Handling Environmental Uncertainty in Design Time Access Control Analysis

Nicolas Boltz, Sebastian Hahner, Maximilian Walter, Stephan Seifferman, Robert Heinrich
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{nicolas.boltz, sebastian.hahner, maximilian.walter, stephan.seifermann, robert.heinrich}@kit.edu

Tomáš Bureš, Petr Hnětynka
Charles University
Prague, Czech Republic
{bures, hnetynka}@d3s.mff.cuni.cz

***Abstract*—The high complexity, connectivity, and data exchange of modern software systems make it crucial to consider confidentiality early. An often used mechanism to ensure confidentiality is access control. When the system is modeled during design time, access control can already be analyzed. This enables early identification of confidentiality violations and the ability to analyze the impact of what-if scenarios. However, due to the abstract view of the design time model and the ambiguity in the early stages of development, uncertainties exist in the system environment. These uncertainties can have a direct effect on the validity of access control attributes in use, which might result in compromised confidentiality.**

To handle such known uncertainty, we present a notion of confidence in the context of design time access control. We define confidence as a composition of known uncertainties in the environment of the system, which influence the validity of access control attributes. We extend an existing modeling and analysis approach for design time access control with our notion of confidence. For evaluation, we apply the notion of confidence to multiple real-world case studies and discuss the resulting benefits for different stages of system development. We also analyze the expressiveness of the extended approach in defining confidentiality constraints and measure the accuracy in identifying confidentiality violations. Our results show that using the notion of confidence increases expressiveness while being able to accurately identify access control violations.

***Index Terms*—software architecture, design time, access control, uncertainty, known unknowns, confidence**

I. Introduction

Modern distributed software systems, like Internet of Things (IoT), use and process data of the involved entities. As data might flow between different organizations or jurisdictions, there is much concern about unauthorized access of data. The high flexibility, complexity, and amount of data exchange in, for example, Industry 4.0 or Industrial Internet of Things (IIoT) systems makes it crucial to be able to ensure confidentiality [23]. A common measure to ensure confidentiality are access control mechanisms that authorize access or processing of data. In the early stages of the development process, making the right design decisions is crucial, as changes during subsequent stages are often more costly [3]. To already ensure confidentiality in early stages of system development and reduce potential future cost, design time models of the system can be analyzed to identify access control violations [25].

Depending on the underlying access control model, various sources of information are used to determine whether access is granted. Services use different sources, e.g., sensors, to acquire information. Services process the information and provide a resulting access control attribute, e.g., a role or location, that can be used by the access control system. However, these services introduce a degree of uncertainty to the access control system. This uncertainty results from influencing factors of the environment when acquiring the information. The influence of the factors on the access control system can result in reduced validity of statements about confidentiality. Depending on the system design, software architects or security experts might already be able to identify potential influencing factors. For example, the ability of a GPS service to provide an accurate position is highly dependent on its surroundings, like buildings. However, it might not be possible to fully estimate the actual characteristic or value of the factor, especially during early stages of development. As we know that these factors exist, but have only limited knowledge about the factor itself, they represent so-called *known uncertainty* [20].

While there already exist some approaches that enable the representation of access control during design time (e.g. [4, 14, 15, 25, 29]), there is a lack of design time approaches that provide a way of representing the associated *known uncertainty*. The lack of information reduces the quality in which the model represents the system under construction. This results in less accurate or even wrong predictions, made on the basis of the model. Taking the described *known uncertainty* into account increases the expressiveness in defining confidentiality constraints within the context of an analysis.

In this paper, we present an approach for handling known, environmental uncertainty in access control during design time. To this end, we enable the explicit representation of the known uncertainty and integrate the additional knowledge into an analysis. Our main contributions are:

C1 We propose a *notion of confidence* in the validity of access control attributes, based on *three types of influencing factors* of the system environment.

C2 We propose an access control analysis process for design time models that integrates *confidence* into model creation and constraint definition. We propose the use of *Fuzzy Inference Systems* (FIS) [17] to combine influencing factors to a resulting confidence value.

This paper is structured as follows: In Section II, we discuss related work. Section III presents a running example. In Section IV, we describe how we identify types of environmental factors and define our notion of confidence. Section V describes the extension of the existing design time access control analysis approach. Section VI presents the evaluation and discussion of the results. Section VII concludes the paper.

II. Related Work

In this section, we discuss related work within the field of design time uncertainty and access control.

Troya et al. [27] provide a survey of the uncertainty representation in software models. They summarize definitions, notations, and application domains of existing publications. Within their taxonomy of uncertainty, our approach can be classified as *measurement uncertainty*. They also state, that fuzzy set theory is commonly used to represent measurement uncertainty, which aligns with our approach.

Approaches that consider uncertainty during design time often cover the handling of uncertainty regarding architectural design decision-making. The impact of design decisions on a system's properties (e.g., security, scalability, etc.) is difficult to estimate, as in early stages of development, decisions are often made under uncertainty. To aid software architects in making informed decisions, there exist approaches regarding the design space exploration under uncertainty [9, 30]. As precisely quantifying the impact of design decisions is often not possible, Esfahani et al. [9] apply fuzzy logic. However, they only focus on uncertainty of the structure of the design time model and do not explicitly consider access control. While Walter et al. [30] are explicitly concerned with confidentiality and access control, they also only focus on uncertainty of the system structure.

Existing approaches regarding uncertainty in access control also utilize fuzzy logic, e.g., to represent security patterns [12] or to create a risk-adaptive access control model to cope with the uncertainty of risk [6]. However, these approaches cover other kinds of uncertainty or cover it in a more generalized way that can not be directly utilized to analyze confidentiality. In the field of access control, there exists a magnitude of approaches. Only a few take the described *known uncertainty* into account when making access decisions [1, 7]. However, these approaches do not allow the representation of multiple uncertainties or a way to define and analyze access control constraints during design time.

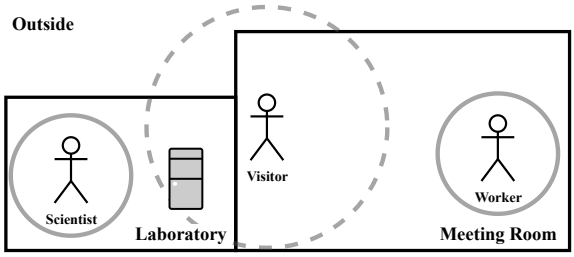

Figure 1: Change in accuracy of Visitor's GPS location.

III. Running Example

In this section, we introduce a running example. The example is based on a scenario where GPS positioning is used to derive location attributes for access control. The scenario revolves around a company, that has a dedicated research laboratory. We identify three actors:

Scientist: An actor who works in the laboratory and has exclusive access to the room.

Worker: An actor working for the same company as the scientist. In the meeting room waiting for the visitor.

Visitor: An actor that is not affiliated with the company but has a meeting with the worker in the meeting room.

In the scenario, a database exists, that is composed of sensitive research data. To enhance security, access to the database is only permitted if the accessing actor is located inside of the laboratory. The location of each actor is monitored using a GPS location service. To obtain the location of the worker and scientist, the location services communicate with their respective high-sensitivity GPS sensors (HSGPS). To obtain location information of the visitor, the location service communicates with the embedded GPS sensor of the visitor's mobile phone. For this example, we focus on the differences of running the two types of GPS sensors indoors [16]. Initially, the visitor is outside of the building, while the worker and scientist are inside. Once the visitor enters the building, the signal attenuation increases which results in a reduced signal-to-noise ratio (SNR). A reduced SNR increases the time needed to acquire satellite data and calculate a position (time to first fix, TTFF), and the amount of positioning errors [16]. While the HSGPS sensors of the worker and scientist are powerful enough to provide accurate positioning, the accuracy of the embedded GPS sensor of the visitor's mobile phone suffers when used inside, which leads to uncertainty regarding the actual position. The circles, shown around the actors in Figure 1, indicate the position that is estimated by the GPS sensors. As can be seen by the dashed line circle around the visitor, the reduced accuracy *introduces uncertainty* about the position. Deriving a location from this supplied position could place the visitor in the meeting room, laboratory, or outside. However, falsely placing the visitor in the laboratory, breaks the described security measure and theoretically grants the visitor access the laboratory database.

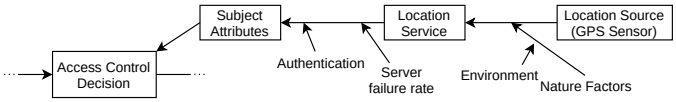

Figure 2: Excerpt of a trust chain, which incorporates types of factors for a GPS location example.

IV. Defining Confidence for Access Control

In this section, we draw conclusions about influencing factors and derive our notion of confidence, which tackles **C1**.

Hu et al. [13] provide a definition and considerations regarding attribute-based access control (ABAC). As a consideration, they describe a trust chain, concerning the attributes used to make access control decisions. Trust chains help determine the ownership of information and services, as well as requirements for technical solutions to establish valid trust relationships. The predicate of a trust relationship revolves around the idea that the access control system can trust the validity or correctness of the information, supplied by the owner, e.g., an authorization service. Depending on the access control model, many trust relationships are required to achieve a properly working access control system. Figure 2 shows a trust chain of a location attribute. The *subject attributes* have a trust relationship with the *location service*, which in turn has a trust relationship with the *location source*. As indicated by the additional arrows, the trust relationships are impacted by influences of the environment of the system.

Ardagna et al. [1] define a confidence value that combines the service used for location determination and environmental conditions. Similar to the trust of Hu et al. [13], confidence describes the certainty that a location is valid, which is dependent on the environment. The spatial context definition of Cuppens and Miege [7] precisely combines the hardware and software architecture and the environment as an additional condition for access control. Based on the work of Hengartner and Zhong [11] we can identify environmental factors that influence the uncertainty in access control. These factors can be applied to access control attributes, align well with the idea of trust relationships of Hu et al. [13] and can be represented in software architecture. These environmental factors are:

Source: The source that is used by the service to obtain information that is needed for the access control attribute. Location information, for example, might be derived from a physical access control mechanism or GPS data. Each of these sources has a different margin of error or accuracy.

Natural Factors: The nature factors that either influence the source's or the service's ability to return correct information or attributes. Depending on the sensitivity and rating, the accuracy with which GPS sensors can determine a location is heavily influenced by the physical environment, e.g., surrounding buildings [16], and weather conditions, e.g. cloudy sky [1].

Age: The age of an attribute, which depending on the underlying information might degrade validity. Depending on the attribute, age can be a combination of the time it took to gather the information from the source, the time it takes to process the information to an access control attribute and the overall time that has passed since this access control attribute has been created.

The previously described factors and the associated uncertainties are all known, but especially when multiple factors need to be combined, their influence on the validity of an access control attribute is hard to describe. To provide a way of representing this known uncertainty, we define our *notion of confidence* in the validity of access control attributes. Confidence combines the known uncertainty of environmental factors to a single value, which represents the level of confidence that a corresponding attribute is valid. We further define that confidence and the known uncertainty are not directly associated with an access control attribute, but rather with the service that is used by the system to derive the attribute. A service uses a source to receive information. The service processes this information into an attribute and provides it to the access control system.

The factors in our running example (SNR, TTFF, amount of positioning errors) decrease the overall accuracy and thereby influence confidence. The SNR mainly depends on the natural environment of the GPS sensor and can be regarded as a *Nature Factor*. The TTFF describes the time needed to acquire satellite data and calculate a position, which aligns with the description of *Age*. The amount of positioning errors depends on the computing power and design of the used sensor. As the sensors are the source of the services, the amount of positioning errors can be regarded as the factor type *Source*. An increase in any of the three described factors might reduce confidence in the location attribute. In our running example, the SNR outside and inside is the same for both, the HSGPS and embedded mobile GPS sensor. However, for the more powerful HSGPS sensors, the TTFF and amount of positioning errors do not increase as drastically with an increase in SNR, compared to the mobile GPS sensor. This results in *high confidence* in access control attributes, that are derived by services that use the HSGPS sensors, and *low confidence* in access control attributes derived by services using the mobile phone GPS sensor.

V. Extending Architecture-Based Access Control Analysis

In this section, we first describe our approach to calculating confidence using Fuzzy Inference Systems (FIS) [17]. To accomplish this, we create a FIS metamodel to represent acquired factors and define calculation rules. Secondly, we describe how we extend the design time confidentiality analysis process of Seifermann et al. [25], to create an access control analysis process, that integrates confidence, which tackles **C2**.

A. Calculating Confidence from Factors

For over two decades, fuzzy sets and fuzzy logic have been used to describe uncertainty [17]. Fuzzy logic is also already used in related work regarding design time uncertainty and uncertainty in access control [6, 9, 12, 27].

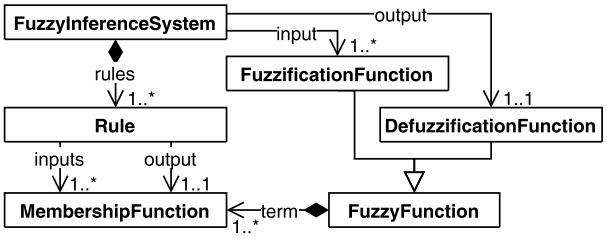

Figure 3: Class diagram excerpt of the FIS metamodel.

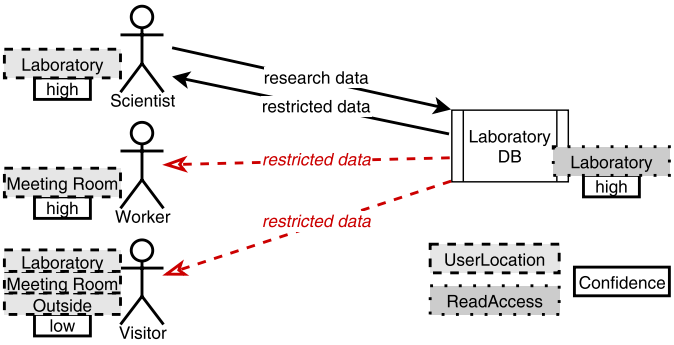

Figure 4: Data flow diagram of the running example.

For our approach, we propose the use of FISs [17], to combine factors to a resulting confidence value. A FIS is made up of four main components [17]: 1) Fuzzifier, 2) Defuzzifier, 3) Fuzzy Inference Engine, 4) Knowledge Base. A *fuzzifier* first translates the crisp values of environmental factors into fuzzy values, by applying them to a set of membership functions. The *fuzzy inference engine* uses the fuzzy input from the fuzzifier and created rules to infer a fuzzy output. The *defuzzifier* translates the fuzzy output of the inference engine to a confidence value, by aggregating the fuzzy outputs. We implement a FIS metamodel as a way to enable software architects to create FISs, which is independent of existing tools.Figure 3 shows an excerpt of the class diagram representation of the FIS metamodel. We represent the environmental factors as *fuzzification functions*, which are made up of *membership functions*. Each membership function defines a fuzzy set and represents a *linguistic value* the environmental factor can take on. The corresponding membership functions define a degree of membership to the fuzzy set within an interval of $[0, 1]$. Using the value of the environmental factor, e.g. $30\,\%$ SNR, the fuzzifier calculates a degree of membership for each fuzzy set of the associated fuzzification function. In our running example, the SNR can take on the linguistic values of 'low', 'medium', and 'high'. The linguistic values of age are 'new' and 'old'. As an example the environmental factor values of $30\,\%$ SNR would result in the membership degree of 0 for 'high', 0.75 for 'medium', and 0.5 for 'low'. *3 minutes of age* would result in the membership degree of 0 for 'new' and 1 for 'old'. Rules to combine the environmental factors are defined by combining a linguistic value of each environmental factor and defining a result. A rule, which combines the most negative linguistic values of SNR and age and consequently results in 'low' confidence is defined like this: *IF SNR is 'low' AND age is 'old' THEN confidence is 'low'*. Depending on how the FIS is set up, the membership degrees of the linguistic values of a rule are combined. The result of the rule with the highest combined membership degrees is returned.

We chose to use FISs for our approach, as they have positive properties described in the following, which we take advantage of. Klir and Yuan [17] describe that through fuzzification an enhanced ability to model real-world problems is gained, lowering overall solution cost. The use of fuzziness also serves to better capture human common-sense reasoning and decision making [17, pages 32f.]. When setting up a calculation rule in general, the system's properties and environment are abstracted and simplified. As a result, information about the inputs and their influence on the result is lost. A FIS conserves more knowledge about the inputs and their influence on the calculated confidence value than a conventional mathematical function by mapping inputs to membership functions and working with natural language concepts. Additionally, a FIS is capable of capturing the meanings of sentences in natural language [17, p. 32f.], which enables, e.g., a software architect or security expert to easily map statements or requirements about the influence of environmental factors on confidence, to calculation rules.

We implement a model-to-text transformation, which creates a textual representation of a FIS model instance. To run a FIS instance, we import the created textual representation using fuzzy logic control libraries. For our implementation, we use the FuzzyLite Libraries for Fuzzy Logic Control [22].

B. Applying Confidence to Access Control Analysis

To implement our calculated confidence in a design time access control analysis, we chose to extend the existing approach of Seifermann et al. [25]. They utilize the data flow diagram (DFD) notation of DeMarco [8] and extend the syntax with confidentiality-related information. Most notably, they define characteristics, which represent properties or attributes of a node or data. Each characteristic has a characteristic type. The characteristic type specifies a set of fixed value properties or attributes, called labels. Each property or attribute a characteristic represents is represented by a label of the characteristic's characteristic type. Additionally, they add a data processing behavior definition that can be assigned to any kind of node in the diagram. Processing behavior defines how attribute labels are changed if data flows through a node. An automated model transformation translates the DFDs to a Prolog logic program. The resulting program is analyzed for violations against access control requirements using queries that propagate labels along the data flows. As confidence is directly related to such attributes, we extend characteristics by adding confidence labels. In the same way, we extend characteristic types by adding a set of possible confidence labels for a corresponding characteristic. Figure 4 shows an extended DFD representation of the situation in our running example. The visitor has entered the meeting room and the worker and visitor try to read from the laboratory DB. The

highlighted edges are the data flows that should be identified as violations. Conventional analysis approaches, like the one of Seifermann et al. [25], are only able to identify the data flow to the worker as a violation, as the visitor has the required laboratory label. However, by comparing the attribute and confidence label pairs of the laboratory DB with the attribute and confidence pairs of the actors, *both* data flows can be correctly identified as violations.

To realize an analysis that takes confidence into account, we extend the underlying DFD metamodel, the semantics of the DFD to Prolog transformation, and modify the syntax of the resulting Prolog logic program. We extend the DFD metamodel, by adding a FIS to each characteristic and defining an additional set of possible confidence labels for the corresponding characteristic types. We further extend the data processing behavior definitions to also define how confidence labels are changed during data flow. During the transformation, the FIS calculates the actual confidence label of the corresponding characteristic. The resulting confidence labels are transformed to Prolog similar to how attribute labels are transformed. The syntax of the Prolog program is modified in a way, that every logic statement that included attribute labels also includes a corresponding confidence label. The code of our extension of the analysis process, including the FIS metamodel, the extension of the DFD metamodel, and the modified DFD to Prolog transformation, is available in our replication package [5].

VI. Evaluation

This section describes the used case studies, evaluation design, and results, as well as the threats to the validity of our approach. We chose a case study-based evaluation, as it is commonly done for design-time security approaches. Also, case studies can show the applicability and enable comparability between different approaches [28].

A. Case Studies for Evaluation

We reuse the six case studies that were used by the original confidentiality analysis approach of Seifermann et al. [25]. Three of the case studies represent the RBAC model and were originally described as part of the iFlow approach by Katkalov et al. [15]. The three remaining case studies represent the Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Attribute-Based Access Control (ABAC) models. For more detailed information Seifermann et al. [25] have created a data set, which contains detailed descriptions of the case studies and examined scenarios.

As the described case studies do not include information that can be interpreted and used to calculate confidence, we need to extend the existing case studies. Therefore we add real-world environmental factors that are comprehensible and consistent with the case study description. We extract exact values for these environmental factors that produce a potential issue from existing research. As the research regarding real-world environmental factors, without any background knowledge, is very time-consuming, it would have to be done for each case study individually to be consistent. We base most of our evaluation on the ABAC case study and our running example (see Section III), which aligns with DAC case study.

The ABAC case study describes a banking system, which is deployed in the USA and Asia. Actors in the case study can be Clerks or Managers. Clerks can register customers, look them up and determine a credit line. Managers can additionally also register celebrities, or move customers between regions. Seifermann et al. [25] define four attribute labels, which are used to annotate the DFD and define access control goals: 1) The *Role* of an actor, e.g. Clerk, Manager. 2) The *Location* of an actor, e.g. USA, Asia. 3) The *Status* of a customer, which data is processed, e.g. Regular, Celebrity. 4) The *Origin* of the customer data, e.g. USA, Asia. To add our notion of confidence and cover all aspects of the model, we derive *two* main scenarios from the described case study. The first covers the confidence in the *Role* attribute of the actors. The second scenario covers the confidence in the *Location* attribute of the actors. Due to space constraints, we only describe the second scenario, which covers confidence in the *Location* attribute, in-depth. However, we have created a dataset [5], which contains descriptions and illustrations for all our case studies, as well as the actual DFD instances used for our evaluation.

The scenario that introduces a confidence issue in the *Location* of actors focuses on the resolution of the country geolocation from the IP addresses of the actors in the system. Actual accurate information about the geolocation of a related IP address is only known by the internet service provider. This data is not available for commercial applications, like the banking system of the ABAC case study. The banking system has to rely on public or commercial geolocation databases and geolocation services [18, 19]. These databases use specially aggregated blocks of IP addresses to match the IP addresses to geolocations [26]. For our scenario, we narrow down the locations to the specific regions *USA* and *Hong Kong*. As a cost-saving measure, the banking system switches the geolocation database, which provides the location information, from the commercial NetAcuity [19] database to the free MaxMind-GeoLite [18] database. For the USA, this change only reduces the country geolocation accuracy by a negligible amount. However, for countries like Hong Kong, this change reduces the country geolocation accuracy by more than 40% [10]. The low accuracy could result in incorrect system behavior once deployed and opens up the possibility for attacks utilizing the inaccurate country geolocation resolution. In our example, the location attribute is not always properly set to Hong Kong. As a result, the access control system, which is in place, might block most data flows to the actors, effectively locking the clerk out of accessing the Hong Kong customer storage.

The DFD approach of Seifermann et al. [25] has no way of representing this change. With our notion of confidence, this change in databases can be represented in the model.

B. Evaluation Questions and Design

For our evaluation, we use the Goal Question Metric (GQM) approach [2] and define the following goals: **G1** Examine the

applicability of the approach in different stages of system development. **G2** Examine the accuracy of the approach in identifying access control violations.

Goal **G1** evaluates the applicability. We ask: **Q1.1** Is the required information to apply the proposed notion of confidence to the described metamodel and analysis process available during design time? **Q1.2** Does adding the notion of confidence to the described metamodel and analysis process produce additional value if a system is already deployed and running? **Q1.3** Is the expressiveness of the original DFD approach [25] affected by the addition of the proposed notion of confidence?

To answer question **Q1.1** we extract what-if scenarios from the case studies used to evaluate the approach of Seifermann et al. [25]. We analyze whether it is possible to identify real-world environmental factors that can be applied to the extracted scenarios, by exclusively using the extracted design time DFD and existing research. For question **Q1.2**, we also take situation-dependent runtime information into account.

To answer question **Q1.3** we compare the expressiveness of the original DFD approach of Seifermann et al. [25], to our extended version. To evaluate the expressiveness, Seifermann et al. [25] use their extended DFD syntax to create instances of the six case studies that cover typical access control functionality (see Subsection VI-A). We use our extended DFD approach to model semantically matching case studies and scenarios, and compare how far we can cover the same set of cases. We create case studies and scenarios that semantically match the original case studies and scenarios, by adding *default* confidence labels to every attribute label. *Default* confidence labels represent complete confidence in the validity of the corresponding attribute, effectively emulating a DFD without confidence labels.

Goal **G2** evaluates the accuracy. We ask **Q2.1** What is the accuracy of the analysis in identifying issues that result from mismatched attributes? **Q2.2** What is the accuracy of the analysis in identifying issues that result from low confidence? **Q2.3** What is the accuracy of the analysis in identifying issues that result from a combination of mismatched attributes and low confidence? To answer the questions regarding goal **G2**, we base our evaluation on the way the data flow diagram approach of Seifermann et al. [25] was evaluated. They define a pair of scenarios for every DFD instance of the six case studies described in Subsection VI-A. A pair consists of the scenario without an issue and the scenario where an issue has been introduced. Seifermann et al. [25] derive the issues either from related work or by themselves. Each issue leads to violations concerning the analysis. By adding our notion of confidence, we identify three different kinds of issues concerning the analysis. An issue can be due to mismatched attribute labels, due to mismatched confidence labels, or a combination of both. Based on this distinction, we define four different types of scenarios, which need to be considered to evaluate the accuracy of our contribution: *S0)* No issue; *S1)* An issue that is due to mismatched attribute labels. Introduced by adding an illegal data flow to the DFD; *S2)* An issue that is due to mismatched confidence labels. Introduced by identifying a scenario based on the DFD that reduces the confidence label of an attribute; *S3)* An issue is introduced, that is due to mismatched attribute and confidence labels, by combining *S1* and *S2*. We create pairs of scenarios, which consist of an *S0* scenario and either an *S1*, *S2* or *S3* scenario, depending on the evaluation question. For question **Q2.1**, we reuse the six scenarios that were used for the DFD approach of Seifermann et al. [25]. We add *default* confidence labels to the scenarios of Seifermann et al. [25]. This way, confidence should not introduce an issue to the *S0* scenarios and should not affect the mismatched attributes of the *S1* scenarios. We also use our running example without invalid data flows as *S0* scenario and add a data flow to the worker to form a *S1* scenario. For question **Q2.2**, we use the ABAC case study as *S0* and the two extended ABAC scenarios described in Subsection VI-A as *S2* scenarios. We also use our running example *S0* scenario and add a data flow to the visitor as a third *S0* and *S2* scenario pair. For question **Q2.3**, we combine the *S1* ABAC scenario from Q2.1 with the *S2* ABAC scenario with a role violation of Q2.2 to form a *S3* scenario. We also extend the *S1* scenario of our running example, by lowering the confidence in the worker's location attribute to create a second *S0* and *S3* scenario pair.

The metrics used to evaluate the questions concerned with goal **G2** are precision and recall [21]. Precision (p) calculates the ratio of correctly identified issues to the sum of overall identified issues: $p = \frac{t_p}{t_p+f_p}$. Recall (r) calculates the ratio of correctly identified issues to the number of actually introduced issues: $r = \frac{t_p}{t_p+f_n}$.

C. Evaluation Results and Discussion

To answer **Q1.1**, we reckon that information about sensors and environmental factors can initially be assumed and refined during the iterative development process. As we have shown with our running example (see Section III) and the extended ABAC scenarios (see Subsection VI-A), it is possible to make some general assumptions about environmental factors from existing literature [10, 26]. More precise measurements, e.g., regarding the signal attenuation inside the actual buildings, could be conducted, or technical specifications of used sensors become available during the subsequent development process. Through the iterative refinement of the model, the results of an analysis using the model can increasingly reflect the real-world situation the system is to be deployed in. If a design time model is already used to validate system requirements regarding access control, our confidence can be added once system requirements change in a way that makes it necessary to handle the known uncertainty of the environment.

For **Q1.2**, the real-world situation is already known as we consider deployed systems. We reckon that during runtime, information about environmental factors is already present. The additional information enables the software architect to create a very detailed model representation of the system, including our notion of confidence. These detailed models can be used to investigate changes to the system and their potential

impact on the access control of the deployed systems before implementing them.

Our extended location ABAC scenario described in Subsection VI-A presents a situation where using our notion of confidence is especially worthwhile for deployed systems. In the described situation, changes in the environmental factors do not introduce new potential issues with confidentiality but rather introduce functional issues.

Changes to the database used for country geolocation affect the accuracy in resolving the geolocation from IP addresses of the actors in Hong Kong. As a result, data flows might be blocked, effectively locking them out of accessing the Hong Kong customer storage. With our notion of confidence, this information is added to the design time model representation of the system. The changes to the database result in lower confidence in the location attribute and therefore enable an analysis approach to identify the issue beforehand, preserving the functionality of deployed systems.

To answer question **Q1.3** we compare the expressiveness of the DFD approach of Seifermann et al. [25], to our extended version. To evaluate the expressiveness, Seifermann et al. [25] use their extended DFD syntax to create instances of six case studies, that cover a broad spectrum of access control models. Further, Seifermann et al. [25] define Prolog queries for each case study. As we have only slightly modified the design time model and analysis to consider confidence, we can simply modify the DFD instances of Seifermann et al. [25] to use our extended DFD syntax, while still representing the semantically identical case study. Similarly, we can replicate the corresponding queries almost exactly. This suggests, that adding our notion of confidence did not negatively impact the expressiveness. The added confidence, on the other hand, enables attributes to incorporate information about environmental factors. By including confidence, environmental factors have a direct effect on the expressive power of the corresponding attribute. This leads to an overall increase in expressiveness.

To answer the questions regarding goal **G2**, we executed the model instances for the *S0* and *S1* scenario pairs (**Q2.1**), *S0* and *S2* scenario pairs (**Q2.2**) and *S0* and *S3* scenario pairs (**Q2.3**) of the described case studies. For each question, we correctly did not identify non-existing issues in the *S0* scenarios. For the seven *S1* scenarios defined for **Q2.1**, the three *S2* scenarios defined for **Q2.2** and the one *S3* scenario defined for **Q2.3** we were able to exclusively identify all contained issues correctly.

For **Q2.1**, this results in a precision of $p = \frac{7}{7+0} = 1.0$ and a recall of $r = \frac{7}{7+0} = 1.0$. When comparing results, we can see that the addition of confidence did not impact the ability of the DFD approach of Seifermann et al. [25] to correctly identify access control issues, that result from mismatched attributes. This shows, that while we extend the ability of the DFD approach to represent and identify issues that result from confidence, we did not impair the original accuracy.

For **Q2.2**, this results in a precision of $p = \frac{3}{3+0} = 1.0$ and a recall of $r = \frac{3}{3+0} = 1.0$. We did expect similar results, as in our extension of the DFD approach, attribute issues and confidence issues both result from mismatched labels. The way confidence labels are compared is equal to how the attribute labels are compared. Bad accuracy for *S1* scenarios should result in bad accuracy in *S2* scenarios and vice versa.

For **Q2.3**, this results in a precision of $p = \frac{2}{2+0} = 1.0$ and a recall of $r = \frac{2}{2+0} = 1.0$. We did expect similar results, as the analysis is already able to identify issues from mismatched attributes and confidence independently. As we use the same query for the ABAC *S3* scenario that was used for Q2.1 and Q2.2, solutions that were found for Q2.1 and Q2.2 can still be found, effectively creating the union of both results.

D. Threats to Validity

As our approach is evaluated with case studies, we discuss the internal validity, external validity, construct validity, and reliability of our contribution, as characterized by Runeson et al. [24]. As our implementation of an analysis is realized with an extension of the DFD approach of Seifermann et al. [25], the same threats of validity apply. Due to lack of space, we can not elaborate on these threats. Additionally, we identify the following threats to validity regarding our approach:

The main threat to the internal validity of our evaluation of applicability is whether our chosen questions and discussed results hold enough weight to make a proper statement about applicability. The discussed additional value of our contribution for runtime systems in question Q1.2 highly depends on the system and its area of application. However, we were able to create multiple problematic real-world situations, using values of environmental factors taken from existing literature.

The main threat to external validity of our accuracy evaluation stems from the small number of S2 and S3 scenarios we cover when answering questions Q2.2 and Q2.3. We try to partly mitigate this threat by defining our S2 and S3 scenarios using real-world environmental factors and measurements from existing literature, removing our bias towards using solely made-up values that positively influence the accuracy.

To mitigate threats to construct validity and properly structure our evaluation we applied the GQM approach. Using metrics to summarize the applicability of an approach is not sufficiently possible, as the applicability is generally very dependent on variations and limitations of the actual system an approach is applied at. Finally, the precision and recall metrics are common for evaluating accuracy and are used to evaluate related work [4, 25, 29].

To mitigate threats regarding the reliability of our evaluation, we publish a data set and replication package [5]. The replication package contains the implementation and model extension, as well as all model instances of every analyzed case study. The data set contains further illustrations and descriptions of the case studies used for the evaluation. This allows others to better reproduce our results.

VII. Conclusion and Future Work

In this paper, we presented a design time approach of handling uncertainties of a systems environment within the context of access control. Based on related work, we identified three types of factors in the environment of a software system,

that influence the validity of the resulting access control attributes. By combining these factors, we introduced a notion of confidence in the validity of attributes. Using fuzzy inference systems, we represent and combine these environmental factors to a resulting confidence. Based on the DFD approach of Seifermann et al. [25], we implemented an access control analysis process for design time models, which integrates confidence in the model creation and constraint definition.

When considering the uncertainty taxonomies of Perez-Palacin and Mirandola [20] or Troya et al. [27], we are only able to handle a small part of the overall uncertainty. In future work, we aim to look into applying our notion of confidence to a broader spectrum of uncertainty.

Acknowledgment

The work of Nicolas Boltz is funded by the German Federal Ministry for Economic Affairs and Climate Action within the project Software-Defined Car (SofDCar) due to a resolution of the German Bundestag (grant number 19S21002K). This work is also partially funded by the DFG (German Research Foundation) – project number 432576552, HE8596/1-1 (FluidTrust), by the Czech Science Foundation project 20-24814J, by Charles University institutional funding SVV 260451, from the topic Engineering Secure Systems of the Helmholtz Association (HGF), and by KASTEL Security Research Labs (46.23.03).

References

[1] Claudio A Ardagna et al. "Supporting location-based conditions in access control policies". In: *ACM CCS*. 2006, pp. 212–222.

[2] Victor R Basili and David M Weiss. "A methodology for collecting valid software engineering data". In: *TSE* 6 (1984), pp. 728–738.

[3] B. Boehm and V.R. Basili. "Top 10 list [software development]". In: *Computer* 34.1 (2001), pp. 135–137.

[4] Nicolas Boltz et al. "Context-Based Confidentiality Analysis for Industrial IoT". In: *SEAA*. 2020, pp. 589–596.

[5] Nicolas Boltz et al. *Data Set and Replication Package*. DOI: 10.5281/zenodo.6417441.

[6] Pau Chen Cheng et al. "Fuzzy Multi-Level Security: An experiment on quantified risk-adaptive access control". In: *IEEE SP* (2007), pp. 222–227.

[7] Frédéric Cuppens and Alexandre Miege. "Modelling contexts in the Or-BAC model". In: *ACSAC*. 2003, pp. 416–425.

[8] Tom DeMarco. "Structure analysis and system specification". In: *Pioneers and Their Contributions to Software Engineering*. 1979, pp. 255–288.

[9] Naeem Esfahani et al. "GuideArch: Guiding the exploration of architectural solution space under uncertainty". In: *ICSE*. 2013, pp. 43–52.

[10] Manaf Gharaibeh et al. "A look at router geolocation in public and commercial databases". In: *IMC*. 2017, pp. 463–469.

[11] Urs Hengartner and Ge Zhong. "Distributed, Uncertainty-Aware Access Control for Pervasive Computing". In: *PerComW*. 2007, pp. 241–246.

[12] Hilary H Hosmer. "Using fuzzy logic to represent security policies in the multipolicy paradigm". In: *ACM SIGSAC Review* 10.4 (1992), pp. 12–21.

[13] Vincent C. Hu et al. "Guide to Attribute Based Access Control (ABAC) Definition and Considerations". In: *NIST Special Publication* 800.162 (2014).

[14] Jan Jürjens and Pasha Shabalin. "Automated verification of UMLsec models for security requirements". In: *UML*. Springer. 2004, pp. 365–379.

[15] Kuzman Katkalov et al. "Model-driven development of information flow-secure systems with IFlow". In: *SOCIALCOM*. 2013, pp. 51–56.

[16] Mikkel Baun Kjærgaard et al. "Indoor positioning using GPS revisited". In: *PerCom*. 2010, pp. 38–56.

[17] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*. Vol. 4. Prentice hall New Jersey, 1995.

[18] *MaxMind-GeoLite2*. URL: https://dev.maxmind.com/geoip (visited on 07/23/2021).

[19] *NetAcuity*. URL: https://www.digitalelement.com/solutions/ip-location-targeting/netacuity/ (visited on 07/23/2021).

[20] Diego Perez-Palacin and Raffaela Mirandola. "Uncertainties in the modeling of self-adaptive systems: A taxonomy and an example of availability evaluation". In: *ICPE*. 2014, pp. 3–14.

[21] David M. W. Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". In: *JMLT* 2.1 (2020), pp. 37–63.

[22] Juan Rada-Vilela. *The FuzzyLite Libraries for Fuzzy Logic Control*. 2018. URL: https://fuzzylite.com/.

[23] Vasja Roblek et al. "A Complex View of Industry 4.0". In: *SAGE Open* 6.2 (2016).

[24] Per Runeson et al. *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, 2012.

[25] Stephan Seifermann et al. "Detecting violations of access control and information flow policies in data flow diagrams". In: *JSS* 184 (2022).

[26] Yuval Shavitt and Noa Zilberman. "A geolocation databases study". In: *J-SAC* 29.10 (2011), pp. 2044–2056.

[27] Javier Troya et al. "Uncertainty representation in software models: a survey". In: *SoSyM* 20.4 (2021), pp. 1183–1213.

[28] Alexander Van den Berghe et al. "Design notations for secure software: a systematic literature review". In: *SoSyM* 16.3 (2017), pp. 809–831.

[29] Maximilian Walter et al. "Architectural Attack Propagation Analysis for Identifying Confidentiality Issues". In: *ICSA*. 2022, pp. 1–12.

[30] Maximilian Walter et al. "Architectural Optimization for Confidentiality under Structural Uncertainty". In: *ECSA Post-Proceedings*. accepted to appear. 2022.