Certifiably Secure: Does it Matter?

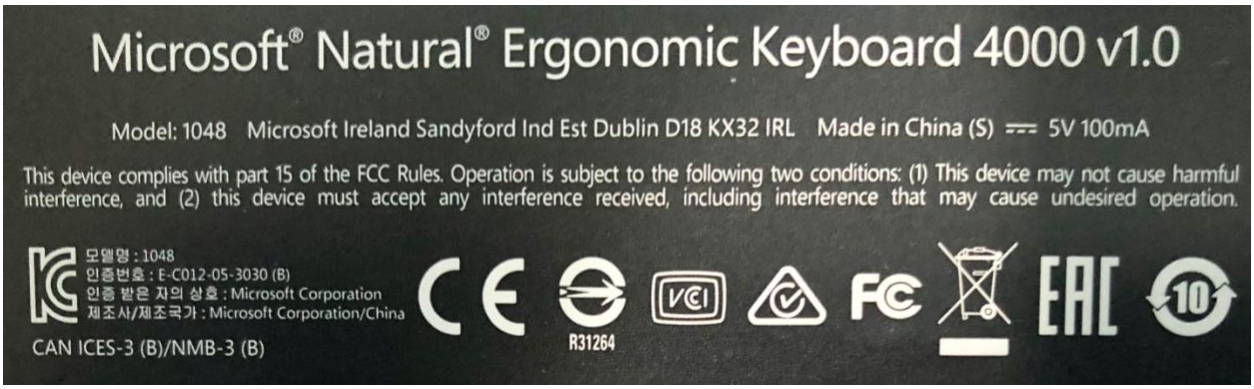
Charles Wilson, Principal Engineer, Cybersecurity Development Lifecycle Practice

11/3/20 10:47:00 AM

Category: security-certification

Tags: security, cybersecurity, autonomous vehicles, certification, ISO 21434

When I turn my keyboard over (which doesn't happen very often), I see the following:



As someone who spent years designing and implementing wireless keyboards, I'm familiar with most of what's going on here. Most people aren't and that's okay.

Even without all these certifications, I'm pretty sure that this device wouldn't have had any problems playing nicely with the other devices connected to my computer. I don't give these certification marks a second thought in my daily life. You probably don't either.

Here's the question: If a device does what it's supposed to and was designed and built by professionals who followed industry best practices, do the certifications really matter? And what does it mean when we say a device is certified?

In my post, **Purpose-driven Security** [2], I mentioned that security was a relatively recent addition to the list of things we cared about. When it comes to software development, the paint has yet to dry on the certification sign. It's not something that the software industry has really ever been able to get behind. People still argue over how many spaces should be in an indent level [3].

Let's take this apart, define terms, identify where we want to go, and do a little cost-benefit analysis.

Coming to Terms

Oxford defines *certification* as "the action or process of providing someone or something with an official document attesting to a status or level of achievement." In the case of security, the certification of interest is **ISO/SAE 21434** (**Road Vehicles — Cybersecurity Engineering**) [1].

When we look at this standard (*currently a draft standard*), we see that what is being certified is the process of governance, product creation, operational management, and decommissioning. Basically, the entire lifecycle of the product in the context of security is under consideration.

For those of you who've not run away screaming "No, that's impossible" in a Luke Skywalker-esque voice, let's just take a breath and ask, "Why is the scope of 21434 so big?"

Security Has Practical Limits

Unlike other aspects of a system, security doesn't isolate well. The various aspects of security interact.

Because of this, you need to reach a certain aggregate level before you can assert that the system is secure. Beyond this, security becomes stronger, but you reach a point of diminishing returns.

Let's consider a very relatable example, the humble cheese sandwich. As a backpacker, one quickly learns that there are numerous actors waiting for you to fail to properly secure your food. They provide varying levels of threat.

No one would argue that leaving the sandwich out, on a plate, unattended just invites trouble. We could leave it in a shopping bag near the cooking gear, but that's not much better.

Against most actors, a cooler with a good clasp provides a good balance of convenience and security, but this isn't always an option. A bear bag in a tree away from the camp may secure the sandwich from that actor, but squirrels and the like will have little trouble getting at it.

Additionally, accessing the sandwich is now much harder. Finally, we could use a steel bear-proof box bolted to a concrete pier. Although a valid option, you can't take it with you.

What we really want is enough security to protect the sandwich until we're ready to eat it and as little inconvenience getting to it as possible. As a side note, one should never underestimate insider threats, such as the pet dog. They represent a persistent threat and are fully aware of all your security measures and operating procedures.

Just like this example, below a certain point we effectively have no security and above a certain point we have no utility.

Security Can't Be Bolted On

In fast-moving world, who has time for systematic consideration of security? Most projects build the product, expose it to all the penetration testing they can budget for, patch the egregious issues, and call it good.

That time was taken to do pen testing is great, but what about the things that you didn't think of poking at? Or didn't have time to poke at?

How can you attest to the soundness of your design with respect to security? Were poor cryptographic choices made? How will updates be handled? What about when the components need to be repaired by the supplier?

These questions all point to the same truth. You have to start with the assumptions that security is an intrinsic attribute of the system, and that the system dictates the type and strength of security necessary.

Show All Work

The thing about security certification is that it's a high bar to get over. Consider the following areas that need to be included:

- **Governance** (project management, roles and responsibilities, document tracking, training, issue tracking, traceability, audit, continuous improvement)
- **Infrastructure** (tooling, build systems, source code control, code signing, archiving, provisioning, updates, decommissioning)
- **Design** (requirements, threat modeling, attack surface analysis, design review)
- **Implementation** (secure coding, secure reuse, secure code review, static analysis, dynamic analysis, fuzz testing)
- **Verification** (penetration testing, threat model review, attack surface analysis review)
- **Operation** (threat monitoring, incident response, incident reporting)

Not only is there a lot here, but it needs to be tied together creating a coherent story of how security is inherent in the product. We'll cover that issue in an upcoming post, **Traceability: Making the Case for Certification** [4].

There and Back Again

Don't panic. There's no expectation that anyone is going to have all these areas under control out of the gate. Take a look again at the governance list. See the last item, continuous improvement? Everyone is expected to make their best effort and show how they are improving in those areas where they are weak.

Even if it takes a full development cycle to emplace all the processes, you get credit for the things you are able to do. You may not get certification the first time around, but you will be well-positioned for the second iteration.

But Is It Worth the Effort?

Let's come back to the original question of whether certification is worthwhile.

If we set aside getting certified because a jurisdiction requires it, we're left with certification as an external validation of a specific security culture. We show that we consider security important enough that we are willing to put in place, track, and audit (internally and externally) policies, processes, and procedures that all support making informed and documented actions and decisions; and being willing to be held accountable for those actions and decisions. Attaining certification makes a powerful statement when you consider the time and effort required to attain it.

Everyone agrees that security is important, helps to ensure safety, and is the right thing to do. At the end of the day, that agreement is just talking the talk. Certification is walking the walk.

References

1. **ISO/SAE DIS 21434 Road Vehicles – Cybersecurity Engineering**
   https://www.iso.org/standard/70918.html
2. **Purpose-driven Security**
   https://github.com/nutonomy/AVCDL/tree/main/background_material/blog_posts/purpose-driven%20security.pdf
3. **Indentation Style**
   https://en.wikipedia.org/wiki/Indentation_style
4. **Traceability: Making the Case for Certification**
   https://github.com/nutonomy/AVCDL/tree/main/background_material/blog_posts/ traceability%20-%20making%20the%20case%20for%20certification.pdf