Establishing a Root of Trust in Embedded Linux and IoT


by Anita Buehrle on April 18, 2022

With IoT, 5G and embedded devices becoming a larger part of everyone's daily lives, security—and more importantly, trust in our technology—is on everyone's minds. Embedded devices don't have a good security track record; the last several years saw a significant number of high-profile hacks that could prevent people from widely accepting IoT into their homes.

The proliferation of hacks and the threat to basic infrastructure resulted in a move toward regulating the security of critical software. Specifically, Executive Order 14082, issued by United States, drew up a list of security practices, including the inclusion of a software bill of materials (SBOM), with every application run by the U.S. federal government. The National Institute of Standards and Technology (NIST) is also creating reference architectures and templates for application security as a result of the Executive Order. Regulation is coming to software security that will likely impact every company that produces code or sells products running on code.

IoT and Embedded Need More Than SBOMs

When it comes to securing the software supply chain (SSC) on embedded devices, regulation and SBOMs are only a small part of the story. From a developer and operator point of view, the more important issue is knowing that what you are running and deploying to are from trusted sources and also that the attack surface is minimized. Most importantly, in the event of a malware breach, you must also mitigate, isolate and prevent any further damage to your network.

Identifying Security Gaps in IoT Devices

IoT and embedded devices have been with us for quite some time and, with the exception of your mobile phone, are typically devices that you don't think about much. Take your router, for example; you probably only worry about it when the internet goes down. I'm sure that many of you don't ever log into it or even know how to upgrade your router's software and firmware.

Unchanged default logins on these types of devices are one of the most common problems and the easiest way into an embedded device. But aside from default passwords, another issue that arises is when a critical vulnerability has been reported. Can you easily patch and update the device's firmware and software?

"One of the main problems with IoT security at the present time is that the rush to market often de-prioritizes security measures that need to be built into our update processes and into the devices themselves," said Ricardo Mendoza of Pantacor.

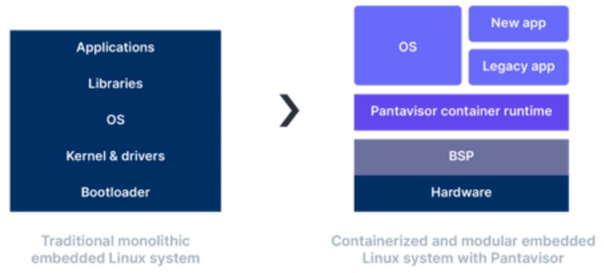
As more and more IoT devices come online, the ability to quickly respond and update firmware and software is critical to security. But the ability to update quickly is also only part of the solution when it comes to securing and trusting your IoT devices. What's also needed is a guarantee that what you're updating is coming from a verified and trusted source.

Modular Vs. Monolithic Systems and Deploying Updates

Unlike cloud development, embedded software development has not embraced a lot of the newer high-velocity strategies. One of the main reasons for this is a lack of adoption of immutable infrastructure, like containers. While there are some solutions out there that run Docker containers, all of them require a significant number of resources to run and support the official Docker engine container manager.

Even though large-spec devices will prevail in the end, today, the internet of things is mostly made up of low-spec devices. Smart lightbulbs, thermostats and other such IoT devices run on extremely limited resources, some with a minimum of 32MB of NAND, NOR or EMMC storage and a minimum of 64MB of RAM.

Most embedded devices have a monolithic architecture which makes them error-prone and time-consuming to update. To update and patch a device quickly, you need a modular architecture. Containers provide that ability, but the container manager or engine on the device must be small and efficient enough to run in a low-spec environment. Pantavisor is one such open source minimal container runtime that enables developers to componentize embedded systems and go from a monolithic architecture to a microservices or component-based approach.



Embedded Linux systems: monolithic vs. modular

Containerized Host OS Automates Updates

Another advantage to an embedded-first container manager is its ability to containerize the host OS and run the containers in the userland alongside any apps. A modular architecture based on containers means that updates can be deployed separately and when necessary using the latest automated deployment pipelines and other Agile strategies employed in the DevOps world.

Docker engine vs Minimal container runtime

Root of Trust in Embedded Systems

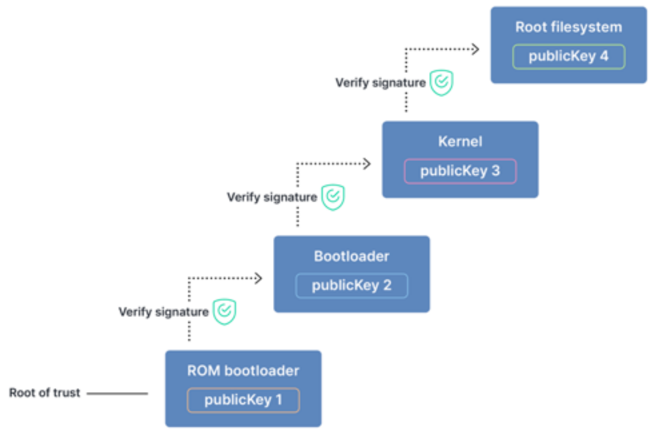
Devices in the field must adapt to the needs of consumers and to keep up with these demands, software and firmware must be continuously deployed without compromising the core contract of security and privacy between the end user and the provider. While being able to update the device itself is important, even more essential is having a framework of trust in place.

Signing software components to validate their authenticity isn't anything new. Signing and validation technologies are well-known and in use today for monolithic embedded firmware. But the push to modernize embedded architecture by means of modular software delivery means we need to also take the tried-and-tested concepts of signature validation to the next level of complexity.

Where Does the Root of Trust Start?

The foundation of trust begins at the level of the ROM code and the bootloader on the hardware and is provided by the manufacturer. The bootloader public key's storage location depends on the board implementation, but typically implements some sort of system on chip (SOC) security; for example, one-time programmable memory (OTP) or trusted platform module (TPM) hardware. Once the ROM confirms the bootloader signature, the bootloader is launched.

Each process then is verified through a signature and trusted at that point to run in the system. This is a standard root of trust implemented on most Linux systems, both desktop or embedded.

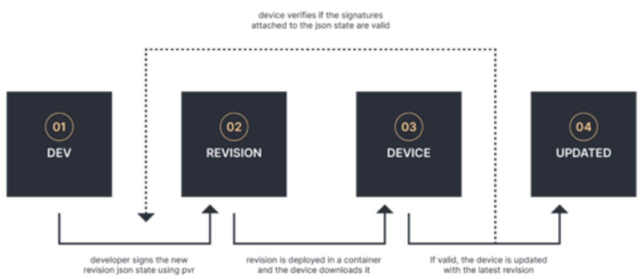


Secure boot process in an embedded Linux stack

Embedded State Revisions and Trust

Embedded systems have the added complication of not only booting and verifying builds from the ROM to the Linux stack and the apps running in the userland, but in the case of a failed update, the system must also be able to rollback to a good state with a revision and verify it on bootup. In addition to this basic functionality, any signing and verification system for embedded must also be lightweight enough to run smoothly in low-spec embedded environments.

In the chain of trust, a container runtime manager, like Pantavisor, for instance, that runs in the init namespace is first verified by the bootloader. Any state revisions are built into the container, then signed, deployed and then verified before they run on the device. An additional step before the deployment may also include a vulnerability scan of the container itself in an automated pipeline.

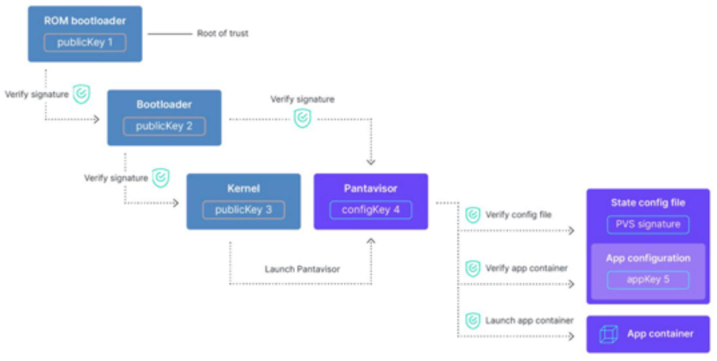


Developer workflow for signed and verified containerized updates

JWT Standard for Verifying Embedded Updates and Revisions

Pantavisor implements a standard JSON configuration that describes the components of a Linux container. The JSON file kept in Git represents the device state at a point in time. They are signed on checkout using Pantavisor Signatures (PVS) on the command line and verified with the JSON Web Token (JWT) standard or root certificates present on the device.

Once built, the container is signed, deployed and verified by the device against any number of artifacts (depending on your integrity settings) on the device before running in the trusted environment. In the case of a deployment problem, the system can also be remotely rolled back by selecting a previously verified “good configuration” that will get deployed in place of the bad deployment.



Verifying containerized apps in embedded Linux

Final Thoughts

Verification and trust is one significant way to reduce the attack surface for malware and other outside bad actors to find a way to take down your device fleets. But what if a CVE beyond your control occurs? Perhaps an exploit in critical infrastructure is found and reported on much later—even years later—after you’ve deployed your trusted code?

In this case, what’s needed is a way to mitigate the damage before an update can be applied to stop the exploit from spreading across your network. One such solution is open source Pantacor lightweight container technology that allows you not only sign, verify and guarantee code updates, but also enables early detection and isolation when a bad actor or CVE is identified.

embedded, iot, Linux, Root of Trust, smart devices

← Can GitOps Improve Application Security?

99% of Executives Listed on More Than Three-Dozen Data Broker Websites →