A Method for Developing Qualitative Security Risk Assessment Algorithms

Gencer Erdogan and Atle Refsdal

Citation:
Erdogan G., Refsdal A. (2018) A Method for Developing Qualitative Security Risk Assessment Algorithms. In: Cuppens N., Cuppens F., Lanet JL., Legay A., Garcia-Alfaro J. (eds) Risks and Security of Internet and Systems. CRiSIS 2017. Lecture Notes in Computer Science, vol 10694. Springer, Cham

A Method for Developing Qualitative Security Risk Assessment Algorithms

Gencer Erdogan and Atle Refsdal

SINTEF Digital, Oslo, Norway, {gencer.erdogan,atle.refsdal}@sintef.no

Abstract. We present a method for developing qualitative security risk assessment algorithms where the input captures the dynamic state of the target of analysis. This facilitates continuous monitoring. The intended users of the method are security and risk practitioners interested in developing assessment algorithms for their own or their client's organization. Managers and decision makers will typically be end users of the assessments provided by the algorithms. To promote stakeholder involvement, the method is designed to ensure that the algorithm and the underlying risk model are simple to understand. We have employed the method to create assessment algorithms for 10 common cyber attacks, and use one of these to demonstrate the approach.

Keywords: security risk assessment, risk assessment algorithms, qualitative risk assessment

1 Introduction

Decision makers need to understand security risks to determine how to deal with them. Many managers, particularly at the business level, expect quantified assessments of risks in terms of estimated likelihood and monetary loss. Unfortunately, providing trustworthy numbers is very difficult. This requires not only insight into the systems, threats, vulnerabilities and business processes of the organization, but also access to good empirical data and statistics to serve as a foundation for quantified estimates. Such data is often unavailable. Even if we can obtain it, analyzing the data to understand its impact on the assessment is a major challenge [15].

This means that providing good quantitative assessments is not always feasible. In such cases, a qualitative approach can be a good alternative. By qualitative, we mean that we use ordinal scales, for which the standard arithmetic operators are not defined, to provide assessments. Each step is usually described by text, such as {Very low; Low; Medium; High; Very high}. More informative descriptions of each step can of course be given. Ordinal scales allow us to order values, thereby making it possible to monitor trends. Since security risk assessment is costly when performed manually, we need to find ways to reduce the effort required to update assessments.

The contribution of this paper is a *method for creating executable algorithms* for qualitative security risk assessment. The input to the algorithms captures the current state of the target of analysis, such as the presence of vulnerabilities, suspicious events observed in the application or network layer, and the potential consequence of security incidents for the business processes. The output from the algorithms is an assignment of qualitative risk values to all identified risks. Hence, an updated risk assessment can be obtained by rerunning the algorithms with new input. This facilitates continuous monitoring of the risk level.

Our goal is to create a method that does not require programming skills or extensive effort, ensures that the underlying risk models are documented in a comprehensible format, and results in transparent algorithms that can be understood by all stakeholders. We combine a graphical risk modeling technique, extended with a construct to capture dynamic factors, with a decision modeling technique to define the algorithm. The novelty of our approach lies in the integration of these techniques in an overall method that exploits their strengths. Both techniques are well-established, supported by freely available tools, and designed to provide models that can be easily understood.

In Sect. 2, we give an overview of the method, which consists of three steps. The next three sections explain each step. We then relate our method to other approaches in Sect. 6. Finally, in Sect. 7, we discuss the approach and report on initial experiences before concluding in Sect. 8.

2 Method Overview

Our method is illustrated in Fig. 1. In relation to the general security risk management process documented in ISO 27005 [10], the method starts at the risk assessment phase. That is, we assume that the context has been established, including purpose, scope, and target of analysis.

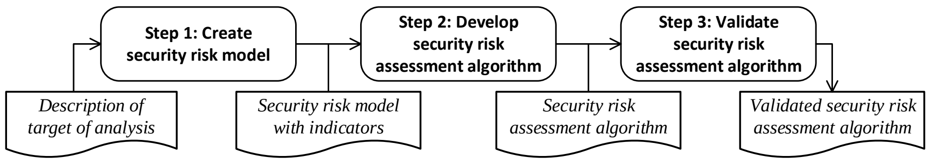

Step 1: Create security risk model
Step 2: Develop security risk assessment algorithm
Step 3: Validate security risk assessment algorithm
Description of target of analysis
Security risk model with indicators
Security risk assessment algorithm
Validated security risk assessment algorithm

Fig. 1. Method overview.

Step 1 takes as input a description of the target of analysis, which may be in the form of system diagrams, use case documentation, system manuals etc. Based on the description, we create a risk model to identify and document the assets, risks, threats and vulnerabilities of relevance for the target of analysis. At this point, estimates of likelihood and consequence values are represented by parameters. We also identify indicators that capture dynamic factors influencing the risk level, such as the presence or absence of a certain vulnerability, or the expected consequence for an asset if a given incident occurs. The indicators represent the input to the final security risk assessment algorithm, as each indicator will define one input variable. The output from Step 1 is a security risk model with indicators, as well as parameters representing likelihood and consequence estimates.

In Step 2, we develop the security risk assessment algorithm based on the security risk model. This is done in a modular fashion. We exploit the structure of the risk model developed in Step 1 and transform each part of the risk model to a corresponding part of the algorithm. The output of this step is an initial version of the security risk assessment algorithm.

In Step 3, we validate the risk assessment algorithm based on expert judgment to check if it produces correct output, and adjust as necessary. The output of this step is a validated security risk assessment algorithm.

3 Step 1: Create Security Risk Model

For creating security risk models, we have chosen to use CORAS [12], which is a graphical risk modeling language that has been empirically shown to be intuitively simple for stakeholders with very different backgrounds [18]. Moreover, CORAS comes with a method that builds on established approaches, in particular ISO 31000 [9]. The method includes detailed guidelines for creating CORAS models, which can be applied to carry out Step 1.

Figure 2 illustrates a CORAS risk model, which we use as a running example throughout the rest of this paper. This risk model is one of 10 risk models we developed in the WISER project [20]. These models were primarily intended for an arbitrary European SME. The risk model describes a Hacker carrying out an HTTP Verb tampering attack or a reflection attack in the authentication protocol to gain access to restricted files/folders. The risk is that the server provides access to restricted files/folders, which in turn has an impact on confidentiality. The dashed arrows, which are not themselves part of the CORAS model, are used to point out the different model elements. The unbroken arrows are relations in CORAS. There are three kinds of relations used to connect different nodes. The *initiates* relation goes from a threat to a threat scenario or an unwanted incident. The *leads to* relation goes from a threat scenario or an unwanted incident to a threat scenario or an unwanted incident. The *impacts* relation goes from an unwanted incident to an asset.

Notice that the risk model represents likelihoods, conditional likelihoods, and consequences as parameters. For example, the likelihood *l_S2* represents the likelihood of threat scenario *S2*. Our naming convention for parameters and other risk model elements is provided in Appendix A. We use the parameters later in the process to create the assessment algorithm.

Having created the risk model, next we identify indicators and attach them to the relevant risk model element. An indicator may be assigned to any risk-model element. We distinguish between four different types of indicators: *business configuration* (represented by the color blue in Fig. 2), *test* (green), *network-layer monitoring* (yellow, not included in the example), and *application-layer monitoring* (red). Values for the *business configuration* indicators are obtained by asking

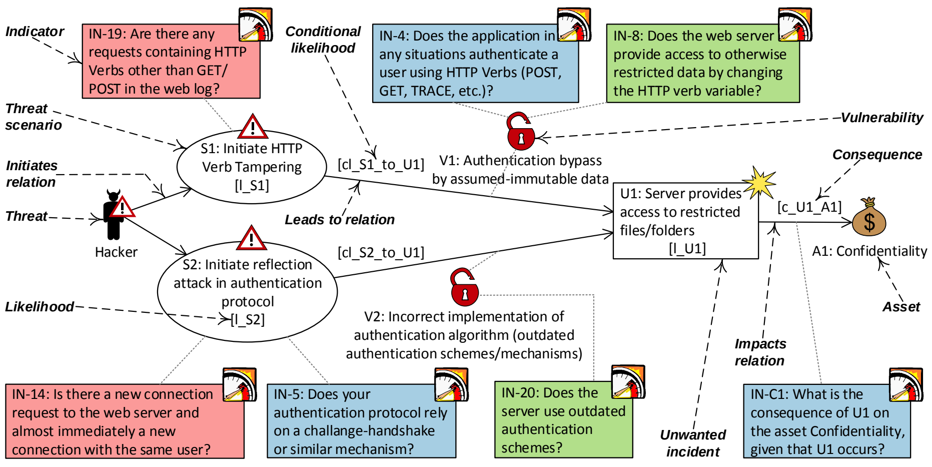

Indicator
IN-19: Are there any requests containing HTTP Verbs other than GET/ POST in the web log?
Conditional likelihood
IN-4: Does the application in any situations authenticate a user using HTTP Verbs (POST, GET, TRACE, etc.)?
IN-8: Does the web server provide access to otherwise restricted data by changing the HTTP verb variable?
Threat scenario
Vulnerability
S1: Initiate HTTP Verb Tampering [l_S1]
[cl_S1_to_U1]
V1: Authentication bypass by assumed-immutable data
Consequence
Initiates relation
U1: Server provides access to restricted files/folders [l_U1]
[c_U1_A1]
Threat
Leads to relation
Hacker
S2: Initiate reflection attack in authentication protocol [l_S2]
[cl_S2_to_U1]
A1: Confidentiality
Likelihood
V2: Incorrect implementation of authentication algorithm (outdated authentication schemes/mechanisms)
Asset
Impacts relation
IN-14: Is there a new connection request to the web server and almost immediately a new connection with the same user?
IN-5: Does your authentication protocol rely on a challange-handshake or similar mechanism?
IN-20: Does the server use outdated authentication schemes?
Unwanted incident
IN-C1: What is the consequence of U1 on the asset Confidentiality, given that U1 occurs?

Fig. 2. CORAS risk model.

business related questions. The indicator values are thus based on expert knowledge. Values for the *test* indicators are obtained by carrying out vulnerability tests. Values for the *network-layer monitoring* indicators and the *application-layer monitoring* indicators are obtained by monitoring the network layer and the application layer, respectively. However, all indicator types are treated the same in the guidelines presented in the next section.

4 Step 2: Develop Security Risk Assessment Algorithm

In this section, we show how to build an assessment algorithm from a CORAS model. But first we introduce the tool we use for algorithm definition and execution.

4.1 A brief introduction to DEXi

DEXi [7] is a computer program for development of multi-criteria decision models and the evaluation of options. We use DEXi because it has been designed to produce models that are comprehensible to end users [6]. The comprehensibility of DEXi seems to be confirmed by the fact that it has been applied in several different domains, involving a wide range of stakeholders [4–6]. For a detailed description, we refer to the DEXi User Manual [3].

A multi-attribute model decomposes a decision problem into a tree (or graph) structure where each node in the tree represents an attribute. The overall problem is represented by the top attribute. All other attributes in the tree represent sub-problems, which are smaller and less complex than the overall problem. Each attribute is assigned a value. The set of values that an attribute can take is called the *scale* of the attribute. DEXi supports definition of ordinal scales; typically, each step consists of a textual description.

Every attribute is either a basic attribute or an aggregate attribute. *Basic attributes* have no child attributes. This means that a basic attribute represents an input to the DEXi model, as its value is assigned directly, rather than being computed from child attributes. When using DEXi as a standalone tool, the user manually selects values for all basic attributes.

Aggregate attributes are characterized by having child attributes. The value of an aggregate attribute is a function of the values of its child attributes. This function is called the *utility function* of the attribute. The utility function of each aggregate attribute is defined by stating, for each possible combination of its child attribute values, what is the corresponding value of the aggregate attribute. The DEXi tool automatically computes the value of all aggregate attributes as soon as values have been assigned to the basic attributes. Hence, a DEXi model can be viewed as an algorithm where the basic attribute values constitute the input and the values of the aggregate attributes constitute the output. A java library and a command-line utility program for DEXi model execution is also available [7], meaning that functionality for executing DEXi algorithms can be easily integrated in software systems.

Figure 3 shows an example of a DEXi model which consists of three aggregate attributes and three basic attributes; the latter are shown as triangles. The top attribute, which is an aggregate attribute, is named *Risk* and has two child attributes (*Likelihood* and *Consequence*) that are also aggregate attributes. The *Likelihood* attribute has in turn two basic attributes as child attributes (*Likelihood indicator 1* and *Likelihood indicator 2*), while the *Consequence* attribute has one basic attribute as child attribute (*Consequence indicator 1*).

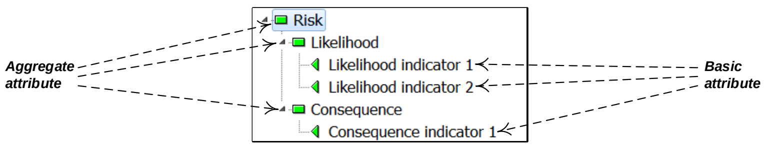


Fig. 3. DEXi model.

In the remainder of Sect. 4, we show how to build a security risk assessment algorithm, in the form of a DEXi model, based on a CORAS model. We use the model in Fig. 2 as an example. This means that the decision problem represented by the top attribute in the DEXi model concerns deciding the risk level. We start by explaining how each fragment of the CORAS model can be schematically translated to a corresponding fragment of the DEXi model in Sect. 4.2 to Sect. 4.7. A summary of the schematic translation is provided in Appendix A. Having thus shown how to build the DEXi model structure, we provide guidelines for defining scales and utility functions in Sect. 4.8.

4.2 Risk

CORAS representation For any risk, the risk level depends on the likelihood of the unwanted incident and its consequence for the asset in question. A risk corresponds to a pair of an unwanted incident and an asset, including the *impacts* relation from the incident to the asset. In Fig. 2, this corresponds to the unwanted incident *U1* and the *impacts* relation to asset *A1*. The likelihood of *U1* is denoted by *l_U1*, while the consequence of *U1* for asset *A1* is denoted by *c_U1_A1*.

DEXi representation A risk is represented as a top (i.e. orphan) attribute that has two child attributes, one representing the likelihood of the incident and one representing the consequence for the asset in question. Figure 4(a) shows the corresponding DEXi-representation of the CORAS fragment described above. We use the variable/node names in the risk model to express the corresponding DEXi fragment to make it easier to understand the connection between a CORAS risk model and its corresponding representation in DEXi. Hence, the top attribute *R1* in Fig. 4(a), which represents the risk, has the two child nodes *l_U1* and *c_U1_A1*. Notice that *R1* does not occur as a separate name in the CORAS diagram, as a risk is represented by the combination of the incident, the asset, and the relation between them, rather than by a separate node.

4.3 Node with incoming *leads-to* relations

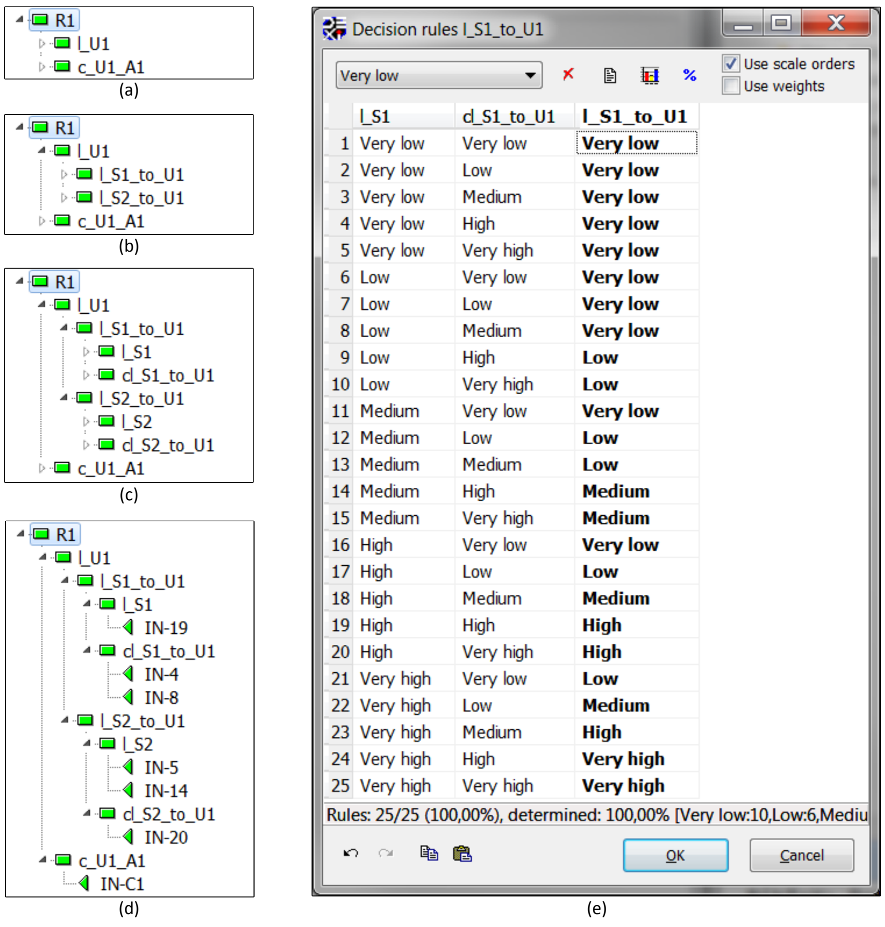
CORAS representation Figure 2 shows that the unwanted incident *U1* has two incoming *leads-to* relations, one from *S1* and one from *S2*. This means that the likelihood of *U1* depends on the likelihood contributions from *S1* and *S2*.

DEXi representation The likelihood of a node with incoming *leads-to* relations[1] is represented by an attribute with one child attribute for every incoming *leads-to* relation. The attribute *l_U1* in Fig. 4(b), which represents the likelihood of *U1*, therefore has two child attributes, *l_S1_to_U1* and *l_S2_to_U1*, representing the likelihood contributions from *S1* and *S2* via their outgoing *leads-to* relations.

4.4 Node with outgoing *leads-to* relation

CORAS representation The contribution from a *leads-to* relation to a target node depends on the likelihood of the source node and the conditional likelihood that an occurrence of the source node will lead to an occurrence of the target node. The latter is assigned to the *leads-to* relation. Figure 2 includes two *leads-to* relations, one from *S1* to *U1* and one from *S2* to *U1*. The likelihood contribution from the relation from *S1* depends on the likelihood of *S1* and the conditional likelihood that *S1* leads to *U1*, and similarly for *S2*.

[1] Recall from Section 3 that threat scenarios and unwanted incidents are the only node types that may have incoming *leads-to* relations.



(a)

```
R1
  l_U1
  c_U1_A1
```

(b)

```
R1
  l_U1
    l_S1_to_U1
    l_S2_to_U1
  c_U1_A1
```

(c)

```
R1
  l_U1
    l_S1_to_U1
      l_S1
      cl_S1_to_U1
    l_S2_to_U1
      l_S2
      cl_S2_to_U1
  c_U1_A1
```

(d)

```
R1
  l_U1
    l_S1_to_U1
      l_S1
        IN-19
      cl_S1_to_U1
        IN-4
        IN-8
    l_S2_to_U1
      l_S2
        IN-5
        IN-14
      cl_S2_to_U1
        IN-20
  c_U1_A1
    IN-C1
```

(e)

Decision rules l_S1_to_U1

Very low

☑ Use scale orders
☐ Use weights

	l_S1	cl_S1_to_U1	l_S1_to_U1
1	Very low	Very low	**Very low**
2	Very low	Low	**Very low**
3	Very low	Medium	**Very low**
4	Very low	High	**Very low**
5	Very low	Very high	**Very low**
6	Low	Very low	**Very low**
7	Low	Low	**Very low**
8	Low	Medium	**Very low**
9	Low	High	**Low**
10	Low	Very high	**Low**
11	Medium	Very low	**Very low**
12	Medium	Low	**Low**
13	Medium	Medium	**Low**
14	Medium	High	**Medium**
15	Medium	Very high	**Medium**
16	High	Very low	**Very low**
17	High	Low	**Low**
18	High	Medium	**Medium**
19	High	High	**High**
20	High	Very high	**High**
21	Very high	Very low	**Low**
22	Very high	Low	**Medium**
23	Very high	Medium	**High**
24	Very high	High	**Very high**
25	Very high	Very high	**Very high**

Rules: 25/25 (100,00%), determined: 100,00% [Very low:10,Low:6,Mediu

Fig. 4. Screenshots from the DEXi tool.

DEXi representation The likelihood contribution from a *leads-to* relation is represented by an attribute with two child attributes, one representing the likelihood of the source node and one representing the conditional likelihood that an occurrence of the source node will lead to the target node. As illustrated in Fig. 4(c), the attribute *l_S1_to_U1* representing the likelihood contribution from *S1* to *U1* therefore has two child attributes, *l_S1* representing the likelihood of *S1* and *cl_S1_to_U1* representing the conditional likelihood of *S1* leading to *U1* (and similarly for *l_S2_to_U1*).

4.5 Node with attached indicators

CORAS representation Indicators can be attached to a node to show that the indicators are used as input for assessing the likelihood of the node. Figure 2 shows that indicator *IN-19* is attached to threat scenario *S1*, while indicators *IN-5* and *IN-14* are attached to threat scenario *S2*.

DEXi representation Indicators attached to a node are represented as basic attributes under the attribute representing the node. Figure 4(d) shows the complete DEXi tree structure derived from the CORAS risk model in Fig. 2. The basic attributes in Fig. 4(d) correspond to the indicators in Fig. 2. As shown in Fig. 4(d), we add the indicator *IN-19* as a child attribute of *l_S1* and the indicators *IN-5* and *IN-14* as child attributes of *l_S2*.

Notice that we may have cases where a node has incoming *leads-to* relations in addition to attached indicators. In such cases, the attribute representing the node can have child attributes representing the incoming branches, as explained in Sect. 4.3, as well as the child attributes representing indicators.

4.6 *Leads-to* relation with attached indicators

CORAS representation Indicators can be attached to a *leads-to* relation from one node to another to show that the indicators are used as input for assessing the conditional likelihood of an occurrence of the source node leading to the target node. This is typically done by attaching the indicators to a vulnerability on the relation, as such indicators normally say something about the presence or severity of the vulnerability.

Figure 2 shows that indicator *IN-20* is attached to vulnerability *V2* and thus on the *leads-to* relation going from *S2* to *U1*. Similarly, indicators *IN-4* and *IN-8* are attached to vulnerability *V1*.

DEXi representation Indicators attached to a *leads-to* relation are represented by basic attributes under the attribute representing the conditional likelihood assigned to the relation. For example, two basic attributes representing IN-4 and IN-8 are children of *cl_S1_to_U1*. Hence, the conditional likelihood *cl_S1_to_U1* depends on the indicators *IN-4* and *IN-8*.

4.7 Other CORAS model fragments

We have not provided separate guidelines for threats, *initiates* relations, and indicators attached to *impacts* relations. For the latter, the reason is that a CORAS model does not provide any support for consequence assessment beyond the assignment of a consequence value to the *impacts* relation from an unwanted incident to an asset. All indicators relevant for consequence assessments are therefore represented as basic attributes directly under the attribute representing the consequence, as illustrated by *c_U1_A1* in Fig. 4(d). In our example, the single indicator attached to the *impacts* relation from *U1* actually provides the consequence value directly, which means that the *IN-C1* attribute could have been attached directly under *R1*, without the intermediate *c_U1_A1* attribute. We chose to include *c_U1_A1* to illustrate the general structure.

Concerning threats and *initiates* relations, we rarely assign likelihoods to these CORAS elements in practice, as estimating threat behavior is very difficult. Instead, we assign a likelihood directly to the target node of the *initiates* relation. An indicator assigned to a threat or to an *initiates* relation can therefore be handled as if it was assigned directly to the target node, following the guidelines of Section 4.5.

4.8 Defining scales and utility functions

Before defining utility functions, we need to define scales for the attributes. We strongly recommend using ordered scales consistently such that increasing the value implies increasing the (contribution to) the risk level, as this simplifies the definition of the utility functions. For all aggregate attributes in our running example, we use the scale {Very low; Low; Medium; High; Very high}.

We also make sure to follow the same scale order for the basic attributes representing the indicators. For example, consider the indicator *IN-19: Are there any requests containing HTTP Verbs other than GET/POST in the web log?* attached to *S1*. Since this is a yes/no question, the scale for the indicator only has two steps: Yes and No. A positive answer may indicate that someone has tried to prepare for an attack, and hence an increased likelihood. Therefore, for this indicator scale, the order from lowest to highest value should be {No; Yes}.

Assuming we have ordered all scales in this manner, increasing the value of a child attribute should never lead to a decrease in the value of its parent. The following restriction therefore guides the definition of utility functions:

Utility function restriction 1. *The value of an attribute should be monotonically increasing in all its child attributes. It does not have to be strictly increasing.*

For example, the risk *R1* should be monotonically increasing in the likelihood *l_U1* and the consequence *c_U1_A1*. Figure 5 illustrates an example of how a utility function fulfilling this restriction might be defined.

For the likelihood contribution from a *leads-to* relation, we also need to consider that the conditional likelihood of the source node leading to the target node will only affect the target node to the extent that the source node actually occurs. We therefore add the following restriction:

Utility function restriction 2. *The value of the attribute representing the likelihood contribution from a leads-to relation should not be higher than the value of the attribute representing the likelihood of the source node.*

For example, *l_S1_to_U1* should never be higher than *l_S1*. The screenshot from the DEXi tool in Figure 4(e) shows a definition of the utility function of *l_S1_to_U1* that respects both utility function restrictions presented above.

For a threat scenario or unwanted incident with incoming *leads-to* relations, the likelihood can clearly not be lower than the highest contribution from the incoming relations. We therefore add the following restriction:

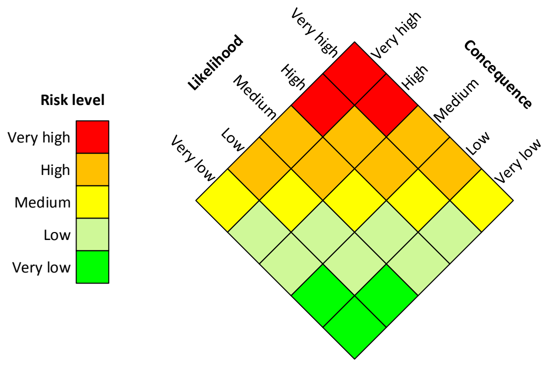


Fig. 5. Example of risk level defined as a monotonically increasing function of likelihood and consequence.

Utility function restriction 3. *The value of an attribute representing the likelihood of a node with one or more incoming leads-to relations should be at least as high as the highest value of the attributes representing the contributions from the incoming leads-to relations.*

For example, *l_U1* should be at least as high as the highest of *l_S1_to_U1* and *l_S2_to_U1*. Restrictions 2 and 3 apply under the assumption that the same scale is used for the attributes representing likelihood of nodes.

If a node has several incoming *leads-to* relations and/or attached indicators, combinatorial explosion can make it very hard to define the utility function for the attribute representing the node, due to the number of child attributes. In such cases, we recommend reducing the granularity of the scales of the child attributes or restructuring the model, as further explained in [3].

5 Step 3: Validate Security Risk Assessment Algorithm

Before putting the algorithm produced in Step 2 in operation, it should be validated to verify that its output correctly reflects reality. This can be done in many ways, depending on the data and resources available. When dealing with the kind of security risk assessment addressed in this paper, we typically need to rely on expert judgment. We first select a set of validation scenarios and then validate each of these with a team of experts.

A validation scenario is a set of indicator values representing one possible snapshot of the dynamic factors that influence the risk level. Thus, the number of possible scenarios is the product of the number of possible values for each indicator. This often results in many possible scenarios, which may be infeasible to validate. We therefore need to select a reasonable number of scenarios depending on the available effort. As a minimum, we suggest selecting validation scenarios based on the following two criteria: (1) cover the extreme scenarios (yielding the minimum and maximum risk values), and (2) cover each path in the CORAS risk model, meaning that for each path p (from the threat to the unwanted incident) in the risk model, there must be a scenario where one or more indicators along the path is triggered and the indicators for all other paths are not triggered unless these indicators also affect path p. By triggered, we mean that the indicator value contributes to the increase of likelihood. For the reasons given in Sect. 4.7 concerning consequence assessment, we focus on likelihood validation.

Our example in Fig. 2 includes six Boolean indicators affecting the likelihood assessment (as well as one affecting the consequence). This gives 64 possible scenarios. Table 1 gives an example of four scenarios fulfilling the criteria mentioned above. This represents the absolute minimum number to fulfill the coverage criteria, and we recommend using more validation scenarios. The Scenarios SC1 and SC4 satisfy the first coverage criterion. SC1 is the minimum risk scenario where no indicators are triggered, and SC4 is the maximum risk scenario where all indicators are triggered. The Scenarios SC2 and SC3 satisfy the second coverage criterion, where SC2 covers the top path of the risk model (involving "Initiate HTTP Verb Tampering") and SC3 covers the bottom path of the risk model (involving "Initiate reflection attack in authentication protocol"). The column *l_U1* shows the resulting likelihood value of *U1* for each scenario.

Table 1. Example of validation scenarios.

Scenario	**IN-19**	**IN-4**	**IN-8**	**IN-14**	**IN-5**	**IN-20**	**l_U1**
SC1	No	No	No	No	No	No	Very low
SC2	Yes	Yes	Yes	No	No	No	High
SC3	No	No	No	Yes	Yes	Yes	Medium
SC4	Yes	Yes	Yes	Yes	Yes	Yes	Very high

With respect to validating the selected scenarios, we recommend using a well-established approach such as the Wide-band Delphi method [1]. The Wide-band Delphi method is a forecasting technique used to collect expert opinion in an objective way, and arrive at consensus conclusion based on that. Another similar estimation approach is the Constructive Cost Model (COCOMO) [2].

6 Related Work

Most security risk approaches aim to provide assessments capturing the risk level at a single point in time, rather than continuous monitoring. However, there are also approaches that address dynamic aspects and offer support for updating assessments based on new information, such as the ones proposed by Poolsappasit et al. [14], Frigault et al. [8], Si-chao et al. [17], and Krautsevich et al. [11]. Common for all these is that they offer quantitative assessments based on variants of Bayesian Networks or Markov Chains. Building and understanding the models therefore requires specialized expertise. Many security and risk practitioners, and most managers and decision makers, do not possess this expertise.

Our qualitative approach based on CORAS and DEXi is designed to be simple and aimed at a broader user group.

DEXi is one of many approaches within the field of multi-criteria decision making (on which there is huge literature [19]), and has been tried out in a wide range of domains, such as health care, finance, construction, cropping systems, waste treatment systems, medicine, tourism, banking, manufacturing of electric motors, and energy [6, 7]. To the best of our knowledge, DEXi has not been used for security risk assessment. However, it has been applied to assess safety risks within highway traffic [13] and ski resorts [5]. Although they focus on safety risks, the approaches provided by Omerčević et al. [13] and Bohanec et al. [5] are similar to our approach in the sense that they use DEXi models as the underlying algorithm to compute an advice based on relevant indicators. Unlike our approach, they do not employ any dedicated risk modeling language to provide a basis for developing the DEXi models.

CORAS is a comprehensive framework for model-driven risk analysis. In addition to the risk modeling language, the framework consists of a tool and a comprehensive method [12]. However, CORAS focuses on quantitative assessment and does not address development of executable algorithms. Roughly speaking, what we have done in the work presented here is to insert the DEXi approach into the risk estimation step of the CORAS method and add indicators to capture dynamic aspects. The first use of measurable indicators as dynamic input to provide risk level assessments based on CORAS was presented in [16]. This is a quantitative approach aimed at developing mathematical formulas for assessing risk levels, where indicators are represented by variables in the formulas.

7 Discussion of the Approach and Initial Experiences

Our experience from applying the presented method to develop CORAS models and corresponding DEXi models for 10 common cyber attacks indicates that the method is easy to use. Therefore, our hypothesis is that most security and risk practitioners can adopt it without extensive additional training. In future work, we hope to test this hypothesis empirically with practitioners who have not been involved in developing the method.

This work was done in the context of the WISER project [20], which offers a framework for real-time security risk assessment where values for the basic attributes are automatically assigned from test tools, monitoring infrastructure and user interfaces. The risk assessment results (i.e. the output of the algorithms) are presented in a dashboard, which is part of the WISER framework. The DEXi models constitute the qualitative assessment algorithms offered by the framework, and will be tested on three pilots as part of the validation of the WISER framework. Notice that the method presented here does not require adoption of the WISER framework. It is possible to use only the DEXi tool to run the algorithms and view the results, if one is willing to manually feed the indicator values by assigning values to all basic attributes.

The effort required to apply the method obviously depends on several factors, including the complexity of the attack to be captured, the chosen abstraction level for the risk model, the number of indicators, and the choice of validation approach. As a rough rule of thumb, we estimate that applying the method will typically take from 20 to 40 hours of work. This estimate is based on our experience from the WISER models and applies for general attacks that are already well understood, such as the one presented here. If a more complex CORAS model needs to be developed from scratch in Step 1, the effort may be significantly higher. Due to the guidelines provided for Step 2, we can reasonably assume that the effort required to develop the DEXi model will grow more or less in proportion to the number of elements in the CORAS model.

While the schematic translation to obtain the DEXi model structure is straightforward and could even be automated, the definition of scales and utility functions is based on subjective judgment. Here we see a potential for more extensive guidelines. These could, for example, provide guidance on the degree of impact on parent nodes for different types of indicators.

An inherent limitation of the approach is that new threats and attack types can only be addressed by creating new risk models and algorithms. The reason is that the only dynamic aspects that can be captured by the algorithms are those covered by the identified indicators. Periodic evaluations should therefore be performed to decide whether new or updated models and algorithms are required. This limitation applies for all approaches that rely on human experts.

8 Conclusion

We have presented a method for developing security risk assessment algorithms, using ordinal scales with textual descriptions for risk level assessments. Indicators constitute the input to the algorithms and capture the current state of the target of analysis, such as the presence of vulnerabilities, suspicious events observed in the infrastructure or the potential impact of security incidents on business processes. By producing an algorithm, rather than an assessment capturing the risk level at a single point in time, the method facilitates continuous assessment and monitoring of trends.

The method is designed to provide risk models documented in a comprehensible format and transparent assessment algorithms that can be understood by all stakeholders, without requiring programming skills or extensive effort. This promotes user involvement, critical scrutiny and improvement, and helps build trust in the results. Based on our initial experiences, we believe our work can contribute to enhanced security and better decision making by helping organizations to obtain transparent and comprehensible security risk assessments.

Acknowledgments. This work has been conducted as part of the WISER project (653321) funded by the European Commission within the Horizon 2020 research and innovation programme.

References

1. B. W. Boehm. *Software Engineering Economics.* Prentice Hall, 1981.
2. B.W. Boehm, C. Abts, A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D.J. Reifer, and B. Steece. *Software Cost Estimation with COCOMO II.* Prentice Hall, 2000.
3. M. Bohanec. DEXi: Program for Multi-Attribute Decision Making. User's Manual v 5.00 IJS DP-11897, Institut "Jožef Stefan", Ljubljana, Slovenija, 2015.
4. M. Bohanec, G. Aprile, M. Costante, M. Foti, and N. Trdin. A Hierarchical Multi-Attribute Model for Bank Reputational Risk Assessment. In *DSS 2.0 – Supporting Decision Making with New Technologies*, pages 92–103. IOS Press, 2014.
5. M. Bohanec and B. Delibašić. Data-Mining and Expert Models for Predicting Injury Risk in Ski Resorts. In *Proc. 1st International Conference on Decision Support System Technology (ICDSST'15)*, pages 46–60. Springer, 2015.
6. M. Bohanec, M. Žnidaršič, V. Rajkovič, I. Bratko, and B. Zupan. DEX Methodology: Three Decades of Qualitative Multi-Attribute Modeling. *Informatica (Slovenia)*, 37(1):49–54, 2013.
7. DEXi: A Program for Multi-Attribute Decision Making. `http://kt.ijs.si/MarkoBohanec/dexi.html`. Accessed January 9, 2017.
8. M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring Network Security Using Dynamic Bayesian Network. In *Proc. 4th ACM Workshop on Quality of Protection (QoP'08)*, pages 23–30. ACM, 2008.
9. International Organization for Standardization. *ISO 31000:2009(E), Risk management – Principles and guidelines*, 2009.
10. International Organization for Standardization. *ISO/IEC 27005:2011(E), Information technology – Security techniques – Information security risk management*, 2011.
11. L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Risk-Aware Usage Decision Making in Highly Dynamic Systems. In *Proc. 5th International Conference on Internet Monitoring and Protection (ICIMP'10)*, pages 29–34. IEEE, 2010.
12. M. S. Lund, B. Solhaug, and K. Stølen. *Model-Driven Risk Analysis: The CORAS Approach.* Springer, 2011.
13. D. Omerčević, M. Zupančič, M. Bohanec, and T. Kastelic. Intelligent response to highway traffic situations and road incidents. *Proc. Transport Research Arena Europe 2008*, pages 21–24, 2008.
14. N. Poolsappasit and I. Ray. Dynamic Security Risk Management Using Bayesian Attack Graphs. *International Journal On Advances in Intelligent Systems*, 9(1):61–74, 2012.
15. A. Refsdal, B. Solhaug, and K. Stølen. *Cyber-Risk Management.* Springer, 2015.
16. A. Refsdal and K. Stølen. Employing Key Indicators to Provide a Dynamic Risk Picture with a Notion of Confidence. In *Proc. 3rd IFIP International Conference on Trust Management (TM'09)*, pages 215–233. Springer, 2009.
17. L. Si-chao and L. Yuan. Network Security Risk Assessment Method Based on HMM And Attack Graph Model. In *Proc. 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 517–522. IEEE, 2016.
18. B. Solhaug and K. Stølen. The CORAS Language – Why it is designed the way it is. In *Proc. 11th International Conference on Structural Safety & Reliability (ICOSSAR'13)*, pages 3155–3162. Taylor and Francis, 2013.

19. M. Velasquez and P.T. Hester. An analysis of multi-criteria decision making methods. *International Journal of Operations Research*, 10(2):56–66, 2013.
20. Wide-Impact cyber SEcurity Risk framework (WISER). `https://www.cyberwiser.eu/`. Accessed February 8, 2017.

A Schematic Translation from CORAS to DEXi

Table 2 shows the naming convention for risk model elements including likelihood and consequence parameters. The lower-case letters x and y in the table represent integers.

Table 2. Naming convention.

Name	**Meaning**
Ax	Asset x
Rx	Risk x
Sx	Scenario x (threat scenario)
Ux	Incident x ('U' stands for unwanted incident)
Vx	Vulnerability x
l_Ux	Likelihood of Ux
l_Sx	Likelihood of Sx
c_Ux_Ay	Consequence of Ux for Ay
cl_Sx_to_Sy	Conditional likelihood of Sx leading to Sy
cl_Sx_to_Uy	Conditional likelihood of Sx leading to Uy
cl_Ux_to_Sy	Conditional likelihood of Ux leading to Sy
cl_Ux_to_Uy	Conditional likelihood of Ux leading to Uy
l_Sx_to_Sy	Likelihood contribution from Sx to Sy
l_Sx_to_Uy	Likelihood contribution from Sx to Uy
l_Ux_to_Sy	Likelihood contribution from Ux to Sy
l_Ux_to_Uy	Likelihood contribution from Ux to Uy
IN-x	Indicator x
IN-Cx	Consequence indicator x

Figure 6 shows an overview of risk model fragments and their schematic translation from CORAS to DEXi. In all except the "Risk" row, we have used threat scenarios to illustrate nodes. However, these threat scenarios may also be replaced by unwanted incidents. The lower-case letters x, y, z, u, v, and n in Fig. 6 represent integers.

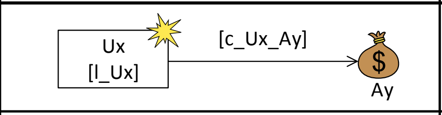
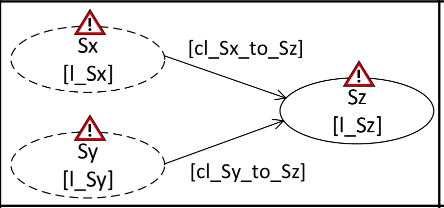
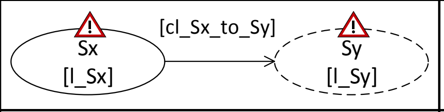
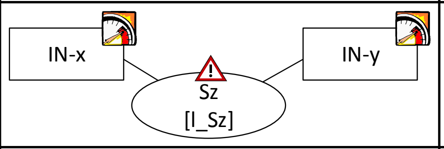
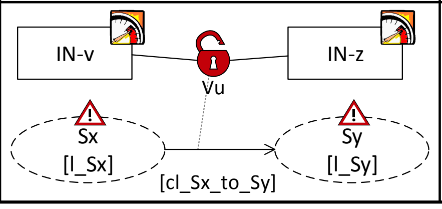
Fragment in risk model	**CORAS representation**	**DEXi representation**
Risk (Section 4.2)		Rn l_Ux c_Ux_Ay
Node with incoming *leads-to* relations (Section 4.3)	 Sx [l_Sx] [cl_Sx_to_Sz] Sz [l_Sz] Sy [l_Sy] [cl_Sy_to_Sz]	l_Sz l_Sx_to_Sz l_Sy_to_Sz
Node with outgoing *leads-to* relation (Section 4.4)		l_Sx_to_Sy l_Sx cl_Sx_to_Sy
Node with attached indicators (Section 4.5)		l_Sz IN-x IN-y
Leads-to relation with attached indicators (Section 4.6)	 IN-v IN-z Vu Sx [l_Sx] Sy [l_Sy] [cl_Sx_to_Sy]	cl_Sx_to_Sy IN-v IN-z

Fig. 6. Schematic translation from CORAS to DEXi