Hewlett Packard Enterprise

Technical white paper

Check if the document is available in the language of your choice.

BUILDING A FOUNDATION FOR ZERO TRUST WITH SPIFFE AND SPIRE

CONTENTS

ZERO TRUST SECURITY MODEL ESSENTIAL TO DEAL WITH DYNAMIC INFRASTRUCTURE AND EMERGING THREATS

The nature of cybersecurity threats is changing. Cybercriminals and nation states have developed a wealth of techniques to morph their attack patterns and bypass many of the traditional perimeter-based security systems and networks in place. The result is that the attacks being waged today may not look anything like they did yesterday.

In the Zero Trust security model, organizations do not automatically trust anything inside or outside their perimeters. The default assumption is that everything should be denied access—no matter its origin—until it has been verified.

The Zero Trust model abolishes long-standing assumptions, such as the relative safety of an organization's own internal services communicating with each other within the firewall. Any service an organization runs that requires access to any other of the organization's services must identify itself and be authenticated and subsequently authorized. Organizations with a massive installed base of interconnected applications may see this model as a paradigm shift, but the business drivers that increase its need are compelling.

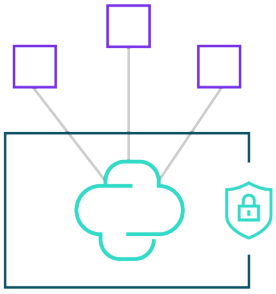
Widespread adoption of public cloud

The perimeter is moving away from the data center and is now crossing multiple clouds. This is because most organizations have embraced public cloud. Public cloud platforms enable teams to deploy innovative services far more rapidly and frequently, changing from one deployment every few months to many per day.

Traditional tools and protocols, which were designed for static infrastructures, are ill-equipped to work in today's environments. This is particularly true for cross-service authentication. When it comes to authentication, static identifiers like IPv4 or IPv6 addresses are largely meaningless in today's elastic, ephemeral environments. Strategies like employing firewall rules to establish network perimeters are now effectively obsolete; these approaches are simply too static.

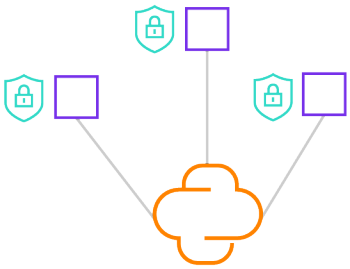
Protecting the integrity of service-to-service communications is increasingly important with services now running inside and outside data centers. Zero Trust is particularly well suited to organizations that embrace the cloud.

Perimeter-based



- Perimeter-based
- Attempts to build a trusted wall
- Relies on IP addresses or physical locations
- Difficult to implement for today's dynamic environments

Zero Trust



- Assumes bad guys are everywhere
- Uses cryptographic identities for authenticating every system/user
- Enables universal enforcement across hybrid infrastructures

FIGURE 1. The evolution from perimeter-based defense to Zero Trust

Proliferation of microservices and containerization

The ability to provision and deprovision resources on-demand—one of the key benefits of public cloud—enables the high-velocity creation of suites of microservices which each have smaller scope of responsibility. Dividing complex applications into smaller, specialized fragments makes them easier to build, deploy, and maintain. But as the number of microservices grow, so does the surface area.

The ready availability of on-demand cloud resources has also created a need for application portability. Containerization, the process of packaging up an application's code and all its dependencies, meets that need. A containerized application runs quickly and reliably from one computing environment to another, making it easy to move even proprietary applications from an on-premises server to the cloud and back again.

New security challenges with scalable architectures

Reliance on location-specific details such as IP address has several shortcomings in a world of dynamically scheduled and elastically scaled applications running in the cloud. While many organizations extend their existing network security practices to the cloud, this is increasingly challenging to manage as applications scale to multiple clouds, multiple on-premises data centers, and multiple geographic regions.

One option is to use conventional user name and password credentials for applications, but these are difficult to manage and can potentially be stolen.

SERVICE AUTHENTICATION IS FUNDAMENTAL TO THE ZERO TRUST SECURITY MODEL, YET EXISTING SOLUTIONS HAVE LIMITATIONS

The Zero Trust model is predicated on the assumption that every entity is untrusted and should be denied access to all resources until its identity can be authenticated. Service identity and authentication is perhaps the most fundamental piece of building Zero Trust environments. Existing approaches have limitations:

- **Shared secrets** such as static passwords and tokens are the simplest form of authentication, but they introduce risk and complexity. These secrets are a security risk since they can be easily compromised and are hard to rotate or revoke in case of a breach. Tools like secrets managers can typically control, audit, and securely store secrets, but they are on behalf of a service in a central storage vault. Workloads must individually authenticate to the vault before performing actions such as secret retrieval or data decryption. The challenge in using secrets managers is not only managing the content of the secret store, but also how to securely store the credential used by the workload to access the secret store itself. This is sometimes called "credential zero," or more broadly, the process of "secure introduction."
- **Authentication protocols** are responsible for generating long-lived identity documents in various formats for workloads when they interact with other systems. These may include, access or refresh tokens (OAuth) or Kerberos tickets. But generating the identity documents require management such as rotation. In general, these authentication protocols are not designed for modern, dynamic environments.
- **Cloud provider tools** provide authentication mechanisms and tools for services running in their platforms. But these tools are generally limited to their platform, which makes cross-platform, service-to-service communication difficult. Moreover, because services can be uncoupled from any underlying platform, yet still need to retain their unique identity, identity credentials cannot be platform-dependent.

SPIFFE AND SPIRE: UNIVERSAL SERVICE IDENTITY CONTROL PLANE FOR THE ZERO TRUST SECURITY MODEL

Inspired by the production infrastructure at Netflix, Google™, Facebook, and others, the SPIFFE and SPIRE projects provide a universal identity control plane for establishing trust among services across dynamic, heterogeneous infrastructure.

SPIFFE is a set of open-source standards for securely identifying software systems in dynamic and heterogeneous environments. Services that adopt SPIFFE can easily and reliably mutually authenticate wherever they are running. SPIRE is a production-ready implementation of the SPIFFE that performs node and workload attestation to securely issue identity credentials to services and verify the identity of other services based on a predefined set of conditions.

SPIFFE and SPIRE allow you to give cryptographic, platform-agnostic identity to individual services across heterogeneous environments and organizational boundaries.

SPIFFE design goals include:

- **Platform-agnostic, cryptographic-based identity:** SPIFFE and SPIRE allow you to give cryptographic, platform-agnostic identity to individual services across heterogeneous environments and organizational boundaries. It provides a secure identity, in the form of a specially crafted X.509 certificate, to every service in a modern production environment. Services can use these identity documents when authenticating to other services, for example by establishing a TLS connection or by signing and verifying a JavaScript web token (JWT).
- **Multifactor attestation:** SPIRE orchestrates a real-time, zero-trust attestation process that can lean on a configurable union of trusted third parties to provide a strongly attested identity. Multifactor policies used to issue an identity establish greater trust in provisioned identities. This reduces the reliance on secret management and network-based controls.
- **Automated lifecycle management:** SPIRE automatically issues, distributes, and renews short-lived credentials to ensure that the organization's identity architecture accurately reflects the operational state of the services. The entire process occurs without human intervention and significantly reduces the operational overhead associated with credential management.
- **Short-lived credentials:** The identities issued by SPIRE can be configured to auto-new short- in anywhere from a few minutes to a few hours. This reduces the attack radius and eliminates the need to rotate or explicitly revoke credentials when they are no longer needed.

- **API-based identity management:** The SPIFFE workload API allows services to obtain their identities without the need for authentication. This API abstracts cloud provider identity interfaces by providing a standard way of accessing identity and reduces the burden of developing and maintaining cloud-ware apps.
- **Extensible, web-scale architecture:** SPIRE allows operators and developers to easily extend support to identity providers, certificate authorities, and systems. It is designed for highly dynamic and distributed environments.

HOW SPIFFE AND SPIRE WORK

SPIFFE fundamentals

SPIFFE is a new industry-standard protocol for services to receive their identities. The standard itself does not mandate how the identities are created or secured. Instead, it specifies how an individual service can retrieve its own identity, so that services can be developed to rely on a single method for obtaining their identity.

SPIFFE consists of four parts:

- The standard for SPIFFE IDs, or plain text strings identifying a service
- The standard for SPIFFE Verifiable Identity Documents (SVIDs)
- The standard for Trust Bundles, or bundles of public keys that can be used to verify SVIDs
- The Workload API, a standard API that a workload can use to obtain its own SPIFFE IDs, SVIDs, and Trust Bundles and receive a push notification when one of these updates

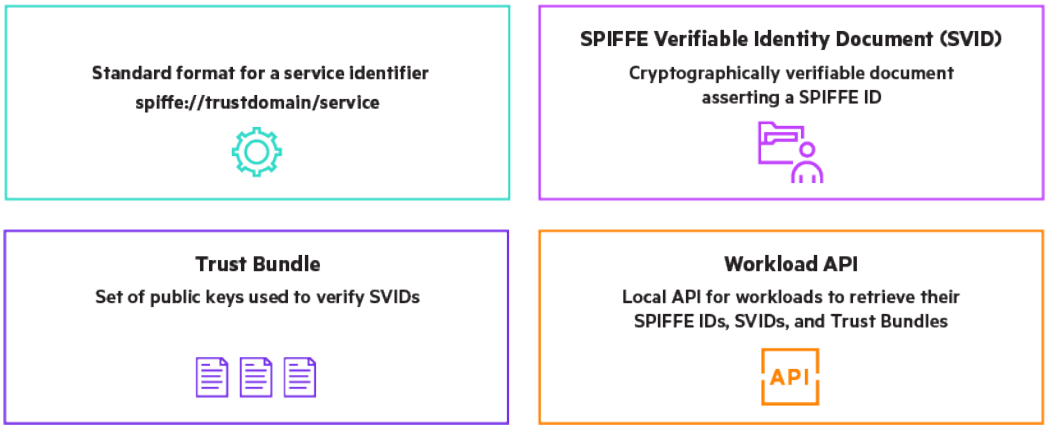


FIGURE 2. SPIFFE in four pieces

The SVIDs can be in one of two formats: X.509 certificates or JWTs. Certificate SVIDs can be used to establish end-to-end mutual TLS encrypted connections. JWTs are useful for situations where end-to-end mutual TLS encryption is not necessary or desirable, such as when using load balancers. JWTs are also useful for authenticating to a variety of cloud services that already support JWT-based authentication.

Whether using JWT SVIDs or X.509 SVIDs, the SPIFFE IDs, Trust Bundle format, and Workload API are the same.

SPIRE architecture

SPIRE is the reference implementation of the SPIFFE standard. SPIRE consists of two components: the SPIRE server, which is responsible for creating SVIDs and Trust Bundles, and many instances of the SPIRE agent, which are responsible for delivering identities to the workloads.

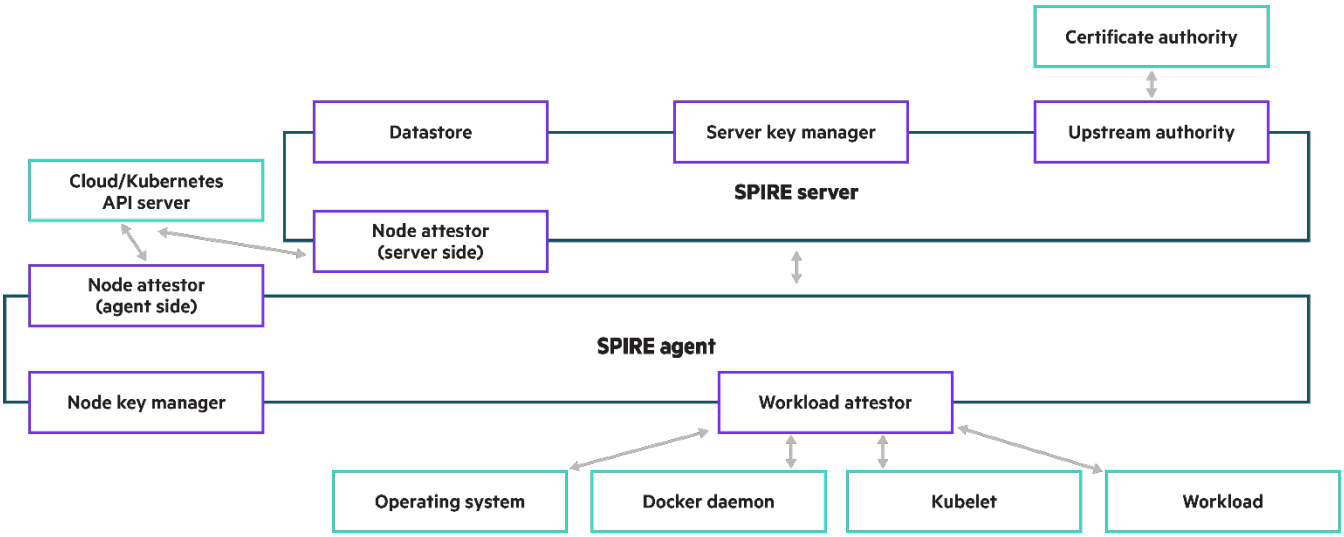


FIGURE 3. SPIRE is a pluggable architecture

Server startup

The SPIRE server is responsible for creating all SVIDs. It contains the signing keys, which means it typically needs to be in a highly secure isolated environment. (In some environments, it may be appropriate to run alongside workloads.).

When it starts up, the SPIRE server's first responsibility is to create its own signing key pair. It can either use a self-signed key pair or request a signed key pair from another source such as a corporate certificate authority. While in operation, the SPIRE server periodically rotates this key pair, typically every four hours.

The SPIRE server generates a Trust Bundle in SPIFFE format based on its own public keys and posts it in a well-known location for agents to access. This is called the "Bootstrap Trust Bundle." This can be distributed through file sharing, HTTP, or platform-specific mechanisms like S3 objects or Kubernetes (K8S) ConfigMaps. Each time the signing key pair rotates, the server updates the Trust Bundle.

Node attestation

The SPIRE agent runs alongside the workloads. Typically, there is one SPIRE server per bare-metal server, VM, or cloud instance. While it is possible to run multiple copies at the same time, it is generally unnecessary.

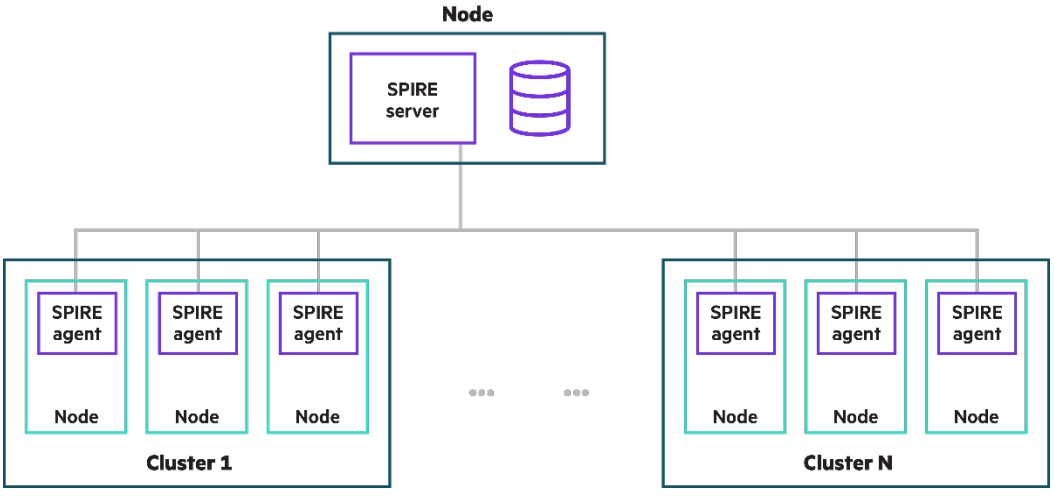


FIGURE 4. A single dedicated SPIRE server

When the SPIRE agent signs up, its first job is to retrieve the Bootstrap Trust Bundle that the SPIRE server previously posted. Based on the Bootstrap Trust Bundle, the SPIRE agent connects to the SPIRE server and validates the server's identity.

Next, the SPIRE agent must prove its identity to the SPIRE server. This process is called "node attestation," and several different methods are supported. On cloud instances or on K8S, the SPIRE agent can use mechanisms that are specific to each cloud provider to prove the identity of the instance to the server. For bare-metal instances or VMs, other methods are supported for the agent to prove its identity including Bootstrap certificates, string join tokens, and others.

Node: A logical or physical entity that runs computational workloads in a computer systems context. A Node is the underlying compute system that a workload runs on.

Workload: A workload is a single piece of software, deployed with a particular configuration for a single purpose that can consist of multiple running instances, where all of them perform the same task.

Workload attestation

SPIRE operation is controlled by registration entries. Each registration entry maps the properties of a SPIRE agent or workload to a SPIFFE ID. The exact properties that can be chosen depend on the plugins that are loaded. Examples include:

- For SPIRE agents: cloud instance name, cloud instance tag, presence of a pre-loaded certificate or join token
- For workloads: user ID and group ID of the workload, hash of the binary file that contains the workload, Docker image ID, K8S service accounts and namespaces

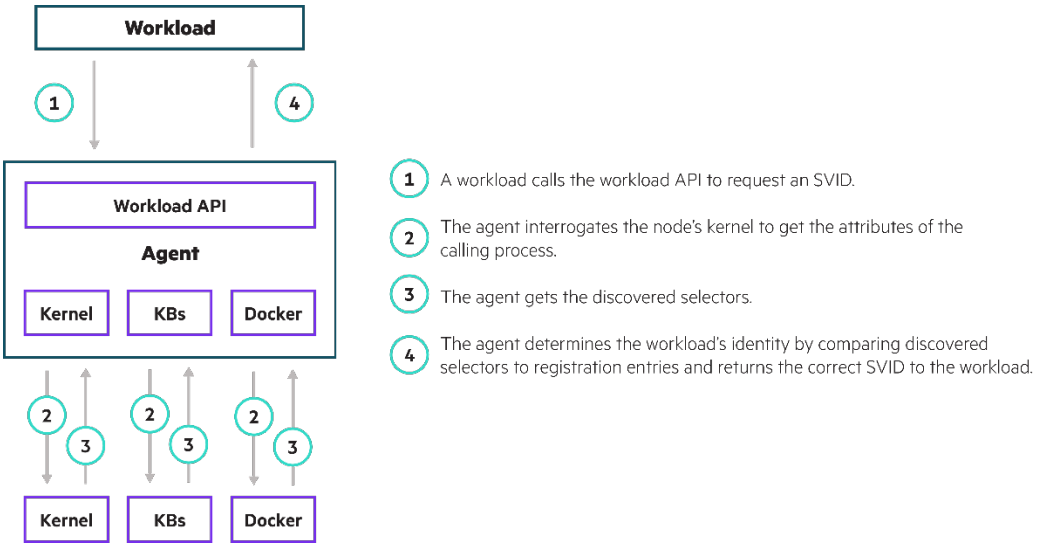


FIGURE 5. Workload attestation

By combining these, it is possible to create registration entries that assign a SPIFFE ID to a specific group of workloads running on a specific group of nodes. For example:

- If a service is running on a cloud instance with the PostgreSQL tag and its binary matches the PostgreSQL image hash, assign it the SPIFFE ID for PostgreSQL services.
- If a service is running in K8S on a node tagged with "frontend" in an image tagged with "frontend-service" and under the service account "frontend-service," then assign it the SPIFFE ID for frontend services.

These policies can be changed dynamically. All the needed SVIDs are generated and signed on the SPIRE server, and then pushed to the necessary SPIRE agents. (SPIRE agents do not receive SVIDs that they will never need to distribute, both for performance reasons and to limit the blast radius in case a SPIRE Agent is compromised.).

When a workload connects to the SPIRE agent, the SPIRE agent performs a process called "workload attestation." In workload attestation, the properties of the workload are retrieved from the operating system, from Docker, or from K8S and then compared against registration entries that already exist. If the workload matches a registration entry, then it receives SVIDs for that registration entry.

Workloads
Once the SPIRE agent is running, workloads can connect to it and retrieve their own SPIFFE ID, SVID, and Trust Bundle. As these change, they are pushed automatically. (In certain situations, a workload might have access to multiple SPIFFE IDs and SVIDs, and there may be multiple active Trust Bundles. However, in most configurations there is only one of each.).

Workloads connect to the SPIRE agent using the Workload API. This is a gRPC API, similar to many APIs used for other cloud-native infrastructure tools. In some cases, it may be challenging to use the Workload API directly. For these cases, an Envoy sidecar proxy can connect to the workload API, retrieve SVIDs and Trust Bundles, and encrypt and authenticate traffic. Envoy is a widely used proxy capable of handling large numbers of connections and is fully supported by SPIRE.

"With SPIRE we can deploy a consistent, 'dial-tone' authentication across all our platforms. The burden of authentication and security is now encapsulated from the developers so they can focus on business or application logic. This has improved our deployment velocity overall. We are also less likely to get 'production errors' due to configuration issues such as using development credentials in production. Standardized authentication with SPIRE has also simplified compliance and audit since we have mutual TLS across trust domains and platforms."

– Eli Nesterov, Security Engineering Manager, ByteDance

Federation and nesting
In large organizations, a single SPIRE server is not enough. SPIRE supports two mechanisms for scaling beyond one server: federation and nesting. In federation, two peer SPIRE servers exchange Trust Bundles, and these Trust Bundles are distributed to every SPIRE agent and workload. In nesting, a higher-tier SPIRE server generates SVIDs, which are then themselves used as signing certificates by downstream SPIRE servers. These two approaches to scaling have different security and availability properties.

Pluggability
SPIRE interacts with a variety of upstream certificate authorities, cloud providers, and workload platforms like K8S. The SPIRE core does not contain specific code for any of these outside services. Instead, it is a fully pluggable architecture. As new platforms arise, it is easy to write small SPIRE plugins that add support. Many organizations have unique requirements and write their own plugins to support them.

"SPIRE is now a key component of Uber's next infrastructure, but we are also using a side-car approach to retrofit authentication into legacy infrastructure. While SPIFFE and SPIRE are commonly known to work in modern, cloud native architectures, we can adapt the projects to our proprietary legacy stack quickly. SPIRE can provide a critical bridge of trust within Uber's next-gen and legacy infrastructure and positively impact internal security and developer efficiency."

– Ryan Turner, Software Engineer 2, Uber

BENEFITS OF ADOPTING SPIFFE

- **Enable defense in depth:** Provides strongly attested identities to reduce the likelihood of breach through credential compromise
- **Reduce operational complexity:** Consistent, automated management of identity to reduce the burden of operations and development teams
- **Interoperability:** Simplifies the technical aspects of service interoperability and integration across multiple stacks
- **Compliance and auditability:** Enables mutually authenticated TLS and multiple roots of trust to meet regulatory requirements

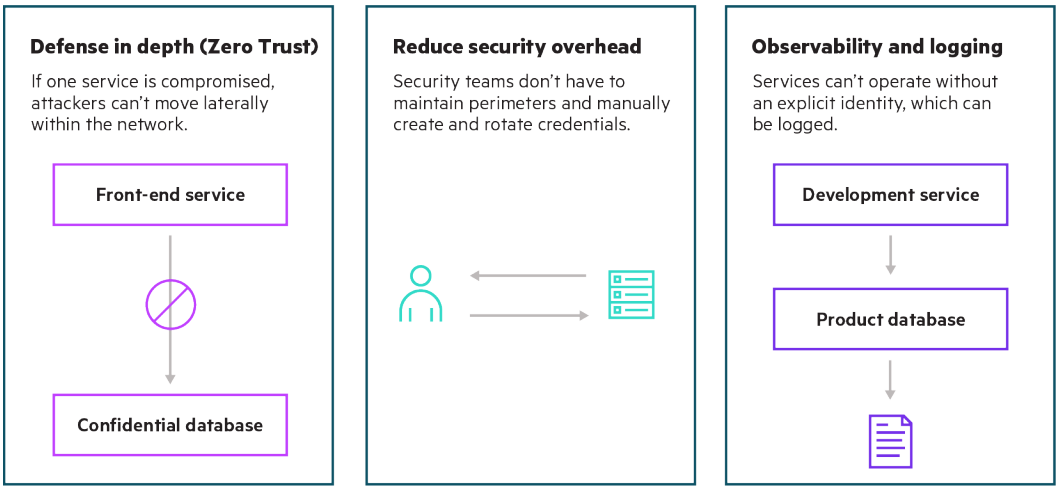


FIGURE 6. Three key benefits of SPIFFE adoption

"We could not rely on traditional perimeter-based security tools and processes to secure our next-generation applications and infrastructure. Zero Trust, a fine-grained, automated approach to security, made a lot of sense to us."

– Bobby Samuel, VP AI Engineering, Anthem

A SPIFFE CUSTOMER CASE STUDY

Anthem: Securing cloud-native healthcare applications with SPIFFE

Bobby Samuel, VP AI Engineering, Anthem

Rising healthcare costs in the industry are compelling organizations like Anthem to rapidly innovate and rethink how we interact with providers, employer groups, and individuals. As part of this initiative, we are developing a host of applications that will help us drive costs down by securely opening healthcare data access.

We have started building the supporting next-generation infrastructure based on cloud-native technologies such as K8S. This new infrastructure will drive rapid innovation and engage a broader ecosystem of organizations and developers. An example of this would be our HealthOS platform. HealthOS will enable third parties to build Health App capabilities to deliver into front-end interfaces, leveraging an ocean of de-identified health data.

But at nearly every major enterprise, especially healthcare organizations, someone is trying to get their data with malicious intent. Protected healthcare information (PHI) sells for much higher than financial information. Thus, malicious actors such as hackers and script kiddies find healthcare systems and the corresponding health information highly lucrative. With the adoption of cloud-native architectures, the risk and complexity rise further. The risk of a breach is even higher since the threat radius increases significantly while manual security reviews and processes become cloud-scale inhibitors.

Building a foundation for Zero Trust architecture

We could not rely on traditional parameter-based security tools and processes to secure our next-generation applications and infrastructure. Zero Trust, a fine-grained, automated approach to security, makes a lot of sense to us, especially for the future, as we plan to operate across organizational boundaries and cloud providers.

Identity and authentication for both users and services are among the Zero Trust security model's core principles. Zero Trust allows us to rely less on network-based controls than on authenticating every system or workload. SPIFFE and SPIRE have enabled a foundational authentication layer for our Zero Trust security architecture. They allow each workload to cryptographically prove who they are before they start communicating.

A shift away from secret management

Typically, when you think of authentication, you think of user names, passwords, and bearer tokens. Unfortunately, these types of credentials were becoming a risk and source of complexity at Anthem. They tend to be long-lived, and management or rotation of them is tricky. We wanted to shift away from this type of secret management practices in general. Instead of asking a service, "what do you have?" we want to ask, "who are you?" In short, we wanted to move to cryptographic identities, such as SPIFFE. We can see additional benefits of using a strongly attested identity in the future, such as establishing mutual TLS between workloads and bubbling identity up to the applications.

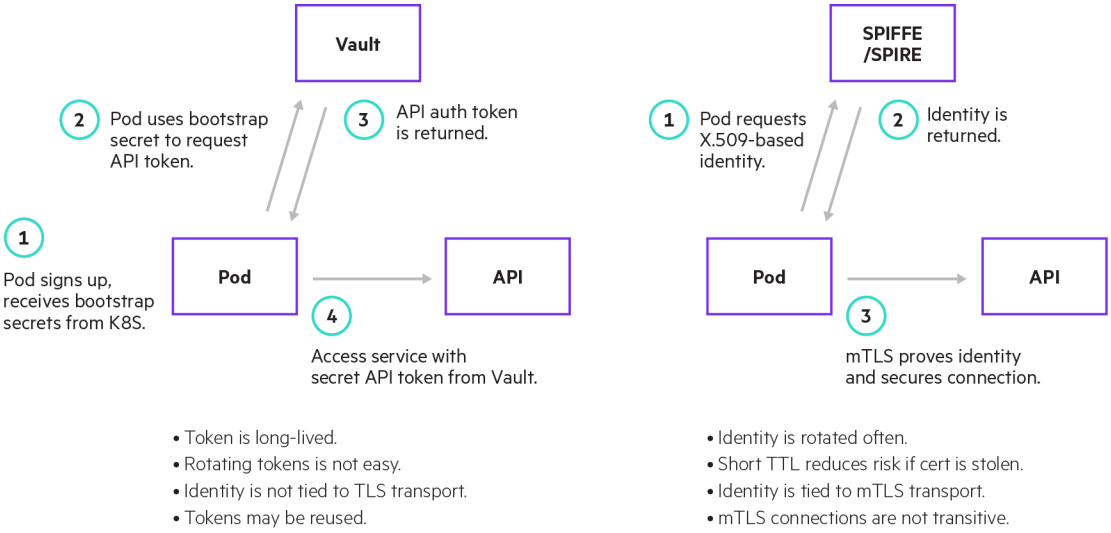


FIGURE 7. Enabling cryptographic identities with SPIFFE

Building security as part of the infrastructure with SPIFFE

Security is often considered an inhibitor to deployment by the development teams. DevOps teams want to deploy new innovative features faster. However, they must go through manual tickets, processes, integrations, and reviews related to security controls.

At Anthem, we doubled down on removing obstacles for our development teams by making security a function of the infrastructure. With the adoption of technologies like SPIFFE, we can abstract the complexity of security controls away from development teams and provide consistent rules across various platforms. SPIFFE, along with other Zero Trust–based technologies, will help us drive our system provision time from three months to less than two weeks in most scenarios.

Security is becoming an enabler at Anthem with SPIFFE leading the charge.

CONCLUSION

As organizations evolve their application architectures to incorporate emerging infrastructure technologies, their security models must evolve too. Software today takes the form of tightly linked microservices that may be spread across thousands of virtual machines in public clouds, private clouds, hybrid clouds, or private data centers, and security solutions must take forms that can provide adequate security in this environment. In today's rapidly expanding infrastructure world, SPIFFE and SPIRE help keep systems secure.

LEARN MORE AT

hpe.com/us/en/software/spiffe-spire-open-source.html

a50004027ENW, July 2021, Rev. 1

Hewlett Packard Enterprise