Robust Security Updates for Connected Devices

Jonathan Sönnerup && Jonathan Karlsson, *Lund University*

The Internet of Things (IoT) revolution has just started and there is a tremendous increase in the number of devices connected to the Internet. Some companies estimate it to reach a number between 20 and 50 billion connected devices by the year of 2020.

With such a remarkable amount of devices, society will face an unprecedented security challenge. The software does not only need to implement security features and be built in a robust way, it also has to stay updated by receiving security patches in case vulnerabilities are found. A big problem is products using obsolete software versions – products no longer maintained by the manufacturing company. All of these products can pose a threat to the society and user privacy if the software is outdated and exposed to the Internet. Recently (January 2016), exploitable IoT devices have been easier to find due to the search engine Shodan which lets users search the internet for connected devices, often vulnerable ones. One can easily find a camera feed, lacking authentication mechanisms, of sleeping babies for example. It is of utmost importance to find viable ways to increase the security in all IoT devices, even the older ones. In a survey, it is presented that it takes on average 100-120 days for businesses to remediate a vulnerability, after it has been publicly known. It is also shown that only after 40-60 days, there is a 90 per cent risk that the vulnerability already has been exploited. With a cost efficient and fast security update mechanism, the security can be maintained in the long term.

One way to increase the security is to always keep the devices up-to-date. In order to achieve this, a well-defined patch management process is necessary, here depicted in Figure 1. Since the devices may be located in inaccessible areas, automatic updates are of interest. This process may be utilized in any area of software patch management, but the focus will be on IoT devices. IoT devices differ from the usual computers and smartphones. They often have a small, low-power processor, or microcontroller unit (MCU) with a small amount of flash memory and RAM. Due to the resource constrains in the devices, ordinary communication and security protocols are often not applicable. New, optimized protocols are developed targeting smaller devices but the choice is left to the developers. Security has always come in second hand and for IoT, there is no exception. It is a difficult task to implement robust security in devices with small memory and slow processors but it is an absolute necessity. The thesis work has been done at a camera company, hereinafter known as “CC”.

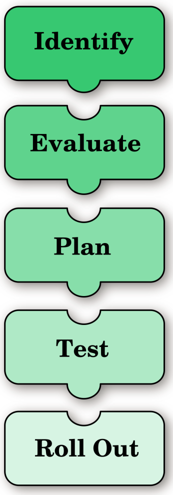

Figure 1: *The process of keeping things up-to-date*

Results and Conclusions

- We have analyzed the current solutions of assessing software vulnerabilities to determine their strengths and weaknesses. We then applied the current methods to known vulnerabilities to determine what can be improved. By taking configuration and environment parameters into account, one gets a more efficient evaluation which is more fine-grained. This is also easier to automate and will thus reduce the time for companies to remediate vulnerabilities. The improved process is seen in Figure 2, where additions to the normal systems (CVSS, CWSS etc.) have been added, as well as configuration and environmental considerations.

- We have also looked into use cases of current update processes of different devices. As references, two well-working update processes were considered – Android OS and Chrome OS. Those were compared to update processes at customers of CC and the IoT world to map problems and possibilities. A proposed general solution for IoT devices and also bigger systems is presented and here seen in Figure 3. This is a semi-automated solution which needs some form of user input, for example a IT department that decides when to update. A fully automated solution was also presented, which is the same as the semi-automated but without the user input.

- We have implemented a proof-of-concept program to update files on a CC camera and to show the need for a signed and verified firmware. An example of the program in use is seen in Table 1, where our patch is compared to the current update process at CC with replacing the whole firmware.

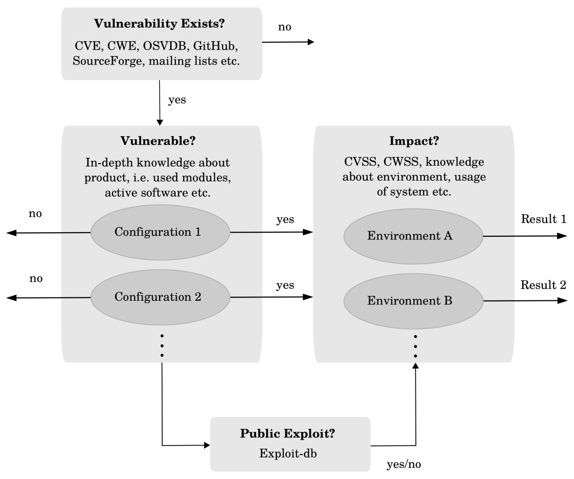
Vulnerability Exists?
CVE, CWE, OSVDB, GitHub, SourceForge, mailing lists etc.
no
yes
Vulnerable?
In-depth knowledge about product, i.e. used modules, active software etc.
Impact?
CVSS, CWSS, knowledge about environment, usage of system etc.
no
Configuration 1
yes
Environment A
Result 1
no
Configuration 2
yes
Environment B
Result 2
Public Exploit?
Exploit-db
yes/no

Figure 2: *The process of keeping things up-to-date*

Table 1: *Amount of data being saved by patching compared to sending a full firmware.*

Full Firmware	61 MiB
Patch	0.3 MiB
Difference	60 MiB
Data saved (1M devices)	60 TiB

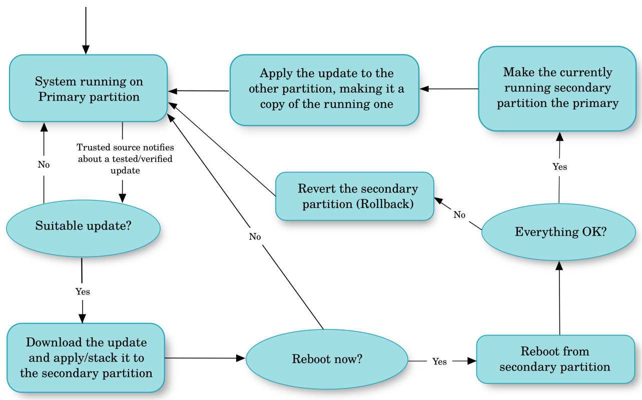
System running on Primary partition
Apply the update to the other partition, making it a copy of the running one
Make the currently running secondary partition the primary
Trusted source notifies about a tested/verified update
No
Yes
Revert the secondary partition (Rollback)
No
Suitable update?
No
Everything OK?
Yes
Download the update and apply/stack it to the secondary partition
Reboot now?
Yes
Reboot from secondary partition

Figure 3: *The process of keeping things up-to-date*