AADT: Asset-Driven Attack-Defense Tree

Nan Messe[1][a] and Avi Shaked[2][b]
[1]*IRIT, CNRS, UT2, France*
[2]*Department of Computer Science, University of Oxford, Oxford, OX1 3QD, U.K.*

Keywords: Attack Tree, Risk Management, Security Modelling and Assessment, Usable Security.

Abstract: Attack trees are widely used in threat modelling and risk analysis to describe threats and identify attacks that realize them. Their extension, attack-defense trees, allows for depicting interactions between attackers and defenders. However, current graphical representations of attack(-defense) trees focus primarily on threat scenarios and do not account for the representation of domain elements and their hierarchical organization. Compromised domain elements (e.g., systems, sub-systems, components, etc.) are thus not directly highlighted in any of these tree representations, which requires additional effort from decision-makers during impact analysis. To help make impact analysis more explicit and enable stakeholders to assign and evaluate security controls more effectively, we propose a novel methodology for graphical secure system modelling and assessment, the "Asset-Driven Attack-Defense Tree" (AADT). AADT is an extension of the attack-defense tree, combining the security and system views for seamless secure system development. AADT's main contribution lies in bridging the system and security views while representing domain elements, associated vulnerabilities, and security controls at different levels of abstraction, which is aligned with the system development lifecycle. This layered representation is especially useful in the fast-evolving cyber threat landscape, where diverse attack techniques often exploit similar vulnerabilities. By associating vulnerability categories with domain elements and proposing high-level security controls, AADT helps stakeholders manage a broad spectrum of attacks targeting similar vulnerabilities, thus enabling a more proactive and structured approach to secure system development. We also present a formalism for AADT and illustrate the AADT methodology using a simple, real-world scenario based on the existing security body of knowledge.

1 INTRODUCTION

Establishing the security of systems is a dynamic and ongoing effort. The number of cyberattacks grows, the attackers are active and attack techniques are sophisticated and diverse. This is exacerbated in times of stressful, real-world events, such as the war in Ukraine and the COVID-19 pandemic (Ramadan et al., 2021). The constant stream of data breaches and harmful cyberattacks has demonstrated that attack intensity and sophistication will only increase in the future. Attack tree (Schneier, 1999) is a conceptual representation that is widely used for securing systems. It provides a systematic way of characterizing diverse system threats and attack scenarios (Gadyatskaya and Trujillo-Rasua, 2017).

Typically, an attack tree design activity necessitates both system domain expertise and security expertise, with the former providing in-depth knowledge of the system and the latter offering security knowledge and experience (Gadyatskaya and Trujillo-Rasua, 2017). However, current graphical representations of an attack tree focus primarily on threat scenarios. They do not account for the representation of domain elements and their hierarchical organization. In addition, the standard approach for attack trees, in which only leaf nodes are annotated with values, can be restrictive, as information for intermediate nodes are often more readily available than that of the leaf nodes (Gadyatskaya and Trujillo-Rasua, 2017).

Attack-defense tree is an extension of attack tree (Kordy et al., 2014). It allows nodes that represent defensive measures to appear as leaf nodes of the tree. This is useful for depicting interactions between attackers and defenders. However, how can one be certain that a defensive measure implemented today will still be effective in the future? How can newly

[a] https://orcid.org/0000-0002-3766-0710
[b] https://orcid.org/0000-0001-7976-1942

Messe, N. and Shaked, A.
AADT: Asset-Driven Attack-Defense Tree.
DOI: 10.5220/0013310100003928
In *Proceedings of the 20th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2025)*, pages 565-572
ISBN: 978-989-758-742-9; ISSN: 2184-4895

discovered attacks and defenses be documented in a systematic and efficient manner? More importantly, how can stakeholders assign and evaluate the effectiveness of security controls in system designs and threat scenarios?

To answer these questions, we propose an extension of attack-defense tree, termed Asset-Driven Attack-Defense Tree (AADT). This extension includes information about systems' assets and their vulnerabilities, in addition to attack activities and defensive measures. AADT highlights compromised and protected assets, making impact analysis more concrete for decision makers.

This paper is structured as follows. In Section 2 we review related work about attack tree, its common extensions and its integration with risk management. In Section 3, we introduce AADT, formalize AADT and illustrate its graphical representation. A case study is described in Section 4. In Section 5, we discuss the significance of our work and future work. Finally, we conclude this paper in Section 6.

2 RELATED WORK

In this section, related works about attack tree, its extensions and its integration with risk management are reviewed.

2.1 Attack Trees

Attack trees allow structuring diverse threat scenarios and reasoning about these scenarios at different levels of abstraction (Gadyatskaya and Trujillo-Rasua, 2017). It was originally proposed by Bruce Schneier in 1999 (Schneier, 1999). The security community uses attack trees to model threat scenarios in various domains, such as Automated Teller Machines (Fraile et al., 2016), Supervisory Control and Data Acquisition systems (Nielsen, 2011) (Ten et al., 2007) and Radio Frequency Identification systems (Bagnato et al., 2012).

In an attack tree, the final goal of an attacker is represented by the root of the tree, which is subsequently refined and decomposed into sub-goals captured by child nodes. The actions that can be directly executed by the attacker are represented by the leaf nodes.

Research in attack trees mainly focuses on improving the expressiveness of the formalism through new operators. Prominent examples include adding the sequential-AND operator (Jhawar et al., 2015) (Arnold et al., 2014), designing various quantitative assessment methods to prioritize risks (Bagnato et al., 2012) (Mauw and Oostdijk, 2006) (Buldas et al., 2006) (Kordy et al., 2012) (Gadyatskaya et al., 2016c) and proposing extensions of attack trees, such as the ones described in the following subsection.

Furthermore, new attack tree semantics are also being developed, and support the transformation of attack trees into other formalisms, such as timed automata (Kumar et al., 2015) (Gadyatskaya et al., 2016a), stochastic games (Aslanyan et al., 2016) and Markov chains (Jhawar et al., 2016). These formalisms offer advanced computational capabilities but they do not address the coordination of attack trees between security experts, system engineers, risk managers and other decision makers, which remains a gap.

2.2 Extending Attack Trees to Incorporate Defenses

Several attack tree extensions have been proposed in the literature, to introduce the defenders' viewpoint into attack trees. Prominent extensions include attack-defense tree (ADTree) (Kordy et al., 2014) (Fraile et al., 2016), attack response tree (ART), attack countermeasures tree (ACT) (Roy et al., 2012a) (Roy et al., 2012b) and attack protection tree (APT) (Ali and Gruska, 2019).

ADTree (Kordy et al., 2014) provides an intuitive and visually appealing representation of interactions between an attacker and a defender of a system. In ADTree, defense measures – alternately security controls – are only applied to leaf nodes. ACT (Roy et al., 2012a), on the other hand, extends the defense tree with countermeasures to mitigate attackers' actions, where defenses can be applied to any level of nodes in the tree. ART extends the attacker-defender interaction to find an optimal policy from a list of countermeasures. However, ART is not popular as it suffers from a state-space explosion problem (Roy, 2010). APT allows to define a set of protection actions at the point in which the attacker interacts with the system. It is worth noting that all above extensions do not include the domain aspect – i.e., the domain elements and their organizational structure – in the representation of the tree structure. This omission necessitates additional effort from system engineers when integrating the security aspect into the system development, potentially leading to increased complexity, higher risk of errors, and prolonged development times.

2.3 Integration with Risk Management

Research efforts have also been put into the integration of attack trees with security risk management methodologies (Pieters and Davarynejad, 2014) (Paul and Vignon-Davillier, 2014) (Gadyatskaya et al., 2016b). The security design process and risk management life cycle typically include the following main phases (Force and Initiative, 2013) (Messe, 2021):

- The identification of the critical assets to be protected;
- The analysis of vulnerabilities of the identified assets;
- The threats that may exploit these vulnerabilities;
- The prioritization of risks relating to threats, vulnerabilities and assets;
- The selection and implementation of the security controls that are needed to protect the identified assets;
- The assessment and the monitoring of security controls in order to verify that they are properly implemented.

Asset identification is fundamental for the success of secure system development, since its results directly influence the subsequent steps in risk management. The lack of support for asset identification in attack-tree-based methods hinders the ability of pertinent stakeholders, namely systems engineers, risk managers and decision makers to evaluate the potential impact of threats on systems. For example, TRADES Tool (Shaked, 2023) is a research informed open-source systems security design and assessment tool, supporting the security design process and risk management life cycle. While the tool has recently been extended to support the analysis of assets with respect to vulnerabilities (Shaked et al., 2024), it does not offer a graphical representation to communicate the design between stakeholders and support their decision making.

3 ASSET-DRIVEN ATTACK DEFENSE TREE (AADT)

In this section, we first define AADT terminologies and then formalize them. We then illustrate the graphical representation of AADT.

3.1 Terminology

A vulnerable asset is any asset that has vulnerabilities that can be exploited by attackers. The exploitable vulnerabilities make the vulnerable asset valuable to the attackers, who can use it to compromise a system. Previous work has shown that common types of vulnerable assets can be extracted from widely used security knowledge bases, such as CAPEC[1], CWE[2], CVE[3], OWASP[4], and ATT@CK[5] (Messe et al., 2020). Attack and mitigation information are also associated with those extracted vulnerable assets. Vulnerable assets extracted from these knowledge bases are independent of a specific domain and, thus, reusable. Vulnerable assets that are involved in domain modelling are annotated as vulnerable domain assets.

We define an Asset-Driven Attack-Defense Tree (AADT) as a node-labelled tree describing three information items: (i) the attack measures (strategies and actions) an attacker might employ to compromise a vulnerable domain asset; (ii) the defense measures (strategies and actions) that a defender can use to protect the vulnerable domain asset; and (iii) the vulnerable domain asset itself. Accordingly, an AADT has three types of nodes: attack nodes, defense nodes and vulnerable domain asset nodes. It is worth noting that non-vulnerable assets are not considered in this paper, as we deal exclusively with the security aspect.

The purpose of AADT is to model systems together with attack-defense scenarios. The first two key features of an AADT are the representation of refinement and the representation of decomposition. Each node may have one or more children that either represent the refinement of the parent node or represent its decomposition. A leaf-node is defined as a node that does not have any children.

For attack and defense nodes, the refinement of a node of an AADT is disjunctive. The goal of a disjunctively refined node is achieved when at least one of its children's goals is achieved. The decomposition of a node is conjunctive. The goal of a conjunctively decomposed node is achieved when all of its children's goals are achieved.

For vulnerable domain asset nodes, the refinement of node A into node B means that A is the type or generalization of B, or B is a specialization of A. The decomposition from node A into node B means that A contains B or B is a component of A.

Another key feature of an AADT is the representation of association, which aims at connecting different types of nodes. Each vulnerable domain asset node can be associated with attack nodes and defense

[1]https://capec.mitre.org/
[2]https://cwe.mitre.org/
[3]https://cve.mitre.org/
[4]https://owasp.org/
[5]https://attack.mitre.org/

nodes.

As explained shortly, when drawing an AADT, we depict vulnerable domain asset nodes by triangles, attack nodes by circles and defense nodes by rectangles, as shown in Figure 1. Refinement relations are indicated by solid directional lines between nodes, and decomposition relations are indicated by dotted directional lines. Association relations connecting different types of nodes are described by solid non-directional lines.

3.2 AADT Formalism

To formalize AADT, we denote an attacker by A and a defender by D. We represent the set of measures that can be employed by the attacker as β, and the set of defense measures as P. We denote by $L = \{s_1, s_2, ..., s_n\}$ the set of vulnerable domain asset nodes in the tree. The attack measures are defined on the attack nodes as a function $A : L \rightarrow 2^{\beta}$, and defense measures are associated with nodes to block the attack measures $D : L \rightarrow 2^{P}$. This expresses that an attacker tries to perform attack measure $b \in A(s)$ on a given vulnerable domain asset node s, and that there is a defense measure assigned by the defender $\alpha \in D(s)$ that will block the malicious activity and protect the vulnerable domain asset. For simplicity, we denote this interaction as $s^{\frac{\alpha}{b}}$.

For a vulnerable domain asset node s, if each $b \in A(s)$ is associated with a defense measure $\alpha \in D(s)$, then the vulnerable domain asset node is said to be protected. We denote this as $s \vdash (A, D)$. For example, consider a vulnerable domain asset "weak password" as the root node $s_0 \in L$. This asset is subject to the attack measure "password brute forcing attack" $b \in \beta$. A defense measure to protect the s against b can be "implementing a password throttling mechanism" $\alpha \in P$. The general syntax of AADT is generated by the following formula:

$$T = op(s_0(t(s_1), t(s_2), ..., t(s_n))) \quad (1)$$

Where T is a tree, $op = \{OR, AND\}$ is either the refinement relation (OR) or the decomposition relation (AND). Each node except leaf nodes is either AND or OR node. Each asset node can be associated with the two other types of nodes. s_0 is the root vulnerable domain asset node and $t(s_i)$ is a subtree (alternatively, an intermediate tree) with the root s_i. Based on this syntax, we formalize AADT as follows.

Definition 1. An Asset-Driven Attack-Defense Tree (AADT) is a tuple $T = (L, E, \beta, P, A, D)$, where L is the set of vulnerable domain asset nodes; E is the set of links between nodes (either refinement, decomposition or association); β, P are the disjoint attack measures and the defend measures, respectively; A, D represent the attacker – $A : L \rightarrow 2^{\beta}$ – and the defender – $D : L \rightarrow 2^{P}$ – respectively.

The vulnerable domain asset node $s_i \in L$ is said to be secure if $\forall b \in A(s_i), \exists \alpha \in D(s_i) : s_i^{\frac{\alpha}{b}}$. If for each s_i, we have $s_i \vdash (A, D)$, then this means that all vulnerable domain asset nodes are protected. We denote this as $S \vdash (A, D)$.

In the above example of the weak password vulnerable domain asset, the attack node "password brute forcing attack" can be refined by the child attack node "dictionary-based password attack" $b_1 \in \beta$. Defenders can perform protection measures such as "leveraging multifactor authentication" $\alpha_1 \in P$ to defend against this attack and protect the vulnerable asset "word-based password" $s_1 \in L$, which is a refinement of s_0.

3.3 Graphical Representation of AADT

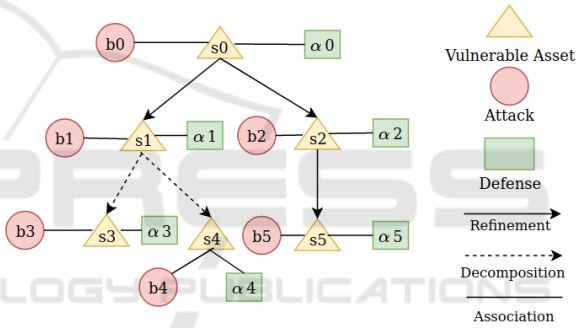

Figure 1: An example of Asset-Driven Attack Defense Tree.

Figure 1 illustrates the graphical representation of an AADT, with six vulnerable domain asset nodes and corresponding attack and defense nodes. For simplicity and readability, we only show one-to-one relations, knowing that one-to-many relations are also possible. s_3, s_4 and s_5 are leaf nodes. Equation 2 presents the corresponding AADT (T_1) construction that can capture three types of nodes. In T_1, s_0 is the root vulnerable domain asset node. It is refined by either s_1 or s_2, with the latter refined by s_5. The node s_1 is decomposed into s_3 and s_4. Each vulnerable domain asset node s_i is threatened by the attack node b_i and protected by the defense node α_i.

$$T_1 = s_0^{\frac{\alpha_0}{b_0}}(s_1^{\frac{\alpha_1}{b_1}}(s_3^{\frac{\alpha_3}{b_3}}, s_4^{\frac{\alpha_4}{b_4}}), s_2^{\frac{\alpha_2}{b_2}}(s_5^{\frac{\alpha_5}{b_5}})) \quad (2)$$

4 EXAMPLE ILLUSTRATION

To illustrate the features of AADT, we consider the following scenario concerning an anti-spam product, which can be a system's component. This scenario is shown in Figure 2.

To develop a system, a system engineer could require an anti-spam product as a component of the system. This anti-spam product should be modelled as an architectural element. In this scenario, the system engineer has chosen the product SpamTitan to play the role of the anti-spam product. More precisely, he/she has chosen the version 7.07 of SpamTitan. In an anti-spam product, such as SpamTitan, configuration files are available. One of the configuration files in SpamTitan 7.07 is snmpd.conf, which allows controlling the Net-SNMP agent's operation and the management information provided.

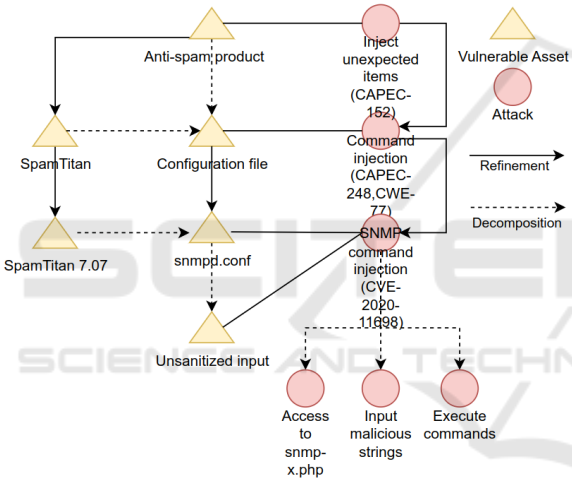

Figure 2: Using AADT to capture attack scenario for an anti-spam product.

From the attacker's point of view, there is a possibility of injecting unexpected items (CAPEC-152[6]) into a vulnerable anti-spam product, as shown in Figure 2. As an anti-spam product contains configuration files, the attacker could carry out command injection attack (CAPEC-248[7]). In the case of SpamTitan 7.07, which contains the configuration file snmpd.conf, a remote attacker could perform SNMP command injection in the file snmpd.conf through the page snmp-x.php, which alters the system's behavior from its intended operation[8]. For that, the attacker has to get access to snmp-x.php file, input strings that contain a malicious command and execute the command to obtain a privilege or capability that the attacker would not otherwise have. The success of this attack is based on the fact that snmpd.conf allows unsanitized user input constructing all or part of a command.

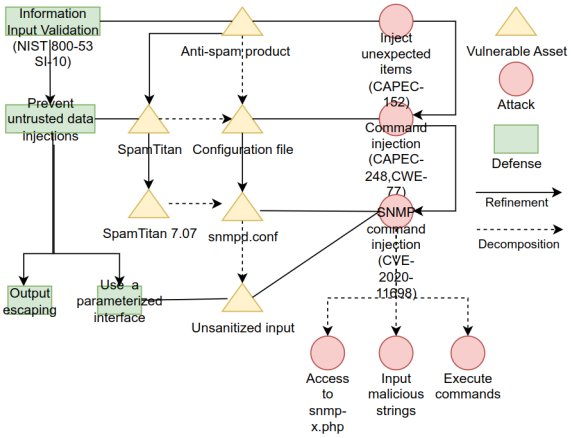

Figure 3: Using AADT to capture attack scenario and defense measures for an anti-spam product.

From the defender's point of view, to neutralize special elements that could modify the intended command, all input information should be validated (NIST SP 800-53 SI-10) (Group, 2020) and filtered for potentially unwanted characters. Untrusted data injections may be prevented using a parameterized interface or output escaping (also known as output encoding). This is shown in Figure 3. Parameterized interfaces keep data and code separate, ensuring that malicious or unintended data cannot alter the meaning of commands. Output escaping marks specific characters to signal the interpreter's parser about the trust level of the data.

5 DISCUSSION

Asset is an essential concept in threat modelling, as the attackers' aim is to compromise assets and the defenders' aim is to protect them. Until now, the concept of asset was absent in attack-trees-based security modelling. Here, we propose a novel extension – Asset-Driven Attack-Defense Tree (AADT) – which incorporates the fundamental Asset concept into the attack-trees-based security modelling. We demonstrated the novelty using a simple scenario. From our demonstration, it is clear that for any elaboration of the scenario, the specific AADT in Figure 3 could be easily extended with new system assets, attacks and defenses. Future research should provide a more in-depth evaluation of AADTs, exploring additional, more complex scenarios and applications.

[6]https://capec.mitre.org/data/definitions/152.html
[7]https://capec.mitre.org/data/definitions/248.html
[8]https://nvd.nist.gov/vuln/detail/CVE-2020-11698

As time passes, attack techniques become more and more diverse and defense measures have to evolve to follow the fast-changing pace of attackers. In comparison, assetsare relatively stable due to their domain-specific nature and could therefore be relevant underpinning elements (Uzunov and Fernandez, 2014; Messe et al., 2020). Adding and focusing on the vulnerable asset perspective as an anchor addresses the current limitations of the attack-defense tree, which concentrates on concrete actions without fully integrating domain elements. We plan to elaborate the AADT formalism with a notion of time as well as relate to information flows within the system.

In contrary to attack-defense tree, which annotates concrete attack and defense actions, AADT offers a novel, multi-level approach. This approach allows depicting nodes in different levels of abstraction. Using the refinement relation in AADT, it is possible to mix abstract assets with concrete ones. This allows to explicitly capture corresponding attack and defense categories as well as actions. This characteristic is particularly useful in architectural design, which can be performed in several system development stages and therefore involves domain elements in different levels of abstraction. Capturing asset/attack/defense categories promotes the ability to mitigate newly-discovered attack techniques of existing categories. For example, a newly-discovered attack about Poetry Argument Injection has been discovered[9], as its abstraction – Command Injection – has already been included in the current AADT in Figure 3. Accordingly, in our example scenario, the relevant defense measures were already incorporated into the design, and, therefore, the anti-spam product is secured against this specific attack. By organizing vulnerabilities into abstraction layers, AADT enables the deployment of adaptable, category-level defenses that remain effective as new threats emerge within existing vulnerability classes. This approach helps ensure that defensive measures implemented today maintain their relevance and robustness in the face of future, unanticipated attacks.

Like ART, ADTree and APT, AADT faces the state-space explosion problem, which hampers its comprehensibility. It is thus necessary to provide means for visualizing large AADT, such as the one proposed by the REsPASS project (Probst et al., 2015) (Lenin et al., 2016). By structuring the tree in a circular pattern, linearizing complex attack-defense scenarios, and allowing zooming in and out, the visualization can disguise the tree's complexity to reduce the cognitive effort needed to process a complex AADT. We intend to explore the adaptation of this visualization technique for AADT. We also plan to experiment with formal verification of AADTs, using a model checker (e.g. UPPAAL (Behrmann et al., 2006)).

Creating the AADT representations is labor intensive. We are working towards automating the generation of AADTs. For example, one of the concrete methods we are exploring is the automatic extraction of assets from the CAPEC attack pattern description, which is in natural language. Another direction we are currently attempting is automatically identifying vulnerabilities/attacks using the Common Platform Enumeration (CPE)[10] as an identifier for assets.

Future work can integrate the new AADT representation into security modelling tools. Specifically, we are considering extending TRADES Tool with AADT representations. TRADES Tool already supports the relevant concepts in its models, namely attack and defense measures, assets and relations between them. TRADES also supports extraction and association of CVEs with assets based on their CPEs. We intend to rely on those to automate the generation of AADT representations. This would also make sure that the derived AADTs are synchronized with the models that represent the systems' design and security posture. In addition, this would help to minimize the effort of creating AADTs.

6 CONCLUSION

In this paper, we introduce an extension of attack-defense trees, the Asset-Driven Attack-Defense Tree (AADT), which focuses on vulnerable asset information as a novel type of node. Unlike traditional attack-defense trees, which center on threat scenarios and defensive measures, AADT addresses the need for representing domain elements and their hierarchical structure. By explicitly incorporating vulnerable assets, such as systems, sub-systems, and components, AADT provides decision-makers with a cohesive view of system architecture and security, enhancing impact analysis and supporting the strategic design of security controls.

AADT also improves on traditional models by representing security controls and vulnerabilities at multiple levels of abstraction. This layered approach aligns with system development stages, allowing stakeholders to implement adaptable security controls that target both general and specific attack categories, such as high-level defenses against command

[9]https://www.cve.org/CVERecord?id=CVE-2022-36069

[10]https://cpe.mitre.org/

injection rather than focusing only on individual vulnerabilities.

Additionally, the explicit representation of compromised assets and associated vulnerabilities allows stakeholders to evaluate potential threats more effectively and assign security controls with minimal effort. By associating vulnerability categories with assets, AADT facilitates proactive, category-level mitigation strategies that can address future attacks within the same vulnerability class. We formalize AADT and demonstrate its unique strengths in refinement and impact analysis through a case study. Future integration of the AADT representation into security modelling tools can facilitate security experts' understanding of systems designs and security posture as well as their active role in designing trustworthy systems.

ACKNOWLEDGEMENTS

This work is supported by ICO, Institut Cybersécurité Occitanie, funded by Région Occitanie, France, and by Innovate UK, grant number 75243.

REFERENCES

Ali, A. T. and Gruska, D. P. (2019). Attack protection tree. In *CS&P*.

Arnold, F., Guck, D., Kumar, R., and Stoelinga, M. (2014). Sequential and parallel attack tree modelling. In *International Conference on Computer Safety, Reliability, and Security*, pages 291–299. Springer.

Aslanyan, Z., Nielson, F., and Parker, D. (2016). Quantitative verification and synthesis of attack-defence scenarios. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 105–119. IEEE.

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

Behrmann, G., David, A., Larsen, K. G., Håkansson, J., Pettersson, P., Yi, W., and Hendriks, M. (2006). Uppaal 4.0.

Buldas, A., Laud, P., Priisalu, J., Saarepera, M., and Willemson, J. (2006). Rational choice of security measures via multi-parameter attack trees. In *International Workshop on Critical Information Infrastructures Security*, pages 235–248. Springer.

Force, J. T. and Initiative, T. (2013). Security and privacy controls for federal information systems and organizations. *NIST Special Publication*, 800(53):8–13.

Fraile, M., Ford, M., Gadyatskaya, O., Kumar, R., Stoelinga, M., and Trujillo-Rasua, R. (2016). Using attack-defense trees to analyze threats and countermeasures in an atm: a case study. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 326–334. Springer.

Gadyatskaya, O., Hansen, R. R., Larsen, K. G., Legay, A., Olesen, M. C., and Poulsen, D. B. (2016a). Modelling attack-defense trees using timed automata. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 35–50. Springer.

Gadyatskaya, O., Harpes, C., Mauw, S., Muller, C., and Muller, S. (2016b). Bridging two worlds: reconciling practical risk assessment methodologies with theory of attack trees. In *International Workshop on Graphical Models for Security*, pages 80–93. Springer.

Gadyatskaya, O., Jhawar, R., Kordy, P., Lounis, K., Mauw, S., and Trujillo-Rasua, R. (2016c). Attack trees for practical security assessment: ranking of attack scenarios with adtool 2.0. In *International Conference on Quantitative Evaluation of Systems*, pages 159–162. Springer.

Gadyatskaya, O. and Trujillo-Rasua, R. (2017). New directions in attack tree research: catching up with industrial needs. In *International Workshop on Graphical Models for Security*, pages 115–126. Springer.

Group, J. T. F. T. I. I. W. (2020). Security and privacy controls for information systems and organizations. Technical Report NIST Special Publication (SP) 800-53, Rev. 5, Includes updates as of September, 2020, National Institute of Standards and Technology, Gaithersburg, MD.

Jhawar, R., Kordy, B., Mauw, S., Radomirović, S., and Trujillo-Rasua, R. (2015). Attack trees with sequential conjunction. In *IFIP International Information Security and Privacy Conference*, pages 339–353. Springer.

Jhawar, R., Lounis, K., and Mauw, S. (2016). A stochastic framework for quantitative analysis of attack-defense trees. In *International Workshop on Security and Trust Management*, pages 138–153. Springer.

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

Kordy, B., Mauw, S., and Schweitzer, P. (2012). Quantitative questions on attack–defense trees. In *International Conference on Information Security and Cryptology*, pages 49–64. Springer.

Kumar, R., Ruijters, E., and Stoelinga, M. (2015). Quantitative attack tree analysis via priced timed automata. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 156–171. Springer.

Lenin, A., Gadyatskaya, O., Ionita, D., Pieters, W., Tanner, A., Saraiva, S., Muller, C., Willemson, J., Ford, M., and Muller, S. (2016). Technology-supported risk estimation by predictive assessment of socio-technical security.

Mauw, S. and Oostdijk, M. (2006). Foundations of attack trees, information security and cryptology-icisc 2005. In *8th international conference, Seoul, Korea*, pages 186–198.

Messe, N. (2021). *Security by Design: An asset-based approach to bridge the gap between architects and security experts*. PhD thesis, Université de Bretagne Sud.

Messe, N., Chiprianov, V., Belloir, N., El-Hachem, J., Fleurquin, R., and Sadou, S. (2020). Asset-oriented threat modeling. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 491–501. IEEE.

Nielsen, J. R. (2011). Evaluating information assurance control effectiveness on an air force supervisory control and data acquisition (scada) system.

Paul, S. and Vignon-Davillier, R. (2014). Unifying traditional risk assessment approaches with attack trees. *Journal of Information Security and Applications*, 19(3):165–181.

Pieters, W. and Davarynejad, M. (2014). Calculating adversarial risk from attack trees: Control strength and probabilistic attackers. In *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*, pages 201–215. Springer.

Probst, C. W., Willemson, J., and Pieters, W. (2015). The attack navigator. In *International Workshop on Graphical Models for Security*, pages 1–17. Springer.

Ramadan, R. A., Aboshosha, B. W., Alshudukhi, J. S., Alzahrani, A. J., El-Sayed, A., and Dessouky, M. M. (2021). Cybersecurity and countermeasures at the time of pandemic. *Journal of Advanced Transportation*, 2021.

Roy, A. (2010). *Attack countermeasure trees: A non-state-space approach towards analyzing security and finding optimal countermeasure sets*. PhD thesis, Duke University.

Roy, A., Kim, D. S., and Trivedi, K. S. (2012a). Attack countermeasure trees (act): towards unifying the constructs of attack and defense trees. *Security and communication networks*, 5(8):929–943.

Roy, A., Kim, D. S., and Trivedi, K. S. (2012b). Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, pages 1–12. IEEE.

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

Shaked, A. (2023). A model-based methodology to support systems security design and assessment. *J. Ind. Inf. Integr.*, 33:100465.

Shaked, A., Messe, N., and Melham, T. (2024). Modelling tool extension for vulnerability management. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24)*, New York, NY, USA. ACM.

Ten, C.-W., Liu, C.-C., and Govindarasu, M. (2007). Vulnerability assessment of cybersecurity for scada systems using attack trees. In *2007 IEEE Power Engineering Society General Meeting*, pages 1–8. IEEE.

Uzunov, A. V. and Fernandez, E. B. (2014). An extensible pattern-based library and taxonomy of security threats for distributed systems. *Computer Standards & Interfaces*, 36(4):734–747.