Why IoT security is failing. The Need of a Test Driven Security Approach

Cristian Săndescu, Octavian Grigorescu, Răzvan Rughiniș,
Răzvan Deaconescu, Mihnea Călin
Computer Science Department
University Politehnica of Bucharest
Bucharest, Romania
octavian.grigorescu@codaintelligence.com, cristian.sandescu@codaintelligence.com, razvan.rughinis@cs.pub.ro, razvan.deaconescu@cs.pub.ro

***Abstract*— Internet of Things has evolved very quickly in recent years, being used by more and more people in many industries such as medical, automotive, energy, etc. This ecosystem brought with it multiple security issues that led to a significant growth of the daily challenges for cyber security professionals, and to novel types of cyber-attacks worldwide. In this paper, we discuss the existing vulnerabilities and attacks in the context of Internet of Things, followed by a proposal of a monitoring and security testing framework that analyzes and evaluates a complex IoT application from the incipient phase throughout its developing process.**

Keywords—IoT; networking; cyber-security; vulnerabilities;

I. INTRODUCTION

Internet of Things is a concept advanced in 1999 by Kevin Ashton. IoT is a growing Internet-based architecture where exchanges of services and assets take place, and it has a strong effect on the security and privacy of the engaged stakeholders [1]. The IoT landscape is intrinsically defined by a series of technological and market-related challenges and limitations. The security was a problem from the beginning. IoT looms over the industry in the last years alongside the security issues that have not been addressed convincingly so the consumers' and businesses' confidence has decreased as time went on.

Due to costs and physical operating conditions, a lot of IoT devices such as sensors from agriculture or industry are designed to have small dimensions and weight, so the storage and processing capacities are reduced compared to PCs and even smartphones. Real-time programs must ensure that response within particularized time constraints and computing has to be finished in a time frame. Several IoT devices do not have any source of energy except for a battery, thus some limitations on the resources and energy consumption are imposed. Many IoT systems do not implement the necessary protection mechanisms.

Many types of IoT devices are built for very specific tasks and are strongly coupled with the physical layer of communication. Thus, there are multiple types of processors, memories, buses and other hardware segments which make the IoT landscape extremely heterogeneous. Lots of devices do not have a MMU (Memory Management Unit), so several 90s' standard hardware security measures such as memory isolation can not be enforced on these type of devices. Cryptographic algorithms are CPU intensive and consume many resources, energy, and introduce delays in the IoT-enabled workloads that seem unacceptable, especially on Real Time Devices. Thus, many IoT devices communicate in plaintext or by using weak forms of encryption thus being susceptible to MiTM attacks (Man-in-The-Middle).

Last but not least, the IoT industry is in an early development stage which means focus is mostly placed on functionality rather than security. IoT projects stress engineering teams with complex integration issues as most projects involve at least: industry-specific hardware, communication protocols, IP networking, central processing workloads (datacenter or cloud-enabled), web applications and data repositories. The security of each component is important, however the overall security of the integrated system is totally different than the sum of its components. Let's take Mirai phenomenon [2] as an example. This family of malware (Mirai, Satori, Okiru, etc.) exploits well-known common weaknesses and vulnerabilities in IoT devices which may seem almost benign if analysed individually, and transforms them into massive worldwide DDoS cyber-attack weapons against any potential target on the Internet. It is thus crucial to identify and define the relevant threats for a certain infrastructure not only by looking at individual components, but also by looking at the system as a whole.

In this paper we focus on security challenges of IoT software development and we propose a test-driven security approach.

This section introduces the concept of Internet of Things, the problems encountered in this area so far and the motivation of creating an intelligent security system. The rest of the paper is organized as follows. Section II describes the state of the art in the IoT security status. Section III presents two related solutions. Section IV describes the security vulnerabilities in IoT. Section IV discusses the proposal in detail. Section V

978-1-5386-5541-2/18/$31.00 ©2018 IEEE

highlights the proposed design and architecture. Section VI concludes the paper and proposes directions for further work.

II. State of the Art

The evolution of the Internet of Things will affect multiple application domains. These can organized by the type of scale, heterogeneity, network, availability, coverage, repeatability, user involvement and impact [3]. There are multiple classifications and taxonomies within the IoT environment but we refer to the one that divides it into four application domains: personal and home, enterprise, utilities, and mobile [4]. There are many devices in IoT environment and they are very different between multiple types of areas, so from a security point of view the mission of protecting them is more difficult.

As much as we rely more and more on IoT devices in daily life, the developers have to take into consideration data availability, web and mobile applications which are based on their data, as well as our access to physical things managed by the IoT systems.

In certain applications, the lack of availability could mean damaging the equipment, loss of revenues or even loss of life. For example, in smart cities, IoT infrastructure is responsible for essential services such as traffic control; in medical environment, the devices can have very important in keeping people alive.

To guarantee high availability, the IoT devices have to be protected against cyber-attacks and physical manipulation. IoT systems must include redundancy to eliminate individual failure points and they should be designed to be durable and fault tolerant in order to be recovered in short time.

To identify and approach diverse vulnerabilities for manifold IoT devices, current research is focused on static analysis and dynamic analysis of the firmware and of the source code of the applications running on IoT. Fuzzing is a software testing approach that happens in a Black Box, which mostly consists in finding implementation faults by feeding the target with thousands of random generated inputs in an automated fashion. It is the art of automatic bug finding. A fuzzer is a program that injects input randomly and feeds the input to the target software [5].

Multiple IoT devices, in order to achieve their purpose, connect to different networks and transfer data between them. Considering the fact that the mobile devices have to communicate with many other devices, the attack surface will significantly increase. Thus, in many cases, the hackers prefer such devices because the spread of the injected malicious code will speed up.

III. Related Work

Based on the software market evolution, we estimate that secure design of IoT systems is the long-term solution as the market matures. Usage of secure IoT development frameworks, libraries and protocols is more effective while using a secure design lifecycle management-approach with a certification schema as proposed by the Armour Project [6]. This approach is definitely cheaper and safer on a longer time span, however it presents several challenges for rapid prototyping in terms of technical constraints, time and budget increase in the initial phase.

On the other hand, researchers within the Anastacia Project [7], [8] are bringing SDN benefits towards the IoT landscape, for securing IoT infrastructures using classical IP Security techniques such as Firewalls, IDS (Intrusion Detection Systems) and SIEM (Security Incident and Event Management) by developing a Security Policy Orchestrator running on-top of a fully virtualized SDN environment.

IV. Security Vulnerabilities in IoT

The users want devices that just work, that need minimal interaction in order to function properly and they do not take into consideration the security issues. Energy consumption and scalability issues are very important cross-platform concerns [14] [15]. Most of the devices are designed without security best-practice in mind, and problems appear both at the software and hardware levels. After a security exploit is identified, the process of updating is not mounting in a reasonable time and sometimes it is completely absent.

We enumerate below several vulnerabilities and certain mitigation techniques:

A. Traffic Interception

Traffic Interception is the process of capturing packets in order to steal confidential data or to inject malicious code or data in the traffic. Thus, it can lead to MiTM (Man in The Middle) attack that allows an eavesdropper to listen to the communication between a client and a server, two servers or two clients. To prevent this, a strong asymmetric encryption has to be used, SSL pinning or key replenishment (do not use hardcoded keys inside the system).

B. Web attacks

On Web environment there are multiple types of attacks such as injection flaws from which we enumerate command injection, local file inclusion, and SQL injection [9]. Several solutions to mitigate these types of attacks are developing the applications with security concerns by using appropriate privileges, by taking into consideration every user input without trusting them.

C. Known vulnerabilities

There are over 100.000 well-known vulnerabilities (Common Vulnerability Exposures) discovered over time for applications, operating systems and hardware devices [10]. In order to mitigate the risk, a vulnerability scanning is necessary to identify an entire list of the vulnerabilities, and a testing process that filters the important ones that can have an impact on the organization. The most outstanding attacks exploit well-known vulnerabilities that already have a related patch. Hundreds of patches are released monthly, many of which apply to applications and operating systems residing in the organization's network. The problem resides in establishing

the proper order and process for installing them, and criteria through which to decide what to ignore.

To get rid of certain vulnerabilities in an easily method, it is advisable to use the OTA (Over The Air) update process through which the new version of the firmware or software is deployed automatically.

D. Default or weak credentials

Many IoT devices ship with default credentials such as "admin" password or without one, with hardcoded credentials that once discovered are available from all the clients of that gadget, or with weak credentials that can be identified using techniques such as dictionary or brute force attack. This risk situation can be fixed with usage of strong authentication methods by creating a secure design from the beginning and building in a series of security tests for device hardening that continuously monitor the entire application.

E. Device identification

Device identification represents the act of identifying different components in a network based on communications' analysis, without their knowledge. The attackers try to identify valuable targets to crack the network, to violate the privacy by tracking in an unauthorized manner or to compromise the communication protocol. Having the identity of a host, the eavesdropper can start a spoofing attack where he attempts to gain unauthorized access to a certain system, or sensitive data by pretending to be a valid user. To mitigate this malicious action, a strong encryption is required together with a bootstrapping process.

F. Local data compromise

There is a high risk if the attackers compromise one device so they have access to local data, because this situation can escalate in case of existence of sensible data such as hard-coded keys or cleartext data. To prevent this risk situation, a strong encryption would be effective, or, even better, having activated a physical tampering-proof system that will interrupt the device and refuse to permit local changes or key reset.

G. (Distributed) Denial of service

The (Distributed) Denial of Service attack is meant to make inaccessible or to shut down a host or network by flooding the target with traffic, or by sending certain packets that exploit a vulnerability and cause to happen a crash. The existence of billions of IoT devices in the world can constitute an attack vector that can easily scale to large a size of DDoS method. To avoid this kind of malicious actions, the IoT devices must have a system design that take into consideration security policies, best practices and include automatic security testing that evaluates the application state.

V. Proposed Design and Architecture

Dynamic IoT-System Security Testing (DISST) is a framework that tests each component of the application and the entire system that interconnect them by evaluating the security properties. The main idea is that the workflow steps have be done by having the knowledge about the application's context. This framework is composed of a network of agents that work in a coordinated manner, in order to inspect network streams and protocols by analysing different parameters.

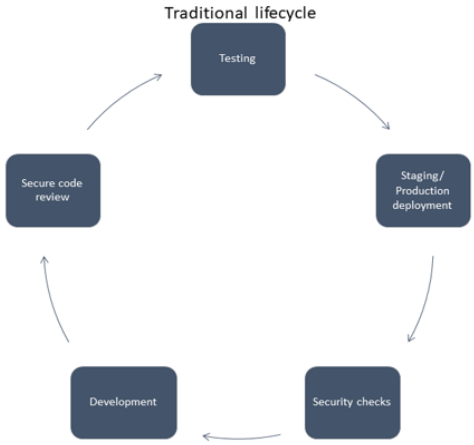

Fig. 1. Traditional lifecycle

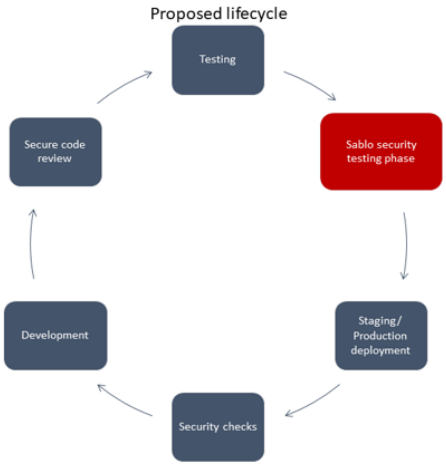

Fig. 2. Proposed lifecycle

We envision an implementation of such a model in a platform called Sablo. The main components are:

1. The IoT **Simulation Engine** - built on SDN and containers. This will give the ability to test the software in a controlled designated test environment;
2. The IoT **Traffic Analyser/Interceptor module** - its purpose is to capture network packets and to run parsing algorithms;
3. The IoT **Attack Generation engine** - its scope is to examine the methods of communication and APIs and then start the fuzzing stage by generating multiple types of packages in order to break the system.

Sablo makes traffic analyses between system components, extracting metadata and details about used protocols and diagnosing network issues. To fulfill this purpose, we use DPI

(Deep Packet Inspection) techniques to get the wanted data and send it to the central engine that evaluates and tests them.

Nowadays, current solutions often focus on data acquisition from diverse functional systems and miss collecting employee-generated data for the final reporting of the security status. Thus, security self-assessments have to be automated to reduce associated costs and efforts [11].

We want to create a virtual environment of the entire structure that simulates each component and also each network between the components. This environment can consist of multiple Containers, Special Boxes or Virtual Machines. Regarding containers, Balena - which is a Moby-based container engine purpose-built for embedded and IoT use cases and compatible with Docker containers - would be a very good solution in creating the virtual environment [12].

We consider that all these parsings, analyzes and evaluations need a graphical representation, so we also propose a GUI (Graphical User Interface) including a map of the components, in order to be easy interpretable and understandable by the developers or system administrators. This graphical architecture is designed to offer several reports of the security tests, and, most importantly, to highlight the problems and possible security breaches.

Sablo is designed to identify multiple protocols at different layers of the network stack such as network transport and also application. We are interested in IP addresses, MAC addresses, open ports, list of running processes, number of interchanged packets, the amount of data, among other parameters.

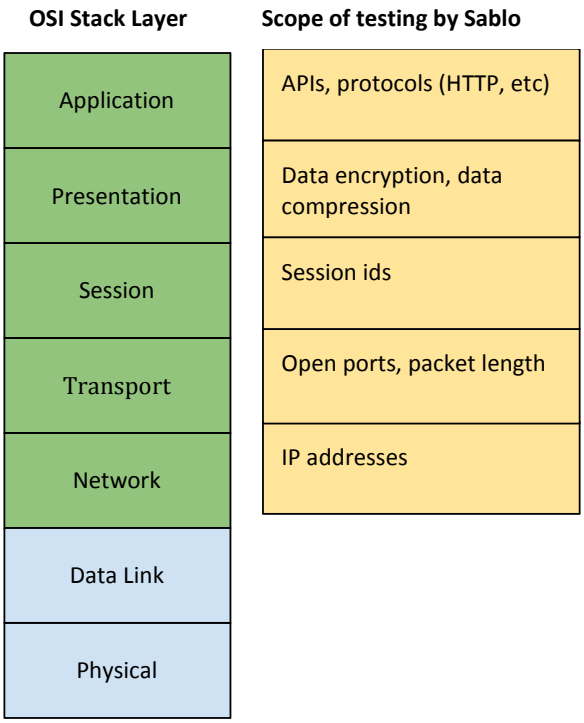

Fig. 3. Sablo testing scope mapped on OSI Stack

Each agent that is deployed on each component of the application collects multiple parameters of the environment, configurations and Operating System, and also binds itself to all capturing interfaces to have full access to the network traffic. The agents inspect the captured packages and determine the communication API (Application Programming Interface) in order to use it further in the workflow of the framework.

Having the defined architecture, the network links and the API formats, Sablo starts searching for corner cases inside each component and also runs fuzzing tests to identify implementation bugs that can lead to unavailable service, a malfunction, or a dangerous vulnerability.

During the development of a new application or system, multiple software tests are created to find implementation bugs. In general, those are run periodically or after a git push of any developer, to avoid new failures in the code that can influence a subroutine that worked perfectly before. There are two categories of tests:

- **unit tests** that test a single component or module of a code to check if it is ready to be used in the application;
- **integration tests,** where individual units of a program are combined and tested as a group to check if they are working fine.

We propose to use tests to capture the APIs and the signatures, and then use them to generate many other tests traversing as many paths as possible, to find problems that can drive to vulnerabilities.

The Sablo system is composed of several internal entities that work together in a coordinated manner to fulfill their objective, such as:

- The agent that is deployed on the one of the target application components;
- The remote agent that connects to the component instead of being deployed;
- A sniffer that monitors the whole application network and analyzes the sent packages between modules. It resides in a Container, a Virtual Machine or a Box and it receives traffic generated by the application.
- The central engine that represents the brain of the whole system that deals with processing and interpreting the data provided by the previous entities.

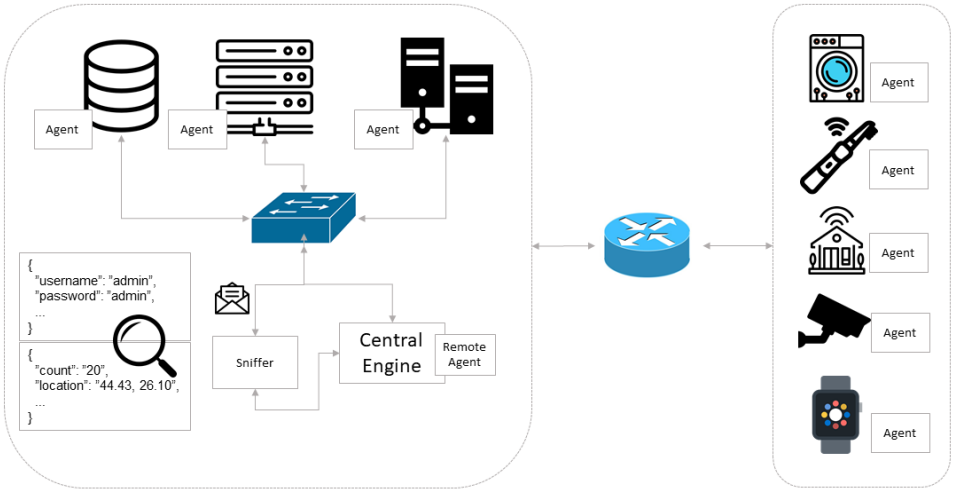

Fig. 4. Sablo system architecture

We believe that an implementation API for the proposed solution would be very useful for the development team in order to add the Sablo framework through the continuous integration and continuous deployment server, as a new step in the entire workflow. In proposed design, we will also create a plugin-based SDK (Software Development Kit), to have the possibility to extend the capabilities to new protocols and new security checks.

The proposed system can integrate very well with CODA Footprint and it comes as a complementary solution to it. Footprint is a platform that audits and entire organizations' critical services in real-time, analyzes whether the organization's critical services are adequately protected, and makes sure the information security defenses are effective 24/7. CODA Footprint is a web-based application that gathers and stores large amounts of information over time for future analysis [13].

VI. Conclusions

In the paper different types of threats were presented alongside with mitigation techniques, and a solution was proposed for protecting organizations' IoT environments against diverse types of attacks and vulnerabilities, by setting a continuous monitoring system that periodically tests the entire infrastructure of the IoT applications during the development period.

Vulnerabilities and security problems are inevitable. The large number of interconnected devices, the complexity of the systems, and the diversity in devices, applications, services, protocols make very difficult the process of determining an incident. The strategies of vulnerability and breach detection include: monitoring the network communications and the journals of activity for anomalies, penetration tests, and ethical hacking, in order to expose the identified vulnerabilities and the application of analysis to identify and notify when an incident happens.

The IoT vendors have to quickly take on responsibility for developing safer devices and keeping them up to date. If the issue of security is not addressed effectively, it will become an important problem to the entire future of IoT. We consider that the paradigm of "Security by Design" is a required approach for the Internet of Things world and it can solve multiple security problems.

We argue that a Dynamic IoT-System Security Testing (DISST) model will complement and enhance research and development efforts. It will allow IoT builders to detect security weaknesses and vulnerabilities at any point in the development lifecycle, without interfering with their rollout/technology plan. It can also be used in conjunction with Anastasia or similar frameworks to train IDS/XL-SIEM detection capabilities, or to refine policies based on application-specific attack patterns.

Future work is needed to explore several protocols from all mentioned levels in the network stack, multiple IoT devices from different types of industry, and to invest in the machine learning area to make a more intelligent system able to generate security test cases autonomously.

The work presented in this paper has been funded by SABLO - Distributed security platform for IoT, subsidiary contract no. 1269/22.01.2018, from the NETIO project ID: P_40_270.

References:

[1] Rolf H. Weber, Internet of Things - New security and privacy challenges

[2] Mirai Botnet Attack Type https://www.corero.com/resources/ddos-attack-types/mirai-botnet-ddos-attack.html

[3] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, T. Razafindralambo, "A survey on facilities for experimental

Internet of Things research", *IEEE Communications Magazine* 49 (2011) 58–67
[4] J. Gubbi, R. Buyya, S. Marusic, "Internet of Things (IoT): A vision, architectural elements, and future directions"
[5] OWASP Fuzzing https://www.owasp.org/index.php/Fuzzing
[6] Armour Project https://www.armour-project.eu/wp-content/uploads/2016/08/D21-Generic-test-patterns-and-test-models-for-IoT-security-testing.pdf
[7] Anastacia Project v0.5 thttp://www.anastacia-h2020.eu/deliverables/ANASTACIA-WP2-T2.2-CNR-D2.2-AttackThreatsAnalysisAndContingencyActionsInitialReport-v0.5.pdf
[8] Anastacia Project v1.0 http://www.anastacia-h2020.eu/deliverables/ANASTACIA-WP3-T3.1-UMU-D3.1-InitialSecurityEnforcementManagerReport-v1.0.pdf
[9] OWASP Top 10 Application Security Risks https://www.owasp.org/index.php/Top_10-2017_Top_10
[10] Common Vulnerabilities and Exposures https://cve.mitre.org/
[11] C. Martin, R. Nasr, M. Hoersken "Automating Information Security Assessments Using Intelligent Software Agents"
[12] Docker Balena https://github.com/resin-os/balena
[13] O. Grigorescu, C. Săndescu, R. Rughiniș, "CODA Footprint Continuous Security Management Platform" in *RoEduNet Conference: Networking in Education and Research 2016*