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 constrained 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 Device Level Firewalls

Firewalls define rules used for the purpose of allowing or blocking traffic entering a system.

3.3 Linux Iptables and Netfilter

Iptables is a software package for filtering traffic at the Network Layer (layer 3 of the OSI model). This package interacts with the "netfilter" kernel hooks, which are five hooks that can be triggered at different stages of packet processing. They work by applying a set of rules to incoming or outgoing packets. These rules can be pre-applied, i.e. defined by the manufacturer, or user-defined.

Iptables are broken down into a number of layers. The first of these is the table. The table is used to separate rules by function. The default tables are the "filter" table, the "NAT" table, the "mangle" table, the "raw" table, and the "security" table. Tables contain a number of chains, which are associated with the five netfilter hooks, and are triggered when a packet invokes a rule.

The built in iptable chains are triggered by each of the five netfilter hooks as follows:

- NF_IP_PRE_ROUTING -> PREROUTING
- NF_IP_LOCAL_IN -> INPUT
- NF_IP_FORWARD -> FORWARD
- NF_IP_LOCAL_OUT -> OUTPUT
- NF_IP_POST_ROUTING -> POSTROUTING

4 IoT Vulnerabilities

Neshenko et al.[6] define 9 classes of IoT vulnerabilities. These are deficient physical security, insufficient energy harvesting, inadequate authentication, improper encryption, unnecessary open ports, insufficient access control, improper patch management capabilities, weak programming practices, and insufficient audit mechanisms. While a

number of these are not relevant to the susceptibility of these devices to malicious attack for their future use in a large scale botnet, for example physical security - an attack is not going to physically connect their PC to thousands of IoT devices to build their botnet - issues such as improper and encryption authentication methods, open ports, and an inability to update flaws in firmware certainly leave these devices vulnerable to attack on a large scale.

4.0.1 Common Vulnerabilities and Exposures - IoT Logs

CVE, or Common Vulnerabilities and Exposures, is a publicly accessible list of cybersecurity vulnerabilities[7]. This list is comprised of reported vulnerabilities from user and engineers, with details of the vulnerability and level of exposure to the system due to the vulnerability. Each CVE entry contains a CVE ID, details of the vulnerability, and Common Vulnerability Scoring System (CVSS) score. The CVSS score is calculated using a number of metrics, including at its base level the attack vector and complexity, the requirement for user interaction, and the impact of the vulnerability on a users confidentiality or the availability of the service[8]. The score defines a severity rating from Low to High for the listed vulnerability. With the CVSS 3.0 rating system, scores from 0.1 to 3.9 are low severity, 4.0 to 6.9 is medium severity, 7.0 to 8.9 are high severity, and 9.0 to 10 are critical severity.

Listed in Table 1 are the currently listed CVE reports for the IoT operating systems RIOT, Contiki, Amazon FreeRTOS and Zephyr OS.

Table 1: IoT Operating System CVE Logs

Operating System	CVE ID	CVE Score
RIOT-OS	CVE-2019-16754	5.0
	CVE-2019-15702	5.0
	CVE-2019-15134	7.8
Contiki-OS	CVE-2017-7296	4.3
	CVE-2017-7295	7.8
Amazon FreeRTOS	CVE-2018-16603	4.3
	CVE-2018-16602	4.3
	CVE-2018-16601	6.8
	CVE-2018-16600	4.3
	CVE-2018-16599	4.3
	CVE-2018-16598	4.3
	CVE-2018-16527	4.3
	CVE-2018-16526	6.8
	CVE-2018-16525	6.8
	CVE-2018-16524	4.3
	CVE-2018-16523	5.8
ZephyrOS	CVE-2018-1000800	7.5
	CVE-2017-14202	4.6
	CVE-2017-14201	4.6
	CVE-2017-14199	7.5

RIOT OS Vulnerabilities

The listed RIOT OS vulnerabilities range from medium to high severity. The two medium severity vulnerabilities are protocol vulnerabilities. The first of these is an issue with the OS's implementation of the MQTT messaging protocol[9]. Due to the implementation, null pointer dereferencing results in a segmentation fault, creashing the system[10]. The second of these protocol vulnerabilities is a TCP parsing error[11],

whereby the system can be places in an infinite loop while attempting to parse a packet with an unknown option specified, and an option length of zero[12].

The highest severity vulnerability listed for RIOT OS is another protocol vulnerability. This is cited as exposing the devices running this operating system to Denial of Service attacks. An error with the implementation of the TCP handshake, in which a memory leak occurs when an ACK is received by the device as the first packet in the handshake instead of a SYN packet[13]. This memory leak can lead to an exhaustion of memory resources on the device.

Contiki-OS

Both the first and second listed CVE reports for Contiki-OS refer to the same Cross-Site Scripting vulnerability in the mqtt.html configuration page. The first of these reports is a medium severity vulnerability[14], while the second is listed as a high severity vulnerability[15]. This difference in severity is due to the level of detail given in the CVE reports. This vulnerability in the operating systems included MQTT configuration implementation causes a null pointer dereference error, which can be initiated via a HTTP POST request, and can cause the host device to crash, causing a denial of service[16].

Amazon FreeRTOS

ZephyrOS

5 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/.

[6] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019. DOI: 10.1109/comst.2019.2910750.

[7] *Common vulnerabilities and exposures (cve)*. [Online]. Available: https://cve.mitre.org/index.html.

[8] *Common vulnerability scoring system calculator*. [Online]. Available: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator.

[9] *Vulnerability details : Cve-2019-16754*. [Online]. Available: https://www.cvedetails.com/cve/CVE-2019-16754/.

[10] Riot-Os, *Asymcute: Fix null pointer dereference by nmeum · pull request #12293 · riot-os/riot*. [Online]. Available: https://github.com/RIOT-OS/RIOT/pull/12293.

[11] *Vulnerability details : Cve-2019-15702*. [Online]. Available: https://www.cvedetails.com/cve/CVE-2019-15702/.

[12] ——, $Gnrc_tcp : Optionparsingdoesn'tterminateonallinputs, potentialdosissue$#12086$riot-os/riot$. [Online]. Available: https://github.com/RIOT-OS/RIOT/issues/12086.

[13] *Vulnerability details : Cve-2019-15134*. [Online]. Available: https://www.cvedetails.com/cve/CVE-2019-15134/.

[14] *Vulnerability details : Cve-2017-7296*. [Online]. Available: https://www.cvedetails.com/cve/CVE-2017-7296/.

[15] *Vulnerability details : Cve-2017-7295*. [Online]. Available: https://www.cvedetails.com/cve/CVE-2017-7295/.

[16] 262588213843476, *Contiki mqtt and xss*. [Online]. Available: https://gist.github.com/jackmcbride/c9328627f1ee104ce84f3fb7eff42f1e.