Attack-Defense-Trees and other Security Modeling Tools

Benjamin Löhner
Advisor: Heiko Niedermayer
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: b.loehner@tum.de

ABSTRACT

A recent study [14] shows that US companies took an average of 206 days to detect a data breach. With increasingly complex computer systems and networks, mitigating such attacks quickly becomes a major task for modern organizations. Taking into account the small margin of error, security modeling and risk management tools present a viable solution to cope with this issue. In this paper, we therefore give an overview over some existing tools on security modeling. After an in-depth view of attack defense trees, we cover alternative approaches using UML as well as practical implementations in the form of the risk management software Verinice. These tools are then evaluated and compared to each other regarding various situations and questions from a practical viewpoint. Finally, each tools strengths and weaknesses are summarized.

Keywords

IT-Security, Security Modeling, Attack Defense Trees, Unified Modeling Language, Misuse Cases, Mal-activity Diagrams, Extended Statechart Diagrams, Verinice

1. INTRODUCTION

Information is one of the main assets of many organizations. In most modern businesses it is stored digitally, distributed via networking infrastructure and processed on several endpoints. If this data is accessed by unauthorized actors, the consequences can be devastating, ranging from business crises to threads with nation wide impact. A rather extreme example are the ransomware attacks on Germany in 2017, which temporarily affected public infrastructure like the national railway and hospitals internal databases [11]. It is known that its way of infection was based on the EthernalBlue exploit, which was published by WikiLeaks after a data breach at the CIA [28]. To keep the chance of an attack as low as possible, a system must be sufficiently protected. To aid this process, security models and risk management tools have been introduced. With the above risks in mind, the demand on such tools is high and manifold: First, it should not only enable the user to build a good model of the real components to be protected, but also help in finding possible attack vectors. After the surfaces are identified, it is desirable for a model to support the addition of possible defenses. Depending on the situation, a user might also want to evaluate the resulting model mathematically, generating metrics like probabilities to measure the risk of a successful attack. Depending on the situation, further, more specific questions may surface: How do we model dynamic defenses? Can the tool deal with a defense failing? In this paper, we present three tools for security modeling. Then, we compare them by a number of features, including the requirements and questions above. Finally, we evaluate their performance and suitability in practical scenarios.

2. SECURITY MODELING TOOLS

2.1 Attack defense trees

2.1.1 *Concept and features of ADTs*

Similar to attack trees, ADTs are trees with labeled nodes. These are split into two categories: Attacks an attacker might launch against a systems component (*attack nodes*) and countermeasures a defender employs to ensure the systems protection (*defense nodes*). While the former are drawn as circles, defense nodes are depicted as rounded rectangles. Edges represent causal relationships. By default, edges are *disjunctive*, meaning that the parent nodes attack is considered successful, if at least one of its children's condition is true. On the contrary, edges adjacent to a parent can be marked *conjunctive* by grouping them together with connecting arcs. Conjunctive edges simulate the behavior of ATs by requiring every nodes condition in the group to be fulfilled for the parents action to be successful. The distinction between attack and defense nodes induces two kinds of relationships: Edges can connect two nodes of the same category, thus decomposing them into components or different possibilities (*refinements*). Edges connecting attacks and defenses are called *countermeasures* and are highlighted by dotted lines. To simplify the structure, a node can only have one child of the opposite type, which is usually drawn as the rightmost child. Apart from these guidelines, the basic ADT can be modeled freely, including parents with edges and connected nodes of mixed kinds.

2.1.2 *An example*

Figure 1 shows an example ADT based on the example in Kordy's 2014 paper [18]. As noted on the trees root, the main goal is the protection of "Data Confidentiality". Indicated by the two conjunctive edges leading to the defense nodes "Network Security" and "Physical Security", a breach in either of these fields would result in data being exposed. The latter is only directly affected by the possibility of a "Break In", which is then further decomposed into various possible entry points. As any entry would result in a successful break in, disjunctive edges are used. Network security on the other hand is more multilayered, requiring multiple defense mechanisms to be active simultaneously in order to protect the companies information. Namely in the

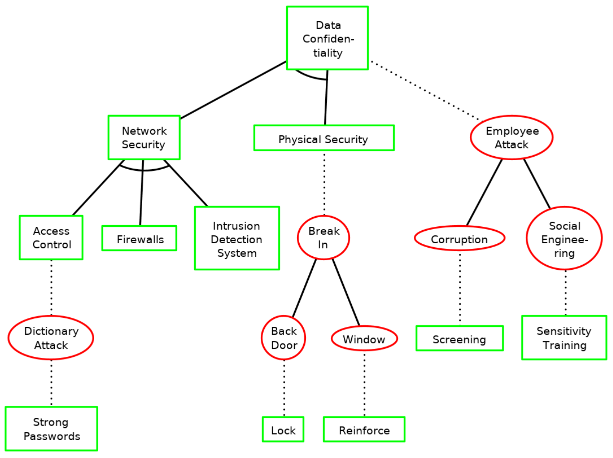


Figure 1: Example attack defense tree

example, "Access Control", "Firewalls" and "Intrusion Detection Systems" are listed and further refined. Additionally, there could be an attack on the companies employees. This possibility is further refined into "Corruption" and "Social Engineering", which respectively is protected against by "Screening" during the recruitment process and "Sensitivity Training" of employees.

2.2 Extended UML

Unified Modeling Language (UML) is a general purpose tool for visualizing various views and components of computer and software systems. As of 2018 it is one of the foundational modeling tools in software engineering and taught in almost every kind of software related education. Consequently, this extensive use led to various extension to make UML applicable to security modeling. Therefore in this section, we present UML-based tools for security modeling. For each tool, we first explain its design and extensions over standard UML, followed by an example and finally provide some information on the main scope and the models use cases.

2.2.1 *Misuse case diagrams*

Misuse case diagrams (MCDs) [26] are an extension to common UML use case diagrams (UCDs). As opposed to UCDs, which only feature neutral *actors* to interact with the system, MCDs differentiate between non-evil *actors* and *misusers*, commonly distinguished by different colored symbols. Similarily, *uses cases* are accompanied by their counterpart *misuse cases*, using the same color code. These components form nodes in a directed unweighted graph. For edges, ordinary use case relationships like *extend*, *generalize* or *include* are supported, while the security specific interactions are expressed by the new tags *threaten* and *mitigate*. A use case which has a directed edge of latter type to a misuse case is called *security use case*.

The example in figure 2 taken from Sindres paper [26] depicts use and misuse cases for an e-commerce store. In addition to the normal use cases of a system, security and misuse cases are added. For instance, the actor "Customer" can "Order goods". One possible abuse would be to either intercept traffic or use other means to capture sensitive data

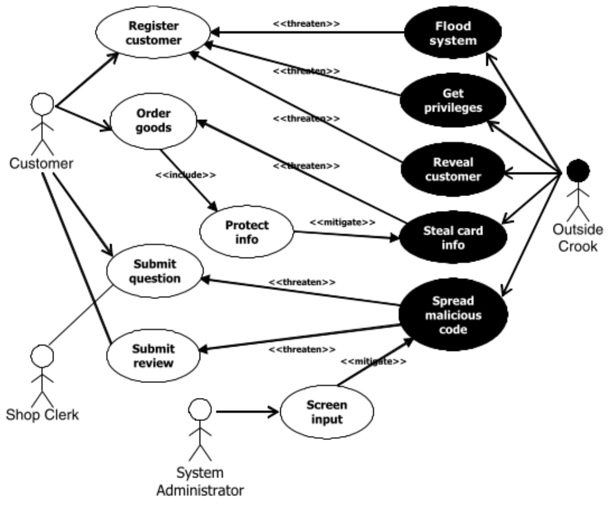


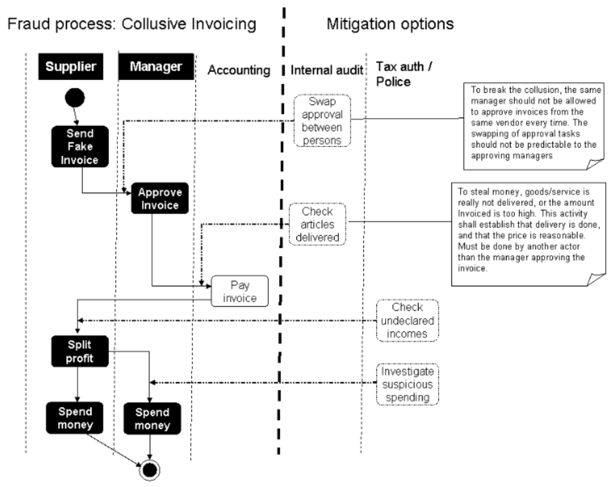
Figure 2: Example misuse case diagram

while the user interacts with the ordering website. This is depicted by the misuser "Outside Crook"'s use case of "Stealing card info", which has a threatening relationship to the "Order goods" case. In this example, "Order goods" includes a "Protect info" node, which could use measures such as transport level encryption to protect against the attack, indicated by the mitigate relationship to the "Steal card info" node. Likewise, other use cases partly involving different actors are modeled.

By design, MCDs are most suitable for modeling threads at the system boundary, providing a good overview of its attack surfaces.

2.2.2 *Mal-activity diagrams*

Mal-activity diagrams (MADs) [25] use the same base syntax as activity diagrams, while providing similar extensions as the misuse case models. Namely, they support *malicious actors* and *malicious activities*, distinguished by a different color. In addition, we can use *malicious decision boxes* to model decisions with malicious intent.



To break the collusion, the same manager should not be allowed to approve invoices from the same vendor every time. The swapping of approval tasks should not be predictable to the approving managers

To steal money, goods/service is really not delivered, or the amount Invoiced is too high. This activity shall establish that delivery is done, and that the price is reasonable. Must be done by another actor than the manager approving the invoice.

Figure 3: Example mal-activity diagram

In the example taken from Sindres paper [25] in figure 3 on the left side, a supplier and a manager collude to get fraudulent invoices paid and spend the profit themselves. A mal-activity in the ”Supplier” column labeled ”Send Fake Invoice” and one edge directed to a ”Manager” node ”Approve invoice” model the malicious collaboration between supplier and manager. After the next neutral activity ”Pay invoice” is completed by the ”Accounting” department, a similar structure of mal-activities is used to describe the splitting and spending of the profit. Similarly, the defense mechanisms are depicted on the right. There, the node ”Swap approval between persons” and ”Check articles delivered” add new conditions to the completion of ”Approve Invoice” and ”Pay invoice” respectively. Their exact function and behavior is described in the constraint boxes on the very right. Likewise, ”Split profit” and ”Spend money” is extended by ”Check undeclared incomes” and ”Investigate suspicious spending” activities by the ”Tax auth / Police”.

MADs are more suitable for modeling attacks either inside or outside of a system with a focus on involved actors.

2.2.3 *Extended statechart notation*

To avoid symbol ambiguity, extended statechart models (ESMs) [8] introduce six new types of states. Theses are given the descriptive names *threatened state*, *vulnerable state*, *defensive state*, *compromised state*, *quarantine state* and *recovery state*, which are also used to label their depictions (e.G. <<`vulnerable`>>). In addition, they can optionally be distinguished by different color and symbols. Furthermore, the extended notation provides *thread-* and *countermeasure events* to model offensive or defensive relationships as well as *inital thread-* and *final compromised nodes* to indicate special start or final states. Like extended states, they can be colored differently. Optionally, thread event lines can be drawn dashed.

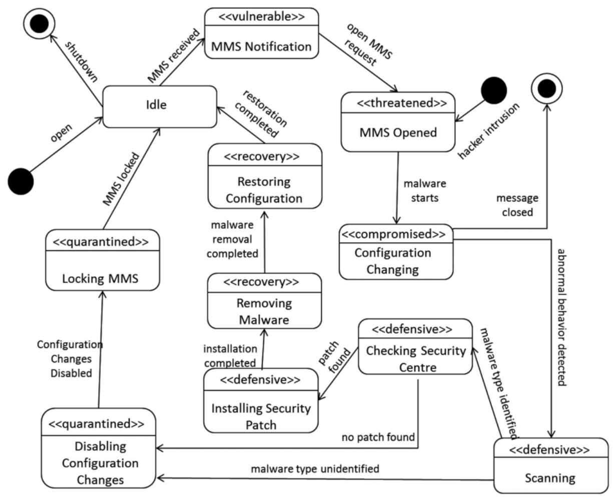


Figure 4: Example extended statechart diagram

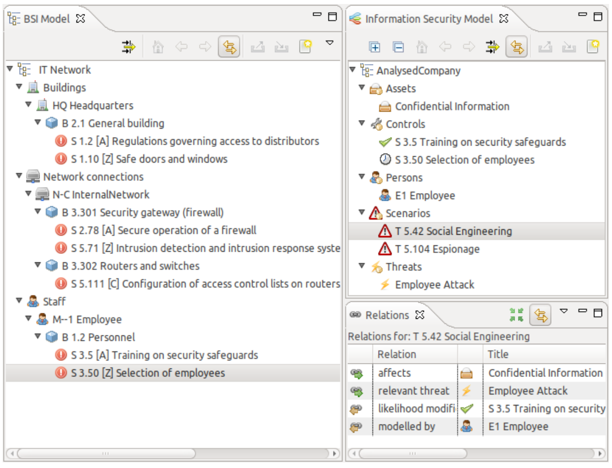
Extensive use of the different kinds of states can be seen in the example in figure 4 taken from El-Attars paper [8]. It shows how Android phones are attacked by the ”AndroidOs.FakePlayer” malware and what defense mechanisms can be used to treat such an attack. It first starts by describing the way of infection, starting at the initial node on the left. After the phone receives an MMS, it transitions from ”Idle” to an vulnerable state, as it display the ”MMS Notification”. As of this time, the system does not get infected as long as the user leaves the notification unused. If it is opened however, we transition the edge ”open MMS request” to the threatened state ”MMS Opened”. This is the point where the hacker actually gains access to the device, indicated by a new initial node ”hacker intrusion”. From there on, the ”malware starts”, leaving the device in a compromised state, after its ”Configuration Changed”. From there on, different security measures are modeled, following the same principles as above. If abnormal behavior is detected, the device launches a defensive security scan. If it can identify the malware, it transitions through multiple defensive states and finally reach recovery states where it removes the virus and restores the devices original state. In the case of no successful identification, it continues to transition through quarantined states to disable to misbehaving components. Either way, the thread is mitigated and phone goes into ”Idle” again.

ESMs are particularly useful to model one single process in great detail.

2.3 Models in practical risk management tools - Verinice

In the industry, the models presented are most of the time not used in their theoretical forms. Instead, their concepts are incorporated into risk management tools, which include common scenarios and defenses, as well as providing ways to automate and simplify some of the proof- and risk probability analysis tasks.

In the following section, we show the parallels between theory and practice by describing some key aspects of the application Verinice [10].


BSI Model

- IT Network
  - Buildings
    - HQ Headquarters
      - B 2.1 General building
        - S 1.2 [A] Regulations governing access to distributors
        - S 1.10 [Z] Safe doors and windows
  - Network connections
    - N-C InternalNetwork
      - B 3.301 Security gateway (firewall)
        - S 2.78 [A] Secure operation of a firewall
        - S 5.71 [Z] Intrusion detection and intrusion response syste
      - B 3.302 Routers and switches
        - S 5.111 [C] Configuration of access control lists on routers
  - Staff
    - M--1 Employee
      - B 1.2 Personnel
        - S 3.5 [A] Training on security safeguards
        - S 3.50 [Z] Selection of employees

Information Security Model

- AnalysedCompany
  - Assets
    - Confidential Information
  - Controls
    - S 3.5 Training on security safeguards
    - S 3.50 Selection of employees
  - Persons
    - E1 Employee
  - Scenarios
    - T 5.42 Social Engineering
    - T 5.104 Espionage
  - Threats
    - Employee Attack

Relations

Relations for: T 5.42 Social Engineering

Relation	Title
affects	Confidential Information
relevant threat	Employee Attack
likelihood modifi	S 3.5 Training on security
modelled by	E1 Employee

Figure 5: Example of models created with Verinice

Even though Verinice internal data structures are not visible, we can distinguish two modes of operation. In the ”BSI Model” we can map scenarios from the German IT baseline catalog (BSI) [9] to our business structure. This model is especially useful to find defenses that are commonly used. The insights gained in this mode can then be used to generate

”to do” lists to aid the implementation of those measures.

In figure 5 on the left hand side, we can see how that model looks in practice. The company is decomposed into common attack vectors like ”Buildings”, ”Network Connections” and ”Staff”. Each of that categories is then refined into individual instances like ”Headquarters”, which contain measures to be taken to ensure their safety. In this case, the entries ”5.1.2 [A] Regulations governing access to distributors” and ”5.1.10 [Z] Safe doors and windows” from the BSI catalog are chosen. Optionally, own company specific defense could be added. Similarly, ”Network Connections” and ”Staff” is further refined with specific instances and mitigation strategies.

The second mode of operation, the ”Information Security Model” (ISM) is used to model relationships between business entities like assets or employees, attack scenarios and defenses. It can then be used to calculate detailed reports about the business security state, including risk possibilities and costs in case of attacks. To create reports close to reality, Verinice supports a wide selection of attributes for each of the entities, as well as many different kinds of relationships.

To get an idea of the ISM, figure 5 shows an example on the right hand side. The ”AnalysedCompany” contains the asset ”Confidential Information” and the employee ”E1 Employee”. In addition, it implements the controls ”T 5.3.5 Training on security safeguard” and ”T 5.3.30 Selection of employees” from the BSI catalog. Furthermore, the scenarios ”T 5.42 Social Engineering” and ”T 5.104 Espionage” are possible. These entities are connected by using relationships. One example are the relationships related to the social engineering scenario, as shown in the bottom right edge of figure 5. There, the malicious social engineering scenario is shown to be one member of the relevant thread ”employee attack”. It is modeled by the person ”E1 Employee” and directly affects the asset ”Confidential Information”. Besides the attack side, there also is a defense. As shown in the relationships panel the likelihood of this kind of attack is reduced by the control measure ”Training on security safeguards”. Similarly, the scenario espionage is modeled.

3. COMPARISON

After we briefly introduced common security modeling tools, this section compares and evaluates their usefulness in answering practical questions.

In general, Verinice is a practical tool build for use in industry, while the other models are developed from a more academic perspective. This becomes even more obvious considering the use cases: On one hand, Verinice is well-suited for aiding a companies general decision making in instances like asset management, where the value of an asset and the cost of its protection need to be compared to each other. On the other hand it can actually suggest measures to ensure the security of systems and devices. In many situations, this can be done by employees with limited training, as many steps of the vulnerability and defense finding process are automated by external tools or simplified by Verinice’s databases.

On the contrary, the UML-based modeling tools and ADTs are better-suited to gain insight into the attack vectors of possibly new systems. They provide the means to structurally examine systems on different levels and visualize their components and their weaknesses. The resulting abstract models can then be used to find possible defenses or evaluate the efficiency of known countermeasures. It is therefore mainly used in academia.

To provide more detailed insight, the following subsections compare the tools in their capability to solve practical problems.

3.1 Discovering attack vectors and defenses

One of the main use cases of security modeling tools is to map a real word situation and its defenses in order to find uncovered or not sufficiently protected components.

ADTs are very suitable for this purpose, as they enable a user to start modeling the system at a very generic degree, which can then be refined to an arbitrary level of detail. Each of the resulting subcomponents can then be evaluated by their current state of protection, leading to quite a complete analysis of the system.

UML on the other hand does not offer multiple levels of detail in one model. Instead, a user first needs to select the scope of his analysis. If he wants to keep an overview over the full system and possible loopholes on the system boundary, MCD are most suited. Close views of single components or processes can be modeled with MADs and ESMs. While the former is more useful to model interactive processes involving multiple actors, the latter provides stronger focus on the process itself. Even though MADs and ESMs could be used to discover exploitable components, they are more suited to develop defenses, once the possible attacks are known. To fully analyze a system, a combination of multiple of above types is often needed, making the approach less structured than using ADTs.

Verinice faces the problem that it provides almost no visual representation of relationships between attacks, defenses and system components. As the tree like structure and a word-based search are the only ways of navigating the provided data, manual attack discovery is complex and important links are easily overlooked. The main strengths of Verinice in that area are therefore its features as a practical software tool. For instance, it enables a user to directly import data automatically generated by vulnerability scans on the network or the connected hosts. Besides the ability to find loopholes by itself, Verinice offers the functionality of importing databases of common scenarios. The data can either be used to automatically add suitable defenses and relationships to the vulnerabilities discovered before, or can lead the user through a catalog of commonly found attack vectors, who can then add them and their countermeasures to his security model. The databases are based on the German baseline security catalog [9] or the ISO 27000 standards [15], adding the general advantages of standardized procedures. Besides that, especially for non-standard systems, non-IT components or very process-based situations, above features can not be utilized. Therefore, Verinice is only partly suited for attack discovery.

3.2 Dealing with failing defenses

Due to their static nature, one central question is how well the presented tools can model measures after defenses fail.

With ADTs, the closest such feature are multiple disjunctive child defense nodes. If one of these nodes fail, the parent system is still protected by the other remaining defense nodes. Dynamic processes of the form ”if defense A fails, activate measure B” can not be expressed, thus significantly limiting ADTs capabilities for that kind of situation. One proposal for an extension of attack trees to include this feature are Boolean logic Driven Marcov Processes [22].

As opposed to ADTs single-model approach, UMLs multi-layered modeling is able to better capture such situations. Like ADTs, MCDs static nature make them unsuitable for that task. Instead, MADs and ESMs can be used to add more detail to selected processes. While activity-diagrams and the presented extensions support conditional nodes to choose different paths, ESMs can be used to express parting ways by drawing multiple outgoing edges for an activity.

Verinice on the other hand does not support any kind of event sequences, rendering its model more limited than UML. It makes up for that shortcoming with its feature-richness regarding different relationships and accurate metrics. While the user can specify an attack's impact on availability, confidentiality, integrity and overall value of an asset, defenses can be modeled to reduce the weight of such consequences. Alternatively, they can be altered to decrease the likelihood of a specific attack's success. As a result, Verinice can approximate an attack's probability and simulate the business state after its success, including the expected usefulness of other relevant defenses. It is therefore still superior to ADTs in evaluating consequences of a failing defense.

3.3 Prioritizing defenses and dependencies

In practice it is often desirable to rank priorities. This is useful if decisions regarding the implementation of a defense need to be made and factors like cost, usefulness or impact on business operation are compared.

ADTs do not provide direct means of ranking of any kind. The only way to model an order is by using the refinement feature, which can be used to express the dependency of a parent node on the success of its child nodes. Priorities between direct children of a node are not possible. For attack trees, Baca and Peterson proposed a notion using integer annotations to rank the attacks- and countermeasures mitigation impact [3], which might be transferable to ADTs.

While the presented extension to UML do not directly support ranking, the UML base notation does. By using UML constraints, any kind of dependency can be informally annotated, including notes on prioritization with any of the above factors. As these notations do not carry any formal meaning, their use is limited to aid a discussion of an attack's or defense's suitability, but are useless for any kind of automatic mathematical processing or proofs.

As already mentioned in ”Dealing with failing Defenses”, Verinice supports weighting of assets, relationships, attacks and defenses. Thus, limited prioritization within the boundaries of Verinices supported metrics is possible. In this case, automatic processing and report generation from those values is supported, making it partly superior to UML.

Table 1: Strengths and weaknesses of the presented tools

Model /Tool	Strenghts	Weaknesses
Attack de-fense tree	• Different levels of detail in one model • Mathematically thoroughly defined	• Limited to static processes • Unprioritized defenses
UML	• Widely used and easily learnable • Allows for dynamic process modeling • Multiple models for different levels of detail	• Often requires multiple models for complete analysis • Mostly useful for visualization, no mathematical foundation • Only informally prioritized defenses
Verinice	• Automatic analysis via external tools • Access to standardized scenario catalogs • Report generation including accurate metrics • Prioritizing of defenses possible	• Steep learning curve • Time consuming for non-standard situations • Almost no visualization of relationships

3.4 Strengths and weaknesses

To complete the comparison, a brief overview of the presented tools is shown in table 1. The row labeled UML contains information on the collective use of misuse cases, mal-activity diagrams and extended statechart diagrams.

4. RELATED WORK AND FURTHER READING

ADTs base their ideas on other concepts, including fault trees [29], threat trees [2, 30] and attack trees [24]. The latter was then extended by various researchers as summarized by Piètre-Cambacédès and Bouissou [22]. Many approaches extend the features and semantics of ADTs. In a theoretical context, the relation between ADTs and game theory has been analyzed [17]. ADTs computational aspects have been studied in [19], where semantics based on De Morgan lattices were used. Furthermore, ADTs quantity capabilities were evaluated in a practical case study [4].

In addition to the UML models presented in this paper, other approaches were proposed to enable secure software engineering, including UMLSec [16] and SecureUML [21]. As a contrast to MADs, there was an attempt to directly model secure business processes, concentrating more on modeling a process after security problems and countermeasures have been found [23]. Similar to MCDs and MADs, abuse frames are another example of an approach to system exploitation from the perspective of an attacker [20]. Misuse cases have also received a number of extensions, like specialization and generalization support [27]. Beside the theoretical perspective on UML, there has also been various tests on UML in realistic scenarios [1, 7, 12].

As Verinice is a practical tool, academic literature beyond its short mention is limited. Consequently, information can be found concerning its data sources ISO-27000 and the BSI baseline catalog [5, 6, 13].

5. CONCLUSION

In general, there is no tool to fit every situation. Verinice has strengths in its automatization capabilities and the high accuracy of its generated output, but it can only be used to model certain standard scenarios. UML on the other hand provides a high degree of freedom and flexibility while being easy to learn, but lacks a rigid formal foundation. ADTs form a compromise: While unifying multiple levels of accuracy in one model, they still provide a formal structure to work with. In practice, a combination of above tools may be employed. ADTs could be used as an overview to find attack vectors in top-level components, while detailed vulnerability discovery on process level might be done with MCDs or MADs. The resulting information could then be fed into Verinice to calculate actual attack probabilities and cost impact. Either way the presented models form a good base to thoroughly analyze a system to identify and later evaluate possible defenses, making risk management easier and general system security better.

6. REFERENCES

[1] I. Alexander. Initial industrial experience of misuse cases in trade-off analysis. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 61–68. IEEE, 2002.

[2] E. G. Amoroso. *Fundamentals of computer security technology.* PTR Prentice Hall New Jersy, 1994.

[3] D. Baca and K. Petersen. Prioritizing countermeasures through the countermeasure method for software security (cm-sec). In *International Conference on Product Focused Software Process Improvement*, pages 176–190. Springer, 2010.

[4] A. Bagnato, B. Kordy, P. H. Meland, and P. Schweitzer. Attribute decoration of attack–defense trees. *International Journal of Secure Software Engineering (IJSSE)*, 3(2):1–35, 2012.

[5] K. Beckers, H. Schmidt, J.-C. Kuster, and S. Faßbender. Pattern-based support for context establishment and asset identification of the iso 27000 in the field of cloud computing. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 327–333. IEEE, 2011.

[6] I. BSI. Baseline protection manual. 2000.

[7] F. den Braber, T. Dimitrakos, B. A. Gran, K. Stølen, and J. Ø. Aagedal. Model-based risk management using uml and up. *Issues and Trends of Information Technology Management in Contemporary Organizations*, pages 515–543, 2002.

[8] M. El-Attar, H. Luqman, P. Kárpáti, G. Sindre, and A. L. Opdahl. Extending the uml statecharts notation to model security aspects. *IEEE Transactions on Software Engineering*, 41(7):661–690, July 2015.

[9] B. für Sicherheit in der Informationstechnik. Bsi-grundschutz katalog. Available at `http://www.bsi.de/gshb/deutsch/index.htm` (Aug. 2018), 1996.

[10] S. GmbH. Verinice website. Available at `https://verinice.com/` (Aug. 2018).

[11] Heise. Ransomware wannacry befällt rechner der deutschen bahn. Available at `https://www.heise.de/newsticker/meldung/Ransomware-WannaCry-befaellt-Rechner-der-Deutschen-Bahn-3713426.html` (Jun. 2018).

[12] S. H. Houmb, F. Den Braber, M. S. Lund, and K. Stølen. Towards a uml profile for model-based risk assessment. In *Critical systems development with UML-Proceedings of the UML'02 workshop*, pages 79–91, 2002.

[13] E. Humphreys. *Implementing the ISO/IEC 27001 information security management system standard.* Artech House, Inc., 2007.

[14] P. Institute. 2017 cost of data breach study: Global overview. Available at `https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=SEL03130WWEN` (Aug. 2018).

[15] Information technology – Security techniques – Information security management systems – Overview and vocabulary. Standard, International Organization for Standardization, Geneva, CH, Feb. 2018.

[16] J. Jürjens. *Secure systems development with UML.* Springer Science & Business Media, 2005.

[17] B. Kordy, S. Mauw, M. Melissen, and P. Schweitzer. Attack–defense trees and two-player binary zero-sum extensive form games are equivalent. In *International Conference on Decision and Game Theory for Security*, pages 245–256. Springer, 2010.

[18] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. Attack–defense trees. *Journal of Logic and Computation*, 24(1):55–87, 2014.

[19] B. Kordy, M. Pouly, and P. Schweitzer. Computational aspects of attack–defense trees. In *Security and Intelligent Information Systems*, pages 103–116. Springer, 2012.

[20] L. Lin, B. Nuseibeh, D. Ince, and M. Jackson. Using abuse frames to bound the scope of security problems. In *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pages 354–355. IEEE, 2004.

[21] T. Lodderstedt, D. Basin, and J. Doser. Secureuml: A uml-based modeling language for model-driven security. In *International Conference on the Unified Modeling Language*, pages 426–441. Springer, 2002.

[22] L. Piètre-Cambacédès and M. Bouissou. Beyond attack trees: Dynamic security modeling with boolean logic driven markov processes (bdmp). In *2010 European Dependable Computing Conference*, pages 199–208, April 2010.

[23] A. Rodríguez, E. Fernández-Medina, and M. Piattini. Capturing security requirements in business processes through a uml 2.0 activity diagrams profile. In *International Conference on Conceptual Modeling*, pages 32–42. Springer, 2006.

[24] B. Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[25] G. Sindre. Mal-activity diagrams for capturing attacks on business processes. In P. Sawyer, B. Paech, and P. Heymans, editors, *Requirements Engineering: Foundation for Software Quality*, pages 355–366, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[26] G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, Jan 2005.

[27] G. Sindre, A. L. Opdahl, and G. F. Brevik. Generalization/specialization as a structuring mechanism for misuse cases. In *Proceedings of the 2nd symposium on requirements engineering for information security (SREIS'02), Raleigh, North Carolina*, 2002.

[28] F. times. Leaked cia cyber tricks may make us wannacry some more. Available at `https://www.ft.com/content/a7a6c91c-3a35-11e7-ac89-b01cc67cfeec` (Jun. 2018).

[29] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. Fault tree handbook. Technical report, Nuclear Regulatory Commission Washington DC, 1981.

[30] J. D. Weiss. A system security engineering process. In *Proceedings of the 14th National Computer Security Conference*, volume 249, pages 572–581, 1991.