IRONCORE LABS

SaaS Trust Models for Security and Data Privacy

You can't compare the security and privacy of SaaS offerings until you understand trust models.

INTRODUCTION TO SAAS TRUST MODELS

Zero-trust is the ultimate goal in privacy, but it's rarely an option. The good news is that it doesn't have to be either zero-trust or full-trust. There's a sliding scale of trust models in the middle that SaaS buyers should understand. Keep reading to see why these models are important.

Trust powers business. Consumers and businesses buy from brands they trust. But in the world of software-as-a-service, trust is about far more than brand reputation.

Our natural inclination is to trust the brands that our peers trust. This is why large companies are able to keep selling software even when their products stagnate. But even large companies need to watch out for the trapdoor at their feet and the thing that can turn their biggest advocates into their loudest critics: privacy.

Zoom recently discovered this the hard way, when users learned that their security and privacy expectations weren't being met.

> *Privacy is becoming a reason for consumers to purchase a product, in the same way that "organic," "free trade" and "cruelty-free" labels have driven products sales in the past decade.*
>
> *— Gartner*

What is the purpose of a trust model?

Trust models at a general level can be applied in many different settings. In cryptographic settings, there are very explicit attack models that serve as shorthand for understanding what different actors in secure protocols can or cannot learn. These have names like, "Adaptive chosen-ciphertext attack (CCA2)." These are incredibly useful when evaluating cryptographic protocols against need, but we lack a similar way to talk about SaaS offerings.

Having a common language and definitions make conversations about data privacy needs much easier. Different needs lead to different models and there's a rich spectrum to choose from depending on use cases.

For example, [Customer Managed Keys]() (CMK, aka Bring Your Own Keys or BYOK), when done right, is a trust-but-verify model. This means the customer has control of a master key or keys for decrypting the data that their service provider holds and can see when their data is accessed. The service provider does not have access to this data without the consent and access monitoring of their customer.

What are the SaaS trust models?

There are a number of models that are in widespread use today, though they aren't often called the same thing. It's worth noting that we sometimes see a mixture of models depending on the data or on the level of service.

Here are the trust models we see from most common to least common.

FULL-TRUST

This is the default and classic state for SaaS, IaaS, and PaaS: The service provider has full access to all of the data of its customers but promises not to abuse that privilege. The service provider makes contractual promises in the form of terms-of-service and privacy policies. The single most crucial test for whether a service is "full-trust" is whether or not an administrator of the service could access a customer's data. So even if the data is encrypted, if that encryption is transparent to the administrator or they have access to the keys, it's a full-trust model.

TRUST-BUT-VERIFY

This model is similar to full-trust, except the customer retains an independent audit logging capability on the access of sensitive data so they know who is accessing it, when, and from where. Additionally, the customer has the ability to revoke access to their data from the service provider without relying on the service provider to remove the data. In other words, an employee or system at the provider can see all data, but the customer knows when they access it and can shut off that access at any time.

The Salesforce Shield product is an example of a trust-but-verify model for the data that is protected by that product.

SEMI-TRUST

A semi-trusted provider facilitates access to data (or the ability to decrypt or access it) but can't decrypt the data or grant access itself. Semi-trusted services generally have access to metadata and users are generally not anonymous. A semi-trusted player doesn't have access to the data (zero-trust in this regard) but is trusted with other operations such as revocation.

For example, a semi-trusted server might hold encrypted encryption keys and hand them out, but would not have a way to decrypt those keys nor to determine who can. This is an important model when looking at zero-trust data architectures as it is frequently as good as "zero trust" in an environment where anonymity is unimportant or disallowed.

SPLIT-TRUST

A split-trust security model requires two or more parties to collaborate to be able to see data. A common implementation of this model encrypts data and gives one entity the keys and another entity the encrypted data. Each entity promises not to share its data with the other entity.

We sometimes see people talk about split-trust models within a single entity where trust is spread between servers. At a server-level, this may be true, but at a service provider level, it is not.

So a service provider who splits trust between servers internally cannot say that it meets the SaaS Split-trust model since administrators within the service provider could still access data without needing the collaboration of a 3rd-party.

Note: if the customer holds their own keys, that is not a split-trust model as split-trust implies the customer trusting two or more parties and trusting those parties not to collude. Customer-held keys can be either a trust-but-verify model if data is decrypted and seen by the service provider at their servers, or a zero-trust model if the data is only decrypted by the customer.

ZERO-TRUST

If the service provider holds encrypted data, but can't gain access to the decrypted data at the servers and has limited or no insights about the data it holds, then the model is zero-trust. We sometimes call this zero-visibility or zero-trust data to avoid confusion with the network-level concept of zero-trust.

In a zero-trust data scenario, the service provider doesn't have keys, doesn't decrypt data, and never sees nor could see decrypted data on its servers. The customer or the customer's employees are, however, able to view data by decrypting it in their client (browser, mobile app, or whatever) or on their own servers.

EPHEMERAL-TRUST

When a service provider sees some data on their server, but encrypts it before persistently storing it, we have an ephemeral-trust model. This is useful when storing "toxic data" — that is, private user-generated content like chat logs or conference call recordings that could have regulated data (health, financial, insider-trading, etc.) inside — where the service provider needs one-time access to the data to deliver promised value to the customer. For example, in the case of audio, the service might generate a transcript on their server, then encrypt the transcript and the source audio such that the service provider can no longer access it. This helps the service provider to minimize their exposure in the case of a hack or an overly curious administrator.

COMBINATIONS OF EACH

In many cases where IronCore has worked with customers, we've seen a mixture of trust models. For example, it may be that attachments follow a zero-trust model while sensitive or regulated data follows a trust-but-verify model. For the purposes of these models, we generally ignore data that a customer would not deem to be private.

What's Next? Ask more questions.

Perhaps with a shared language that defines these trust models, we can all better evaluate service providers according to how much trust we must give them if we use their service. The right answer depends on what data we're entrusting, what harm could come from a curious admin or hacker, potential regulatory penalties, and more. But taken together, we can better evaluate the risk we take in partnering with a SaaS provider.

Here are a few questions you can ask service providers to understand what trust model they use:

- Is there anyone in your company who can access my sensitive data?
- If so, is that access persistent?
- If not, what do you know about the data that you can't see? What metadata do you have?
- Is there a way for me to know when someone in your company accesses my data, even if that person is a database or systems administrator?
- Do you have any service providers (for example, Amazon Web Services) where one of their admins could potentially see my data (for example, if they were compelled by a law enforcement order or any other reason)?
- Do you offer a way to encrypt the data such that I can hold the keys?

KNOWLEDGE IS POWER

Unfortunately, in most cases today, SaaS customers find themselves placed in a full-trust data control model with their service provider.

What can they do about it?

Customers who understand their SaaS provider's trust model are in a better position to demand the data control they need to keep sensitive data safe.

SaaS businesses can benefit, too, from taking a proactive approach. Companies that are transparent about their trust model and give customers increased control are winning against competitors and setting the standard for others to follow.

Trust your trust model. Let's talk.

About IronCore Labs

IronCore Labs, the data control and privacy platform, enables software developers and businesses to rapidly build privacy solutions with data control. Whether meeting GDPR or other compliance requirements, handling sensitive data safely, or deploying systems that give customers control of their data, IronCore removes the traditional obstacles to keeping data private and secure, while accelerating time to market for application-layer encryption.

IronCore Labs
1750 30th Street #500
Boulder, CO 80301, USA

Inquiries
Email: info@ironcorelabs.com
Phone: +1.415.968.9607

CONNECT WITH US
blog.ironcorelabs.com
linkedin.com/company/ironcore-labs
twitter.com/ironcorelabs
ironcorelabs.com