Course Final Project

Security Issues in IOT

Tianhua Chen(191AEM030)

Preface

In 2016, DYN, the most important DNS service provider in the United States, encountered a large-scale DDoS attack, which made hundreds of websites such as twitter, Spotify, Netflix, Airbnb, CNN and Wall Street Journal inaccessible. In this network attack, hackers used a large number of Internet of things devices. Nowadays, more and more items are labeled as "smart" and become networked devices. Internet of things devices are usually connected by default, and their code is often open source software, so they are easy to be hacked. In this landscape, IoT deployers are confronted with pressing security challenges, including more vulnerabilities and security attacks. The latter require new and more intelligent approaches to IoT security, notably approaches that are able to tackle complex, unpredictable and sometimes asymmetric attacks at scale.

Internet of things vulnerability classification

Deficient physical security: The majority of IoT devices operate autonomously in unattended environments. With little effort, an adversary might obtain unauthorized physical access to such devices and thus take control over them. Consequently, an attacker would cause physical damage to the devices, possibly unveiling employed cryptographic schemes, replicating their firmware using malicious node, or simply corrupting their control or cyber data.

Insufficient energy harvesting: IoT devices characteristically have limited energy and do not necessary possess the technology or mechanisms to renew it automatically. An attacker might drain the stored energy by generating flood of legitimate or corrupted messages, rendering the devices unavailable for valid processes or users.

Inadequate authentication: The unique constraints within the context of the IoT paradigm such as limited energy and computational power challenge the implementation of complex authentication mechanisms. To this end, an attacker might exploit ineffective authentication approaches to append spoofed malicious nodes or violate data integrity, thus intruding on IoT devices and network communications. Under such circumstances, the exchanged and employed authentication keys are also always at risk of being lost, destroyed, or corrupted. In such cases, when the keys are not being stored or transmitted securely, sophisticated (or otherwise effective) authentication algorithms become insufficient.

Improper encryption: Data protection is of paramount importance in IoT realms, especially those operating in critical CPS (i.e., power utilities, manufacturing plants, building automation, etc). It is known that encryption is an effective mechanism to store and transmit data in a way that only authorized users can utilize it. As the strength of cryptosystems depend on their designed algorithms, resource limitations of the IoT affects the robustness, efficiency and efficacy of such algorithms. To this end, an attacker might be able to circumvent the deployed encryption techniques to reveal sensitive information or control operations with limited, feasible effort.

Unnecessary open ports: Various IoT devices have unnecessarily open ports while running vulnerable services, permitting an attacker to connect and exploit a plethora of vulnerabilities.

Insufficient access control: Strong credential management ought to protect IoT devices and data from unauthorized access. It is known that the majority of IoT devices in conjunction with their cloud management solutions do not force a password of sufficient complexity. Moreover, after installation, numerous devices do not request to change the default user credentials. Further, most of the users have elevated permissions. Hence, an adversary could gain unauthorized access to the device, threaten data and the entire Internet.

Improper patch management capabilities: IoT operating systems and embedded firmware/ software should be patched appropriately to continuously minimize attack vectors and augment their functional capabilities. Nevertheless, abundant cases report that many manufacturers either do not recurrently maintain security patches or do not have in place automated patch-update mechanisms. Moreover, even available update mechanisms lack integrity guarantees, rendering them susceptible to being maliciously modified and applied at large.

Weak programming practices: Although strong programming practices and injecting security components might increase the resiliency of the IoT, many researchers have reported that countless firmware are released with known vulnerabilities such as backdoors, root users as prime access points, and the lack of Secure Socket Layer (SSL) usage. Hence, an adversary might easily exploit known security weaknesses to cause buffer overflows, information modifications, or gain unauthorized access to the device.

Insufficient audit mechanisms: A plethora of IoT devices lack thorough logging procedures, rendering it possible to conceal IoT-generated malicious activities.

TABLE V
SECURITY IMPACT OF IOT VULNERABILITIES

Layers	Vulnerabilities	Security Impact			
		Confidentiality	Integrity	Availability	Accountability
Device-based	Deficient physical security	○	●	●	○
	Insufficient energy harvesting	○	○	●	○
Network-based	Inadequate authentication	●	●	○	○
	Improper encryption	●	●	○	○
	Unnecessary open ports	●	○	●	○
Software-based	Insufficient access control	●	●	●	○
	Improper patch management capabilities	○	○	●	○
	Weak programming practices (e.g. root user, lack of SSL, plain text password, backdoor, ect.)	●	●	○	○
	Insufficient audit mechanism	○	●	○	●

Legend: ● vulnerability has significant impact on particular security concept,
○ vulnerability does not have significant impact on a particular security concept

Fig.1 SECURITY IMPACT OF IOT VULNERABILITIES

PRACTICAL APPLICATION

ANALYSIS OF IOT SECURITY DATA

Preface

There is a surge of interest in approaches pertaining to security issues of Internet of Things deployments and applications that leverage machine learning and deep learning techniques. A key prerequisite for enabling such approaches is the development of scalable infrastructures for collecting and processing security-related datasets from IoT systems and devices. By introducing such a scalable and configurable data collection infrastructure for data-driven IoT security. It emphasizes the collection of (security) data from different elements of IoT systems, including individual devices and smart objects, edge nodes, IoT platforms, and entire clouds. The scalability of the introduced infrastructure stems from the integration of state of the art technologies for large scale data collection, streaming and storage, while its configurability relies on an extensible approach to modelling security data from a variety of IoT systems and devices. The approach enables the instantiation and deployment of security data collection systems over complex IoT deployments, which is a foundation for applying effective security analytics algorithms towards identifying threats, vulnerabilities and related attack patterns.

I. Architecture for Data-Driven IOT Security

The SecureIoT platform provides data-driven IoT security services based on the SECaaS (Security-as-a-Service) paradigm. The operation of the platform is driven by the use of data analytics techniques for security related data that are collected from a number of IoT components, which may range from simple devices to complex IoT deployments and may span multiple platforms and administrative domains and include edge/cloud infrastructures. In practice the SecureIoT platform analyses data from these IoT systems and devices in order to provide risk assessment and compliance auditing services, along with a range of security automation (e.g., alerts) and visualization services

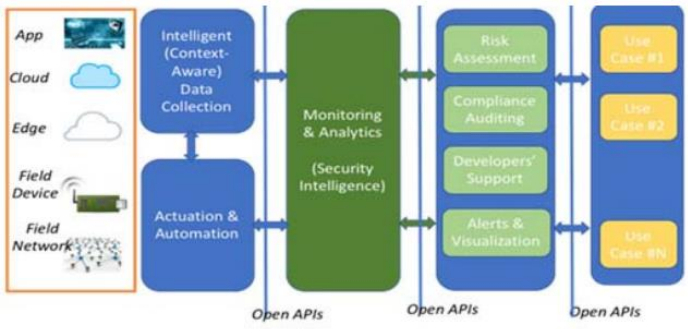

Fig.2 High-Level Logical View of the SecureIoT Architecture

The SecureIoT platform is logically structured as a multilayered security monitoring and enforcement system. Fig.2 gives a high level overview of the various layers of the platform, which include (from left to right):

- **IoT Systems Layer**: This layer comprises the various elements of IoT systems that can act as sources of security information. The elements may be deployed on different IoT platforms and span multiple administrative domains.
- **Data Collection and Actuation Layer**: This layer is in charge of interacting with the field for two purposes: (i)collecting security related data from the above-listed elements through various probes, and (ii) driving security related automation and actuation tasks such as the configuration of the security properties

of IoT systems.

- **Security Intelligence**: This layer analyses the collected data in order to identify security-related events and indicators in the form of incidents, threats and attacks. It is characterized as an "intelligence" layer, because it is the place where intelligent reasoning over the security context takes place by means of data analytics techniques.
- **Security Services (SECaaS)**: This layer comprises SECaaS services that are provided by the SecureIoT platform, including Risk Assessment (RA) services, Compliance Auditing (CA) services and Developer Support (DS) services. This layer implements also other services that are based on the data processing outcomes of the Security Intelligence layer, such as alerting and visualization services.
- **Security Use Cases**: This layer leverages the security services layer in order to provide security functionalities to specific IoT applications. At this level, different IoT applications leverage SECaaS services (e.g., risk assessment) in order to enhance their security and cyber-resilience.

II. Data collection, Actuation and Intelligence Infrastructure

A. Data Collection and Actuation

Fig. 3 gives the anatomy of the data collection and actuation layer of the SecureIoT platform with its main modules and the high-level interactions between them as well as with other layers of the SecureIoT platform, namely, the field systems and devices layer, and the security intelligence layer.

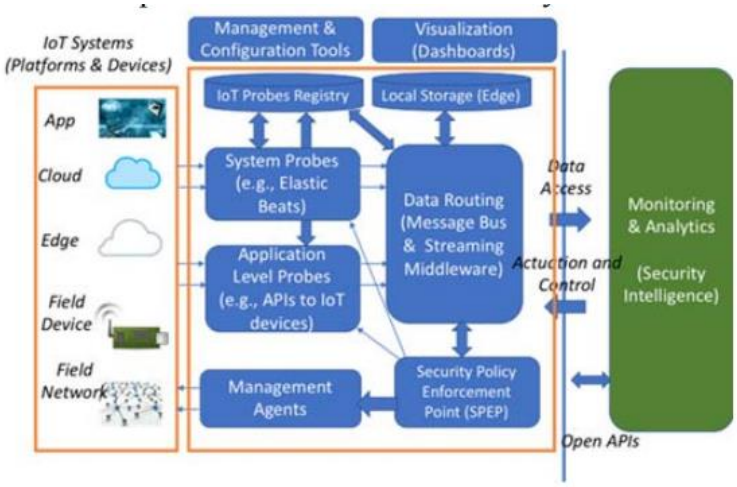

Fig.3 Snapshot of the Security Data Collection Module Data Collection and Actuation Architecture

B. Data Intelligence

The Data Intelligence component (depicted in Fig.4) provides predictive security functionalities based on data analytics techniques. The Data Intelligence component is fed with the security data that are collected from the deployed probes and outputs alerts, notifications, and directives that are used to drive the SPEP module for enforcing actions to the IoT system.

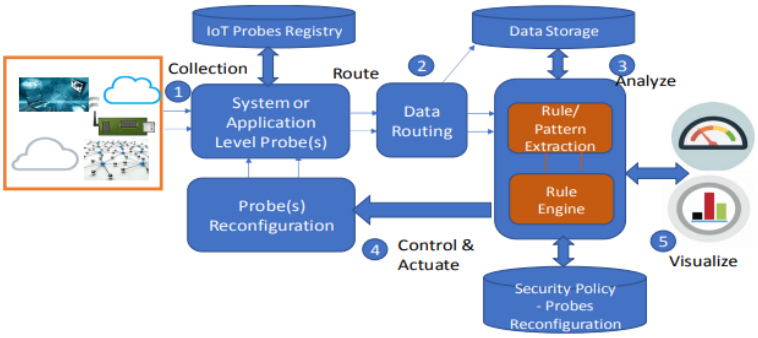

Fig.4 Snapshot of the Security Data Collection Module Data Collection and Actuation Architecture.

The Data Intelligence component comprises two subcomponents: The Rule/Pattern extraction and the Rule Engine. The Rule/Pattern extraction component performs statistical analysis on past collected security data and tries to discover rules that govern relationships among them. The Rule Engine applies the discovered rules to security data that are collected from the IoT probes and fed into it. Whenever the antecedents of a rule are satisfied its consequent is enabled. The consequent may entail the raise of an alarm or notification to the dashboard, or the creation of a directive that must be fed to a SPEP component.

III. Data Modelling for Security Analytics and Configuration

In this section we present the data models that support the operations of the SecureIoT data collection infrastructure.

A. Use of the SecureIoT Data Model

The SecureIoT Data Model is a specification of the different types of data that are collected, streamed, stored and processed by the SecureIoT platform. The purpose of the Data Model is to provide a common specification to all parts of the SecureIoT platform for the different types of data that may be exchanged between them.

B. Data Model Specification The Data Model consists of four main entities:

• DataKind (DK): It describes the kind of captured data along with their types, formats and semantics.

• Platform: It provides an identification and description of the observed IoT platform.

• Probe: Provides an identification and description of the probes deployed to an IoT platform. Each probe instance is bound to an IoT platform by referencing its id.

• LiveDataSet: It provides the structure of the captured observations for both batch and stream data. Each LiveDataSet is associated with a probe, which generates the data. It carries information on the probe and the time when it has generated/grouped the data and also on the captured measurements.

C. Extensible Data Model Specifications and Data Model Integration Concept

SecureIoT foresees IoT applications that may be deployed on multiple IoT platforms, whereby security data may be collected from a number of them. Consequently, a dynamic data model has been developed to cover the diversity of the various IoT platforms that need to be monitored. For this reason, an Objects Identification registry is put into use, which enables the SecureIoT platform to store the observed data from the probes deployed to different IoT platforms in a single database and, in parallel, to use the IoT platform third-party database to enrich the observations with the scenario semantics.

Fig.5 shows an example of SecureIoT DB systems interactions. Each object inserted to each of the SecureIoT

DB systems (Probe Registry, Security Templates, Security Knowledgebase, etc.) is also registered with the same context and a unique identification number to the CIR as well. In this way we can combine all the different DB systems to provide business context to the data produced and stored to the SecureIoT data storage. This procedure is managed by a Management Console which is responsible for binding objects from the different DB systems to the CIR.

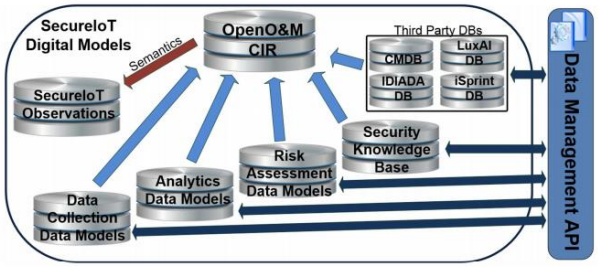

Fig.5 Data Model Integration and Common Interoperability Registry

IV. Prototype Implementation

This section presents a prototype implementation of the data collection, streaming, and storage infrastructure of the SecureIoT platform, whose architecture was covered in the previous sections.

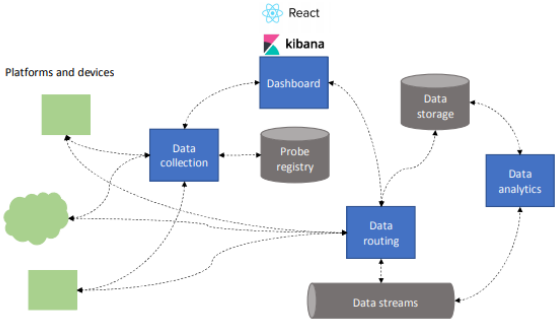

Fig.6 Prototype Implementation of the Data Collection, Streaming and storage infrastructure

The prototype implementation of the infrastructure comprises the following technologies:

• Elastic Beats are used for collecting and shipping data. They implement the System and Application Level probes of the SecureIoT architecture.

• Logstash is used for filtering, parsing, and transforming data to an internal representation. It corresponds to the Data Routing component of the SecureIoT architecture.

• Apache Kafka is used for both storing and streaming data. It supports the functions of the Data Routing and the Local Storage components of the SecureIoT architecture.

• ElasticSearch provides persistence storage services for unstructured data. It corresponds to the Logical Storage component of the SecureIoT architecture. It is used to store historic security data that are subsequently

• Kibana provides monitoring services as well as a management GUI. It corresponds to the visualization (dashboard) component of the SecureIoT architecture.

The current prototype implementation is targeted to a single IoT platform, and as such it does not include a CIR component. Fig.6 shows is a high level view of the prototype infrastructure implementation setup, while Fig.7 depicts a model of its deployment. The following sections give details of the interfaces between components of the infrastructure.

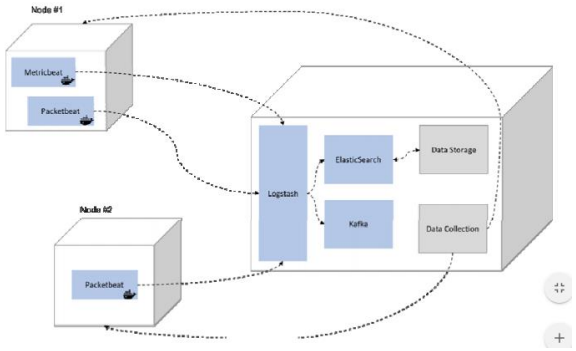

Fig.7 Deployment of the prototype implementation

A. Data collection

Data collection has been implemented as a Spring Boot application. The component uses MongoDB to maintain its internal state, which comprises (1) the metrics it can collect, (2) the platforms and the nodes it collects metrics from, and (3) the collectors it has spawned for that purpose. Data collection exposes the following REST API that allows users to monitor metrics on nodes that may belong to different platforms.

• Create Collector: Creates a collector. The request body contains fields metric for specifying the metric to collect, and node for specifying the node to collect the metric from. The invocation returns the id of the created collector along with its status.

• Start Collector: Starts a collector. The request body contains the id of the collector to be started.

• Stop Collector: stops a collector. The request body contains the id of the collector to be stopped.

• Delete Collector: deletes a collector. The request body contains the id of the collector to be deleted.

• Search Collector: searches for collectors that match a set of criteria. The request body contains fields metric, node, platform, and status, for specifying criteria. The invocation returns a list of collectors that match the input criteria.

Apart from the above endpoints, data collection provides endpoints that allow users to create, update, delete and search for platforms, nodes and metrics.

Each collector is currently implemented as a Beat with the appropriate configuration. For example, a collector that collects system-level CPU usage from a server is Metricbeat deployed on that server and configured to collect CPU usage. All beats are configured to ship data to Logstash, which in turn sends them to both Elasticsearch and Kafka using the corresponding output plugins. That way analysis can be done both on data at rest (Elasticsearch) and on data in transit (Kafka).

B. Data storage

Data storage has been also implemented as a Spring Boot application. The component serves as an abstraction layer over Elasticsearch. It exposes a REST API for querying the stored data. The request body contains a field query that contains the query string to be executed. The invocation returns the list of data objects that satisfy the query.

C. Security Use Case Implementation

The presented data collection infrastructure is used in the scope of various security use cases involving risk assessments and observation of assets' behaviour, including observation of the behaviour of smart objects like robots and connected vehicles. The implementation of these use cases involves the following steps: (i) Modelling of assets (such as computing systems, IoT devices and smart objects) based on the presented data

model; (ii) Implementing probes for collecting data from the various assets; (iii) Processing and analysing the collected data within a risk assessment engine as a means of scoring risks associated with the asset; (iv) Analysing the system-level and application-level behaviour of the IoT system, including its visualization in proper dashboards. For example, in the scope of the socially assisted robot security use case of a SecureIoT project, the behaviour of robots is monitored and visualized, while remarkable deviations from normal mobility patterns (i.e. normal behaviour) of the robot are spotted.

Conclusion

Machine learning and deep learning approaches are increasingly employed for securing IoT systems. These approaches require collection and management of very large amounts of security data for training and building supervised and unsupervised learning systems that must be efficient and able to adapt to different security contexts and deployment configurations. It is therefore important to build scalable, extensible and well-designed infrastructures for collecting security data from all the different elements that comprise nontrivial systems including devices, edge/fog nodes and cloud computing infrastructures. The presented solutions are configurable, scalable and intelligent, leveraging on existing BigData infrastructures. The paper has also described a prototype implementation to demonstrate the concept. Moreover, some general principles for designing a security data collection infrastructure have been discussed.

Reference

1,SμV—The Security MicroVisor: A Formally-Verified Software-Based Security Architecture for the Internet of Things, Italy, 1545-5971 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.

2, Scalable and Configurable End-to-End Collection and Analysis of IoT Security Data, Greece, 978-1-7281-2171-0/19/©2019 IEEE.