AKAMAI WHITE PAPER

Moving Beyond Perimeter Security

A Comprehensive and Achievable Guide to Less Risk

Author:
Charlie Gero – CTO, Akamai Technologies
Enterprise and Advanced Projects Group

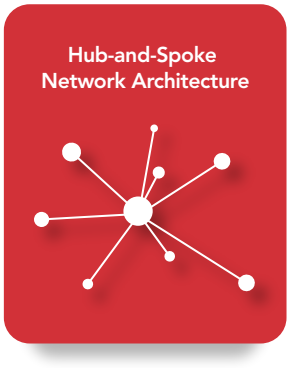
Akamai

Table of Contents

A Brief History of Network Architecture

It's hard to imagine that almost 50 years have passed since the first four computer systems were strung together on the Internet's precursor, ARPANET. As that technology has evolved from rudimentary packet-switched networks to a complex and dizzying array of autonomous systems around the world acting in concert to get data where it needs to be, so too have the threats that utilize that technology. We have seen over the past 10 years crippling DDoS attacks, data breaches that affect hundreds of millions of people, data exfiltration from sensitive government systems, the rise of ransomware, and so much more.

But for all of the change that has happened over this period of time, both good and bad, there is one thing that has remained stubbornly constant: the basic hub-and-spoke network architecture that most companies utilize.



This architecture used to make sense. Long ago, before the Internet was a bustling place of business and core infrastructure, companies placed their workloads in data centers. These data centers housed the critical infrastructure and applications necessary to perform their duties. As branch offices, retail storefronts, and satellite locations would come online, they too would need access to the centralized applications, and so companies would build out their networks to mirror that need, with all networking backhauling to their core data centers. After all, the data center was the central location where all the action occurred.

As time progressed, the Internet began to emerge as a commercially viable disrupter. SSL was invented by engineers at Netscape in 1994, enabling online commerce, and employees began to demand certain corporate services be attainable through the Internet. Naturally, businesses and carriers who had been in the practice of building complex global networks serviced these requests by doing what they knew best: deploying these services in the same data centers their internal applications were hosted in, and purchasing Internet links to provide a route to them. This fortuitously served a double purpose: Outside consumers could get in, but internal employees spread out across myriad branch offices could now get out. For the time being, hub-and-spoke was still the reigning champion of network architectures.

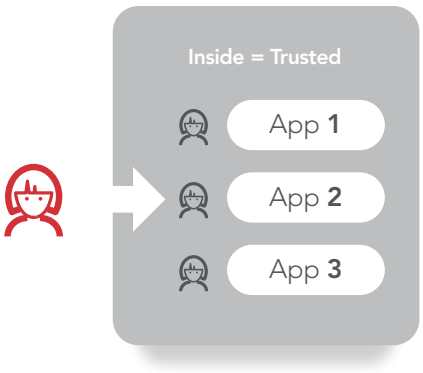
Over time, threat actors began to capitalize on this architecture, causing a whole new industry to be born: the data center security stack. Since the hub-and-spoke architecture funnels Internet traffic at data centers, large powerful boxes began to be developed for those high-capacity lines. Firewalls, intrusion detection, and prevention would rule inbound traffic while secure web gateways would enforce acceptable use in the outbound direction. The proliferation of these security systems being deployed at centralized choke points would further serve to cement hub-and-spoke as the dominant network architecture. For a time, the castle-and-moat approach to security seemed viable, and the notion of a network perimeter where everyone outside is bad and everyone inside is good remained dominant.

The Rise of Cloud

In 1964, five years before our four machines at ARPANET came online, Bob Dylan wrote *The Times They Are a-Changin'*. A more true statement could not have been said about the dominance of cloud in the past decade. As SaaS emerged, companies began to realize that it is both easier and far safer to rely upon third parties to manage supporting systems that aren't part of their core initiatives. Why run an internal CRM when you can have SaaS providers do it in the cloud, on their own managed servers, which is seemingly far safer and more performant?

Even systems long thought to be too critical to migrate such as email, Active Directory (AD), and identity are now moving into global cloud infrastructure with the rise of Office 365 and G Suite.

Further accelerating this change is the emergence of infrastructure as a service (IaaS) and cloud compute. Instead of just offloading ancillary support systems through SaaS, products such as Amazon AWS, Microsoft Azure, and Google's Compute Platform allow businesses to virtualize the very physical infrastructure itself in an on-demand, pay-as-you-consume fashion. This adds a never-before-seen level of agility in deploying your mission-critical core business applications.



Simply put, your applications are on the move. But they are not alone. Today's workforce is increasingly mobile. For many corporations, employees are just as likely to be found in a coffee shop, working at home, or in an airport as they are in a cubicle in an office.

As a result, the network perimeter no longer exists. At least not in any recognizable form. Your employees and applications are in many cases just as likely to be outside of the moat as they are inside. And with advanced persistent threats and malware, you are highly likely to inadvertently let the malicious actors inside of the perimeter with full access to your most valuable assets.

Utilizing a security and access approach that made sense 20 years ago in the modern world is at best misaligned and at worst perilous. Forrester Research says in *Future-Proof Your Digital Business With Zero Trust Security*:

> **The data economy renders today's network, perimeter-based security useless. As businesses monetize information and insights across a complex business ecosystem, the idea of a corporate perimeter becomes quaint — even dangerous.**

And this isn't just theory. This is evident in the massive amount of data breaches we've seen in the past five years, the vast majority of which happened as a result of trust being abused inside of the network perimeter.

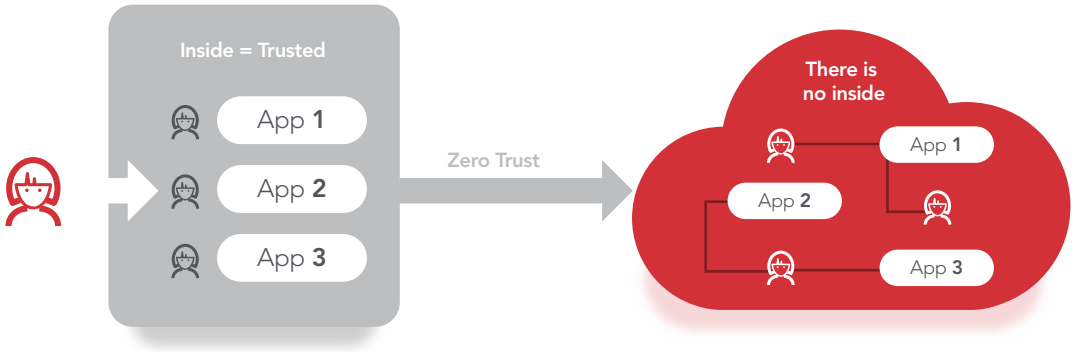
Further exacerbating this problem: Applications that were designed to live inside of a network perimeter often have the worst security profiles. After all, if you were a developer 10 years ago and assumed that only authorized employees with good intentions could reach your system, would you have been as defensive as the coder today who knows vast armies of hackers will try to exploit his or her Internet-based application?

So what are you to do?

Zero Trust

Roughly five years ago, John Kindervag, a thought leader in this space and a Forrester analyst at the time, proposed a solution that he termed "Zero Trust." The principle behind it is quite simple, but very powerful: Trust is not an attribute of location. You shouldn't trust something simply because it is behind your firewall. Instead, you should take a very pessimistic view on security where every machine, user, and server should be untrusted until proven otherwise.

The method of proof for this is strong authentication and authorization, and no data transfer should occur until trust has been established. In addition, analytics, filtering, and logging should be employed to verify correctness of behavior and to continually watch for signals of compromise.

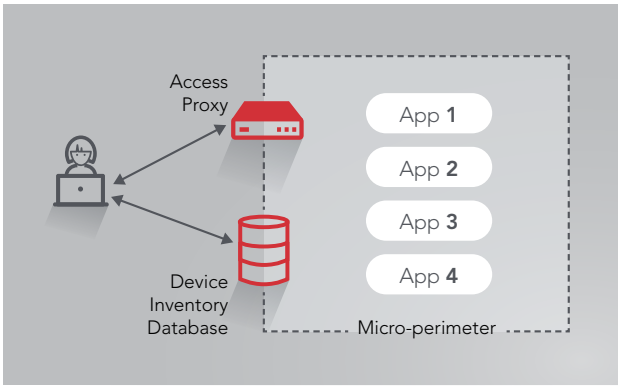


This fundamental shift in posture defeats a vast amount of the compromises we have seen in the past decade. No longer can attackers spend time exploiting weaknesses in your perimeter, and then exploit your sensitive data and applications because they made it *inside of the moat*. Now there is no moat. There are just applications and users, each of which must mutually authenticate and verify authorization before access can occur.

How does one accomplish this?

Google BeyondCorp™

A few years ago, Google debuted their vision for zero trust access in several seminal white papers known as BeyondCorp. What makes BeyondCorp compelling is that it attempts to solve the zero trust problem for all applications, without modifying their code, and achieve access in an effortless manner regardless of where the application or user lives. In a simplified diagram for discussion purposes, it looks like the following:



In this diagram, we see a number of items:

- **A User and Laptop:** In Google's model, users needing access to internal applications have a managed laptop with a certificate installed on it. This laptop also runs a small software agent.
- **Access Proxy:** The server that sits on top of the micro-perimeter line is known as an access proxy. Its job is to enforce secure access policies to applications within the micro-perimeter. It is the only entity that can directly reach the applications (aside from the applications communicating directly with each other) and lives in the DMZ of the micro-perimeter.
- **Device Inventory Database:** The database in the diagram is responsible for keeping a record of the security posture of all Google employee laptops and devices. It is periodically updated with new information.
- **Applications:** These are the applications that employees need access to. These might be things like Git, email, internal knowledge bases, finance applications, databases, etc. These are not directly accessible outside of the micro-perimeter.
- **Micro-perimeter:** The applications are cordoned off from the rest of the world by a micro-perimeter. The only server that can bridge that gap is the access proxy. Additionally, this micro-perimeter need not be in a physical data center. It is just as valid to have this perimeter in a cloud compute environment like GCP or AWS.

Operationally, the BeyondCorp workflow is extremely simple and yet powerful. On a regular basis, the software agent running on the Google employee's machine connects to the Device Inventory Database and gives a posture assessment of the device. It details attributes like which operating system is installed, which patches are present, the state of applications, AV, etc.

The Device Inventory Database takes this information, and through the assistance of a component known as the Trust Inferrer (not shown) produces an evaluation or score of the laptop. In Google's actual implementation, data is not limited to the software agent on the end user's device. It can also come from vulnerability scanners running on their network, router and firewall logs, etc. In essence, the Device Inventory Database continually collects as much information as it can to determine, out of band, the security posture of the device itself.

Any change of state on a device or externally can affect this evaluation or score. For example, if a security vulnerability is discovered in a particular OS, Google can easily and instantly instruct the Device Inventory Database to downgrade the scores of all machines running that release.

When a user on a device attempts to access an application, they are directed to the access proxy. The access proxy has secure access policies defined on it for each application, which determines the users that can access the resource as well as the minimum security posture required on the device being used.

Indeed, it is the application-centric combination of user identity *and* posture of the device that makes this so powerful. For example, Git might only be accessible by users in the developer group with laptops that are running a modern OS that is up to date on all patches. A company directory might have a much less restrictive policy defined, which allows anyone with a valid account to access the data, as long as they are on a machine running an operating system more modern than Windows XP. It is not good enough to simply say the finance group has access to sensitive accounting information. The policies ensure that the machines they access from are safe as well.

Upon connecting to the access proxy, user authentication begins. This is done over a mutually authenticated session, using the laptop's certificate for the client side. Assuming the authentication was successful, the access proxy now knows not only who is trying to access, but from which machine.

Executing that information against the policies defined above and in conjunction with the Device Inventory Database, the access proxy is now in a position to make a decision whether to allow traffic to the application or deny it.

One final thing to note about BeyondCorp is that it is aspirational. Zero trust in general is a strategy, and BeyondCorp is simply one method to achieve it. Google currently does not make this software available outside of cloud deployments and has altruistically offered up their thought leadership to the community about how to achieve their level of zero trust.

Akamai Cloud Security Services

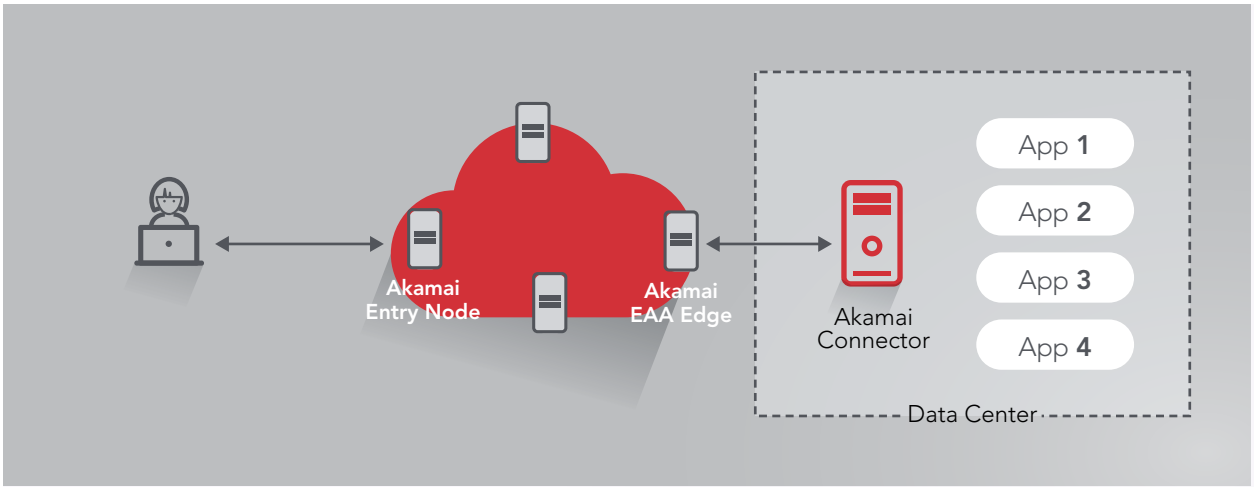
In the classic BeyondCorp model, the access proxy is a server placed inside of a DMZ in order to grant or deny access to services within the micro-perimeter. If you think about the amount of work such a server has to do, it's pretty daunting.

At minimum, the access proxy is responsible for terminating the client-side TLS connections, enforcing authentication, querying the device inventory database, applying policy, and then proxying connections to services within the micro-perimeter, many of which will also need encryption. This is already quite a bit of work, but if we then add on services like a web application firewall (WAF) and behavioral analytics, the load can be downright intense.

And that's just CPU. Certain functions such as caching, while possible from the data-center side of an Internet uplink, offer far more benefit when placed on the Internet side of the uplink, closest to the requesting users where there is effectively infinite bandwidth.

While the standard BeyondCorp access proxy architecture is a quantum leap forward in security, the characteristics of running it entirely within your own DMZ limits the ability of the cloud to absorb attacks, provide infinite bandwidth for caching, and autoscale resources as needed.

Akamai is a cloud-native company that has designed a slight architectural modification to the classic access proxy, which allows it to live in the cloud, scale on demand, execute CPU-heavy resources on our network instead of your equipment, absorb attacks, and deliver cached content closest to your users. We call this Enterprise Application Access, or EAA for short, and it looks as follows:



In this architecture, you migrate your applications to a micro-perimeter, exactly as you would in the classic BeyondCorp model. However, instead of placing your access proxy in the DMZ, you run a small VM called an Akamai Connector *within* your micro-perimeter. It does not need to be, nor should it be, inside the DMZ. Its address should be on private IP space and not directly reachable from the Internet. In fact, it should look exactly like any other application you would place within the micro-perimeter.

When the Akamai Connector starts up, it immediately establishes an encrypted connection to the Akamai cloud. Once connected to Akamai, it downloads its configuration from Akamai servers and is ready to service connections.

When a user of your internal applications attempts to access a service within the micro-perimeter, they are directed to Akamai via a DNS CNAME, and connect to an Akamai Entry Node. It is at this node, closest to the user, where Akamai can apply things such as WAF, bot detection, behavioral analytics, and caching. This gives us best-in-class performance as well as the ability to keep potential threat actors as far away from your physical locations, applications, and data as possible.

Assuming the end user passes all checks, they are then routed through Akamai's advanced network overlay to the Akamai EAA Edge, where normal authentication, multi-factor authentication (MFA), single sign-on, and device identity functions are performed. Assuming the user and machine are authorized, the connection from the client is then stitched together with the outbound connection from the Akamai Connector. Traffic from the user session flows through this stitched proxy connection to the Akamai Connector, which then connects to the requested application or service. At that point, a complete data path is established.

There are a distinct and significant advantages to this method of access. The activities that are most performance and security sensitive take place at the network edge closest to the end user, where Akamai has more than 200,000 machines spread around the globe. Additionally, the sensitive ingress path into the micro-perimeter happens over a reverse application tunnel, effectively removing the IP visibility of the perimeter and reducing the risk of volumetric attacks.

The use of this paradigm does not fundamentally change the concept of anything learned thus far. It simply makes it more efficient. As we discuss phasing, we will reference this approach as we feel it is faster and safer, but the general concepts apply even to the classic BeyondCorp access proxy approach.

Application Access vs. Network Access

Readers of this might be inclined to think about this as a VPN, but they would be doing themselves a great disservice. VPNs provide network-level access, and due to their technological underpinnings, ensure that security and performance are inversely related to manageability and simplicity.

If you opt for a simple VPN setup, you probably do what many companies do — you allow logged-in users to have IP-level access to your entire network. We know how dangerous this is. Why should call center employees have IP access to source code repositories? Or why should a contractor using your billing system have access to the credit card processing terminals? Access should be to just those things needed to perform a role.

To fix this, you can begin to partition applications via VLANs onto separate segments behind a firewall and enforce archaic IP range–based rules for individual users or groups at the VPN aggregator. This is brittle and very prone to errors. Often, administrators find that connectivity breaks at the worst possible times. Maybe someone is doing maintenance and moves machines to a new rack or needs to re-IP them to a new range. All of a sudden, users are locked out and support calls come rolling in. Or perhaps an application's architecture changes during a software upgrade and users are redirected to another machine as part of the workflow, but that machine is inaccessible to certain users or groups because the firewall rules were not updated. This architecture requires all changes to have a very high degree of communication between application owners, network administrators, and security groups to ensure zero downtime.

Historically, we have significant evidence of what often happens when the above coordination fails. Administrators want to follow best practices, but in times of desperation, the dreaded IP ANY/ANY ALLOW rule gets added as a quick fix to allow affected users to access everything until the problem underneath can be diagnosed and repaired. But there often isn't time to go back and fix past holes. Again, to overcome the security downsides of unfettered horizontal access, significant complexity and operational overhead needs to be introduced when using a VPN, and that complexity often results in holes and quick fixes that worsen security posture over time.

The same tradeoff happens regarding performance. In a VPN's simplest form, all traffic is directed back toward the data center. This can result in extremely slow access to Internet properties and SaaS due to the hairpin effect, as well as increased congestion on Internet uplinks within the data center for non-business traffic such as Facebook and YouTube. Why backhaul normal Internet access for a user who is already off premise?

To overcome this performance burden, administrators often deploy split tunnels, again marking which IP ranges should travel down the VPN and which should egress directly to the Internet. This is very effective and simple when you only have one internal perimeter. However, it begins to get much more complex as you add multiple data centers and virtual private clouds (VPCs) in IaaS providers. Administrators must then determine if they are going to install VPN aggregators in every data center and how to manage multipoint split tunnels effectively.

Again, all of these solutions are theoretically possible. You may already be using some combination of them. The problem is that the tasks required to do them well, maintain them, and provide proper security and performance over the lifetime of their implementations are often far too operationally complex to continually get correct. In many cases, companies convince themselves that because employees can access their applications, everything must be working optimally. They then find themselves caught off guard when one of the quick fixes above results in a catastrophic breach or a performance degradation that causes an outage or vastly decreased employee productivity.

This is not to say VPNs don't provide value. Far from it, in fact. Site-to-site access for multiple data center infrastructure is one case where they shine. It is simply to say that network-level access is not the correct paradigm to use when discussing users accessing applications. This is because network-level access enforces an unnatural compromise in simplicity vs. security and performance.

What makes proxy-based approaches like BeyondCorp and Akamai Enterprise Application Access so appealing is that it is application-level access. With application-level access, performance and security are *decoupled* from complexity. This is inherently obvious in how easy it is to explain.

You simply take all applications that have locality with each other (all hosted in the same data center or same VPC, for instance), place them into a private network IP space or a restricted VLAN, place an access proxy in the DMZ of that micro-perimeter, and you're finished.

Application owners set their own security policies on the access proxy — who can access and at what device posture — and even more interesting, users can be anywhere. There is no distinction of on-premise vs. off, because there is no network perimeter that includes the end users. An employee in a coffee shop is equal to an employee in an office. All that matters is whether or not the user is authorized and whether or not the machine is safe.

With application-level access, performance is theoretically best in class, despite the ease of deployment and use. Users simply utilize the raw Internet to access applications directly, no matter where they are hosted or where the user is, allowing the Internet to route packets to their destination without having to go through aggregators or intermediaries that aren't in path.

In fact, with application-level access, internal networks often dissolve into simple Guest Wi-Fi. Remember, for zero trust to truly be effective, you cannot treat internal users any differently than external. They are all untrusted by default.

Desired End State

Application Access

In a modern zero trust deployment, all applications should be segmented into their own micro-perimeters based on location and purpose. These micro-perimeters should be on private VLANs with private IP space, and directly inaccessible from anything other than the access proxy that fronts them.

We leave it to the discretion of each company to determine if they wish to provide even further micro-segmentation *within* the micro-perimeters to stop horizontal escalation and movement if an application is compromised. We see theoretical value in such exercises, but in practice find that the level of complexity in getting such segments right is often far too high to justify their added security. Anyone who has tried to unwind the touchpoints that a complex application has behind the perimeter can testify as to how brittle it can be to segment applications from each other. However, if your environment allows for this in an easy and maintainable way, we encourage its adoption.

All users, whether or not they are on- or off-premise, should be forced to access all applications through access proxies, such as Akamai's Enterprise Application Access, which perform not only standard authentication, but MFA as well. Additionally, there should be a robust Device Inventory Database keeping track of the security posture of all devices allowed to access your resources should application owners wish to craft policy around the device postures themselves. Some form of certificate management and/or mobile device management (MDM) will likely be needed to provision certificates on the machines as well as revoke certificates when security demands.

In addition, we strongly believe that zero trust does not end with authentication and authorization. A modern stack will also require some level of inspection and analytics of the traffic passing through the access proxies. This will help shield against advanced persistent threats and willfully malicious end users.

One crucial security system that should be layered on top of your access proxies is a WAF to ensure that application-level attacks are not being launched (intentionally or inadvertently) from your end users toward your internal applications. Other advanced systems such as human/bot detection can be leveraged for non-API sites to help ensure malware is not masquerading behind valid endpoints.

As you bring your applications online and make them accessible through access proxies, DDoS prevention becomes even more important. You should align yourself with providers who can absorb attacks against your micro-perimeters and access proxies, allowing continued operation under intense loads.

And finally, to ensure performance is best in class for your applications and that users not only accept this shift in access, but champion it, your access proxies should be fronted by networks that can provide performance benefits. Specifically, CDNs and Internet routing overlays should be part of your arsenal to not only make access available, but make it more performant than prior methodologies have ever allowed.

Internet Access

It can be useful to think of solutions like Akamai Enterprise Application Access as protecting your applications from malicious actors. What then protects your users from becoming those very actors through malware infections? This is where prevention and detection become crucial for outbound traffic.

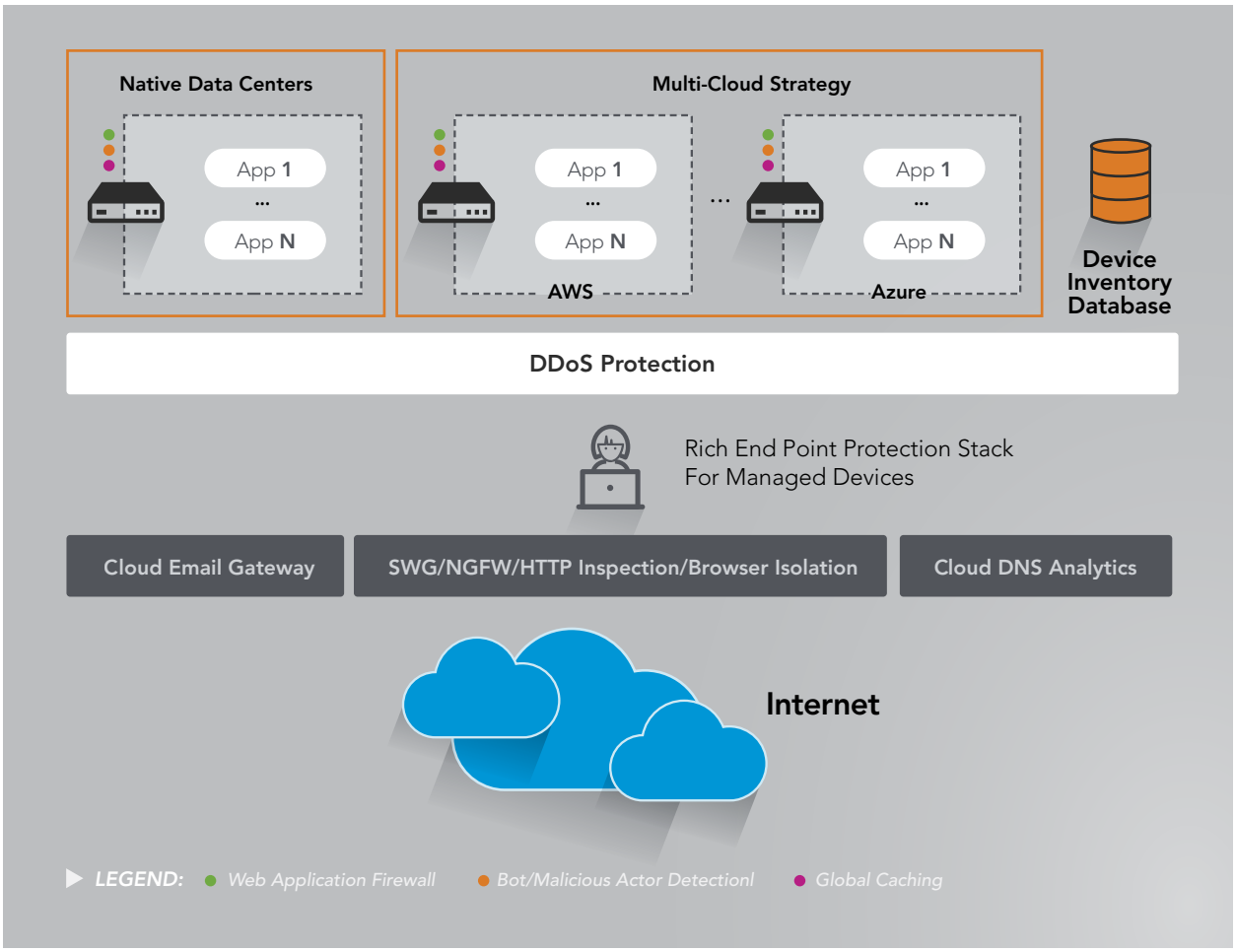
As such, endpoint security for managed devices will be even more important as we march into the future, and we expect machine learning–driven AV solutions to continue to outpace classic AV stacks in their abilities to identify zero-day threats. Furthermore, some companies may find that application whitelisting or desktop process isolation systems provide an even more comprehensive threat prevention story, although at the cost of rigidity in deployment that some may find too difficult to utilize.

In addition, cloud-based preventive stacks should be deployed to safeguard end users. Classic systems such as SWGs and NGFWs will continue to be important, but will be assisted by point solutions covering the highest-risk use cases. These include remote browser isolation, sandboxing and scanning of executables from the web, and email gateways that detect phishing and malware attachments.

While prevention is important, we feel detection will continue to be the most important toolkit administrators have to combat the constant threat posed to their users. One such system that is lightweight to deploy, and yet incredibly effective, is the DNS firewall. Since DNS is a common substrate used by both good and malicious software, it serves as an excellent signal to detect infestations, and the ease of deployment of such solutions will make them invaluable.

Architectural Visualization

Putting it all together, we expect your users and applications to look something like the following in a fully realized zero trust world:



Getting from A to B

We have seen a pretty comprehensive study of zero trust thus far, including how a complete zero trust/CARTA-enabled enterprise should look. However, much like security is best realized when it is easy, deployment of new architectures must also be simple and phase-able. No company of any significant size would be able to convert their entire network overnight to this vision. As zero trust itself is a strategy, so too is its deployment.

We believe the best approach to reaching a complete zero trust architecture is to begin transitioning applications into the Enterprise Application Access model in small batches that are manageable. The batch size will vary by the number of resources you can dedicate, with some companies only able to do one at a time. Regardless, each application being transitioned will go through several stages along its path to complete zero trust. At each stage, you will verify that the application is functioning correctly and that authorization is being enforced as expected.

As of early 2018, Enterprise Application Access supports HTTP(S), VNC, RDP, and SSH apps without the need for a client to be installed. Posture assessment and additional protocols through the use of a client are scheduled for later in 2018. As such, we will stagger web-based applications first in our phasing to align with current EAA capabilities.

Pre-Staging Assumptions

The following assumptions are made with respect to the pre-staging state of your network:

- Applications and users within your *current* perimeter may reach each other directly over IP.
- You have installed an Akamai EAA Connector into your *current* perimeter.
- You have created a walled-off VLAN for eventual placement of corporate applications.
- You have installed an Akamai EAA Connector into the walled-off VLAN.
- Your off-premise enterprise users will continue to have a VPN installed during the phasing in order to reach those applications not yet transitioned.
- If managed device identification is desired, certificates will be installed on end-user machines through some out-of-band method such as MDM.

With these prerequisites met, we now can begin discussing the stages that each application will journey through on its way toward a complete transition. These stages are meant to reflect the highest level of granularity, and as such, you may wish to combine some of them together as you become more accustomed to the process.

User Grouping

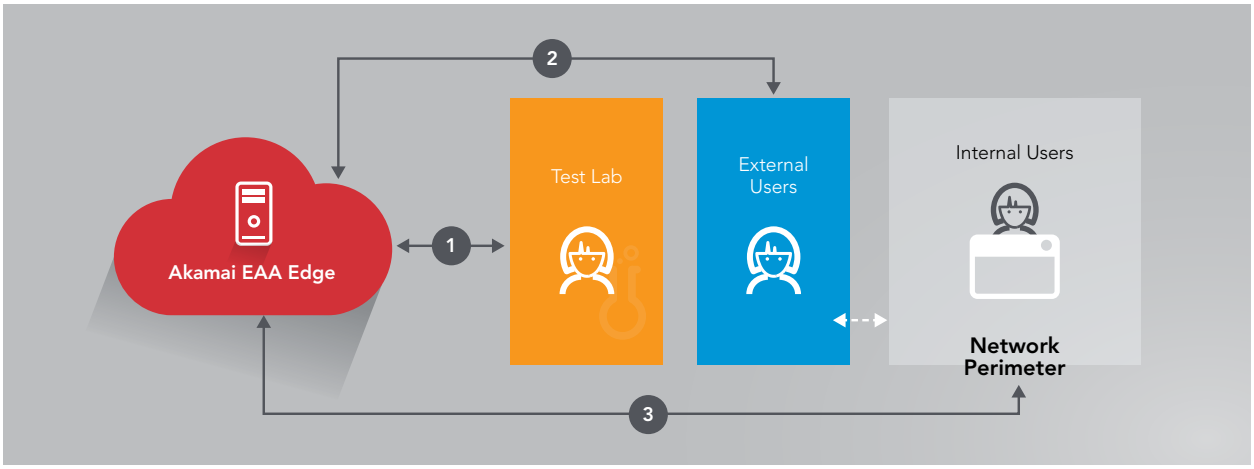
Part of our application staging process will involve slowly phasing this feature out to select groups of users. We propose having three general sets of users defined for each application:

- **Test Lab:** These users will be responsible for verifying the functional integrity of the application when it is first made accessible through Enterprise Application Access. They should be familiar with the application's operational semantics enough to test both standard behavior as well as edge cases.

  As certain additional security features and performance features are layered on, these users should also have the background to test that security is behaving as expected and performance gains are truly being realized.

  We encourage organizations that may have a large amount of applications to consider investing in tools that will allow standard non-application-specific features such as generic performance testing, and certain security checks to be performed regularly and automatically without the need for heavy human oversight. Specific security checks that should be performed with such tools are SQL injection, cross-site scripting, bot detection, and command injection.

- **External Users:** This will be the first set of production users that Enterprise Application Access is rolled out to and consists of all valid users who will be accessing the application outside of your network perimeter. This group will operate in a dynamic fashion. In other words, as users travel off-premise they will dynamically enter this group, and as they travel back within the network perimeter, they will dynamically leave.
- **Internal Users:** This will be the final set of production users that Enterprise Application Access is rolled out to, completing the migration to all users of the application. This set consists of all valid users within the network perimeter that will be accessing this application. Like the External Users group above, membership in this group is dynamic. Users will join this group automatically upon entering your internal network perimeter, and will join the External Users group upon leaving.



User Grouping Methodology

In our phasing example, we are going to be utilizing Akamai's EAA product as our access proxy. Akamai onboards applications into its distributed network through the use of DNS: Application hostnames are CNAME'd to Akamai to route users to our platform.

Therefore, users who do hostname lookups that result in the CNAME chain to Akamai will be on-ramped to Akamai's global network, while users whose lookups result in the original internal A record being returned will continue to access the application using the outdated perimeter method. We will take advantage of this architecture as a way to phase the different user groups onto Enterprise Application Access for each application by controlling which groups follow the CNAME chain and which users receive the internal A record.

We will achieve this through the use of DNS Views, also known as split-horizon DNS, to define the above three sets of users. A DNS View is a method by which two or more users querying the exact same hostname can receive completely different answers depending on what their source IPs are. For example, a user in China who queries www.foo.bar could receive one distinct IP, while a user in the United Kingdom could receive a completely different IP, even though they queried the exact same hostname.

We will utilize this method by organizing our different user groups by their source CIDR blocks. All Test Lab users will reside in one CIDR block, Internal Users will be within the CIDR block that defines your entire internal network, and External Users will be sourced from all IPs that don't match the first two sets.

All common DNS servers that enterprises deploy in modern environments today support this feature. As an example, we give a sample configuration for the open-source ISC BIND package in the appendix, where the applications you wish to enable Enterprise Application Access for live within the yourhost.com domain. Now that we understand how to partition the various user groups, we can begin discussing the actual application rollout stages.

1. Application Precheck Stage

In this stage, you check to make sure that the application meets the requirements of the access proxy you have deployed. In the case of Akamai EAA, you will make sure that the application is of a supported protocol: HTTP, HTTPS, RDP, or SSH.

2. Access Proxy Preparation Stage

Next, you will configure your access proxy to be aware of the application as well as its specific security and access rights. In the case of Akamai EAA, you will configure this in the cloud and the configuration will be pushed to all of your Akamai Connectors instantly as well as the entire Akamai network, such that all Akamai Edge machines are capable of servicing your end users.

3. Test Lab Enrollment Stage

Now that the access proxy is aware of the application in question, we can begin onboarding users. In this stage, assuming we are utilizing the Akamai EAA access proxy, you will modify your DNS Views as noted above such that members of the Test Lab group get CNAME'd to Akamai for the specific application hostname upon lookup.

Immediately following this action, members of the Test Lab will be directed into Akamai's global network whenever they attempt to access the given application.

During this time, Test Lab members should verify authentication is working correctly, multi-factor authentication is appropriately configured, and single sign-on works across all other previously onboarded applications. More importantly, in-depth tests around the application's functional correctness should be run at this time.

4. Security Upgrade Stage

Now that members of the Test Lab are safely able to access the application using Enterprise Application Access, you should consider enabling advanced security features that otherwise were impossible in the traditional perimeter model. In this section, we are going to reference specific Akamai security products that work well with, and are enabled by, the Akamai EAA access proxy. However, whether you use these specific products or not, the general recommendations and deployment stage remain valid.

We recommend a minimum of enabling the Akamai Kona Site Defender product in order to get WAF functionality. Once this is engaged, members of the Test Lab should verify that SQL injection, cross-site scripting, and command injection attacks against the application are rejected by the WAF platform.

Another capability that can be easily enabled if you are utilizing the Akamai EAA access proxy in conjunction with Akamai Ion is Lightweight Clientless Posture Assessment. Using minimal configuration, application owners can set policies indicating which browsers and operating systems should be denied access to the application. For example, through header inspection, Akamai is capable of filtering out obsolete versions of Firefox, which may pose security risks.

It is in this stage that application owners should also consider bot vs. human detection as a simple yet incredibly powerful addition to security. If the application in question is not an API server and should never be accessed programmatically, you can enable this feature through the Akamai Bot Manager product and instruct the platform to reject connections that cannot be determined with a high degree of certainty through advanced analytics to have originated from a human. This goes a long way toward shutting down malware and other advanced persistent threats that masquerade behind valid user sessions.

This stage is also where you will determine if the given application should be restricted to managed devices only. If so, and you deploy certificates to your end devices, you can upload your public Certificate Authority certificate to Akamai EAA such that it can reject all connections originating from machines that are unmanaged.

Finally, geography and IP-based restrictions can be enabled as well. If you know of specific CIDR-based whitelists or blacklists that should be used, or of geographic regions that should be denied access, this can be enabled and tested at this time.

Regardless of the features you enable, it is during this stage that your Test Lab will be expected to ensure that the security options are not only working, but are not inhibiting the functional correctness of the application.

5. Performance Upgrade Stage

With the application onboarded and security now implemented, the next thing to consider is if performance is deficient. In traditional perimeter-based access and security, enterprises are often limited in performance by the robustness of the application server and the enterprise's associated links between branch locations.

Sometimes features such as on-premise caching can be deployed to mitigate these issues, but they fall far short when employees go off-premise and travel, as caching elicits the greatest performance upgrades when it is closest to the end user.

In this stage, you will assess if your applications need a performance upgrade, and if so, utilize caching and other techniques available to you. If you have been using the Akamai EAA access proxy, one obvious upgrade would be to enable Akamai's Advanced Global CDN through our Ion product. This will enable caching worldwide across Akamai's quarter million servers, delivering best-in-class performance to your end users, regardless of where they are located. Additionally, by enabling this, you will gain access to Akamai's Advanced Overlay Network, where forward error correction (FEC), route optimization, and packet replication provides near-zero packet loss and performance-based routing.

It should be noted that this stage may optionally be delayed until the External User Enrollment Stage occurs, as External Users may be best positioned to assess the relative performance gains while outside of the network perimeter.

Either way, we recommend performance instrumentation at this phase *as well as* prior to application enrollment, in order to accurately assess the performance gains. This may be through the use of browser plugins, waterfall graphs, or third-party performance monitoring services.

6. External User Enrollment Stage

By the time you have reached this stage, the application has been onboarded and made accessible via Enterprise Application Access, had significant security upgrades applied, optionally been made much more performant, and tested by an internal Test Lab to gain confidence that it is functioning as expected.

It's now time to roll it out to a wider audience. At this point, we recommend phasing this to external users. They are the ones for whom this style of access will eventually lead to VPN removal. Additionally, they are the users most often affected by performance issues, and are in the most hostile environments where their very location puts your applications and data at risk. Utilizing better performance and security with this set of users is where the greatest benefits of Enterprise Application Access will be immediately visible.

We recommend prior to entering this stage via DNS View modification, notification is issued to these users such that they are not caught off guard by the transition. If all has been configured appropriately up to this point, the transition should effectively be nearly invisible to them, with the exception of increased performance. However, advanced notice will alert users to keep a particularly close eye on functional correctness such that if anything is amiss, they will know why and be prepared to contact the appropriate administrator.

7. Internal User Enrollment Stage

In this stage, external users have been enrolled for some time and all bugs in operation have been addressed. At this point, you can remove all references to the application from your DNS View specific configurations and add it as a CNAME entry to the common view. All users should then immediately begin accessing this application using your Enterprise Application Access proxy.

All procedures that were used to notify external users should be applied to Internal Users in this stage as well. It is hoped that through the prior six stages, any errors or misconfigurations will have been discovered and remedied. Assuming this is correct, it is at this point that the application will have successfully been *operationally* transitioned to Enterprise Application Access and all users should be enjoying the benefits of easier, faster, and safer access.

8. VLAN Migration Stage

After an appropriate amount of time has passed in the Internal User Enrollment Stage, you can finally shift the application into the walled-off VLAN. Before this occurs, even though all valid users are being directed through Enterprise Application Access using DNS, the application server itself is still reachable directly through its IP address and thus vulnerable to malware within your network perimeter.

This final stage removes all direct IP access, effectively walling the application off from anything but the access proxy itself.

Once this is complete, the application is officially and *completely* transitioned to Enterprise Application Access.

Post-Staging Operations

Once all applications have been transitioned to Enterprise Application Access, you can begin investigating the complete removal of VPN clients from end-user systems.

In addition, you may now consider taking the action of converting your internal user network into Guest Wi-Fi, as all application access is now transitioned to a zero trust application access model.

Finally, it is in this phase that you should consider protecting the connectivity back to your datacenter and micro-perimeters from advanced DDoS attacks. The Akamai Prolexic product is one such solution that can assist here.

Summary

Traditional hub-and-spoke networking architectures, along with the castle-and-moat security perimeter they utilize, simply cannot effectively provide performance or security in today's burgeoning cloud-and-mobile world. This is a problem all companies must begin facing, lest they be left behind in a vulnerable state. Failure to transition to safer enterprise security architectures is the number one cause of corporate breaches today, and it's only going to get worse. Simply put, you are not safe behind the perimeter, because the perimeter itself no longer exists.

Just how do we achieve zero trust?

Akamai's cloud security services can be combined to build a comprehensive and concrete zero trust architecture, not only enabling safe app access in a cloud-native world, but leveraging the cloud to remove the need for internal corporate networks almost completely.

By utilizing our advanced distributed access proxy with Enterprise Application Access along with the power of the global Akamai platform, you can finally transition to a perimeter-less world in an incredibly easy way, phasing applications in one at a time, reducing your migration risk profile to near zero, and leveraging the platform's extensive 20-year history of proven performance and security solutions.

As this field continues to evolve, you can rest assured that Akamai will be there with you at each step, helping you to transition your network to an architecture that not only provides access to your applications and data, but does so in an extremely easy-to-manage way while maintaining the highest levels of security and performance.

Appendix

All common DNS servers that enterprises deploy in modern environments today support DNS Views. As an example, we give a sample configuration for the open-source ISC BIND package, where the applications you wish to enable Enterprise Application Access for live within the yourhost.com domain. The main configuration file will look similar to the following:

```
named.conf
acl testlab { 10.0.2.0/24; };
acl internal { 10.0.0.0/8; };

view “testlab” {
    match-clients { testlab; };
    recursion yes;
    zone “yourhost.com” {
        type master;
        file “yourhost.com.testlab.zone”;
    };
};

view “internal” {
    match-clients { internal; };
    recursion yes;
    zone “yourhost.com” {
        type master;
        file “yourhost.com.internal.zone”;
    };
};
```

```
view “external” {
    match-clients { any; };
    recursion no;
    zone “yourhost.com” {
        type master;
        file “yourhost.com.external.zone”;
    };
};
```

We will use a single zone file for all entries that are shared amongst all views. These are the hosts and DNS entries that remain constant regardless of which user group you are in. It will be specific to your organization, but might look similar to the following:

```
common.zone
a$TTL 86400
@       IN      SOA     ns1.yourhost.com.    ns1.yourhost.com. (
                        2001062501 ; serial
                        21600      ; refresh after 6 hours
                        3600       ; retry after 1 hour
                        604800     ; expire after 1 week
                        86400 )    ; minimum TTL of 1 day

yourhost.com. IN      NS      ns1.yourhost.com.
yourhost.com. IN      NS      ns1.yourhost.com.

ns1.yourhost.com.     IN      A       10.0.1.1
ns2.yourhost.com.     IN      A       10.0.1.2

server1.yourhost.com.       IN      A       10.0.1.101
server2.yourhost.com.       IN      A       10.0.1.102
additional common entries…
```

Each user group that has been transitioned to access an application using Enterprise Application Access will have a specific zone file that looks similar to the following (where app1.yourhost.com is the application that has been transitioned in this example):

```
$INCLUDE “common.zone”;
app1.yourhost.com.   IN     CNAME  app1.yourhost.com.sohacloud.akamai.com.
```

Zone files for user groups who have yet to transition to accessing the application via Enterprise Application Access would look as follows (where the IP given is just an example):

```
$INCLUDE “common.zone”;
app1.yourhost.com.   IN     A      10.0.1.105
```

Putting this together in a more concrete example, if the Test Lab and External User groups have transitioned to accessing an internal Jira application via Enterprise Application Access, but Internal Users are still using the original perimeter style access, the zone files would look similar to the following:

```
yourhost.com.testlab.zone
$INCLUDE “common.zone”;
app1.yourhost.com.   IN     CNAME  jira.yourhost.com.sohacloud.akamai.com.
yourhost.com.external.zone
$INCLUDE “common.zone”;
app1.yourhost.com.   IN     CNAME  jira.yourhost.com.sohacloud.akamai.com.
yourhost.com.internal.zone
$INCLUDE “common.zone”;
app1.yourhost.com.   IN     A      10.0.1.156
```