Internet of Things Security, a State of the Art

Paul Florence - perso@florencepaul.com
Célia Prat - cprat@etud.insa-toulouse.fr
Benjamin Bigey - bigey@etud.insa-toulouse.fr
Lucien Menassol - menassol@etud.insa-toulouse.fr
Jérôme Miarivelo Mampianinazakason - mampiani@etud.insa-toulouse.fr

***Abstract*—The fast, large-scale deployment of the Internet of Things (IoT) is raising major security concerns. This document aims at presenting an end-to-end state-of-the-art of security in this field, by describing the various security attacks than can be performed on a generic IoT system.**

In this state-of-the-art we link each security exploit with the IoT functionality it targets. To understand the specificities of IoT, we first describe the basic functionnalities found in such systems. In a second phase, we give an overview of the different types of security attacks in computer systems. Finally, we detail which attack can be used to compromise each IoT functionality and how.

I. Introduction

The Internet of Things (IoT) can be defined as the remote interconnection of numerous sensors and objects, through large-scale communication protocols and with few to no human intervention. IoT is providing us with a booming network of smart devices with life-changing applications in sectors such as health-care, life sciences, smart home, municipal infrastructure, agriculture, education [1]... However, the fast and heterogeneous deployment of such systems raises major security concerns, as smart things manufacturers in such a booming market are likely to quickly develop small low-energy devices, at the expense of allocating the necessary time and processing power to address security issues.

In the following we will consider that an IoT system is made of three basic components [1]: a thing, the controller, and the cloud. The thing is connected to the Internet (it could be a smart home system, a smart watch...) while the controller is a program on a device such as a mobile phone or PC. The controller has two ways of communicating with the thing: either through a unique router if both the controller and the thing are on the same local network, or through the cloud if the controller is not on the local network (see picture below). In this case, the thing builds a permanent connection to the cloud while the controller periodically sends or requests information.

In such a system we can consider ten distinct functionalities (as defined in [1]).

Upgrading | The firmware (either an embedded Linux system or a micro-controller) allows for code updating.

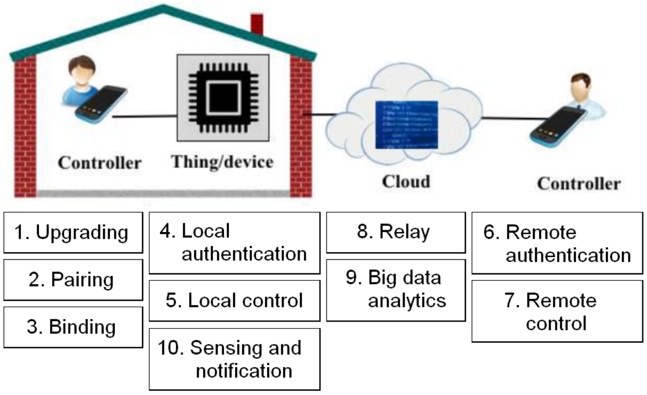

Figure 1. Basic functionalities of an IoT system [1]

Pairing | Pairing is the act of connecting to the thing in order to set up the initial configuration. When first powered on, a smart thing usually behaves as a wireless router which allows the controller to connect. This can be done using protocols such as WiFi, Bluetooth, ZigBee, bar-code/scanner, or NFC.

Binding | The process of configuring the thing's communications through the controller once pairing is done. For instance, the process of connecting the thing to the Internet (by giving it WiFi credentials for example) is binding. Connecting the thing to other devices such as client smart-phones or sensors can also be a required binding.

Local authentication | Within a local network, the controller may be connected to an open port on the thing which is dedicated to a protocol that requires authentication: Secure SHell (SSH) or Radio Frequency IDentification (RFID) for example.

Local control | Once authenticated the controller can send commands through the local network to control the thing.

Remote authentication | If the controller is not in the local network, and has to go through the cloud, it will have to be authenticated on a remote application server which will forward commands to the thing.

Remote control | The thing needs to register to a cloud server in order to allow remote control. The remote application server can send commands to the thing.

Relay | Some application servers and some routers are needed to relay authentication and control messages between all components.

Big Data analysis | Big Data analytic capabilities are deployed on the cloud in order to collect and analyse data from its users.

Sensing and notification | Smart things getting information from the environment (such as temperature) and notifying the user about those parameters or about some behaviours like repeated login attempts.

II. Common exploits types

A. *Physical Attack*

1) Node jamming

Jamming can be viewed as a form of Denial-of-Service attack, whose goal is to prevent users from receiving timely and adequate information. The jamming attacks are mainly classified into active and passive attacks. In active jamming, the jamming node is continuously generating some malicious packets. When the victim receives these packets, a "malicious" code will start running on their machine. The passive jamming is activated only when jamming node detects some event on channel, this attack consists of occupying the communication channel by sending interference at a precise frequency, for example: jam the 2.4 GHz frequency in a way that drops the signal to a level where the wireless network can no longer function [2].

2) Sleep Deprivation Attack

A sensor network is composed of nodes interacting with each other. They are organized in clusters and one of them is the head of this cluster. The main problem in sensor network remains their autonomy. Indeed, these nodes have limited batteries supplies. One of the major challenge in this field is to reduce the node's consumption. In order to save energy, sensors have different modes, including a deep sleep mode. It is a period of time during which the device does not communicate with the network, and switches off the most consuming part of its system (like transceivers for example). Sensors spend most of their time in deep sleep mode [3].

The sleep deprivation attack consists in preventing sensors from entering that mode. How to keep sensors in active mode is not complicated: sensors just have to receive a normal message. To make it look real, attackers have to be a part of the cluster and be the head of it. It is actually possible because clustering algorithms are based on reliability: each node is trusted. Algorithms can hence be easily manipulated.

Once attackers have compromised a node, they just have to enter the cluster and take the head of it. Malicious cluster heads pretend sending normal messages while it actually sends just enough so that the receiving device does not get to enter their low power sleep mode.

As a result, the target will not get the opportunity to save energy. Its consumption is going to raise significantly and hence, its life-time is going to be shortened. Sleep deprivation can be classified as an active attack [4].

3) Physical Damage

When the device is accessible, the attacker can damage components of the system [5]. It can lead to the loss of some features, and sometimes even make the device unavailable, like in a Denial of Service attack. The attacker can also plug an external device into the system. He can access it and do whatever he wants with it.

4) Social Engineering

There is another type of physical attack: when the attacker contacts and manipulates users who own an IoT system. What he can typically do is obtaining information to guess more easily the user's password for example. The user does not always give intentionally these indications. However, we tend to create passwords linked with our lives so that it is easier to remember, and this can be exploited by attackers. Attackers can also pretend they are suppliers, and ask for private information that people wouldn't usually give, but they feel confident to give to their suppliers.

Not only passwords are affected. The attackers can also get sensitive information on the system that can be used for wrong purposes.

B. *Network Attack*

1) Traffic Analysis Attacks

Traffic analysis is the process of intercepting and examining messages in order to deduce information from patterns in communication. It can be performed even when the messages are encrypted and cannot be decrypted. It leads to a "degradation of network performance, it increases packet collision, increases contention and creates traffic distortion" [6].

2) Spoofing

Spoofing is a malicious practice in which communication is sent from an unknown source disguised as a known source to the receiver. Spoofing can take on many forms: IP, ARP, DNS, email.

IP Spoofing: The attacker sends packets to the victim using the address of a trusted machine as source address. He has to neutralize the trusted machine so that this machine is unable to send packets to the victim. This can be done by SYN flooding. Then, the attacker have to guess the correct sequence number, and try to communicate with the victim. The only major complication in this attack remain in the fact that, if it works, the victim will send its answers to the source address used, so to the trusted machine and not to the attacker. This is a blind attack [7].

ARP spoofing: The Address Resolution Protocol (ARP) enables to find the MAC address corresponding to a given

IP address. A partial mapping called the ARP cache, is kept on every router and machine. The main purpose of ARP spoofing is to corrupt this ARP cache, by associating the attacker's MAC address to the IP address of the machine he wants to spy on. Hence, by this process, all the packets destined to the victim will be received by the attacker. But how to corrupt the cache? This is simple. When a router gets a packet with IP X, and does not have it in his cache, ARP broadcasts on the network a request to know the MAC address of the machine with IP X. Only the machine who has IP X answer to the router with its MAC address. This is where attackers can interfere. They send forged ARP replies to the router indicating they have IP X with their own MAC address. The router will update his table with the wrong association.

DNS spoofing: When users want to visit a website, a Domain Name System (DNS) request is made to convert the domain name they enter into the IP address of the website. Attackers can hack DNS servers to modify the IP address of a domain so that when users think they are connecting to their bank website for example, they are connecting to a fake site. So that users believe in it, the graphical interface has to be similar. Attackers can then get sensible information.

Email spoofing: It is when the email seems to come from a legitimate source while it actually is from an impostor. It makes people more likely to open emails that can be meant for spreading viruses [7].

3) Man in the Middle

In most cases this attack requires three players at least. There is the victim, the entity with which the victim is trying to communicate, and the "Man in the Middle" (MITM) system, who is intercepting the victim's communications. The attacker can access the communication media between the two end points [8], and optionally modify their messages.

MITM attacks can occur on different layers of the OSI Model, from the Data Link layer to the Application one, and even in cellular networks [8]. It is based on different approaches depending on the layer: IP spoofing for example for the network and transport layer, interception of the encryption keys through the analysis of the network communications [8].

MITM attacks compromise all three parameters of the CIA triad: **confidentiality** in the way that the attacker accesses the communication, **integrity** by being able to modify messages, and **availability** by having the opportunity to delete part of the exchange [8].

4) Denial of Service (DoS)

DoS attacks can be carried out by network insiders and outsiders. Their goal is to make the network unavailable to authenticate users by flooding and jamming with likely catastrophic results. By flooding the control channel with high volumes of artificially generated messages, the network's nodes, on-board units and roadside units cannot sufficiently process the surplus data. Hence, devices become unavailable. DOS can affect physical layer, link layer,network layer, transport layer and application layer.

Distributed Denial of Service (DDoS) happens when attackers compromise a lot of vulnerable devices to lead a synchronized attack on a victim [6].

C. Software Attack

1) Malicious software

These malwares have a behaviour which allows the attacker to endanger the three points of the CIA triad, he may get privileges on a system to modify the way it works, to access, alter or destroy information. There are many types of malware: a **virus** is a self-replicating software that infect a computer by inserting its own code in legitimate software. A **worm** is also a self-replicating malicious software but it does not need host software. A **Trojan horse** is a malicious code that hides itself within seemingly harmless programs. It does not replicate itself but it can collect information or create back-doors.

2) Code injection

The aim of this attack is to inject a code in a vulnerable application in order to change the course of its execution. There are different types of code injection attacks:

- **SQL injection:** it is used to attack data driven application thanks to an injection of a SQL statement in an entry field which allow the attacker to steal, modify or destroy data.
- **Script injection:** it consists in injecting malicious code. One of the most known script injection attack is Cross Site Scripting (XSS). It allows the attacker to inject client-side scripts into web pages viewed by other users. For instance, the script can steal cookies or redirect the user to another web page.
- **Command injection** (or shell injection): it allows an attacker to execute an arbitrary command on the host operating system via a vulnerable application.

III. Exploits on IoT functionalities

Since we are presenting this document as a joined work with a second team, we will only go through functionnalities 1, 2, 4, 5 and 9. You can find the other functionalities detailed in [9].

A. Upgrading

Applying updates and patches is crucial for connected things in order to improve security [10], but it is also a major attack vector. For example, *HP LaserJet printers* are vulnerable during the upgrading process, such as described in [11].

Upgrading attacks are particularly powerful because the device is in a state that allows the attacker to potentially upload a modified firmware. They are usually performed after successfully exploiting an upgrading protocol.

TFTP Man in The Middle | Most devices uses Trivial File Transfer Protocol (TFTP) to download updated firmware. This is a protocol that is vulnerable to Man-In-The-Middle (MITM) attacks, and thus the downloaded binary can be compromised by an attacker. This was experimented in [12] where the authors used ARP poisoning to target a Smart Hub device and modify its firmware.

Bluetooth Man in The Middle | Bluetooth is often used as a way to deliver updates to the device, through a mobile phone that behaves as a relay to the Internet. However it is also possible to perform MITM attacks on the Bluetooth stack by hijacking the Bluetooth pairing process through a vulnerability in the secure Secure Simple Pairing (SSP) protocol when the connection uses Just Work Association Model [13]. The writers of [14] have described various MITM attacks on one of the latest version of the protocol[1] using vulnerabilities in SSP such as impersonating an user and purposely choosing a weak security association. The security properties of SSP Association Models for Bluetooth and Bluetooth Low Energy (BLE) are formally analysed by the author of [15] which highlights issues related to the use of a passkey and a cryptographic public key exchange in the SSP security model.

The BLE stack is also exposed to quite a few vulnerabilities. Even if it uses more secure SSP options, it is still sensible to weak Random Number Generator (RNG) strength [16]. Prior to version 4.2 BLE pairing frames were not encrypted, which made the Long Term Key (LTK) easily guessable by an attacker [17]. The authors of [18] have listed some more Bluetooth exploits involved in MITM attacks. Finally, the authors of [19] take a look at vulnerabilities from unsafe software implementations of the protocol which allow, for instance, an attacker to perform a buffer overflow of unlimited size on the Linux kernel stack.

WiFi Man in the Middle | It is possible to perform MITM on the physical layer using properties of WiFi's channels [20]. This allows the attacker to inject malicious packets, and to perform a Key Re-installation Attack (KRACK) by compromising the 4-way handshake used to provide reciprocal authentication and session key agreement on the Wifi Protected Access II (WPA2) protocol. After the KRACK, the attacker is able to use the session key to decrypt WPA2 packets. If successful, the attacker can eavesdrop on all the connection established by the thing, and inject forged packets. This attack could be used to compromise a higher level protocol, such as TCP or HTTP used to carry the upgrading data. It is also possible to compromise NTP, which in turn compromises TLS certificates that could have been used to prevent MITM [21].

Encryption vulnerability | Some IoT suppliers might use encrypted connection to mitigate the risk of MITM such as advised in [7]. Nonetheless the secret key is often extractable by an attacker [12] allowing him to spoof the supplier identity during the upgrade process.

Secure boot compromising | On Linux systems, manufacturers might set up a Secure Boot to make sure that they do not run a compromised firmware. Secure Boot is used to ensure that only trusted code runs on the device. A chain of trust is created, where the roots are cryptographic keys programmed on one time fuses. They are then used to verify the BIOS code, which will then verify the next link in the chain. However, BIOSes, bootloaders and operating systems can still be vulnerable in case of an unsafe implementation. For example the Universal Boot Loader (U-Boot) enables cryptographic verification of signed kernel images. Yet it was vulnerable to a buffer overflow due to insufficient boundary check during the file system image load or in the case of a network image boot. This exploit allowed the attacker to run untrusted code [22][2] [23][3].

B. Pairing

An IoT object usually bootstraps in two steps: pairing and then binding [1]. An external controller like a smart-phone usually configures these devices when they power on for the first time and so the pairing process is an immediate attack vector.

A pairing attack is interesting for an attacker because a thing is initially likely to easily trust anyone since it is not yet configured, and never-used objects may be massively compromised. A common strategy for an attacker is to force the IoT device to re-pair with a malicious device, allowing to perform a MITM attack.

Physical access re-pairing | An attacker having physical access to the device can alter the configuration settings [12], which could include issuing a new device pairing request.

Bluetooth re-pairing attack | In the case of a Bluetooth communication, the re-pairing attack is done by spoofing the MAC address of any of the already paired devices. The next time the victim tries to communicate, the attacked thing whose MAC address is not spoofed will be forced to re-pair with the attacker device [18].

Old Bluetooth versions vulnerabilities | The communications' security between devices is only as secured as the oldest Bluetooth version between the connected things. This forced security downgrade is today a major vulnerability because many older devices are still being used and weaknesses from older Bluetooth revisions are still exploitable [24]. For instance, versions before *Bluetooth 1.2* used static unit keys for the pairing process which can

[1] Bluetooth v4.0

[2] Common Vulnerability Exposure n°18440

[3] Common Vulnerability Exposure n°18439

be re-used. An attacker can retrieve the key to eavesdrop on the original devices.

BLE pairing attacks | IoT devices that use BLE to pair should calculate the Long Term Keys (LTK) in order to make sure that any communication happening in the future between the two pairing devices is secure. However, mobile applications that work with these devices often do not verify the MAC address of the real thing and the device performs a new pairing without requiring any action ("just works"). The end user only feels a service or product disruption and the attacker performs a MITM attack when the user restarts BLE pairing [25].

Z-Wave Downgrade Attack | 2,400 vendors use the `Z-Wave` wireless protocol embedded in an estimated 100 million smart-home devices. Today's Z-Wave systems are configured to support a strong S2 Z-Wave pairing security process [25]. Yet a Proof of Concept (PoC) attack demonstrates how an attacker could select the S0 pairing which is a security breach whither obtaining easily the network key is a well-known issue [26]. Even in a S2 Z-Wave pairing, an attacker within RF range is able to steal the encryption key when a controller pairs with the IoT device and so compromises it [27].

Network infrastructures weaknesses | The controller can communicate with the thing through several communication channels such as WiFi, Bluetooth, ZigBee or barcode/scanner. The hardware and network infrastructure used induce vulnerabilities developed in this paper [28].

C. Local authentication

The authentication process aims at identifying an authorized user or an authorized controller device to allow control over the system. Identifying an user most often requires him to input credentials (username/password), while authenticating a known controller device can be an automated process using protocols such as Radio Frequency Identification (RFID) or using its MAC address as an identifier. This process is considered compromised when an unauthorized user is able to control the thing, or if an authorized user is prevented from controlling his thing. Let us see which security attacks can compromise those processes when controlling the thing locally.

RFID Spoofing | The RFID technology uses radio frequencies to transmit data through wireless communication. Each device is attributed a tag, in the form of a microchip, which provides them with an unique identifier. Tags can either be passive or active, with the communication always being initiated by an active tag (reader). Passive tags are not battery powered but can use the energy from the request signal to communicate their identifier to the RFID reader. In the majority of RFID systems, there is no proper authentication process [10]. This means that tags can be accessed by anyone, therefore it is possible for an attacker to spoof the controller's RFID tag and pretend to be the authorized device to gain control [10].

MAC Spoofing | When a thing recognizes its controller device thanks to its MAC address, it is easy to spoof its identity. The MAC address of any device being public information available on the local network, an attacker device can use the controller's MAC address as its own and easily gain control over the thing.

SSH/Telnet dictionary attack | Some devices implement access protocols such as Telnet or SSH. If they are listening on the default port (respectively 23 and 22) they can be vulnerable to a dictionary attack if default credentials or unsafe credentials are used [1]. This connection can be accessed on the local network but also for a remote connection if the firewall allows it.

Buffer Overflow | Buffer overflow attacks can allow bypassing the authentication process. For instance, the Home Network Administration Protocol (HNAP) used in D-Link networking equipments (including connected home outlets) has shown to be vulnerable to a buffer overflow attack allowing to run any command on the exploited device without going through the authentication step, including starting up a telnet server to get a root shell [29]. This exploit uses one of the buffer overflow vulnerabilities of the widely used `libc` library.

Bluetooth PIN brute-force | If the device requires a PIN code to authenticate a user in Bluetooth pairing, it can be vulnerable to a brute-force attack, given that the PIN code is relatively short (usually 4 digits are used, meaning there are almost 10,000 possible combinations) [16] [18].

RFID Denial of Service | It is possible for an attacker to send noise signals over the radio frequencies used in RFID communication. Those signals will interfere and therefore alter the RFID request or response, making the communication establishment impossible [10]. Indeed, any DoS attack preventing the thing to operate compromises the ability of an user to be authenticated (battery exhaustion, network queries…).

D. Local control

Local control implies proximity between the attacker and the compromised thing. Short range communication technologies such as Bluetooth, WiFi or ZigBee, represent significant attack vectors for operating control on the thing. But while taking local control usually occurs after compromising the local authentication, some vulnerabilities relate to a situation where the thing is controlled directly, without authentication, for example by injecting keystrokes through unencrypted wireless mouse communication [30].

Physical attack | Sometimes local control on the thing can be gained through physical interaction with the thing itself. Attacking the simple electrical or mechanical parts can often allow to bypass the security provided by the "smart" aspect of the thing. It can be with something as simple as a powerful magnet, in this case to bypass a smart electricity meter cap by forcing the internal relay to close, even when the electricity provider remotely disabled electricity delivery [31].

Backdoor planting and access | After compromising authentication, one of the possible strategies for the attacker is to plant a backdoor onto the thing to allow its access and control in the future, even if he loses the compromised authentication to the thing. There has also been records of attacker using already present backdoors installed by a third-party, such as the manufacturers of smart things for debugging. A large scale automated search for backdoors in IoT device revealed that between 0.9 and 2.1% of the devices already contains one [32].

Attack self-duplication | Since it is frequent to have IoT devices of the same type communicating through their own local network, the attacker will also often try to use the compromised thing as a new attack vector to extend the reach of its attack. A group of researchers showed that with IoT, this attack could take a severe scale by simulating an attack on smart public lighting through a Zigbee vulnerability, showing that the propagation would be total for a city such as Paris [33].

Data stealing and tampering | This attack aims to steal sensitive data stored in the thing without the user noticing. BlueBugging for example is an attack over Bluetooth targeted primarily at smartphones that allows among others the attacker to steal messages and contact information, without the owner noticing, or even edit or delete this data[34]. Things reacting to our behaviour and habits also offer vectors for private data to leak, like vulnerabilities in smart lights that react dynamically to an audio or video feed and can expose it [35].

Covert information channel | Diverting the use of the thing can allow the creation of a covert channel, to secretly leak information from a compromised thing. In the case of smart light-bulbs, it is possible to leak information through very discreet or unexpected means like infrared emission, thus in this case bypassing the protection of an air-gapped system [35].

Voice commands hijacking | Voice-controlled devices are particularly prone to local control attacks, as the authentication required for a voice command is often fallible or inexistent. It is possible to trigger unintended behaviour from a smart assistant by voice replay attack, or crafting a digital audio signal that will be inaudible to humans but will be picked up by the thing and trigger correctly the voice recognition to execute a malicious command [36].

Critical systems control | While the attacks presented are already harmful to the thing owner, another problematic raised by the evermore integrated aspect of IoT is the danger arisen by the control of critical systems such as smart cars. In this case exploiting the numerous new communication channels offering possible attack vectors (car to phone, to road infrastructure, to smart home, to manufacturer...) could have dramatic consequences in an unsecured context [37].

E. Big data analysis

In the IoT era, some devices are created to store our data for a personalized service involving data analysis. From smart watches to smart phones or even smart scales, all those devices do not have enough processing power to handle large data. This is why all this data is sent through Internet to a cloud platform. The machine to machine communication, which includes traffic generated by IoT devices, is expected to reach 67 percent of all Internet traffic [38]. This traffic is due to the transit of users' data to the cloud platform data-set. This kind of communication makes the IoT system vulnerable to basic network and Machine Learning (ML) threats.

Acquiring data means that all those devices have sensors. Integrity of the data is a capital challenge [39]. Altering this data, for example by intercepting it with a MITM, can lead to a service disruption. For instance, a service that relies on users' data-set to provide information upon analysis can provide false information if the data-set is significantly compromised. We will illustrate those vulnerabilities by the example of a connected watch.

After capturing data, it is transferred to the Cloud. This data feed is transiting through the IoT network, and can be intercepted. In most networks a routing algorithm decides the path taken by data.

Wormhole Attack | When an IoT device wants to send its collected information to the cloud platform, it tries to find a path. That means that it attempts to find the next node to reach based on a factor given by the network, called a metric. On many networks with Routing Information Protocol (RIP), the metric is the count of hops to reach a network. The wormhole attack uses routing algorithm in the IoT nodes network at its advantage and makes all packets transiting by the wormhole [40]. When the wormhole node receives all the data, it can drop it or even alter and send it to the cloud [40].

Data security breach | As data is sent to the cloud platform, it is necessary to ensure data confidentiality and client privacy, through at least encryption [41]. For instance, a connected watch sends to the cloud its geographic position and velocity. This information can be anonymous but still is sensitive data. If an attacker gets this information on your running habits for example, he knows where you have been, and how many times.

Denial of Service | Attacking directly the cloud platform can disrupt the IoT services. Users' devices store their data into the cloud, and sometimes are expecting some. For example, your watch sends your location and velocity to a cloud platform and gets the distance and time to your house. If the cloud platform does not answer, the service is disrupted. The cloud platform can be forced to a Denial of Service state by several attacks, for instance SQL injection if it is unsecure enough, buffer overflow, or any threats that can prevent it from answering [40].

ML mechanics in IoT can be split into three parts:

- The data-set: users provide data to contribute to the data-set.
- The trainer: the data-set is fed to a trainer, which uses some ML architectures to build a specific ML model.
- The resulting ML model: it is built to do a specific task. We can add an API to this ML model to give at a client the opportunity to make queries and get some responses back, which is called ML inference.

Model poisoning | By inserting non-reliable data into the data-set, the attacker can alter the resulting model. This already happened, when in 2016, Microsoft released a chatbot on Twitter which was supposed to learn from interacting with other Twitter users, through ML. After some days of interactions with up to no good users, the chatbot started to tweet racist and sexist statements.

Compromised trainer | The analysis uses third-party libraries to process the data-set into the ML model, for instance TensorFlow library[4] available for Python, JavaScript, C++ or even Java. But if those libraries are jeopardized, then all results extracted from it cannot be trusted [42]. Furthermore, this vulnerability can lead to data-set privacy violation. As the library used by the trainer can access data, the attacker can also access this knowledge.

User profiling | Client can use the API to make requests and get information or details from the ML inference. With all those requests, made intentionally or not, an analyst attacker can create a profile associated to this user. By storing those queries, he can track the client interests or activities [43].

Model evasion | Machine learning classifier is a specific way of using ML models. For example, we can use it to split up malicious or benign inputs. As mentioned earlier, to create a ML model, we use a data-set as a base for our knowledge. This attack consists of making intelligent queries on the API, and get some information on the data used to create our ML model [44].

Model inversion | This attack is close to **Model evasion** but is going further on the reward. Let's take a look at a recognition device which just answer if an incoming user is authorized or not. With the same method as model evasion, making intelligent requests and changing a little our input to fool our ML model, an attacker can disguise malicious inputs to a legitimate one [45].

IV. Conclusion

We have seen that, if unsafely manufactured, the Internet of Things can become an open gate to security attacks, especially considering the amount of sensitive data it carries. Computer systems can be exploited in many different ways, and a such a wide remote-operating network is only as secure as its weakest component. It is primordial that manufacturers are provided with security guidelines and best practices that will enable them to build up appropriate security plans for each device available on the market but also for their cloud services.

[4] https://www.tensorflow.org/

References

[1] Z. Ling, K. Liu, Y. Xu, C. Gao, Y. Jin, C. Zou, X. Fu, and W. Zhao, "IoT Security: An End-to-End View and Case Study," *arXiv:1805.05853 [cs]*, May 2018.

[2] S. Sowmya and P. D. S. K. Malarchelvi, "A survey of jamming attack prevention techniques in wireless networks," in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, 2014, pp. 1–4.

[3] M. Pirretti, S. Zhu, N. Vijaykrishnan, P. McDaniel, M. Kandemir, and R. Brooks, "The Sleep Deprivation Attack in Sensor Networks: Analysis and Methods of Defense," *International Journal of Distributed Sensor Networks*, vol. 2, no. 3, pp. 267–287, Jul. 2006.

[4] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *Proceedings of the 7th annual international conference on Mobile computing and networking - MobiCom '01*, 2001, pp. 272–287.

[5] J. Deogirikar and A. Vidhate, "Security attacks in IoT: A survey," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017, pp. 32–37.

[6] D. D. Virmani, A. Soni, S. Chandel, and M. Hemrajani, "Routing Attacks in Wireless Sensor Networks: A Survey," 2014.

[7] A. Mosenia and N. K. Jha, "A Comprehensive Study of Security of Internet-of-Things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586–602, Oct. 2017.

[8] M. Conti, N. Dragoni, and V. Lesyk, "A Survey of Man In The Middle Attacks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[9] E. Soussi, A. Tanem, Y. Anser, L. Rouch, and E. Tissot, "Safety evaluation of connected objects." INSA, Feb-2019.

[10] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of Things: Security vulnerabilities and challenges." IEE, 2015.

[11] A. Cui, M. Costello, and S. J. Stolfo, "When Firmware Modifications Attack: A Case Study of Embedded Exploitation," p. 13, 2013.

[12] M. B. Barcena and C. Wueest, "Insecurity in the Internet of Things," *Security Response, Symantec*, p. 20, 2015.

[13] M. Herfurt and C. Mulliner, "Bluetooth Security Vulnerabilites and Bluetooth project." 2005.

[14] S. Sandhya and K. A. S. Devi, “Contention for Man-in-the-Middle Attacks in Bluetooth Networks,” in *2012 Fourth International Conference on Computational Intelligence and Communication Networks*, 2012, pp. 700–703.

[15] M. Patrick and C.-W. P. Raphael, “Analyzing the Secure Simple Pairing in Bluetooth v4.0,” *Wireless Personal Communications*, 2012.

[16] P. Stirparo, J. Loeschner, and M. Cattani, “Bluetooth technology: Security features, vulnerabilities and attacks,” *JRC Scientific and Technical Reports*, p. 27, 2011.

[17] Z. Wondimu K., “Exploiting Bluetooth Low Energy Pairing Vulnerability in Telemedicine,” Morgan State University, 2015.

[18] S. S. Hassan, S. D. Bibon, M. S. Hossain, and M. Atiquzzaman, “Security threats in Bluetooth technology,” *Computers & Security*, vol. 74, pp. 308–322, May 2018.

[19] S. Ben and V. Gregory, “BlueBorne Technical White Paper,” *BlueBorne*, p. 41, 2017.

[20] G. door Mathy, “Advanced WiFi Attacks Using Commodity Hardware,” *Mathy Vanhoef, PhD*. Oct-2015.

[21] M. Vanhoef and F. Piessens, “Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS ’17*, 2017, pp. 1313–1328.

[22] “CVE-2018-18440,” *CVE Search*. Nov-2018.

[23] “CVE-2018-18439,” *CVE Search*. Nov-2018.

[24] A. Lonzetta, P. Cope, J. Campbell, B. Mohd, and T. Hayajneh, “Security Vulnerabilities in Bluetooth Technology as Used in IoT,” *Journal of Sensor and Actuator Networks*, vol. 7, no. 3, pp. 7–8, Jul. 2018.

[25] Q. GROULARD and Y. SMAL, “Internet of Things security and privacy,” PhD thesis, Université Catholique de Louvain, Louvain, 2017.

[26] L. Rouch, J. François, F. Beck, and A. Lahmadi, “A Universal Controller to Take Over a Z-Wave Network,” pp. 4–5.

[27] B. Fouladi and S. Ghanoun, “Security Evaluation of the Z-Wave Wireless Protocol,” pp. 3–4.

[28] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, “Internet of Things security: A survey,” *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.

[29] “Hacking the D-Link DSP-W215 Smart Plug,” */dev/ttyS0*. 2014.

[30] M. Newlin and Bastille Threat Research Team, “Mousejack Whitepaper - Injecting Keystrokes into Wireless Mice,” 2016.

[31] Samuel Demeulemeester, “On a hacké Linky ! Et sans outils high-tech...” *Canard PC Hardware*, no. 36, 2018.

[32] John Toterhi, “Hunting for Backdoors in IoT Firmware at Unprecedented Scale.” 2018.

[33] Eyal Ronen, Colin O’Flynn, Adi Shamir, and Achi-Or Weingarten, “IoT Goes Nuclear: Creating a ZigBee Chain Reaction.”

[34] Nateq Be-Nazir Ibn Minar and Mohammed Tarique, “Bluetooth Security Threats and Solution: A Survey,” *IJDPS*, Jan. 2012.

[35] A. Maiti and M. Jadliwala, “Light Ears: Information Leakage via Smart Lights,” *arXiv:1808.07814 [cs]*, Aug. 2018.

[36] Y. Gong and C. Poellabauer, “An Overview of Vulnerabilities of Voice Controlled Systems,” *arXiv:1803.09156 [cs]*, Mar. 2018.

[37] Tamás Bécsi, Szilárd Aradi, and Péter Gáspár, “Security issues and vulnerabilities in connected car systems,” 2015.

[38] U. Ahsan and A. Bais, “A Review on Big Data Analysis and Internet of Things,” in *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2016, pp. 325–330.

[39] N. Haron, J. Jaafar, I. A. Aziz, M. H. Hassan, and M. I. Shapiai, “Data trustworthiness in Internet of Things: A taxonomy and future directions,” in *Big Data and Analytics (ICBDA), 2017 IEEE Conference on*, 2017, vols. 2018 -, pp. 25–30.

[40] P. P. Lokulwar and H. R. Deshmukh, “Threat analysis and attacks modelling in routing towards IoT,” in *I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017 International Conference on*, 2017, pp. 721–726.

[41] M. Abdur, S. Habib, M. Ali, and S. Ullah, “Security Issues in the Internet of Things (IoT): A Comprehensive Study,” *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, 2017.

[42] C. Song, T. Ristenpart, and V. Shmatikov, “Machine Learning Models that Remember Too Much,” *arXiv:1709.07886 [cs]*, Sep. 2017.

[43] E. Malmi and I. Weber, “You Are What Apps You Use: Demographic Prediction Based on User’s Apps,” *arXiv:1603.00059 [cs]*, Feb. 2016.

[44] H. Dang, Y. Huang, and E.-C. Chang, “Evading Classifiers by Morphing in the Dark,” *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS ’17*, pp. 119–133, 2017.

[45] M. Fredrikson, S. Jha, and T. Ristenpart, “Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures,” in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.