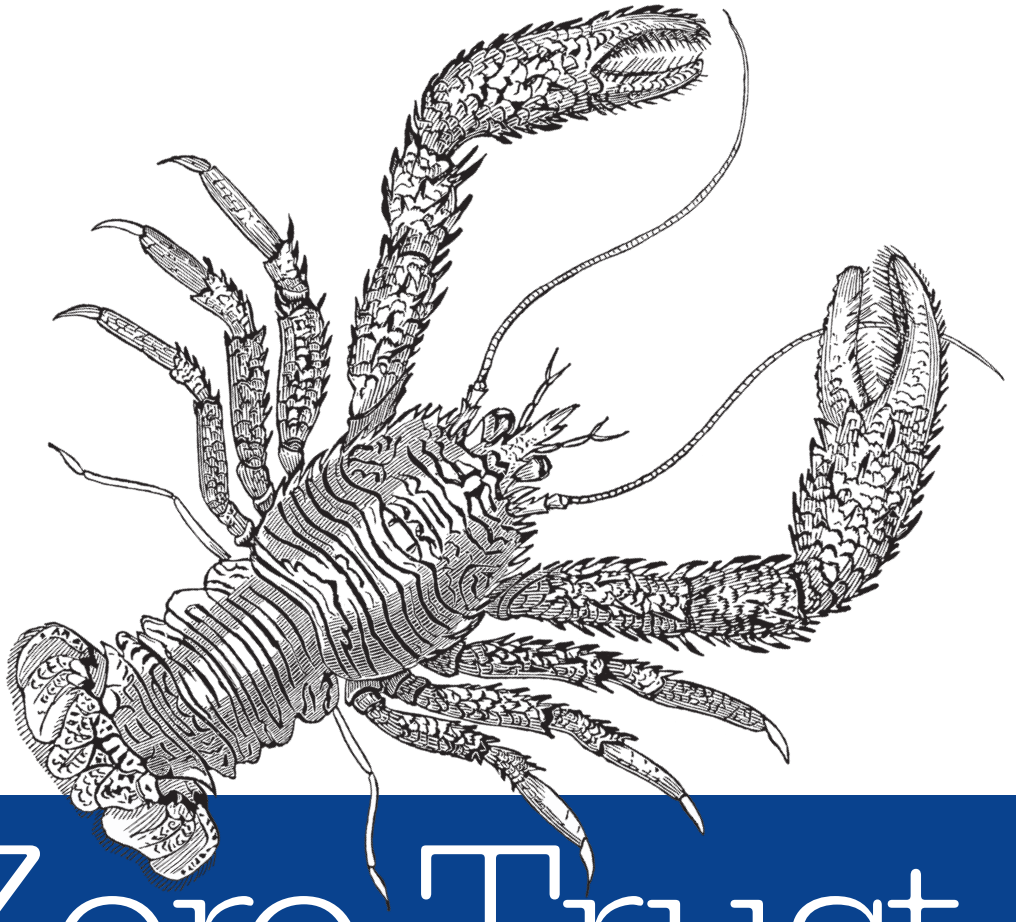
O'REILLY®



Zero Trust Networks

BUILDING SECURE SYSTEMS IN UNTRUSTED NETWORKS

Evan Gilman & Doug Barth

CHAPTER 2
Managing Trust

Trust management is perhaps the most important component of a zero trust network. We are all familiar with trust to some degree—you probably trust members of your family, but not a stranger on the street, and certainly not a stranger who looks threatening or menacing. Why is that?

For starters, you actually *know* your family members. You know what they look like, where they live; perhaps you've even known them your whole life. There is no question of who they are, and you are more likely to trust them with important matters than others.

A stranger, on the other hand, is someone completely unknown. You might see their face, and be able to tell some basic things about them, but you don't know where they live, and you don't know their history. They might appear perfectly cromulent, but you likely wouldn't rely on one for important matters. Watch your stuff for you while you run to the bathroom? Sure. Make a quick run to the ATM for you? Definitely not.

At the end, you are simply taking in all the information you can tell about the situation, a person, and all you may know about them, and deciding how trustworthy they are. The ATM errand requires a very high level of trust, where watching your stuff needs much less, but not zero.

You may not even trust yourself completely, but you can definitely trust that actions taken by you were taken by you. In this way, trust in a zero trust network always originates with the operator. Trust in a zero trust network seems contradictory, though it is important to understand that when you have no *inherent* trust, you must source it from somewhere and manage it carefully.

There's a small wrinkle though: the operator won't always be available to authorize and grant trust! Plus, the operator just doesn't scale :). Luckily, we know how to solve that problem—we delegate trust as shown in Figure 2-1.

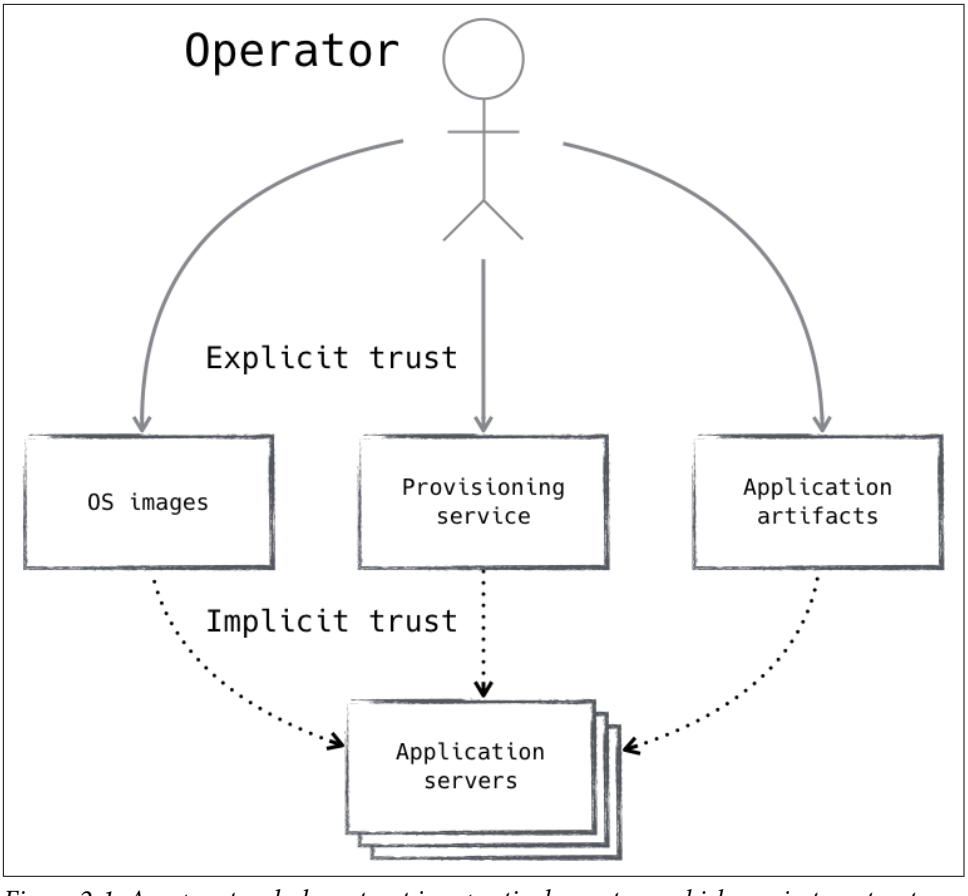


Figure 2-1. An operator declares trust in a particular system, which can in turn trust another, forming a trust chain

Trust delegation is important because it allows us to build automated systems that can grow to large scale and to operate in a secure and trusted way with minimal human intervention. The trusted operator must assign some level of trust to a system, enabling it to take actions on behalf of the operator. A simple example of this is autoscaling. You want your servers to provision themselves as needed, but how do you know a new server is one of yours and not some other random server? The operator must delegate the responsibility to a provisioning system, granting it the ability to assign trust to, and create, new hosts. In this way, we can say that we trust the new server is indeed our own, because the provisioning system has validated that it has taken the action to create it, and the provisioning system can prove that the operator has granted it the ability to do so. This flow of trust back to the operator is often referred to as a *trust chain*, and the operator can be referred to as a trust anchor.

Threat Models

Defining threat models is an important first step when designing a security architecture. A *threat model* enumerates the potential attackers, their capabilities and resources, and their intended targets. Threat models will normally define which attackers are in scope, rationally choosing to mitigate attacks from weaker adversaries before moving onto more difficult adversaries.

A well-defined threat model can be a useful tool to focus security mitigation efforts. When building security systems, like most engineering exercises, there is a tendency to focus on the fancier aspects of the engineering problem to the detriment of the more boring but still important parts. This tendency is especially worrisome in a security system, since the weakest link in the system is where attackers will quickly focus their attention. Therefore, the threat model serves as a mechanism for focusing our attention on a single threat and fully mitigating their attacks.

Threat models can also be useful when prioritizing security initiatives. Fighting state-level actors is pointless if a system's security measures are insufficient to defend against a simple brute force attack on a user's poor password. As such, it is important to start first with simpler personas when building a threat model.

Common Threat Models

There are many different techniques for threat modeling in the security field. Here are some of the more popular ones:

- STRIDE
- DREAD
- PASTA
- Trike
- VAST

The varying threat modeling techniques provide different frameworks for exploring the threat space. Each of them is after the same goal: to enumerate threats to the system and further enumerate the mitigating systems and processes for those threats.

Different threat models approach the problem from different angles. Some modeling systems might focus on the assets that an attacker would be targeting. Others might look at each software component in isolation and enumerate all the attacks that could be applied to that system. Finally, some models might look at the system as a whole from the attacker's perspective: as an attacker, how might I approach penetrating this system. Each of these approaches has pros and cons. For a well-diversified mitigating strategy, a blend of the three approaches is ideal.

If we were to look at the attacker-based threat modeling methodology, we are able to categorize attackers into a list of increasing capabilities (ordered from least to most threatening):

1. *Opportunistic attackers*
   So-called script kiddies, who are unsophisticated attackers taking advantage of well-known vulnerabilities with no predetermined target.

2. *Targeted attackers*
   Attackers who craft specialized attacks against a particular target. Spear phishing and corporate espionage might fall under this bucket.

3. *Insider threats*
   A credentialed but everyday user of a system. Contractors and unprivileged employees generally fall into this bucket.

4. *Trusted insider*
   A highly trusted administrator of a system.

5. *State-level actor*
   Attackers backed by foreign or domestic governments and assumed to have vast resources and positioning capabilities to attack a target.

Categorizing threats like this is a useful exercise to focus discussion around a particular level to mitigate against. We will discuss which level zero trust targets in the next section.

Zero Trust's Threat Model

In RFC 3552, the Internet Threat Model is described. Zero trust networks generally follow the Internet Threat Model to plan their security stance. While reading the entire RFC is recommended, here is a relevant excerpt:

> The Internet environment has a fairly well understood threat model. In general, we assume that the end-systems engaging in a protocol exchange have not themselves been compromised. Protecting against an attack when one of the end-systems has been compromised is extraordinarily difficult. It is, however, possible to design protocols which minimize the extent of the damage done under these circumstances.
>
> By contrast, we assume that the attacker has nearly complete control of the communications channel over which the end-systems communicate. This means that the attacker can read any PDU (Protocol Data Unit) on the network and undetectably remove, change, or inject forged packets onto the wire. This includes being able to generate packets that appear to be from a trusted machine. Thus, even if the end-system with which you wish to communicate is itself secure, the Internet environment provides no assurance that packets which claim to be from that system in fact are.

Zero trust networks, as a result of their control over endpoints in the network, expand upon the Internet Threat Model to consider compromises at the endpoints.

The response to these threats is generally to first harden the systems proactively against compromised peers, and then facilitate detection of those compromises. Detection is aided by scanning of devices and behavioral analysis of the activity from each device. Additionally, mitigation of endpoint compromise is achieved by frequent upgrades to software on devices, frequent and automated credential rotation, and in some cases frequent rotation of the devices themselves.

An attacker with unlimited resources is essentially impossible to defend against, and zero trust networks recognize that. The goal of a zero trust network isn't to defend against all adversaries, but rather the types of adversaries that are commonly seen in a hostile network.

From our earlier discussion of attacker capabilities, a zero trust network is generally attempting to mitigate attacks up to and including attacks originating from a "trusted insider" level of access. Most organizations do not experience attacks that exceed this level of sophistication. Developing mitigations against these attackers will defend against the vast majority of compromises and would be a dramatic improvement for the industry's security stance.

Zero trust networks generally do not try to mitigate all state-level actors, though they do attempt to mitigate those attempting to compromise their systems remotely. State-level actors are assumed to have vast amounts of money, so many attacks that would be infeasible for lesser organizations are available to them. Additionally, local governments have physical and legal access to many of the systems that organizations depend upon for securing their networks.

Defending against these localized threats is exceedingly expensive, requiring dedicated physical hardware, and most zero trust networks consider the more extreme forms of attacks (say a vulnerability being inserted into a hypervisor which copies memory pages out of a VM) out of scope in their threat models. We should be clear that while security best practices are still very much encouraged, the zero trust model only requires the safety of information used to authenticate and authorize actions, such as on-disk credentials. Further requirements on endpoints, say full disk encryption, can be applied via additional policy.

Strong Authentication

Knowing how much to trust someone is useless without being able to associate a real-life person with that identity you know to trust. Humans have many senses to determine if the person in front of them is who they think they are. Turns out, combinations of senses are hard to fool.

Computer systems, however, are not so lucky. It's more like talking to someone on the phone. You can listen to their voice, read their caller ID, ask them questions...but you can't see them. Thus we are left with a challenge: how can one be reasonably assured

that the person (or system) on the other end of the line is in fact who they say they are?

Typically, operators examine the IP address of the remote system and ask for a password. Unfortunately, these methods alone are insufficient for a zero trust network, where attackers can communicate from any IP they please and insert themselves between yourself and trusted remote host. Therefore, it is very important to employ strong authentication on every flow in a zero trust network.

The most widely accepted method to accomplish this is a standard named X.509, which most engineers are familiar with. It defines a certificate standard that allows identity to be verified through a chain of trust. It's popularly deployed as the primary mechanism for authenticating TLS (formerly SSL) connections.

SSL is Anonymous

The most widely consumed TLS configuration validates that the client is speaking to a trusted resource, but not that the resource is speaking to a trusted client. This poses an obvious problem for zero trust networks.

TLS additionally supports mutual authentication, in which the resource also validates the client. This is an important step in securing private resources; otherwise, the client device will go unauthenticated. More on zero trust TLS configuration in "Mutually Authenticated TLS" on page 155.

Certificates utilize two cryptographic keys: a *public key* and a *private key*. The public key is distributed, and the private key is held as a secret. The public key can encrypt data that the private key can decrypt, and vice versa, as shown in Figure 2-2. This allows one to prove they are in the presence of the private key by correctly decrypting a piece of data that was encrypted by the well-known (and verifiable) public key. In this way, identity can be validated without ever exposing the secret.

Certificate-based authentication lets us be certain that the person on the other end of the line has the private key, and also lets us be certain that someone listening in can't steal the key and reuse it in the future. It does, however, still rely on a secret, something that can be stolen. Not necessarily by listening in, but perhaps by a malware infection or physical theft.

So while we can validate that credentials are legitimate, we might not trust that they have been kept a secret. For this reason, it is desirable to use multiple secrets, stored in different places, which in combination grant access. With this approach, a potential attacker must steal multiple components.

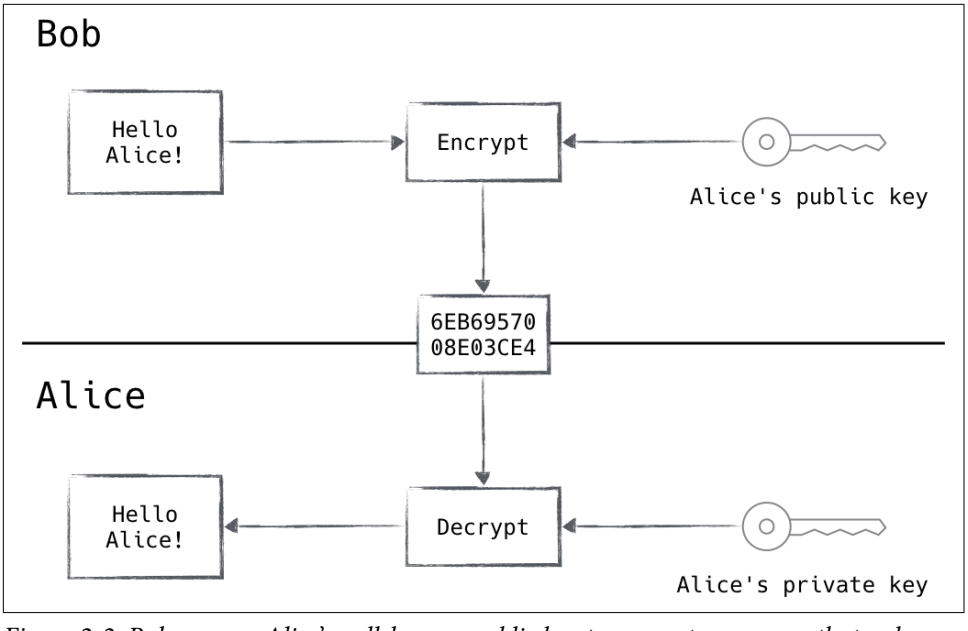


Figure 2-2. Bob can use Alice's well-known public key to encrypt a message that only Alice is able to decrypt

While having multiple components goes a long way in preventing unauthorized access, it is still conceivable that all these components can be stolen. Therefore, it is critical that all authentication credentials be time-boxed. Setting an expiration on credentials helps to minimize the blast radius of leaked or stolen keys and gives the operator an opportunity to reassert trust. The act of changing, or renewing, keys/passwords is known as *credential rotation*.

Credential rotation is essential for validating that no secrets have been stolen, and revoking them when required. Systems utilizing keys/passwords that are hard or impossible to rotate should be avoided at all cost, and when building new systems this fact should be taken into account early on in the design process. The rotation frequency of a particular credential is often inversely proportional to the cost of rotation.

Examples of Secrets Expensive to Rotate

- Certificates requiring external coordination
- Hand-configured service accounts
- Database passwords requiring downtime to reset
- A site-specific salt that cannot be changed without invalidating all stored hashes

Authenticating Trust

We spoke a little bit about certificates and public key cryptography. However, certificates alone don't solve the authentication issue. For instance, you can be assured that a remote entity is in possession of a private key by making an assertion using its public key. But how do you obtain the public key to begin with? Sure, public keys don't need to be secret, but you must still have a way to know that you have the *right* public key. *Public key infrastructure*, or PKI, defines a set of roles and responsibilities that are used to securely distribute and validate public keys in untrusted networks.

The goal of a PKI is to allow unprivileged participants to validate the authenticity of their peers through an existing trust relationship with a mutual third party. A PKI leverages what is known as a *registration authority* (RA) in order to bind an identity to a public key. This binding is embedded in the certificate, which is cryptographically signed by the trusted third party. The signed certificate can then be presented in order to "prove" identity, so long as the recipient trusts the same third party.

There are many types of PKI providers. The most popular two are *certificate authorities* (CAs) and *webs of trust* (WoTs). The former relies on a signature chain that is ultimately rooted in the mutually trusted party. The latter allows systems to assert validity of their peers, forming a web of endorsements rather than a chain. Trust is then asserted by traversing the web until a trusted certificate is found. While this approach is in relatively wide use with Pretty Good Privacy (PGP) encryption, this book will focus on PKIs that employ a CA, the popularity of which overshadows the WoT provider.

What Is a Certificate Authority?

Certificate authorities act as the trust anchor of a certificate chain. They sign and publish public keys and their bound identities, allowing unprivileged entities to assert the validity of the binding through the signature.

CA certificates are used to represent the identity of the CA itself, and it is the private key of the CA certificate that is used to sign client certificates. The CA certificate is well known, and is used by the authenticating entity to validate the signature of the presented client certificate. It is here that the trusted third-party relationship exists, issuing and asserting the validity of digital certificates on behalf of the clients.

The trusted third-party position is very privileged. The CA must be protected at all costs, since its subversion would be catastrophic. Digital certificate standards like X.509 allow for chaining of certificates, which enables the root CA to be kept offline. This is considered standard practice in CA-based PKI security. We'll talk more about X.509 security in Chapter 5.

Importance of PKI in Zero Trust

All zero trust networks rely on PKI to prove identity throughout the network. As such, it acts as the bedrock of identity authentication for the majority of operations. Entities that might be authenticated with a digital certificate include:

- Devices
- Users
- Applications

Binding Keys to Entites

PKI can bind an identity to a public key, but what about a private key to the entity it is meant to identify? After all, it is the private key which we are really authenticating. It is important to keep the private key as close to the entity it was meant to represent as possible. The method by which this is done varies by the type of entity. For instance, a user might store a private key on a smart card in their pocket, where a device might store a private key in an onboard security chip. We'll discuss which methods best fit which entities in Chapters 5, 6, and 7.

Given the sheer number of certificates that a zero trust network will issue, it is important to recognize the need for automation. If humans are required in order to process certificate signing requests, the procedure will be applied sparingly, weakening the overall system. That being said, certificates deemed highly sensitive will likely wish to retain a human-based approval process.

Private Versus Public PKI

PKI is perhaps most popularly deployed as a public trust system, backing X.509 certificates in use on the public internet. In this mode, the trusted third party is publicly trusted, allowing clients to authenticate resources that belong to other organizations. While public PKI is trusted by the internet at large, it is not recommended for use in a zero trust network.

Some might wonder why this is. After all, public PKI has some defensible strengths. Factors like existing utilities/tooling, peer-reviewed security practices, and the promise of a better time to market are all attractive. There are, however, several drawbacks to public PKI that work against it. The first is cost.

The public PKI system relies on publicly trusted authorities to validate digital certificates. These authorities are businesses of their own, and usually charge a fee for signing certificates. Since a zero trust network has many certificates, the signing costs

associated with public authorities can be prohibitive, especially when considering rotation policies.

Another significant drawback to public PKI is the fact that it's hard to *fully* trust the public authorities. There are lots of publicly trusted CAs, operating in many countries. In a zero trust network leveraging public PKI, any one of these CAs can cut certificates that your network trusts. Do you trust the laws and the governments associated with all of those CAs too? Probably not. While there are some mitigation methods here, like certificate pinning or installing trust in a single public CA, it remains challenging to retain trust in a disjoint organization.

Finally, flexibility and programmability can suffer when leveraging public CAs. Public CAs are generally interested in retaining the public's trust, so they do employ good security measures. This might include policies about how certificates are formed, and what information can be placed where. This can adversely affect zero trust authentication in that it is often desirable to store site-specific metadata in the certificate, like a role or a user ID. Additionally, not all public CAs provide programmable interfaces, making automation a challenge.

Public PKI Strictly Better Than None

While the drawbacks associated with public PKI are significant, and the authors heavily discourage its use within a zero trust network, it remains superior to no PKI at all. A well-automated PKI is the first step, and work will be required in this area no matter which PKI approach you choose. The good news is that if you choose to leverage public PKI initially, there is a clear path to switch to private PKI once the risk becomes too great. It begs the question, however, if it is even worth the effort, since automation of those resources will still be required.

Least Privilege

The principle of least privilege is the idea that an entity should be granted only the privileges it needs to get its work done. By granting only the permissions that are always required, as opposed to sometimes desired, the potential for abuse or misuse by a user or application is greatly reduced.

In the case of an application, that usually means running it under a service account, in a container or jail, etc. In the case of a human, it commonly manifests itself as policies like "only engineers are allowed access to the source code." Devices must also be considered in this regard, though they often assume the same policies as the user or application they were originally assigned to.

Privacy as Least Privilege

The application of encryption in the name of privacy is an often-overlooked application of least privilege. Who *really* needs access to the packet payload?

Another effect of this principle is that if you *do* need elevated access, that you retain those access privileges for only as long as you need them. It is important to understand what actions require which privileges so that they may be granted only when appropriate. This goes one step beyond simple access control reviews.

This means that human users should spend most of their time executing actions using a nonprivileged user account. When elevated privileges are needed, the user needs to execute those actions under a separate account with higher privileges.

On a single machine, elevating one's privileges is usually accomplished by taking an action that requires the user to authenticate themselves. For example, on a Unix system, invoking a command using the `sudo` command will prompt the user to enter their password before running that command as a different role. In GUI environments, a dialog box might appear requiring the user's password before performing the risky operation. By requiring interaction with the user, the potential for malicious software to take action on behalf of the user is (potentially) mitigated.

In a zero trust network, users should similarly operate in a reduced privilege mode on the network most of the time, only elevating their permissions when needed to perform some sensitive operation. For example, an authenticated user might freely access the company's directory or interact with project planning software. Accessing a critical production system, however, should require additional confirmation that the user or the user's system is not compromised. For relatively low-risk actions, this privilege elevation could be as simple as reprompting for the user's password, requesting a second factor token, or sending a push notification to the user's phone. For high-risk access, one might choose to require active confirmation from a peer via an out-of-band request.

Human-Driven Authentication

For particularly sensitive operations, an operator may rely on the coordination of multiple humans, requiring a number of people to be actively engaged in order to authenticate a particular action. Forcing authentication actions into the real world is a good way to ensure a compromised system can't interfere with them. Be careful, however—these methods are expensive and will become ineffective if employed too frequently.

Like users, applications should also be configured to have the fewest privileges necessary to operate on the network. Sadly, applications deployed in a corporate setting are often given fairly wide access on the network. Either due to the difficulty of defining policies to rein in applications, or the assumption that compromised users are the more likely target, it's now become commonplace for the first step in setting up a machine to be disabling the application security frameworks that are meant to secure the infrastructure.

Beyond the traditional consideration of privilege for users and applications, zero trust networks also consider the privilege of the device on the network. It is the combination of user or application and the device being used that determines the privilege level granted. By joining the privilege of a user to the device being used to access a resource, zero trust networks are able to mitigate the effects of lost or compromised credentials. Chapter 3 will explore how this marriage of devices and users works in practice.

Privilege in a zero trust network is more dynamic than in traditional networks. Traditional networks eventually converge on policies that stay relatively static. If new use cases appear that require greater privilege, either the requestor must lobby for a change in policy; or, perhaps more frequently, they ask someone with greater privilege (a sysadmin, for example) to perform the operation for them. This static definition of policy presents two problems. First, in more permissive organizations, privilege will grow over time, lessening the benefit of least privilege. Second, in both permissive and restrictive organizations, admins are given greater access, which has resulted in malicious actors purposefully targeting sysadmins for phishing attacks.

A zero trust network, by contrast, will use many attributes of activity on the network to determine a riskiness factor for the access being requested currently. These attributes could be temporal (access outside of the normal window activity for that user is more suspicious), geographical (access from a different location than the user was last seen), or even behavioral (access to resources the user does not normally access). By considering all the details of an access attempt, the determination of whether the action is authorized or not can be more granular than a simple binary answer. For example, access to a database by a given user from their normal location during typical working hours would be granted, but access from a new location at different working hours might require the user to authenticate using an additional factor.

The ability to actively adjust access based on the riskiness of activity on a network is one of the several features that make zero trust networks more secure. By dynamically adjusting policies and access, these networks are able to respond autonomously to known and unknown attacks by malicious actors.

Variable Trust

Managing trust is perhaps the most difficult aspect of running a secure network. Choosing which privileges people and devices are allowed on the network is time consuming, constantly changing, and directly affects the security posture the network presents. Given the importance of trust management, it's surprising how underdeployed network trust management systems are today.

Defining trust policies is typically left as a manual effort for security engineers. Cloud systems might have managed policies, but those policies provide only basic isolation (e.g., super user, admin, regular user) which advanced users typically outgrow. Perhaps in part due to the difficulty of defining and maintaining them, requests to change existing policies can be met with resistance. Determining the impact of a policy change can be difficult, so prudence pushes the administrators toward the status quo, which can frustrate end users and overwhelm system administrators with change requests.

Policy assignment is also typically a manual effort. Users are granted policies based on their responsibilities in the organization. This role-based policy system tends to produce large pools of trust in the administrators of the network, weakening the overall security posture of the network. These pools of trust have created a market for hackers to "hunt sys admins", seeking out and compromising system administrators. Perhaps the gold standard for a secure network is one without highly privileged system administrators.

These pools of trust underscore the fundamental issue with how trust is managed in traditional networks: policies are not nearly dynamic enough to respond to the threats being leveled against the network. Mature organizations will have some sort of auditing process in place for activity on their network, but audits can be done too infrequently, and are frankly so tedious that doing them well is difficult for humans. How much damage could a rogue sysadmin do on a network before an audit discovered their behavior and mitigated it? A more fruitful path might be to rethink the actor/trust relationship, recognizing that trust in a network is ever evolving and based on the previous and current actions of an actor within the network.

This model of trust, considering all the actions of an actor and determining their trustworthiness, is not novel. Credit agencies have been performing this service for many years. Instead of requiring organizations like retailers, financial institutions, or even an employer to independently define and determine one's trustworthiness, a credit agency can use actions in the real world to score and gauge the trustworthiness of an individual. The consuming organizations can then use their credit score to decide how much trust to grant that person. In the case of a mortgage application, an individual with a higher credit score will receive a better interest rate, which mitigates the risk to the lender. In the case of an employer, one's credit score might be used as a

signal for a hiring decision. On a case-by-case basis, these factors can feel arbitrary and opaque, but they serve a useful purpose; providing a mechanism for defending a system against arbitrary threats by defining policy based not only on specifics, but also on an ever-changing and evolving score.

A zero trust network utilizes this insight to define trust within the network, as shown in Figure 2-3. Instead of defining binary policy decisions assigned to specific actors in the network, a zero trust network will continuously monitor the actions of an actor on the network to update their trust score. This score can then be used to define policy in the network based on the severity of breach of that trust (Figure 2-4). A user viewing their calendar from an untrusted network might require a relatively low trust score. However, if that same user attempted to change system settings, they would require a much higher score and would be denied or flagged for immediate review. Even in this simple example, one can see the benefit of a score: we can make fine-grained determinations on the checks and balances needed to ensure trust is maintained.

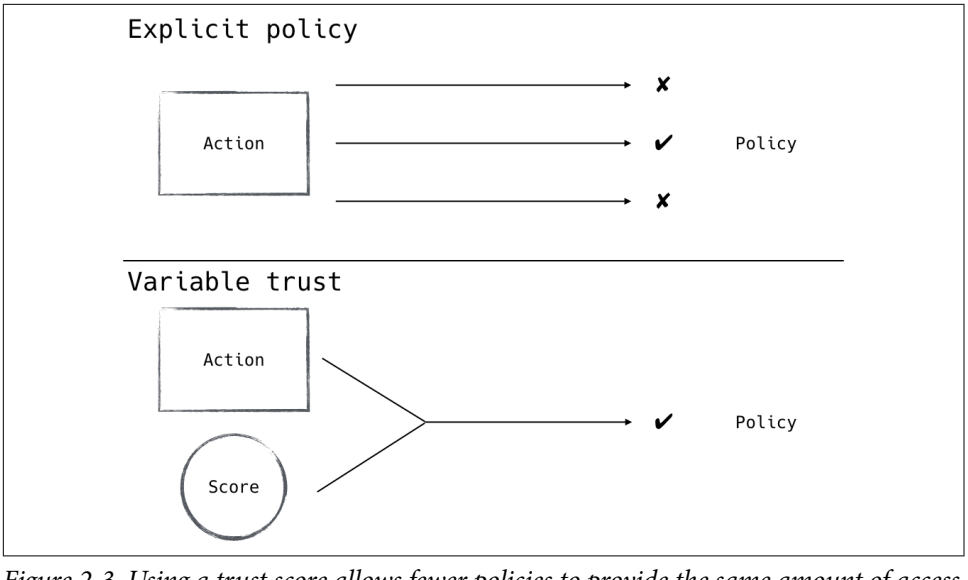


Figure 2-3. Using a trust score allows fewer policies to provide the same amount of access

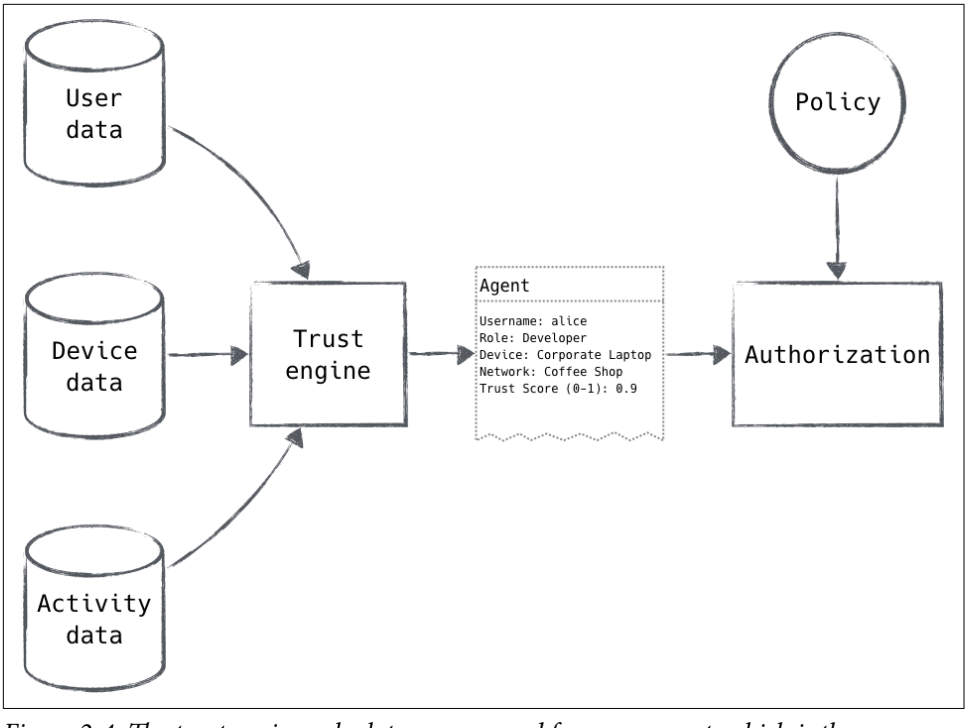


Figure 2-4. The trust engine calculates a score and forms an agent, which is then compared against policy in order to authorize a request. We'll talk more about agents in Chapter 3.

Monitoring Encrypted Traffic

Since practically all flows in a zero trust network are encrypted, traditional traffic inspection methods don't work as well as intended. Instead, we are limited to inspecting what we can see, which in most cases is the IP header and perhaps the next protocol header (like TCP in the case of TLS). If a load balancer or proxy is in the request path, however, there is an opportunity for deeper inspection and authorization, since the application data will be exposed for examination.

Clients begin sessions as untrusted. They must accumulate trust through various mechanisms, eventually accruing enough to gain access to the service they're requesting. Strong authentication proving that a device is company-owned, for instance, might accumulate a good bit of trust, but not enough to allow access to the billing system. Providing the correct RSA token might give you a good bit more trust, enough to access the billing system when combined with the trust inferred from successful device authentication.

Strong Policy as a Trust Booster

Things like score-based policies, which can affect the outcome of an authorization request based on a number of variables like historical activity, drastically improve a network's security stance when compared to static policy. Sessions that have been approved by these mechanisms can be trusted more than those that haven't. In turn, we can rely (a little bit) less on user-based authentication methods to accrue the trust necessary to access a resource, improving the overall user experience.

Switching to a trust score model for policies isn't without its downsides. The first hurdle is whether a single score is sufficient for securing all sensitive resources. In a system where a trust score can decrease based on user activity, a user's score can also increase based on a history of trustworthy activity. Could it be possible for a persistent attacker to slowly build their credibility in a system to gain more access?

Perhaps slowing an attacker's progress by requiring an extended period of "normal" behavior would be sufficient to mitigate that concern, given that an external audit would have more opportunity to discover the intruder. Another way to mitigate that concern is to expose multiple pieces of information to the control plane so that sensitive operations can require access from trusted locations and persons. Binding a trust score to device and application metadata allows for flexible policies that can declare absolute requirements yet still capture the unknown unknowns through the computed trust score.

Loosening the coupling between security policy and a user's organizational role can cause confusion and frustration for end users. How can the system communicate to users that they are denied access to some sensitive resource from a coffee shop, but not from their home network? Perhaps we present them with increasingly rigorous authentication requirements? Should new members be required to live with lower access for a time before their score indicates that they can be trusted with higher access? Maybe we can accrue additional trust by having the user visit a technical support office with the device in question. All of these are important points to consider. The route one takes will vary from deployment to deployment.

Control Plane Versus Data Plane

The distinction between the control plane versus the data plane is a concept that is commonly referenced in network systems. The basic idea is that a network device has two logical domains with a clear interface between those domains. The data plane is the relatively dumb layer that manages traffic on the network. Since that layer is handling high rates of traffic, its logic is kept simple and often pushed to specialized hardware. The control plane, conversely, could be considered the brains of the

network device. It is the layer that system administrators apply configuration to, and as a result is more frequently changed as policy evolves.

Since the control plane is so malleable, it is unable to handle the high rate of traffic on the network. Therefore, the interface between the control plane and the data plane needs to be defined in such a way that nearly any policy behavior can be implemented at the data layer with infrequent requests being made to the control plane (relative to the rate of traffic).

A zero trust network also defines a clear separation between the control plane and data plane. The data plane in such a network is made up of the applications, firewalls, proxies, and routers that directly process all traffic on the network. These systems, being in the path of all connections, need to quickly make a determination of whether traffic should be allowed. When viewing the data plane as a whole, it has broad access and exposure throughout the system, so it is important that the services on the data plane cannot be used to gain privilege in the control plane and thereby move laterally within the network. We'll discuss control plane security in Chapter 4.

The control plane in a zero trust network is made up of components that receive and process requests from data plane devices that wish to access (or grant access to) network resources, as shown in Figure 2-5. These components will inspect data about the requesting system to make a determination on how risky the action is, and examine relevant policy to determine how much trust is required. Once a determination is made, the data plane systems are signaled or reconfigured to grant the requested access.

The mechanism by which the control plane affects change in the data plane is of critical importance. Since the data plane systems are often the entry point for attackers into a network, the interface between it and the control plane must be clear, helping to ensure that it cannot be subverted to move laterally within the network. Requests between the data plane and control plane systems must be encrypted and authenticated using a non-public PKI system to ensure that the receiving system is trustworthy. The control/data plane interface should resemble the user/kernel space interface, where interactions between those two systems are heavily isolated to prevent privilege escalation.

This concern with the interface between the control plane and the data plane belies another fundamental property of the control plane: the control plane is the trust grantor for the entire network. Due to its far-reaching control of the network's behavior, the control plane's trustworthiness is critical. This need to have an actor on the network with a highly privileged role presents a number of interesting design requirements.

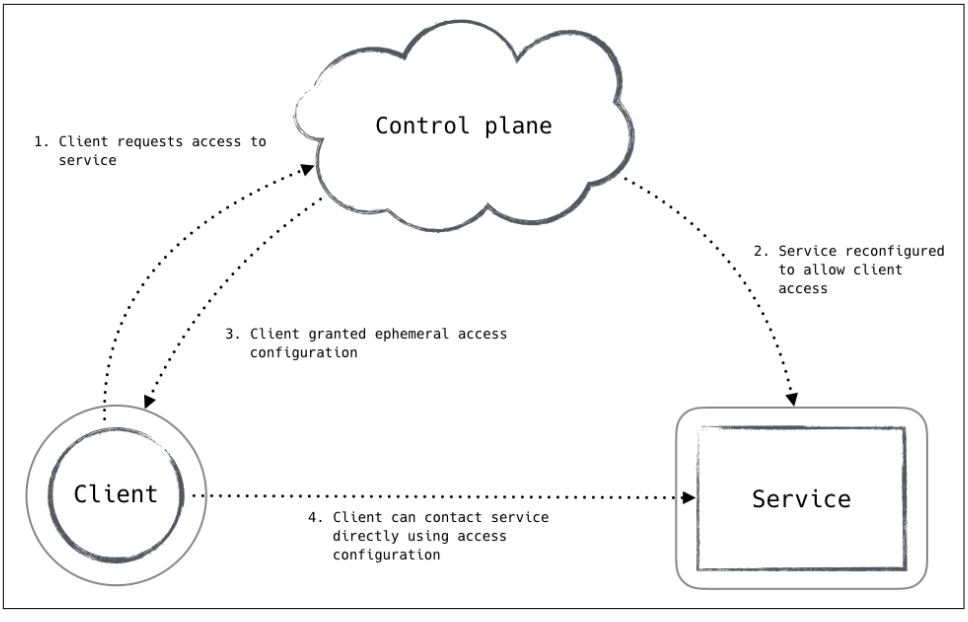


Figure 2-5. A zero trust client interacting with the control plane in order to access a resource

The first requirement is that the trust granted by the control plane to another actor in the data plane should have limited real-time value. Trust should be temporary, requiring regular check-ins between the truster and trustee to ensure that the continued trust is reasonable. When implementing this tenet, leased access tokens or short lifetime certificates are the most appropriate solution. These leased access tokens should be validated not just within the data plane (e.g., when the control plane grants a token to an agent to move through the data plane), but also between the interaction between the data plane and the control plane. By limiting the window during which the data plane and control plane can interact with a particular set of credentials, the possibility for physical attacks against the network is mitigated.

Summary

This chapter discussed the critical systems and concepts that are needed to manage trust in a zero trust network. Many of these ideas are common in traditional network security architectures, but it is important to lay the foundation of how trust is managed in a network without any.

Trust originates from humans and flows into other systems via trust mechanisms that a computer can operate against. This approach makes logical sense: a system can't be considered trusted unless the humans who use it feel confident that it is faithfully executing their wishes.

Security has frequently been viewed as a set of best practices, which are passed down from one generation of engineers to the next. Breaking out of this cycle is important, since each system is unique, and so we discussed the idea of threat models. Threat models attempt to define the security posture of a system by enumerating the threats against the system and then defining the mitigating systems and processes which anticipate those threats. While a zero trust network assumes a hostile environment, it is still fundamentally grounded in the threat model, which makes sense for the system. We enumerated several present-day threat-modeling techniques so that readers can dig deeper. We also discussed how the zero trust model is based on the internet threat model and expands its scope to the endpoints that are under the control of zero trust system administrators.

Having trust in a system requires the use of strong authentication throughout the system. We discussed the importance of this type of authentication in a zero trust network. We also briefly talked a bit about how strong authentication can be achieved in today's technology. We will discuss these concepts more in later chapters.

In order to effectively manage trust in a network, you must be able to positively identify trusted information, particularly in the case of authentication and identity. Public key infrastructure (or PKI) provides the best methods we have today for asserting validity and trust in a presented identity. We discussed why PKI is important in a zero trust network, the role of a certificate authority, and why private PKI is preferred over public PKI.

Least privilege is one of the key ideas in these types of networks. Instead of constructing a supposedly safe network over which applications can freely communicate, the zero trust model assumes that the network is untrustworthy, and as a result, components on the network should have minimal privileges when communicating. We explained what the concept of least privilege is and how it is similar and different than least privilege in standalone systems.

One of the most exciting ideas of zero trust networks is the idea of variable trust. Network policy has traditionally focused on which systems are allowed to communicate in what manner. This binary policy framework results in policy that is either too rigidly defined (creating human toil to continually adjust) or too loosely defined (resulting in security systems that assert very little). Additionally, policy that is defined based on concrete details of interactions will invariably be stuck in a cat-and-mouse game of adjusting policy based on past threats. The zero trust model leans on the idea of variable trust, a numeric value representing the level of trust in a component. Policy can then be written against this number, effectively capturing a number of conditions without complicating the policy with edge cases. By defining policy in less concrete details, and considering the trust score while making an authorization decision, the authorization systems are able to adjust to novel threats.

Zero trust networks make a clear distinction between the control plane systems and the data plane systems. We discussed at a high level how these two systems interact with each other to allow expected communication to flow through the network. In later chapters we will flesh out more of the control and data plane systems that manage communication in the network.

The next chapter digs into a fundamental entity in zero trust networks that is used to authorize actions on the network.