Tool Support for Risk-driven Planning of Trustworthy Smart IoT Systems within DevOps

Andreas Thompson[1] and Gencer Erdogan[2]
[1]University of Oslo, Oslo, Norway
[2]Software and Service Innovation, SINTEF Digital, Oslo, Norway

Keywords: Security, Privacy, Cyber-risk, DevOps, IoT, Method, Risk-driven, Planning, Tool Support, Smart Home.

Abstract: There is a serious lack of support for trustworthy smart IoT systems within DevOps. Security and privacy are often overlooked in DevOps cultures and almost absent in the context of IoT. In this paper, we focus on the planning stage of DevOps and propose a tool-supported method for risk-driven planning considering security and privacy risks. Our method consists of five steps: establish context, analyse dataflow, model privacy and security risk, develop risk assessment algorithm based on risk model, and execute risk assessment algorithm. Our tool supports this method in the first and the last step and facilitates dynamic risk assessment based on input provided by the user or collected from the monitoring stage into predefined risk models. The output of the tool is a risk assessment which the end users, e.g. developers, can use as decision support to prioritize certain parts of the target under analysis in the next cycle of DevOps. The tool and the method are evaluated in a real-world smart home case. Our initial evaluation indicates that the approach is comprehensible for our intended users, supports the planning stage in terms of security and privacy risk assessment, and feasible for use in the DevOps practice.

1 INTRODUCTION

As the number of Internet-of-Things (IoT) devices grows ever more rapidly (Gartner envisions that 21 billion IoT endpoints will be in use by 2020 (Meulen, 2017)), systems and networks become more complex, and the need for proper security becomes important. According to Leukert (2016), risks related to security, trust, privacy and identity management are major challenges in today's IoT systems. Ardagna, Damiani, Schütte, and Stephanow (2018) reports that assessing the quality and security risks of IoT systems however is not a simple task, and due to services and devices having constraints on cost, time to market and functionality, developers of these services and devices often disregard this assessment. Because IoT systems typically operate in highly dynamic environments, they need to be able to continuously evolve and adapt, to ensure and increase their trustworthiness. The DevOps software engineering culture and practice aims at shorter development cycles, increased deployment frequency and more dependable releases, which makes for a more agile and dynamic development process. However, according to Taivalsaari and Mikkonen (2017), there is a serious lack of support for trustworthy smart IoT systems in DevOps.

Since the DevOps practice is a continuous loop of planning, developing, releasing and monitoring, automating and streamlining the process is key. By developing a software tool for risk assessment of the IoT system architecture to be used in the planning stage of DevOps, we can help ensure trustworthy execution of IoT systems, as well as reduce manual labour in the DevOps cycle.

The contribution of this paper is twofold. First, we propose a method with the purpose of assisting developers in the planning phase of DevOps with identifying security and privacy risks. Second, we present a web-based tool to support our method in executing risk assessment algorithms to facilitate dynamic risk assessment. Our tool is open source and freely available online (Thompson, 2019).

To address the abovementioned needs and develop artefacts that appropriately meet these needs, we define the following success criteria and their justification.

Success Criterion 1: The tool and method must be easy to use and comprehensible for developers.

As the main beneficiaries of this tool are developers, the tool must be comprehensible and the features sufficiently intuitive for developers.

Thompson, A. and Erdogan, G.
Tool Support for Risk-driven Planning of Trustworthy Smart IoT Systems within DevOps.
DOI: 10.5220/0009189307420753
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 742-753
ISBN: 978-989-758-399-5; ISSN: 2184-4356

Success Criterion 2: The tool-supported method should support the planning of trustworthy smart IoT systems in the DevOps practice in terms of security and privacy risk assessment.

By this we mean that the tool must provide the developers with risk levels based on the risk assessment. This is necessary to support developers prioritize the areas of the system that need to be treated in order to mitigate the privacy and security risks the system is exposed to.

Success Criterion 3: The tool and method should be appropriate for use in the DevOps practice in terms of adapting to new plans and flexible in response to changes in the system.

For the tool to be efficient in a DevOps environment, it is important that it follows the agile paradigms. Features such as importing and exporting information will help the iterative nature of the practice.

In Section 2, we describe our research method. In Section 3, we describe the steps of our tool-supported method, while in Section 4 we provide an overview of our tool and apply our tool-supported method in a real-world smart home case. In Section 5, we provide related work. In Section 6, we discuss the extent to which we have fulfilled our success criteria described above, before concluding in Section 7.

2 RESEARCH METHOD

Figure 1 illustrates the steps of our research method, which is in line with the design science approach by Wieringa (2014). Although Figure 1 illustrates sequential steps, the method was carried out iteratively where some of the steps were revisited during the process.

Figure 1: Research method.

In Step 1, we identified three success criteria which act as requirements for our tool-supported method based on the background and needs as explained in Section 1.

In Step 2, we developed our tool-supported method for risk-driven planning of trustworthy smart IoT systems in DevOps. The method consists of five main steps, namely: establish context, analyse data flow, model privacy and security risk, develop risk assessment algorithm, and finally execute risk assessment algorithm. We also developed a tool to support the first and the last step of our method.

In Step 3, we evaluated the tool-supported method in a realistic smart home case study to assess the feasibility of our approach w.r.t. our success criteria.

3 TOOL-SUPPORTED METHOD FOR RISK-DRIVEN PLANNING

Before explaining our tool-supported method, it is necessary to place our approach in the DevOps process. The aim of our tool-supported method is to support risk-driven planning in the DevOps process. As illustrated in Figure 2, the DevOps process consists of the following steps: planning, coding, building, testing, release, deployment, operation and monitoring. The process then repeats iteratively.

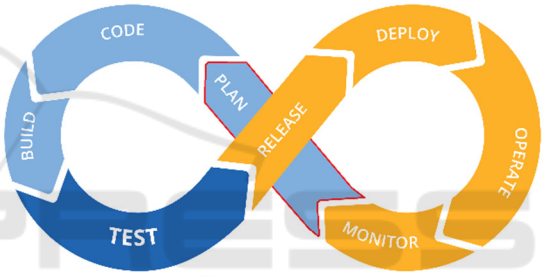

Figure 2: The DevOps process.

Our tool-supported method is to be used continuously in the planning phase of the DevOps cycle. Planning is the first step of every cycle, and the aim is to define criteria and functionalities to be fulfilled by the end of each phase. Planning is usually done without the use of distinct tools in short iterations, and teams are planning with high-level objectives in mind.

With this in mind, our tool is suitable for quick iterations, with a high-level of abstraction. Our tool provides a simple to use, high-level modelling interface. Moreover, one of the main objectives of our method is to define machine-readable risk assessment algorithms and execute them in our tool. Once a risk model and its corresponding risk assessment algorithm has been developed, the risk assessment can be executed repeatedly in our tool based on information provided by the user or collected dynamically from the monitoring stage. This means that once a risk model and its corresponding risk assessment algorithm is defined, continuous risk assessment is possible if no updates to the system model is made. An update to the system models means that the risk model also needs

update. The output of the tool is a risk assessment, which then is used by the developer to plan the DevOps cycle with respect to prioritized risks. Our method supports trustworthy planning in the sense that it focuses on privacy and security risks, which according to Myrbakken and Colomo-Palacios (2017) is often overlooked in DevOps cultures and almost absent in the context of IoT according to a recent systematic literature review by Nguyen et al. (2019).

As illustrated in Figure 3, our tool-supported method consists of five steps. In Step 1, we establish the context following the principles of the corresponding step in the risk management standard ISO/IEC 27005 by ISO (2018). This step establishes the foundation for the risk assessment that follows in the subsequent steps of our method. Step 1 takes as input the system description (documentation, diagrams, expert knowledge from the system owners, etc.). Based on the system description, we identify the services in the (IoT) system architecture and how they communicate. The services are first added conceptually to the list of services in our tool, then based on the list of services, we create a relational model by drag-and-drop functionality from the list of services into the drawing area of our tool, and finally draw edges between the services representing the communication between services.

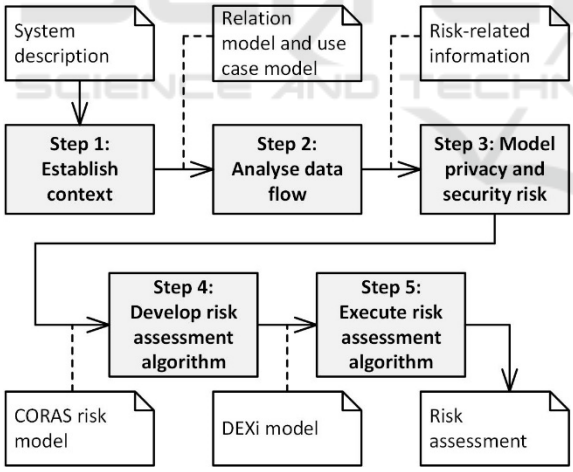

Figure 3: Tool-supported method for risk-driven planning.

Having created the relation model, we then create use case models based on the relation model to obtain an overview and easier understand the relationship between different use cases.

Finally, as part of Step 1, we also identify security and privacy assets we are interested in protecting, as well as scales for likelihood and consequence. The security and privacy assets are important to identify as part of the context establishment because the assets are what motivates the conduction of the risk assessment in the first place. If there are no assets to protect, then there is also no need to conduct risk assessment. This is true for defensive risk assessment, which is the case of our approach. The likelihood and consequence scales are important later in the process to estimate the identified risks.

In Step 2, we use the relation models and use cases created in Step 1 to create data flow diagrams capturing information flow. Moreover, with the knowledge of what information flows through the different actors and services, we identify information regarding potential privacy and security risks using the STRIDE method by Microsoft. STRIDE is a lightweight and effective methodology to identify vulnerabilities based on Data Flow Diagrams including trust boundaries, which can in turn be used as basis for security and privacy risk modelling and assessment.

In Step 3, we use the CORAS approach by Lund, Solhaug, and Stølen (2011) to create graphical risk models based on the data-flow diagrams created in the previous step in order to capture privacy and security risks the target of analysis is exposed to.

In Step 4, we schematically translate CORAS risk models developed in Step 3 into machine-readable risk assessment algorithms in terms of DEXi models. DEXi is a computer program for development of multi-criteria decision models and the evaluation of options as described by Bohanec (2017).

In Step 5, we execute the risk assessment algorithms in our tool. The algorithm produces a risk assessment in terms of risk levels based on input provided by the user or collected from the monitoring.

4 SMART HOME CASE

In this section, we apply our tool-supported method in a real-world smart home case. However, before applying our approach, it is necessary to provide an overview of the different parts of our tool.

4.1 An Overview of Our Tool

Figure 4 illustrates the main parts of our tool from a graphical user interface perspective. The top section of the tool has an in-browser drawing area where the end user can draw a relational model of the target system. For example, in Figure 4 we have illustrated a Smart Phone that communicates with the services Amazon Echo, Nest app and Millheat app. On the right hand-side of the drawing area, there is a list of available services the end user can drag and drop into the drawing area to model the IoT system architecture

considered. The end user can also add or remove services. The complete list of services is stored in the back-end database for later use. This part of the tool is mainly used in Step 1 (establish context) of our method illustrated in Figure 3.

Having developed the relational model in the tool, the user carries out Steps 2, 3 and 4 of our method illustrated in Figure 3, and then returns to our tool in Step 5. The remaining parts of the tool is dedicated to the risk models and their corresponding risk assessment algorithms capturing security and/or privacy risks the system is exposed to.

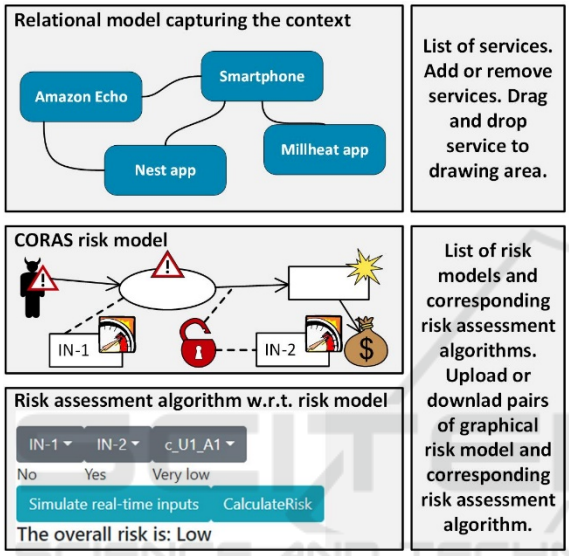

Figure 4: Main parts of our tool from a GUI perspective.

The end user uses the menu on the right-hand side to upload a pair of CORAS risk model and corresponding risk assessment algorithm in terms of a DEXi model. The user may upload more than one pair, and the complete list of pairs of {CORAS risk model, DEXi file} is stored in the back-end database for later use. Having uploaded CORAS risk models and corresponding risk assessment algorithms, the end-user can select a pair to assess risks and the tool will automatically displays the CORAS risk model in the middle section of the tool as well as input options for the risk assessment algorithm in the bottom section of the tool as illustrated in Figure 4. The user is now able to execute the risk assessment algorithm as part of Step 5 of our method by selecting values for what we refer to as risk indicators (IN-1, IN-2, etc.). The tool then automatically produces a risk level where the possible risk levels are {Very Low, Low, Medium, High, Very High}. The meaning of these risk levels is described as part of the context establishment in Step 1 of our method. The end user, e.g. the developer, can now use this information to plan the next cycle of the DevOps process considering these risk levels and prioritize development activities.

4.2 Establish Context

Our smart home case considers a family of three living in the outskirts of a large city. The family consists of a mother, a father and an eight-year-old daughter.

Table 1: Devices and services required by the family.

Device	Description
Tail it kids	A smart watch for kids with functionality of tracking, phoning and an emergency SOS button.
Mill AV1200WIFI	Smart home heater which can be controlled from your smartphone.
Amazon Echo	A smart home device which plays music, makes calls, sets alarms, answers questions, controls other smart devices and can buy things online for you.
Nest Cam IQ Outdoor	High quality outdoor camera that alerts your phone when spotting a person.
Nest Cam IQ Indoor	High quality indoor camera with speakers.
Nest Hello Doorbell	Smart home doorbell which includes a camera, a continuous video log, and the possibility of alerting when people arrive at the door.
Nest Protect 2. Generation	Smart home smoke detector with alerts of which room there is smoke, and the possibility of sending warnings to your phone.
Philips Hue Bridge	A smart home bridge which allows for scheduling of timers and control of your lights when away or from your phone.
Philips Hue E26 Bulbs	A smart light with the possibility of changing colours.
Philips Hue Lightstrip	A smart light strip with the possibility of changing colours.
Philips Hue Motion Sensor	A motion sensor which triggers lights when motion is detected.

The father has recently picked up an interest for smart homes and wants a large collection of smart devices to help automate the family's daily lives, help save money on power, and keep their home and family members safe.

The family has different requirements with respect to security and privacy. The mother is sceptical of smart home technology and wants to keep the privacy high for herself and their daughter. Privacy is not a concern for the father, but he is more concerned about their valuables and requires a way to watch their home when physically not at home. Both the mother and father own an android smart phone, which they wish to use together with services to control their devices, and to view their child's location and phone usage if there should be a need for it. To achieve their goals, the family needs a selection of devices, which come with several services. Table 1 lists the devices and services selected in this case.

Based on the above description and selected devices and services, we create a relational model using our tool to capture the high-level overview of how these services are related and communicate. Figure 6 illustrates the resulting relational model in which we include all services and devices in Table 1. Note that Smartphone, Nest app, Philips Hue app, Millheat app and Amazon Services are added to the model in Figure 6 for completeness (these are implicitly part of the services in Table 1). Each node in the relational model in Figure 6 represents a service (or device). Each node is graphically divided in two where the upper part of the node shows the name of the service, and the lower part of the node shows incoming (In) and outgoing (Out) communication from and to other services, respectively.

Next, we describe the use cases addressing the objectives described above. The use cases for this smart home case are the following:

1. The dad uses his phone to check on his daughter's location with the use of the Tail It Kids smart watch and the Tail It app.
2. The mother uses Amazon Echo to buy a new hair product online.
3. The family drives to cabin and are expecting a package delivery while away. The Nest automated doorbell detects a person at the door and alerts the father.
4. While on the way home from the cabin trip, the father wants to come home to a heated house. Using the Millheat app, he turns the heater to 22 degrees Celsius.
5. When they arrive home, the Philips Hue motion sensor senses motion and turns on their Philips Hue lighting.

Due to space limitations we focus on Use Case 2 (buy product from Amazon) in the following sections. Our complete smart home case study is available online in a technical report by Thompson (2019). Figure 5 illustrates the use case model of Use Case 2 described above, in which we include the services captured in Figure 6. Recall that the mother is worried about her privacy, we therefore identify "confidentiality of mother's information" as an asset we need to protect throughout the analysis.

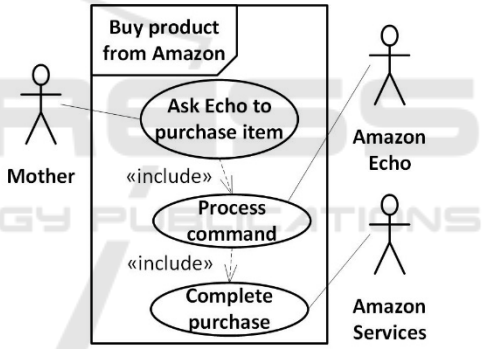

Figure 5: Use case model of Use Case 2.

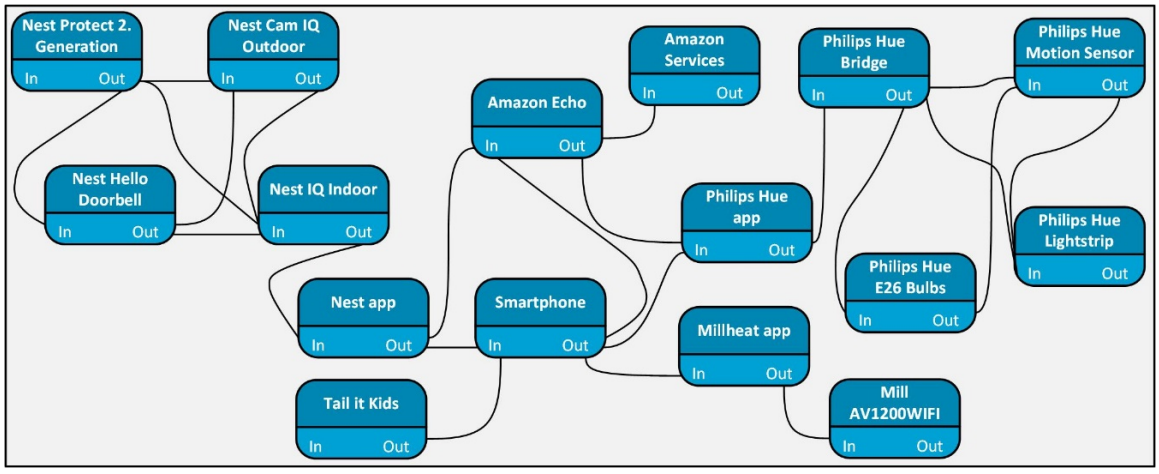

Figure 6: Devices and services to be used by the family.

Finally, as part of the context establishment we identify a likelihood scale (Table 2) and a consequence scale (Table 3) for the abovementioned privacy asset. These scales will later be used to estimate the likelihood and consequence of risks, respectively.

Table 2: Likelihood scale.

Likelihood	Description
Very High	Happens more than two times a year.
High	Happens between once a year to twice a year.
Medium	Happens once every 2 years.
Low	Happens once every 5 years.
Very Low	Happens once every 10 years.

Table 3: Consequence scale for asset "confidentiality of mother's information".

Consequence	Description
Very High	Sensitive information regarding health and/or beliefs is leaked or used inappropriately.
High	Sensitive location data, or data gathered from devices in an automatic fashion is leaked or used inappropriately.
Medium	Information that can be used to identify a specific person is leaked and/or is shared with a third party.
Low	Information that can be used to identify a specific person is used for advertisement.
Very Low	Information that can be used to identify a specific person is collected by a company.

4.3 Analyse Data Flow

After performing the context establishment, we continue with a data flow analysis to gather information related to security and privacy. For each of the use cases presented in the previous section, we create a Data Flow Diagram (DFD). Moreover, we base ourselves also on the different companies' privacy policies in mind for their respective services as we do not have access to proper data flow within their systems. Some of these policies are vague and group different types of information into a general term "your data" or just "data" which is why the models are kept at a high-level of abstraction.

In addition, we also create tables for each of the DFDs in which we extract information from empirical sources, such as privacy policy from the company producing the service and based on that identify information that may be of relevance for security and privacy risks.

Figure 7 shows the DFD diagram for Use Case 2. The mother speaks to Amazon Echo, and a recording of the voice is sent to the Amazon Cloud to be processed. Amazon then stores the recording, transaction details and billing information which have been given by the mother earlier. Transaction details, costumer details and marketing info may then be shared with third parties. Based on the DFD diagram, and the identified trust boundaries, we look for security and privacy related information that may be useful for the next step, which is risk modelling.Table 4 shows a table in which we have gathered security and priv acy related information for the Amazon Echo service. This information was gathered from sources: Privacy Policy by Amazon (2019), Report by Wired (2018), and Paper by Kumar et al. (2018).

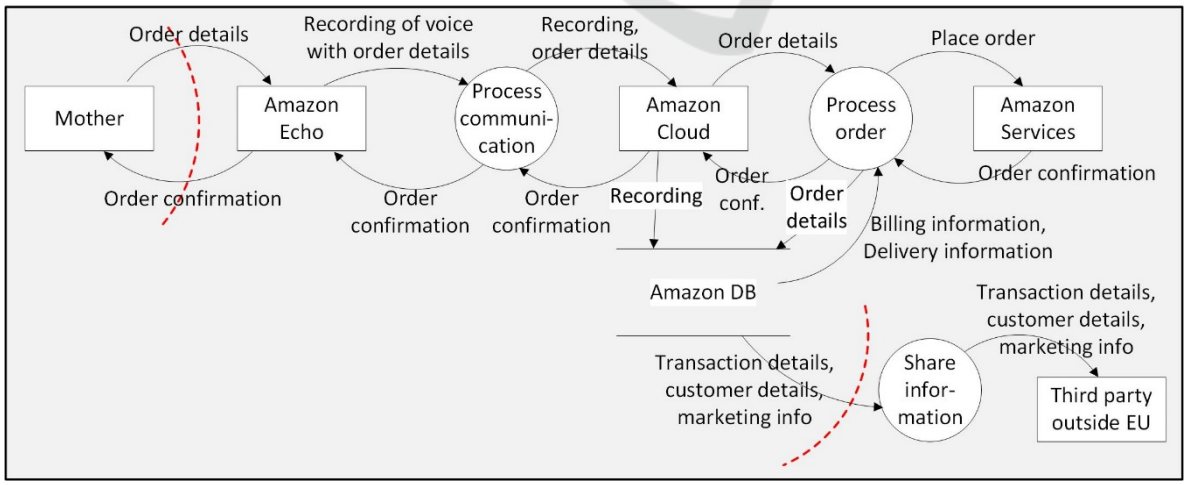

Figure 7: DFD diagram for Use Case 2: The mother uses Amazon Echo to buy a new hair product online.

Table 4: Security and privacy related information gathered for Amazon Echo from different sources.

Information regarding Privacy	Information regarding Security
Amazon receive and store certain types of information whenever a person interacts with them. Amazon shares user data with their parent corporation, Amazon.com, Inc., and the subsidiaries it controls, and they may share personally identifiable information with third parties. Alexa, a part of the Amazon Echo, also has third party “skills”, each with their own privacy policy.	Alexa has so called skills, which can be used to perform an attack called "Skill squatting". Another attack was disclosed by Chinese hackers at DefCon in 2018, where they could remotely control an Alexa. This attack has been patched.

4.4 Model Privacy and Security Risk

For each of the DFD diagrams and security and privacy related information identified in the previous step, we create a CORAS risk model capturing potential threats, threat scenarios, vulnerabilities, and unwanted incidents that may harm the security/privacy assets we need to protect. Figure 8 illustrates the CORAS risk model created for Use Case 2 taking into account the information captured by the DFD diagram in Figure 7 and related information in Table 4.

The CORAS risk model in Figure 8 illustrates two potential threats. An unintentional threat (Amazon) which may initiate threat scenario "S1: share information of user for marketing purposes", which in turn may lead to the unwanted incident "U1: user information of mother is compromised". The second threat is an intentional threat (Hacker) which may initiate threat scenario "S2: Gain access to device recordings", which in turn also leads to unwanted incident U1. The hacker may exploit the vulnerabilities "V1: outdated or insecure device" and "V2: poor network configuration" in order to cause unwanted incident U1. Finally, the unwanted incident U1 may harm the asset we need to protect "A1: confidentiality of mother's information". The reader is referred to Lund et al. (2011) for a detailed explanation of the CORAS semantics.

In addition, the risk model represents indicators (IN-1, IN-2, IN-3, and IN-4). Indicators are used to assess the likelihood and consequence of risks captured by the risk model. An indicator may be attached to a threat scenario or unwanted incident to help assess the likelihood of that threat scenario or unwanted incident. Indicators can also be attached to vulnerabilities in order to assess the conditional likelihood going from one threat scenario or unwanted incident to another threat scenario or unwanted incident.

In a typical CORAS risk analysis, we provide the estimates for likelihood, conditional likelihood, and consequence directly in the risk model. However, in our approach we parameterize these values which we will use in the next step as variables in the corresponding risk assessment algorithm. The likelihood of threat scenarios S1 and S2 are represented by l_S1 and l_S2, respectively. The likelihood of unwanted incident U1 is represented by l_U1. The conditional likelihood going from S1 to U1 is represented by cl_S1_to_U1, while the conditional likelihood going from S2 to U1 is represented by cl_S2_to_U1. Finally, the consequence of U1 on A1 is represented by c_U1_A1. All these variables are used in the next step to construct the risk assessment algorithm.

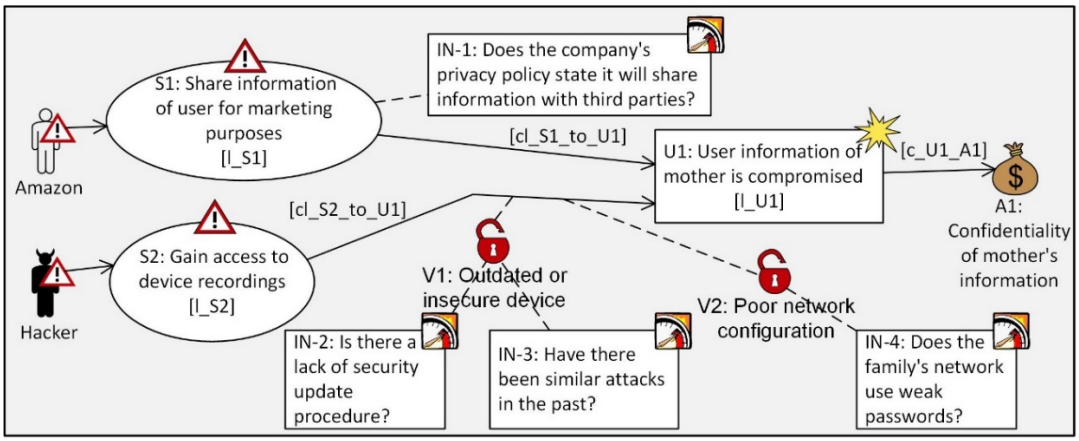

Figure 8: CORAS risk model with indicators for Use Case 2.

4.5 Develop Risk Assessment Algorithm

Once the CORAS risk models are completed, we schematically translate the risk models into DEXi models following the guidelines provided by Erdogan and Refsdal (2017). Figure 9 represents the DEXi model corresponding to the risk model in Figure 8.

According to (Erdogan & Refsdal, 2017), DEXi models have attributes that are either a *basic attribute* (illustrated as green triangles in Figure 9) or an *aggregate attribute* (illustrated as green rectangles in Figure 9). Basic attributes have no child attributes. This means that a basic attribute represents an input to the DEXi model, as its value is assigned directly, rather than being computed from child attributes.

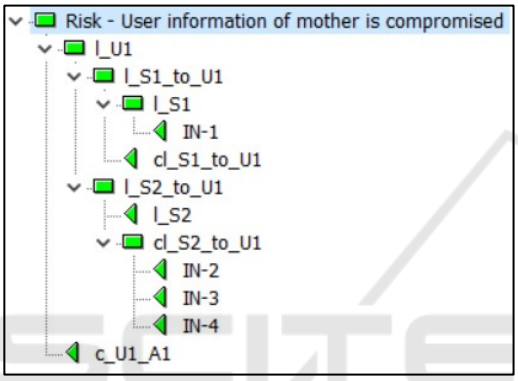

Figure 9: DEXi model corresponding to the risk model in Figure 8.

Aggregate attributes are characterized by having child attributes. The value of an aggregate attribute is a function of the values of its child attributes. This function is called the utility function of the attribute. The utility function of each aggregate attribute is defined by stating, for each possible combination of its child attribute values, what is the corresponding value of the aggregate attribute (illustrated in Figure 10). The DEXi tool automatically computes the value of all aggregate attributes as soon as values have been assigned to the basic attributes. Hence, a DEXi model can be viewed as an algorithm where the basic attribute values constitute the input, and the values of the aggregate attributes constitute the output. Our tool provides an API that makes use of the DEXi java library provided by Bohanec (2017) to execute DEXi files. For example, having developed the DEXi model in Figure 9, we upload this model to our web-based tool which automatically creates a graphical user interface to select input to the model as explained in Section 4.1. Thus, we make the risk models "executable" to the end user by letting the end user select the input variables. Notice that for the risk assessment values that do not have indicators attached we define them as basic attributes. For example, the likelihood of S2 (l_S2) has no indicators attached. Thus, we define l_S2 as a basic attribute and let the end user select the likelihood value (see Figure 8 and Figure 9).

Figure 10 is a screenshot from the DEXi tool in which we have defined the utility function for the aggregate attribute named cl_S2_to_U1, which is the conditional likelihood going from S2 to U1 (see Figure 8 As can be seen from Figure 10, the conditional likelihood is assigned different values depending on the values of the indicators. For example, if the answers to the questions in indicators IN-2, IN-3, and IN-4 (depicted in Figure 8) are "Yes", then it is very likely that the hacker will successfully obtain the user information of the mother. In this case, the value for cl_S2_to_U1 is therefore set to Very High.

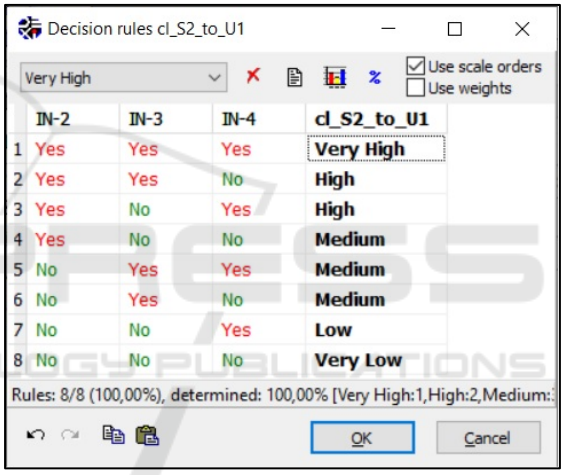

Figure 10: Utility function for conditional likelihood cl_S2_to_U1 based on indicator values IN-2, IN-3, IN-4.

4.6 Execute Risk Assessment Algorithm

Finally, having created risk assessment algorithms in terms of DEXi models, we return to our tool and upload the CORAS risk model together with its corresponding DEXi model. Figure 11 is a screenshot of our tool in which we see the feature to upload and download a pair of CORAS risk model and corresponding DEXi model.

Having uploaded a pair of {CORAS model, DEXi model}, the tool generates a GUI just below the risk model (as explained in Section 4.1) enabling the end user to select input to the risk assessment algorithm. Due to space limitations, we do not include a complete screenshot showing the uploaded risk model together with the system model created in the context establishment step. However, all models created in the

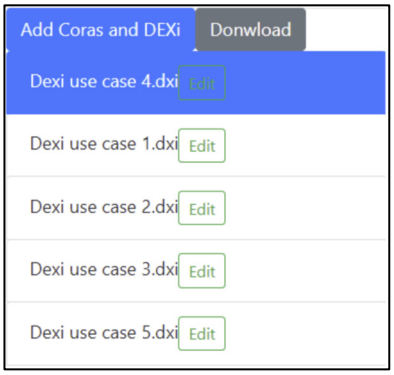

Figure 11: Screenshot of our tool showing upload/download feature of CORAS and DEXi models.

smart home case (including all use cases described earlier) are available online as provided by Thompson (2019). Figure 12 illustrates only the portion of the GUI generated after uploading a pair of {CORAS model, DEXi model}. In this example, the risk model has two indicators for likelihood estimation (IN-1 and IN-2) and one indicator for consequence estimation (c_U1_A1). We see from Figure 12 that the user has selected "No" for IN-1, "Yes" for indicator IN-2, and "Very High" as consequence value c_U1_A1. Beneath the indicator values, we see a button to calculate risk based on the provided values, which when pressed calculates the risk based on the provided values and (in this example) returns the text "The overall risk is: High". This risk value is dynamically calculated when changing the selectable values (IN-1, IN-2, and c_U1_A1). The button "Simulate real-time inputs" is added as a proof-of-concept showing that the tool facilitates the possibility to assess risks automatically based on input obtained from external sources (such as monitoring data). In other words, it demonstrates the capability of the tool's API for dynamically executing algorithms and that this can be easily integrated in other software systems.

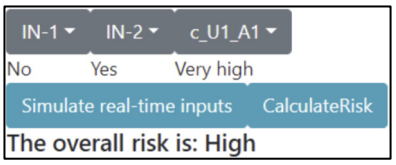

Figure 12: Screenshot of our tool showing the execution of a risk assessment algorithm based on selected values.

As mentioned in Section 4.1, having uploaded the CORAS/DEXi risk models, we obtain a complete overview of the relations between the services in the first part of the tool, alongside a depiction of the appropriate risk models and their input variables in the second part of the tool.

Using the tool, we may select which risk model we wish to use for calculations, and then either manually provide input for the risk algorithms, or use the tool to automate the process by monitoring variables and feed the risk assessment algorithm via the tool's API. In relation to our smart home case, the latter results in an automated risk analysis of the system based on the devices/services being monitored in the family's home.

5 RELATED WORK

For state-of-the-art tool support for risk-driven planning of IoT devices and services, the choices are limited. There have been a lot of research on decision support systems (DSS) in general such as Zeleny (1998) and Bi, Da Xu, and Wang (2014).

Gupta et al. (2015) present an analysis of the state-of-the-art in decision support systems, and critical shortcomings in the existing tools. While not all their work is relevant to this paper, the research is valuable to us. Currently there does not exist any clear mechanisms to collect data about different characteristics of available IoT devices and services, while ensuring trustworthiness and up-to-datedness. There is also a lack of tools to help users link the characteristics of IoT devices and services with the actual risks for their applications or infrastructures.

When it comes to tools, there are many available for use in the DevOps practice. For the different stages there are different tools. Popular ones are Jenkins (2019) or Maven (2019) for testing and NewRelic (2019) for monitoring. These tools are valued as automation; monitoring and continuity are key in the DevOps practice.

The planning phase does not have a lot of dedicated tools, there exists a few such as Jira (2019), but teams often utilize backlogs and Kanban boards to better gain an overview of what needs to be done when as reported by Hüttermann (2012) and Kim, Humble, Debois, and Willis (2016). Overall, there is a lack of tool-support for risk-driven planning in DevOps and almost absent in the context of IoT as reported in a comprehensive systematic literature review by Nguyen et al. (2019).

There also exists general privacy and security risk management tools such as RiskWatch (2019) and ISRAM by Karabacak and Sogukpinar (2005). However, while these tools are quantitative in terms of risk assessment, our tool produces qualitative risk assessment. Moreover, our approach to construct risk assessment algorithms is based on the approach reported by Erdogan and Refsdal (2017). However, while they aim at creating risk assessment algorithms only to be executed inside a software not reachable by

the end user, we bring the risk assessment algorithm to the end user by providing the option of manually selecting risk-indicator values via our web-based tool and apply it in the context of DevOps. In addition, we provide the option of executing the risk assessment algorithm via the API of our tool (Thompson (2019)).

In addition, our approach is different from the approaches above in the sense that we provide a tool to support the agile nature of DevOps and to automate risk estimation (once risk models have been created) as part of risk-driven planning for trustworthy smart IoT systems.

6 DISCUSSION

In this section, we discuss the extent to which we have fulfilled our success criteria described in Section 1.

6.1 Fulfilment of Success Criterion 1

The first criterion states: *The tool and method must be easy to use and comprehensible for developers*.

The first module of our tool is the modelling tool. This module is used for context establishment, where the user creates system diagrams to depict the services and devices to be used (or that are already in use) in a given IoT system. The modelling is fairly straight forward and resembles many other popular modelling languages such as UML class diagrams. Developers often have experience with UML, and it is therefore reasonable to argue that the modelling section of our tool is easy to use and comprehensible by developers.

According to Xie, Lipford, and Chu (2011), developers often need to consider security when developing software. For capturing risk models, we chose CORAS because as reported by Solhaug and Stølen (2013), it has been empirically shown that the CORAS language is intuitively simple for stakeholders with different backgrounds. We may therefore argue that the risk-model section of the tool is reasonably easy to understand by developers. CORAS is also based on the international standards like ISO 27005 and ISO 31000. Moreover, Xie et al. (2011) report that as part of development activities, developers often use concepts such as threat, unwanted incident, vulnerability and risk.

The design of the tool is made such that roughly half the screen is to be used for modelling of the IoT system, while the bottom half is for depicting the CORAS models and executing their respective DEXi algorithms (as illustrated in Figure 4). This design follows the design of the method, in that the user starts by using the tool to design the IoT system, then later uses the tool again to visualize the CORAS diagrams, and then execute the risk assessment algorithms defined in DEXi corresponding to the CORAS models. This helps making the tool pedagogical and easy to use. According to Bohanec, Žnidaršič, Rajkovič, Bratko, and Zupan (2013), DEXi has also been designed to produce models that are comprehensible to end users.

6.2 Fulfilment of Success Criterion 2

The second criterion states: *The tool-supported method should support the planning of trustworthy smart IoT systems in the DevOps practice in terms of security and privacy risk assessment*.

Our tool-supported method is constructed to fit within the DevOps planning phase, and with security and privacy risk assessment in mind. The planning phase of DevOps is the first activity of the DevOps cycle. This first step either has initial input from developers/domain experts (when first starting the DevOps process) or receives additional input from the final monitoring step of the cycle. The input for this step is therefore either initial input from developers/domain experts as part of initiating the DevOps process, or additional input from the monitoring. The planning phase must however output something of value to the next step; coding. Figure 2 illustrates the DevOps process. As our tool and method may receive risk assessment values from the monitoring, and supports changes based on this input, we may argue that our tool and method satisfy the first criterion of the planning phase. Moreover, our tool and method produce risk values as an output, which may be used in the next step (coding) as to help decide what areas need to be worked on and treated regarding privacy and security risks.

Automation, measurements, and tools are key points in the DevOps practice. There exist many tools for DevOps users to use for the different steps of the cycle. Typical tools are Git (2019) for code management, Gradle (2019) or Maven (2019) for build automation, Jenkins (2019) for automation of building, testing and deployment/release, and Nagios (2019) for monitoring. While these tools are powerful and greatly help the steps mentioned, there is a serious lack of tool support for the planning phase as reported by Nguyen et al. (2019). We mentioned Jira (2019) as a popular planning tool, but other than this there are few popular planning tools. By building a tool to suit planning in DevOps for privacy and security risk assessment, we aim to fill this gap.

Since our tool has an API for input and execution of our DEXi risk assessment algorithms, we facilitate automatic and dynamic risk assessment. This can be

seen in the tool by clicking on the "Simulate real-time" button (as illustrated in Figure 12), where the front-end makes frequent API calls to the back-end to simulate input from monitors. Thus, by using the DEXi API provided by our tool and feeding the risk model algorithms with risk-indicator values from a running system, it is possible to support automatic real-time risk assessment.

6.3 Fulfilment of Success Criterion 3

The third criterion states: *The tool and method should be appropriate for use in the DevOps practice in terms of adapting to new plans and flexible in response to changes in the system.*

We can argue that due to the flexible nature of the method, it is appropriate to use within the agile DevOps planning phase. If there are changes made to the system regarding architecture after risk diagrams and risk assessment have been created, it is possible to create new risk models and algorithms or update existing ones ready to be executed.

Our tool and method support modular modelling. The user may focus on small parts of the system, creating smaller system diagrams (example provided in Figure 6), specific risk models for the given context, and risk assessment algorithms. This allows for the possibility of assessing risk with regards to specific parts of a system. It also allows for flexibility when making changes to the system, as not all parts depend on each other. On the other hand, one may also create large system diagrams, encapsulating larger parts of the system. This may be helpful if we want to create a risk picture of a larger part of the system. One may then also create large CORAS diagrams, which results in fewer but larger risk assessment diagrams. Because of the possibility of creating either small diagrams and algorithms, large diagrams and algorithms, or a combination of large and small diagrams and algorithms, we may argue that the tool and method is flexible in response to changes to the system.

7 CONCLUSION

In general, there is a serious lack of support for trustworthy smart IoT systems within DevOps. In addition, there are only a few tools aimed at the planning phase of DevOps. To the best of our knowledge, there is no tool support specifically for risk-driven planning with focus on security and privacy risks. To fill this gap, we propose a tool-supported and risk-driven method for planning in DevOps considering security and privacy risk.

We have applied our tool-supported method in a real-world smart home case. Our initial results indicate that the approach is easy to use and comprehensible for developers, supports the planning of trustworthy smart IoT systems in the DevOps practice in terms of security and privacy risk assessment, and is appropriate for use in the DevOps practice in terms of adapting to new plans and flexible in response to changes in the system.

The tool supported method has, at the time of writing, been carried out in its entirety by the authors. This was a deliberate first-hand evaluation as part of developing our approach to investigate its applicability and feasibility. However, this is also a threat to validity as it has not yet been tried out by potential users other than the authors. These aspects need therefore to be addressed in future work as case studies and experiments.

On the positive side, however, we know that CORAS has been tried out in practice in sharp industrial cases as reported by Lund et al. (2011). The DEXi tool has also been tried out in industrial cases as reported by Bohanec et al. (2013) and Bohanec (2017). The STRIDE method by Microsoft (2019) including DFD diagrams is one of the most popular approaches in identifying cyber risks with respect to data flow. Finally, the schematic translation of CORAS risk models to risk assessment algorithms in terms of DEXi models have been tried out in full-scale industrial pilots in the EU projects WISER and CYBERWISER.eu (2019). In other words, the different parts of our tool-supported method have been thoroughly tried out individually in real-life industrial settings, which in turn supports the feasibility and applicability of our method.

Nevertheless, considering the serious lack of tools and approaches for planning within DevOps with focus on security and privacy risks, we believe our approach is one step towards the right direction in addressing this need and interesting for the Security/DevOps community.

ACKNOWLEDGEMENTS

This work has been conducted as part of the CYBERWISER.eu project (786668) as well as the ENACT project (780351) funded by the European Commission within the Horizon 2020 research and innovation programme.

REFERENCES

Amazon. (2019). Amazon Privacy Notice. Retrieved from https://www.amazon.co.uk/gp/help/customer/display.html/ref=gss?nodeId=502584

Ardagna, C. A., Damiani, E., Schütte, J., & Stephanow, P. (2018). A case for IoT security assurance. In *Internet of Everything* (pp. 175-192): Springer.

Bi, Z., Da Xu, L., & Wang, C. (2014). Internet of things for enterprise systems of modern manufacturing. *IEEE Transactions on industrial informatics, 10*(2), 1537-1546.

Bohanec, M. (2017). DEXi: A Program for Multi-Attribute Decision Making. Retrieved from https://kt.ijs.si/MarkoBohanec/dexi.html

Bohanec, M., Žnidaršič, M., Rajkovič, V., Bratko, I., & Zupan, B. (2013). DEX methodology: three decades of qualitative multi-attribute modeling. *Informatica, 37*(1).

CYBERWISER.eu. (2019). CYBERWISER.eu - Cyber Range & Capacity Building in Cybersecurity. Retrieved from https://www.cyberwiser.eu/

Erdogan, G., & Refsdal, A. (2017). *A method for developing qualitative security risk assessment algorithms.* Paper presented at the International Conference on Risks and Security of Internet and Systems (CRiSIS'17).

Git. (2019). Git. Retrieved from https://git-scm.com/

Gradle. (2019). Gradle. Retrieved from https://gradle.org/

Gupta, S., Muntes-Mulero, V., Matthews, P., Dominiak, J., Omerovic, A., Aranda, J., & Seycek, S. (2015). *Risk-driven framework for decision support in cloud service selection.* Paper presented at the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'15).

Hüttermann, M. (2012). *DevOps for developers*: Apress.

ISO. (2018). ISO/IEC 27005:2018(en) Information technology — Security techniques — Information security risk management. In.

Jenkins. (2019). Jenkins. Retrieved from https://jenkins.io/

Jira. (2019). Atlassian Jira. Retrieved from https://www.atlassian.com/

Karabacak, B., & Sogukpinar, I. (2005). ISRAM: information security risk analysis method. *Computers & Security, 24*(2), 147-159.

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*: IT Revolution.

Kumar, D., Paccagnella, R., Murley, P., Hennenfent, E., Mason, J., Bates, A., & Bailey, M. (2018). *Skill squatting attacks on Amazon Alexa.* Paper presented at the 27th USENIX Security Symposium (USENIX'18).

Leukert, B. (2016). *IoT 2020: Smart and secure IoT platform*. Retrieved from http://www.iec.ch/whitepaper/pdf/iecWP-loT2020-LR.pdf

Lund, M. S., Solhaug, B., & Stølen, K. (2011). *Model-Driven Risk Analysis - The CORAS Approach*: Springer-Verlag Berlin Heidelberg.

Maven. (2019). Apache Maven Project. Retrieved from https://maven.apache.org/

Meulen, R. v. d. (2017). Retrieved from http://www.gartner.com/newsroom/id/3598917

Microsoft. (2019). Microsoft Threat Modeling. Retrieved from https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

Myrbakken, H., & Colomo-Palacios, R. (2017). *DevSecOps: a multivocal literature review.* Paper presented at the 17th International Conference on Software Process Improvement and Capability Determination (SPICE'17).

Nagios. (2019). Nagios. Retrieved from https://www.nagios.org/

NewRelic. (2019). New Relic. Retrieved from https://newrelic.com/

Nguyen, P., Ferry, N., Erdogan, G., Song, H., Lavirotte, S., Tigli, J.-Y., & Solberg, A. (2019). *Advances in deployment and orchestration approaches for iot-a systematic review.* Paper presented at the 2019 IEEE International Congress on Internet of Things (ICIOT'19).

RiskWatch. (2019). RiskWatch. Retrieved from https://riskwatch.com/

Solhaug, B., & Stølen, K. (2013). *The CORAS Language-Why it is designed the way it is.* Paper presented at the 11th International Conference on Structural Safety and Reliability (ICOSSAR'13).

Taivalsaari, A., & Mikkonen, T. (2017). A roadmap to the programmable world: software challenges in the IoT era. *IEEE Software, 34*(1), 72-80.

Thompson, A. (2019). Tool-support for risk-driven planning: risk-tool-frontend and risk-tool-backend. Retrieved from https://github.com/ribako/risk-tool-frontend

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*: Springer.

Wired. (2018). Hackers Found a (Not-So-Easy) Way to Make the Amazon Echo a Spy Bug. Retrieved from https://www.wired.com/story/hackers-turn-amazon-echo-into-spy-bug/

Xie, J., Lipford, H. R., & Chu, B. (2011). *Why do programmers make security errors?* Paper presented at the 2011 IEEE symposium on visual languages and human-centric computing (VL/HCC).

Zeleny, M. (1998). Multiple criteria decision making: eight concepts of optimality. *Human Systems Management, 17*(2), 97-107.