Updates in IoT are more than just one iota

Martin Orehek, Alf Zugenmaier

Munich University of Applied Science
firstname.lastname@hm.edu

Abstract. Deeply embedded systems try to make do with very limited hardware. A number of these systems are expected to remain in the field for extended period of times. At least for security reasons, the software running on these systems will need to be updated. We analyze update sizes for one embedded platform and compare these with the sizes assumed by 3GPP's Cellular IoT traffic model. Because of the expected time it takes to update all devices, we present some ideas of how the network could help to ensure security until updates are applied.

1 Introduction

The Internet of Things (IoT) is envisaged to contain a plethora of different types of devices - from devices such as car infotainment systems that have hardware and processing power that rival any office computer - to deeply embedded systems with severe constraints in hardware, such as individual sensors and actors. For ease of deployment, many of these devices are expected to communicate wirelessly. For example, 3GPP has considered this in their recent study on Cellular IoT [3].

Because there is such a wide variety of different hardware platforms, there are (still?) many different operating systems. An overview of operating systems for IoT is provided in [BKS15,HBPT15].

To get an understanding of update frequency and update sizes, we analyzed the Tinkerforge platform[1], a modular system for prototyping and research, in particular the updates for the Tinkerforge Master Brick. Due to the nature of the the platform, there were no time critical security updates. Furthermore, there is no security implemented on the platform.

2 Traffic model

In their study of Cellular IoT [3], 3GPP defined a traffic model for updates. They assume a Pareto distribution with a minimum size of 200 bytes, a shape parameter alpha of 1.5 and no updates larger than 2000 bytes. The cumulative distribution function for these update sizes is shown in Figure 1. The model estimates an update frequency of 2 per year.

[1] www.tinkerforge.com

To compare this traffic model with an existing IoT device, we looked for an IoT device that has been supported for a number years already on a reasonably stable hardware platform. The Tinkerforge platform fulfills this requirement, with only two major hardware versions, for which an archive of firmware versions reaching back to 2011 is available. Tinkerforge has a modular concept. The core functionality and the "application" is run on a Master Brick, which can be extended with Bricklets implementing individual sensors and actors. Master Brick and Bricklets each have their own firmware. For the purpose of this analysis, we are only looking at firmware for the Master Brick. The most recent software version of the Master Brick firmware at the time of writing this paper was 2.3.4. From version 1.0.0 to the most recent version there was no forking or branching of firmwares detectable.

In order to determine the size of the updates, we were using bsdiff [Per03] to generate binary patches from one firmware to the next. Bsdiff generates compressed patch files. If decompression utilities are not available on the IoT platform, a complete firmware image or modified memory pages need to be sent. Thus, we assume bsdiff gives a reasonable lower limit for order of magnitude of the size of the patches.

Looking only at updates of the revision number, i.e. leaving the major and minor version number intact, we find an average compressed patch size of 3300 bytes with a standard deviation of 2700. The cumulative distribution function is depicted in Figure 1.

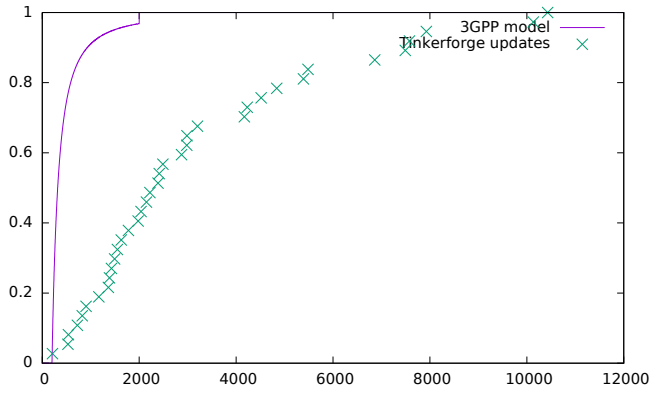

Fig. 1. Cumulative distribution function for update sizes for 3GPP model and Tinkerforge revision updates

It is obvious that the 3GPP model underestimates the updates sizes by approximately a factor of 10. It is unclear whether a Pareto distribution is a good model for update sizes at all.

Within a space of 5 years, there were 47 versions, of which 2 major and 8 minor. Therefore, the 3GPP estimate of update frequency can be considered too low by up to a factor of 5.

The analysis above did not consider any application updates that may be required as well, so the numbers in reality may be higher.

To assess network impact, let's consider a wireless IoT network such as 3GPP's cellular IoT system. In 3GPP's model, updates take up about 0.1The analysis above has shown, that the required capacity for software updates may be underestimated by as much as a factor of 50, thus taking the time to update all devices to over one week. This could disrupt normal services for the applications running on these devices.

3 Network based solution approaches

We would like to propose a few ideas on how to approach the problem of software updates

- Higher data rates would be most desirable, but not always achievable due to ressource limitation on the device side and due to cell sizes on the wireless network side.
- Different types of updates have different urgency. An update for functionality could be scheduled with a lower priority than a security critical update.
- In order to reduce the window of opportunity for all but zero day exploits, it would be advantageous to distribute smaller updates more frequently, rather than sending out one huge update semiannually.
- Network segmentation can help, whereby the IoT network is segmented into security domains. Access to these domains is only through gatekeepers which are sufficiently hardened and enforce access control rules.
- The network could offer the service of filtering out certain potentially malicious traffic before it arrives at the IoT devices until they are fully patched. For vulnerable IoT devices, the network operator could deploy a stateful firewall similar to the one described in Wang et al. [WGSZ04], which approximates the device's state in order to detect potentially harmful traffic. These firewalls could be updated more quickly than the IoT devices themselves.

4 Conclusion

The analysis given in this paper shows that the network load imposed by software updates can be substantial, up to the point where it may become disruptive. Of course the analysis needs to be extended to more IoT platforms to become statistically relevant. Of course, not only network traffic load and speed of delivery is an issue with IoT updates, but also security and reliability of the update delivery as well as installation of the update itself. Furthermore, it remains to be seen whether mechanisms can be put in place to ensure maintenance of software for legacy IoT device.

References

[3GP15] 3GPP. Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT). TR 45.820, 3rd Generation Partnership Project (3GPP), December 2015.

[BKS15] Tuhin Borgohain, Uday Kumar, and Sugata Sanyal. Survey of operating systems for the iot environment. *CoRR*, abs/1504.02517, 2015.

[HBPT15] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes. Operating systems for low-end devices in the internet of things: a survey. *IEEE Internet of Things Journal*, PP(99):1–1, 2015.

[Per03] Colin Percival. Naive differences of executable code. *Draft Paper, http://www. daemonology. net/bsdiff*, 2003.

[WGSZ04] Helen J. Wang, Chuanxiong Guo, Daniel R. Simon, and Alf Zugenmaier. Shield: vulnerability-driven network filters for preventing known vulnerability exploits. In Raj Yavatkar, Ellen W. Zegura, and Jennifer Rexford, editors, *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 30 - September 3, 2004, Portland, Oregon, USA*, pages 193–204. ACM, 2004.