Poster: Improving Legacy IoT Device Security through Open-Source Intervention

Conner Bradley, David Barrera
Carleton University
connerbradley@scs.carleton.ca

***Abstract*—When a vendor can no longer support an Internet of Things (IoT) device, what is next? With the rapid growth of IoT, vendors are constantly motivated to develop new products to keep up with market demands. Over time, older products become "legacy" or "unsupported" by vendors and no longer receive software updates and security patches. Due to the long-term deployment nature of many IoT devices, they may remain active for years with several unpatched vulnerabilities. We first examine the security threat that unsupported and unpatched devices pose to IoT ecosystems, and then propose a model that enables legacy IoT devices to transition to an open-source support model while respecting the original vendor's intellectual property.**

I. Introduction

In recent years, the Internet of Things (IoT) industry has flooded the market with a variety of novel devices intended for deployment in long-term settings. Such devices include smart thermostats, light switches, and door locks. The deployment of these devices poses a unique problem for the industry: can these devices last as long as their analog counterparts? It is certainly possible to engineer embedded hardware that lasts decades; however, can *software* be engineered to endure the test of time?

Most IoT devices have a critical dependency with their vendor – if the vendor ceases to support the device [11], or the vendor ceases to exist [5], then there is no *supported* method for the device to stay up to date. When these IoT devices become unsupported they may retain some of their original functionality; however, these devices will no longer receive software updates to fix vulnerabilities, thus forcing consumers to choose between keeping their unsupported and insecure device or throwing it away. If a device becomes unsupported and it cannot function due to requiring external vendor services then it is effectively useless to consumers and will also get thrown away.

Unsupported IoT devices pose a security problem: as these devices age and their dependencies become increasingly out-of-date, the attack surface of these devices will grow. Bad actors have historically exploited fleets of functional but insecure IoT devices, which is damaging to IoT ecosystems as a whole [2].

As a result, the Internet of Things will create a surplus of unsupported and vulnerable devices. Many IoT devices are not easily recycled due to the use of custom-built hardware and unrecyclable materials [4], [6]. Our current rate of e-waste generation is unsustainable: in 2019 Northern America created 7.7 metric tonnes of e-waste, with only 15% properly recycled [3]. Concrete action is needed for the Internet of Things to grow sustainably and securely.

Within this context, there is a reliance upon a single entity to maintain IoT devices, which is problematic. In an ideal world, a vendor would hand off all sources, binaries, and compiler tooling to an external entity willing to take on supporting legacy devices. This external entity could take over if the original vendor goes out of business or deprecates the device. In practice, this is unlikely to happen as sources and binaries may contain intellectual property that the vendor cannot make open source, compiler tooling may be proprietary and require licensing, and the entity taking on these tasks will have the burden of managing several different toolchains, build processes, and languages.

Prior work in designing IoT software update schemes put the edge case of product discontinuation or maintenance delegation to be out-of-scope of the software update model [7], or rely on an explicit delegation by pushing new firmware that contains the needed changes to pull updates from a new source [12]. We believe that addressing this edge case is beneficial to IoT security: allowing the delegation of legacy device software maintenance to new maintainers ensures that these devices can run longer and more securely.

We propose a multifaceted model that allows IoT vendors to safely transition the burden of legacy device maintenance onto a third party. Our model takes into consideration a variety of angles of various stakeholders such as intellectual property (IP) concerns, development tooling, customized hardware, and device longevity.

II. Open-Source Transition Model

One major challenge with allowing IoT devices to fetch firmware from alternate sources is prior knowledge of how the device is constructed. Existing firmware update models require the delegated firmware developer to have some prior knowledge of the device, typically in the form of source code. Without prior knowledge, a new maintainer will need to reverse-engineer the device, which is possible but infeasible at a large scale [9]. To enable an open-source development model while respecting stakeholders' IP concerns, we need to provide a clear separation of responsibility between the original device manufacturer and future possible delegate stakeholders. Instead of building monolithic device firmware, we propose adding lightweight abstractions that allow for a

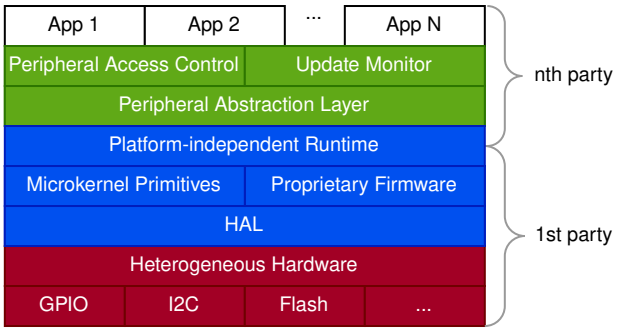


Fig. 1. Our IoT system development model separates responsibility between the first-party maintainers and future delegate maintainers. Red indicates physical hardware, blue indicates native-executable firmware, and green indicates platform-independent programs.

clear separation of responsibility for maintenance concerns on IoT. As shown in Figure 1, the device vendor is responsible for the physical hardware and peripheral selection, along with developing a minimal Hardware Abstraction Layer (HAL) that provides access to the underlying hardware – any proprietary firmware required for peripherals to function is included in this layer, isolating it from higher layers of the system.

Under our model, the core applications that run the IoT device are developed using a lightweight platform-independent runtime that exposes standardized interfaces to the underlying hardware, such as the WebAssembly Micro Runtime (WAMR) [1], [8]. This way the core applications and abstractions for the IoT device can be developed using interfaces and tooling that is independent of the device's underlying hardware. Both a first-party IoT vendor, and third-party open-source developers can develop generic and re-usable software for heterogeneous IoT devices, without any proprietary tooling, sources, or firmware blobs [8].

We plan on leveraging the new IETF SUIT standard for resource-constrained firmware updates [7] to implement our update distribution model. We plan on integrating and extending the TUF design for survivable key compromise [10] into IETF SUIT to allow a key-delegation system allowing private vendors and third-party maintainers to openly author and distribute device firmware.

To transition software updates from first-party vendors to third-party development, the IoT device vendor needs to choose a *transition heuristic* that defines when an IoT device will switch to third-party sources. Potential heuristics include checking if a vendor API endpoint is unusable for a certain amount of time (dynamic), checking if an amount of time has elapsed (temporal), or pushing an update to the IoT device to enable the transition (manual). During the pre-transition period, the IoT vendor can publish firmware and applications for their IoT device through their channels.

Threat Model: anything grouped under the first-party responsibility is assumed to be trusted. Over the supported lifetime of the device, the vendor will have the ability to fix any bugs in the hardware abstraction layer or proprietary firmware, which will be minimal if not any as these components are designed to require a minimal implementation. The platform-independent runtime is assumed to be trusted. Higher-level components that rely on platform-independent programs are assumed to be untrusted.

III. Conclusion and Discussion

We presented our preliminary work that extends the lifetime of IoT devices by allowing for third-parties to develop IoT systems software using open standards. Future work includes evaluating various platform-independent runtimes such as WAMR [1], along with developing a standard set of interfaces that can be implemented on popular IoT operating systems.

References

[1] T. B. Alliance, "Webassembly micro runtime (wamr)."
[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Aug. 2017.
[3] V. Forti, C. P. Baldé, R. Kuehr, and G. Bel, "The global e-waste monitor 2020," *Quantities, flows, and the circular economy potential*, 2020.
[4] S. Higginbotham, "The IoTs E-Waste Problem Isnt Inevitable," Jul 2021.
[5] G. Hoffman, "Anki, jibo, and kuri: What we can learn from social robots that didn't make it," Jul 2021.
[6] S. Lechelt, K. Gorkovenko, L. L. Soares, C. Speed, J. K. Thorp, and M. Stead, "Designing for the End of Life of IoT Objects," in *Companion Publication of the 2020 ACM Designing Interactive Systems Conference*. ACM, Jul. 2020.
[7] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, "A Firmware Update Architecture for Internet of Things," RFC 9019, Apr. 2021.
[8] N. Mäkitalo, T. Mikkonen, C. Pautasso, V. Bankowski, P. Daubaris, R. Mikkola, and O. Beletski, "WebAssembly Modules as Lightweight Containers for Liquid IoT Applications," in *Web Engineering*, M. Brambilla, R. Chbeir, F. Frasincar, and I. Manolescu, Eds. Springer International Publishing, 2021.
[9] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer IoT devices: A multidimensional, network-informed measurement approach," in *Proceedings of the Internet Measurement Conference*. ACM.
[10] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine, "Survivable key compromise in software update systems," in *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*. ACM Press, 2010.
[11] N. Statt, "Nest is permanently disabling the revolv smart home hub," Apr 2016.
[12] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, and E. Baccelli, "Secure Firmware Updates for Constrained IoT Devices Using Open Standards: A Reality Check," *IEEE Access*, vol. 7, 2019.

Improving Legacy IoT Device Security through Open-Source Intervention

Conner Bradley David Barrera

Carleton University

Problem: Legacy IoT Devices

What happens when an IoT device is no longer supported by the original manufacturer?

- Unsupported devices can remain in service, running aging embedded software
- The **hardware** will likely be usable, but the **software** will not be updated or patched in any way.
- **Legacy, unsupported firmware** deployed at a large scale poses a **significant security threat** [1]
- Our current rate of e-waste generation is unsustainable [2], [3]. What can we do to securely and sustainably keep legacy IoT devices functioning?

Existing Update Efforts

There are several existing designs for software updates in the IoT space, and more generally in the embedded systems space [4]–[6].
Unfortunately, these approaches are (mostly) dependent on a single vendor to provide support, and aspects revolving around device longevity are ignored.

Open Source Intervention

We propose an architectural shift in the IoT space. Instead of building monolithic firmware, we propose developing an open set of interfaces that allow IoT vendors and Open-Source developers to target heterogeneous IoT platforms.

Additionally, our efforts focus on how we can develop software supply chains for liquid IoT applications [7] that allow for a secure transition from vendor to open-source maintainer.

End of Hardware Life

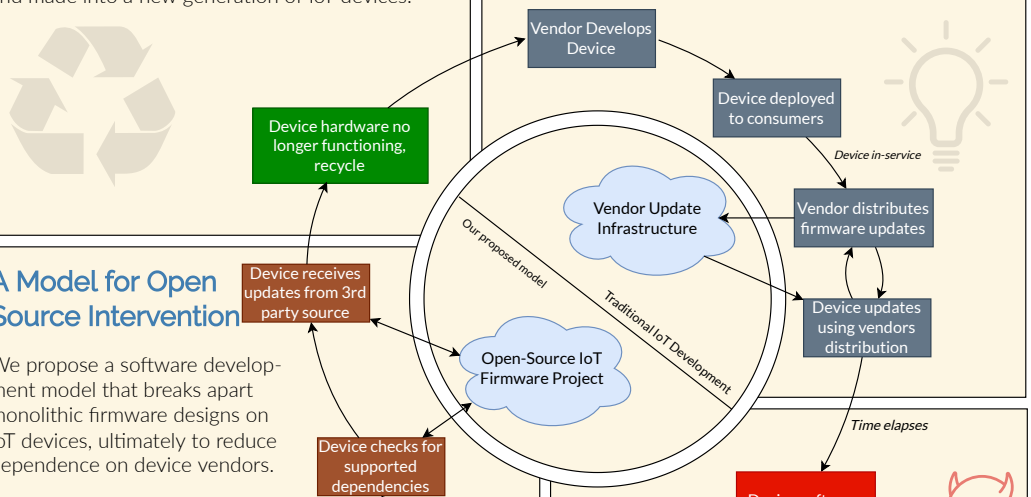
The end of an IoT device can start as early as when the device becomes unsupported; however, If the hardware powering the device is still functional, we can extend the lifetime of the IoT device by leveraging our open-source IoT development model.

At this point the device can be recycled normally, and made into a new generation of IoT devices.

IoT Circle of Life(Span)

All IoT devices start here: developed through R&D efforts by vendors to fill some kind of market need.

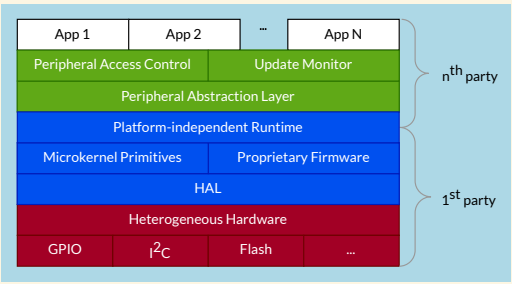
The end-of-life of an IoT device needs to be considered during early stages of product development. We propose developing metrics that allow an IoT device to identify if it has gone outside of the vendor's SLA, or if deteting if their vendor has gone out of business.



A Model for Open Source Intervention

We propose a software development model that breaks apart monolithic firmware designs on IoT devices, ultimately to reduce dependence on device vendors.

Our design allows unsupported IoT devices to be transitioned to third-party software development, without requiring any source code or intellectual property to be given between parties.



Our proposed design, native firmware (blue) is developed only by the original vendor in a microkernel design. Everything else is abstracted to run on a Platform Independent runtime, which exposes standardized APIs across heterogeneous IoT platforms.

Unsupported, Vulnerable

Over time devices become unsupported, and vendors go out of business. Unpatched bugs present a security risk to IoT ecosystems.

In other contexts we have seen open-source communities step in to provide solutions for unsupported platforms. Why can't we do this with IoT?

- Heterogeneous hardware
- Proprietary tooling, methods, and firmware
- If every IoT vendor open sourced their IoT platform code, it would be a maintenance nightmare!

Future Work

- Integrating and extending existing update frameworks to allow for a safe transition to third-party maintainers, such as open source communities
- Building **memory safe** solutions for our platform independent runtime including integrating with microkernel inspired IoT operating systems such as Tock [8]
- Finish developing the standardized platform-independent interafaces for IoT device peripherals.

References

[1] M. Antonakakis, T. April, M. Bailey, *et al.*, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*, USENIX Association, Aug. 2017.

[2] V. Forti, C. P. Baldé, R. Kuehr, and G. Bel, "The global e-waste monitor 2020," *Quantities, flows, and the circular economy potential*, 2020.

[3] S. Higginbotham, *The IoT's E-Waste Problem Isn't Inevitable*, Jul. 2021.

[4] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, *A Firmware Update Architecture for Internet of Things*, RFC 9019, Apr. 2021.

[5] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine, "Survivable key compromise in software update systems," en, in *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*, ACM Press, 2010.

[6] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, and E. Baccelli, "Secure Firmware Updates for Constrained IoT Devices Using Open Standards: A Reality Check," *IEEE Access*, vol. 7, 2019.

[7] N. Mäkitalo, T. Mikkonen, C. Pautasso, *et al.*, "WebAssembly Modules as Lightweight Containers for Liquid IoT Applications," in *Web Engineering*, M. Brambilla, R. Chbeir, F. Frasincar, and I. Manolescu, Eds., Springer International Publishing, 2021.

[8] A. Levy, M. P. Andersen, B. Campbell, *et al.*, "Ownership is theft: Experiences building an embedded os in rust," in *Proceedings of the 8th Workshop on Programming Languages and Operating Systems*, ser. PLOS '15, Monterey, California: Association for Computing Machinery, 2015.