chist-era SUCCESS

M2.2: Detailed Conceptual Framework First Draft

Grant Agreement:	N/A
Project Acronym:	SUCCESS
Project Name:	SecUre aCCESSibility for the internet of things
Instrument:	CHIST-ERA Call 2015
Thematic Priority:	SPTIoT
Start Date:	1 December 2016
Duration:	36 months
Document Type[1]:	T (Technical Report)
Document Distribution[2]:	CO (Confidential)
Document Code[3]:	SUCCESS-UGA-PR-001
Version:	v1.6
Editor (Partner):	S. Bensalem, A. Legay (UGA)
Contributors:	Stefano Schivo, Ioana-Domnina Cristescu
Workpackage(s):	WP2
Reviewer(s):	F. Kammüller (MU)
Due Date:	Month 18
Submission Date:	May 2018
Number of Pages:	11

Funded by EPSRC Engineering and Physical Sciences Research Council, ANR, NWO Netherlands Organisation for Scientific Research

[1]MD = management document; TR = technical report; D = deliverable; P = published paper; CD = communication/dissemination.
[2]PU = Public; PP = Restricted to other programme participants (including the Commission Services); RE = Restricted to a group specified by the consortium (including the Commission Services); CO = Confidential, only for members of the consortium (including the Commission Services).
[3]This code is constructed as described in the H2020 Project Handbook.

M2.2: Detailed Conceptual Framework First Draft

Ioana-Domnina Cristescu[1]

[1]INRIA

REVISION HISTORY

Date	Version	Author	Modification
2.10.2018	1.0	F. Kammüller	Started the document
26.10.2018	1.1	F. Kammüller	Defined structure
27.10.2018	1.2	F. Kammüller	Added MU part and emphasized human centricity
13.11.2018	1.3	I. Cristescu	Added UGA part
14.11.2018	1.4	S. Schivo	Added UT part on ATs and meta-models
17.11.2018	1.5	I. Cristescu	Modifications on UT part
27.11.2018	1.6	F. Kammüller	Summary in Conclusions
27.10.2018	1.7	F. Kammüller	Approval

APPROVALS

Role	Name	Partner	Date
Project Manager	F. Kammüller	MU	27.11.2018

Contents

1 Executive Summary

This is the draft for the detailed conceptual framework. An earlier first draft of the conceptual framework has been provided in Milestone M2 (M2.1).

This draft relates strongly to the Design Principles of SUCCESS as described in D2.1. It elaborates these Principles giving more details on the common treatment of the human centred aspects of the developed formal methods and how Attack Trees are integrated into the process. This detailed description of the phases of the SUCCESS process paves the way to consider the chain of steps of formal methods employed as a common framework.

2 Conceptual Framework Description

2.1 Extracting Formal Architecture and Attack Scenarios from Requirements

In the early requirements phases we use eFRIEND for requirements elicitation of ethical requirements [7]. We then map the non-functional requirements to functional (technical) requirements:

- Use Isabelle Infrastructure framework as detailed in M2.1 and [6, 5] to
  - Derive a formal specification of high level system infrastructure with actors and Security and Privacy policy from requirements specification;
  - Perform analysis of this formal specification by formal proof of security properties with special regard to GDPR mandatory privacy requirements using the attack tree calculus as formalised in the Isabelle Infrastructure framework [5]. The attack tree calculus allows a direct mapping between a logical description of the attack target in the temporal logic CTL and attack trees. This is an ideal preparation for Modelchecking (applied in the following steps of the SUCCESS process).

The end result of this early process is a high level formal system architecture description and security and privacy policies with properties that can be formally proved in the interactive theorem prover Isabelle. Most importantly, it introduces the human (actor) as part of the system specification. Since the Isabelle Infrastructure framework augments the earlier formalisation of Insider threats [8], the psychological disposition of the actor can be modelled and become part of the formal reasoning process.

For the pilot of the SUCCESS project the architecture derived by this process is depicted in Figure 2.1 [5].

The following Isabelle theorem expresses the GDPR property that data ownership cannot be changed by the system [5]. This property is expressed formally in the temporal logic CTL on the logical model of the abstract system specification of the above SUCCESS architecture. An important development outcome of this early formal analysis is a formal policy definition and a DLM type data labeling mechanism to attach access rights to data.

```
theorem GDPR_three: h ∈ gdpr_actors ⟹ l ∈ gdpr_locations ⟹
  owns (Igraph gdpr_scenario) l (Actor h) d ⟹
  gdpr_Kripke ⊢  AG {x. ∀ l ∈ gdpr_locations. owns (Igraph x) l (Actor h) d }
```

The abstract system architecture needs to be mapped systematically onto concrete IoT systems while still guaranteeing the preservation of corresponding security and privacy properties. This is provided in the next step of security enforcement with attack trees on BIP models and probabilistic rare event model checking. A central scientific step forward is that the human (actor) that has been identified in the Isabelle Infrastructure model can now be represented as a BIP component (as discussed in the following section).

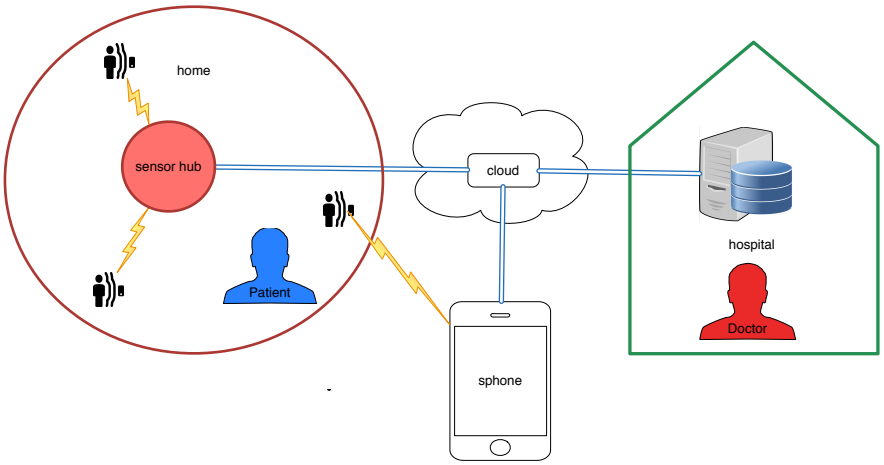

Figure 2.1: IoT healthcare monitoring system for SUCCESS project

2.2 Security enforcement in IoT systems using Attack Trees

2.2.1 Modeling security attacks using Attack Trees

In IoT systems, the security risks are numerous as such systems involve non-heterogeneous devices that are connected to a shared network and that carry critical tasks, and hence, are targets for malicious users. Therefore, developing a systematic mechanism that considers security aspects at an early stage of system design helps detecting and preventing attacks against IoT systems.

Attack trees are intuitive and practical formal methods to identify and analyze attacks on the security of a system. As their name suggests, attacks are modeled as trees, where the leaves represent elementary steps needed for the attack, and the root represents a successful attack. The internal nodes are of two types, indicating whether all the sub-goals (an AND node) or one of the sub-goals (an OR node) must be achieved in order to accomplish the main goal.

We developed a formal language for modeling IoT systems, where one or several participants in the system (IoT devices, humans) are represent as an *entities*. Entities have a *behaviour* (or a *process*) and a *knowledge*, used to allow (or disallow) its interaction with the rest of the system. For instance, an entity can send an email to another entity only if it knows its email address. Or an user needs to know the url of a website in order to access it; we say that the url is part of the user's knowledge.

A malicious entity in the system, called the *Attacker* carries out attacks against the system. The other IoT entities can inadvertently help the Attacker, by leaking their sensitive data. Equipped with the acquired knowledge the Attacker can then communicate with the IoT entities undetected. The attack tree provided with the model acts as a *monitor*: It observes the interactions the Attacker has with the system and detects when an attack is successful.

2.2.2 *Rare events* Statistical Model Checking and Attack Trees

In our approach [2], an IoT system is analyzed using statistical model checking (SMC), which consists of sampling the executions of an IoT system and computing the probability of a successful attack based on the number of executions for which the attack was successful.

However, the evaluation may be difficult if a successful attack is rare, meaning that it occurs with a probability 10^{-3} or smaller. The method requires a large number of simulations for a correct estimate of a rare event which is not always feasible.

We therefore use *importance splitting* [4], a special SMC method developed for rare events. Importance splitting assumes that an execution leading to a rare event can be decomposed into several intermediate steps. Instead of executing a system until the rare event occurs, the execution is stopped after one of the intermediate steps is reached. The execution is restarted then from that step onwards. The method of importance splitting is well suited for attack trees, as the intermediate steps leading to a rare event are the nodes in the tree leading to a successful attack.

During the model checking phase, the attack tree is transformed into a monitor, which observes all the events occurring in the system. The leaves of the tree are some of the interactions between the Attacker and the other components in the model. The branches of the tree are internal transitions to the monitor component. Monitors consists of Boolean formula that can be evaluated using the executions of the system: when an events occurs that correspond to a node in the Attack Tree, the monitor updates the Boolean formula. An attack is successful when the Boolean formula evaluates to true.

To implement this we rely on BIP, a heterogeneous component-based model for which an execution engine is developed and maintained [1]. The IoT model is translated into a BIP model and the attack tree into a BIP monitor. The two form a BIP system. The execution engine of BIP produce executions which are the input of Plasma [3], our model checker. In the case of importance splitting, Plasma interacts with the BIP engine to guide the executions as described above.

2.3 Attack Trees (AT)

Attack Trees (ATs) are a widespread formalism for modeling security scenarios. Their popularity extends beyond the scope of the SUCCESS project: many independently developed tools allow users to perform a number of different analyses on ATs. It is often the case that a specific tool supports a specific variant of ATs, concentrating on a special set of gates in addition to AND and OR. We support the interoperation among many of those different tools, and promote the integration of the most popular variants of ATs through the definition of a generic Attack Tree meta-model [9]. We built ATTop, a *software bridging tool* founded on the AT meta-model that uses model transformations in order to support information exchange between different tools and variants of ATs, allowing users to analyse their ATs in different ways. The flexibility provided by ATTop is important for SUCCESS, as it facilitates the interoperability among the different tools we built and supports the joint effort of enforcing security and privacy.

2.3.1 The Attack Tree meta-model

As a way to unify the most popular AT variants, we introduced the Attack Tree meta-model, which we illustrate in Figure 2.2. The model is divided between a (tree-like) *Structure*, and a set of *Values*. The Structure meta-model represents the basic attack steps and how their possible compositions lead to the attacker's objective (root of the tree). Attribute values are attached to nodes in the tree in order to describe the characteristics of attack steps: for example, we can represent the cost of a step, either in terms of money or time. Thanks to this separation between structure and attributes, we are able to easily plug into the same AT structure different

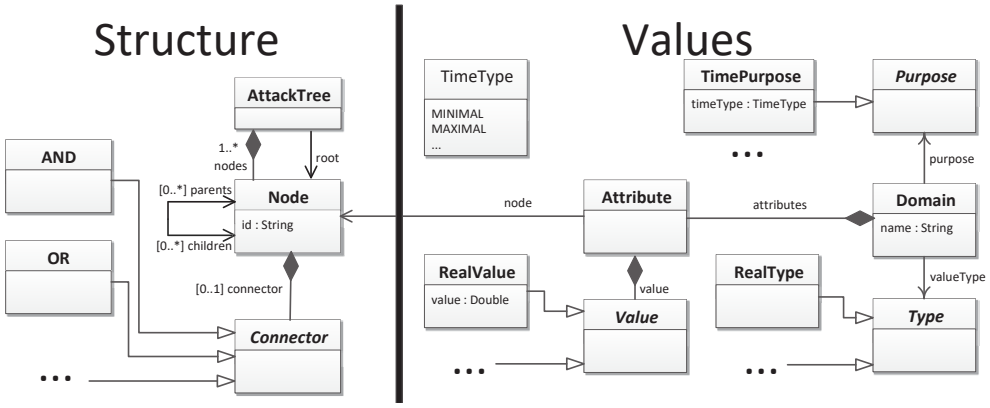

Figure 2.2: The Attack Tree meta-model. An attribute (right) is linked to a node (left) via reference.

attacker profiles: each type of attacker (from script kiddie to international hacker organization) has access to different resources, and has different objectives w.r.t. the same target system. The definition of attacker profiles allows us to reflect all these categories of attackers with different attributes sets referring to the same AT: in this way, the chances of a successful attack (see Sect. 2.2) can be computed based on a set of different profiles in a more efficient way.

2.3.2 ATTop and model transformations

The ATTop tool has the explicit objective of supporting interoperability among different AT tools. This is achieved through *model transformations* based on the encompassing AT meta-model: because they allow us to use the same concepts in different tools, we call this type of transformations *horizontal* (see Fig. 2.3).

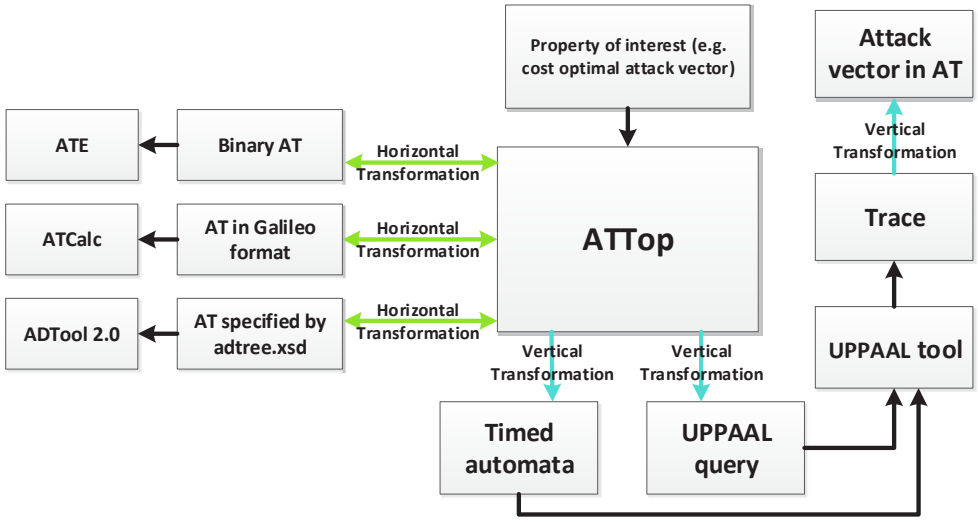

Figure 2.3: The horizontal and vertical transformations used in ATTop to support tool interoperability and provide the integration of different domains.

In addition to the supporting tool interoperability, ATTop allows to analyse ATs using a *vertical* transformation to the Timed Automata formalism. Transparently to the user, an AT is transformed into a network of Timed Automata, which is analysed with the UPPAAL model checker: for example, the most cost-effective attack scenario can be requested. The results are then translated back into the AT domain: in the example in Figure 2.4, the resulting attack scenario is presented as a sequence of basic attack steps in the original AT model. In this way, the power of the Timed Automata formalism can be exploited without requiring the user to learn its intricacies.

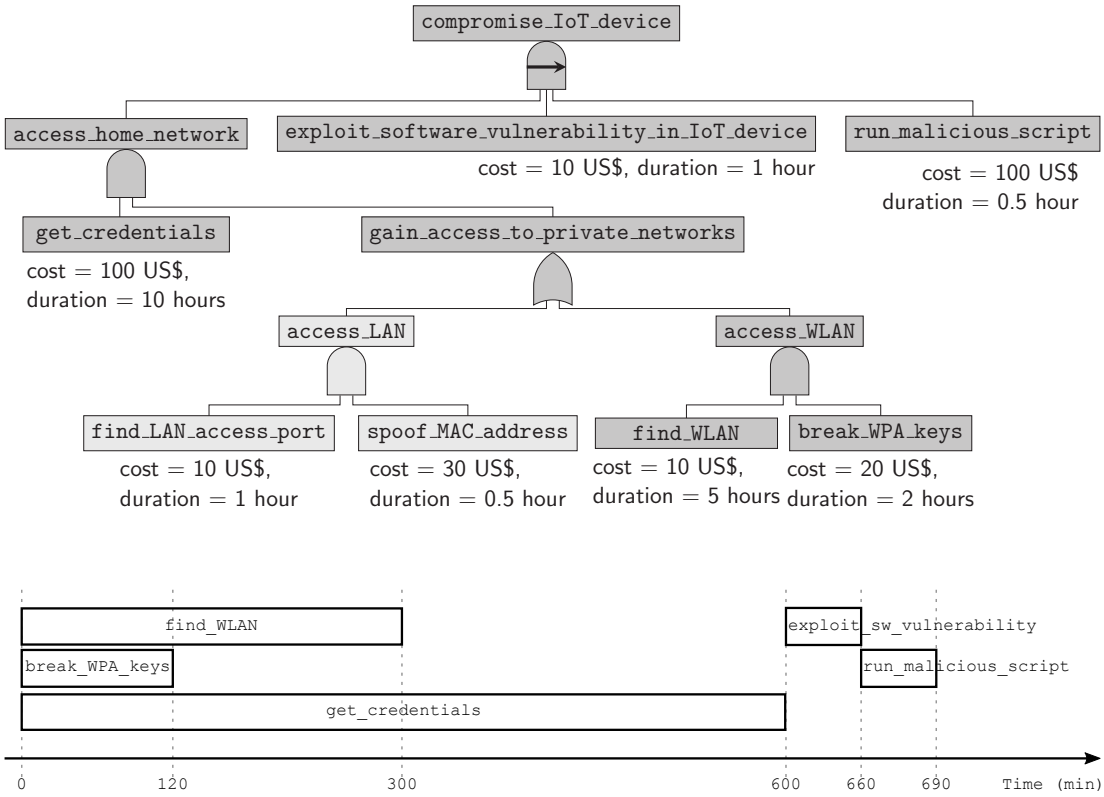

Figure 2.4: **Above**: an example of Attack Tree representing the compromise of a IoT device by an attacker. The topmost gate is a sequential AND (SAND) gate: the three steps it composes need to be performed in a sequence from left to right. Attributes are represented directly below the nodes, while the attack vector representing the most cost-effective attack is highlighted in darker gray. Note that this attack is not the most effective time-wise. **Below**: the highlighted attack vector represented as a schedule of actions to be taken by the attacker.

3 Conclusion

This Detailed Conceptual Framework summarizes the formal methods that we have developed and adapted to the problem domain of IoT healthcare in the Project. It first explains how we derive a formal high level IoT infrastructure architecture including actors and a policy from the ethical, legal (GDPR), and other security requirements of the application. Based on this, the enforcement of the policy is provided by using a dedicated IoT language for modeling the system, applying rare events statistical model checking together with attack trees to produce a BIP model and monitor for executing the system. While in this process the IoT system model gets refined from high level specification all the way down to a fine grained BIP component system model, attack trees expressing the attacks weave through the process. Therefore, the dedicated attack tree tool ATTop provides a common platform to express attack trees uniformly and translate them between different model checking and other analysis tools.

Summarising, the provided formal methods line up to define a Conceptual Framework that allows to develop secure IoT systems from the requirement specification down to executable BIP implementations.

A remaining practical challenge is how to port the outcome of this process onto the pragmatic installation of the IoT Pilot that has been developed in parallel in the Project as described in Deliverable D4.1.

Bibliography

[1] A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T.-H. Nguyen, and J. Sifakis. Rigorous component-based system design using the bip framework. *IEEE Software*, 28(3), 2011.

[2] D. Beaulaton, I. Cristescu, A. Legay, and J. Quilbeuf. A modeling language for security threats of iot systems. In F. Howar and J. Barnat, editors, *Formal Methods for Industrial Critical Systems*, pages 258–268, Cham, 2018. Springer International Publishing.

[3] B. Boyer, K. Corre, A. Legay, and S. Sedwards. Plasma-lab: A flexible, distributable statistical model checking library. In K. Joshi, M. Siegle, M. Stoelinga, and P. R. D'Argenio, editors, *Quantitative Evaluation of Systems*, pages 160–164, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[4] C. Jégourel, A. Legay, S. Sedwards, and L. Traonouez. Distributed verification of rare properties with lightweight importance splitting observers. *CoRR*, abs/1502.01838, 2015.

[5] F. Kammüller. Attack trees in isabelle. In *20th International Conference on Information and Communications Security*, volume 11149 of *LNCS*. Springer, 2018.

[6] F. Kammüller. Formal modeling and analysis of data protection for gdpr compliance of iot healthcare systems. In *IEEE Systems, Man, and Cybernetics, IEEE SMC'18*. IEEE, 2018.

[7] F. Kammüller, J. C. Augusto, and S. Jones. Security and privacy requirements engineering for human centric iot systems using efriend and isabelle. In *IEEE/ACIS 15th International Conference on Software Engineering Research, Management and Application, SERA2017*, CPS. IEEE, 2017.

[8] F. Kammüller and C. W. Probst. Modeling and verification of insider threats using logical analysis. *IEEE Systems Journal, Special issue on Insider Threats to Information Security, Digital Espionage, and Counter Intelligence*, 11:534–545, June 2017 2017.

[9] R. Kumar, S. Schivo, E. Ruijters, B. M. Yildiz, D. Huistra, J. Brandt, A. Rensink, and M. Stoelinga. Effective analysis of attack trees: A model-driven approach. In A. Russo and A. Schürr, editors, *Fundamental Approaches to Software Engineering, 21st International Conference, FASE 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings.*, volume 10802 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2018.