Trust and Insecurity

Column: Security

Trust and Insecurity

Bob Frankston
IEEE Consumer Technology Society

PREFACE

WHEN I was first advocating home networking at Microsoft, we encountered a problem. The existing systems and applications had implicitly assumed that they were inside a safe environment and did not consider threats from bad actors. Early Windows systems had not yet provided the file system with access control and other protections though there were some attempts to have separate logins to keep some settings separate.

The temporary solution was to take advantage of the limitations of using a Network Address Translator to share a single IP address. That made internal systems invisible by default.

This problem became most visible when I first got a broadband connection at home in 1995—I could look at the files on neighbors' home computers because of the way NETBIOS worked at the time and there were no filters.

I expected that when IPv6 became available, it would use encryption by default and that applications written for IPv6 would be written to protect themselves from bad actors.

Alas, that did not happen. One reason is that the firewall approach became institutionalized as a feature. Part of this was a simplistic view of security we see in Windows and its idea of private versus public networks that assume a static and unambiguous distinction between good actors and bad actors. It does not help that the systems try to guess whether you are on a public/private network using heuristics. And when it guesses wrong, you start to see perverse and hard-to-diagnose failures.

Rather than making things simple, such approaches make things more complicated and bring back the limitations and complexities of the physical topology. This is why, we have SSIDs for Wi-Fi. They emulate the physical boundary of the wired network, and the limitations of the physical topology.

The idea of a security perimeter is very tempting because it emulates the security perimeters—the walls around our homes and the walls that once surrounded villages. The village walls are no longer the norm because the idea does not work well in the real world.

PERIMETERS

When I first learned about Thread®,[a] I was excited because it is a distributed (or meshed) protocol that builds on the readily available 802.15.4 radios used for Zigbee radios. Soon,

Digital Object Identifier 10.1109/MCE.2022.3231779
Date of current version 9 February 2023.

[a]Thread Benefits (threadgroup.org). [Online]. Available: https://www.threadgroup.org/What-is-Thread/Thread-Benefits

4 2162-2248 © 2023 IEEE Published by the IEEE Consumer Technology Society IEEE Consumer Electronics Magazine

DOI: https://doi.org/10.1109/MCE.2022.3231779

Preface

When I was first advocating home networking at Microsoft, we encountered a problem. The existing systems and applications had implicitly assumed they were inside a safe environment and didn't consider threats from bad actors. Early Windows systems hadn't yet provided file system with access control and other protections though there were some attempts to have separate logins to keep some settings separate.

The temporary solution was to take advantage of the limitations of using a Network Address Translator (NAT) to share a single IP address. That made internal systems invisible by default.

This problem became most visible when I first got a broadband connection at home in 1995 – I could look at the files on neighbors' home computers because of the way NETBIOS worked at the time and there were no filters.

I expected that when IPv6 became available, it would use encryption by default and that applications written for IPv6 would be written to protect themselves from bad actors.

Alas, that did not happen. One reason is that the firewall approach became institutionalized as a feature. Part of this was a simplistic view of security we see in Windows and its idea of private vs. public networks that assume a static and unambiguous distinction between good actors and bad actors. It doesn't help that the systems try to guess whether you are on a public/private network using heuristics. And when it guesses wrong, you start to see perverse and hard-to-diagnose failures.

Rather than making things simple, such approaches make things more complicated and bring back the limitations and complexities of the physical topology. This is why we have SSIDs for Wi-Fi. They emulate the physical boundary of the wired network … and the limitations of the physical topology.

The idea of a security perimeter is very tempting because it emulates the security perimeters – the walls around our homes and the walls that once surrounded villages. The village walls are no longer the norm because the idea doesn't work well in the real world.

Perimeters

When I first learned about Thread®[i], I was excited because it is a distributed (or meshed) protocol that builds on the readily available 802.15.4 radios used for Zigbee radios. Soon, though, I was disappointed to find that there was a security perimeter.

In home and corporate networks, perimeter security is a problem because the system is vulnerable once there is a breach. That is a classic Trojan Horse attack. We've seen examples where an internal system such as an HVAC has been compromised, thus giving the attacker full access to all the other systems.

The perimeter security, AKA firewalls, are an issue because they second-guess networked relationships. I can't just make a connection between two devices – I have to make sure that every setting in the path is just right, and when things fail, I can't necessarily guess which of the rules in the path have been violated and how to fix it.

Some approaches double down on the idea of perimeter security by relying on subnets and other mechanisms to partition communities, such as having a separate subnet for IoT devices. Not only is this complicated to set up and maintain, but it also requires carefully defining each relationship so that one's PC or other controller can reach all the different classes of devices. What happens if a device needs to access a file server?

When using Zigbee, I need to create a separate Zigbee network for each family of devices. By default, Philips devices need their own Zigbee physical network, and I need to worry about which repeater serves which family of devices. Bluetooth has its own problems in that it defines pairwise relationships, so to replace a device (such as a smartphone) I need to carefully re-pair each device to each other. Bluetooth added its own meshing protocol for its own devices – yet another family of protocols and devices.

As with Thread, the assumption is that the devices would connect wirelessly – I have not seen any mention of wired connections for this since the topology of the networks is over-defined – limiting the ability to innovate.

The Simplicity of IP

The stark simplicity of a common IP address space is a sharp contrast with the approach of creating twisting and winding passages between devices.

I can simply place a device anywhere, wired or wireless and expect them to interconnect. Alas, in the current implementations, we still have the constraints of the physical networks.

But because the IP protocol does not have timing limitations built-in, it can be overlayed over other networks and can use encryption to provide a degree of privacy.

For now, we need to add shims using port forwarding, VPN tunnels and other techniques. The key is to honor the strict architectural separation between the relationships and the physical facilities. I use the term Ambient Connectivity™ for the approach of assuming connectivity when developing the protocols. This allows for great flexibility and innovation in providing general connectivity (including a Public Packet Infrastructure[ii]). Any connectivity thus benefits all applications.

Trust Relationships

HTTPS is the protocol used to protect the web. If you see the 🔒 lock icon, you assume you are safe. That's not really true because all it is saying is that the name you typed in the browser matches the certificate name for the site. To get a certificate, the site owner must demonstrate that they can respond to a test message using that DNS name or that they have control of the zone file.

That latter technique enables me to get certificates for internal devices that are not publicly visible and thus inaccessible for a test message used to verify a certificate.

It does not mean you can trust the site any more than you can trust a merchant just because they have a storefront. But it helps ensure you've reached the right destination.

As we see, the real issue isn't security; it is the trust relationships. Those relationships are nuanced and ambiguous. I often point out that there are people I trust with my life but not my money.

A key part of the Internet architecture is the strict separation of the semantics from the packets. The network can't know the meaning of the packets. This is why we use the term "best efforts". If the network doesn't know the intent of the packet, it can't provide services such as reliable and timely delivery. Retransmission, for example, requires arbitrary buffering in the network which frustrates streaming.

More to the point, it can't know which bits are good bits and which bits are bad bits – the network equivalent of Maxwell's Demon. If it claimed to know, it would be second-guessing the intent of the applications causing perverse failures and complexity.

Only the applications know enough to manage their trust relationships and "favors" inside the network are a problem because they don't have the necessary knowledge and become a source of difficult-to-diagnose failures.

There isn't a universal answer to trust, but we can provide better tools. One approach I wrote about is to create communities of things[iii] sharing a secret. That doesn't build in the limits of physical topologies, and devices can be part of multiple communities. It is commonly done within manufacturers' products, but we need such approaches as a successor to managing keys.

To get a sense of the complexity, consider the problem of trusted devices being in error even with the best intentions. There is also the problem of bitrot – when the software and assumptions remain valid, but the conditions around it change. You might trust "childrens.candy," but when the DNS entry for that name expires, it might be repurposed by a bad actor.

That is a reason for the relationships to be defined using abstract identifiers and to not depend on building semantics into the plumbing – a topic I plan to write about in a future column.

Things get even more complicated as the protocols are woven into the fabric of the physical world. When you have smart devices in the home, how do you implement policies without requiring people to log in to each light switch and without having to be explicit about ambiguous social policies? Do your children have the ability to turn on your bedroom light at 3 AM? Those whose kids tell Alexa to play loud, annoying music as they leave the house have experienced the problem.

Learning

As in the early days of Windows, many devices assume they are behind a security perimeter. As imperfect as the approach is, it has given us breathing room to learn better approaches. Unfortunately many have used it as an excuse to declare victory rather than trying to take advantage of the opportunity for learning.

Just as we learn to safetly navigate the physical world, we need to develop new best practices for a connected world. That won't happen without tools for managing communities of devices. Even then, we will get failures. But we can limit the harm if we are wary about trusting other devices and don't naively rely on firewalls as security boundaries.

We tend to focus on the use cases that work with our current approaches. This complacency too often frustrates innovation and limits our future.

---

[i] Thread Benefits (threadgroup.org)
[ii] https://rmf.vc/IEEEPPI
[iii] https://rmf.vc/IEEECommunitiesOfThings