Appendix D:

IoT Vulnerabilities

1 Introduction

There are an estimated 31 billion devices currently in use. As these devices are being adopted in all aspects of life - from home devices, medical devices, industrial devices - there are valid concerns raised about the security capabilities of these devices.

2 IoT Device Resources

Internet of Things devices are typically low-power devices and sensors. These devices are network connected, and often interact with a users personal information. As an affect of the low power resources used on these devices, the devices critical functions take precedence in terms of memory or processing performance. This leaves little availability in terms of available resources for aspects such as security.

3 OS Security Features

Non-resource constrained, "fully fledged" operating systems which run on consumer PCs and enterprise server hardware have a number of security features which can be implemented to protect against attack. These range from firewall and packet filtering software, to firmware verification software.

3.1 Secure Boot

Secure boot is a security mechanism used to verify that software being run on a system is from a trusted source. The vendors whose software has been granted permission to run on the system are stored in firmware[1]. This allows for any software being loaded onto the device at boot time can be verified by comparing the vendor signature against the keys which are stored in firmware.

3.1.1 UEFI Secure Boot

UEFI found the need to implement the secure boot mechanism as protections for post-boot code injection had become competent enough that attackers were beginning to focus their efforts on pre-boot firmware injection. Pre-boot malware injection includes rootkit and bootkits, which, before secure boot, proved extremely difficult to detect.

There are a number of intended results for pre-boot exploits, ranging from control over the operating system, or user identification credential snooping, to control over ROM and PCIe peripherals.

Vendor signatures are stored in databases in the devices firmware. The two types of keys used for accessing these databases are Platform Keys, and Key Exchange Keys[2]. Platform keys are the vendor defined key, which prevents modifications to the Key Exchange Key. The Key Exchange Key prevents modifications to the signature database. This mechanism allows vendors with valid KEK's to modify the signature database, such as inserting or deleting a signature. By comparing each code module's signature to the database at boot time, a decision can be made on whether to load this module and proceed with the boot process, or not to load the module and terminate the boot process.

The device manufacturer controls the signatures which are included from the factory. Vendors such as Microsoft offer programs for vendors to submit modules for authorization, to have their signatures added to the signatures database. This system has been mirrored by a number of Linux distributions.

Secure boot is compatible with the x64 platform, and is enabled by default on Windows and most Linux distributions[3]. By enabling this mechanism by default, the user must disable the feature to open themselves to vulnerability.

3.1.2 Secure Boot in IoT Systems

While IoT devices typically have a one-time setup, they can be rebooted multiple times over there lifetime, especially in the case of smart home IoT devices, as they are relocated or power cycled. By initiating their boot process, these devices become vulnerable to the injection of malicious code modules. There is a recognition for the need of a secure boot type mechanism for Internet of Things devices, with implementations such as MCUboot in development for a number of microcontroller based operating systems such as RIOT and Zephyr[4]. There are, however, a number of limitations which have to be considered, namely the lack of available storage on these contrained devices. As a workaround for this constraint the boot code can be written to non-modifiable ROM, although this could raise issues in the devices future, as patches required to the devices firmware to mitigate other vulnerabilities could not be implemented[5].

3.2 IPtables

4 Conclusion

References

[1] *Secure boot*. [Online]. Available: `https://wiki.ubuntu.com/UEFI/SecureBoot`.

[2] R. Wilkins and B. Richardson, *UEFI Secure Boot in Modern Computer Security Solutions*, [Online]. Available: `https://www.uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf`.

[3] C. Hoffman, *How secure boot works on windows 8 and 10, and what it means for linux*, Jul. 2017. [Online]. Available: `https://www.howtogeek.com/116569/htg-explains-how-windows-8s-secure-boot-feature-works-what-it-means-for-linux/`.

[4] JuulLabs-OSS, *Juullabs-oss/mcuboot*. [Online]. Available: `https://github.com/JuulLabs-OSS/mcuboot`.

[5] I. S. Foundation, *Device secure boot*. [Online]. Available: `https://www.iotsecurityfoundation.org/best-practice-guide-articles/device-secure-boot/`.