Automatic online quantification and prioritization of data protection risks

Sascha Sven Zmiewski
University of Duisburg-Essen
Essen, Germany

Jan Laufer
University of Duisburg-Essen
Essen, Germany

Zoltán Ádám Mann
University of Amsterdam
Amsterdam, Netherlands

ABSTRACT

Data processing systems operate in increasingly dynamic environments, such as in cloud or edge computing. In such environments, changes at run time can result in the dynamic appearance of data protection vulnerabilities, i.e., configurations in which an attacker could gain unauthorized access to confidential data. An autonomous system can mitigate such vulnerabilities by means of automated self-adaptations. If there are several data protection vulnerabilities at the same time, the system has to decide which ones to address first. In other areas of cybersecurity, risk-based approaches have proven useful for prioritizing where to focus efforts for increasing security. Traditionally, risk assessment is a manual and time-consuming process. On the other hand, addressing run-time risks requires timely decision-making, which in turn necessitates automated risk assessment.

In this paper, we propose a mathematical model for quantifying data protection risks at run time. This model accounts for the specific properties of data protection risks, such as the time it takes to exploit a data protection vulnerability and the damage caused by such exploitation. Using this risk quantification, our approach can make, in an automated process, sound decisions on prioritizing data protection vulnerabilities dynamically. Experimental results show that our risk prioritization method leads to a reduction of up to 15.8% in the damage caused by data protection vulnerabilities.

CCS CONCEPTS

• **Security and privacy** → *Privacy protections*; *Intrusion detection systems*; *Vulnerability management*.

KEYWORDS

risk assessment, data protection, vulnerability, risk prioritization, self-adaptation

ACM Reference Format:
Sascha Sven Zmiewski, Jan Laufer, and Zoltán Ádám Mann. 2022. Automatic online quantification and prioritization of data protection risks. In *The 17th International Conference on Availability, Reliability and Security (ARES 2022), August 23–26, 2022, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3538969.3539005

ARES 2022, August 23–26, 2022, Vienna, Austria

ACM ISBN 978-1-4503-9670-7/22/08…$15.00
https://doi.org/10.1145/3538969.3539005

1 INTRODUCTION

An increasing number of software applications process personal data, raising concerns about data protection [1, 18]. This is mirrored by recent data protection regulation, such as the General Data Protection Regulation (GDPR) of the European Union (EU). Data protection covers several aspects of security and privacy. In particular, a key data protection requirement is that personal data should only be accessible by authorized actors [26].

Software can be made reasonably secure *by design* for a known and stable environment. However, increasingly often, the environment is subject to dynamic changes at run time [4]. Thus, a static system configuration may not be able to ensure the continued satisfaction of data protection requirements. For example, an application running in the cloud may be migrated during run time from one data center to another with different security properties, requiring different data protection mechanisms of the application to be active. To address this problem, previous work proposed RADAR, an approach using *self-adaptation* to detect and mitigate data protection vulnerabilities automatically at run time [10].

While self-adaptation is a powerful method for the continued assurance of data protection in dynamic environments, it also poses new challenges. One such challenge is what to do if multiple data protection vulnerabilities are detected simultaneously, and not all of the detected data protection vulnerabilities can be mitigated at the same time. In such a case, the identified data protection vulnerabilities have to be *prioritized*, i.e., it has to be decided which ones to mitigate first.

For prioritizing security issues, risk-based approaches have proven useful [25]. Security management standards and best practices, such as ISO 2700x, advocate continuous identification, assessment, and handling of security risks [3]. This ensures that efforts to improve security are put into the areas where this is the most beneficial. Thus, quantifying the risk associated with different data protection vulnerabilities could be a good basis for prioritizing them.

Traditionally, risk assessment is done by human experts in a time-consuming process. In contrast, we need timely mitigation of data protection vulnerabilities that arise at run time, before they can be exploited by actual attacks. To allow timely mitigation, our aim is to *automatically* assess the risk associated with data protection vulnerabilities at run time as the vulnerabilities emerge.

In this paper, we propose an approach for automatically quantifying the risks associated with data protection vulnerabilities arising at run time. Our approach is based on a new mathematical model that formalizes the key properties of data protection vulnerabilities: the dependence of the caused damage on the type and amount of the involved data and on the time that elapsed since the vulnerability arose. This model allows us to automatically compute the risk values of vulnerabilities, and thus to prioritize the vulnerabilities

based on risk. We implement our approach as an extension to the RADAR self-adaptive data protection system [10]. We investigate two different methods for using the risk values to prioritize the identified data protection vulnerabilities in RADAR. Controlled experiments show the results of risk prioritization and the effect of different parameters. In particular, prioritization can save up to 15.8% of the damage stemming from data protection vulnerabilities.

The rest of the paper is organized as follows. Section 2 explains the background of our study. Section 3 describes our approach, and Section 4 shows how our approach can be embedded into RADAR. Section 5 evaluates our approach. Related work is described in Section 6, and Section 7 concludes the paper.

2 PRELIMINARIES: THE RADAR APPROACH

The research presented in this paper is performed in the context of RADAR, an approach for ensuring data protection in systems that are exposed to changes at run time [10]. RADAR automatically identifies threats to data protection at run time and determines the best adaptations to keep data protected. RADAR serves as an example system to motivate and validate our approach. However, our approach could also be integrated into other systems that detect and mitigate data protection vulnerabilities.

The elements of RADAR are shown in Fig. 1 and explained below.

Run-time model and meta-model: RADAR performs reasoning on a run-time model (box 2.5) representing the data-protection-related aspects of the current configuration of the system and its environment. The run-time model is kept up to date using monitoring. The meta-model (box 1.3) specifies which types of nodes and edges can exist in the run-time model and which attributes the nodes can have. The meta-model supports node types representing for example data records and datasets as well as software components and computing nodes. The meta-model provides a common language for the run-time model and for other RADAR artefacts (PCPs and adaptation rules – see below).

PCPs and PCP instances: A problematic configuration pattern (PCP - box 1.1) describes a sub-structure which, if found in the run-time model, means that there is a possibility of a data breach. A PCP can represent a vulnerability stemming from one system component or from the interplay of multiple components. A PCP instance (box 2.2) is a specific instance of such a pattern found in the run-time model. Multiple PCP instances can co-exist, even multiple instances of the same PCP.

Adaptation rules and adaptations: Adaptation rules (box 1.2) describe how the system may be adapted to mitigate PCPs. An adaptation rule may involve adding or removing objects and relations as well as changing attribute values in the run-time model. Multiple adaptation rules may correspond to a PCP. An adaptation rule consists of the associated PCP, preconditions required for the adaptation rule to be applicable, and the changes to be carried out. If an adaptation rule is applicable, the adaptation (i.e., an instance of the adaptation rule – box 2.3) can be applied by performing the changes described by the rule.

Problematic configuration identification: A pattern-matching algorithm (box 2.1) is used to detect PCP instances in the run-time model. A subset of objects in the run-time model matches a PCP, if objects, attribute values, and relations described in the PCP match a respective counterpart in the run-time model, also accounting for any prohibition of nodes, relations, or attribute values by the PCP. The pattern-matching algorithm extends basic graph pattern matching by supporting object types, inheritance, attributes, and named relations.

Reconfiguration: If at least one PCP instance is detected in the run-time model, RADAR searches for a way to reconfigure the system so as to mitigate the PCP instance(s) (box 2.4). RADAR uses an adaptation planning algorithm to identify the best adaptation. The algorithm needs to take into account that there may be multiple PCP instances in the run-time model, there may be multiple adaptations to mitigate a given PCP instance, and an adaptation to mitigate a PCP instance may also mitigate or create other PCP instances. A sequence of adaptations may be needed to mitigate all PCP instances. In this sequence, the order of the adaptations may be important because an adaptation may become applicable only after another adaptation was carried out.

The algorithm implements a *bounded backtrack search* in the space of possible run-time models reachable from the current run-time model via sequences of adaptations. A search tree is constructed to reason about possible adaptation sequences. Each node in the tree represents a possible run-time model, starting with the current run-time model as the root node. A child node represents the run-time model obtained from the run-time model of the parent node through an adaptation. The search tree is built during the search, by iteratively adding unexplored child nodes of the nodes already visited. Different strategies for selecting the next node to explore were evaluated [10]. Selecting the node with the lowest number of PCP instances led to the best results. Thus, this strategy is used in this work. The search stops if either the search tree has been exhausted or a predefined termination criterion (time limit) is met. Throughout the search, the algorithm keeps track of the best solution (i.e., the one with the lowest number of PCP instances) found so far and the path from the root node to the respective solution. At the end, the sequence of adaptations leading to the best found solution is carried out. PCP instances that could not be mitigated will be taken into account in the next adaptation cycle again, together with any new PCP instances that arise in the meantime.

Searching for the best adaptation sequence can be time-consuming. While RADAR is searching, an attacker could already exploit the PCP instances. Therefore, the search is only allowed for a limited time (10 seconds in [10]). RADAR tries to mitigate as many PCP instances as possible in the given time frame. However, RADAR is lacking the intelligence to know which PCP instances are more important or more urgent than others. The aim of the present paper is exactly to provide this intelligence.

3 PROPOSED APPROACH

The aim of our approach is to quantify the risk associated with data protection vulnerabilities in a system. A *data protection vulnerability* is a configuration of the system and its environment, in which an attacker may gain unauthorized access to some confidential data. Some examples for data protection vulnerabilities [16]:

- A database containing personal data is hosted in a virtual machine on a cloud platform, and the administrative interface

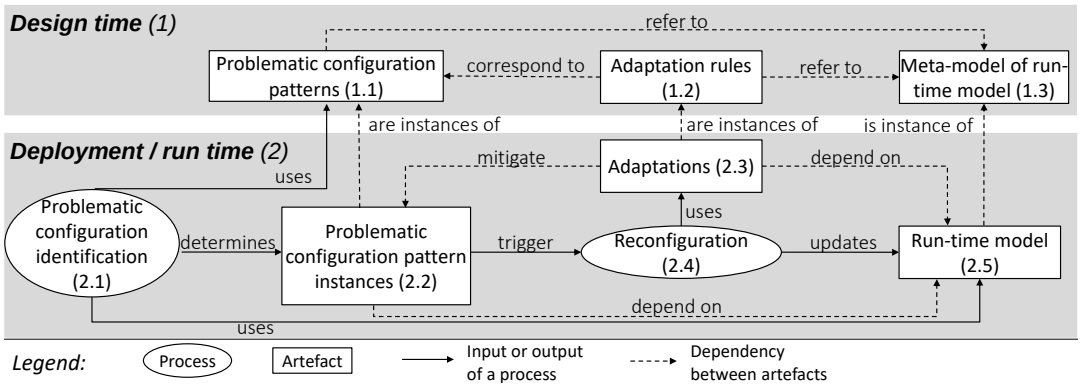


Figure 1: Overview of the RADAR approach. The numbers in the boxes are used in the text to refer to the boxes

of the cloud platform uses a protocol vulnerable to known attacks, potentially allowing an attacker to gain unauthorized access to the data in the database.

- A database containing personal data is hosted in a virtual machine on a cloud platform, and the settings of the scheduler used by the underlying infrastructure are vulnerable to side-channel attacks, potentially allowing an attacker to gain unauthorized access to the data in the database.
- A database containing personal data is accessible to an untrusted application, potentially allowing an attacker to gain unauthorized access to the data in the database.

In the following, we formalize our attack model and present a mathematical model to reason about potential attacks.

3.1 Attack model

A data protection vulnerability is a special kind of security vulnerability, with some typical properties. First, the damage caused by an attack depends on the type of data (e.g., in a hospital, patients' health data is more sensitive than inventory data) and on the amount of data (the more data is stolen, the higher the damage). Second, exploiting a data protection vulnerability takes time. This includes time to detect the existence of the vulnerability, time to prepare an attack, and time to steal each data item. The time for stealing the whole data set depends on the size of the data set. Third, once the attacker has stolen the data, the damage cannot be undone.

To account for these properties, Fig. 2 sketches our model of how the caused damage changes with time. The model can be seen as the impact of the cyber kill chain [29] on data protection. From the moment that a vulnerability appears, it takes the attacker some time to detect the vulnerability and then to prepare an exploit. The time to detect the vulnerability and to prepare the exploit may be very short if the vulnerability is easily detected / exploited, but it may also be long for more intricate vulnerabilities. These preparatory steps of the attacker do not cause damage. During the exploit, the damage increases linearly with time, as proportionately more and more data is stolen. When all data is stolen, the damage reaches its maximum, and it stays at that level.

Our attack model is primarily aimed at modeling attacks against the confidentiality of data. Other types of attacks, for example against the availability of data (e.g., ransomware attacks) could also follow a similar pattern, and could thus be amenable to similar reasoning, but we leave this for future research.

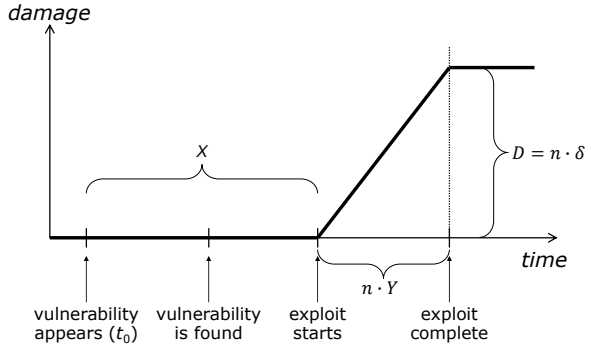


Figure 2: Damage caused by an attack

Our approach is not to wait until an attack actually happens. As soon as the vulnerability appears, the defender can start reasoning about possible attacks to exploit the vulnerability, making it possible to proactively mitigate the risk of such an attack. The defender can use the general model of Fig. 2, but without knowing the exact parameters (e.g., how long the attacker will need to detect the vulnerability). Rather, the defender can perform a probabilistic analysis, as described next, to reason about the risks.

3.2 Expected damage

Traditionally, risk is quantified in terms of probability and caused damage [30]. For a vulnerability v, let p_v denote the probability with which v is exploited, leading to a security breach. Let d_v denote the damage that the security breach would cause. The risk value of vulnerability v is calculated as $r_v = p_v \cdot d_v$. We observe that, in this traditional model of risk, r_v is exactly the *expected value of the damage*. This is because the damage is d_v with probability p_v and 0 with probability $1 - p_v$. Using this insight, we now devise a mathematical model to quantify data protection risks. The used notation is summarized in Table 1.

Let us consider a data protection vulnerability, i.e., a vulnerability that potentially allows an attacker to gain access to a confidential dataset containing n items. The maximum damage that an attacker

Table 1: Notation overview

Symbol	Description
v	A vulnerability
n	Number of data items in the dataset
δ	Damage per leaked data item
D	Maximum damage
t_0	Time when vulnerability appears
X	Time to detect and start exploiting a vulnerability
Y	Time to steal one data item
t	Time since a vulnerability appeared
$d_n(t)$	Damage caused after time t
r_v	Expected damage in the next T time
n_1, n_2	Nearest n values for which $\overline{d_n(t)}$ is known
α	Weight of n_1 in expressing n as affine combination
t_1, t_2	Nearest t values for which $\overline{d_n(t)}$ is known
β	Weight of t_1 in expressing t as affine combination
$f_{X,Y}$	Joint probability distribution function of (X, Y)
$n_{\min}$	Lower limit of n in preprocessing
$n_{\max}$	Upper limit of n in preprocessing
Δn	Increment of n in preprocessing
T	Look-ahead time horizon
$t_{\max}$	Upper limit of t in preprocessing
Δt	Increment of t in preprocessing
M	Number of samples for estimating expected value

can cause is $D = n \cdot \delta$, corresponding to the case when the attacker manages to steal the whole dataset. Here, $\delta > 0$ is a given constant, corresponding to the damage caused by stealing one data item, and thus defining how valuable the data in the given dataset is.

Let t_0 denote the point in time when the vulnerability appears. Let the random variable X denote the time it takes an attacker to start exploiting the vulnerability. (In Fig. 2, X is the total time for noticing the vulnerability and preparing the exploit.) Let the random variable Y denote the time it takes the attacker, after the exploit started, to steal one data item. Let t denote the time that elapsed since t_0. The damage caused by the attacker during this time is, as can be seen from Fig. 2:

$$d_n(t) = \begin{cases} 0 & \text{if } t < X \\ \frac{t-X}{Y} \cdot \delta & \text{if } X \le t < X + n \cdot Y \\ D & \text{if } t \ge X + n \cdot Y \end{cases} \quad (1)$$

Note that, since X and Y are random variables, so is $d_n(t)$. The risk value associated with the given vulnerability v is computed as the expected additional damage in the next period of time of length T:

$$r_v = \mathbb{E}[d_n(t+T) - d_n(t)] = \mathbb{E}[d_n(t+T)] - \mathbb{E}[d_n(t)]. \quad (2)$$

By focusing on the expected damage in the next period of time, we obtain the right metric for risk prioritization. In particular, we disregard the already incurred damage, which is a sunken cost, and thus should not influence the decision-making. We also disregard damage that will be incurred in the far future because that damage can be avoided regardless of how current risks are prioritized.

Algorithm 1: Preprocessing

```
1 for t ← 0 to t_max by Δt do
2     for n ← n_min to n_max by Δn do
3         S ← 0
4         for M times do
5             pick random X, Y values
6             calculate d_n(t) using (1)
7             S ← S + d_n(t)
8         d_n(t)‾ ← S/M
```

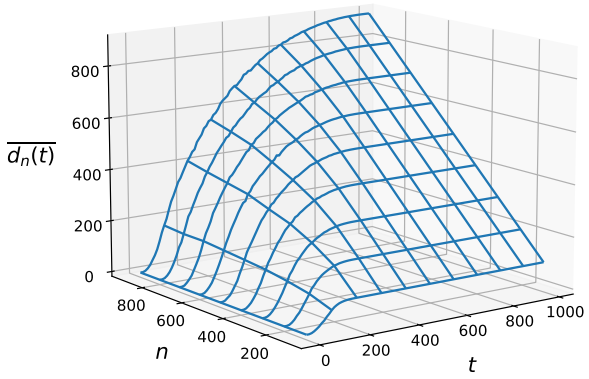


Figure 3: Example for the $\overline{d_n(t)}$ function

3.3 Numerical risk computation

Depending on the joint distribution of X and Y, calculating $\mathbb{E}[d_n(t)]$ can be difficult. To overcome this difficulty, we propose a numerical approximation, using the average from a sufficiently large random sample as an estimate of the expected value.

We approximate $\mathbb{E}[d_n(t)]$ using Monte Carlo sampling for different n and t values in a preprocessing step, as shown in Algorithm 1. For each considered pair of t (line 1) and n (line 2), M random values for X and Y are sampled using the given distribution of (X, Y) (lines 4-5). For these values of X and Y, $d_n(t)$ is calculated according to (1) (line 6). The average of the M values for $d_n(t)$ is stored in $\overline{d_n(t)}$, as an estimate of $\mathbb{E}[d_n(t)]$ (lines 3, 7, 8).

Fig. 3 shows an example. Here, X and Y are independent, $X \sim U(10, 100)$, and $Y \sim U(0, 1)$. Further, $\delta = 1$, $t_{\max} = 1000$, $\Delta t = 10$, $n_{\min} = 100$, $n_{\max} = 1000$, $\Delta n = 100$, and $M = 5000$.

After this preprocessing step, $\mathbb{E}[d_n(t)]$ can be estimated for given values of n and t as follows. The two nearest values of n for which $\overline{d_n(t)}$ has been pre-calculated are:

$$n_1 = \begin{cases} n_{\min} & \text{if } n \le n_{\min} \\ n_{\max} - \Delta n & \text{if } n \ge n_{\max} \\ n_{\min} + \left\lfloor \frac{n - n_{\min}}{\Delta n} \right\rfloor \Delta n & \text{otherwise} \end{cases} \quad (3)$$

$$n_2 = n_1 + \Delta n \quad (4)$$

Algorithm 2: Quantifying a risk

1 Calculate n_1 and n_2 using (3)-(4)
2 Calculate α using (7)
3 Calculate t_1 and t_2 using (5)-(6)
4 Calculate β using (8)
5 Approximate $\mathbb{E}[d_n(t)]$ using (9)
6 Repeat steps 3-5 for $t+T$ instead of t to get $\mathbb{E}[d_n(t+T)]$
7 Calculate r_v using (2)

Similarly, the two nearest values of t for which $\overline{d_n(t)}$ has been pre-calculated are:

$$t_1 = \begin{cases} 0 & \text{if } t \leq 0 \\ t_{\max} - \Delta t & \text{if } t \geq t_{\max} \\ \left\lfloor \frac{t}{\Delta t} \right\rfloor \Delta t & \text{otherwise} \end{cases} \tag{5}$$

$$t_2 = t_1 + \Delta t \tag{6}$$

In addition, we choose

$$\alpha = \frac{n_2 - n}{n_2 - n_1} \tag{7}$$

$$\beta = \frac{t_2 - t}{t_2 - t_1} \tag{8}$$

Since $n_2 > n_1$ and $t_2 > t_1$, α and β are well-defined, and we have $n = \alpha \cdot n_1 + (1-\alpha) \cdot n_2$ and $t = \beta \cdot t_1 + (1-\beta) \cdot t_2$. Using α and β, the following estimate can be computed:

$$\begin{aligned} \mathbb{E}[d_n(t)] \approx \quad & \alpha \quad \cdot \left(\beta \cdot \overline{d_{n_1}(t_1)} + (1-\beta) \cdot \overline{d_{n_1}(t_2)}\right) \\ & +(1-\alpha) \cdot \left(\beta \cdot \overline{d_{n_2}(t_1)} + (1-\beta) \cdot \overline{d_{n_2}(t_2)}\right) \end{aligned} \tag{9}$$

Algorithm 2 summarizes how the risk value r_v can be computed for a vulnerability v. This procedure is carried out for each currently existing vulnerability. Finally, vulnerabilities with larger r_v values are prioritized for mitigation.

A data protection vulnerability is characterized by a vulnerability type and an involved dataset. The vulnerability type determines how quickly an attacker might gain access to the data, i.e., the distribution of X and Y. The involved dataset determines the value of the data, i.e., δ. These pieces of information (distribution of X and Y, as well as δ) are determined at design time by the involved experts. On this basis, the $\overline{d_n(t)}$ functions can be computed for all potential vulnerability types and datasets at design time using Algorithm 1. At run time, the values of n and t become available[1]. Thus, the specific values of $\overline{d_n(t)}$ can be computed, and risks can be quantified using Algorithm 2. Note that run-time changes of n or t may lead to a new risk value for the same vulnerability.

4 EMBEDDING INTO RADAR

Our approach presented in the previous section is a generic technique for assessing data protection vulnerabilities in a system and for deciding which ones to mitigate first to reduce the caused damage. The mathematical model in combination with the presented algorithms can be applied in different systems for data protection.

[1]At design time, Algorithm 1 only needs an estimate of the range from which n and t can take values (t_{max}, n_{min}, n_{max})

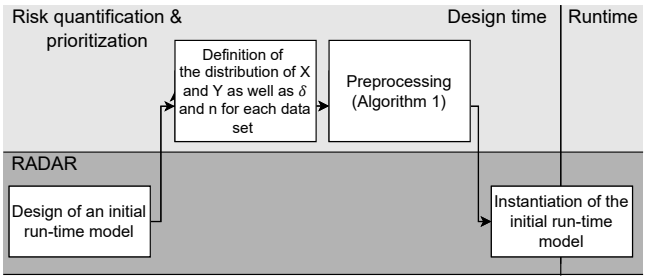


Figure 4: Design-time interaction between RADAR and our approach

To evaluate our approach, we integrated it into RADAR, enabling RADAR to prioritize data protection vulnerabilities based on the associated risks. Yet, the approach remains independent of RADAR.

In RADAR, a data protection vulnerability is a PCP instance that affects a "StoredDataSet" object of the run-time model (see also Section 2). We extend RADAR's meta-model with new attributes in the class "StoredDataSet" to store the values of n and δ. At design time, the activities in Figure 4 are executed. After the design of an initial run-time model, the values of δ and n for each "StoredDataSet" object are defined. (The value of n can change at run-time.) Based on these values, preprocessing according to Algorithm 1 is executed and the resulting values are stored for later use. At the transition from design-time to the run-time phase the run-time model is instantiated.

At run-time, an adaptation cycle is started if PCP instances are detected. Whenever a PCP instance is found, its risk value is calculated using Algorithm 2 and the tracked parameters n, δ, and t, based on preprocessing (see Fig. 5).

In the Adaptation Planing Algorithm, RADAR searches for the best adaptation, i.e., the adaptation that leads to the best system configuration according to a given ordering relation. By default, RADAR prefers system configurations with fewer vulnerabilities. Let R and R' be two possible system configurations, where R contains vulnerabilities $v_1, v_2, \ldots, v_N$, and R' contains vulnerabilities $v'_1, v'_2, \ldots, v'_{N'}$. RADAR considers R better than R' if and only if $N < N'$. We call this ordering *no prioritization*, or NoPrio for short, since all vulnerabilities are considered equal. The risk values allow us to introduce two alternative risk-based prioritization methods.

The idea of *local prioritization* (LocalPrio) is to eliminate the vulnerabilities with the highest risk value first, and then continuing in descending order of risk value. Assume that $r(v_1) \geq r(v_2) \geq \ldots \geq r(v_N)$ and $r(v'_1) \geq r(v'_2) \geq \ldots \geq r(v'_{N'})$. If there is an index $1 \leq j \leq \min(N, N')$ such that $r(v_j) \neq r(v'_j)$, then let j be the smallest such index. R is better than R' if and only if $r(v_j) < r(v'_j)$. If there is no such index, then the NoPrio ordering decides.

The idea of *global prioritization* (GlobalPrio) is to minimize the sum of the risk values of the vulnerabilities in the system configuration. R is better than R' if and only if $\sum_{i=1}^{N} r(v_i) < \sum_{i=1}^{N'} r(v'_i)$.

Finally, the best adaptation resulting from the Adaptation Planning Algorithm is executed and the next adaptation cycle starts.

We extended RADAR to keep track of vulnerabilities (see vulnerability lifelines in Fig. 5). During an adaptation cycle, RADAR

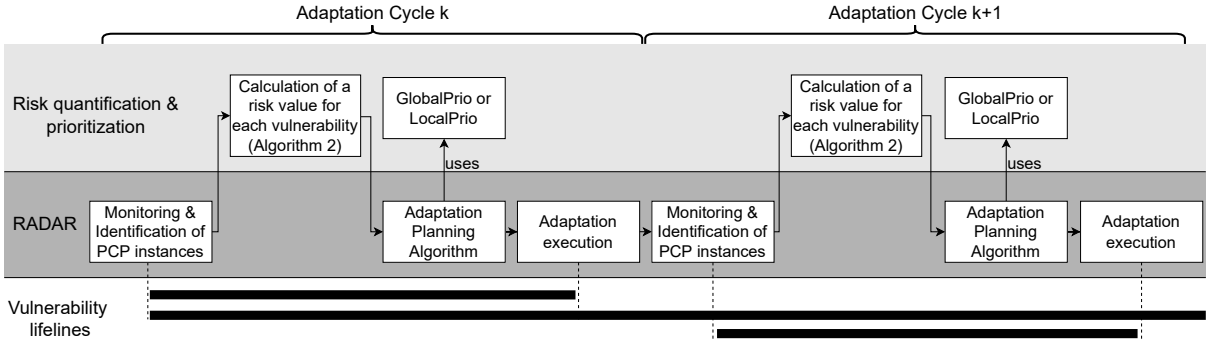


Figure 5: Run-time interaction between RADAR and our approach

compares the detected vulnerabilities to the list of vulnerabilities found in prior cycles. An unknown vulnerability is then added to the list. An initial risk value is calculated when RADAR first detects a vulnerability. If RADAR is unable to mitigate a vulnerability during an adaptation cycle, the vulnerability is found in the next cycle again, and its risk value is updated. Since t has increased for the given vulnerability, its new risk value will be usually different.

In the new, extended version of RADAR, the search algorithm can be run with any of the three prioritization methods (NoPrio, LocalPrio, GlobalPrio). Our implementation is publicly available[2].

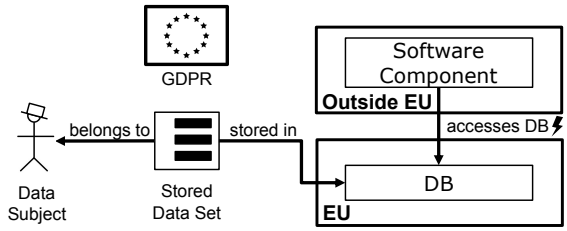


Figure 6: Example data protection vulnerability

5 EVALUATION

5.1 Experimental setup

We experimentally evaluate the integration of our method into RADAR and compare the three prioritization methods. For this, we use a run-time model comprising 337 nodes, including 16 "StoredDataSet" objects. This initial run-time model was based on a project partner's real cloud system, which was multiplied 21 times around a central "DataSubject" object [10]. This initial run-time model does not contain any vulnerabilities.

In a cloud system, a database containing personal data of EU citizens may be accessed by a software component, which processes the data. Due to a resource allocation decision of the cloud provider, the software component may be migrated from a server in the EU to a server outside the EU. This may lead to the processing of data of EU citizens outside the EU which is in general not allowed by the GDPR [7]. This scenario is modeled by the vulnerability type "OutsideEU" [10], shown in Fig. 6. This example shows that our approach can also be applied to data protection vulnerabilities beyond mere confidentiality (in this case, geolocation constraints). Here, too, our attack model can be applied: the longer the software component is outside the EU, the higher the risk that an attacker can exploit it to get personal data outside the EU. This vulnerability can be mitigated, for example, by moving the software component back to a server in the EU or by stopping the data access. We inject 30 vulnerabilities of this type into the run-time model by local modifications (changing attributes, removing or adding relations between nodes). We chose the number of vulnerabilities such that RADAR cannot mitigate them all in one adaptation cycle. Note that our approach is independent of the details of the PCPs, so we can expect that other data protection vulnerabilities (like the ones presented in [16]) would lead to similar results.

In some experiments, all vulnerabilities are injected at once. In others, the injection of vulnerabilities occurs in two phases to simulate the coexistence of older and newer vulnerabilities. First, some vulnerabilities are injected into the initial run-time model and detected by RADAR, but not yet mitigated. These simulate *older vulnerabilities*, unmitigated from a previous adaptation cycle. The remaining vulnerabilities are injected after a pause of $\tau = 10$ sec, in the next adaptation cycle. We call this group *newer vulnerabilities*.

After these injections, the run-time model is in a vulnerable state, and RADAR is applied to it for one adaptation cycle. If a vulnerability is mitigated in this adaptation cycle, we calculate the damage the vulnerability caused and store this value. Vulnerabilities that are not mitigated in this adaptation cycle are considered to be mitigated in the next adaptation cycle, i.e., after τ time.

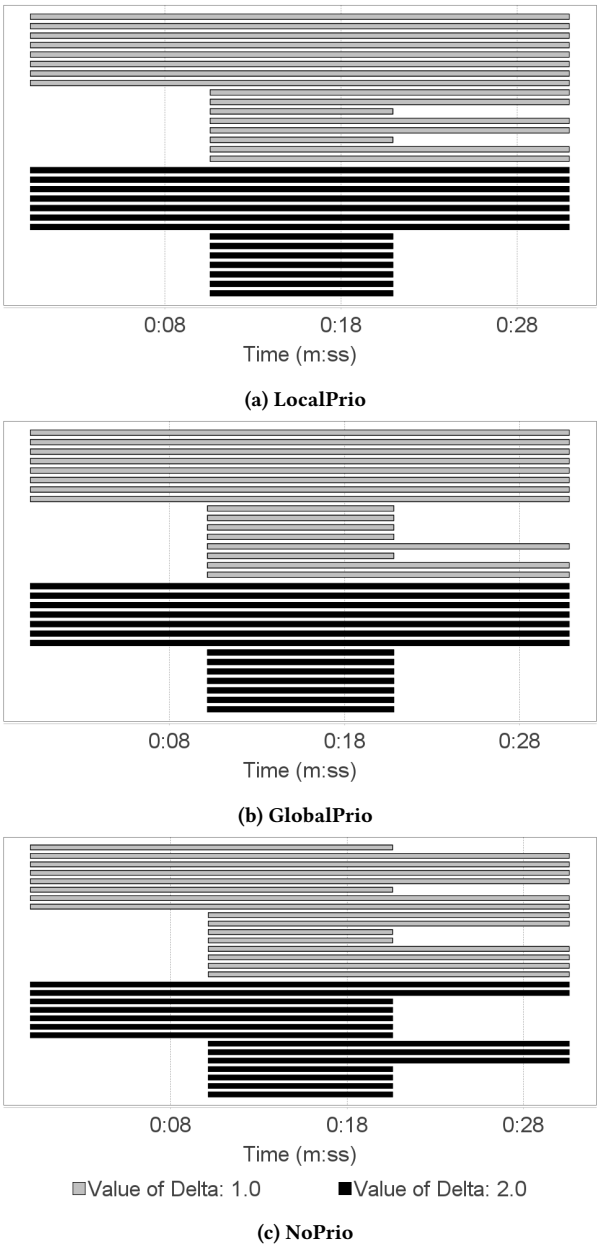
The experiments were run on a Fujitsu Celsius w550 workstation with Intel Xeon E3-1275v5 3.6GHz CPU and 64GB DDR4 memory. The machine was running Ubuntu 16.04 and Open-JDK 64-Bit Server VM (build 11.0.2+9) as Java Virtual Machine.

For Algorithm 1, the following values were used: $n_{min} = \Delta n = 10,000$, $n_{max} = 100,000$, $t_{max} = 100$, $\Delta t = 1$, $M = 5,000$, $X \sim U(1, 10)$, $Y \sim U(0, 0.001)$, and X and Y are independent.

5.2 An exemplary scenario

In a first experiment, we show how RADAR prioritizes the mitigation of different vulnerabilities. We set $\delta = 1.0$ for one half of the datasets in the run-time model, and $\delta = 2.0$ for the other half.

[2] https://git.uni-due.de/spsazmie/radar_prio



(a) LocalPrio

(b) GlobalPrio

(c) NoPrio

Figure 7: Lifelines of vulnerabilities with different prioritization methods

The assignment of the value to datasets is done randomly. We set $n = 20,000$ for all datasets. We then apply RADAR once with each prioritization method. For each vulnerability, we record its δ, the moment it is detected, and the moment it is mitigated.

The results are visualized in Fig. 7. Each plot displays a different prioritization method. The horizontal axis displays the amount of time the vulnerability existed. The bars starting further left belong to older vulnerabilities, while those starting further right belong to newer vulnerabilities. The vulnerabilities are grouped in the diagrams according to the values of δ and t_0.

Both LocalPrio (Fig. 7a) and GlobalPrio (Fig. 7b) lead to a quick mitigation of all newer vulnerabilities with $\delta = 2.0$. Obviously, vulnerabilities with $\delta = 2.0$ have a higher risk value and are thus prioritized over vulnerabilities with $\delta = 1.0$. Newer vulnerabilities are prioritized over older ones. The older vulnerabilities are already almost completely exploited, which results in lower potential damage for the future and thus in a lower risk value than the newer ones, which are associated with higher potential future damage[3].

Both LocalPrio and GlobalPrio mitigate some newer vulnerabilities with $\delta = 1.0$. Here, GlobalPrio is able to mitigate more vulnerabilities than LocalPrio.

NoPrio (Fig. 7c) shows no pattern when mitigating the vulnerabilities, as both newer and older vulnerabilities, as well as those with higher and lower value of δ are equally selected for mitigation.

5.3 Statistical analysis

In the following experiments, we compare the amount of damage for different methods of prioritization. We use the setup of the previous experiment, but increase the higher δ to 100.0. We repeat the experiment 100 times to convey more information about the distribution of results.

Fig. 8 compares the results for the different prioritization methods in a boxplot. We group the vulnerabilities by δ and report the total damage for these groups. In addition, the total damage for all vulnerabilities is also shown. The jagged line between 5,000,000 and 20,000,000 shows a break in the values of the vertical axis, as no damage values in this range were recorded and thus this range was cut out to increase readability.

For vulnerabilities with $\delta = 100.0$, both GlobalPrio and LocalPrio lower the damage significantly compared to NoPrio. For vulnerabilities with $\delta = 1.0$, it is not visible from the figure, but NoPrio leads to lower damage than GlobalPrio and LocalPrio. However, the total damage for all vulnerabilities is hardly influenced by the vulnerabilities with $\delta = 1.0$. The bottom line is that GlobalPrio and LocalPrio lower the total damage significantly compared to NoPrio. This is a direct consequence of prioritization: GlobalPrio and LocalPrio focus on mitigating only vulnerabilities with $\delta = 100.0$, while NoPrio mitigates vulnerabilities with $\delta = 100.0$ and $\delta = 1.0$ alike.

There is no clear difference between the distributions of results for LocalPrio and GlobalPrio. In general, the comparison criterion for LocalPrio is more restricting than for GlobalPrio, since LocalPrio requires to always mitigate the vulnerability with the highest risk value first, whereas GlobalPrio allows a temporary relaxation of this rule. This can lead to a larger search space for GlobalPrio, which may be both advantageous (it is possible to find a better solution) and disadvantageous (in a given amount of time, only a smaller fraction of the search space can be searched). These two opposing effects seem to be balanced.

[3]This holds true for the given values of the parameters. For other values, it is also possible that older vulnerabilities are to be prioritized over newer ones. For example, if the newer vulnerabilities are not yet expected to lead to damage in the next time period, while older vulnerabilities already start to lead to damage, then older vulnerabilities are prioritized.

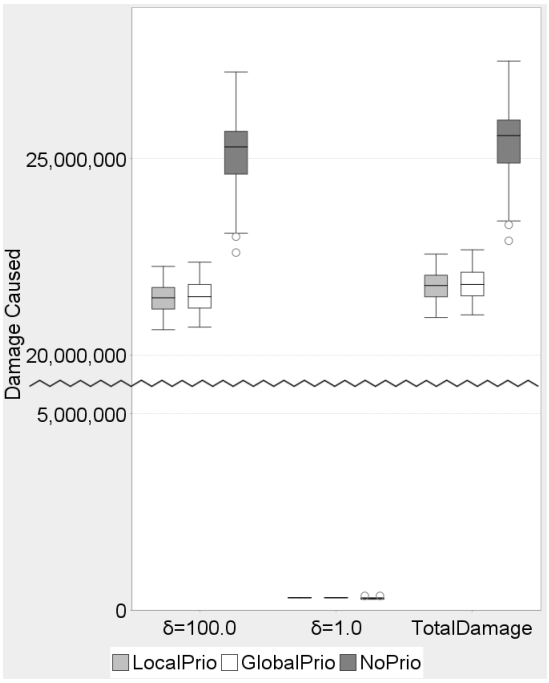


Figure 8: Results for 100 runs of the experiment

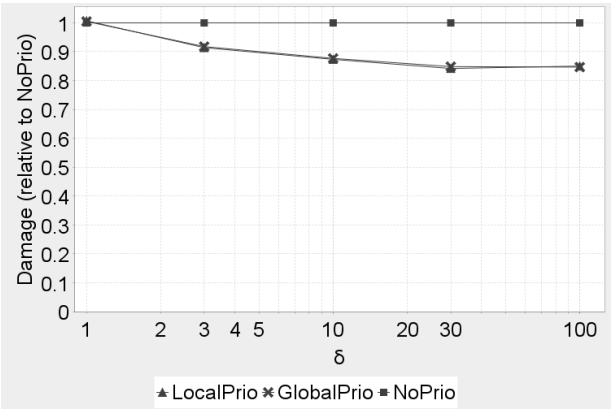


Figure 9: Effect of different values for δ_b

5.4 Impact of δ

To evaluate the impact of δ on the results, we fix the other parameters. All datasets have n = 100,000, and all vulnerabilities are injected at the same time. Half of the datasets are assigned $\delta_s = 1.0$. The other datasets are assigned δ_b, for which the values 1, 3, 10, 30, 100 are used. For each value of δ_b and each prioritization method, the experiment is repeated 100 times and the median of the damage is reported in Fig. 9. The results are normalized such that the damage for NoPrio is 1. The horizontal axis uses logarithmic scale.

For $\delta_b = \delta_s$, the risk value of every vulnerability is equal. Thus, the prioritization methods do not lead to notably different results.

For $\delta_b \neq \delta_s$, prioritization leads to damage reduction. In this case, different risk values occur and LocalPrio and GlobalPrio mitigate vulnerabilities with a high risk value first. Consequently, the damage is lower than with NoPrio. As δ_b increases, the damage reduction increases as well. This is because higher differences in δ make it increasingly important to prioritize vulnerabilities with a high risk value, leading to a growing advantage for LocalPrio and GlobalPrio over NoPrio. This advantage cannot grow beyond some limit, since not all vulnerabilities with δ_b can be mitigated and even the ones that are mitigated lead to some damage before mitigation. Prioritization reduces the maximum damage up to 15.8%.

The results of LocalPrio and GlobalPrio are again nearly identical.

5.5 Impact of dataset size

In the next experiment, we set the size of half of the datasets to n_s = 10,000 and the other half to n_b. We vary n_b from 10,000 to 100,000 by steps of 10,000. We fix all other parameters uniformly: all datasets have $\delta = 1.0$, and all vulnerabilities are injected simultaneously.

For each value of n_b and each prioritization method, we perform the experiment 100 times. The median of the resulting damage is reported in Fig. 10.

If $n_b = n_s$, the risk values of all vulnerabilities are equal, and thus prioritization has no effect. As a result, all three prioritization methods lead to the same amount of damage. In our setup, this happens at n_b = 10,000. When $n_b \neq n_s$, the risk values of vulnerabilities affecting datasets of different size are different, making prioritization useful. Indeed, both LocalPrio and GlobalPrio lead to better results than NoPrio. This difference in the results increases with growing n_b. This is because for relatively small values of n_b, attackers can likely fully steal the datasets before the vulnerabilities can be mitigated. However, as n_b grows, it takes attackers longer to fully steal the datasets, giving more possibilities to mitigation, and thus making prioritization more important.

The damage increases for all three methods with an increasing value of n_b, since the datasets become more valuable. However, the gradient of all three curves is decreasing. This is because of our assumption that each vulnerability unmitigated in the current adaptation cycle will be mitigated in the next adaptation cycle. Thus, the probability that attackers can steal more data is limited even if more data is available.

5.6 Impact of older vs. newer vulnerabilities

In the last experiment, we investigate the impact of the time for which vulnerabilities exist. From the 30 vulnerabilities injected into the run-time model, there are $\lfloor q \cdot 30 \rfloor$ older vulnerabilities and $\lceil (1-q) \cdot 30 \rceil$ newer vulnerabilities for some $0 \leq q \leq 1$. We test different values for q: 0, 0.25, 0.50, 0.75, 1. The other parameters are set uniformly: we set $\delta = 1.0$ and n = 20,000 for all datasets.

For each value of q and each prioritization method, we perform the experiment 100 times. The median of the resulting damage is reported in Fig. 11. For $q = 0$ and $q = 1$, all risk values are equal. Thus, prioritization has no effect. But for $0 < q < 1$, there are older and newer vulnerabilities, which have different risk values, making prioritization useful. For the given parameters, older vulnerabilities have lower risk value than newer vulnerabilities, because older vulnerabilities are already partially exploited before the mitigation can

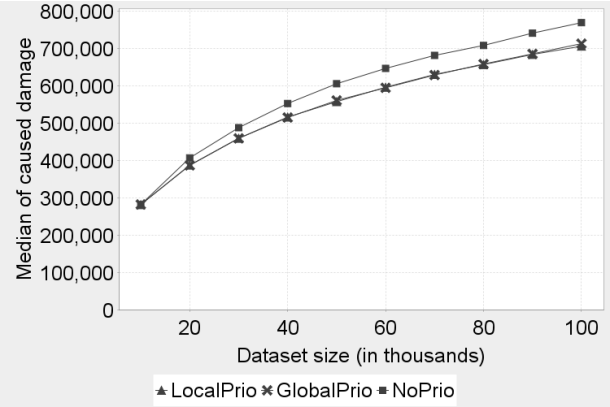


Figure 10: Effect of the value of n_b

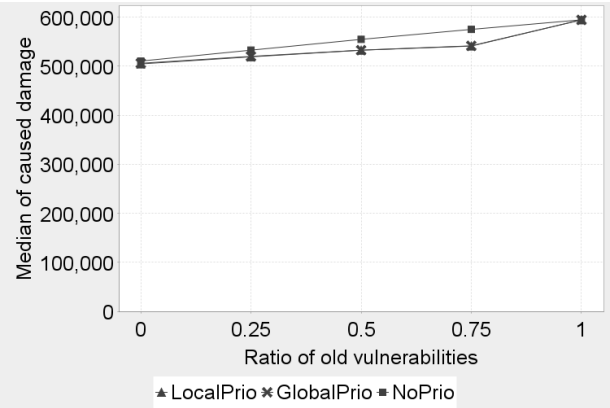


Figure 11: Effect of the value of q

start, thus leaving less room for damage reduction. Consequently, both LocalPrio and GlobalPrio mitigate newer vulnerabilities first, which results in lower damage compared to NoPrio.

As the group of older vulnerabilities, which are already partially exploited, gets larger, the damage increases. NoPrio does not differentiate between older and newer vulnerabilities, and thus chooses less newer vulnerabilities for mitigation. On the other hand, LocalPrio and GlobalPrio give preference to newer vulnerabilities, thus leading to a smaller increase in damage. Therefore, the advantage of LocalPrio and GlobalPrio over NoPrio increases.

6 RELATED WORK

In information security, risk assessment plays an important role. [20] presents a taxonomy of such approaches, consisting of multiple dimensions. However, specifics of data protection and automated online risk assessment are not considered.

The GDPR stipulates a risk-based approach to data protection [6]. Accordingly, there has been increased interest in data protection risk assessment in recent years. For example, [2] proposes a consolidated approach for data protection impact assessment. [27] analyzes how the notion of "risks to the rights and freedoms of data subjects" influences the appropriateness of existing risk assessment processes. [12] investigates possibilities to include data protection risk management in software and systems engineering processes. Along similar lines, [24] proposes to capture risk management and related data protection concerns in the system development process using a dedicated architectural view. [5] proposes a semi-formal model-based approach to data protection risk management. [9] investigates the appropriateness of different modeling languages for modeling data protection concerns. These papers aim at supporting human experts in manual risk management processes at design time. They offer no direct support for automated risk assessment at run time. Hence, they cannot be used for quick prioritization when searching for the best reaction to vulnerabilities at run time.

Automatic risk assessment has been studied in the context of intrusion detection and intrusion protection. [21] identifies three groups of approaches to intrusion risk assessment: attack graph based, service dependency graph based, and non graph based. This categorization is limited to the assessment of potential responses to an intrusion. [14] suggests an approach for an online assessment of IT security risks stemming from an intrusion into the system. As input, information from intrusion detection systems is used, including alert severity and alert confidence. Dempster-Shafer theory is applied to combine these data to an overall risk estimate. Compared to our approach, there are several key differences: (1) the specific characteristics of data protection are not taken into account; (2) risk is calculated for an intrusion, not for a vulnerability; (3) not the damage is estimated directly, but only an artificial score as a proxy. [22] proposes an approach for online risk assessment in the context of a cyberattack response system. Similarly to our approach, [22] also uses a combination of offline and online computations. However, [22] estimates risks based on statistical data from intrusion detection systems, such as the frequency of alerts, and uses fuzzy logic for quantifying the risk. In contrast, we use information more relevant for data protection risks, such as the size of the dataset, and employ probability theory to calculate the risk values.

Automatic risk assessment has also been considered in the context of access control. For example, [13] uses machine learning to predict access control decisions and assesses the risk of a wrong decision to decide whether to trust the prediction or to contact the policy decision point. [17] uses attribute similarity to perform attribute-based access control in cases where the exact attribute values are not known, and takes into account the risk of a wrong positive versus negative decision. These works address a different type of risk, stemming from the system's decision-making and not from an attacker, resulting in different risk models from ours.

Few papers address automated risk management in a more general setting. [15] proposes an Artificial Immune System algorithm for selecting a set of countermeasures to given threats. A risk-based method is used to quantify the effects of potential countermeasures on the threats. Risk level is calculated based on several factors, such as the probability of occurrence of a threat and the negative impact that the threat may cause. These factors are combined to a risk level through an arbitrary formula. In contrast, we provide a probabilistic model from which we derive our way of estimating risks. In addition, [15] does not consider how risks vary with time. [23] proposes an approach for calculating the potential damage that an attack may cause. Attack graphs are used to assess how far an ongoing attack has reached towards the targeted asset, and service

dependency graphs to determine the potential impact of the attack. However, it is not taken into account that in a data breach, stealing each further data item may take additional time.

Some approaches to quantify privacy are also related to our work. As shown in [28], some privacy metrics are specific to information leakage ("Amount of Leaked Information", "Maximum Information Leakage"), but do not account for the sensitivity of the data. Other metrics are time-related ("Time until Adversary's Sucess"), but do not account for how damage increases over time. Our approach combines characteristics of multiple existing metrics to reflect the expected damage from a data breach, considering the amount and sensitivity of the data, as well as the time it takes to exploit the vulnerability and the time since the vulnerability arose.

Previous work leading to the development of RADAR focused on capturing data protection vulnerabilities as PCPs [19], defining the necessary concepts for modeling data protection [11], and the efficient implementation using pattern matching [8]. In addition, the approach of modeling data protection vulnerabilities was validated using a varied set of real-world vulnerabilities [16]. None of these papers addressed the problem of quantifying data protection risks.

7 CONCLUSIONS AND FUTURE WORK

This paper presented a new approach for quantifying data protection risks. Our approach considers specific aspects of data protection risks, including the typical temporal development of the damage caused by a data breach. Using a probabilistic model, we calculate the risk as the expected value of the future damage. We also showed how our approach can be incorporated into RADAR, a state-of-the-art solution for automated mitigation of data protection risks. Our approach allows the prioritization of risks, resulting in up to 15.8% reduction in damage.

Future research could aim at automating those parts of our approach that are currently done manually. For example, an analysis of known vulnerabilities, possibly combined with machine learning, could be used to estimate the distribution of the random variables X and Y. Graph-based approaches could be used to derive PCPs (semi-)automatically. Moreover, the effects of different probability distributions for X and Y could be analysed. The attack model could be extended by taking, e.g., the effectiveness and the costs of implementing an adaptation into account.

Acknowledgement. This work was partially supported by the European Union's Horizon 2020 research and innovation programme under grant 871525 (FogProtect).

REFERENCES

[1] Mark Andrejevic. 2014. The big data divide. *International Journal of Communication* 8 (2014), 1673–1689.
[2] Felix Bieker, Michael Friedewald, Marit Hansen, Hannah Obersteller, and Martin Rost. 2016. A Process for Data Protection Impact Assessment under the European General Data Protection Regulation. In *Annual Privacy Forum*. Springer, 21–37.
[3] Jakub Breier and Ladislav Hudec. 2011. Risk analysis supported by information security metrics. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*. 393–398.
[4] Valentina Casola, Alessandra De Benedictis, Massimiliano Rak, and Umberto Villano. 2020. A novel Security-by-Design methodology: Modeling and assessing security by SLAs with a quantitative approach. *Journal of Systems and Software* 163 (2020), 110537.
[5] Shi-Cho Cha and Kuo-Hui Yeh. 2018. A data-driven security risk assessment scheme for personal data protection. *IEEE Access* 6 (2018), 50510–50517.
[6] Raphael Gellert. 2018. Understanding the notion of risk in the General Data Protection Regulation. *Computer Law & Security Review* 34, 2 (2018), 279–288.
[7] General Data Protection Regulation. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC. *Official Journal of the European Union* (2016), L119.
[8] Florian Kunz and Zoltán Ádám Mann. 2019. Finding risk patterns in cloud system models. In *IEEE 12th International Conference on Cloud Computing (CLOUD)*. 251–255.
[9] Jan Laufer, Zoltán Ádám Mann, and Andreas Metzger. 2021. Modelling Data Protection in Fog Computing Systems using UMLsec and SysML-Sec. In *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 777–786.
[10] Zoltán Ádám Mann, Florian Kunz, Jan Laufer, Julian Bellendorf, Andreas Metzger, and Klaus Pohl. 2021. RADAR: Data Protection in Cloud-Based Computer Systems at Run Time. *IEEE Access* 9 (2021), 70816–70842.
[11] Zoltán Ádám Mann, Andreas Metzger, and Stefan Schoenen. 2018. Towards a run-time model for data protection in the cloud. *Modellierung 2018* (2018), 71–86.
[12] Yod-Samuel Martin and Antonio Kung. 2018. Methods and tools for GDPR compliance through privacy and data protection engineering. In *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 108–111.
[13] Ian Molloy, Luke Dickens, Charles Morisset, Pau-Chen Cheng, Jorge Lobo, and Alessandra Russo. 2012. Risk-based security decisions under uncertainty. In *2nd ACM Conference on Data and Application Security and Privacy*. 157–168.
[14] C.P. Mu, X.J. Li, H.K. Huang, and S.F. Tian. 2008. Online risk assessment of intrusion scenarios using D-S evidence theory. In *European Symposium on Research in Computer Security (ESORICS)*. Springer, 35–48.
[15] Pantaleone Nespoli, Félix Gómez Mármol, and Jorge Maestre Vidal. 2021. A Bio-Inspired Reaction Against Cyberattacks: AIS-Powered Optimal Countermeasures Selection. *IEEE Access* 9 (2021), 60971–60996.
[16] Alexander Palm, Zoltán Ádám Mann, and Andreas Metzger. 2018. Modeling data protection vulnerabilities of cloud systems using risk patterns. In *International Conference on System Analysis and Modeling*. Springer, 1–19.
[17] Sowmya Ravidas, Indrakshi Ray, and Nicola Zannone. 2020. Handling incomplete information in policy evaluation using attribute similarity. In *2nd IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 79–88.
[18] Ira Rubinstein. 2013. Big data: the end of privacy or a new beginning? *International Data Privacy Law* (2013).
[19] Stefan Schoenen, Zoltán Ádám Mann, and Andreas Metzger. 2018. Using risk patterns to identify violations of data protection policies in cloud systems. In *Service-Oriented Computing – ICSOC 2017 Workshops*. Springer, 296–307.
[20] Alireza Shameli-Sendi, Rouzbeh Aghababaei-Barzegar, and Mohamed Cheriet. 2016. Taxonomy of information security risk assessment (ISRA). *Computers & Security* 57 (2016), 14–30.
[21] Alireza Shameli-Sendi, Mohamed Cheriet, and Abdelwahab Hamou-Lhadj. 2014. Taxonomy of intrusion risk assessment and response system. *Computers & Security* 45 (2014), 1–16.
[22] Alireza Shameli-Sendi and Michel Dagenais. 2014. ARITO: Cyber-attack response system using accurate risk impact tolerance. *International Journal of Information Security* 13, 4 (2014), 367–390.
[23] Alireza Shameli-Sendi, Michel Dagenais, and Lingyu Wang. 2018. Realtime intrusion risk assessment model based on attack and service dependency graphs. *Computer Communications* 116 (2018), 253–272.
[24] Laurens Sion, Pierre Dewitte, Dimitri Van Landuyt, Kim Wuyts, Ivo Emanuilov, Peggy Valcke, and Wouter Joosen. 2019. An architectural view for data protection by design. In *IEEE International Conference on Software Architecture (ICSA)*. 11–20.
[25] Steve Taylor, Mike Surridge, and Brian Pickering. 2021. Regulatory Compliance Modelling Using Risk Management Techniques. In *IEEE World AI IoT Congress (AIIoT)*. 0474–0481.
[26] Aggeliki Tsohou, Emmanouil Magkos, Haralambos Mouratidis, George Chrysoloras, Luca Piras, Michalis Pavlidis, Julien Debussche, Marco Rotoloni, and Beatriz Gallego-Nicasio Crespo. 2020. Privacy, security, legal and technology acceptance elicited and consolidated requirements for a GDPR compliance platform. *Information and Computer Security* 28, 4 (2020), 531–553.
[27] Niels Van Dijk, Raphaël Gellert, and Kjetil Rommetveit. 2016. A risk to a right? Beyond data protection risk assessments. *Computer Law & Security Review* 32, 2 (2016), 286–306.
[28] Isabel Wagner and David Eckhoff. 2019. Technical privacy metrics: a systematic survey. *Comput. Surveys* 51, 3 (2019), art. 57.
[29] Tarun Yadav and Arvind Mallari Rao. 2015. Technical aspects of cyber kill chain. In *International Symposium on Security in Computing and Communication*. Springer, 438–452.
[30] Qi Zhang, Chunjie Zhou, Naixue Xiong, Yuanqing Qin, Xuan Li, and Shuang Huang. 2016. Multimodel-based incident prediction and risk assessment in dynamic cybersecurity protection for industrial control systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46, 10 (2016), 1429–1444.