be used to gather Aporeto metadata for system metrics monitoring in your data visualization platform of choice, including Grafana.

Figure 1. Aporeto Security Orchestrator Platform View

Aporeto CLI (APOCTL)

APOCTL allows programmatic access to the Aporeto platform. APOCTL is available for both Linux and Mac OS. Once installed, a security token environment variable is configured that allows scripts and ad-hoc commands to be issued. APOCTL communicates with the Orchestrator via API.

Aporeto Enforcer

Hosts with this agent installed are known as “enforcers” as they collect environment metadata and control access via policies mapped to various host services (i.e. network protocols, such as SSH, FTP, HTTPS). Aporeto Enforcer control points offer a significant improvement in granular access control versus traditional passwords and network ACL policies. Metadata from multiple sources combine to form an identity document for each source and destination of a network packet flow. This identity is cryptographically signed by the Aporeto Controller, which determines in realtime what network flows are allowed or not.

The Aporeto Enforcer agent connects a host node to the control plane and is responsible for obtaining the defined policies. Enforcer agents are available for the following types of hosts:

- Linux (RedHat RHEL, CentOS, Debian, Ubuntu, SuSE)
- Docker 18.02 or later container
- Kubernetes 1.9 or later, OpenShift 3.9+
- Amazon AWS public cloud platform
- Google GCP public cloud platform
- Microsoft Azure public cloud platform
- Microsoft Windows (coming soon-At the time of this writing, Windows Enforcer is in alpha testing and not considered for this testing.)

The Aporeto Enforcer agent features the following options:

- capable of enforcing policies at L3, L4, or L7
- automatically deployed as a daemon-set as part of a k8s cluster build
- can be deployed with scripts on Linux as a process or a Docker container
- can be installed as a sidecar when using a service mesh architecture
- lightweight process consuming minimal RAM and CPU resources
- validates the cryptographic integrity of L3 TCP traffic by adding identity information to the SYN/SYN→ACK portion of the session establishment
- validates the cryptographic integrity of L3 UDP traffic by adding identity information using UDP options

Vulnerability Scanner

Aporeto offers integration with the CoreOS Clair vulnerability scanner application. Processing units matching a certain common vulnerability scoring system (CVSS) score or higher can be tagged for each common vulnerability and exposure (CVE). Then based on the applied tags we can respond as follows:

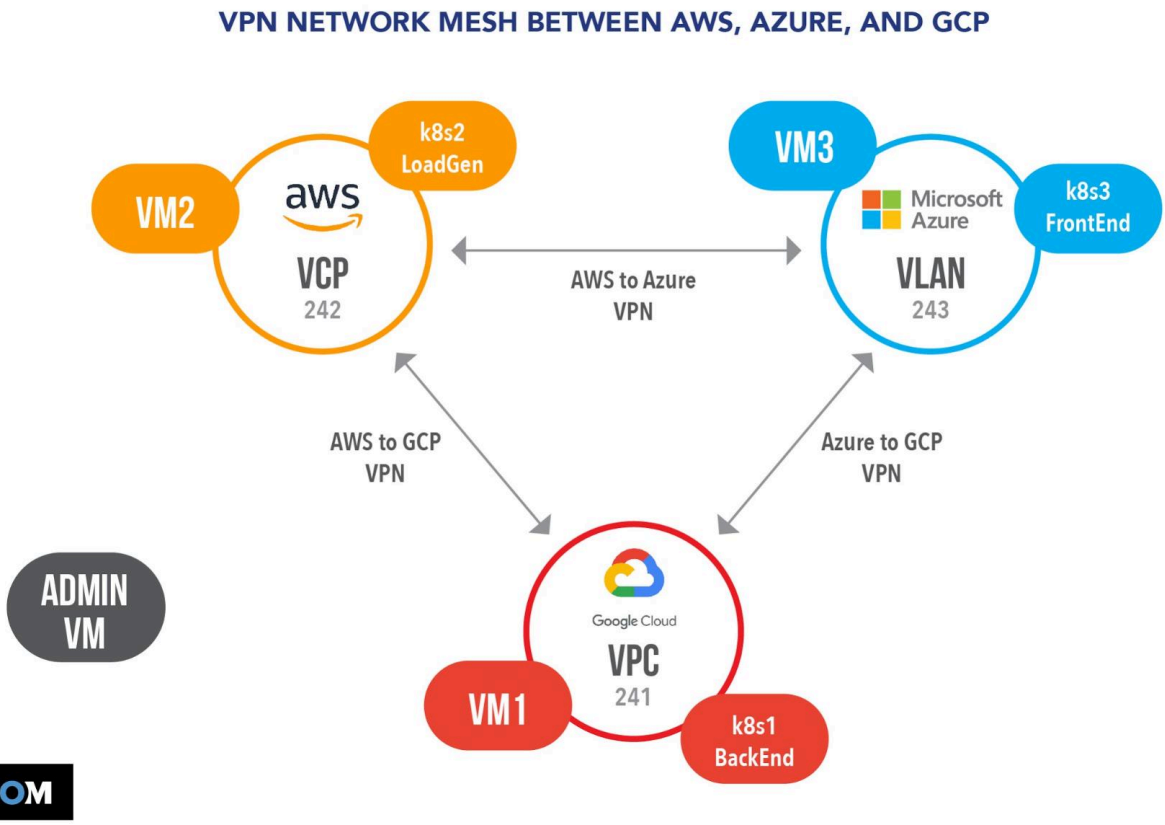
- alerting an administrator via a **Hook Policy**
- takes an **Action** such as blocking the vulnerable traffic

4. Field Test Lab Setup and Methodology

GigaOm Field Test Lab Overview and Setup

GigaOm used a scripted Terraform deployment process to simultaneously establish two parallel multi-cloud test labs. This allowed us to repeatedly set up, test, and destroy complete virtual instances of the multi-cloud test environment. Each GigaOm network security test lab incorporates three public cloud environments: Amazon AWS, Microsoft Azure, and Google Cloud Platform (GCP). A pair of external "Admin" virtual machines were used for running the setup and testing scripts. This approach allowed us to iterate the setup and testing steps using virtual machines and Kubernetes clusters with matching application services, network connections, and user access controls. The code used to create these environments is available on GitHub so that others can reproduce these tests and extend them as needed.

Figure 2. Drawing of the GigaOm Field Test Lab Multi-Cloud VPN Mesh



Test Lab Environments

For each test lab iteration, two parallel environments were built, each with the same set of

components.

- TestLab-A: Test environment protected with Aporeto
- TestLab-B: Legacy protection test environment

Components per Environment

Each public cloud environment–AWS, Azure, and GCPis configured with the following components:

- A larger /16 network subnet split up into smaller /24 network subnets
- An Ubuntu Linux virtual machine with public and private IP addresses
- A RedHat Linux virtual machine with public and private IP addresses
- A Kubernetes cluster with public and private IP addresses
- An external gateway router with Virtual Private Networks to other clouds
- A unique set of SSH keys for testing user access

Once the deployment process completes, we are left with two parallel test environments as follows:

TestLab-A: Protection Test Environment with Aporeto

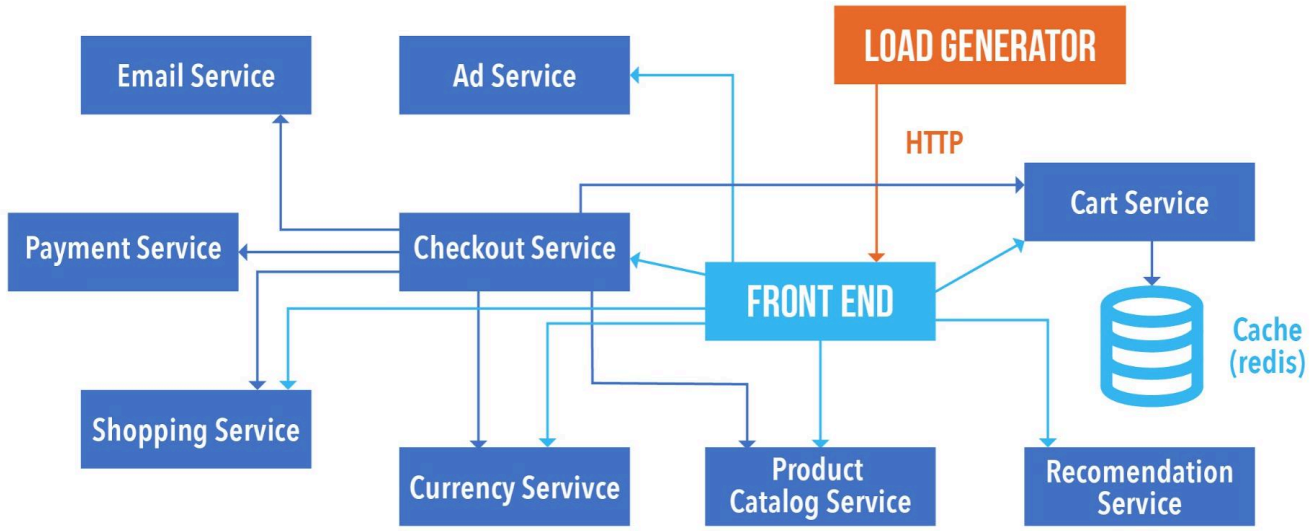
- **Cloud 1A GCP**: 2 x VM within a single network + 1 k8s cluster deployed with external Global Load Balancer(GLB) public IP address
- **Cloud 2A AWS**: 2 x VM within a single network + 1 k8s cluster deployed with internal private IP addresses
- **Cloud 3A Azure**: 2 x VM within a single network + 1 k8s cluster deployed with internal private IP addresses
- VPN linking all three clouds: 1a, 2a, and 3a
- **k8s Set A** – Three to test protection with Aporeto
    - k8s1a – GCP – BackEnd, FrontEnd, LoadGen
    - k8s2a – AWS – Load Generator service
    - k8s3c – Azure – Linux pod

TestLab-B: Legacy Protection Test Environment

- **Cloud 1B GCP**: 2 x VM within a single network + 1 k8s cluster deployed with external Global Load Balancer public IP address
- **Cloud 2B AWS**: 2 x VM within a single network + 1 k8s cluster deployed with internal private IP addresses
- **Cloud 3B Azure**: 2 x VM within a single network + 1 k8s cluster deployed with internal private IP addresses
- VPN linking all three clouds: 1b, 2b, and 3b
- **k8s Set B** – Three to test Legacy protection
  - k8s1b – GCP – BackEnd, FrontEnd, LoadGen
  - k8s2b – AWS – LoadGen Generator service
  - k8s3b – Azure – Linux pod

To emulate a typical enterprise-type application, we chose to use the Hipster Online Store demo application. Hipster is composed of multiple k8s services that are deployed on containers via YAML configuration files, which are customized for the test scenarios.

Figure 3. Hipster Online Store Microservices Demonstration Kubernetes Application



GIGAOM

Figure 3 illustrates the network flows between the various microservices of the Hipster app. Microservices typically use only one virtual CPU and a small amount of RAM. Replicas can be automatically created and destroyed based on load. By increasing the number of replicas we can increase the load and trigger an auto-scale of the k8s cluster nodes to 10 or more. Aporeto protects dynamically scaling microservices, which is challenging to protect using legacy security controls.

These containers are hosted in Kubernetes clusters running on Amazon AWS, Google GCP, and Microsoft Azure. For each cloud provider, a set of Ubuntu and RedHat Linux OS virtual machines was also deployed, allowing us to evaluate the network flows between the virtual machines and the k8s services.

Each of the k8s pods is running on a cluster with multiple virtual machine nodes used for running the containerized workloads. The Hipster app's Loadgen service running on AWS cluster **k8s2** communicates to front-end service on primary GCP cluster **k8s1** using the external IP address of the load balancer.

Each cluster also has an old Ubuntu container deployed for outbound cURL and Nmap testing. This older Ubuntu container has out-of-date software to show the vulnerability protection features of Aporeto in our testing.

Testing Methodology – Story Building Blocks

Each of the following test stories builds on the previous one. Working through each test scenario, we identify the improved security and ease-of-operations when using Aporeto compared to legacy networking security. We perform the following GigaOm field tests:

- Test 1: Protection from External Threats
- Test 2: Protection from Internal Data Center Threats
- Test 3: Protection from Kubernetes Workload Threats
- Test 4: Protection from Multiple Cloud Threats
- Test 5: Protection from Workload Vulnerability Threats

Test 1 Overview: Protection from External Threats

Test 1 explores how to protect public-facing workloads that are configured with external IP Addresses. Inbound network access should be limited to a few authorized hosts and only a specific set of services used for configuration management. All other inbound network traffic is blocked. This test examines the process to deny network traffic from the internet while allowing our external admin machines to ssh into each public-facing cloud machine. To initiate the change in policy we start with the initial use case of an IT administrator connecting to a trusted administration "**jump"** machine. This is where all

management activities take place.

Test 2 Overview: Protection from Internal Data Center Threats

Since we have a VPN established between all three clouds, any machine in one cloud can communicate freely with services in its own cloud or any of the others. All of the Hipster application services are running in a single K8S cluster, which is using private IP addressing and protected from internet-facing threats with the traditional approach of a multi-homed bastion host and an external load balancer. This is a fairly typical setup and easy to protect via a legacy "M&M" security model approach where you have a hard network perimeter around a soft inner network. The drawbacks to this approach are a lack of resilience due to no alternate site and no east-west protection between workloads. This test evaluates micro-segmentation methods to separate internal network traffic based on cloud metadata policy instead of IP addresses.

Test 3 Overview: Protection of Kubernetes Workloads

This test builds on the last configuration, which protected traffic between k8s microservices and legacy applications in different clouds. The goal of this test is to show how traffic between microservices can be protected by using k8s **labels** as tags for firewall policies. For this scenario, we use a YAML configuration file to deploy a couple of Ubuntu containers with different labels. The objective is to only allow traffic from authorized containers matching a certain label to connect to a set of protected HTTP servers. For a truly cloud-agnostic solution, it should not matter where the sources or targets reside.

Test 4 Overview: Protection of Multiple Clouds

In Test 3, we explored using k8s **labels** for workload protection between just two clouds. We expand on this in Test 4 by verifying that a k8s **namespace** policy created in a parent Aporeto namespace is propagated to all the child namespaces and uniformly applied to all the Kubernetes clusters in all of our target clouds such that they simultaneously receive the same network access policy (NAP). Based on the previous configuration, we've shown that each Kubernetes cluster now assumes the policy without additional configuration. These same policies will now be expanded to equally protect container-based workloads regardless of where they are running. This test shows the challenges of adopting a **multi-cloud** strategy and how Aporeto helps address these.

Test 5 Overview: Protection of Vulnerable Workloads

To validate Aporeto's vulnerability protection, we show how to configure the following settings in the Aporeto Security Orchestrator console:

- A vulnerability with a CVSS score 6 or higher is detected sending an alert email
- A vulnerability with a CVSS score of 7 or higher is detected with the response of blocking the vulnerable traffic.

5. Results: Aporeto vs. Legacy

Testing by GigaOm compared the network security controls for long-lived workloads, such as the admin systems used by administrators (which are typically only IP address + user identity), to those network security controls available for workloads with ephemeral IP addresses, such as the Kubernetes microservice container-based workloads with labels and namespaces.

When comparing the Aporeto (workload ID-based) solution to traditional (IP address-based) solutions, we found that Aporeto offers more granular security controls, is simpler to operate, and has a better return on investment.

By running through the various test procedures outlined above, GigaOm compared the ability of Aporeto protection versus legacy protection in scale-up scale-down situations during which IP addresses change frequently. In this section of the report, we compare the results of the protection with Aporeto to the legacy protection methods. Please reference the Part 2 of the report for the details of the test procedures.

Test 1 Results: Protection from External Threats

The results from Test 1 offer a powerful demonstration of a true security-as-code enforcement model. Aporeto handles both policy administration and decision functions centrally. Since the Aporeto Enforcers are distributed to each protected end node, it does not matter where the machines are located. They will instantly receive the maximum level of protection upon deployment. Because the policy is set once with Aporeto Security Orchestrator and any future changes to the environment automatically inherit the protection, significant time was saved deploying our multi-cloud environment. No additional work was necessary. Setting up legacy protection required hours of work and hundreds of lines of code to create the various complicated firewall protection rules for each cloud provider. These concepts are covered in this test:

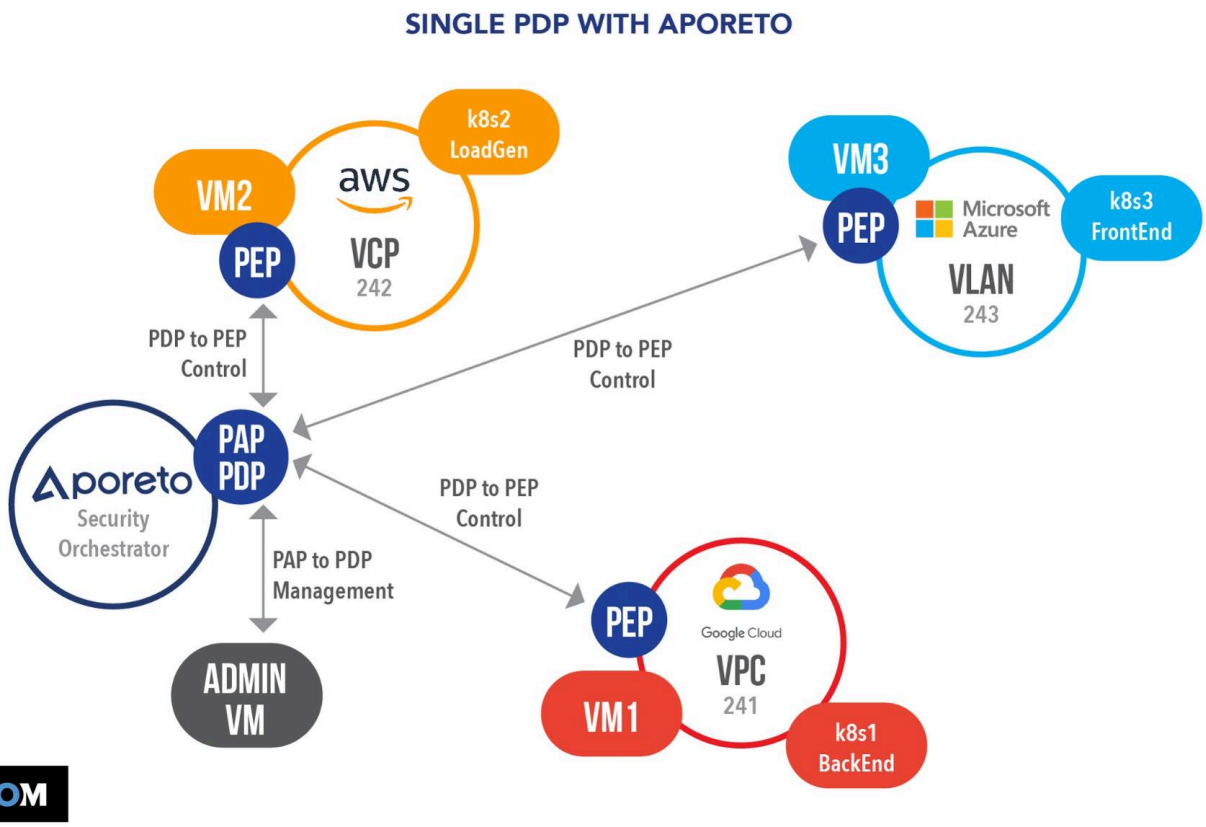
- **Data Plane:** Policy enforcement points (PEP) allow or reject the network flow of the data.
    - Aporeto uses enforcers for PEPs that function to uphold protection regardless of where the workloads are located.
    - Legacy PEPs require different sets of firewall rules at the host or network perimeter as provided by the cloud or infrastructure vendor for each environment – resulting in inefficiency and risk of noncompliance.

- **Control and Management Planes:** Policy decision points (PDP) control policy enforcement points
    - Aporeto Security Orchestrator is a single platform, simplifying management and oversight eliminating security gaps and silos of protection across the enterprise.

- Legacy PDPs are each cloud provider's and firewall vendor's management console, VMware vCenter, Linux IPTables or UFW host-based firewall ACLs, which adds significant unnecessary complexity in an enterprise or multi-cloud environment.
- Legacy policy administration point (PAP) would be the source code management (SCM) or configuration management systems used to manage changes for each cloud provider. For a DevOps environment, these changes could be managed by pull requests on git repos to drive Terraform and Ansible. This seems manageable for a single application on the surface but adds significant time and resources in an enterprise-wide security program.

Figure 4 illustrates the benefits of the protection with the Aporeto model utilizing a centralized PAP and PDP versus the more complex legacy protection model where PDPs are distributed across multiple cloud providers and firewall technologies as presented in figure 6.

Figure 4. Single PDP Protection from External Threats with Aporeto

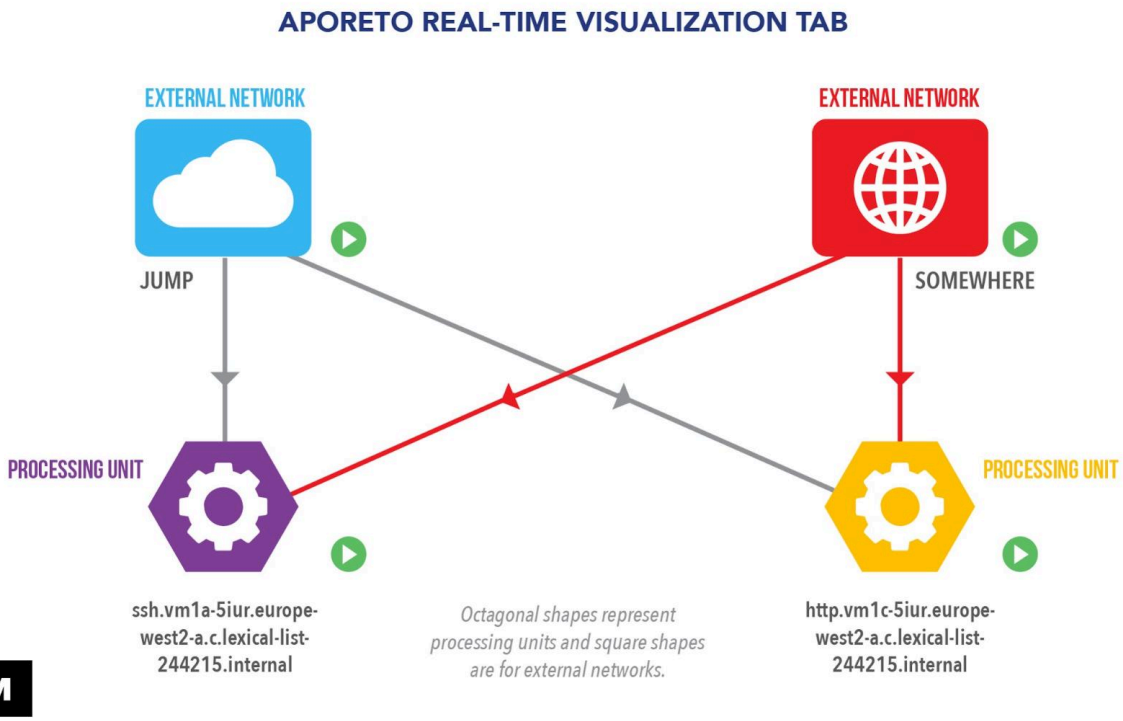


GIGAOM

Validating the benefits of protection from external threats with Aporeto was simple since our workloads were automatically protected. We simply went to the Security Orchestrator dashboard and observed the network real-time visualization tab. This shows us:

1. Green flows that are allowed from the authorized jump machine to the SSH and HTTP processing units (PU).
2. Red flows that are blocked from “somewhere” to the SSH and HTTP PU. These were investigated and found to be rogue hackers from other countries trying to break into our external-facing machines.

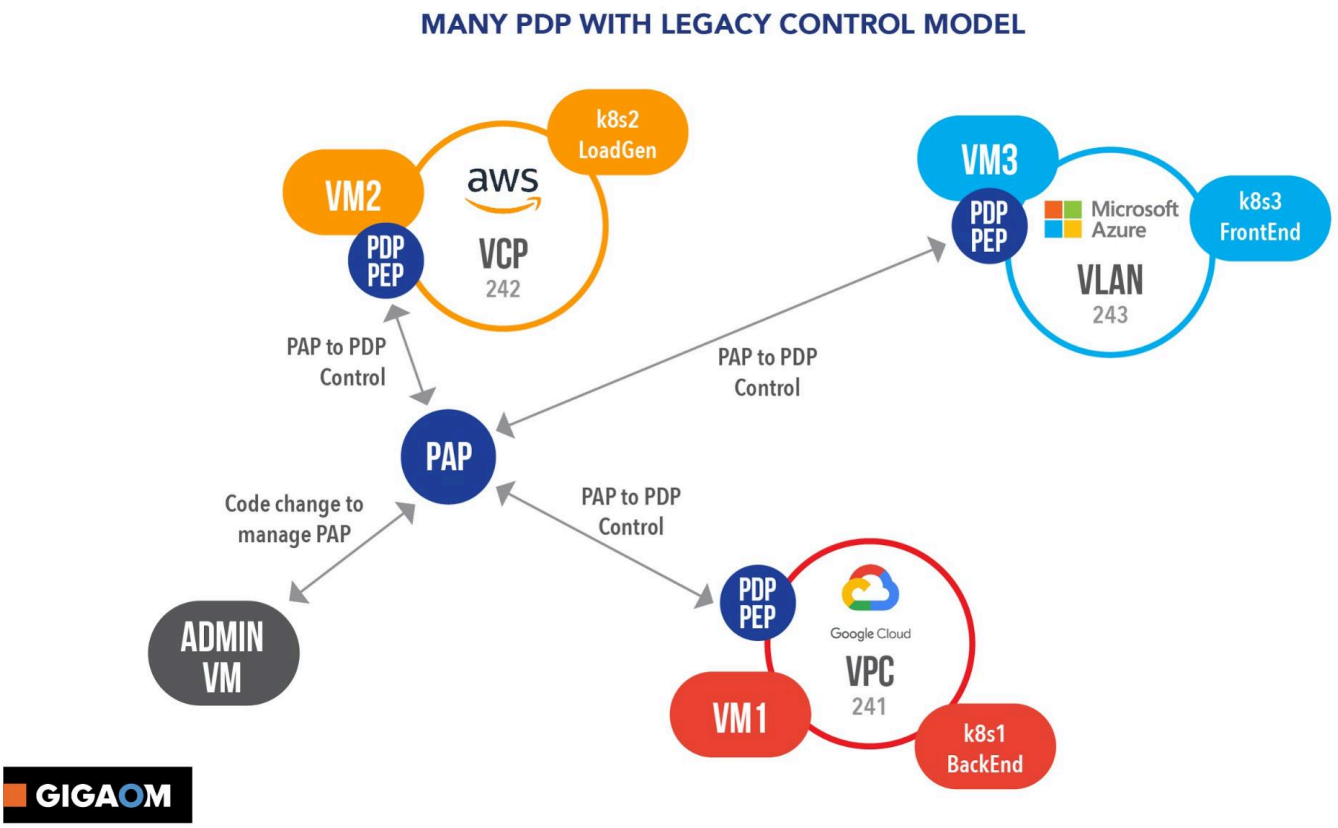
Figure 5. Aporeto Real-time visualization tab



GIGAOM

For the legacy protection from external threats, we attempted to implement the same set of security controls with the native configuration options offered by each cloud provider.

Figure 6. Legacy protection from external threats with multiple PDP



Compare the protection offered with the Aporeto solution to two methods used to provide legacy protection:

1. **Bastion Hosts.** A bastion host is a hardened machine with extra security controls, such as multi-factor authentication, enhanced logging, a patching policy, and a limited set of running services and features. They attempt to address the complexity of constantly changing firewall ACLs but are cumbersome to maintain and do not adhere to the principles of least privilege or separation of duties.

2. **Native Cloud ACLs.** Each public cloud provider has its own proprietary method to establish firewall rules that would secure inbound traffic from the internet into each VM with a public IP address.

Herein lies the challenge; the composition of these sets will change over time so whatever access control lists we make today will become out-of-date. In a perfect world, the policy is written with a list of trusted hosts, a list of machines with public IP addresses that need to be managed by trusted hosts, a list of protocols to allow between the hosts and public machines, and a policy that allows traffic from IP addresses between them. If it was that simple then any change to one of the lists would dynamically update the policy. When you only have a few firewalls and the trust zone or blast radius of your data center is small, then these legacy controls might be adequate. However, today's multi-cloud, microservices-powered enterprise environments are not so simple. Each cloud provider handles policy administration, policy decisions, and policy enforcement differently, which makes security

administration significantly more time consuming and labyrinthine- requiring additional complex code programming, testing, and deployment pushes are needed.

Legacy security policy administration options include:

- **Web-based user interface console –** This is sufficient for minor quick changes but this won't work for making many changes at once. The more changes that are needed result in the loss of time to implement a change and the chance of human error to increase.
- **Command-line interface software development kit (CLI SDK)** – As the rate of change increases, system administrators turn to the CLI to write and test shell scripts that can be enhanced to leverage input files and use variables with parameters to make changes. This method is certainly faster when making many changes at once but very manual as a human is still responsible for building the input lists and running the scripts. Unless a robust testing process is in place, these scripts run the risk of breaking things or taking down entire networks. Vendors upgrade the APIs that these SDKs use and the scripts need to be maintained. Admins come and go too, then an organization is left with technical debt for code that does not work well, leaving an enterprise exposed.
- **API –** the best way to manage configurations across multiple cloud providers is by using Terraform code stored in a source code repository. Using pull requests these changes can be tested, reviewed, and approved; then scheduled as part of a robust continuous improvement/ continuous delivery (CICD) pipeline. Most organizations are doing this today as they adopt DevOps processes. This requires a significant investment in developing staff and set up to get to a functional point. A decent security policy would require network changes involving public IP addresses to go through a review process including regularly scheduled vulnerability management scanning. This can introduce significant delays in the time it takes for a code change to make it through from policy administration and down to policy enforcement.

The policy **decision** point is done at the level of each cloud provider. When using Terraform to create the initial firewall rules, it is important to save and share the state file with information about the deployment. If any changes happen outside the Terraform platform and the state files are not updated, then future plans that apply or destroy sessions will error out and fail. Change management cycles expose a network to zero-day threats.

The **enforcement** of cloud-based access control lists is handled at the virtual private cloud (VPC) network level. Each cloud provider does this in a different manner:

- Amazon AWS uses Security Groups associated with each instance limited to 200 ACLs per VPC and 20 rules per ACL. It is recommended that the same ACLs be used for all machines in a VPC. A VPC is treated as a legacy VLAN with all nodes having the same security posture.
- Google GCP uses Firewall Rules that can be applied to:

    - All instances in a given network
    - Instances with matching target network tags (limit of 70 per rule)
    - Instances by target service accounts (limit of 10 per rule)

- Microsoft Azure uses Network Security Groups (limit of 200 per subscription) that work with:
    - IP addresses
    - Service tags
    - Application security groups
    - VirtualNetworks (VLANS)

In summary, the CICD process is great for changes to a single application and a single cloud infrastructure but troublesome for firewall ACL administration.

It is also important to point out the lack of analytics and visualization with a legacy protection solution. It might be possible to leverage a completely separate multi-cloud aware analytics solution but that would introduce additional operational costs and increase the risk of security incidents due to data leakage and configuration drift. The Aporeto holistic dashboard provides a comprehensive view, ensuring both security and compliance.

Test 2 Results: Protection from Internal Data Center Threats

With Test 2, we see the Aporeto approach shine with its lucidity. The policies are built once and automatically apply to both existing and new workloads. Each policy can be enabled or disabled with the click of a button or scheduled during certain times of day or day of the week. And of course, we get the added benefit of the centralized visualization dashboard and logs to observe both successful and failed policy events.

By comparison, the legacy protection options require coding changes to be made on each cloud provider. Since each provider has its own set of PAP, PDP, and PEP controls, a defense-in-depth security strategy would need firewall rules on each cloud provider to be changed. Making this change requires engineers with different skill sets, a significant amount of code, and in-depth testing to prevent unexpected results. If using the Terraform code, it is important to have access to the latest state file information and test by using a CICD pipeline. If being done manually with either the Web UI or CLI scripts, peer review and testing phases need to be performed to reduce the risk of human error.

See the code we used to create these firewall ACLs here on github. Development and verification of this code took **two days** with multiple iterations and failed attempts. Statistics for this work: 487 lines of

code in 14 changed files. We expand on this code later to build the legacy protections for Test 4.

Test 3 Results: Protection of Kubernetes Workloads

In Test 3. we show how Aporeto protects between container-based workloads running in the same cluster. This solution offers true zero trust protections and micro-segmentation for workloads bringing multi-tenant capabilities to Kubernetes clusters running on any cloud. The beauty of this test is that no IP address information is needed to build the policy. And as we show in the upcoming multi-cloud test, this same policy is propagated down to any resources in the protected child namespaces.

Aporeto provides Kubernetes auto-scale protection in the k8s cluster by automatically installing and configuring the enforcer agent on each node. We deployed multiple replicas of the loadgen service available, as part of the Hipster app, to demonstrate how the number of nodes in the cluster grew automatically as load increased. Once the loadgen service was removed and network traffic decreased, the number of active nodes in the cluster automatically decreased. This “scale down” process takes about 10 minutes with the default Kubernetes cluster settings on GCP.

With Aporeto, we are able to configure network access policies based on the metadata available from the environment where the workloads are running. Kubernetes offers a few options that can be used as tags when building these policies, and Aporeto collects even more metadata about the running environment.

This test cannot be performed using legacy security controls since these ephemeral workloads have constantly-changing IP addresses. We tested this by using the **NodePort option** when deploying the Hipster app via YAML files. Now with the IP addresses exposed for each k8s host, the node access is also allowed to other (non-selected) microservices that are running on the same node. This violates the principle of least privilege, which is less than ideal for implementing security.

It is technically possible to configure similar network security controls with k8s network policy for a single cluster environment. However, it will not provide protection between globally distributed multi-cloud k8s services and legacy resources, such as virtual machine-based applications. In the event that multiple clusters exist across multiple cloud providers, this network policy configuration would have to be duplicated for each cluster. Furthermore, network policy in Kubernetes is a default “allow-all” configuration, in that, if a policy is not defined, communication is allowed between services within the cluster. For a more secure environment, additional automation would be required to ensure that a network policy object is created with every new Kubernetes namespace or workload.

These policies are often managed within each Kubernetes namespace. Automation is required to ensure they exist before applications are deployed in the case of a zero-trust network security model. In addition, these policies must be replicated to all kubernetes clusters across all cloud providers and will not have an effect on systems outside of the cluster. Aporeto is the clear winner in this K8s workload protection test.

Protection of Kubernetes Workloads with Aporeto Scale Test

To observe the scale up and scale down process with Aporeto, we looked at the list of active enforcers in the default namespace on the Aporeto web UI console using the APOCTL command as shown in the figure below. The ability for Kubernetes clusters to scale up takes just a few seconds as the load is applied. We illustrated this with the activation of loadgen service replicas and saw new nodes created immediately. Once the loadgen service replicas were deleted, the extra k8s cluster nodes and the corresponding Aporeto Enforcers were removed within 10 minutes as they were no longer needed.

Figure 7. Auto Scale List of Aporeto Enforcers on each k8s node and compute vm

```
$ apoctl api list enforcers -n /aporeto/gigaom/mct/gcp/ios8 -o table -c name -c
subnets
                          name                          |          subnets
--------------------------------------------------------+-----------------------------
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-003q | [10.241.1.11/32]
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-48m6 | [10.241.1.4/32]
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-7283 | [10.241.1.3/32]
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-bgsn | [10.241.1.8/32]
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-kcz2 | [10.241.1.15/32]
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-p4ns | [10.241.1.16/32]
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-pqrn | [10.241.1.2/32]
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-qmrh | [10.241.1.14/32]
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-r9ct | [10.241.1.6/32]
  gke-k8s1a-ios8-k8s1a-node-pool-ios8-fc2766f8-xgk6 | [10.241.1.13/32]
  vm1a-ios8                                         | [10.241.1.100/32]
  vm1c-ios8                                         | [10.241.1.200/32]
```

Test 4 Results: Protection of Multiple Clouds

As the application teams mature and business needs change, more resilience is needed and features of other cloud providers should be utilized. A decision to adopt a multi-cloud strategy enables app teams to leverage the best-of-breed services from various providers but also introduces some additional complexities and challenges.

Test 4 continues with the same legacy challenges we had with Test 2 and Test 3. For protection without Aporeto, firewall rules need to be created for each cloud provider. This is a multistep process that needs to be codified into the deployment scripts. This development work takes a significant amount of time with repeated testing to get everything working between the three clouds, VPNs, gateways, static routes, subnets, compute nodes, and k8s clusters. Even once everything is operational, making a change is a complicated and error-prone process. Most organizations will shy away from this type of legacy micro-segmentation setup due to the risk of service impact during any change. Heavy investment in automation and configuration coding changes would be required for every new cluster that is deployed in each new cloud environment, a costly and impractical use of valuable budget.

Expanding our network access policy with Aporeto took a few minutes versus days with the legacy method. Since we can enable and disable the policy from the central dashboard with the click of a button, there was a low risk if we needed to back the policy out. Not something we can say about the legacy method of pushing code across multiple cloud providers using different technology footprints. The ability to visualize the change in real-time using the Aporeto dashboard provides confidence to the operations staff that normal traffic flows are allowed and unauthorized access attempts are being rejected.

Test 5 Results: Protection of Vulnerable Workloads

In Test 5, we look at vulnerable workload protection with Aporeto compared to legacy vulnerability management methods. There are many legacy, cloud-native, and third-party vulnerability management solutions available. Some are included as part of a cloud provider's standard offerings and some offer a comprehensive solution that can provide a common platform that works across physical, virtual, and newer container-based workloads. Most of these legacy solutions only serve as detective security controls to enable reporting on potential application vulnerabilities. While checks can be put in place to alert or block the deployment of new images, without the latest security patches there is little being done to mitigate the risk of existing workloads running with vulnerable software while waiting for a patch to be tested and released to production. What is needed is a system that can not only detect the vulnerabilities but provide real-time remediation or mitigation as a compensating control.

Our testing shows how Aporeto can not only detect and alert on vulnerabilities found on running workloads, but it can also block these vulnerable workloads from accessing the network. The nice thing about using Aporeto for vulnerability management is that the policies created centrally can be applied to all the workloads regardless of their location. There are no additional scanners, appliances, or agents needed. Since the enforcers are distributed, the scanner will also periodically update running processing units to detect any new vulnerabilities. We set our update interval timer to the default running every 2 hours, although this is configurable.

In our testing, we found that each cloud provider has a different method to detect vulnerabilities in their container registries. These alternative solutions offer limited capabilities with additional cost and complexity. For example, most of these solutions only offer vulnerability monitoring and reporting capabilities. This will help prevent known vulnerabilities from getting introduced but will not mitigate against newly discovered vulnerabilities on currently running workloads the way Aporeto does.

6. Conclusion

Key Findings:

GigaOm testing has found that enterprise security policy enforcement is assured with Aporeto application-specific controls, enabling developers to create application elements within predefined security stipulations. This is especially important for ephemeral workloads that require micro-segmentation, such as:

- Containers running on Docker and Kubernetes K8S
- Serverless code
- Mobility applications
- Internet of Things (IoT)
- Modern dynamic cloud-based workloads.

The Aporeto Enforcer control points offer a significant improvement in granular access control versus traditional passwords and network ACL policies. Aporeto provides high-performance security in a globally-distributed, multi-cloud, hybrid network by centrally managing policies and distributing these policies to endpoints for low latency, policy decision processing.

Stronger Security

Our testing clearly shows the security benefits of the Aporeto cryptographic identity approach versus legacy network controls:

- Organizations can achieve a true zero trust security posture with Aporeto.
- The identity documents used by Aporeto are protected with an encrypted SYN, SYN-ACK, ACK handshake similar to RFC 793 including protection from SYN flooding attacks (see RFC 4987).
- Aporeto's protection automatically scales up and down based on load, making it the perfect security solution for dynamic cloud-based workloads. See Test 3.
- In Test 4, we proved Aporeto's ability to protect both Linux-based compute workloads, as well as k8s containerized microservices with ephemeral IP addresses. This unique capability is only available from Aporeto. It protects containerized k8s workloads spread across multiple clusters and clouds with a single set of policies using environmental metadata tags in combination with Kubernetes labels and namespaces.
- In Test 5, we demonstrated how a rogue containerized app can attack other machines on the

network when IP address-based firewall ACL openings allow lateral attacks. We saw that not only does Aporeto detect these vulnerable services, but it automatically remediates this type of attack and prevents lateral attacks from rogue apps.

- Aporeto also has the ability to wrap encryption around otherwise clear-text traffic, such as a containerized Redis databases, like those we are using in our k8s test app. This is a huge benefit in a multi-tenant cloud environment where surprisingly Redis does not offer native encryption but instead suggests proxying traffic through a Bastion host.

Simpler Operations

The GigaOm tests build on each other to show the power of Aporeto. In only a few minutes, we were able to apply Aporeto to fulfill the following tasks:

- Allow SSH to VMs from only trusted admin jump boxes
- Only allow properly labeled k8s pods HTTP access to sensitive services
- Block access based on k8s namespace identifiers
- Create a policy leveraging metadata provided by cloud providers
- Automatically leverage cloud identity documents across hybrid environments to simplify network and security operations with micro-segmentation.

Aporeto will simplify life for network and security operations teams with the following advantages:

- **Aporeto is offered as a managed service**.
  - There are no servers to maintain. Patches and updates are automatically applied. During our testing, new features were introduced without impacting our operations.
  - Global presence with built-in high availability and disaster recovery.

- Three convenient methods to operate the Aporeto solution that enables different modes of operating:
  - The web-based Security Orchestrator console
  - APOCTL Command Line Interface
  - The RESTful API as they integrate with an automated CI/CD build pipeline.

- Operations teams will especially appreciate the ability to visualize allowed and rejected traffic flows between various nodes either in real-time or hindsight.

- The security-as-code enforcement achieved by combining external networks, network access policies, and host service policy. Our testing shows that this is significantly more time consuming and complex to configure with legacy solutions.

Better Return On Investment (ROI)

The Aporeto solution improves ROI with:

- Minimal infrastructure capital equipment expense cost in a hybrid setting with multiple types of workloads (bare-metal Linux, VMs, containers, and serverless) to a comparable security setup protected by legacy firewalls, ACLs, secrets management tools, etc.
- Lower operational expenses due to the automated policy-as-code workflow. This enables security engineers to automate undifferentiated tasks and focus instead on higher-value tasks.
- Future cost savings as it is easier to scale to more sites as the business grows. We showed in our testing how new clouds and workloads were automatically protected when deployed.

7. About Iben Rodriguez

Iben Rodriguez is a Cloud Security & Networking Architect specializing in InfoSec Process Integration Services. Iben brings years of experience advising enterprises, financial institutions, and government organizations on how to deploy cloud security solutions using DevOps methods.
Clients include executive teams from AT&T, Brocade, CA, Cisco, eBay, Ericsson, Google, Huawei, Intuit, Juniper, Microsoft, Pentagon, US Navy, Spirent, and VMware. Active with Linux Foundation, OPNFV, Palo Alto Networks, VMware, and HyTrust, Iben also works with The Center for Internet Security as the Lead on the Amazon AWS, Google GPC, and VMware ESX hardening benchmarks.
Trained in Agile, ITIL, SOX, PCI-DSS, ISO27000, Iben started his security career in the 80s maintaining the data communication systems used by military reconnaissance overseas with the SR71 and U2, followed by stints in the global pharmaceutical and fabless semiconductor industries.

8. About GigaOm

GigaOm provides technical, operational, and business advice for IT's strategic digital enterprise and business initiatives. Enterprise business leaders, CIOs, and technology organizations partner with GigaOm for practical, actionable, strategic, and visionary advice for modernizing and transforming their business. GigaOm's advice empowers enterprises to successfully compete in an increasingly complicated business atmosphere that requires a solid understanding of constantly changing customer demands.

GigaOm works directly with enterprises both inside and outside of the IT organization to apply proven research and methodologies designed to avoid pitfalls and roadblocks while balancing risk and innovation. Research methodologies include but are not limited to adoption and benchmarking surveys, use cases, interviews, ROI/TCO, market landscapes, strategic trends, and technical benchmarks. Our analysts possess 20+ years of experience advising a spectrum of clients from early adopters to mainstream enterprises.

GigaOm's perspective is that of the unbiased enterprise practitioner. Through this perspective, GigaOm connects with engaged and loyal subscribers on a deep and meaningful level.

9. Copyright