Access Control and the Internet of Things

Vinton G. Cerf • *Google*

I've come to the conclusion that we might need to exercise access control over cyber-physical systems (sometimes called the Internet of Things) at the edge of the network in the device or, at least, a local access controller for the device. We can think of the latter as a sort of local hub for device(s) in a residential or enterprise setting. Roughly speaking, a device should be able to determine whether an external source is authorized either to command, control, or configure the device or to obtain information from it.

What occupies my thinking is how this edge device can reliably determine whether queries or commands are coming from a device that's authorized to exercise the privilege of sending these commands, and that the edge device has the ability to confirm this authority.

For purposes of bounding this exercise, I begin with the assumption that the device has the ability to generate its own private and public keys, and that it can determine whether these are strong enough for purposes of protecting against abusive access. It's also reasonable to assume that the edge device has the ability to execute both public key and symmetric key cryptography. I also assume there's a trusted computing module that encapsulates this ability to perform cryptographic function and resists physical tampering. An example of such a module is Google's Vault system (see http://techcrunch.com/2015/05/29/googles-project-vault-is-a-secure-computing-environment-on-a-micro-sd-card-for-any-platform/#.vzoxfd:aKgv).

Based on these three assumptions, I want to explore possible behavioral and architectural designs that achieve the intent, which is to inhibit improper access to the information contained in or generated by the device and to inhibit inappropriate control over the device.

The half-baked idea is that these edge devices should be able to create unforgeable *capabilities* that they can issue to specific interlocutors so that communication between the edge device and its controller can be secured. Because controllers and edge devices might move around, the bona fides will likely need to be tied to identifiers that can be associated with different (that is, changing) IP addresses over time. This suggests that domain names or other unique identifiers should be a part of the solution. It seems clear that Domain Name System Security Extensions (DNSSEC) will be important to assure that the mapping from domain name to IP address can be validated when an edge device needs to communicate with the controller or perhaps another edge device. More important, however, is the need for both the controlled device and the controller (which might be remote) to be able to confirm the identity and integrity of the other; this will require more than DNSSEC confirmation.

If the controlled device knows its controller's public key and if it can provide to the controller an unforgeable capability that the controller can use to validate its authenticity, then we could have the basis for securing communication from the controller to the controlled device. Similarly, the controller needs to know that it's communicating with a device it knows and not an interloper.

What might a *capability* generated by the controlled device look like? The controlled device might generate an unpredictable *nonce* (large binary value) that it encrypts in the public key of the recipient controller and also digitally signs with its own private key. If the controller wishes to communicate with the controlled device, it must return the nonce, encrypted in the controlled device's public key and digitally signed with the controller's private key.

Published by the IEEE Computer Society
1089-7801/15/$31.00 © 2015 IEEE

The nonce is intended to permit the controlled device to detect replay attacks. This same process could also work to let the controller detect attempts to replay earlier valid messages from the controlled device. Granting the rather casual nature of this description, but assuming it captures some of the mechanisms that might be appropriate, it remains to determine how the controller learns of the controlled device and the controlled device learns of the controller. What's also of great importance is that both devices expect communication from the other.

On the controlled side, we might imagine that local configuration provides the edge device with the public key of the controller and its domain name — this is what security certificates generally accomplish. Plainly, the controlled device needs to have a reason to trust the information that it gets regarding the controller. The controller can assume (I think) that a device that presents itself to be controlled (or monitored) wouldn't do so without having the intent to achieve this objective.

This notional design might merely be a restatement of the utility of public key cryptography, but I would be interested to know whether readers have thought through ideas for securing this kind of system, and perhaps have references to more fully worked out designs that might already be in use.

Vinton G. Cerf is vice president and chief Internet evangelist at Google, and past president of ACM. He's widely known as one of the "fathers of the Internet." He's a fellow of IEEE and ACM. Contact him at vgcerf@gmail.com.