A systematic approach towards security in Fog computing: assets, vulnerabilities, possible countermeasures

Mozhdeh Farhadi, Jean-Louis Lanet, Guillaume Pierre, Daniele Miorandi

A systematic approach towards security in Fog computing: assets, vulnerabilities, possible countermeasures

Mozhdeh Farhadi[1,2] Jean-Louis Lanet[2]
Guillaume Pierre[2] Daniele Miorandi[1]

[1] U-Hopper srl
[2] Univ Rennes, Inria, CNRS, IRISA

Abstract

Fog computing is an emerging paradigm in the IoT space, consisting of a middle computation layer, sitting between IoT devices and Cloud servers. Fog computing provides additional computing, storage and networking resources in close proximity to where data is being generated and/or consumed. As the Fog layer has direct access to data streams generated by IoT devices and responses/commands sent from the Cloud, it is in a critical position in terms of security of the entire IoT system.

Currently, there is no specific tool or methodology for analysing the security of Fog computing systems in a comprehensive way. Generic security evaluation procedures applicable to most information technology products are time consuming, costly and badly suited to the Fog context. In this article, we introduce a methodology for evaluating the security of Fog computing systems in a systematic way. We also apply our methodology to a generic Fog computing system, showcasing how it can be purposefully used by security analysts and system designers.

Keywords: Fog computing, Security, Attack, Common Criteria, Methodology.

1 Introduction

Fog computing is an emerging technology which enriches Cloud computing with additional compute, storage and networking resources in close proximity with the end-user devices which generate and/or consume data streams [8]. With the development of Internet of Things (IoT) systems and applications, increasing volumes of data are being produced by IoT devices at the edges of the network. In this situation, it is often not feasible to send all IoT data to a remote Cloud data center and expect acceptable Quality of Service (QoS), especially for applications with low-latency requirements such as augmented reality, industrial control systems and video streaming. Moreover, many applications such as quantified self which use wearable sensors to monitor individuals life often deal with sensitive personal data. In solely Cloud-based approaches for these applications, all these sensitive data would be sent to the Cloud for processing, leaving the user with little control over the usage of their data.

An IoT system architecture with support of Fog computing comprises at least the following three layers [40]: 1) An IoT device layer (comprising the actual sensors and actuators); 2) A Fog computing layer located very close to the IoT devices; and 3) A Cloud computing layer.

Fog computing therefore acts as a middle layer, sitting between the Cloud and the IoT devices. Depending on the application, some parts of the computation may be delegated to the Fog layer, which prevents one from having to send raw data to the Cloud. As Fog computing servers process local data locally, we also expect a lower usage of long-distance network bandwidth as well as reduced response times. Also, the user data are now processed by computing elements which are geographically close to where data is generated, limiting the diffusion of such data and hence possibly limiting security issues. In some cases these so-called Fog nodes are located on the end-user premises themselves, which makes them more privacy preserving compared to pure Cloud-based approaches.

According to McKinsey [35], security and data privacy represent the biggest threat for the further growth of IoT systems and the adoption of IoT-enabled applications. Fog nodes are located in a strategic place between the sensing devices and the Cloud, where all the business logic resides. Thus, Fog nodes have access to all the application data — whether in the form of sensed data being sent to the Cloud or control data sent back to the IoT devices. In addition, they also have access to precise information about the data's origin location. This creates new security issues, because any weakness in the security of the Fog layer can potentially threaten the security of the entire IoT system. This threat is very important because a very large fraction of IoT computing takes place at the edge of the Internet [56].

Several research papers already address the topic of Fog computing security [45, 52, 60]. The main novelty of our contribution is to introduce a coherent and systematic methodology for analyzing the security issues of Fog computing. Currently there exists no specific method to analyze Fog computing systems in a systematic manner. In this paper we introduce an approach, inspired by the ISO/IEC "Common Criteria" standard [16], to help security analysts, designers and developers in identifying and protecting their Fog-enabled systems against security vulnerabilities. We then analyze a generic Fog computing system based on this methodology. Our analysis brings to the fore the possible vulnerabilities that a Fog system may have. It points out where the system designer

should add countermeasures to tackle such vulnerabilities while designing his Fog computing system.

This paper is structured as follows. Section 2 presents the background on Fog computing and common criteria. Section 3 analyses the state of the art of security in Fog computing and relevant related domains. Section 4 introduces the methodological approach. Section 5 details the identified assets in Fog computing systems. Section 6 defines our threat model, which is then used in Section 7 for vulnerability analysis. Finally, Section 8 concludes this article pointing out promising directions for future research.

2 Background

In this section we first present the concept of Fog computing. We then describe the common criteria methodology and the attack tree which form the building blocks for our approach in analysis of Fog computing security.

2.1 Fog Computing

Fog computing is an emerging technology introduced by Cisco in 2011 [7] to decrease bandwidth usage and latency in IoT applications. According to IEEE [26], Fog computing is defined as "a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the cloud-to-things continuum." As any other new technology, it is important to study its security aspects: otherwise, the new technology not only provides advantages, but may also hurt its users. Fog computing has similarities with Cloud computing but it also has its own properties on some aspects. Thus, the methods and techniques applied in Cloud security, may not necessarily fit the security of Fog computing.

Fog computing uses computation elements distributed in the network. These devices, thereafter called Fog nodes, perform computation or storage tasks for IoT devices in their vicinity. The Fog nodes do not aim to replace the Cloud, but they enrich Cloud services by providing resources along the way from the IoT devices to the Cloud.

The Fog layer is essentially located between the IoT layer and the Cloud layer. It may sometimes be organized along a hierarchical layout, with also possible communication between the nodes in a distributed manner. For example, if an application is containerized and loaded on a Fog node near to a specific IoT device, then when the IoT device moves, the Fog-hosted application may need to be migrated to another Fog node to maintain proximity. Consequently, there would be a need for Fog-to-Fog communication. The network that these Fog nodes use to communicate with each other may be different than the network used by Cloud servers for their internal communication. For instance, in some geographic locations, the only available network for communication among the Fog nodes may be wireless.

As the Fog nodes are geographically distributed, they also may belong to different organizations. Fog infrastructure providers may be Internet Service Providers (ISP), Mobile Network Operators (MNO), Cloud infrastructure providers, smart cities or any organization which owns a set of Internet-connected

gateways with computing capabilities. Fog service providers can be the aforementioned actors, or even end users who own a private Cloud and are interested in exploiting their spare resources [60]. In consequence, the ownership of the Fog nodes may be different than the Cloud model. In the Fog paradigm, any user who has spare resources may deliver them to the nearby IoT devices in the form of Fog nodes.

Because of their proximity to end users and IoT devices, Fog nodes usually have access to personal data, and in particular the data source location. Data that is being handled by Fog nodes may therefore be more security critical than the data being processed in the Cloud servers.

As Fog computing has different properties than Cloud computing, and due to its critical location between the IoT layer and Cloud layers providing it with first-hand data, it needs to be analysed separately. Studies in the area of Cloud and IoT will not cover the needs of the Fog security. Moreover, specific studies in the area of Fog security, does not follow a systematic approach. In this paper, we introduce a methodology to analyse the security of Fog systems following a systematic approach.

2.2 Building blocks for our methodology

We use the Common Criteria approach as the basis for our evaluation methodology of Fog computing security. In addition, we use the attack tree model to analyse, represent vulnerabilities and countermeasures for our defined system. As both the Common Criteria and the attack tree model form our methodology, we introduce them in this section.

2.2.1 Common Criteria

Common Criteria (CC) is an ISO/IEC standard (ISO/IEC 15408) for computer security certification [57]. It provides a methodology to evaluate the security of IT products and to rate products according to Evaluation Assurance Levels (EAL). As the process of getting a CC certificate is not simple, we cannot expect every IoT project use this methodology to get a certificate. On the other hand, due to the importance of security in such projects, we aim to tailor the CC methodology for the security evaluation of Fog computing systems in order to make the process simple and efficient for this specific use case.

In the process of getting a CC certificate, the concerned product should be sent to a CC laboratory for evaluation, which involves financial costs. Moreover, a group of technical people involved in the design and development of the product should explain the product to the evaluation team, which may require months of work.

For evaluating a product according to the CC methodology, a document called Protection Profile (PP) is needed. The CC laboratory evaluates the product according to this document, which is written by an (usually governmental) organization which cares about users' security. A PP is a representation of users' security requirements, and it includes the security requirements for the product. Through the evaluation process, the product is investigated against these specified security requirements. For example, the Federal Office for Information Security in Germany (BSI in short in German language) has written a PP for Smart Card operating systems and defined the expected security characteristics

for a Smart Card OS. Thus, customers of a Smart Card OS in Germany, such as Smart Card solution providers, consider the CC certificate of a Smart Card OS which is evaluated against that PP document when selecting one for their products.

Currently there is no PP document for evaluating a Fog computing product. Moreover, even with the existence of such document, CC evaluation process would not be affordable for all perspective actors, because of the associated time and cost. However, as the CC evaluation process is complete and systematic, we aim to make it easier for use in the domain of Fog computing. For this reason we also make use of a modelling tool called attack trees.

2.2.2 Attack trees

Security has different meanings in different contexts and environments. We need to know "secure from what?" and "secure from whom?" [47]. To secure a system, we need to know all the possible ways that the system can be attacked to be able to design countermeasures to tackle them [47]. Knowing the abilities of the attacker and the operation environment of the system gives us another vector to understand and defend the system. The first question (secure from what) can be modelled using attack trees. The latter question (secure from whom) will be discussed in the threat analysis section at 6.

In every system, assets are the entities that the attackers aims to access and that the system owner/user wants to keep safe. For example, if we define our system as a mobile application, the user data stored in the application is the asset which needs to be kept safe from attackers.

We use attack trees as a graphical model to represent all the different scenarios to attack a system. An attack tree gives the big picture of all the possible routes to attack a system.

Attack trees were introduced by Schneier in [47]. They constitute a convenient approach to analyze the different ways in which a system may be attacked. It is an analytical technique (top-down) where an undesirable event is defined and the system is then analysed to find the combinations of basic events that could lead to this undesirable event. The undesirable event represents either the objective of the attacker or a property of the asset that needs to be protected. The seminal work of Schneider has been extended to Defence Trees [5], Attack Countermeasure Trees [46], Boolean logic Driven Markov Processes [10] and several others.

An attack tree follows a hierarchical structure where the topmost element represents the main objective of the attacker. The other nodes are either referred to as sub-goals or leaf nodes. Figure 1 depicts the attack tree for unauthorized entrance to a locked room.

At the bottom are leaf nodes, which are the potential attacks of a system which cannot be detailed further. In the middle, there are sub-nodes made of a combination of leaf nodes. These leaf nodes and intermediate nodes are combined using a logical relationship of either OR-gate or AND-gate. An OR-gate is used to represent an alternative attack relationship whereas, an AND-gate is used to represent a set of attacks that if occurred together would sequel a certain goal or intermediate node.

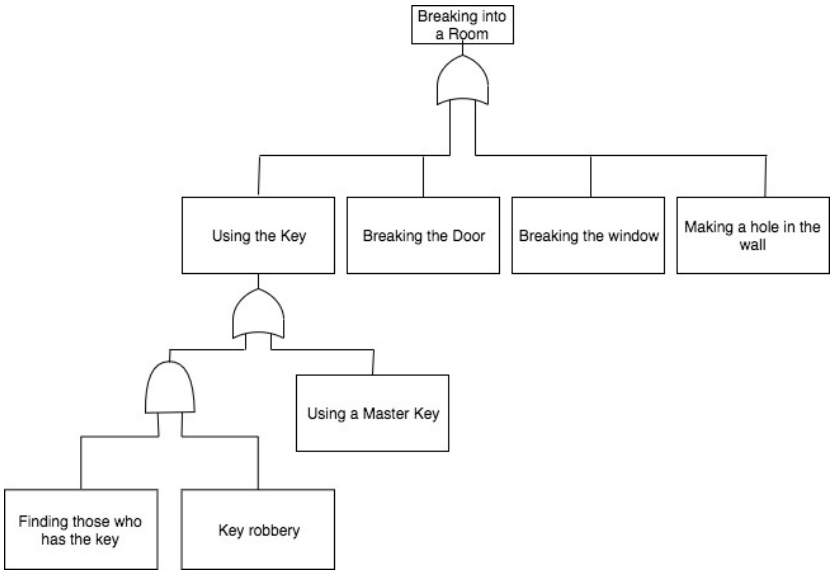

Figure 1: Attack tree for unauthorized entrance into a locked room.

If the attack tree also includes the defence mechanisms, it is called attack-defense tree. In this case, the security analyst is able to check if all the attacks presented in the tree are mitigated using at least one countermeasure.

Figure 2 depicts the corresponding attack-defense tree for unauthorized entrance into the room.

As it can be seen in the Figure 2, the countermeasures are represented with NOT gates. If a countermeasure is located close to the root level, it can tackle more threats. For example, the use of "Biometric key" can cover all the branches under the "using key" attack. As a general rule in attack-defense trees, the higher the countermeasure is in the tree, the better is its coverage.

In this article we use attack trees and attack-defense trees to represent and analyse the security of Fog systems.

3 State of the art / Related work

Fog computing lies at the cross-roads of several well-established research themes: distributed systems, IoT and Cloud computing. In this section we briefly survey the relevant state-of-art on security issues and potential countermeasures. We then present some motivating examples of security threats in Fog computing and show how to use the attack tree model for representing them in a formal way.

3.1 Security in distributed systems

Distributed systems are composed of units (also referred to as nodes) which carry out computational tasks and interact by exchanging messages over communication channels. Units can be geographically dispersed, and may be under

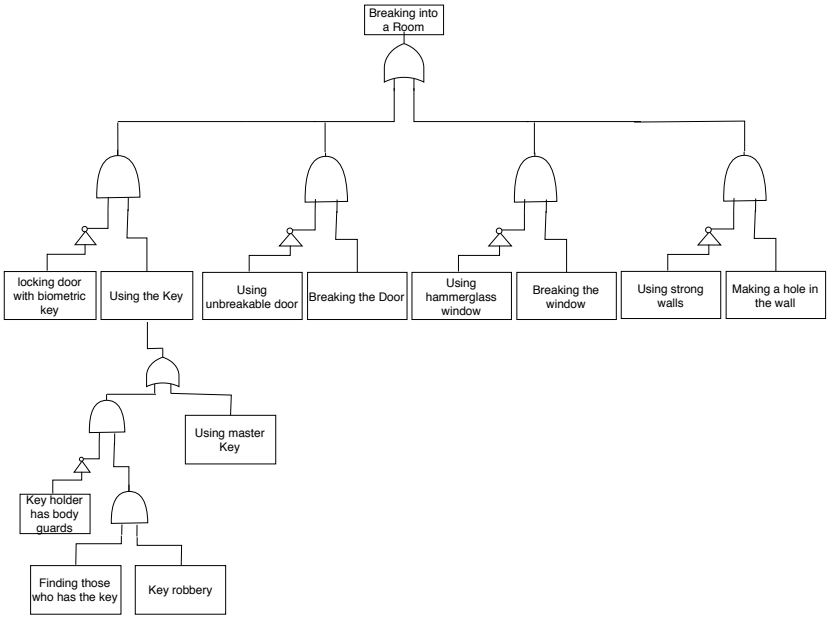

Figure 2: Attack-defense tree for unauthorized entrance into a room.

the administrative control of different entities. In a distributed system the overall system-level functionality emerges out of the coordinated action of single computation units; no single controller is present, but each unit carries out its activities with a well-defined level of autonomy [53].

From a security standpoint, distributed systems present a number of challenges which are not present (or which exist, but in a radically different form) in centralized IT systems. The more radical difference lies in the fact that a unit has little (or no) way to know whether the other units in the system behave correctly. This opens up significant opportunities for the introduction of faulty or malicious software [37]. The latency inherently related to the usage of communication channels means also that the identification of a security threat or attack may incur greater delays, which makes it harder to defend the system in a coordinated fashion. In case units are geographically distributed, it may also be difficult to guarantee the physical security of the different computation units. Messages exchanged among nodes may be eavesdropped or corrupted while in transit, thereby calling for the introduction of appropriate measures to ensure message integrity and confidentiality. At the same time, the fact that overall functionality emerges out of interactions among different units, and the lack of a single point-of-failure, may offer advantages in terms of robustness, reliability and resilience [53]. If data are also distributed across the various nodes, a single compromised node may give the attacker access only to a (potentially small) subset of the system-level information, thereby limiting the impact of such attack in terms of data confidentiality and integrity.

As nodes in a distributed system rely on a communication network to interact by exchanging messages, the security of the network itself is crucial for the operations of the overall system. This covers a whole set of aspects, ranging from

link-level security (ensuring that an established communication channel between two nodes cannot be disrupted and/or messages passing over such channels cannot be modified or eavesdropped) all the way to network-level security (covering aspects such as – distributed – denial of service, routing security etc.). For a complete description and classification of network attacks, we refer the reader to [49].

3.1.1 Trust

Trust is a fundamental notion in distributed systems. By definition, computation units in a distributed system cannot be considered fully trusted a priori. This requires the introduction of mechanisms able to ensure correct functionality even in the presence of misbehaving and/or malicious nodes. A well-known example in this case is the Byzantine general problems, which covers the case in which nodes need to achieve consensus (a fundamental function in distributed systems) in the presence of "traitors" [31]. This has given rise to a rich and active line of research, in which distributed algorithms able to ensure correct functionality in the presence of malicious nodes have been proposed. Yet, fundamental limitations do exist on the fraction of malicious nodes that can be sustained by a system in order to ensure correct operations.

A related notion is that of "reputation." Basically, as it may be hard to identify in a controlled way which nodes are malicious/misbehaving, a widely used idea in distributed systems is for nodes to monitor over time the behaviour of nodes they interact with. This requires nodes to have (i) the knowledge of the expected behaviour by other nodes [trivial if all nodes behave the same, potentially extremely complex in other situations]; and (ii) means to observe the behaviour of other nodes [typically by logging and analysing messages originated by the observed nodes]. Over time, a node can therefore build a score representing the "level of trust" of the nodes it interacts with [32]. Such measure is meant to mitigate the impact of a potentially misbehaving node, by basically retaliating it from interactions or limiting their involvement in critical interactions.

3.1.2 Authentication and non-repudiation

The authentication of the identity of a communication partner represents a key security issue in distributed systems. Authentication provides a basis for auditing and accounting, and is a prerequisite for the implementation of many access control schemes, required to grant or restrict access to given resources. In the case of distributed systems, authentication refers not only to the identity of users (as in the case of centralized systems) but also of nodes (specifically: originators of messages). Typical solutions are based on public key schemes (and certificates) and thereby requires the presence of a trusted certification authority (CA). While this may seem contradictory with the notion of "distributed" systems (a CA may represent a single point of failure for the security of the overall system), it is considered a pragmatic choice and safely used in most cases [50].

Trust management solutions [6], which use a combination of credentials and local security policies to allow direct authorization of security-critical actions, partially alleviate the issues presented by the presence of a centralized CA. Yet,

they require a tight coordination in terms of security policies (distribution, updates, revocation etc.) and present a significant complexity, which makes them not suitable for usage in systems where the availability of resources (computing power, communication bandwidth) may be problematic.

Some lines of research explore the possibility of so-called zero-trust schemes [30]; albeit promising, this research line is not yet consolidated.

Non-repudiation is a security property requiring the fact that no node can deny having sent a message originated from it. Non-repudiation is a prerequisite for the implementation of an audit trail system. Non-repudiation requires both nodes and messages to be identifiable, and it requires means to ensure the authenticity of messages.

3.2 Security in IoT

Security is recognized as one of the main threats (and potential show-stoppers) in the adoption of IoT technologies and services [22, 28, 36]. The merge of virtual (digital) and physical dimensions enabled by IoT makes indeed security issues even more pressing than in traditional IT systems. Data breaches could easily reveal personal and even sensitive information about individuals; attacks may prevent the correct working of IoT-enabled systems such as smart home systems and connected cars, with potentially life-threatening effects. We analyze, with a data-centric approach, in detail the security issues in IoT systems and discuss the potential applicability of IoT security solutions to Fog computing architectures. We consider four data security dimensions: integrity, authenticity, confidentiality and privacy. For more details on the state-of-the-art, we refer the reader to a number of excellent surveys presenting the state-of-the-art in *security in IoT* [63, 62, 4, 39, 51].

Our focus is in particular on data in transit, i.e., data that move dynamically across various nodes. This does not mean, clearly, that security issues related to data at rest are irrelevant, but, rather, that they do not present specific peculiarities in IoT with respect to traditional IT scenarios.

For the data in transit part, instead, the limitations of IoT devices in terms of computing/memory/storage/energy represent challenges for existing approaches and require novel solutions [48].

3.2.1 Data integrity

Data integrity refers to ensuring that a given (IoT-generated) data entity has not been altered or tampered with; otherwise stated, that it has not changed from a given "reference version" (typically: the original one) [2]. In many IoT scenarios data integrity plays a key role: think, for instance, of a health application monitoring physiological parameters; if sensor data is tampered with, this may have noxious consequences.

The traditional approach to ensure data integrity assumes that the attacker does not control the operating system, and is based upon the usage of encryption techniques together with a cyclic redundancy check (CRC). Cyclic redundancy check are commonly used to provide integrity against unintentional alteration of data (due, for example, to an error on the communication channel). Yet they are not sufficient in the presence of a malicious device in the system; the latter one could indeed replace the message with a fake version thereof, which respects the

CRC structure. Encryption by itself does not guarantee the integrity of data, as encrypted data may be easily replaced by a fake version thereof.

If the CRC structure is not known to the attacker, though, a combination of CRC and encryption provides sufficient means for the intended recipient to check whether the data packet has been altered or not. Similarly, checksums are a popular way of checking (not guaranteeing) data integrity; in this case the data entity (packet, message) is provided as input to a hash function; the resulting output is then transmitted to the intended recipient. By repeating the hash function on the received data and by comparing the results, it becomes therefore possible to check whether the data entity has been corrupted/altered.

Encryption however presents the problem of being typically a rather resource-hungry process. In a traditional IoT scenario, where resource-constrained nodes communicate directly with the Cloud, the usage of strong encryption may be problematic from an energy consumption point of view [33]. In a Fog architecture, this issue is still relevant for the IoT-to-Fog communication, whereas it can be safely assumed that the Fog node has sufficient resources to run encryption functions before forwarding data to the Cloud tier.

3.2.2 Data authenticity

Authenticity of data can be seen as a specific case of data integrity, in which the "reference version" is the original (or: authentic) one. In traditional IoT architectures, where data is pushed from the IoT device directly to the Cloud, the boundary between authenticity and integrity is rather blurred [27]; in the Fog case, where this additional middle layer is present, this requires that data is not tampered within the Fog node.

3.2.3 Data confidentiality

Confidentiality refers to the ability to hide information from those unauthorized to view it. This includes, in rigorous terms, the content of the data entity itself, as well as the identity of the sender and of the target recipient of said information. In most cases, however, confidentiality is limited only to the data entity itself, leaving in plain sight the id of sender and receiver. In terms of relevance, think of a smart home scenario in which sensors measure the presence of the inhabitants; such information is confidential and should not leak (e.g., to burglars).

Encryption methods (in particular in their asymmetric form) are commonly used to preserve confidentiality of data content [11]. This can be used to protect the information content, yet in general it is not suited to protect the identity of the sender and recipient, which can be accessed by a malicious eavesdropper by just monitoring packets, e.g. over a wireless link. When data is at rest, authentication methods (passwords, biometric verification etc.) are traditionally used to control (and restrict/prevent) access to confidential data. This requires also the definition of suitable access privileges that map the identity of a user with the data items she is authorized to access.

In the IoT device level, again, the usage of encryption methods is partially limited by the available computational (and energy resources) [23]; if a Fog layer is present, depending on whether data is stored locally or is just forwarded to the Cloud, an appropriate set of access control methods might be required [55].

3.2.4 Data privacy

The term data privacy is used often as a synonymous of data confidentiality, in particular when personal data (as broadly defined, e.g., by the GDPR [15]) are considered. In reality privacy is a legal term, and privacy refers to a normative right by individuals to make sure that some information remains private and not accessed by non-authorized parties.

As IoT data may often refer to individuals (think of wearable devices measuring physiological parameters, all the way to smart meters measuring per-device energy consumption patterns), privacy issues are a primary concern [2, 48, 22, 21]. Solutions aimed at guaranteeing the privacy of data in the IoT sphere include a number of technical measures meant to ensure data confidentiality as well as a set of non-technical measure (e.g., user awareness programs, compliance monitoring systems and internal audit procedures) making sure data is not accidentally disclosed by the relevant controller or processor.

3.3 Security in Cloud computing

Security is universally recognized as one of the main challenges in with Cloud computing [20]. Although Cloud computing shares many security issues with traditional distributed systems, its specificities such as virtualization and multi-tenancy create a number of specific, challenging issues. Such issues can be broadly classified into issues that are faced by the Cloud platform providers, and issues that are faced by the Cloud platform tenants. To fully secure a Cloud-based system it is therefore important that both the provider and its tenants take appropriate security measures.

The Cloud Security Alliance periodically releases a documented list of the top threats to Cloud computing [13]. The main ones are described below.

Data breaches: a data breach is an incident in which sensitive, protected or confidential information is released, viewed, stolen or used by an individual who is not authorized to do so. Although data breaches may happen in a wide range of on-line system, the fact that Cloud tenants lose their ability to have physical access to the servers hosting their information requires the use of more sophisticated solutions.

Insufficient identity, credential and access management: data breaches and enabling of attacks can occur because of a lack of scalable identity access management systems, failure to use multi-factor authentication, weak password use, and a lack of ongoing automated rotation of cryptographic keys, passwords and certificates. It is therefore a responsibility of the Cloud tenants to take informed decisions on how to configure the identity access management systems and how to keep their credentials confidential.

Insecure Interfaces and APIs: Cloud computing providers expose a set of software user interfaces (UIs) or application programming interfaces (APIs) that customers use to manage and interact with Cloud services. The security and availability of general Cloud services is dependent on the security of these basic APIs. Cloud providers must therefore secure their UIs and APIs, but also engage in extensive certification processes to convince their tenants of the security of their services.

System vulnerabilities: System vulnerabilities are exploitable bugs in programs that attackers can use to infiltrate a computer system for the purpose of stealing data, taking control of the system or disrupting service operations. Although this type of vulnerability applies to almost all computer systems, in Cloud environments systems from various organizations are placed in close proximity to each other, and given access to shared memory and resources, creating a new attack surface.

Account hijacking: although attack methods such as Phishing, fraud and exploitation of software vulnerabilities are not specific to the Cloud, hijacking of a Cloud account may have very serious consequences such as eavesdropping on activities and transactions, manipulating data, returning falsified information and redirecting clients to illegitimate sites.

Malicious insiders: A malicious insider threat is a current or former employee, contractor, or other business partner who has or had authorized access to an organization's network, system, or data and intentionally exceeded or misused that access in a manner that negatively affected the confidentiality, integrity, or availability of the organization's information or information systems. Cloud computing increases the seriousness of this thread because it increases the number and size of organizations taking part in any Cloud-based application.

Since Fog computing is defined as an extension of Cloud computing, it clearly also inherits all of its security issues and potential solutions. The purpose of the present article is to extend this analysis including threats that are specific to Fog computing.

3.4 Security in Fog computing

Although Fog computing naturally inherits most of the security issues and possible solution from Cloud computing, it also has a number of specificities due to its decentralized architecture or its expected usage. On the other hand, it also creates potential opportunities to address specific types of privacy and/or security issues.

Roman *et al.* discuss the commonalities and differences between security issues found in Fog computing, mobile edge computing, mobile Cloud computing and Cloud computing [45]. Most security-related issues in these environments relate to identity and authentication, access control, protocol and network security, trust management, intrusion detection, privacy, virtualization, and forensics. A similar set of issues is discussed in [38, 44, 60]. Although these papers try to build an exhaustive list of potential security issues in Fog computing, they do not make use of a systematic methodology to explore these issues in depth.

A specific issue in Fog computing relates to the fact that Fog nodes typically consist of small machines distributed across a large geographical area, and are thus arguably harder to secure physically. This mandates special measures to be taken by clients to verify the authenticity of query results produced by untrusted Fog nodes [42]. Similarly, for the same reasons, we can identify a significant exposure to man-in-the-middle attacks, where an attacker creates fake Fog nodes which attempt to obtain private data from IoT devices and/or to influence

these devices and make them misbehave. Such topics are typically addressed by designing specific authentication and authorization mechanisms [52].

Another source of potential security issues comes from specificities of Fog computing applications and the way they handle data and Fog computing resources. Khan *et al.* extensively study a set of thirteen reference Fog computing applications and analyse which of twelve potential security issues apply to which application [29]. One may expect that some Fog computing applications – such as driving assistance applications for communicating vehicles – will be shared by a large number of entities, which hints at Blockchain-based solutions [34, 59] due to the distributed nature of Blockchain. As Blockchain provides tamper-proof storage and uses consensus mechanism to confirm past events, it can be used as a security means.

Last but not least, Fog computing can also sometimes be seen as an enabler for improving the privacy or security of other systems. Fog nodes may for example be used to address security issues experienced by IoT devices [19, 3]. Also, the fact that Fog platform can process privacy-sensitive data close to the user rather than in a single centralized Cloud may be used to increase the privacy guarantees for applications which handle personal data [18].

3.5 Attack scenarios in Fog computing

In this subsection we present some potential security threats and model them using Attack Trees. Of course scenarios are only a means to illustrate some situations where security can be defeated. We describe hereafter three scenarios related to different elements of the architecture.

3.5.1 Privacy issues with IoT devices

Someone uses a wearable fitness tracker device for sport activities. The new generation of wearable fitness tracker devices has the ability to provide seamless integration with on-line social networks. A well known example is the Fitbit device [1], which sends account and password in plain text to the server and also stores user credentials in plain text. An attacker can connect the device via USB and easily access the credential. As a consequence, user data are accessible to the attacker and user privacy is threatened. Moreover, as the attacker can access the credentials, she will be able to change the user data, whether it is stored on the IoT device or somewhere else (e.g., a Fog node).

This example highlights that all elements of the system, ranging from IoT devices all the way to the Cloud server, must be securely designed. As the authentication process deals with user credentials, it is a key point and should be designed and managed carefully. It is indeed challenging (if not impossible) to build a secure system if elements of the system have been designed with security as a minor concern. This examples also illustrates the fact that not all devices should be considered trustworthy.

3.5.2 Compromised Fog nodes

In [61], the authors show that eavesdropping encrypted communication channel is possible if a Fog node is compromised. In this scenario, a gateway which

[1] Reversing the protocol: `https://github.com/openyou/libfitbit`

serves as a Fog node is replaced by a fake/modified one. Zhang *et al.* evaluated this scenario to demonstrate some issues in Fog computing security [61]. After compromising the gateway, they inserted malicious code into the system (in fact hooking the system calls of the Linux OS), and then analysed the CPU usage. The use case they considered consisted of a video call between a laptop using WiFi and a 3G cell phone. The communication is encrypted from the laptop to the gateway and then encrypted to be sent through the 3G network. They demonstrated that a man-in-the-middle attack consumes few resources in Fog devices and that therefore it cannot be detected by state-of-the-art anomaly detection systems. This attack works on a large number of gateway devices. This example shows the importance of ensuring the integrity of the operating system of the gateway acting as a Fog node.

3.5.3 Network attack

The capability of the network can be largely reduced due to Denial of Service (DoS) attack. These attacks are intended to prevent legitimate users from accessing the services provided by Fog infrastructures. At the end-user level, an attacker can jam the wireless communication channel to prevent certain users from communicating with the infrastructure. The attacker can also deplete the resources of Fog nodes as a means to prevent it from allocating new resources for other users or delaying its responses. In a DoS attack , even a small set of compromised devices can request for unlimited processing/storage services. As a consequence, it can lead to stall the requests made by legitimate devices. The intensity of such an attack rises manifold when a set of nodes simultaneously launch this attack.

3.5.4 Attack tree models

The three scenarios described above cover attacks targeting the following properties:

- Scenario 1 (S1) User data privacy;
- Scenario 2 (S2) Fog node OS code integrity;
- Scenario 3 (S3) Availability of Fog node.

The fault tree for S1, depicted in Figure 3 is a succession of OR gates expressing that only one event is necessary to reach the goal. To break the user privacy, existence of one of the four branches is sufficient. It means that if in a system, either the user data (in the Cloud or Fog) is stored in plain text, or the user credential or user data are sent or stored unencrypted, it can cause privacy threat. In fact, any branch would leave to a data privacy breach.

Figure 4 depicts the second scenario. As shown in the figure, the absence of end-to-end encryption is a pre-requisite for this attack. Afterwards, existence of at least one of the tree possibilities, which are Downgrading the code version, hooking OS functions [9] or using Fake nodes can enable this attack.

For the last scenario, the DoS attack can be achieved by several means. The major ones are ICMP flood attack, TCP SYN flood attack or UDP flood attack, as represented in Figure 5. The attack tree has three different branches with an OR gate. Each of the branches corresponds to one of the mentioned causes.

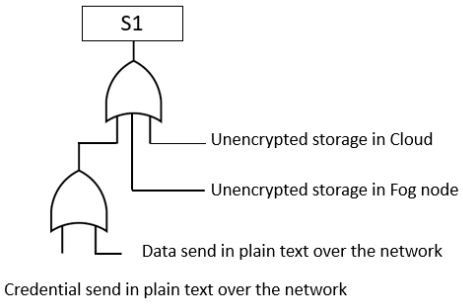

Figure 3: Attack tree for scenario 1 (user data privacy).

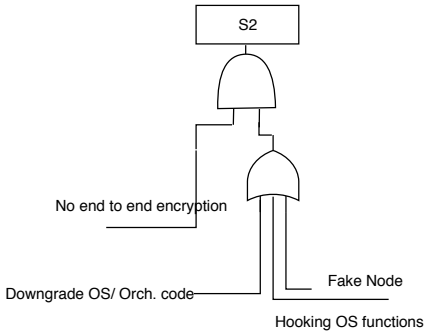

Figure 4: Attack tree for scenario 2 (Fog node OS code integrity).

The presence of one of the causes is sufficient for the attacker to reach the root of the tree.

- ICMP flood attack occurs when the compromised devices send large volumes of ICMP_ECHO_REPLY packets to the victim system. These packets force the victim system to reply and the combination of traffic saturates the bandwidth of the victim's network connection.
- During the SYN flood attack, the attacking system sends SYN request with spoofed source IP address to the victim host. These SYN requests appear to be legitimate. The spoofed address refers to a client system that does not exist. Hence, the final ACK message will never be sent to the victim server system. This results into increased number of half-open connections at the victim side.
- UDP Flood Attacks. User Datagram Protocol (UDP) is a connectionless protocol. When data packets are sent via UDP, there is no handshaking required between sender and receiver, and the receiving system will just receive packets which must be processed. A large number of UDP packets sent to a victim system can saturate the network, depleting the bandwidth available for legitimate service requests to the victim system.

As a consequence of these Attack Trees, all the attacks that should be tackled by the system designer are presented. The trees can also include countermea-

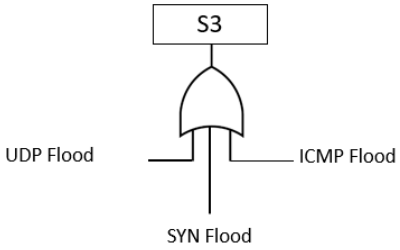

Figure 5: Attack tree for scenario 3 (availability of Fog nodes).

sures against attacks, leading to what are called attack-defense trees. We use attack trees and attack-defense trees in our analysis for Fog computing attacks in Section 7.

4 Methodology

The contribution of this article is the introduction of a specific methodology for analysing the security of Fog computing systems. As we want this methodology to be applicable to many different Fog systems we do not include the details about the operating system, orchestrator and such, and instead try to maintain the discussion at an abstract level about a generic Fog platform.

In our methodology, we start by defining our Target of Evaluation (ToE) as a set of Fog nodes. These Fog nodes can be analysed with three different perspectives: Device level, System level and Service level. In each perspective, we study which parts of the Fog computing system might be valuable for an attacker. For example, in the Device perspective, the physical memory of the Fog node is considered as a valuable element that an attacker may be interested to get access to. In the Service perspective, having access to a running service on top of the Fog node may be valuable for an attacker. Thus, the service availability is considered as an asset. This topic is discussed in Section 5.

In the second step, we define the threat model which identifies the capabilities of an attacker. We aim to study a scenario with reasonably powerful attackers, but exclude the most powerful attacks such as one where an attacker may physically detach a Fog node from the system and examine its full memory and disk content. In the threat model we define all the assumptions about the environment and the attacker that affect our analysis on the Fog system. This topic is discussed in Section 6.

Finally, we analyse the vulnerabilities and possible defences that the generic Fog system may have under the defined attack model. We do so by studying the state of the art of existing attacks, and also by systematically searching for an attack path for an attacker to reach the system's assets. We represent the possible vulnerabilities and their potential countermeasures using attack-defense trees. Our analysis of a generic Fog platform provides a basis for a detailed analysis of other specific Fog systems. As we utilize the attack tree modelling in our methodology, the user is able to have a good understanding about the security of her Fog system visually. This topic is discussed in Section 7.

In Section 7.4, we apply our methodology to a concrete Fog system to represent how this methodology works through two examples. In these two examples we show the case of threats to OS code integrity and OS data integrity in a Fog system. Using the proposed methodology, we also present the required countermeasures for the mentioned vulnerabilities.

5 Identification of assets in a Fog computing system

In our proposed approach, we first analyse the security of Fog computing by identifying the assets of a generic Fog computing system and investigating possible threats to these assets. Identifying the assets of a Fog computing system is the first step before identifying vulnerabilities and, accordingly, appropriate countermeasures.

Assets are entities in the system that are valuable for its owner. The owner can be an infrastructure provider, a service provider, an application provider or even an end user, depending on the type of asset. The owner (or a representative of the owner) defines a set of rules for accessing these entities and protects them against any access that does not follow said rules. Any violation to these rules indicates a security or functionality problem in the system.

Fog computing is a multi-dimensional system. It can be seen as a group of single Fog nodes, a network of heterogeneous elements or a system that provides different services to its users. We therefore analyse Fog computing with the three different perspectives:

- Device level: each Fog node is a single device with its own physical and logical properties.
- System level: a Fog platform is a networked system composed of multiple Fog nodes.
- Service level: Fog computing is a system that provides services to its users.

We discuss these different perspectives in the next sections.

5.1 Fog nodes (device level)

A Fog computing platform is a system of Fog nodes connected to each other and also to the Cloud. Each Fog node comprises of physical and logical properties that an attacker may be interested to access/modify.

Physical assets refer to visible and concrete elements in a Fog node that are valuable for their owner. The main physical asset that we identify in a Fog node is the memory. Fog nodes have volatile (e.g., RAM) and non-volatile (e.g., EEPROM) memory which contains code, logs and data. We therefore categorize a Fog node's memory, either volatile or non-volatile as a physical asset.

Logical assets represent software components that may be valuable to an attacker. Every Fog node has an Operating System (OS), an orchestrator, and any number of running container(s). The OS is the first software layer above the hardware of a Fog node. The orchestrator is a software that logically runs on

top of the Fog nodes OS and manages resources and workloads. The operating environment of the orchestrator is dynamic because Fog nodes are autonomous devices that may join and leave the system. Moreover, IoT devices may be mobile, and the workload they imposed on a Fog node may need to be transferred to another Fog node.

The top-level software layer in a Fog node is the application, which usually run within one or more containers. We refer to this layer as the container level. Each container belongs to a specific application; multiple containers may be present and active at the same time.

In the case where the OS and orchestrator code are open source, the confidentiality of this code is usually not considered as an asset. On the other hand, ensuring the integrity of the OS and orchestrator code is important to avoid the execution of malicious code by the Fog node. The internal data of the OS, the orchestrator and the containers may include security settings such as user names. Any such information related to the running software in the Fog nodes, integrity and confidentiality of these data must also be considered as an asset.

Finally, the container code and data are specific to each service provider, so the confidentiality of the container code and data are important and considered as assets. We can thus identify integrity and confidentiality as properties of the logical assets in the Fog nodes.

- Integrity properties considered as assets include:
  - OS code
  - OS data
  - Orchestrator code
  - Orchestrator data
  - Container code
  - Container data
- Confidentiality properties considered as assets include:
  - OS data
  - Orchestrator data
  - Container code
  - Container data

5.2 Fog as a networked system (system level)

A Fog computing platform can also be seen as a networked system, where multiple Fog nodes interact with one another, or with the Cloud layer. In this perspective, the assets may be divided into two main groups:

- Fog-to-Fog communication
- Fog-to-Cloud communication

The communication channel between elements, either between Fog nodes themselves or the communication channel between a Fog node and Cloud are considered as assets. As may systems use the IP address of network elements

as a mean to identify and access them, we identify every node's IP address as an asset. The integrity and authenticity of IP addresses of Fog nodes are the two properties of this asset.

5.2.1 Fog-to-Fog communication

For-to-Fog communication includes all internal communication between the Fog nodes. For example, in applications where IoT devices are mobile, the orchestrator may need to migrate the application's code and data from Fog node to Fog node to maintain proximity between each IoT device and the Fog node serving it. Thus, we identify two major assets, which are the communication channel and IP address of Fog nodes.

Communication channel refers to the physical or logical transmission medium that enables two or more entities to transfer data between each other [58]. As data are transferred between elements using the communication channel, there is a risk of data manipulation, eavesdropping or even data loss while they are transferred in the channel. The communication channel should therefore ensure data integrity, confidentiality and availability.

Integrity means that the data received at the destination node should be exactly the same as the data sent from the source. Data integrity should be preserved because they may be used as the input for some computation or decision making processes.

Confidentiality means that data should be accessible only by the source of data and its legitimate, intended destination. In some cases the data transferred via the channel should be kept secret, because they contain sensitive information. For example users of a remote health care system may grant access to their health data only to their general practitioner, and not to their neighbours or any arbitrary user.

Availability means that the data should be available to the legitimate, intended destination. For example in Industry 4.0 applications, data received from an IoT device may be used for analysis and sending commands to other IoT devices in a production line of the factory. If the data availability of one IoT device is violated, this might affect the correct behaviour of the entire system.

Addresses of entities are considered as assets because they ensure that communicating elements are able to find each other. Every Fog node may contain end-user information, so it should know whom it is communicating with and avoid sending this information to unknown network elements.

5.2.2 Fog-to-Cloud communication

Fog computing can also be seen an extension of Cloud computing with additional resources located close to the IoT and end-user devices [8]. It then includes the presence of a Cloud, which implies Fog-to-Cloud communication. In the networked system perspective on Fog computing, we identify two main assets:

Communication channel between Cloud and Fog The communication channel between Fog and Cloud should ensure data integrity, confidentiality and availability as described in Section 5.2.1.

Addresses of entities should also be protected to ensure that the communicating elements are able to find and recognize each other.

5.3 Fog-provisioned services (service level)

As Fog computing is meant to provide services to its users, those properties that are instrumental to ensure the provision of Fog services must also be recognized as assets. Service availability, user's privacy and user's data are the main assets in the service-level perspective.

5.3.1 Service availability

The availability of the service provided by the Fog infrastructure is an important property of the system. If the user cannot be confident about the availability of the service, this would affect the overall accountability of the system. Thus, we identify service availability as an asset.

5.3.2 User's privacy

One of the important requirements of services is to protect the privacy of their users, in particular in the context of IoT which often has access to private information. User's privacy is therefore another asset in the system-level perspective. We study privacy in three main categories: privacy of location, privacy of identity and privacy of other personal data related to the user. We define user's privacy as the ability of keeping any information related to each individual private, except for agreed use cases in which said data is disclosed to agreed and legitimate data destinations.

Location privacy dictates that the location and mobility of end users should remain private. Location privacy is clearly an asset in a Fog computing system.

Identity privacy dictates that services should remain ignorant of the identity of their users, unless explicitly agreed otherwise. In other words, there should be no mapping between the user's identity and the available data in the service. Thus, we categorize identity privacy as another asset.

Other personal data that belongs to a specific user may also need to remain private and may therefore also be categorized as an asset.

5.3.3 Users' data

The data related to a user which is stored in a Fog node or accessible by a Fog service is an asset. The two main properties of user's data asset are integrity and confidentiality.

Users' data integrity means that users' data which is stored in a Fog service should remain intact. We identify user's data integrity as a property of the user's data asset.

User's data confidentiality means that these data should not be revealed to any entity other than the legitimate, agreed service that the data is meant to be sent to. In this regard, user's data confidentiality is a property of the user's data asset. It should be mentioned that this category of asset is different than the other personal data described in Section 5.3.2. The reason is that, in the privacy context, even an encrypted data, which meets requirements of confidentiality, may still be used to exploit an individual's usage pattern of a service and therefore violate user's privacy.

6 Threat modeling

Threat modelling defines the assumed capabilities of an attacker of the system. This definition depends on the service level where the attack is set up. We can consider three different parts in the architecture:

- The field level, where the IoT devices and smart objects reside.
- The Fog platform level, which controls multiple devices located close to the edge of the network. Note that the Fog level might be the first security layer in an IoT application, especially when the IoT devices are very resource constrained.
- The enterprise and platform level, which reside at the core and where application- and platform-level security measures are applicable.

This architecture implies two attacker models to be considered: the `insider` model at field level, and the `outsider` model at all the mentioned levels.

6.1 Insider model

The `insider` model assumes that the device is in the wild and thus, can be in the hand of the attacker, e.g., a stolen smart watch, a temperature sensor, a home gateway. The adversary can control the device by physically replacing the entire node, or tampering with the hardware of the device. If an IoT device is compromised, then some important assets can be exposed to the adversary. The adversary can also copy the important assets associated with the captured node to a malicious node, and then fake the malicious node as an authorized node to connect to the IoT network or system.

Considering this model, the device must defend itself against a large variety of attacks to extract the assets from the device. If the device can be disconnected from the network, then there is no means to observe remotely at the Fog level whether the device has been tampered with. Some devices may have built-in security but very few can be protected against hardware attacks. From the security and privacy perspective, the field of IoT currently presents two main inherent weaknesses: (i) security is often not considered at the design phase; and (ii) no solution currently offers a complete end-to-end security approach for this kind of devices.

Thus, at the field level, we need to assume in a first step that IoT devices are trusted devices. This implies that the field level must consider only the `outsider` model. As a consequence, the communication between IoT devices and Fog nodes can be considered secure as long as they are encrypted and the keys are correctly stored and manipulated.

6.2 Outsider model

The `outsider` model assumes only logical access to the devices. In this model, the attacker can perform a set of attack remotely as for example:

Malicious code injection attacks where the adversary can control a Fog node or a device in IoT by injecting malicious code into the memory of the node or device, which is denoted as the malicious code injection attack. The

malicious code not only can perform specific functions, but it can also grant the adversary access into the IoT system, and even possibly full control of the IoT system.

Side channel attacks using software like cache attacks, Rowhammer, ClockScrew, etc. This is a consequence of the ability of uploading code in the device (node or IoT). The consequences range from extraction cryptographic materials to privilege escalation.

Eavesdropping where malicious attackers can listen on communication channels to capture transmitting packets. Attackers may thus be able to read the content or the meta-data of the protocol, and possibly infer information about the device or the node. Encryption is a partial solution as the node must first identify itself, such that the corresponding key can be associated for a ciphered communication. This may point out a privacy issue, as a node or a device can be then easily tracked.

Denial-of-Service where an attacker can flood the target node with unnecessary requests to make it unavailable to the users. A Fog node is potentially more vulnerable to denial-of-service attacks compared with a Cloud node as its available resources are limited.

Many attacks of different nature can be set up by an attacker even with only the `outsider` model where the attack must be performed remotely. Nevertheless, the ability to upload malicious programs or downgrade firmware is an important attack surface. Thus, these operations must be strictly controlled through a strong authentication mechanism.

The mentioned attacks are examples to show potential attacks in Fog computing domain. We refer the reader to [43] for more examples of attacks in this domain with a classifications of attacks and countermeasures.

7 Analysis of vulnerabilities and possible countermeasures

We now analyse threats to the Fog computing assets as identified in Section 5. The goal of this section is to bring to the fore the possible threats that can cause security problems for a Fog computing system. It is worth mentioning that threats are potential attacks that can happen to a system because of the existence of unsolved vulnerabilities. Thus, in this section we utilize both terms whenever proper. The reader should note that this vulnerability analysis is generic, and that different Fog computing system implementations may differ widely. For example, the choice of a specific OS which is used in the Fog nodes may create specific vulnerabilities linked to this OS version. We apply our methodology to a generic Fog computing system to show how the methodology works in practice. We follow the three different perspectives used in assets identification in Section 5 for vulnerabilities analysis: device level, system level and service level. For convenience, Table 1 presents a summary of the assets and vulnerabilities at the device level, system level and service level.

Table 1: Identified assets and threats for a generic Fog computing system.

	Assets		Threats
Device Level	Fog node memory		- Access to the memory using the Fog ports - Disturbing memory by electromagnetic emissions - Rowhammer attack
	Integrity of	OS code	- Usage of unauthorized code - Downgrading code - Exposing APIs with no access control
		OS data	
		Orchestrator code	
		Orchestrator data	
		Container code	
		Container data	
	Confidentiality of	OS data	- Root access grant -Hooking OS/Orch. functions
		Orchestrator data	
		Container code	
		Container data	
System Level	Communication Channel	Integrity of Comm. Channel	- Modification attack
		Confidentiality of Comm. Channel	- Eavesdropping
		Availability of Comm. Channel	- DoS or DDos attack
	Address of entities		- Man in the middle attack - Rogue Fog node - Phishing
Service level	Service availability		- Dos or DDos attacks
	Location privacy		- Location attack - Credential attack - Usage pattern attack
	Identity privacy		
	Other personal data privacy		
	Users data integrity		- Code injection to access users data
	User's data confidentiality		

7.1 Fog as a device

At the device level, we analyse the vulnerabilities to assets of Fog nodes, where Fog nodes are regarded as individual elements with physical and logical properties.

7.1.1 Vulnerabilities to physical assets

In our threat model, the attacker is not able to physically access the IoT device or Fog node so she can only perform attacks remotely. However, although we omit this possibility in our analysis, physical control of computation nodes is not impossible. According to a report by the European Union Agency for Network and Information Security (ENISA), the edge of the network is more prone to attacks because of its easier physical accessibility to the attackers [12].

In the following, we list possible threats to the node's memory, which is the only identified physical asset in the Fog node. We treat memory as a homogeneous physical entity and do not zoom into it to reach the specific data it holds. As memory contains data, there are overlaps between memory threats and data threats. Memory read, memory write or memory copy (bulk memory read) are three main categories for the threats that can happen by exploiting memory vulnerabilities.

The Rowhammer attack exploits the compact architecture of modern dynamic random-access memory (DRAM), which can potentially leak their electrical charges to the nearby memory cells and change their content [24]. The attacker can use attack software and stress memory rows in order to cause electrical leakage and affect other memory cells. As this attack needs only software and can have fatal consequences, such as privilege escalation, it is a strong attack vector for DRAM memory types.

Disturbing memory by electromagnetic emissions requires physical access to the attacked devices. As we do not assume a very strong attacker with

access to expensive equipment, we skip those attacks in our vulnerability analysis. Without access to expensive equipment such as electron microscope, the attacker can only use electromagnetic emissions to disturb arbitrary bits of the memory. Such attacks can be detected using integrity checks. On the other hand, the only effective defence against them is to shield the devices against electromagnetic emissions.

Access to the memory using the Fog node's ports may allow an attacker to access the node's memory, read it, change it or clone it. This attack may allow the attacker to build other, more powerful, attacks such as building fake Fog nodes. A countermeasure against this attack is to perform authentication before granting any access to the memory.

7.1.2 Vulnerabilities to logical assets

As mentioned in Section 5, the main logical assets are code and data. The code can belong to the OS, orchestrator or container. Each of the mentioned types of code has its corresponding data. In this section, we investigate possible vulnerabilities for the two properties of both code and data, which are integrity and confidentiality.

Vulnerabilities to OS code integrity: If the OS code of the Fog node is not an authorized version published by a well-known provider, then the code integrity of the OS is already compromised. Installation of applications or an orchestrator from unauthorized sources can also threaten the OS code integrity, because applications or the orchestrator can contain Trojans with the ability to change the OS code. Similar issues can be caused by the use of malicious updates.

A countermeasure to this vulnerability is to use signed codes by well-known code providers. There are however arguably situations in which one is forced to use unauthorized code, because there is no certified code available. Secure boot is another mechanism for detection of unauthorized changes in the OS code.

Vulnerabilities to OS data integrity: The OS data of the Fog node should not be changed out of the set of defined access rules. OS data includes a huge range of valuable information, such as return addresses in stack, stack pointer and indicators of frames in the stack, which are only examples of OS data in the stack part of the OS. It is not easy to determine how many OS data exist which should be protected. One solution is that the security analyst should be aware of the state of the art in the attacks to be able to, at least, check if a system is vulnerable against the most popular and known attacks.

Vulnerabilities to orchestrator code integrity are identical to vulnerabilities to OS code integrity, but applied to the orchestrator.

Vulnerabilities to orchestrator data integrity are identical to vulnerabilities to OS data integrity, but applied to the orchestrator.

Vulnerabilities to container code integrity: The same reasoning as for the integrity of OS code applies to the container code. As container code also includes interfaces and APIs which will be accessible out of the Fog node, it can be the target of remote attacks. To defeat such attacks, it is necessary that the

APIs and interfaces include strong access control and authentication methods to control access to the resources of the Fog node.

Vulnerabilities to container data integrity: As Fog nodes may support multi-tenancy, the data of different containers should be protected against access by code running in other containers active on the same Fog node. In addition to this, other vulnerabilities as for the integrity of OS data applies for the container data as well.

Vulnerabilities to OS data confidentiality: Some parts of the OS data such as the OS keys should be kept secret. If an attacker gets access to these data, she will be able to control the Fog node or clone this data and create a fake Fog node. As a countermeasure, these sensitive data should be encrypted by a key which is kept in a hidden memory of the Fog node.

Vulnerabilities to confidentiality of orchestrator data are identical to vulnerabilities to OS data confidentiality, but applied to the orchestrator.

Vulnerabilities to confidentiality of container code: As the application code may not necessarily be open source, it may be important to protect it from being disclosed and possibly reverse engineered.

Vulnerabilities to confidentiality of container data: As mentioned in the integrity of container data, the multi-tenancy property of the Fog nodes reveals a vulnerability related to container data access. In addition to this, other vulnerabilities as for the confidentiality of OS data applies for the container data as well.

In Section 7.4, the vulnerabilities to OS data and OS code will be discussed in more detail as examples of application of our methodology.

7.2 Fog as a system

In this section, we analyse vulnerabilities against Fog computing from a (distributed) system perspective. As we categorized the assets of Fog computing in the system level perspective into three different levels, we keep the same classification for the vulnerability analysis.

7.2.1 Vulnerabilities to Device-Fog communication

According to our `outsider` attack model in Section 6, we assume that the communication between a device and the Fog is secure.

7.2.2 Vulnerabilities to Fog-Fog communication

We now discuss vulnerabilities to the communication channel between the Fog nodes. We split these vulnerabilities in two main categories: the communication channel and the address of entities.

Vulnerabilities to communication channel between Fog nodes may be pof the following kinds:

- Modification attack: The message transferred between two Fog nodes can be modified intentionally or unintentionally by a malicious entity while in transit. A simple countermeasure may be to rely on SSL-based communications so the integrity of communications is protected.

- Eavesdropping: The message can be eavesdropped if it was not encrypted for the recipient of the message. SSL-based communication also protects the confidentiality of messages in the communication channel.
- Denial of service: The communication channel may be unavailable because of the high volume of request sent by a malicious entity in the network. Thus, the channel is congested by arbitrary data and may not be available for the real requests.

Fake addresses attack: While Fog nodes communicate with each other, they exchange data which may contain user data. If the two communicating entities do not control the addresses they are sending data to, these data may be sent to a fake Fog node. **Man in the middle attack**, **rogue Fog node attack** and **phishing** are attacks that can exploit this vulnerability in Fog system. Thus, Fog nodes should perform strong authentication between each other before exchanging any information. This authentication may be managed by a central element such as Cloud.

7.2.3 Vulnerabilities to Fog-Cloud communication

The vulnerabilities to Fog-Cloud communication are similar to the Fog-Fog communication. We therefore classify them identically in two main categories: the communication channel vulnerabilities and vulnerabilities to the address of entities.

7.3 Fog as a service

In this section, we analyse the vulnerabilities of Fog computing from a service perspective.

7.3.1 Vulnerabilities to service availability

Fog computing provides services to IoT devices and to the Cloud. Some of the applications that are using Fog services could be sensitive to disruptions of service availability, e.g., health monitoring applications. In this regard, any attack that pushes unnecessary requests to the Fog services will threaten the service availability for the real requests.

7.3.2 Vulnerabilities to user privacy

As Fog nodes usually provide service for nearby users, knowing the location of a Fog node also reveals the approximate location of the IoT devices connected to it and hence, of the user(s). Thus, it is important to keep the location of Fog nodes private.

Another threat to user's privacy can be caused by installing applications on the Fog node, which use hard-coded credentials or publicly available credentials. It can mostly happen with open-source applications available on the Internet. As the users may not change the hard-coded credentials, a lot of applications may have similar credentials and may become an easy target for attackers.

There are also cases where the user wants to stay anonymous while using a service. For example, users of a service that helps people with drug addiction

may want to remain anonymous. On the other hand, if the service becomes part of the public health care system, it may need to access the real identity of the users. If a user provides his/her identity to the system to register into this service, the service should guarantee that this identity will not be shared or used for other purposes. If (part of) the user data is stored on Fog nodes and gets stolen by an adversary, there is the possibility of privacy threat, unless the user names are protected by some encryption scheme. This issue can happen at both the IoT and the Cloud levels. In this example, even simply the user name is an asset to be protected. If a service uses on-premise Fog nodes and the Fog nodes are publicly known for belonging to a specific service, even tracking data from user's device to that Fog node would be enough to threaten the user's privacy. In this case, the problem is not about the data that the user sends, but even the existence of a data flow between IoT devices and the Fog nodes could threaten the user's privacy. Moreover, if an attacker is able to analyse user's data flow pattern with Fog node or Cloud, he can have some predictions about user's habits.

7.3.3 Vulnerabilities to user data

As the two properties of user's data assets are user's data integrity and user's data confidentiality, in this section, we discuss possible vulnerabilities to them.

Due to the fact that malicious OS, orchestrator or Fog applications can threaten the integrity or the confidentiality of user's data, it is important to keep them secure. In addition, memory vulnerabilities could also lead to threats against user's data confidentiality and integrity. A generic countermeasure to keep confidentiality and integrity of user's data is to perform authentication before granting access to user's data.

The input data in the form of user's data might be used to perform malicious operations. A family of attacks collectively referred to as code injection, covers various types of attacks such as cross site scripting (XSS)) [54] and SQL injection [25]. These attacks use input data as a gateway to attack a system. In order to mitigate such attacks, the input data should be filtered and checked.

7.4 Examples

In order to show how our methodology can be applied to a concrete Fog computing system, we hereby provide two examples:

- Example 1 (E1): Fog node OS code integrity. As defined in Section 5.1, OS code is identified as an asset of the Fog node. Later, we defined our threat model, which describes the attacker and our assumptions about the environment. In the vulnerability analysis part, we investigate possible methods that could cause attack. In Figure 6, the attack-defense tree shows different paths able to cause an E1 attack. We have also presented a possible countermeasure in the tree, which is secure boot. In secure boot, the hash of all or some part of the OS code is compared to the hash of the OS code installed on the Fog node during the boot. Thus, it can detect changes in the OS code.

- Example 2 (E2): As OS data includes a lot of various data types. In this example we just choose a function return address as an example of the

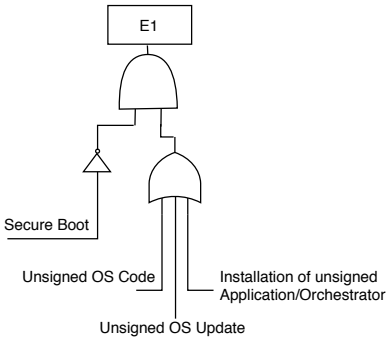

Figure 6: Attack-Defense tree for OS code integrity.

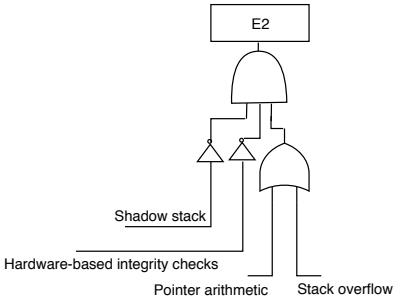

Figure 7: Attack-Defense tree for OS Data integrity (Function return address).

OS data, which is an asset. In the OS, when a function is called, the return address to the caller function is written in the OS stack. If an attacker is able to change the return address, for example to return after the end of an access control procedure, it may obtain access to resources beyond the defined rules and thereby attack the system [1]. There are various techniques to change the return address of a function. Two known techniques are buffer overflow and pointer arithmetic. In order to defend against these attacks, a solution is to build another stack, called shadow stack. A shadow stack, keeps the state of the stack, including the return addresses, and is used to compare the data inside the stack [14] with it. Although using Canary is another possible countermeasure, we do not include it in the attack-tree, because a shadow stack covers it and represents more powerful defence rather than that [17]. There are also hardware-based solutions that protect the return address against attacks. As an example of hardware based solutions, readers are referred to [41]. In Figure 7, the attack-defense tree represents attack to one of the OS data types which is function return address.

8 Discussion and conclusion

This article presents an analytical approach to the security evaluation of Fog computing systems. We started with the Common Criteria methodology, and tailored it it for analysing the security of Fog computing systems. In our methodology, we identified assets of Fog computing and clustered them into device level, system level and service level. For each perspective, we pointed out possible vulnerabilities that the system could face and discussed possible solutions. In order to demonstrate our analysis, we utilized attack-defense trees and represented vulnerabilities with possible countermeasures.

We believe that our approach can provide a valuable tool for both researchers and practitioners. In terms of extensions and enhancements of the current work, a significant contribution would be represented by the creation of a catalogue of possible attacks and defences in Fog computing systems. Yet, it is worth remarking that, as the Fog computing landscape is still in its infancy, the emergence of (de facto or formal) standards and the consolidation of business models for Fog computing infrastructures and services may require frequent updates to said catalogue.

The introduced methodology is based on some assumptions about the security of other elements in an IoT system. For example, "the IoT device is safe" is an assumption that we put in our analysis to simplify the problem of analysing security of a Fog system, while in reality this assumption needs to be satisfied. Although satisfying all of these assumptions is not a trivial work, the authors believe that analysing the security of a complicated system by systematically dividing it to smaller parts, provides helpful guidelines.

The proposed methodology does not provide specific countermeasures for the identified vulnerabilities. We however introduced some directions for tackling said vulnerabilities. As a further work, defining classes of countermeasures will provide more detailed guidance to security analysts. Furthermore, by defining said classes of countermeasures, the security analyst can utilize the attack-defense tree of a Fog system and check if all the identified vulnerabilities are tackled by proper countermeasures introduced beforehand. Involving countermeasures in the analysis can lead to assessing the level of security of a Fog system. As a result, the users of a Fog system can have a sense of the security of the system they using.

In terms of security research topics, our study highlights the relevance of issues related to authentication (between Fog nodes or between Fog and Cloud levels); in this respect approaches towards distributed authentication may have a great potential. Another relevant topic relates to the ability of monitoring and controlling the behaviour of non-certified applications that the user may install and execute: while some solutions devised in distributed systems and Cloud applications may be reused, they require major adaptations to match to the peculiarities of the Fog context.

Acknowledgements

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 765452. The information and views set

out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

References

[1] Abadi, M., Budiu, M., Erlingsson, Ú., and Ligatti, J. Control-flow integrity principles, implementations, and applications. *ACM Transactions on Information and System Security (TISSEC) 13*, 1 (2009).

[2] Abomhara, M., and Køien, G. M. Security and privacy in the internet of things: Current status and open issues. In *2014 international conference on privacy and security in mobile systems (PRISMS)* (2014), IEEE, pp. 1–8.

[3] Alrawais, A., Alhothaily, A., Hu, C., and Cheng, X. Fog computing for the Internet of Things: Security and privacy issues. *IEEE Internet Computing 21*, 2 (2017).

[4] Ammar, M., Russello, G., and Crispo, B. Internet of Things: A survey on the security of IoT frameworks. *Journal of Information Security and Applications 38* (2018), 8–27.

[5] Bistarelli, S., Fioravanti, F., and Peretti, P. Defense trees for economic evaluation of security investments. In *First International Conference on Availability, Reliability and Security (ARES'06)* (2006), IEEE, pp. 8–pp.

[6] Blaze, M., Feigenbaum, J., Ioannidis, J., and Keromytis, A. D. The role of trust management in distributed systems security. In *Secure Internet Programming*. Springer, 1999, pp. 185–210.

[7] Bonomi, F. Connected vehicles, the internet of things, and fog computing. In *The eighth ACM international workshop on vehicular inter-networking (VANET), Las Vegas, USA* (2011), pp. 13–15.

[8] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. Fog computing and its role in the internet of things. In *Proc. MCC workshop on Mobile cloud computing* (2012).

[9] Bouffard, G., and Lanet, J.-L. Reversing the operating system of a java based smart card. *Journal of Computer Virology and Hacking Techniques 10*, 4 (2014), 239–253.

[10] Bouissou, M., and Bon, J.-L. A new formalism that combines advantages of fault-trees and markov models: Boolean logic driven markov processes. *Reliability Engineering & System Safety 82*, 2 (2003), 149–163.

[11] Chen, D., and Zhao, H. Data security and privacy protection issues in cloud computing. In *2012 International Conference on Computer Science and Electronics Engineering* (2012), vol. 1, IEEE, pp. 647–651.

[12] Christina Skouloudi, G. F. Towards secure convergence of cloud and IoT. https://www.enisa.europa.eu/news/enisa-news/towards-secure-convergence-of-cloud-and-iot, Sept. 2018.

[13] Cloud Security Alliance. The treacherous 12 – top threats to cloud computing + industry insights. https://downloads.cloudsecurityalliance.org/assets/research/top-threats/treacherous-12-top-threats.pdf, 2017.

[14] Conti, M., Crane, S., Davi, L., Franz, M., Larsen, P., Negro, M., Liebchen, C., Qunaibit, M., and Sadeghi, A.-R. Losing control: On the effectiveness of control-flow integrity under stack attacks. In *Proc. ACM SIGSAC Conference on Computer and Communications Security* (2015).

[15] Council of European Union. Council regulation (EU) no 269/2014, 2014. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L:2016:119:FULL&from=EL.

[16] Criteria, C. Common criteria for information technology security evaluation, part 1: Introduction and general model (2009) version 3.1, revision 3 (ccmb-2009-07-001), 2009.

[17] Dang, T. H., Maniatis, P., and Wagner, D. The performance cost of shadow stacks and stack canaries. In *Proc. ACM Symposium on Information, Computer and Communications Security* (2015).

[18] Esposito, C., Castiglione, A., Pop, F., and Choo, K.-K. R. Challenges of connecting edge and cloud computing: A security and forensic perspective. *IEEE Cloud computing 4*, 2 (2017).

[19] Farhadi, M., Miorandi, D., and Pierre, G. Blockchain enabled fog structure to provide data security in iot applications. *arXiv preprint arXiv:1901.04830* (2019).

[20] Flexera. RightScale state of the cloud report. https://www.rightscale.com/lp/state-of-the-cloud, 2019.

[21] Frustaci, M., Pace, P., Aloi, G., and Fortino, G. Evaluating critical security issues of the iot world: Present and future challenges. *IEEE Internet of Things Journal 5*, 4 (2017), 2483–2495.

[22] Ganguli, S., and Friedman, T. IoT technology disruptions: A Gartner trend insight report. Tech. rep., Gartner Inc., 2017.

[23] Gou, Q., Yan, L., Liu, Y., and Li, Y. Construction and strategies in iot security system. In *2013 IEEE international conference on green computing and communications and IEEE internet of things and IEEE cyber, physical and social computing* (2013), IEEE, pp. 1129–1132.

[24] Gruss, D., Maurice, C., and Mangard, S. Rowhammer.js: A remote software-induced fault attack in javascript. In *Proc. Intl. Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (2016).

[25] Halfond, W. G., Viegas, J., Orso, A., et al. A classification of SQL-injection attacks and countermeasures. In *Proc. IEEE International Symposium on Secure Software Engineering* (2006).

[26] IEEE Standards Association. IEEE 1934-2018 – IEEE standard for adoption of OpenFog reference architecture for fog computing, 2018. `https://standards.ieee.org/standard/1934-2018.html`.

[27] Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., and Qiu, D. Security of the internet of things: perspectives and challenges. *Wireless Networks 20*, 8 (2014), 2481–2501.

[28] Khan, M. A., and Salah, K. Iot security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems 82* (2018), 395–411.

[29] Khan, S., Parkinson, S., and Qin, Y. Fog computing security: a review of current applications and security solutions. *Journal of Cloud Computing 6*, 1 (Aug 2017).

[30] Kindervag, J. Build security into your network's dna: The zero trust network architecture. *Forrester Research Inc* (2010).

[31] Lamport, L., Shostak, R., and Pease, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS) 4*, 3 (1982), 382–401.

[32] Li, H., and Singhal, M. Trust management in distributed systems. *Computer 40*, 2 (2007), 45–53.

[33] Liu, C., Yang, C., Zhang, X., and Chen, J. External integrity verification for outsourced big data in cloud and iot: A big picture. *Future generation computer systems 49* (2015), 58–67.

[34] Liu, H., Zhang, Y., and Yang, T. Blockchain-enabled security in electric vehicles cloud and edge computing. *IEEE Network 32*, 3 (2018).

[35] McKinsey. The Internet of Things: Five critical questions. `https://www.mckinsey.com/industries/high-tech/our-insights/the-internet-of-things-five-critical-questions`, Aug. 2015.

[36] Miorandi, D., Sicari, S., De Pellegrini, F., and Chlamtac, I. Internet of things: Vision, applications and research challenges. *Ad hoc networks 10*, 7 (2012), 1497–1516.

[37] Moffett, J. D. Security & distributed systems. 1995.

[38] Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M. A., Choudhury, N., and Kumar, V. Security and privacy in fog computing: Challenges. *IEEE Access 5* (2017).

[39] Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G., and Ghani, N. Demystifying IoT security: an exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations. *IEEE Communications Surveys & Tutorials* (2019).

[40] OpenFog Consortium. OpenFog reference architecture for fog computing. https://www.openfogconsortium.org/ra/, 2017.

[41] Ozdoganoglu, H., Vijaykumar, T., Brodley, C. E., Kuperman, B. A., and Jalote, A. SmashGuard: A hardware solution to prevent security attacks on the function return address. *IEEE Transactions on Computers 55*, 10 (2006).

[42] Pang, H., and Tan, K.-L. Authenticating query results in edge computing. In *Proc. ICDE* (2004).

[43] Puthal, D., Mohanty, S. P., Bhavake, S. A., Morgan, G., and Ranjan, R. Fog computing security challenges and future directions [energy and security]. *IEEE Consumer Electronics Magazine 8*, 3 (2019), 92–96.

[44] Puthal, D., Nepal, S., Ranjan, R., and Chen, J. Threats to networking cloud and edge datacenters in the internet of things. *IEEE Cloud Computing 3*, 3 (2016), 64–71.

[45] Roman, R., Lopez, J., and Mambo, M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems 78* (2018), 680–698.

[46] Roy, A., Kim, D. S., and Trivedi, K. S. Attack countermeasure trees (act): towards unifying the constructs of attack and defense trees. *Security and Communication Networks 5*, 8 (2012), 929–943.

[47] Schneier, B. Attack trees. *Dr. Dobb's journal 24*, 12 (1999), 21–29.

[48] Sicari, S., Rizzardi, A., Grieco, L. A., and Coen-Porisini, A. Security, privacy and trust in internet of things: The road ahead. *Computer networks 76* (2015), 146–164.

[49] Simmonds, A., Sandilands, P., and Van Ekert, L. An ontology for network security attacks. In *Asian Applied Computing Conference* (2004), Springer, pp. 317–323.

[50] Steiner, J. G., Neuman, B. C., and Schiller, J. I. Kerberos: An authentication service for open network systems. In *Usenix Winter* (1988), Citeseer, pp. 191–202.

[51] Stellios, I., Kotzanikolaou, P., Psarakis, M., Alcaraz, C., and Lopez, J. A survey of IoT-enabled cyberattacks: Assessing attack paths to critical infrastructures and services. *IEEE Communications Surveys & Tutorials 20*, 4 (2018), 3453–3495.

[52] Stojmenovic, I., Wen, S., Huang, X., and Luan, H. An overview of fog computing and its security issues. *Concurrency and Computation: Practice and Experience 28*, 10 (2016), 2991–3005.

[53] Tanenbaum, A. S., and van Steen, M. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[54] Vogt, P., Nentwich, F., Jovanovic, N., Kirda, E., Kruegel, C., and Vigna, G. Cross site scripting prevention with dynamic data tainting and static analysis. In *Proc. NDSS* (2007).

[55] Wazid, M., Das, A. K., Hussain, R., Succi, G., and Rodrigues, J. J. Authentication in cloud-driven IoT-based big data environment: Survey and outlook. *Journal of Systems Architecture 97* (2019), 185–196.

[56] Weissberger, A. IDC directions 2017: IoT forecast, 5G and related sessions. `http://techblog.comsoc.org/2017/03/04/idc-directions-2017-iot-forecast-related-sessions/`, Mar. 2017.

[57] Wikipedia. Common criteria, 2019. `https://en.wikipedia.org/wiki/Common_Criteria`.

[58] Wikipedia. Communication channel, 2019. `https://en.wikipedia.org/wiki/Communication_channel`.

[59] Xu, X., Chen, Y., Zhang, X., Liu, Q., Liu, X., and Qi, L. A blockchain-based computation offloading method for edge computing in 5g networks. *Software: Practice and Experience*.

[60] Yi, S., Qin, Z., and Li, Q. Security and privacy issues of fog computing: A survey. In *Proc. International conference on wireless algorithms, systems, and applications* (2015).

[61] Zhang, L., Jia, W., Wen, S., and Yao, D. A man-in-the-middle attack on 3g-wlan interworking. In *Proc. IEEE Conference on Communications and Mobile Computing* (2010).

[62] Zhang, Z.-K., Cho, M. C. Y., Wang, C.-W., Hsu, C.-W., Chen, C.-K., and Shieh, S. IoT security: ongoing challenges and research opportunities. In *2014 IEEE 7th international conference on service-oriented computing and applications* (2014), IEEE, pp. 230–234.

[63] Zhao, K., and Ge, L. A survey on the Internet of things security. In *2013 Ninth international conference on computational intelligence and security* (2013), IEEE, pp. 663–667.