Relationship-Based Threat Modeling

Stef Verreydt
imec-DistriNet, KU Leuven
Heverlee, Belgium
stef.verreydt@kuleuven.be

Laurens Sion
imec-DistriNet, KU Leuven
Heverlee, Belgium
laurens.sion@kuleuven.be

Koen Yskout
imec-DistriNet, KU Leuven
Heverlee, Belgium
koen.yskout@kuleuven.be

Wouter Joosen
imec-DistriNet, KU Leuven
Heverlee, Belgium
wouter.joosen@kuleuven.be

ABSTRACT

Threat modeling is a common technique for the systematic analysis of system designs to uncover security and privacy threats. Popular threat modeling techniques, however, currently only consider a very localized system context, which hinders the discovery of more complex attack scenarios that involve multiple interactions throughout a system. This may lead to the underestimation of threats that are not harmful by themselves but enable multiple other high-risk threats. Furthermore, current risk assessment approaches require stakeholders to take the system as a whole into account when providing inputs, which is tedious and error-prone.

This paper introduces relationship-based threat modeling (RBTM). Using explicitly captured threat relationship knowledge, RBTM allows to systematically and automatically generate a threat graph which is then used as input for traceable risk calculations. This removes the need to manually take into account threat relationships during risk assessments and allows stakeholders to clearly identify and communicate the rationale behind the resulting risk values. The outputs of an RBTM analysis were compared to those of a manual one performed by experts to evaluate the soundness of our proposal, which also highlighted the traceability benefits.

CCS CONCEPTS

• **Security and privacy → Software security engineering**; • **Software and its engineering → Risk management**; *Model-driven software engineering*; *Software design engineering*.

KEYWORDS

Threat modeling, risk management, attack trees, attack graphs

ACM Reference Format:

Stef Verreydt, Laurens Sion, Koen Yskout, and Wouter Joosen. 2022. Relationship-Based Threat Modeling. In *The 3rd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS'22), May 16, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3524489.3527303

EnCyCriS'22 , May 16, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9290-7/22/05…$15.00
https://doi.org/10.1145/3524489.3527303

1 INTRODUCTION

Threat modeling enables stakeholders to identify and address security and privacy threats early in the software development lifecycle, reducing development time in the long run [4]. The first step of a threat modeling exercise is to model the system, usually as a Data Flow Diagram (DFD). Potential security threats can then be revealed using threat elicitation approaches such as STRIDE-per-element [12], which iterates over all elements of the DFD.

Elicited threats should be assessed according to their relative importance to avoid wasting resources on irrelevant ones or spending insufficient resources on critical ones. A number of prioritization approaches exist to do so, for example OWASP's risk rating methodology [18], the FAIR risk model [3], or automated prioritization approaches [14] that rely on tool support [13]. All of these, however, require stakeholders to provide estimates for the likelihood and impact of the identified threats, which is a challenging task for multiple reasons. First, possible relationships between threats should be accounted for. For example, an attacker successfully impersonating another user at some web server may lead to tampering or information disclosure threats on databases deeper within the system, so the impact and/or likelihood of these threats should be related. The above-mentioned prioritization approaches provide no guidance on how to systematically account for such relationships. Second, there is the problem of threat explosion: empirical evidence [16, 19] shows that adding a single element to a system model introduces around ten new STRIDE threats. Thus, even for smaller models, tens or even hundreds of impact and likelihood estimates are needed to rank all possibly related threats. Third, changes in the system model require re-evaluating all estimates, as threats and threat relationships may change. This last point is especially relevant early on in the development lifecycle, when models are rapidly changing.

In the context of network security, techniques exist to automatically generate attack[1] graphs (e.g., [7, 10]) which can then be analyzed to (dis)prove security properties or find the probability of certain states being reached. To the best of our knowledge, these techniques have never been applied in the context of early-stage software threat modeling. We believe there are two main reasons for this. First, the precise attack definitions required for attack graph generation are incompatible the high-level threats such as those elicited with STRIDE. Second, the input model for such techniques is

[1]We use the term 'threat' to refer to high-level, undesirable events (e.g., spoofing), whereas we use the term 'attack' for technology or implementation-specific steps which could be leveraged to realize a threat (e.g., a buffer overflow in an ssh server) [8].

a network description (usually in terms of connected hosts, firewalls and so on [7]), but threat modeling is not limited to networks.

In this paper, we propose RBTM, a relationship-based threat modeling approach that enables the systematic incorporation of relationships between threats for security analyses at design time. Based on explicitly captured threat relationship knowledge, RBTM allows to systematically and automatically generate a threat graph which is then used as input for traceable risk calculations, thereby guiding stakeholders towards the most important threats to be mitigated. Furthermore, taking threat relationships into account explicitly removes the need for stakeholders to consider the system as a whole when providing the input for a risk calculation exercise.

The remainder of this paper discusses background information and introduces the running example (Section 2), presents our proposed approach (Section 3), evaluates it based on the running example (Section 4), and discusses related work (Section 5), before concluding in Section 6.

2 BACKGROUND AND RUNNING EXAMPLE

This section provides some background information on threat modeling and introduces the running example.

2.1 Threat Modeling Background

Our proposal relies on the DFD notation, STRIDE (as discussed by Shostack [12]), and an extension to the DFD notation which enables systematically capturing security knowledge.

DFD notation. The first step of a threat modeling exercise is to model the system being analyzed. Such system model usually takes the form of a Data Flow Diagram (DFD). The DFD notation comprises just five elements, namely *processes*, *data stores*, *external entities*, *data flows* and *trust boundaries*.

STRIDE. STRIDE [12] is an acronym that stands for *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service* (*DoS*) and *Elevation of Privilege* (*EoP*). In essence, it is merely a mnemonic to help stakeholders reason about potential threats applicable to some system model. STRIDE-per-element and STRIDE-per-interaction are variants which elicit potential threats by systematically iterating over a system model's elements and interactions respectively. When applying the per-element variant, an elicited threat denotes the threatened element and the STRIDE threat type, for example a tampering threat on process X. With the per-interaction variant, on the other hand, elicited threats also encompass the data flow on which the threat could take place, for example, a tampering threat on process X via data flow Y. The extra context information gained from applying the per-interaction variant makes it the preferred elicitation method for our proposal. Numerous tools [11] exist to apply systematic variants of STRIDE.

Security extension. The STRIDE approach to threat modeling, as discussed above, does not take into account information on existing security countermeasures by default. This lack of security information prevents systematically and/or automatically adjusting the risk of a threat based on whether or not it has been mitigated. Our approach therefore relies on and extends (Section 3.1.2) the solution-aware DFD notation proposed by Sion et al. [15], the main

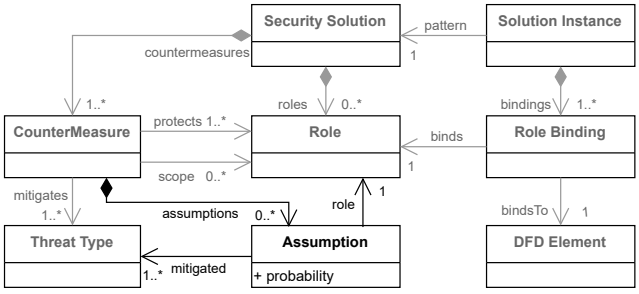

Figure 1: Extended solutions meta-model.
Grey classes and relationships depict the original meta-model by Sion et al. [15], and the extension (Section 3.1.2) is shown in black.

concepts of which are shown in Fig. 1. Given the notation, a Security Solution contains a list of Roles which describe the involved DFDElements. For example, a solution for credential-based authentication could include roles for (i) the data store containing user credentials, (ii) the authenticating process, and (iii) the authenticated flow. A Role can be protected by certain CounterMeasure(s), which determine which specific ThreatType(s) to that element are mitigated, for example the authenticating process being protected from spoofing threats. Additionally, a CounterMeasure can specify a scope to which its protection applies. For example, the spoofing protection is only valid for the authenticated flow, and entities may still be spoofed via other flows. Solutions can be instantiated in concrete models by assigning RoleBindings describing which elements act as the required Roles in a concrete model, for example, data store X contains the user credentials, process Y does the authentication, and data flow Z is authenticated. Based on this information, tools can automatically mark threats as mitigated if they are protected against by a solution.

Data type annotations. Finally, our proposal can leverage data type information if it is captured explicitly in the DFD model, for example as discussed by Tuma et al. [17]. This allows threat relationship knowledge such as "*disclosing user credentials induces spoofing threats*" to be more easily expressed and applied, as will be illustrated in Section 4.

2.2 Running Example

As a running example, we use DocProc, a case developed several years ago in the context of a course on software architecture at our university. DocProc is a generalization of a real-world business-to-business software system for document processing, offering services to automate document generation and delivery. Customer organizations provide DocProc with the raw data and templates required to generate and deliver their documents (e.g., invoices). DocProc then generates the documents and delivers them to their intended recipients via one of the supported delivery channels. DocProc also allows recipients to register themselves to a personal document store (PDS) that contains all documents sent to them.

Modeling the DocProc delivery system at a coarse-grained abstraction level results in a DFD with 4 external entities, 3 processes, 3 data stores, and 23 data flows (Fig. 2). We shortly describe each

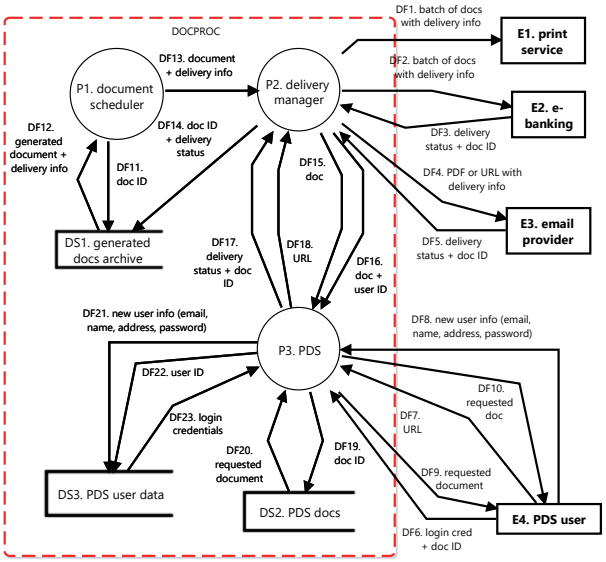

Figure 2: DFD of the DocProc document delivery subsystem. *Circles, closed rectangles, and open rectangles represent processes, external entities and data stores respectively. Arrows denote data flows, and the dashed line denotes a trust boundary.*

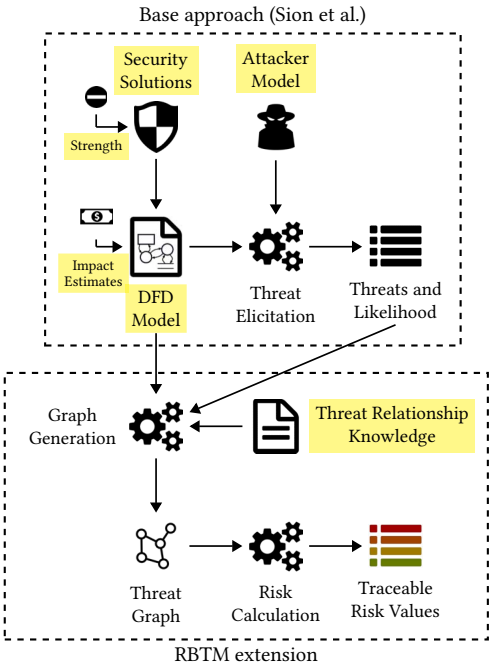

Figure 3: High-level overview of RBTM. *Required inputs are highlighted in color.*

component's responsibilities in what follows. Generated documents and delivery status information is stored in DS1. P1 is responsible for scheduling the generated documents for delivery. P2 handles the actual delivery and tracks the delivery status for the delivery services which support them. The supported delivery methods are print and postal (E1), e-banking (E2) and email (E3). PDS users (E4) have their documents stored in DS2. P3 is responsible for handling incoming PDS requests, including authorization and authentication. Details and credentials of PDS users are stored in DS3.

3 RELATIONSHIP-BASED THREAT MODELING

A high-level overview of the Relationship-Based Threat Modeling (RBTM) approach is provided in Fig. 3. The approach of Sion et al. [13] is leveraged to elicit threats and calculate their likelihood. RBTM then generates a threat graph by applying threat relationship knowledge to the list of elicited threats, thereby also considering the system model and any security solutions. Finally, the threat graph and impact estimates are used as inputs for a risk calculation algorithm, resulting in traceable risk values. This section first provides an overview of the additional inputs required to apply RBTM compared to the base approach, before discussing both the threat graph generation and risk calculation. Finally, we shortly summarize the capabilities required to apply RBTM.

3.1 RBTM Inputs

Besides the inputs needed to apply the base approach, RBTM also requires threat relationship knowledge and knowledge on how solutions can be broken or circumvented.

3.1.1 Threat Relationship Knowledge. We define a threat relationship using the following properties: (i) the cause threat, (ii) the effect threat, (iii) the propagation condition, and (iv) the propagation probability. For example, a tampering threat on a process (*cause*) induces a *DoS* threat on a data flow (*effect*), if that data flow starts from or ends at the threatened process (*propagation condition*) with a probability of 100% (*propagation probability*). In what follows, each of these properties are described in more detail.

Cause and effect threats. In the example above, cause and effect are expressed generically (*a* tampering threat on *a* process), without specifying concrete DFD elements. In contrast, eliciting threats for a specific system model results in concrete context-specific threats (for example, for Fig. 2, tampering with process P2). We make this distinction explicit by referring to these types of threats as, respectively, *abstract threats* and *concrete threats*. Analogously, we refer to generic threat relationships such as the given example as *abstract threat relationships*. *Concrete threat relationships* are then, as the name suggests, relationships between concrete threats. For example, applying the aforementioned abstract threat relationship to the DocProc DFD results in several concrete threat relationships, including that tampering threats on P1 induce *DoS* threats on DF11, DF12, and DF13. The meta-model in Fig. 4 visualizes these concepts.

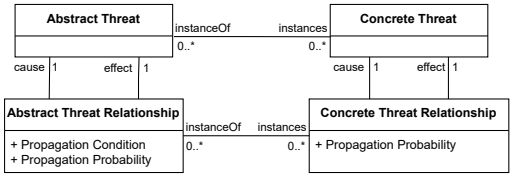

Figure 4: RBTM threats and relationships meta-model.

Propagation condition. There are some restrictions on how threats can induce one another. For example, tampering with a process induces *DoS* threats on data flows, but only on those starting from or ending at the affected process. It is therefore necessary to capture how the cause threat and the effect threat of an abstract threat relationship are related. Examples of other threat relationships requiring propagation conditions are that tampering with a process via some data flow induces *DoS* threats on that same process via the same data flow, and that the sender of a data flow being spoofed may induce *EoP* threats on the recipient of that data flow.

Propagation probability. A script kiddie could accidentally tamper with a web application, but it requires a skilled adversary to specifically tamper with a process to obtain elevated privileges. Turning a spoofing threat into an *EoP* threat, on the other hand, requires no effort at all: an attacker successfully identifying as another user automatically grants them their privileges. A threat relationship is therefore enriched with a propagation probability, which denotes the probability that a successful realization of the cause threat will lead to a successful realization of the effect threat.

3.1.2 Solution assumptions. Introducing new security countermeasures may also introduce new threat relationships. For example, disclosing user credentials may invalidate an authentication solution, and spoofing threats may thus occur even though there is a defense in place. The solution-aware DFD notation discussed in Section 2.1 does not, however, allow capturing how solutions can be broken or circumvented. We therefore extended it with assumptions on which security solutions rely, including a probability with which invalidating the assumption would break the solution. For example, we could add an assumption to the credentials-based authentication solution (Section 2.1) stating that information disclosure and tampering threats on the credentials database break the solution with a probability of 100%. This, in turn, enables systematically finding threat relationships from threats invalidating a solution's assumption to threats mitigated by that solution, as will be illustrated in Section 3.2.

3.1.3 Reusable knowledge. Generally applicable abstract threat relationships, such as spoofing threats inducing *EoP* threats, can be captured in catalogs to enable reuse across multiple system models. The same goes for knowledge on security solutions and attacker models. Having access to reusable catalogs reduces the amount of effort required to provide all inputs for an RBTM analysis, as well as lowering the required security expertise. Developing such reusable catalogs is, however, left for future work.

3.2 Graph Generation

A threat graph generated by RBTM is one where the nodes are concrete threats and edges are concrete threat relationships weighted with propagation probabilities. The concrete threats are found through threat elicitation (top of Fig. 3). RBTM identifies the concrete threat relationships in two ways: (1) by applying abstract threat relationships, and (2) by applying solution assumption knowledge.

The pseudo-code[2] for applying an abstract threat relationship is shown in Algorithm 1. For example, applying the abstract threat

Algorithm 1: Apply an abstract threat relationship.

```
input  : AbstractThreatRelationship atr;
         DFDModel dfd;
output : Set<ConcreteThreatRelationship> result;
cCauses ← atr.cause.instances;
cEffects ← atr.effect.instances;
for ( cCause ← cCauses )
    for ( cEffect ← cEffects )
        checkPropagationConditions(cCause, cEffect, dfd);
        if ( propagation conditions met )
            result.add(new
              ConcreteThreatRelationship(cCause, cEffect,
              atr.probability));
```

relationship "a tampering threat on a process (*cause*) induces a *DoS* threat on a data flow (*effect*), if that data flow starts from or ends at the threatened process (*propagation condition*) with a probability of 100% (*propagation probability*)" to the DocProc case goes as follows. First, concrete instances of the abstract cause threat are identified in the set of elicited threats, for example tampering threats on P2 via DF3 and DF6. Similarly, *DoS* threats on DF1 through DF23 are all concrete instances of the abstract effect threat. A concrete threat relationship is then created for each combination of concrete cause and effect which meets the propagation condition. For example, a tampering threat on P2 via DF3 and a *DoS* threat on DF3 fit the propagation condition, so RBTM creates a relationship between the former and the latter. A tampering threat on P2 via DF3 and a *DoS* threat on DF11, on the other hand, do not fit the propagation condition, so no relationship will be instantiated for this combination.

The pseudo-code for applying solution assumption knowledge is shown in Algorithm 2. In short, each threat which breaks an assumption for a certain solution may allow circumventing that solution and therefore, in turn, induce the threats against which the solution protects. For example, for DocProc, instantiating the credentials-based authentication solution discussed earlier (where DS3 contains user credentials, P2 authenticates requests, and DF6 is the authenticated flow) allows RBTM to identify concrete threat relationships as follows. The solution mitigates the spoofing of external entity E4 on DF6. If the user credentials in DS3 are disclosed via DF23, this breaks one of the authentication solution's assumptions. Therefore, RBTM creates a concrete threat relationship from the information disclosure threat to the mitigated spoofing threat.

[2]The pseudo-code only depicts the main idea. The developed prototype (Section 4) uses a combination of pattern matching and custom code to identify threat relationships.

Algorithm 2: Apply solution assumption knowledge.

```
input  : Set<ConcreteThreat> allThreats;
         SolutionInstance s;
output : Set<ConcreteThreatRelationship> result;
cEffects ← { t ∈ allThreats | s.mitigates(t) };
for ( Assumption a ← s.assumptions )
    cCauses ← { t ∈ allThreats | t.invalidates(a) };
    for ( cCause ← cCauses )
        for ( cEffect ← cEffects )
            result.add(new
              ConcreteThreatRelationship(cCause, cEffect,
              a.probability));
```

3.3 Risk Estimation

The risk R_t of a threat t is generally defined as a combination of its likelihood L_t and impact I_t, i.e., $R_t = L_t \times I_t$. Traditionally, the likelihood of a threat takes into account all possibilities for an attacker to realize it, and the impact considers all potential negative effects, both direct and indirect. For example, assume that a certain spoofing threat has been mitigated by the credentials-based authentication solution described earlier, but that there is an unmitigated information disclosure threat on the data store containing the credentials. If this spoofing threat is impactful, then traditional risk calculation approaches would assign it a high risk value, as the authentication solution that mitigates the threat could be circumvented by leaking user credentials. This may, however, give the false impression that another solution is needed for the spoofing threat, even though it is the information disclosure threat which should be addressed. In other words, the traditional interpretation of risk does not help stakeholders in deciding which threats to mitigate first. Furthermore, providing likelihood and impact estimates requires taking into account the system as a whole, which is challenging for multiple reasons, as discussed earlier. To tackle these issues, RBTM redefines the likelihood and impact in a relationship-based context.

3.3.1 Likelihood. As described in the previous paragraph, a threat receiving a high risk value from RBTM should indicate that a mitigation for *that* threat should be introduced. With RBTM, the likelihood of a threat therefore only concerns an attacker's potential to realize it as an initial entry point for an attack, disregarding any potential threat relationships. Thus, a threat having a likelihood of zero in RBTM does not mean that attackers can never realize the threat, only that they cannot realize it as an initial entry point. For example, a mitigated spoofing threat will receive a low likelihood value from RBTM, meaning that attackers will most likely not be able to use it as an initial entry point. Attackers could still, however, realize that spoofing threat by first realizing unmitigated information disclosure threats on the data store containing credentials, the likelihood of which would be high due to the lack of mitigations. Assigning a likelihood in RBTM is thus greatly simplified, as only the probability of immediately realizing that particular threat, without realizing other threats first, must be taken into account. Threat relationships are accounted for as part of the impact, as will be described next.

3.3.2 Impact. RBTM defines the impact of a threat as a combination of its direct and indirect impacts I_{dt} and I_{it}, i.e., $I_t = I_{dt} + I_{it}$. The direct impact of a threat denotes the extent in which it directly counteracts business goals or missions, disregarding the system as a whole or potential threat relationships. The indirect impact of a threat then accounts for its relationship to other threats which counteract business goals. Whereas the direct impact must be estimated by stakeholders, the indirect impact can be derived systematically by leveraging the generated threat graph. We shortly discuss each of these components in what follows.

Direct impact. In the case of DocProc, retaining customer trust and avoiding fines could be considered core business goals. As information disclosure threats on the data store that contains user documents (DS2 in Fig. 2) directly counteract these business goals, they should be assigned a high direct impact estimate. On the other hand, an *EoP* threat on P2 may induce threats that counteract business goals (e.g., disclosing user documents), but there are no damages linked *directly* to an attacker gaining elevated privileges. The same reasoning can be applied for most spoofing or elevation of privilege threats, and for tampering threats on processes. The number of threats with a significant direct impact is therefore most likely small in comparison to the total number of threats. Furthermore, providing direct impact estimates requires minimal effort, as threats are examined in isolation of the system, and relationships to other threats do not need to be accounted for.

Indirect impact. If a threat t induces another threat t', then the indirect impact of t should include the direct impact of t'. The probability of t inducing t' must, however, also be taken into account, which is why the threat propagation probability is captured. We define the most reliable path (MRP) from t to t' as to the chain of threat relationships which an attacker could exploit to turn threat t into threat t' with the least possible resistance. Similarly to the algorithm used by Sarraute et al. [9], RBTM calculates the MRP by using a modified shortest path algorithm. In summary, we define the indirect impact of a threat t (I_{it}) as follows:

$$I_{it} = \sum_{t'} I_{dt'} \times P_{MRP_{t \to t'}}. \tag{1}$$

Here, $I_{dt'}$ is the direct impact of another threat t' and $P_{MRP_{t \to t'}}$ is the probability of the (MRP) from t to t'. Note that in a worst-case analysis, $P_{MRP_{t \to t'}}$ is always equal to 1 whenever t may lead to t'.

3.4 Required Capabilities

Being able to systematically apply solution and threat relationship knowledge, which can be obtained from reusable catalogs, and only having to provide direct impact estimates, significantly reduces the required effort and (security) knowledge for a risk estimation exercise. In short, applying RBTM requires the capabilities to (i) construct a DFD for the analyzed system, including the applied security solutions; (ii) determine the threats with a direct (business) impact and estimate their impact; (iii) identify threat relationships specific to the analyzed system; and (iv) comprehend the output of RBTM. Here, only the last two require security knowledge, which is a clear improvement over having to manually take solution knowledge and potential threat relationships into account when estimating the relevance of a threat.

4 SOUNDNESS EVALUATION

To evaluate RBTM, we have implemented it as an extension of a previously developed tool, SPARTA [13], which automates the base approach shown in Fig. 3. We define the soundness of RBTM as its ability to prioritize threats that are also found and deemed important by a panel of security experts. The soundness evaluation therefore depends on a baseline, namely an independent threat analysis of DocProc by an expert panel. In what follows, we provide an overview of the baseline, RBTM inputs, and results. The complete analysis data can be found online [1].

4.1 Baseline

The baseline used for this evaluation consists of a list of 35 threats for DocProc. This list was created independently by a panel of six experts, as part of another research project prior to this work.[3] All experts were researchers familiar with security and software design. Each expert independently analyzed the DFD of DocProc (Fig. 2) using STRIDE and noted down the threats that they deemed important and their rationale. The used baseline is the union of these threats, i.e., all threats found by at least one of the experts.[4] The experts also gave a priority to the threats (high, medium, or low), although they rarely agreed (Fleiss' $\kappa = 0.12$, indicating slight agreement). This highlights the need for a systematic relevance assessment approach. When experts disagreed on the priority for a threat, we use the most frequently given one.

4.2 RBTM Inputs

The experts were only given the DFD shown in Fig. 2 and a general case description. To stay as close as possible to the expert analysis, the case-specific inputs used for the RBTM evaluation only include information that was available in either of these sources. This information was complemented with commonly available security knowledge (which experts may have applied implicitly). We shortly discuss the rationale behind the inputs and describe some examples.

4.2.1 Data types. The DFD provided to the experts is annotated with textual data type information. For our evaluation, we added this information to the DFD model using structured annotations.

4.2.2 Security solutions. While no solutions are captured explicitly in Fig. 2, a few can be derived from just the DFD. For example, using the extended solution notation (Fig. 1), we explicitly captured that P3 authenticates users at DF6 through user credentials, but only if the user credentials are not leaked or tampered with, and if P3 itself is not tampered with. Here, the data type annotations allow to more easily capture the assumption that user credentials should not be leaked or tampered with, as tool support can then automatically identify which DFD elements deal with user credentials. The case description provided to the experts also described several assumptions. These include, for example, that "*we do not consider any threats that originate from within the trust boundary*", and that data flows 1 through 10 (DF1–DF10) are "*encrypted and thus considered to have channel confidentiality and integrity*". These were translated to solutions so that the relevant threats would be marked as mitigated. We assumed that security solutions fully mitigate the threats they protect against, so we assigned these threats a likelihood of zero (of being used as entry point, cfr. Section 3.3.1).

4.2.3 Threat relationships. Potential threat relationships were not explicitly provided to the experts, although they may have relied on their experience to reason about them (which they did, as evidenced by their rationale, in which they mention possible causes and effects of threats). With RBTM, these relationships can be included explicitly and obtained from reusable catalogs.

Sixteen generally applicable threat relationships were obtained from the threat tree patterns proposed by Howard and Lipner [4]. For example, in the threat tree pattern for 'Tampering with a process', 'Provide fake credentials' is listed as a possible cause. This was translated to an abstract threat relationship from a spoofing threat on some process to a tampering threat on that same process. Furthermore, five case-specific threat relationships were added, namely that tampering with delivery information induces information disclosure threats on P2 via DF1, DF2 and DF3, and on P3 via DF9 and DF10, as documents are sent to the wrong recipient. The structured data type annotations again allow tool support to automatically identify tampering threats on delivery information.

For the evaluation, we assume a worst-case scenario where all propagation probabilities are 100%. This assumption may not hold in practice. Similarly, while the collection of threat relationships described by Howard and Lipner [4] is not necessarily complete, it serves as an adequate starting point for systematic threat graph generation. Formulating a more complete set of threat relationships and propagation probabilities is left for future work.

4.2.4 Direct impact estimates. As mentioned in Section 3.3, RBTM only requires direct impact estimates for threats with immediate business value. The main rationale applied for the direct impact assignments is as follows. Threats against the integrity and confidentiality of the generated documents, as well as availability of delivery and PDS components, were assigned high direct impact estimates, as they directly counteract the main goals of DocProc. Furthermore, several repudiation threats should be mitigated to ensure accountability, and are thus assigned a medium direct impact estimate Finally, threats which increase the operational costs, such as attackers sending documents to the delivery services in the name of DocProc, are also assigned a medium direct impact estimate.

By leveraging the developed tool, this rationale can be applied automatically using seven relatively simple rules such as "Add 2 to the direct impact of information disclosure and tampering threats on elements dealing with generated documents". The full list of rules can be found online [1].

4.3 Results

Applying RBTM to the DocProc example with the inputs from above resulted in 191 threats, the majority of which (154) were assigned a risk value of zero by RBTM because of a solution or assumption, or not having a (direct or indirect) impact. Three threats identified by the experts were not elicited by RBTM. The reason for this is that the used threat elicitation engine was based on the threat elicitation tables from Shostack [12], which do not identify them as

[3] Since we do not rely on specific attack or vulnerability knowledge, the gap in knowledge between our study and the expert analysis is limited.

[4] The threats in the expert baseline included a mixture of 28 per-element and per-interaction threats. To allow for comparison with RBTM threats, we homogenized them into 35 per-interaction threats.

		RBTM assessment					
		H	M	L	0	NA	Total
Expert	H	**6**	–	–	9	–	15
	M	3	**0**	3	–	–	6
	L	5	–	**3**	3	3	14
	Not mentioned	16	–	1	**142**	–	159
	Total	30	0	7	154	3	194

Table 1: Comparison of risk assigned by experts and RBTM.

relevant. There are only four non-zero possible risk values, mainly due to the discretization of the inputs (e.g., assuming that solutions fully mitigate threats and a propagation probability of 100%). Seven threats received a low risk value (2.25 or 4.5). Such threats have a direct impact but do not induce other impactful threats. Thirty threats were assigned a high risk value (166.5 or 162). These induce other impactful threats, and potentially have some direct impact themselves. Since the difference between low and high risk values is clear, we labeled threats as such to allow easy comparison to the expert analysis (Table 1).

Compared to the expert analysis baseline, RBTM yields a different risk estimate for 40 of the 191 elicited threats. Using Cohen's κ, the inter-rater agreement between RBTM and expert prioritization is measured to be 0.335, indicating fair agreement. If we had not taken into account threat relationships and applied just the base approach (top part of Fig. 3) to the described inputs, the resulting inter-rater agreement would be 0.192, which indicates slight agreement.

These results show that, with respect to the expert baseline, RBTM results in improved risk estimates when compared to a traditional risk estimation method [13] that does not take threat relationships into account, if only direct impact estimates are provided. Only having to estimate the direct impact significantly reduces the amount of effort needed, as potential threat relationships do not need to be accounted for. Furthermore, the prioritization of RBTM shows fair agreement with the judgment of experts (whose reasoning is primarily driven by tacit, informal knowledge). In what follows, we describe cases where the RBTM and expert ratings diverge and elucidate on possible reasons for the discrepancy.

4.4 Discrepancies

We only discuss the most important discrepancies (high vs. low, 0, or not mentioned), as the difference between the other categories is more subtle and open to interpretation.

Nine threats were marked as highly relevant by experts but received a risk value of zero from RBTM. One of these, namely spoofing E4 via DF6, was marked as important by the experts even though a solution for that threat was explicitly mentioned in the DFD and case description. The experts' rationale ("*Possible through weak credential storage at client side*") reveals that they implicitly reason about what threats could invalidate the solution. As discussed in Section 3.3, RBTM would not want to assign high risk values to such threats, as this might give the impression that additional mitigations are needed for the spoofing threats, while in fact it is an information disclosure threat which should be addressed. The eight other threats which were deemed highly relevant by experts, but assigned a zero risk value by RBTM, were all covered by assumptions. Similarly, the expert rationale describes how assumptions may be broken by implicitly reasoning about threat relationships, whereas RBTM does so explicitly by leveraging the threat graph. For all nine threats in this category, the discrepancy thus stems from RBTM's interpretation of risk, where high risk threats are the ones to be mitigated, not the ones due to other (unmitigated) threats.

Sixteen threats were not identified by experts but got assigned a high risk value by RBTM. Nine of them concern leaking user credentials or document URL's, which invalidates the authentication solution. Some of these threats were mentioned by the experts in the rationale for other threats, which seems contradictory. For example, as described earlier, the experts found spoofing threats on E4 via DF6 highly relevant despite their being an authentication solution, because they are "*Possible through weak credential storage at client side*". Even though information disclosure threats on E4 cover "*weak credential storage at client side*", they were not identified as relevant by the experts. The eight other threats in this category were spoofing and *EoP* threats which induced information disclosure threats of credentials, therefore also allowing to circumvent the authentication solution. The output produced by the experts did not allow us to trace why these 16 threats were not mentioned. From the above example, we can only assume that the experts simply forgot to consider them explicitly.

Finally, in five cases, RBTM assigned a high risk value whereas the experts assigned a low one. RBTM may have overestimated these threats due to imperfect threat relationship input or the worst-case assumption. However, inspecting the threat graphs did not reveal any unreasonable threat relationships or scenarios, so experts may also have underestimated such threats. Either way, the advantage of RBTM is that it allows to clearly trace why a threat is assigned a certain estimate by analyzing the threat relationships, which is not possible given the output produced manually by the experts.

4.5 Threats to validity and future work

The main threat to validity of our experiment is that it is based only on the DocProc system. Investigating additional cases is necessary to generalize these findings and gain deeper insights into the reasons behind any deviations. A major obstacle for performing additional studies, however, is the lack of (publicly) available, independently performed, and well-documented threat analysis outcomes. Furthermore, we assumed a worst-case scenario which may not hold in practice, and the used catalog of general threat relationships may be incomplete. In future work, we therefore aim to capture a more complete set of generally applicable threat relationships, including propagation probabilities. Last, context-specific threat relationships are now added to the system model as seemingly isolated threat relationships. For example, the DocProc model captures that tampering with delivery information induces information disclosures, but it does not explicitly capture *why* this is the case. Adding such information directly to a system model would further increase the traceability of an RBTM analysis.

5 RELATED WORK

A recent study [11] compared the most popular threat modeling tools based on several criteria, one of them being the quantification of risk. Except for Threagile [2], all analyzed tools did so based on either predefined values, such as CVSS scores, or required a manual analysis by end-users. Threagile also allows calculation-based ones, but these still require end-users to manually enter impact estimates for each asset, which requires considering the system as a whole. With RBTM, only the direct impact needs to be provided manually, which does not require taking into account the context.

Kaynar [5] provides an overview of attack graph generation and analysis techniques used in the context of network security. While none of these techniques are directly applicable during early-phase threat modeling, RBTM uses similar algorithms. The graph generation algorithm used by RBTM is similar to the one proposed by Ou et al. [7], as the nodes in their graphs are single vulnerabilities rather than system states. The main differences are that attack graphs, in their proposal, have a single attacker goal as a root, whereas RBTM considers all business goals, and that relationships are strict prerequisites (i.e., to reach some node, all children must be reached), which is not the case with RBTM. Furthermore, to the best of our knowledge, none of the existing attack graph generation proposals consider how mitigations to some vulnerabilities could be circumvented: a vulnerability is either mitigated, or it is not. This also leads to one of the main differences between RBTM and existing attack graph analysis techniques. As described by Kaynar [5], several proposals aimed at network hardening aim to calculate the minimal set of vulnerabilities to be mitigated to secure the whole network, but these do not consider model changes introduced by security solutions or any assumptions on which they rely. Finally, RBTM uses a modified shortest path algorithm to calculate the risk values, similar to the proposal of Sarraute et al. [9]. While such algorithms automatically deal with loops in the graph, they may suffer from scalability issues, which is why recent work in the context of network security usually prefers DAG-based formalisms like Bayesian networks [6]. Scalability is less of an issue for RBTM, as nodes in the graph are threats rather than system states, and thus much less prone to combinatorial explosion.

6 CONCLUSION

This paper described RBTM: a systematic approach to incorporate threat relationships into risk-driven threat prioritization. By combining traditional DFD-based threat modeling approaches with threat relationships, RBTM enables systematic and automatic threat graph generation and traceable risk calculation. This, in turn, removes the need to consider the system as a whole when specifying likelihood and impact estimates, as the indirect impact of a threat can be automatically derived from the threat graph. Furthermore, inspecting the generated threat graph allows stakeholders to clearly identify and communicate the reason for a certain risk value.

RBTM was implemented as an Eclipse plugin and compared to an independent expert threat assessment for a specific case. Our analysis shows that RBTM is able to produce a threat ranking which fairly agrees with the expert assessment. In cases where RBTM disagrees with an expert's risk estimate, we were able to clearly identify the reason for the discrepancy by analyzing the generated threat graph. On top of that, this analysis exposed one of the main disadvantages of manual risk estimations, namely that we cannot systematically deduce why experts assigned a certain priority to a certain threat, or why certain threats were not considered at all. Finally, since most solution information and generally applicable threat relationships can be obtained from reusable knowledge, applying RBTM requires minimal security knowledge. Only having to provide direct impact estimates further reduces the effort needed to provide inputs for a risk calculation exercise. RBTM is thus a promising technique to reduce the reliance of threat assessments on security experts, by capturing and automatically applying (part of) their knowledge.

ACKNOWLEDGMENTS

This research is partially funded by the Flemish Research Programme Cybersecurity and the KU Leuven C2-ePIC project.

REFERENCES

[1] 2022. RBTM analysis data. https://doi.org/10.5281/zenodo.6365047.
[2] Christian Schneider. 2021. Threagile. https://threagile.io/.
[3] Jack Freund and Jack Jones. 2014. *Measuring and Managing Information Risk: A FAIR Approach.* Butterworth-Heinemann.
[4] Michael Howard and Steve Lipner. 2006. *The Security Development Lifecycle.*
[5] Kerem Kaynar. 2016. A taxonomy for attack graph generation and usage in network security. *Journal of Information Security and Applications* 29 (Aug. 2016), 27–56. https://doi.org/10.1016/j.jisa.2016.02.001
[6] Barbara Kordy, Ludovic Pièlre-Cambacédès, and Patrick Schweitzer. 2014. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review* 13-14 (2014), 1–38. https://doi.org/10.1016/j.cosrev.2014.07.001
[7] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. 2006. A Scalable Approach to Attack Graph Generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06).* 336–345. https://doi.org/10.1145/1180405.1180446
[8] Paul Saitta, Brenda Larcom, and Michael Eddington. 2005. Trike v.1 Methodology Document [Draft]. (2005), 17.
[9] Carlos Sarraute, Gerardo Richarte, and Jorge Lucángeli Obes. 2011. An algorithm to find optimal attack paths in nondeterministic scenarios. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence - AISec '11.* 71. https://doi.org/10.1145/2046684.2046695
[10] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. 2002. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy.* 273–284. https://doi.org/10.1109/SECPRI.2002.1004377
[11] Zhenpeng Shi, Kalman Graffi, David Starobinski, and Nikolay Matyunin. 2021. Threat Modeling Tools: A Taxonomy. *IEEE Security Privacy* (2021), 2–13. https://doi.org/10.1109/MSEC.2021.3125229
[12] Adam Shostack. 2014. *Threat Modeling: Designing for Security* (1st ed.).
[13] Laurens Sion, Dimitri Van Landuyt, Koen Yskout, and Wouter Joosen. 2018. SPARTA: Security & Privacy Architecture Through Risk-Driven Threat Assessment. In *International Conference on Software Architecture.* IEEE, 89–92. https://doi.org/10.1109/ICSA-C.2018.00032
[14] Laurens Sion, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. 2018. Risk-based Design Security Analysis. In *1st IEEE/ACM International Workshop on Security Awareness from Design to Deployment (SEAD)*, Vol. 1. 1–8. https://doi.org/10.1145/3194707.3194710
[15] Laurens Sion, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. 2018. Solution-Aware Data Flow Diagrams for Security Threat Modeling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing* (Pau, France) *(SAC '18).* 1425–1432. https://doi.org/10.1145/3167132.3167285
[16] Katja Tuma, Riccardo Scandariato, Mathias Widman, and Christian Sandberg. 2018. Towards Security Threats that Matter. In *Computer Security.* 47–62.
[17] Katja Tuma, Laurens Sion, Riccardo Scandariato, and Koen Yskout. 2020. Automating the early detection of security design flaws. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems.* 332–342.
[18] Jeff Williams. 2020. *OWASP Risk Rating Methodology.* https://owasp.org/www-community/OWASP_Risk_Rating_Methodology
[19] Kim Wuyts, Dimitri Van Landuyt, Aram Hovsepyan, and Wouter Joosen. 2018. Effective and efficient privacy threat modeling through domain refinements. 1175–1178. https://doi.org/10.1145/3167132.3167414