future internet

MDPI

Article

Modeling and Security in Cloud Ecosystems

Eduardo B. Fernandez [1,*], Nobukazu Yoshioka [2], Hironori Washizaki [3] and Madiha H. Syed [1]

[1] Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA; msyed2014@fau.edu

[2] Center for Global Research in Advanced Software Science and Engineering (GRACE), National Institute of Informatics, Tokyo 101-8430, Japan; nobukazu@nii.ac.jp

[3] Department of Computer Science and Engineering, Waseda University, Tokyo 169-8555, Japan; washizaki@waseda.jp

* Correspondence: fernande@fau.edu or ed@cse.fau.edu; Tel.: +1-561-297-3466

Academic Editors: Eduardo Fernández-Medina Patón, David G. Rosado and Dino Giuli
Received: 18 January 2016; Accepted: 8 April 2016; Published: 20 April 2016

Abstract: Clouds do not work in isolation but interact with other clouds and with a variety of systems either developed by the same provider or by external entities with the purpose to interact with them; forming then an ecosystem. A software ecosystem is a collection of software systems that have been developed to coexist and evolve together. The stakeholders of such a system need a variety of models to give them a perspective of the possibilities of the system, to evaluate specific quality attributes, and to extend the system. A powerful representation when building or using software ecosystems is the use of architectural models, which describe the structural aspects of such a system. These models have value for security and compliance, are useful to build new systems, can be used to define service contracts, find where quality factors can be monitored, and to plan further expansion. We have described a cloud ecosystem in the form of a pattern diagram where its components are patterns and reference architectures. A pattern is an encapsulated solution to a recurrent problem. We have recently expanded these models to cover fog systems and containers. Fog Computing is a highly-virtualized platform that provides compute, storage, and networking services between end devices and Cloud Computing Data Centers; a Software Container provides an execution environment for applications sharing a host operating system, binaries, and libraries with other containers. We intend to use this architecture to answer a variety of questions about the security of this system as well as a reference to design interacting combinations of heterogeneous components. We defined a metamodel to relate security concepts which is being expanded.

Keywords: software ecosystems; architecture patterns; cloud computing; reference architectures; security patterns; systems security

1. Introduction

Due to their convenience and relative low cost, cloud computing systems have become very successful in attracting small and medium businesses and academic institutions. Their emergence has brought a variety of products or services that complement or extend their basic services. Some clouds are also connected to other (maybe more specialized) clouds and may support cyber-physical systems, as well as a variety of user devices or intelligent machines. We have now what is called a software ecosystem. Ecosystems were initially defined from a biological perspective: systems formed by the interaction of a community of organisms with their physical environment (http://wordnetweb.princeton.edu/perl/webwn). The term was later applied to software systems: "a collection of software systems, which are developed and co-evolve in the same environment" [1]. For software product lines: "An ecosystem is the expansion of a software product line architecture to include systems outside the product which interact with the product" [2]. Software ecosystems are

advantageous to suppliers who can offer a larger variety of products or services, and to consumers who can find more products to help them reach their business goals. Companies such as Microsoft and Apple have been building ecosystems around their products for many years and more recently telecommunications companies such as Cisco [3], Ericsson [4], and others, have been building extensive software ecosystems.

Software ecosystems also include economic and socio-technical aspects [5]; but those aspects are not considered here. In cloud ecosystems their complementary systems may not be produced by the same vendor and may use different protocols, although able to interact with other products in the ecosystem. These complementary systems are a growing set, where new types of products or services constantly appear and provide some useful functions for some types of users. Some of those products may be housed in real devices but they also can be virtualized and executed in any system, including standard processors or cyber-physical systems.

Several authors, e.g., [5–7] have indicated that the lack of reference architectures or other abstract models inhibit the wider adoption of software ecosystems and deny the possibility of exploiting their full potential. This need motivates our work: Architectural models based on patterns are a powerful representation when building or using cloud ecosystems and similar complex systems [8]. A pattern is an encapsulated solution to a software problem. We started by describing models for clouds [9] and then expanded them to describe cloud ecosystems [10]. After a careful search we have not found similar models (see Section 4). We have expanded our initial ecosystem to cover fog systems [11] and containers [12]. Most of the components of this system have already been modeled as patterns by ourselves but some are missing, we identify here the new patterns we need. We present here a systematic method to build ecosystem models for clouds, using patterns and reference architectures, which is our main contribution. We discuss the value of these models with respect to several objectives, which are useful for understanding and analyzing significant aspects of the ecosystem. We do not claim completeness, an ecosystem is open-ended and our model will continue growing when we identify more functions and their patterns.

As indicated, an important value of our model is for analyzing security aspects of an ecosystem and we intend to answer several security questions with it. Our long term objective is to develop a holistic security view across all the elements of the ecosystem and we are starting with security in a fog controlling a variety of devices. In particular, we intend to define policies on what data from the cloud can be sent to the devices or what data from the devices can be sent to the cloud. Devices may contain sensitive data whose disclosure would affect the privacy of users. The fog platform, itself, contains data and the access of that data should conform to cloud policies, as well as device policies; that is, security constraints in the cloud and devices should propagate across up and down levels. Without a unified view it is very difficult to integrate systems which may have their own security policies. As part of this emphasis we show a security metamodel that is being extended to include more concepts.

This work is organized as: Section 2 defines some necessary concepts; Section 3 presents related work on ecosystems and fog security, while Section 4 describes the cloud ecosystem as a pattern diagram, followed by a description of three of its patterns to illustrate their contents. Section 5 considers the use of security metamodels to complement our architectural models when dealing with security aspects. Validation of the models is discussed in Section 6, which also emphasizes possible applications for them Section 7 considers some security issues. We end with conclusions in Section 8.

2. Background

A *pattern* is a solution to a recurring problem in a specific context. Software patterns are categorized as analysis [13], design [14], architecture [15], and security patterns [8]. *Abstract patterns* describe a basic semantic aspect while *Abstract Security patterns* (ASPs), realize one or more security policies able to control (stop or mitigate) a threat or comply with a security-related regulation or institutional policy [16]. Patterns are described using a template composed of a set of sections.

A problem section describes a problem and the forces that constrain and define guidelines for its solution, e.g., "overhead must be reasonable". Pattern solutions are usually described using modeling languages such as the Unified Modeling Language (UML), maybe combined with formal languages such as the Object Constraint Language (OCL). UML diagrams may include class, sequence, state, and activity diagrams. A set of consequences indicate how the pattern solved the specific problem and what are the advantages and disadvantages of using it; *i.e.*, how well the forces were satisfied by the solution. An implementation section provides hints on how to use the pattern in an application. A section on "known uses" lists real systems where this solution has been used previously, *i.e.*, a pattern is an abstraction of a good practice. A section on related patterns indicates patterns that complement or provide alternative solutions to the one in this pattern. A pattern embodies the knowledge and experience of software developers and can be reused in new applications; carefully-designed patterns implicitly apply good design principles. Patterns are also good for communication between designers and to evaluate and reengineer existing systems. While initially developed for software, patterns can describe hardware, physical entities, and combinations of these. Pattern solutions are suggestions, not plug-ins or software components. A *compound* pattern is composed of two or more simpler patterns.

A *Reference Architecture* (RA) is an abstract software architecture, based on one or more domains, with no implementation aspects [12,17,18]. An RA should define the fundamental concepts of a system expressed as ASPs and the interactions among these units. An RA should be reusable, extendable, and configurable; that is, it is a kind of composite pattern for whole architectures and it can be instantiated into a concrete software architecture by adding platform aspects [17]. In addition to class and sequence diagrams, an RA may include a set of use cases (UC), and a set of Roles (R) corresponding to its stakeholders (actors). Types of RAs include those for the technology domain (describe platforms and other design artifacts [18]), application domain (describe different types of applications), and problem domain (similar to domain models, but oriented to software). After adding security patterns to neutralize identified threats in an RA we have a *Security Reference Architecture* (SRA), and we have just produced a SRA for clouds [9]. We can also add compliance patterns to produce a *Compliance Reference Architecture* [19].

Policies are high-level guidelines defining how an institution conducts its activities in its business, professional, economic, social, and legal environment. The institution security policies include laws, rules, and practices that regulate how an institution uses, manages and protects resources. *Regulations* are local or government policies that must be reflected in the implemented system. Industry regulations are called *standards*. Policies, regulations, and standards can be described using UML models.

We describe the relationships between patterns using a *pattern diagram* [15]. In a pattern diagram rounded rectangles represent patterns and arrows indicate the contribution of a pattern to another, e.g., a Container provides virtual environments to PaaS in Figure 1.

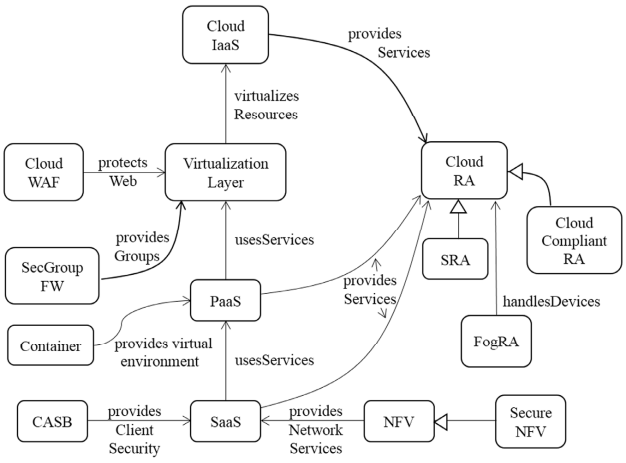

Figure 1. A cloud ecosystem (showing its latest additions).

3. Related Work

As indicated earlier, the word ecosystem has a variety of interpretations. [7] discusses an Internet of Things (IoT) ecosystem from a business perspective. A business ecosystem is "the network of buyers, suppliers and makers of related products or services plus the socio-economic environment, including the institutional and regulatory framework." Our definition is more restricted focusing on the products themselves but also considering policies defining institution or regulations, as well as industry standards that apply to the products. [7] indicates that ecosystems usually have a *hub*, or central product or service; we have a similar concept in our ecosystem. A specialized type of ecosystem is the *learning ecosystem* [20–22]. This type of ecosystem is quite different from ours in that it is a set of homogeneous units with a predefined purpose and a well-defined architecture; ours are heterogeneous systems with an evolving architecture. These ecosystems do not use our definition of pattern and do not use UML models to describe them. The only paper with more abstract models of ecosystems is [6], which presents a model oriented to the business (functional) aspects of ecosystems; in contrast our models emphasize the structural aspects of the interconnection of products.

While there is a good amount of work on ecosystems from the point of view of software architecture, e.g., [2], we have found only a few examples of cloud ecosystems. The US National Institute of Standards and Technology (NIST) described an ecosystem for its Cloud Reference Architecture [23] and, later, an ecosystem for its Security Reference Architecture [24]. However, they included only a Broker to let users access multiple clouds, an Auditor to check compliance with regulations, and a Communications Provider; which is a rather meager set of external functions. They describe their models with words and block diagrams, which we consider not precise enough to guide developers and researchers. The Open Group has a web site with their cloud model [25]. This includes a UML model for the main blocks and a table describing the components involved. There are no UML models for the components and they consider the same main components of the NIST model. A blog presents some ideas about models for cloud ecosystems [26]; however, the models are loose and shown as block diagrams. The Open Services Gateway initiative (OSGi) also has some general ideas about ecosystems [27] but nothing specific about clouds. Ecosystems can also be seen as systems of systems and work on that topic may apply to them. Work on software product line architectures is also relevant [28], as well as work on cloud security requirements [29], but these works do not attempt to model a complete ecosystem.

Given the novelty of fog computing there is little work on its security aspects. Three papers survey general security aspects:

Ref. [30] evaluates business, professional, and government structures and practices for improving IoT security, proposing some structures and rules. The paper focuses mainly in economic and political aspects and does not provide any technical details.

Ref. [31] surveys fog applications and general security issues. They analyze in detail a man-in-the-middle attack, where gateways used as fog devices may be replaced by fake ones.

Ref. [32] considers industrial IoT systems, which are cyber-physical systems where attacks can have very serious effects. The paper surveys some of these attacks and provides a few solutions.

The only paper that discusses fog computing security models and architectures is [33]. The paper proposes a policy-driven security management approach, considering the implementation of a policy enforcement system (Reference Monitor [8]) that can enforce eXtensible Access Control Markup Language (XACML) rules; however, they do not consider rights inheritance or propagation of rights in the ecosystem. We see our work as expanding this work across the ecosystem.

Some works consider the use of cloud computing to support cyberphysical systems (CPSs) [34]. This work is more general than fog computing but some of their ideas are clearly related to this architecture. Along these lines, our work on modeling and finding generic threats for CPSs is relevant and we will apply it to fog computing [35]. It is clear that the security of the combination cloud/Internet of Things is an area that requires more work and where our models appear promising.

4. A Model for a Cloud Ecosystem

We show first the complete ecosystem in the form of a pattern diagram that relates the contributions of the patterns with respect to each other. After building this pattern diagram we need to build detailed models for the components. We show three examples of the patterns in the ecosystem and a part of the SRA. None is shown completely, the idea is to show their main functions, the complete descriptions can be found in their respective references. These descriptions follow the standard pattern template of [15]. Missing parts include sections on Problem, Forces, Consequences, Implementation, Known uses, and Related Patterns. The idea here is that we can build patterns for every participant in the ecosystem, which provides a unified view of the complete system.

4.1. Pattern Diagram of the Ecosystem

Figure 1 shows our current cloud ecosystem. The Cloud Reference Architecture (Cloud RA) is the main pattern that defines the ecosystem (the hub) [9]. As indicated, this can be converted into a Cloud Security RA (Cloud SRA) by adding security patterns to control its identified threats. The Cloud SRA includes, among others, patterns for Authentication, Authorization, and Logging [8]. We just defined a cloud HIPAA-compliant RA [19]. Threats can be enumerated in several ways and we use an approach based on activities in an activity diagram describing its use cases [8]. Cloud Web Application Firewalls and Security Group Firewalls provide filtering functions that can be provided as services through NFVs (see below) or on their own.

The service layers of a cloud are themselves compound patterns and we have written patterns for IaaS, PaaS, and SaaS [36]. They describe the services sold by the cloud provider. Telecommunication companies have discovered that they can provide services to their customers by building their networks as services rented from some cloud provider [4]. The provision of network functions using virtualization, Network Functions Virtualization (NFV), is a network architecture where functions, such as load balancers, firewalls, Intrusion Detection Systems (IDSs), and accelerators, are built in software and offered as services. Each Virtualized Network Function (VNF) may use one or more virtual machines or containers running different software. Typically, NFVs come with some security mechanisms but which ones depend on the vendor. To make the model more flexible we have a pattern for the NFV without security and a derived pattern for the Secure NFV [37].

Cloud Access Security Brokers (CASBs) are security enforcement points between consumers and service providers that apply security controls to access cloud services, usually SaaS services [38]. They may also control access to internal company resources. Security controls may include authentication (credentials and passwords), authorization policy enforcement, intrusion prevention, antimalware filters, security logging/auditing, and encryption. There is no pattern yet for the Virtualization Layer, although we defined a pattern for a Virtual Machine Operating System [8]. An important lower-level pattern for this function is an OpenStack pattern, part of a hierarchy of IaaS patterns [39].

As indicated, analyzing threats and neutralizing them with patterns we arrived to secure units of the SRA. Figure 2 shows a class model for the secure Virtual Machine (VM) image repository system. The Virtual Machine Image Repository holds a set of Virtual Machine Images (VMIs) that can be used to instantiate virtual machines. The Reference Monitor uses a filter that scans all VM images before being published or retrieved. The Authenticator is an instance of the Authenticator Pattern that allows the Reference Monitor to authenticate the users who can publish or retrieve images if the Authorizer authorizes them. The Reference Monitor pattern enforces the authorization rights defined in the Authorizer. The Security Logger/Auditor keeps track of accesses to the repository. Our latest additions are the Container and the Fog Reference Architecture, described below.

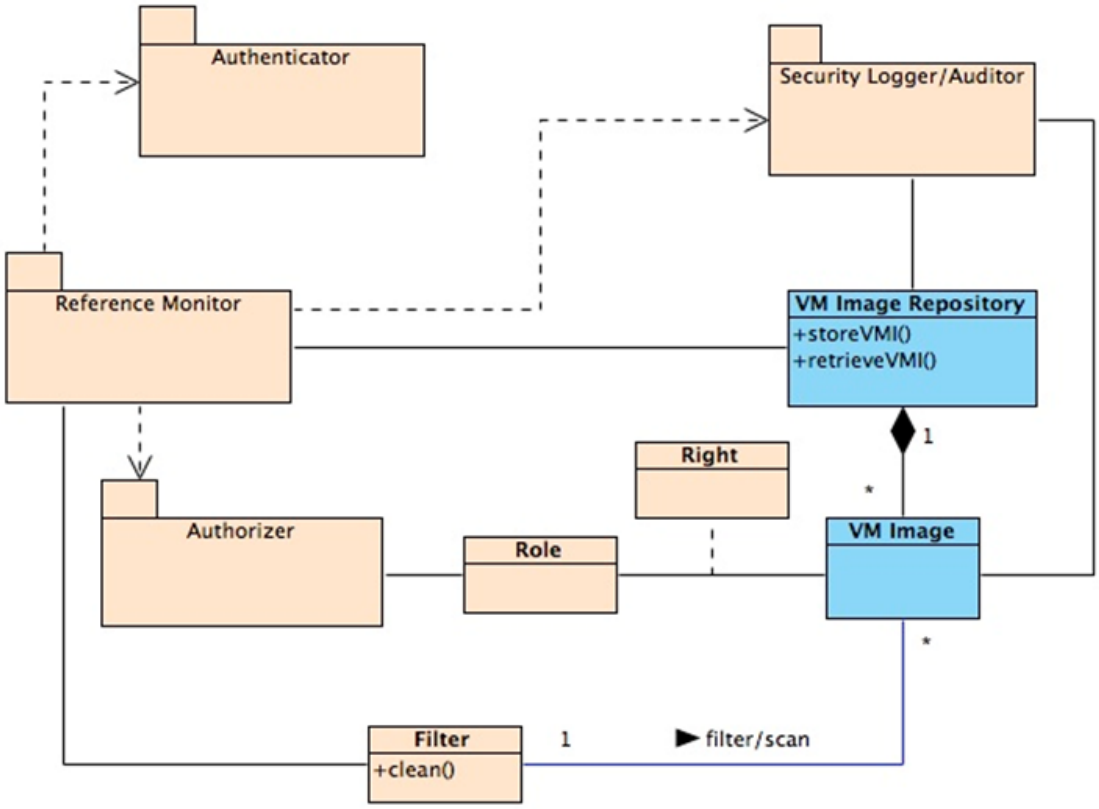

Figure 2. Secure Virtual Machine Image (VMI) Repository System.

4.2. Some of the Patterns in the Ecosystem

4.2.1. Cloud Access Security Broker

Intent

Cloud Access Security Brokers (CASBs) are security enforcement points between consumers and service providers that apply security controls to the consumer's users to access cloud services, usually SaaS services. They may also control access to internal company resources. Security controls may include authentication (credentials and passwords), authorization policy enforcement, intrusion prevention, antimalware filters, security logging/auditing, and encryption.

Solution

Using an intermediary system, called a CASB, which provides security controls (authentication and authorization), can monitor the use of services by users, and can perform malware detection when users access cloud applications. Additionally, other services such as performance, identity, and search can be provided. Figure 3 shows the class diagram of the CASB. Consumers (users) request services through the Broker which, in turn, gets them from one of the Service Providers. The Broker includes a set of security mechanisms such as a SecurityLogger/Auditor, an Authorizer, an Authenticator, an Encryptor, and maybe others. Consumers and CASBs can be mutually authenticated. The CASB enforces rights for the consumers when they try to access an application. Internal Resources (applications) can also be controlled by the CASB. An Identity Federation provides identifiers for consumers and SPs to support authentication. Figure 4 shows the sequence diagram for the use case "access an application service": a consumer requests a service to the CASB, which invokes an authentication protocol, when authenticated the consumer can access the service if authorized for it; this interaction is logged.

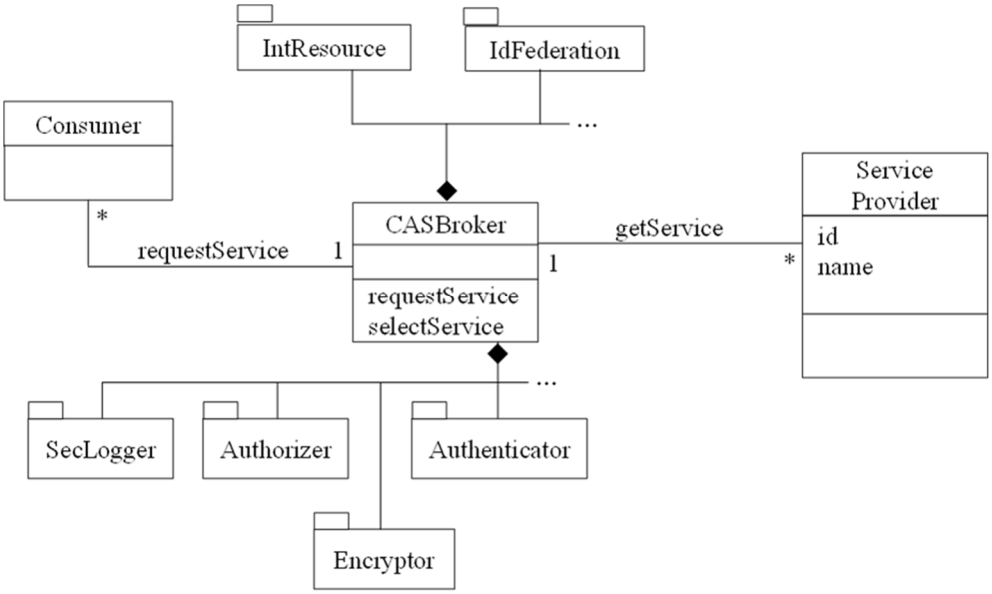

Figure 3. Class diagram of the CASB pattern.

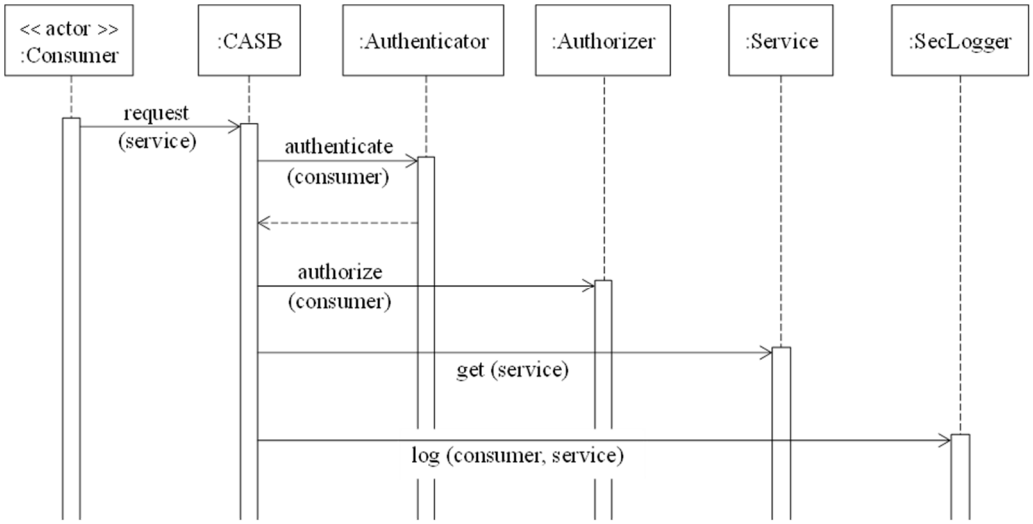

Figure 4. Sequence diagram for the use case "Access an application service".

The CASB enforces institution policies in any access as well as protecting against malware. In other words, the CASB is an extended Reference Monitor [8].

4.2.2. The Software Container

Intent

A Software Container provides an execution environment for applications sharing a host operating system, binaries, and libraries with other containers. Containers are lightweight, portable, extensible, reliable, and secure.

Solution

Provide a runtime environment that can support the isolated execution of applications on a shared host operating system (OS); this is a Software Container (Figure 5). They may also share binaries

and libraries with other containers. Containers provide isolated execution and extensible services to the application.

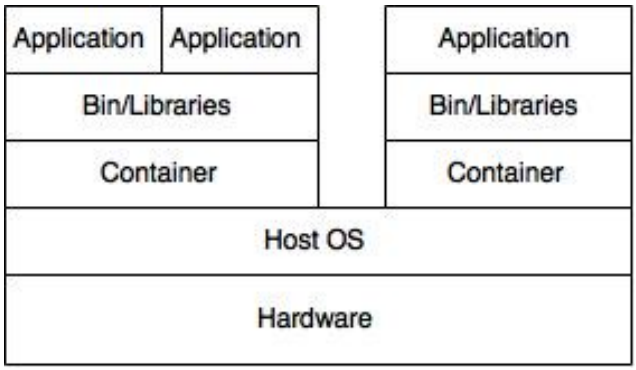

Figure 5. Two containers sharing one OS.

Figure 6 shows the class diagram for this pattern. A Container controls a set of Applications sharing a host OS that provides a set of Resources. An Interceptor mediates the services provided to the application by the container. Applications hosted in containers can be accessed remotely through Proxies, where the Container acts as a broker. The client interacts with the Application Proxy, which represents the application. The application interacts with the Client Proxy, which represents the client. The Container provides a set of Services to the applications. Container Images are stored in image repositories.

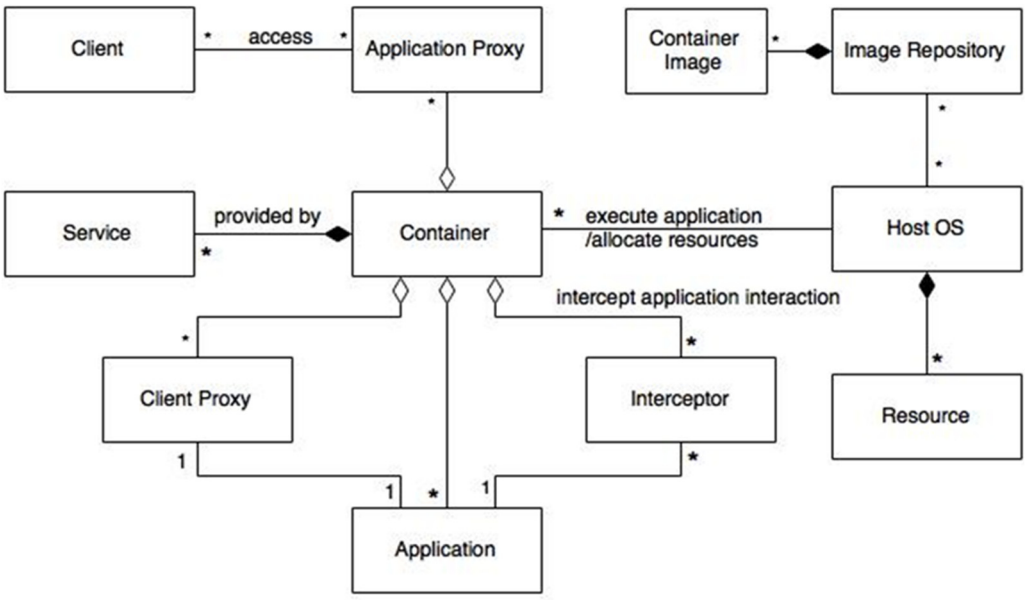

Figure 6. Class diagram of the container pattern.

4.2.3. Fog Computing

Our current work also includes modeling fog computing. Fog Computing is a highly virtualized platform that provides compute, storage, and networking services between end devices and cloud computing data centers [40–42]. Fog computing systems are key systems for the Internet of Things, they can control for example smart grids or traffic lights [31,43]. We have completed a pattern for fog computing [11]. Figure 7 shows its class diagram. The Fog is a collection of several distributed tiny

clouds called Fog Nodes. They can be resource-rich servers, routers, access points, mobile devices, *etc*. A Fog Node has resources which include hardware (compute, networking, and storage) capabilities. The nodes provide real-time analytics using an Analytics Engine. Applications can be hosted in the fog nodes using virtualization, provided by a Virtual Machine Monitor (VMM), which can create virtual machines (VMs), and/or Containers. A Distributed Database stores both application data and necessary metadata for service orchestration; it also has information about the hardware and software capabilities of nodes, information about the status of fog nodes and services, policies for security, filtering, and configuration. Fog computing uses policy-based service orchestration. A Policy Manager is triggered by service requests and a Decision Maker Engine (Reference Monitor) gathers relevant policies and metadata to decide about requests. Data is transferred between fog nodes, the decision maker, and the various components of the Fog. The Fog also provides Authentication and Authorization services. In addition, services like filtering, aggregation of data, logging, *etc.*, can be provided.

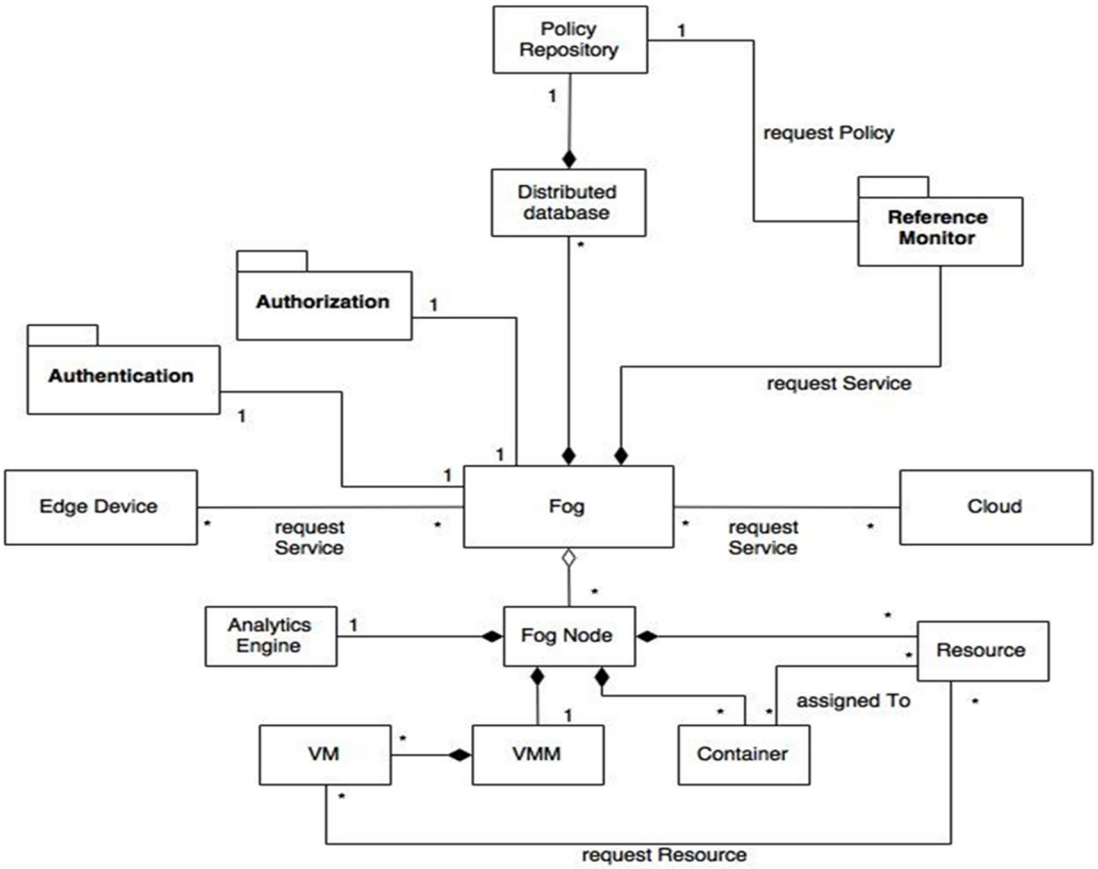

Figure 7. Class diagram of the Fog Computing pattern.

5. Metamodels for Security Concepts

The metamodel of Figure 8 relates the security concepts we are using in the ecosystem. *Threats* take advantage of *Vulnerabilities* that can exist in any cloud service level. Threats come from analysis of Use Cases or from published Threat Lists. Each use case has a set of Roles that describe the participants in the use case. We can stop a threat by removing the initial vulnerability or by controlling its propagation (by removing other vulnerabilities) through the use of a *Security Pattern*. The security pattern to use can be selected from the countermeasures defined in the *Misuse Pattern* which describes the threat. Threats that lead to misuses are the goals of the attacker and are performed through low-level threats in the *Threat List* or directly through a use case operation. Use cases include the roles that participate in the

use case. Some threats can happen in all service levels. For example, a buffer overflow is a language problem and allows escalation of privilege by the attacker operating at any level. Other threats are specific to the level; for example, a financial application can be attacked by taking advantage of a lack of proper authentication in remote access to accounts. If the threat takes advantage of a flaw in an application, it may compromise the security of that application. If the threat affects the IaaS level, it affects all the cloud computations, and if it happens at the PaaS level it can affect all the applications developed or deployed in the cloud.

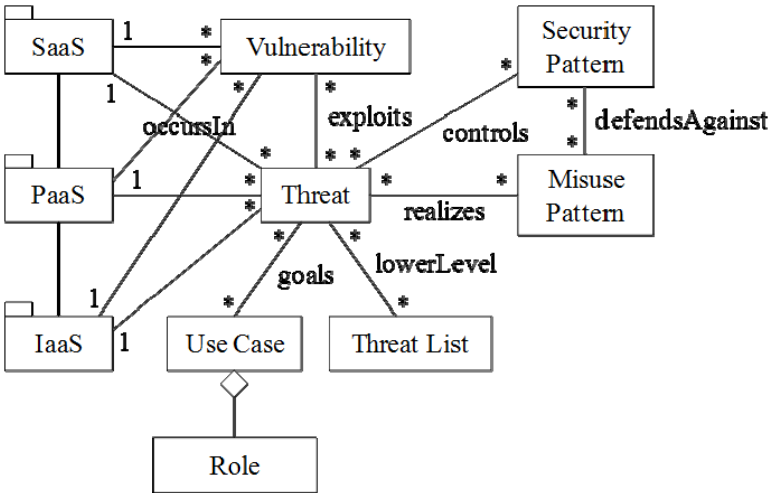

Figure 8. Metamodel for security concepts.

We have started to expand this metamodel by integrating and extending existing cloud security metamodels together with newly added concepts. Figure 9 shows how the metamodel would be used in cloud services development and maintenance. Our metamodel provides a basis for describing and accumulating security and privacy-related knowledge over different layers so that it becomes much easier to select and combine the right patterns and related knowledge for addressing these issues in cloud services. Moreover, designers could refer to the metamodel for designing high-level architectures of cloud service systems in efficient and effective manner. To confirm the usefulness and feasibility of the metamodel, we conducted a case study that describes a cloud security pattern based on the metamodel [44].

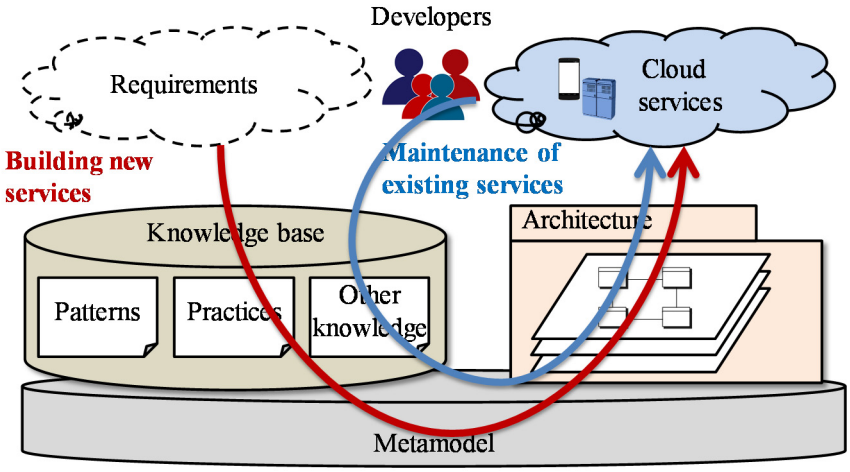

Figure 9. Metamodel and cloud services.

6. Validation of the Models

Ecosystem model validation requires first to validate the patterns used in its construction; then RAs and SRAs must also be validated, followed by the complete ecosystem.

Specific patterns are normally evaluated by submitting them to some pattern conference, e.g., Pattern Languages of Programs (PLoP) or EuroPLoP. In these conferences, a pattern paper is developed with the help of a shepherd and then discussed in a workshop by about ten people. The pattern is then published and exposed for criticism. We have followed this route for all our patterns. Of course, the ultimate evaluation comes when developers use them in their designs but the patterns shown here are too new for this to have happened.

RAs are abstract models and cannot be evaluated with respect to security or performance through experimentation or testing. An RA is similar to a pattern, and it has a similar use; it is a paradigm to guide implementation of new systems or evaluation of existing systems as well as other uses described below. Their evaluation must be based on how well they represent the relevant concepts of the systems they describe, how well they handle potential threats, how complete they are, how precise they are, how they can be applied to the design or evaluation of systems, and how useful they are for other relevant functions. Again, their final validation comes from practitioners who can find them useful and convenient to build concrete architectures. In this respect, we showed in [9] that our SRA included all the functions found in the SRAs of commercial and experimental clouds.

A type of validation (or at least justification) of our ecosystem models comes from describing their advantages:

Control of heterogeneity: The involved components come from different vendors and follow a variety of standards and protocols. An abstract model can unify this heterogeneity and provide a way to understand and analyze global aspects of these systems.

A *holistic security view*. Many authors, e.g., [45,46], emphasize the need to develop secure systems in a holistic way. Systems built piecemeal omit important interactions that may result in vulnerabilities. Enterprises have started to realize the value of holistic approaches to security [3,47]. An ecosystem provides such a holistic view by indicating the places where security mechanisms can be attached and their effect on the functional parts of the architecture. As we did in Figure 2, we can extend the UML model of the functional ecosystem by indicating all the points where threats are neutralized with corresponding security patterns. We can trace the propagation of attacks and study where to place defenses for greater effect. Many threats result of the interaction of different units and cannot be discovered by analyzing each unit in isolation. Privacy rules are defined in the clouds and in devices but we need to make sure that interactions with the components still respect these rules. We elaborate on other security aspects in Section 7.

Other quality factors: Holistic views are useful to combine quality factors such as safety or reliability with security.

Compliance with standards and regulations. An RA can be used to support security standards and regulations, which can be described as policies which in turn can be implemented as patterns and made part of the SRA. The ecosystem helps architects or designers to identify what components of the cloud system are associated with the standard and can be used to comply with the specific rules of the standard. Relating specific regulations to specific security mechanisms can be used to demonstrate compliance [19].

Support for software development [48]. DevOps is an increasingly popular agile process to build software that relies heavily on containers. We explored the use of our Container pattern [12] for this purpose.

Support for virtualization. It is possible to assign the software processes of the ecosystem to a variety of hardware platforms, some or all of which can be virtualized. For example, one can build a virtual drone implemented using two physical devices.

Support for service contracts. In an ecosystem users or institutions may want to rent services involving more than one product. This requires a service level agreement indicating the obligations

of providers and consumers. An ecosystem model can make these services transparent and indicate where compliance would be monitored.

7. Security Issues of Ecosystems

We intend to develop a holistic security view across all the elements of the ecosystem. In particular, we intend to define policies on what data from the cloud can be sent to the devices or what data from the devices can be sent to the cloud. Devices may contain sensitive data whose disclosure would affect the privacy of users. The fog platform, itself, contains data and the access of that data should conform to cloud policies, as well as device policies; that is, security constraints in the cloud and devices should propagate across levels. We intend to perform a systematic analysis of threats, keeping in mind that the introduction of new products may bring new vulnerabilities; each use case of a new product or service must be analyzed to consider possible attacker goals related to it.

Each component may have its own set of policy rules or may inherit from other components; in the latter case there can be conflicts [49]. Fog platforms may communicate with other fogs and may need authorizations to perform actions in remote fog systems. Some of this work has already been done in isolated fogs [33], but it is not clear how these results apply to the new context defined by the components of a cloud ecosystem.

We need to define policies on how data from the cloud can be used in the devices or what data can be sent from devices to the cloud. The fog also sends commands to the devices and devices may send events to the cloud. The fog, itself, has a database and operations that can be accessed from the cloud or the devices. We need to require that devices have a basic separation of computing environments as using two virtual machines or two separated environments. We need to build a detailed security architecture to provide these functions, express it using XACML rules and build several examples. We have developed patterns for XACML [8], which can be applied to describe precisely these rights.

Consider a set of rights for cloud resources (R, O, t), where R is a set of roles, O is a set of objects, and t is an access type. Each right represents what a given actor or role can do with specific resources; for example, in fog managing traffic lights, "Role 'Traffic Light Controller' can activate traffic lights". Those rights can be defined in the fog itself or can be inherited from higher-level rights defined in the cloud. We may need to add new constraints in the form of predicates to the rights in the fog to access devices, e.g., rights such as "Role Traffic Light Maintenance Worker" can activate or deactivate only the lights in a specific zone. The fog may have also new roles. We need to add rights in devices to access fog resources; this may give device users rights to access cloud data and it is related to the Bring Your Own Device (BYOD) problem.

An important security need is management. The functions of such a system include determination of assets, assignment of rights, consideration of regulations, policy definition, and privacy. The metamodel of Section 5 is valuable for this purpose. A model oriented to fulfill the ISO 27000 security regulations in clouds is given in [50], but there is no such a model for ecosystems.

8. Conclusions

Clouds require a variety of complementary components to be effective and cloud ecosystems are starting to become widespread. Some are implicit ecosystems like the combination of clouds and wireless devices. New components, such as containers and fog computing platforms, are appearing. We believe that a holistic, unified treatment is fundamental to handle the complexity of cloud-based systems and allow different kinds of users to analyze the synergy of the total ecosystem. Patterns provide a unified way of representing all the components of the ecosystem and can represent both functional and non-functional aspects. Pattern models are especially useful for handling security and privacy, a unified approach reduces complexity, one of the most important weaknesses used by attackers and can enable analysis of the propagation of threats and data leaks. We show that we can start from patterns, combine them to produce reference architectures, and aggregate those two concepts to build models for complete ecosystems. While we presented these models in terms of

clouds, the methodology is general and can be used to build other types of ecosystems. We have defined architectures for all the current components and we are using this architecture to analyze security and related aspects. We identified several other directions where this ecosystem appears valuable and we intend to study some of them. In the immediate future we are concentrating on security aspects of the fog systems as part of a cloud ecosystem.

Acknowledgments: We thank our reviewers who provided valuable suggestions and references. Part of this work was performed during the visit of Fernandez to Tokyo in March of 2015, supported by the National Institute of Informatics of Japan.

Author Contributions: Eduardo B. Fernandez wrote the paper and proofread it. Sections 4.2.2 and 4.2.3 are based on joint work with Madiha H. Syed. Sections 4.2.1 and 5 are joint work with Nobukazu Yoshioka and Hironori Washizaki.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lungu, M. Reverse Engineering Software Ecosystems. Ph.D. Thesis, University of Lugano, Lugano, Switzerland, 2009.
2. Bosch, J. From software product lines to software ecosystems. In Proceedings of the 13th International Software Product Line Conference, San Francisco, CA, USA, 24–28 August 2009; pp. 111–119.
3. Cisco Corp. Cisco Cloud Strategy for Cloud Providers. Available online: http://www.cisco.com/c/dam/en/us/solutions/collateral/data-center-virtualization/cloud-infrastructure/at-a-glance-c45-730761.pdf (accessed on 15 April 2016).
4. Basilier, H.; Darula, M.; Wilke, J. Virtualization network services—The telecom cloud. *Ericsson Rev.* **2014**, *3*, 2–9.
5. Jansen, S.; Finkelstein, A.; Brinkkemper, S. A Sense of Community: A Research Agenda for Software Ecosystems. In Proceedings of the 31st International Conference on Software Engineering, New and Emerging Research Track, Vancouver, Canada, 16–24 May 2009.
6. Boucharas, V.; Jansen, S.; Brinkkemper, S. Formalizing software ecosystem modeling. In Proceedings of the 1st International Workshop on Open Component Ecosystems, Amsterdam, The Netherlands, 24 August 2009.
7. Mazhelis, O.; Luoma, E.; Warma, H. Defining an Internet-of-Things ecosystem. In Proceedings of the NEW2AN/ruSMART 2012, San Petersburg, Russia, 27–29 August 2012; pp. 1–14.
8. Fernandez, E.B. *Security Patterns in Practice: Building Secure Architectures Using Software Patterns*; John Wiley & Sons: Chichester, UK, 2013.
9. Fernandez, E.B.; Monge, R.; Hashizume, K. Building a security reference architecture for cloud systems. *Requir. Eng.* **2015**. [CrossRef]
10. Fernandez, E.B.; Yoshioka, N.; Washizaki, H. Patterns for Security and Privacy in Cloud Ecosystems. In Proceedings of the 23rd IEEE International Requirements Engineering Conference, Ottawa, ON, Canada, 24–28 August 2015.
11. Syed, M.H.; Fernandez, E.B.; Ilyas, M. A Pattern for Fog Computing. In Proceedings of the VikingPLoP 2016, Leedam, The Netherlands, 7–10 April 2016.
12. Syed, M.H.; Fernandez, E.B. The Software Container pattern. In Proceedings of the 22nd Conference on Pattern Languages of Programs 2015, Pittsburgh, PA, USA, 24–26 October 2015.
13. Fowler, M. *Analysis Patterns: Reusable Object Models*; Addison-Wesley: Boston, MA, USA, 1997.
14. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley: Boston, MA, USA, 1994.
15. Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerland, P.; Stal, M. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*; Wiley: Chichester, UK, 1996.
16. Fernandez, E.B.; Yoshioka, N.; Washizaki, H.; Yoder, J. Abstract security patterns for requirements specification and analysis of secure systems. In Proceedings of the Requirements Engineering track (WER) of the 17th Ibero-American Conference on Soft. Eng. (CIbSE 2014), Pucon, Chile, 23–25 April 2014.
17. Avgeriou, P. Describing, instantiating and evaluating a reference architecture: A case study. Enterprise Archit. Available online: http://www.rug.nl/research/portal/files/14407113/2003EnterpArchitJAvgeriou.pdf (accessed on 15 April 2016).

18. Angelov, S.; Grefen, P.; Greefhorst, D. A framework for analysis and design of software reference architectures. *Inf. Softw. Technol.* **2012**, *54*, 417–431. [CrossRef]
19. Yimam, D.; Fernandez, E.B. Building Compliance and Security Reference Architectures for Cloud Systems. In Proceedings of the IEEE Int. Conf. On Cloud Engineering (IC2E) 2016, Berlin, Germany, 4–8 April 2016.
20. García-Holgado, A.; García-Peñalvo, F.J. The evolution of the technological ecosystems: An architectural proposal to enhancing learning processes. In Proceedings of the First International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'13), Salamanca, Spain, 14–15 November 2013; García-Peñalvo, F.J., Ed.; ACM: New York, NY, USA; pp. 565–571.
21. García-Holgado, A.; García-Peñalvo, F.J.; Rodríguez-Conde, M.J. Definition of a Technological Ecosystem for Scientific Knowledge Management in a PhD Programme. In Proceedings of the Third International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'15), Porto, Portugal, 7–9 October 2015; Alves, G.R., Felgueiras, M.C., Eds.; ACM: New York, NY, USA; pp. 695–700.
22. García-Peñalvo, F.J.; Hernández-García, Á.; Conde-González, M.Á.; Fidalgo-Blanco, Á.; SeinEchaluce Lacleta, M.L.; Alier-Forment, M.; Llorens-Largo, F.; Iglesias-Pradas, S. Learning services-based technological ecosystems. In Proceedings of the Third International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'15), Porto, Portugal, 7–9 October 2015; Alves, G.R., Felgueiras, M.C., Eds.; ACM: New York, NY, USA; pp. 467–472.
23. National Institute of Standards and Technology (NIST). Cloud Computing Reference Architecture. Available online: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505 (accessed on 15 April 2016).
24. National Institute of Standards and Technology (NIST). Cloud Computing Security Reference Architecture. Available online: http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/CloudSecurity/NIST_Security_Reference_Architecture_2013.05.15_v1.0.pdf (accessed on 15 April 2016).
25. The Open Group Cloud Ecosystem Reference Model. Available online: http://www.opengroup.org/cloud/cloud/cloud_ecosystem_rm/index.htm (accessed on 18 April 2016).
26. Chou, D. Rise of the Cloud Ecosystems. Available online: http://blogs.msdn.com/b/dachou/archive/2011/03/16/rise-of-the-cloud-ecosystems.aspx (accessed on 15 April 2016).
27. Open Services Gateway initiative (OSGi). Available online: http://www.slideshare.net/bosschaert/osgi-cloud- ecosystems (accessed on 16 April 2016).
28. Mellado, D.; Fernández-Medina, E.; Piattini, M. Security requirements engineering framework for software product lines. *Inf. Softw. Technol.* **2010**, *52*, 1094–1117. [CrossRef]
29. Shei, S.; Alcañiz, L.M.; Mouratidis, H.; Delaney, A.; Rosado, D.G.; Fernández-Medina, E. Modelling secure cloud systems based on system requirements. In Proceedings of the ESPRE, Ottawa, NT, Canada, 25 August 2015; pp. 19–24.
30. Axelrod, C.W. Enforcing security, safety and privacy for the Internet of Things. In Proceedings of the 2015 IEEE Long Island Systems, Applications and Technology Conference, New York, NY, USA, 1 May 2015.
31. Stojmenovic, I.; Wen, S. The Fog Computing paradigm: Scenarios and security issues. In Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, (ACSIS), Warsaw, Poland, 7–9 September 2014; pp. 1–8.
32. Sadeghi, A.R.; Wachsmann, C.; Waidner, M. Security and privacy challenges in industrial Internet of Things. In Proceedings of the ACM DAC'15, San Francisco, CA, USA, 8–12 June 2015.
33. Dsouza, C.; Ahn, G.J.; Taguinod, M. Policy-driven security management for fog computing: Preliminary framework and a case study. In Proceedings of the IEEE International Conference on Information Reuse and Integration, San Francisco, CA, USA, 13–15 August 2014; pp. 16–23.
34. Taherkordi, A.; Eliassen, F. Towards Independent in-Cloud Evolution of Cyber-Physical Systems. In Proceedings of the 2nd IEEE International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA 2014), Hong Kong, China, 25–26 August 2014.
35. Fernandez, E.B. Preventing and unifying threats in cyberphysical systems. In Proceedings of the 17th IEEE High Assurance Systems Engineering Symposium (HASE), Orlando, FL, USA, 7–9 January 2016.
36. Hashizume, K.; Fernandez, E.B.; Larrondo-Petrie, M.M. A pattern for Software-as-a-Service in Clouds. In Proceedings of the Workshop on Redefining and Integrating Security Engineering, part of the ASE International Conference on Cyber Security, Washington, DC, USA, 12–14 December 2012.
37. Fernandez, E.B.; Hamid, B. A pattern for Networks Functions Virtualization. In Proceedings of the EuroPLoP2015, Kloster Irsee, Germany, 8–12 July 2015.

38. Fernandez, E.B.; Yoshioka, N.; Washizaki, H. Cloud Access Security Broker (CASB): A pattern for accessing secure cloud services. In Proceedings of the 4th AsianPLoP (Pattern Languages of Programs) 2015, Tokyo, Japan, 5–7 March 2015.
39. Fernandez, E.B.; Washizaki, H.; Yoshioka, N. Patterns for Secure Cloud IaaS. In Proceedings of the Asian Pattern Languages of Programs (PLoP) Conference, Taipei, Taiwan, 24–26 February 2016.
40. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the 1st ACM Mobile Cloud Computing Workshop, Helsinki, Finland, 17 August 2012.
41. Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog Computing: A Platform for Internet of Things and Analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*; Springer International Publishing: Berlin, Heidelberg, Germany, 2014.
42. Yi, S.; Hao, Z.; Qin, Z.; Li, Q. Fog computing: Platform and applications. In Proceedings of the 2015 Third International IEEE Workshop on Hot Topics in Web Systems and Technologies, Washington, DC, USA, 12–13 November 2015; pp. 73–78.
43. Ragget, D. The web of things: Challenges and opportunities. *Computer* **2015**, *48*, 28–32. [CrossRef]
44. Washizaki, H.; Fukumoto, S.; Yamamoto, M.; Yoshizawa, M.; Fukazawa, Y.; Okubo, T.; Ogata, S.; Fernandez, E.B.; Kaiya, H.; Kato, T.; *et al.* A Metamodel for Security and Privacy Knowledge in Cloud Services. submitted for publication. 2016.
45. Brown, A.; Apple, B.; Michael, J.B.; Schumann, M. Atomic-level security for web applications in a cloud environment. *Computer* **2012**, *45*, 80–83. [CrossRef]
46. Fernandez, E.B.; Yoshioka, N.; Washizaki, H.; VanHilst, M. An approach to model-based development of secure and reliable systems. In Proceedings of the Sixth International Conference on Availability, Reliability and Security, Vienna, Austraia, 22–26 August 2011.
47. Cisco White Paper. Security Everywhere. Available online: http://www.cisco.com/web/offers/pdfs/security-everywhere-whitepaper.pdf (accessed on 18 April 2016).
48. Syed, M.H.; Fernandez, E.B. Cloud ecosystems support for Internet of Things and DevOps using patterns. In Proceedings of the First International Workshop on Interoperability, Integration, and Interconnection of Internet of Things Systems (I4T), part of the IEEE International Conference on Cloud Engineering (IC2E), Berlin, Germany, 4–8 April 2016.
49. Wood, C.; Summers, R.; Fernandez, E.B. Authorization in Multilevel Database Models. *Inf. Syst.* **1979**, *4*, 155–161. [CrossRef]
50. Beckers, K.; Coté, I.; Fassbender, S.; Heisel, M.; Hofbauer, S. A pattern-based method for establishing a cloud-specific information security management system. *Requir. Eng.* **2013**, *18*, 343–395. [CrossRef]