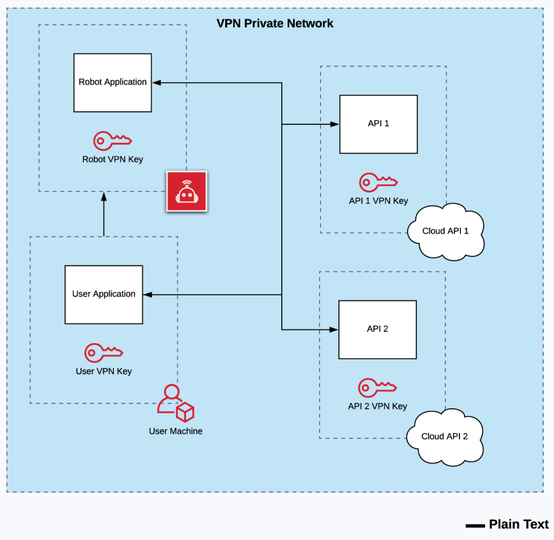
Formant

Formant security

Security by design

At Formant we treat security as a first-class feature, and evaluate all engineering decisions and implementations with the security of our stack and our customers data in mind. Security is a feature of our product that is constantly being improved due to the fast-paced nature of the field and the feedback we have from our customers.

The security of our platform covers data transport and storage, user/robot authentication and authorization, network protocols, and API design. This first post will cover how we secure end-to-end encryption and identity throughout our stack.



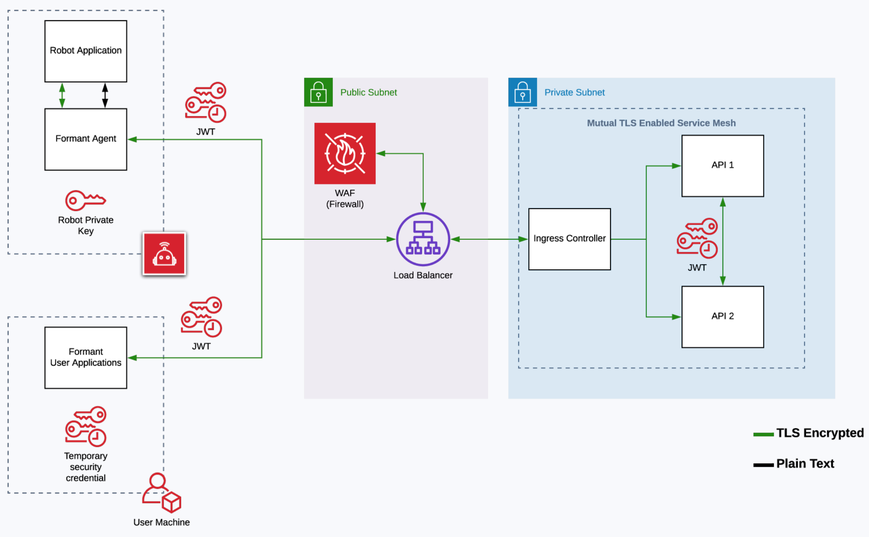
Traditional

Traditional

The following diagram depicts a classic implementation we see. It leverages a VPN to provide a private network.

This design is what's known as a crunchy outer shell or perimeter-based security. While making it easy to have robot applications and other cloud APIs communicate easily within a private network, the security of this design is immediately in jeopardy when any VPN key is compromised, or when an attacker gains access behind the perimeter defense. Can you imagine an attacker getting into your VPN and connecting to any robot's ROS Master? We also generally see non-encrypted traffic between services and robots, such as APIs running on http and not https.

Notwithstanding the security concerns (in some cases, they are a powerful tool to debug and access your robot fleet), VPNs are hard to manage. You need to configure rules governing key expiration and key rotation. VPN logs need to be made available for audits (to check for anomalous access).



End-to-end

End-to-end

At Formant we take a modern, cloud-native approach to security — an approach that provides end-to-end encryption and identity across all robots, networks, operators and APIs. The diagram below depicts the Formant security model.

The Formant security model is built on cloud native principles such as zero-trust authentication and and scoped permissions for devices, users, and services. We also follow best practices for edge deployed software (the Formant Agent) and our cloud web application and backend APIs. Security is baked into our architecture and peer review processes as well, including regular reviews and tests of the infrastructure, web application, and API layers.

Speak to an expert formant.io/get-started

Formant

Provisioning tokens

Starting with the robot, the Formant Agent uses a one-time, short-lived provisioning token to generate the Robot Private Key. This private key is never sent to the cloud. The corresponding Robot Public Key is used to provision the robot and is stored on our backend to verify signed requests from robots.

Load balancers, WAFs, and public vs. private subnets

Our first layer of security is our public-facing load balancer. Attached to the load balancer is a Web Application Firewall that provides protection against DDOS, blacklists bad IPs, and enforces other security based rules. We also run our entire API layer in private subnets ensuring no external access to any of our cloud compute instances or databases.

Enforcing identity of robots, users, and services

At the application layer, we authenticate and authorize every API call with the client's robot, user, or service identity. This design stands in contrast to an "API token" pattern that supports anonymous clients. The identity associated with every API call in the Formant system allows us to support fine-grained access control, permission revocation, and auditing. For additional protection, we ensure that robot and user private keys are never sent on the wire. Service tokens within our backend rely on symmetric encryption and rotate frequently.

An important implementation detail is how a robot, user, or service proves their identity. For each of these API calls we leverage JSON Web Tokens to prove the identity of the caller. These are short-lived tokens which are industry standard in modern cloud applications. It allows us to never need to send a permanent private key across the wire.

TLS end-to-end

Many applications deployed to public clouds will be happy enough with TLS termination at the load balancer and HTTP (plain text) traffic behind it. Formant's cloud employs mutual TLS to ensure encrypted traffic everywhere — between the ingress controller and APIs, as well as between APIs.

We leverage several technologies to achieve this. First, our APIs are running within a Kubernetes cluster, which allows us to deploy Istio, an open, platform-independent service mesh that provides traffic management, policy enforcement, and telemetry collection. Within this service mesh, we enable mutual TLS, a feature of Istio that allows us to encrypt all traffic between the ingress controller and APIs, as well as between APIs. The service mesh also allows us to employ a second layer of defense against misbehaving applications or bad actors with layer 7 (application networking layer) policy rules.

Customizations for LANs: encryption via gRPC

For distributed on-site LANs where the Formant Agent may be running on a separate host we support an advanced feature to encrypt gRPC traffic between a Robot Application and the Formant Agent. In most cases we see the Robot Application running on the same host as the Formant Agent in which case plain text traffic is acceptable, however we do encourage our security-conscious customers to consider using this feature.

Ask an expert

Designing for security is a key tenet in our engineering process. Something we've often observed is treating security as a "bolt-on" feature. Many times these efforts end in application security designs that leave security holes (i.e. crunchy outer shell design or permanent API keys). If you are thinking about improving your robot security model or thinking of an audit, we can help. Please get in touch with an expert on our services team to get started.

Speak to an expert formant.io/get-started