Simulation

Introduction

With all the research, development and testing of autonomous technologies, the programmed logic can be exceptionally reliable and safe – many can surpass humans. However, data corruption has the potential to alter this logic and may make the device a hazard while some cases may result in a complete failure of a device. To reduce this risk, there needs to be measures to ensure data integrity.

System files and program files can become corrupted rendering the device unusable or malfunctioning (manifesting as bugs, glitches and crashes). For instance, a critical device such as a smart pacemaker may make incorrect decisions and submit incorrect data/diagnoses to a hospital based on erroneous data or may outright fail; a case of data corruption may be fatal.

A device or a network can be attacked by hackers: malicious intent could include spying, damage or botnets (usually a network of devices that can be controlled for malicious activity). The Defense Advanced Research Projects Agency (DARPA) is a governmental organisation of the USA and had launched a program, High-Assurance Cyber Military System (HACMS), in 2012 – it is a growing concern for governments that the Internet of Things (IoT, the network of devices, vehicles and appliances that are connected to the internet) may be used by cybercriminals and terrorists. When the IoT is compromised, there could be widespread damage and worse, fatalities if critical devices are compromised such as stopping pacemakers.

A device may be compromised by a vulnerability or an intercepted update. When there is new software from the manufacturer or authority, it needs to be delivered to the device. A man-in-the-middle attack consists of intercepting the software between the device and the authority and replacing it with a modified version.

Existing Solutions

Filesystems are structures as to how a computer may store information – it contains folders and files. ZFS is a filesystem that aims to detect data corruption and to heal corrupted data. ZFS employs checksum algorithms to verify the integrity of data on the filesystem and repairs it with redundant information (such as a mirror, essentially a backup, of the corrupted data).

Checksums are mathematical algorithms that outputs a unique number (a hash) based on its input. A change in the input data will change the hash calculated, thus is useful for detecting data corruption and cryptography. Below is a text editor for you to experiment with different inputs and the outputs for the SHA1 checksum algorithm. Text is stored as a number in a computer – each character is represented by a unique number. By editing the text, you are changing the input data that is inputted.

Input	Output (SHA1 algorithm)
Please enter text here.	93A85C932C703D2208587AFC6738BF265E5EF892

Checksums are used to verify the authenticity of data. When delivering data, such as software, the authority could use a trusted third party to provide certificates (digital certificates are based on checksums); these certificates are difficult to forge by a man-in-the-middle attack. Therefore, a smart device like a pacemaker could verify that the software update is from the manufacturer or medical authority which mitigates risk to the patient.

To ensure privacy, data sent between devices and the authority should use encryption which are mathematical methods to scramble information so that only the intended recipient could read it and not any outsiders.

Demonstration

SAFE produced a program to simulate how a smart device could use checksums to ensure data integrity. The program was written in Python (version 3.6.5) which is a programming language.

The device directories (folders) are set up as follows:

STORAGE
- incoming
- outgoing
- sent
- store
- system

The *system* folder will have a file containing its checksum information. The program will periodically calculate the checksum of the *system* folder and compare it to the hash in the file. if they match, the folder is verified to have no data corruption; otherwise, there is data corruption and an alarm is raised.

Any files sent from an authority will enter the *incoming* folder and the program will calculate its checksum and compare with the hash inside its file containing the checksum information. If the files pass the test, they will be moved to the *store* folder. The files inside the *store* folder undergoes the same process as the *system* folder.

When files are created and are to be forwarded to the authority, they are temporarily placed in the *outgoing* folder. The program will calculate the checksums of the files and then store it in a separate verification file. The files and the verification file are then moved to the *sent* folder (for this purpose, the *sent* folder is a placeholder and a representation of data that is sent to the authority).

Conclusion

The program only raises an alarm in the simulation. For real-world devices, the alarm could be a trigger for shutting down the device to prevent further damage and sending a message to the authority that an engineer will need to fix the issue. For critical devices, however, like pacemakers, a redundant system could act as a failover; the main system will shut down while the failover system will resume control of the pacemaker – this allows for uninterrupted service which could save a life.

Upon receiving information that is corrupted, a request that the data be resent will occur. Software updates are resent from the authority and medical data are resent from the pacemaker which also reduces risk and prevents misdiagnosis.

Sun Microsystems. (2007). *ZFS Administration Guide*. [PDF] Santa Clara: Sun Microsystems. Available at: http://www.dubeiko.com/development/FileSystems/ZFS/zfsadmin.pdf, [Retrieved: 28 Aug. 2018].

Richards, R. (2012). High-Assurance Cyber Military Systems (HACMS). [Online] Defense Advanced Research Projects Agency. Available at: https://www.darpa.mil/program/high-assurance-cyber-military-systems, [Retrieved: 28 Aug. 2018].

SSLDRAGON, (2017). *How SSL certificates protect you from man-in-the-middle attacks*. [Online] Available at: https://www.ssldragon.com/blog/how-ssl-certificates-protect-you-from-man-in-the-middle-attacks/, [Retrieved: 28 Aug. 2018].