IoT: The Internet of Terror

IF IT SEEMS LIKE THE SKY IS FALLING, THAT'S BECAUSE IT IS.

who is KV?

click for video

Dear KV,

We're deploying a consumer IoT (Internet of Things) device, with each device connected to a cloud service that acts as the platform from which it will be controlled. The device itself isn't dangerous: it's a simple, slimmed-down tablet to be used in hotel chains to replace an alarm clock and TV remote, and to provide access to room service. The device is battery operated, rechargeable, and cheap enough that hopefully no one will want to steal it. Guests can't load any information into it, and, unlike a typical tablet, it does not serve as a web browser.

We have an engineer who seems overly worried about the security of the communication between the device and the back end. We are working on the alpha version of the system now. All communication between the device and the cloud is unencrypted, and no user data crosses the network. Who cares what TV channel you're watching or what you ate for breakfast? On such a lightweight embedded device, the battery-life cost of encryption is significant and reduces battery life by 25 percent in our tests. We expect many users will not replace them in their charging cradles, since most people can't remember to charge their own phones every night; thus, battery life will be very important to the project.

Even if we do turn on some encryption, we would like it to be as little as possible, again, to preserve battery life. I know a longer key is harder to break, but it also means that we'll be using a lot more of the battery protecting what is, in reality, data that is hardly a state secret. What do you think is the right level of encryption, if any, to use in such a system, and how can we get the annoying engineer to shut up?

Not So Secret

Dear Not So,

It is true that many security-focused engineers can sound like Chicken Little, running around announcing that the sky is falling, but, unless you've been living under a rock, you will notice that, indeed, the sky IS falling. Not a day goes by without a significant attack against networked systems making the news, and the Internet of Terror is leading the charge in taking distributed systems down the road to hell—a road that you wish to pave with your good intentions.

Before we get to the question of encryption and key length, I would like to point out two things. An IoT device is nothing more than an embedded system with a TCP/IP stack. It is not a magical object that is somehow protected from attackers because of how cool, interesting, or colorful it is.

Second, and I can't believe I have to point this out to people who would read or write to KV, but the Internet has a lot of stuff on it, and it's getting bigger every day. Once upon a time, there were fewer than 100 nodes on the Internet, and that time is long past. KV has three Internet-enabled devices within arm's reach, and, if you think a billion users of the Internet aren't scary, try multiplying that by 10 once every fridge, microwave, and hotel alarm clock can spew packets into the ether(net).

Let's get this straight: if you attach something to a network—any network—it had better be secured as well as possible, because I am quite tired of being awakened by the sound of the gnashing of teeth caused by each and every new network security breach. The Internet reaches everywhere, and if even one-tenth of one percent of the people on it are bad actors, then that's one million potential attackers across the earth, and each one may be in control of far more than three devices!

If you think a billion users of the Internet aren't scary, try multiplying that by 10 once every fridge, microwave, and hotel alarm clock can spew packets into the ether(net).

Encryption, in and of itself, is not a solution to securing a system. There are many components that go into building a secure system and service, most of which I won't go into here due to limitation of space and my blood pressure medicine, which my doctor tells me not to grind between my teeth. Encrypted communication is one tool to be used when securing a networked service. You say that there are no "state secrets" in your system, and, I guess, so long as the president isn't staying in one of your IoT-equipped hotels, perhaps that's true. Even if the president no longer eats children for breakfast, there are plenty of other risks involved in building a system as you've described.

If attackers gain access to the private Wi-Fi network that the hotel uses to deploy your system—which is likely, because the hotel will probably pick a password such as "1234567890"—then they can see all the traffic used to perform operations. Want to wake your neighbors at 01:00, 03:00, and 05:00? Simple, just create a request with their room number and the appropriate settings. Perhaps you would like to put a hotel out of business? Easy enough, just order expensive champagne from room service for everyone staying in the hotel. Hotel guests will gladly accept it and then sue to have it removed from their bills. Attacks don't have to occur at the NSA/FSB/GCHQ or similar levels to run you and your customers out of business. I think you can see why your engineer is talking about encryption.

You didn't mention authentication at all, so let's talk a bit about that as well, because authentication protocols also require the use of cryptographic algorithms—the ones, you complain, that drain your batteries. The system you describe needs the ability to discern who made a request as a way of preventing the attack I just described, where I ordered champagne for everyone in the hotel. How are you going to make sure that room 237 really ordered those drinks?

Now that I've convinced you that connections to the back-end system need to be both encrypted and authenticated, we can split hairs on how big a key to use, and, you know what they say, the bigger the key, the deeper the lock! Choosing algorithms and key sizes is where we can start to talk about device features and power usage.

At the moment, the AES (Advanced Encryption Standard) set of algorithms is what is used most often for encryption, and this protocol has at least three standard key lengths: 128, 192, and 256 bits. There are various recommendations on key lengths, but I rather like the comparison of several sets of recommendations on this website: https://www.keylength.com/en/4/. NSA—whoops, I mean NIST—suggests AES-128 as a minimum from 2016 through 2030, and as your system does not contain state secrets, it ought to be sufficient for your use.

Many embedded microprocessors now come with dedicated circuits to offload cryptographic operations. The offloading of crypto algorithms is nothing new and is subject to the same "cycle of life" that we see in other areas of software and systems engineering, where specialized services first appear as add-on peripherals, then show up in the CPU itself. In the 1990s microprocessors weren't able to keep up with the then-new algorithms being used to set up VPNs (virtual private networks), so companies produced cryptographic offload chips.

Processors got faster and larger, and what couldn't be handled by frequency scaling was put into special instructions to handle the harder parts of crypto. Embedded processors, like all processors, continue to gain more circuits, including those for cryptography. Any device, such as yours, that drives a display is bound to have some sort of cryptographic engine, and the only reason to ignore it is sheer laziness. It is very likely that your embedded processor already has

Related articles

➡ Kode Vicious Battles On
George Neville-Neil
A koder with attitude, KV answers your questions. Miss Manners he ain't.
http://queue.acm.org/detail.cfm?id=1059801

➡ Pervasive, Dynamic Authentication of Physical Items
Mandel Yu and Srinivas Devadas
The use of silicon PUF circuits
http://queue.acm.org/detail.cfm?id=3047967

➡ Security Collapse in the HTTPS Market
Axel Arnbak, et al.
Assessing legal and technical solutions to secure HTTPS
http://queue.acm.org/detail.cfm?id=2673311

acceleration for AES, and if it doesn't, spend a quarter and buy a better one.

KV

Kode Vicious, *known to mere mortals as George V. Neville-Neil, works on networking and operating-system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, and rewriting your bad code (OK, maybe not that last one). He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. Neville-Neil is the co-author with Marshall Kirk McKusick and Robert N. M. Watson of* The Design and Implementation of the FreeBSD Operating System *(second edition). He is an avid bicyclist and traveler who currently lives in New York City.*