D. Ferraris, C. Fernandez-Gago, and J. Lopez, "A Trust-by-Design Framework for the Internet of Things", *2018 9th IFIP International Conference on New Technologies Mobility and Security (NTMS)*, 2018.
http://doi.org/10.1109/NTMS.2018.8328674
NICS Lab. Publications: https://www.nics.uma.es/publications

A Trust-by-Design Framework for the Internet of Things

Davide Ferraris, Carmen Fernandez-Gago, Javier Lopez
Network, Information and Computer Security Lab
University of Malaga, 29071 Malaga, Spain
{ferraris,mcgago,jlm}@lcc.uma.es

Abstract

The Internet of Things (IoT) is an environment of interconnected entities, that are identifiable, usable and controllable via the Internet. Trust is necessary in a system such as IoT as the entities involved should know the effect of interacting with other entities. Moreover, the entities must also be able to trust a system to reliably use it. An IoT system is composed of different entities from different vendors, each of them with a different purpose and a different lifecycle. So considering trust in the whole IoT system lifecycle is useful and necessary to guarantee a good service for the whole system. The heterogeneity and dynamicity of this field make it difficult to ensure trust in IoT. We propose a trust by design framework for including trust in the development of an IoT entity considering all the phases of the life-cycle. It is composed of the K-Model and transversal activities.

Keywords: Security, Trust, Internet of Things (IoT), System Modelling Language (SysML), K-Model

1 Introduction

The Internet of Things comprises two words: Internet and Things. The word *internet* focuses on the communication part and with the internet many possibilities arise (e.g. distance communication) but also many problems (e.g. threats, attacks). The word *things* signifies the entities involved in this communication exchange, they can be inanimate objects or humans.

Trust is necessary in IoT because the entities involved should know the other entities they have to interact with in order to trust or distrust them.

Decision making is a fundamental part of trust management [13]. This process permits the entity to decide how to act according to the data it has collected and computed. These data have to be measured so they can be used and how to measure trust has become a key issue for IoT.

According to the dynamicity of an IoT system, we can say that it is composed of different entities from different vendors, each of them with a different purpose and a different life-cycle. Considering this, an holistic approach is needed.

We consider trust to be the centre of an IoT entity, because we think it is useful and necessary to guarantee a good service. For this reason, we propose a trust-by-design framework considering it from the early stages of the life-cycle to the last phase: from cradle to grave. We propose a trust by design framework that will help developers to include trust in IoT scenarios from the early stages of a system life-cycle.

Starting from the need of an IoT entity it is important to elicit the right requirements according to the possible context. Trust requirements are central and related to other system requirements such as security and privacy requirements. Then we have a modelling phase. Using trust models [9] and System Modelling Language (SysML) [5] we will implement the best ones to match the requirements and need specification. After this phase there is the development phase, where the developers will create the IoT trusted entity following the previous specifications. Furthermore, there are two phases: verification and validation. The verification phase will be used to ensure that the models, the entity and the requirements match. The validation phase will be utilized to test the entity in its real environment and to control if the requirements and the needs have been fulfilled. Finally, there is the utilisation phase where the smart entity deployed will coexist with other smart entities, it will have to trust or distrust them in a dynamic environment where new devices could join the same network.

The structure of the paper is as follows. In Section 2 we describe the related work. In Section 3 we explain our proposed K-Model. This model in addition to the transversal activities explained in Section 4 compound the framework. In Section 5 we apply the framework to a practical scenario. Finally in Section 6 we draw our conclusions and discuss future work.

2 Related Work

Trust is defined in British English by the Cambridge Dictionary as "to believe that someone is good and honest and will not harm you, or that something is safe and reliable"[1]. With the premise that is necessary to guarantee trust, the problem is to find a suitable way to establish trust between entities. Typically in a trust interaction there are two entities involved, precisely one is the trustor and the other one is the trustee [8].

To consider trust during the development of an IoT entity is a key point, but to guarantee trust we have also to guarantee other security aspects. According to Hoffman et al. [6] and Pavlidis [11] trust is strongly dependent on other security properties like privacy and reliability. All these properties must be aggregated to compute a trust value.

[1] http://dictionary.cambridge.org/dictionary/english/trust

With respect to trust, reputation is more objective. We can say that it can be a parameter for trust decision [7].

To know entity's reputation it is necessary to collect information about it. The collector can be a third party or another entity, this depends on the architecture of the IoT system network, specifically if it is centralised or distributed [12]. In the first case there is an entity (i.e. Smart Hub) that have to collect reputation about the other entities and provide these values to those other entities. In the second case every entity collect information about other entities and shared it.

Coming back to trust management, Moyano et al. [9] classified trust models in two main categories: decision models and evaluation models. The first decision model developed was PolicyMaker [1]. These models are based on policies and rules that help the decision maker to establish trust with an entity or not. On the other hand, the first evaluation model was developed by Marsh [8]. These models evaluate trust value of an entity and then the decision maker decide whether to trust the entity or not. This classification was made to find commonalities between these types of models and provide a framework to deal with the diversity of these approaches.

Inserting trust and its characteristics into models and frameworks is a challenge that other authors have tried to address in previous works. An interesting paper regarding the possibility of using trust in a modelling language has been proposed by Uddin and Zulkernine [15]. UMLTrust focuses on the system design and specification phase. They use class, state machine and use cases diagrams enriched by trust. The framework cover many parts of the System Development Life Cycle (SDLC) and they use stereotypes and tags to bring trust to the models. On the other hand, they do not consider parameters correlated to trust like security or privacy. Neither do they consider important phases like needs and problem elicitation, verification and validation. Finally there is no backtracking.

Ruan [13] proposed a general trust management framework. It is interesting because it comprises three main tasks: *trust modelling*, *trust inference* and *decision making*. Each of these tasks is context-dependent. This is a focus point we will also take in consideration. However, this framework was not developed specifically for IoT and covers only the modelling phase. We cover also other phases as we will explain later.

Sharma et al. [14] presented a generic framework to manage trust in IoT from a parameters perspective, considering each qualitative and quantitative parameter. Their paper proposed a trust management solution focusing on requirements that are useful to manage trust. A weakness of this framework is that there is only one feedback from the last phase to the first phase. In addition, the context here is never taken into consideration.

Gago et al. [3] introduced a framework to help designers and developers integrate trust into IoT. They state that privacy and identity requirements must be present during trust and reputation management to enhance trust. In our work, we have focused on both software and hardware perspectives. However in this framework there is no feedback between phases, we have added this feature to

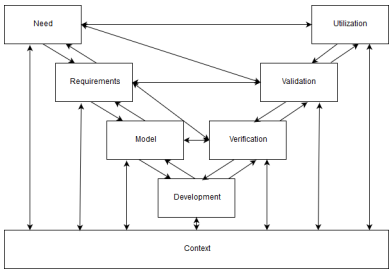

Figure 1: K-Model: with the *Transversal Activities* it compounds our framework

ensure verification and traceability. In this work there is no connection between privacy, trust and identity requirements. We have connected requirements to ensure traceability between them. Finally, we will go through the development phase to the utilisation of the entity, passing on to proper verification and validation that in their model are missing.

3 K-Model

The model we proposed in Fig. 1 is based on the V Model developed by Forsberg and Mooz [4]. It is developed for IoT, but it can be adapted to other fields. We choose the name K-Model because of the composition of the original V model with the context layer. In fact, we believe that it is important in developing an IoT entity to consider the context in every phase. In addition, we have identified some transversal activities. They do not belong to any phase in particular but they assist the whole process to enhance the model, from the original need to the utilisation phase. These activities, discussed in Section 4, with the K-Model they compound our framework. The K-Model has been divided into two main parts: the V model and the context layer. Furthermore, each phase has connections. One is the connection with the previous and the following phase (in the case they exist). The backward connection guarantees traceability to the previous phase. The forward connection guarantees specifications for the next phase. Another type of connection is between the left and right sides of the K-Model. These direct connections are important to guarantee that the specifications are fulfilled. Finally, each phase is connected to the context. The connections will be explained in more depth when discussing each phase.

Users and vendors are equally important for this framework to help the developers to create the desired entity.

To guarantee a holistic perspective to build a trust-by-design IoT entity, we have designed the K-Model as follows.

3.0.1 Need

The first phase is about the need and problem elicitation. In this phase we place all the needs that the users and vendors have about their intended entity

ID	Requirement specification	BT	FT	IT

Table 1: Requirement specification

or software for IoT. It is important from this phase to know if the intended IoT entity will work in a centralised, i.e. with a Smart Hub, or a distributed IoT architecture to build the right product since the first phase. These parameters are fundamentals for the needs document (see Section 4) that is one input for the following phase.

3.0.2 Requirements

The second phase concerns the requirements. Trust requirements will be at the centre but to guarantee trust it is also necessary to guarantee other security properties as stated by Hoffman [6] and Pavlidis [11]. According to them, we have identified seven type of requirements: trust, privacy, identity, security, usability, safety and availability. Trust is correlated to each of them and they cover all the aspects that can increase trust in an entity. Some of them can be in conflict i.e. privacy and identity requirements.

An important characteristic of the requirements that we enhance is the traceability. These requirements follow the specification of IEEE 830-1993 [2] where two types of traceability have been proposed: backward traceability (BT) and forward traceability (FT). BT means that each requirement has to refer to its source. FT means that each requirement leads to a model specification or a feature. In addition to them, we insert another type of traceability: inner traceability (IT). By IT we mean the traceability between different requirements. For example, if we need to change a privacy requirement this can affect trust and security requirements as well. If the connection is not specified we could have a problem in the case of change or relaxing of a requirement. A requirement description is shown in Table 1.

ID is the identification of the requirement, it must be unique and related to its type of requirement. For example, a trust requirement ID is *TRST-XX* where XX is the number.

Requirement specification is the text of the requirement.

BT, *FT* and *IT* are the IDs of the related requirements, documents or model specification. Finally, the requirements must be saved in a proper database.

3.0.3 Model

The model implementation is dependent on the requirement specification. According to the works of Uddin [15] we have identified the possibility to add trust to SysML and UML. SysML has mostly the same diagrams of UML except one that is useful in this framework: the Class Diagram. It can be helpful for software developers. Moreover, in UML the Requirement Diagram is not present. This diagram is important for our framework, thus we will use a composition of

these two modelling languages. In addition, we use trust models to ensure trust in the process.

Using SysML, UML and trust models we can assure that trust requirements specification will match the modelling phase. Finally, depending on the intended IoT architecture (i.e. centralised with a Smart Hub) these models will be used to define the proper architecture and the type of trusted communications between the IoT entities.

3.0.4 Development

In this phase we take into consideration all the features from the previous phases that will be verified in the next phase. This phase can also be managed by a developer who has not followed the previous phases. The documents and the models developed in the previous phases will give the developer all the necessary knowledge to develop the right IoT entity. The developer will use a top down approach to go deeply from the needs to the functionalities. In addition, it will be used a bottom up approach to permit a proper verification and validation phases.

3.0.5 Verification

This phase is used to verify that *the entity has been built in the right way*. This means that all the specifications have been followed in a correct way. In the verification phase the functionalities of the entity are tested as well as its correct implementation. Requirements related to the functionalities and not depending on the environment are checked in this phase as well as the models implemented.

3.0.6 Validation

By validation we mean that *the right entity has been built*. This mean that the desired entity has been developed as the users and vendors wanted. This phase checks that the needs have been met and the entity works properly in a real system's environment. All the remaining requirements are checked assuring that the entity fulfils its intended purpose.

3.0.7 Utilization

Once an entity has been developed and correctly validated it is possible to use it. This phase is connected to the needs and requirements phase to ensure that the entity works properly as it was intended and in its proper IoT environment. It is important to highlight that for the final user the overall procedure is like a black box.

We can state that an IoT entity has three possibilities in this phase: join, stay or leave an IoT network. The dynamicity of IoT requires trust to be considered also during this phase. Therefore, it is necessary to develop an entity capable of responding to trust value changes in real time. Moyano [10] developed the idea of trust@run.time, which is an aspect that we take into consideration. In

fact, trust decision making, trust modelling and trust metrics are needed when an IoT entity joins a new network, because it must collect trust information about the other objects and decide how to proceed. Staying in a network means dealing with the dynamicity of the network. Finally, when an entity leaves a network there are several possibilities: it will be dismissed, it will never come back or it will again join the network. Trust metrics can help model these different situations.

3.0.8 Context

In each phase of the K-Model we have to take context into consideration. For us, the context depends not only on the environment or the IoT architecture (i.e. centralised with a Smart Hub), but also on the different services that a smart entity can provide alone or with another smart entities.

4 Transversal Activities

We have identified seven transversal activities that in addition to our K-Model are part of the framework: documentation, metrics, decision gates, traceability, threat analysis, risk management and decision making. These activities can be present in one or more phases as we now describe.

4.0.1 Documentation

It is important to have connections between decisions and to know why a certain decision has been made. The documents are a guide for the next phases but also a feedback for the previous phases in case of errors. This feedback can permit going back to a previous phase to correct any problem that might have arisen.

4.0.2 Metrics

Metrics are important in all phases to measure the performance and efficiency of the entity.Trust metrics are useful to help the entities decide how to proceed in a particular action or with which entity cooperate. The most important phase where they will be used is the last one, where the objects will decide how to behave on the basis of the defined metrics.

4.0.3 Gates

The gates decide whether to continue to the next phase. In addition, they permit a backup point to be created to which it is possible to come back to in the case of problems.

4.0.4 Traceability

The traceability is guaranteed by the connection between phases to ensure backward and forward movement through the framework. The traceability permits to safely modify requirements or specification avoiding unintended consequences.

4.0.5 Threat Analysis

To develop a new product it is important to consider the possible threats. It is beyond the scope of this paper to go deeply into this topic because specific threats depend on a specific system.

4.0.6 Risk Management

Risk management is connected to threat analysis but not solely, as a risk can also arise as a result of malfunctions or bad implementation. During the early phases of the V, it is important to find the possible risks and find the proper mitigation plan. It is important to guarantee the resiliency of an entity but also its correctness.

4.0.7 Decision Making

The decision making process can be used in each phase of the framework. It is very important for the utilisation phase where trust plays a key role in the decisions that entities will make in their IoT environment. Moreover, it can be used in the requirements phase to decide which requirement can be relaxed or modified. Finally, in the modelling phase it is important to decide which model to use.

5 Scenario: Smart Cake Machine

Let us assume an IoT scenario where some users are going to use a Smart Cake Machine (SCM) that tells them which ingredients are needed for the cakes and the recipes are downloadable from a website or inserted by the users. The SCM can check the Smart Fridge (SF) for an ingredient or can send a request to the trusted nearest supermarket to buy missing ingredients. The SCM can communicate with the Smart Hub to check the trusted users allowed to use it. From these needs and considering the context, the requirements are elicited (i.e. security requirements to allow the SCM to buy only the missing ingredients and trust requirements to choose the most trusted supermarket). Up next, the proper models are used, i.e. class diagram to build the software and trust models to check the allowed users or the trusted supermarkets. Once again the context is fundamental to use the proper models. During the development phase the SCM is built following the previous phases and the context. During the verification phase the SCM is tested about its functionality: i.e. if the recipes are downloadable. Then, the SCM is validated in its intended environment, i.e.

to use the correct ingredients for the chosen recipes. Finally, in the utilisation phase the SCM can join other smart devices and contexts, i.e. if a trusted SF is available, the SCM can check it for needed ingredients, otherwise it can check the trusted supermarket to buy them. This scenario can change over time due to the dynamicity of the IoT, however, it is not shown deeply because of space limitations.

6 Conclusion and Future Work

The IoT has brought about new security challenges. To guarantee trust has become a key issue. In this paper, we have proposed a framework based on Software, Security and System Engineering approaches to ensure trust and the other security properties in the whole IoT entity life cycle. For future work, we will expand the phases and the transversal activities of the framework. We will follow the new entity from cradle to grave in order to develop a new smart trusted entity. We will validate and apply this framework to a more complex IoT scenario and develop an IoT network that should deal with IoT entities.

Acknowledgment

This paper has been accepted as a short paper.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 675320.

This research has been partially supported by the Spanish Ministry of Economy and FEDER through the project PRECISE (TIN2014-54427-JIN).

This work reflects only the authors' view and the Research Executive Agency is not responsible for any use that may be made of the information it contains.Âă

References

[1] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 164–173. IEEE, 1996.

[2] IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board. Ieee recommended practice for software requirements specifications. Institute of Electrical and Electronics Engineers, 1998.

[3] Carmen Fernandez-Gago, Francisco Moyano, and Javier Lopez. Modelling trust dynamics in the internet of things. *Information Sciences*, 396:72 – 82, 2017.

[4] Kevin Forsberg and Harold Mooz. The relationship of system engineering to the project cycle. In *INCOSE International Symposium*, volume 1, pages 57–65. Wiley Online Library, 1991.

[5] Sanford Friedenthal, Alan Moore, and Rick Steiner. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[6] Lance J Hoffman, Kim Lawson-Jenkins, and Jeremy Blum. Trust beyond security: an expanded trust model. *Communications of the ACM*, 49(7):94–101, 2006.

[7] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[8] Stephen Paul Marsh. *Formalising trust as a computational concept*. PhD thesis, Department of Computing Science and Mathematics, University of Stirling, 1994.

[9] Francisco Moyano, Carmen Fernandez-Gago, and Javier Lopez. A conceptual framework for trust models. In *9th International Conference on Trust, Privacy and Security in Digital Business (TrustBus 2012*, volume 7449 of Lectures Notes in Computer Science, pages 93–104. Springer Verlag, Sep 2012.

[10] Francisco Moyano Lara. *Trust engineering framework for software services*. PhD thesis, 2015.

[11] Michalis Pavlidis. Designing for trust. In *CAiSE (Doctoral Consortium)*, pages 3–14, 2011.

[12] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279, 2013.

[13] Yefeng Ruan and Arjan Durresi. A survey of trust management systems for online social communities–trust modeling, trust inference and attacks. *Knowledge-Based Systems*, 106:150–163, 2016.

[14] Avani Sharma, Emmanuel S Pilli, Arka P Mazumdar, and MC Govil. A framework to manage trust in internet of things. In *Emerging Trends in Communication Technologies (ETCT), International Conference on*, pages 1–5. IEEE, 2016.

[15] Mohammad Gias Uddin and Mohammad Zulkernine. Umltrust: towards developing trust-aware software. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 831–836. ACM, 2008.