SECURING THE INTERNET OF THINGS- A CISCO PROPOSED FRAMEWORK

INTERNET OF THINGS AND PROBLEMS IN SECURING IT

The Internet of Things or IoT is an all-encompassing, omnipresent and ever growing network that enables the monitoring, analysis and control of a remote environment (virtual or physical) using data indigenously generated by that environment. There has been a noticeable shift from consumer based IPv4 internet of tablets and computers to the operational technology based IPv6 of machine-to-machine and machine-to-human interactions. There are several reasons that validate this shift, the least of which of being that IPv4 cannot accommodate the sheer volume of devices now connected to the internet. The capability of embedded and distributed intelligence collection is essential to the functioning of the IoT and this makes security of this network of prime importance.
There are several reasons that securing the IoT poses a difficult conundrum. Firstly, scalability is a concern considering that most embedded systems in industrial, data storage and energy applications are remotely based with little to no human accessibility and backup connectivity after installation. Most of these embedded devices have limited resources in terms of memory, computing prowess and power to enable the use of complex cryptographic security. Also, most of these devices have several entities that need different levels of access to the data from these devices. Managing these entities' clearance requirements, responsibilities and liabilities becomes a cumbersome task. Another major reason is that these systems have a lengthy lifetime (typically 40 to 50 years). Cypto algorithms do not share the same lifetime which means the system security will be redundant during the life-cycle of the system itself.

SECURITY REQUIREMENTS AND THREATS

The IoT has a different and unique set of security requirements as opposed to the internet. The different devices need to be authenticated for multiple networks and multiple users with different levels of data clearance. They require strong authentication and data protections especially in healthcare and industrial applications. Updation of security algorithms and evolution in the face of unknown physical and virtual crisis.
Some of the threats that are posed to the IoT can be enumerated as consumer OS worms that jump from ICT to IoT with Os'es like Android, script kiddies that pollute unprotected webcams and home automation systems, organized cyber crime and cyber terrorism.
There are four different particular challenges that need to be tackled while deigning a security framework for the IoT. Firstly the secure authentication and authorization of devices while avoiding security algorithms that occupy too much memory space and computing power and require human intervention. Secondly, most of these systems are scattered across geographical space and time meaning different time zones and lifetimes. Third- all the communication channels that help transmit and exchange data as well as their stacks and protocols need to be protected and secured. And lastly, most of these devices generate a geographical and user-behavior signature that may grievously compromise privacy.

PROPOSED FRAMEWORK

There are four levels to the proposed security framework:

1. Authentication: This level is at the core and provides as well as verifies the device identity information of an IoT entity. This mechanism needs to be of small computive footprint and long lifetime. Some of the proposed solutions are RFID, MAC Address, Shared Secret and Hardware footprints.
2. Authorization: This level build on the core authentication and establishes a trust relation between the device and network for transport. Available solutions map well to the IoT requirement but the scalability factor needs to be addressed.
3. Network Enforced Policy: This includes the security of all elements that are involved in the transmit and exchange of data. Most wireless and wired connections already have established security policies that can be used to great effect within the IoT sector as well.
4. Security Analytics: Visibility and Control – This layer utilizes services of all the network elements to provide telemetry and gaining visibility of the network with the end aim of gaining control and securing the network. It includes threat detection and threat mitigation by shutting attack and proper remediation of affected channels. It also aims to develop defense capabilities using the MPP or massive parallel data base platform for security analysis.

The security implications of an attack on the IoT are vast as well as innately harmful. Securing the IoT will never be a perfected process considering the astounding number of entities involved, it is safe to say that it will be a continually evolving process.

Reference : http://www.cisco.com/c/en/us/about/security-center/secure-iot-proposed-framework.html

OWASP TOP 10 – EXPLORING LACK OF BINARY PROTECTION

BINARY PROTECTION AND ITS USE

Binary protection (hardening) is a method to prevent the tampering of code in mobile applications in order to protect the integrity of the app and the privacy of the users. It analyzes and manipulates binary files of these web apps in order to make it difficult to reverse engineer the app thus providing an additional layer of security to the user and the proprietor. Unfortunately, most app developers avoid or skip this function thus exposing users to a variety of risks.

Binary protection doesn't necessarily render the code tamper proof but it certainly makes it difficult for the hacker to tamper it. Typically, a hacker will analyze the code and then reverse engineer it in order to introduce some hidden functionality or features into the app. This hidden feature may be of a varied number of types and levels of harmfulness. Introduction of some malware is usually the recourse.

OWASP judges the prevalence of this type of attack to be common and detectability to be easy.

Hosting the code in an untrustworthy environment (physically un-monitorable) exposes the code to introduction of unwanted modifications. These modifications are usually done using tools like ClutchMod, dex2jar or GDB. It is also possible to use the pseudocode of this app using Hopper/IDA Pro to introduce malware or extra functionalities to steal data. The presentation layer or interface is also susceptible. Most importantly the binary executable can be modified using a hex editor to bypass all security measures.

IMPLICATIONS OF THE LACK OF BINARY PROTECTION

There are severe implications for the user and the owner as well.

Most times, adversaries will just reverse engineer the code and copy it in order to resell it to another application. This results in theft of intellectual property but the culprit is usually never apprehended. Theft inevitably leads to a revenue loss and reputation loss to the developer.

For the users, in apps containing sensitive personal and financial information, this overlook is a major risk. It may lead to leak of information and further escalate to several other frauds. It also compromises the users experience with the app.

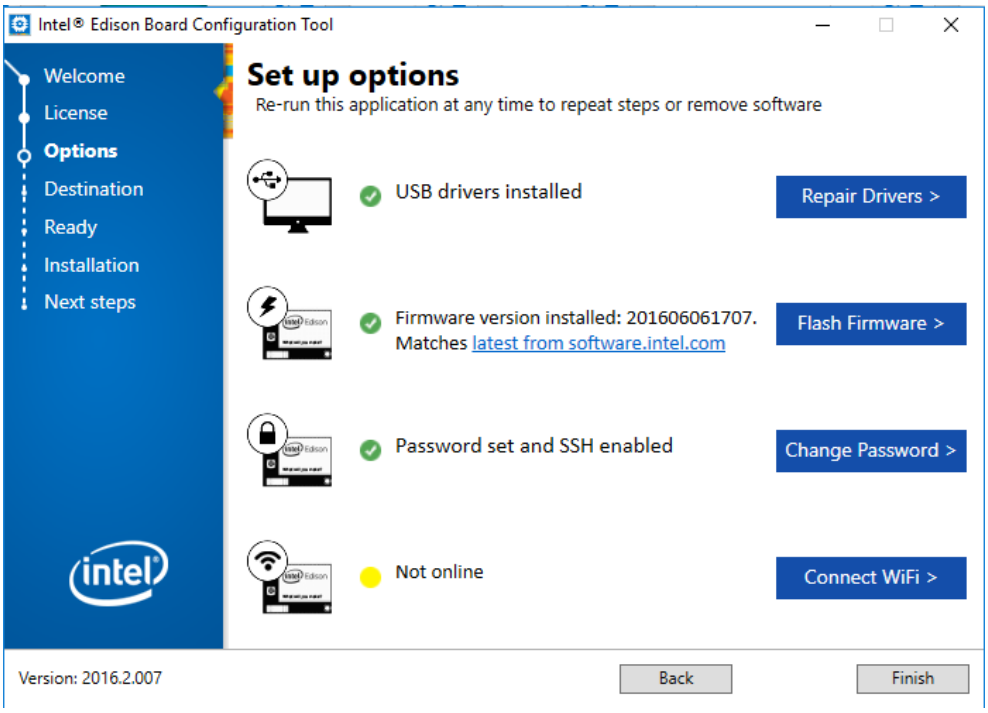
VIABLE SOLUTIONS TO DETECTING AND AVOIDING AN ATTACK ON THE CODE

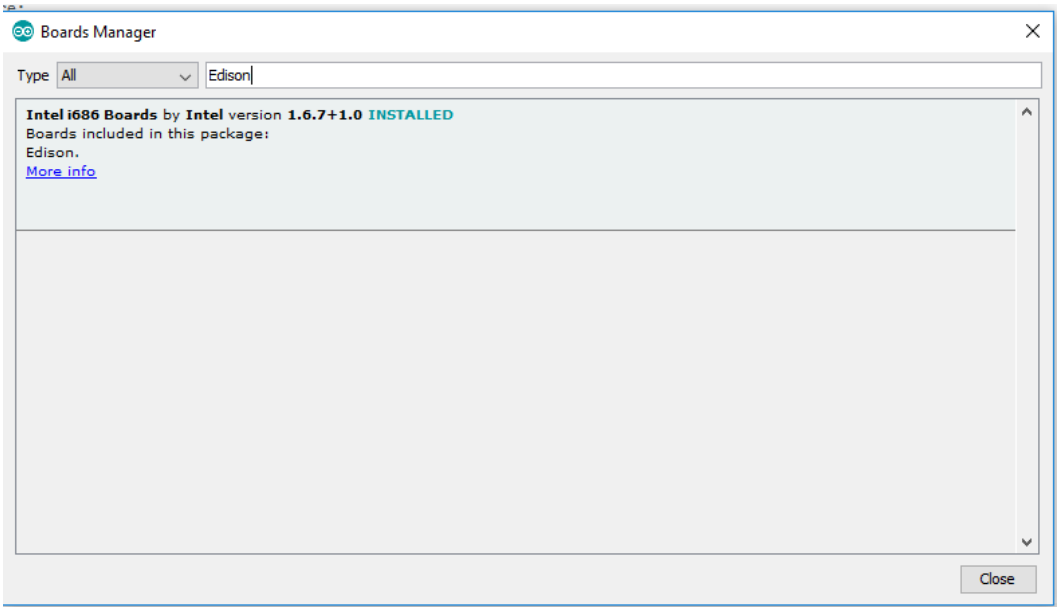
The major hindrance in detection is the fact that most app stores do not have an alarm clause for such occurrences and the developer generally just stumbles onto it when the code shows up in a similar application on the app store - most times by accident.

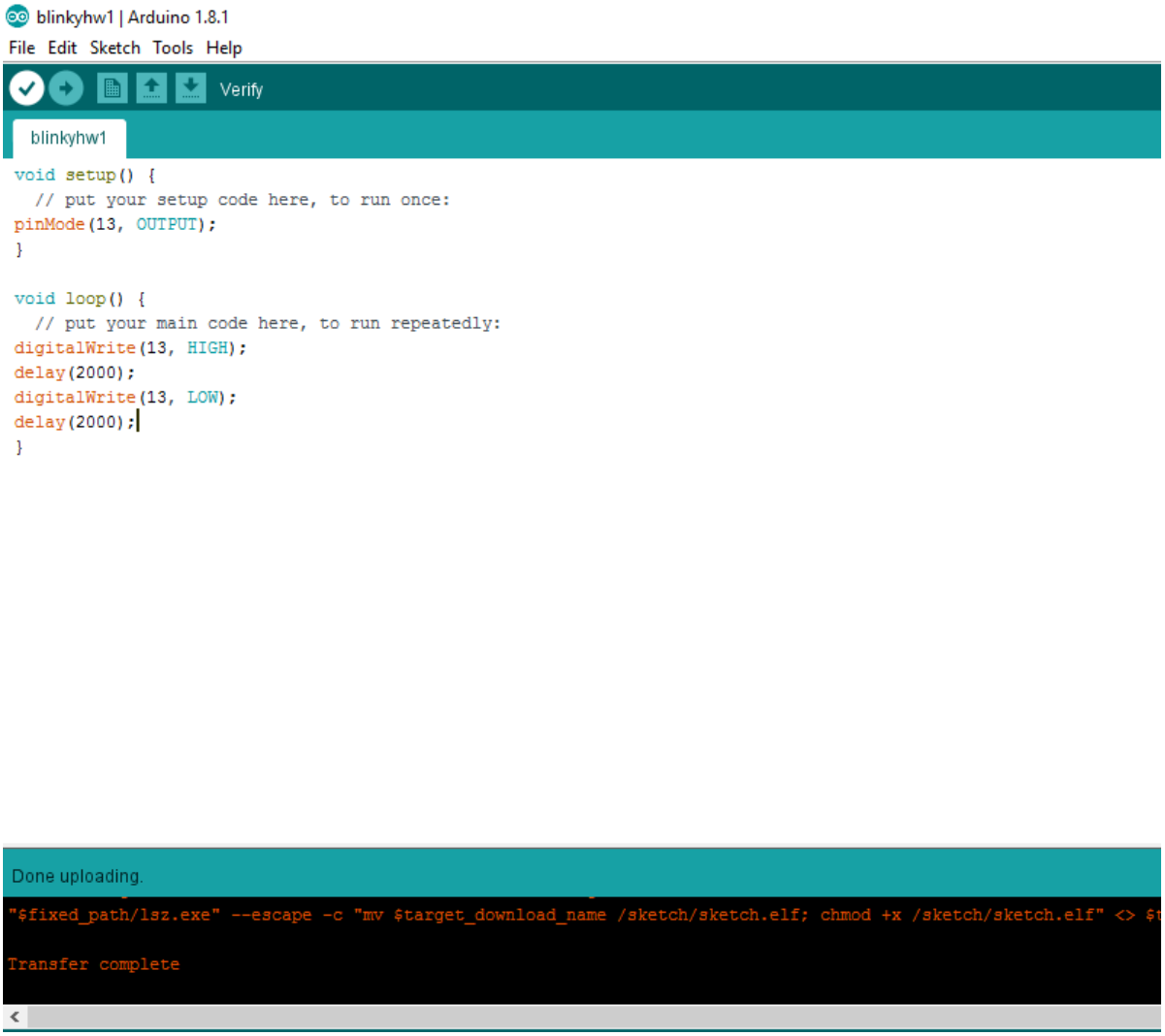
The very obvious solution to prevent these attacks is to follow secure coding techniques like checksums, jailbreak detection, debugger detection and certificate pinning controls.
Rigorous static and dynamic analysis at the developers end can ensure closing all of most vulnerabilities that could lead to reverse engineering of code.
The most viable solution would be to include in the code a run time option of detecting any tampering and a consequent alert system. Code should be able to detect an injection or modification and react in several different ways. Some include a shut down of the app and code as well as a stall alert.
Policing of app stores is another idea that will discourage hackers from attempting to take disadvantage of the lack of binary protection.
It would be good to remember that although binary protection is a good idea, it will never fully insure or protect the code from any modification. It will greatly slow down and complicate the process of modifying code and information like passwords and/or pincodes and credit card data.

Reference: https://www.owasp.org/index.php/Mobile_Top_10_2014-M10

SCREEN SHOTS FOR INSTALLATION AND BLINKY ARDUINO PROGRAM






blinkyhw1 | Arduino 1.8.1
File Edit Sketch Tools Help
Verify
blinkyhw1
void setup() {
// put your setup code here, to run once:
pinMode(13, OUTPUT);
}
void loop() {
// put your main code here, to run repeatedly:
digitalWrite(13, HIGH);
delay(2000);
digitalWrite(13, LOW);
delay(2000);
}
Done uploading.
"$fixed_path/lsz.exe" --escape -c "mv $target_download_name /sketch/sketch.elf; chmod +x /sketch/sketch.elf" <> $t
Transfer complete