Threat modeling at run time: the case for reflective and adaptive threat management (*NIER track*)

Dimitri Van Landuyt
imec-DistriNet, Department of Computer Science, KU Leuven
Heverlee, Belgium
dimitri.vanlanduyt@cs.kuleuven.be

Liliana Pasquale
School of Computer Science University College Dublin
Dublin, Ireland
liliana.pasquale@ucd.ie

Laurens Sion, Wouter Joosen
imec-DistriNet, Department of Computer Science, KU Leuven
Heverlee, Belgium
firstname.lastname@cs.kuleuven.be

***Abstract*—Threat modeling is an analysis activity aimed at eliciting viable and realistic security and privacy threats in the design of a software-intensive system. Threat modeling allows for a *by-design* approach, mitigating problems before they arise and avoiding later costly development efforts.**

However, it mainly pays off in software construction approaches that rely on planned architectures, in which sources of threats can be anticipated beforehand. These axiomatic assumptions are, however, increasingly untrue in contemporary software development practices in which software systems evolve drastically in later stages. In addition, software-intensive systems are increasingly faced with uncertainty in their operational contexts, and these are nearly impossible to enumerate in early development stages.

In this article, we first present the idea of *reflective threat modeling*, which involves the automated derivation of architectural system models from run-time and operational system artifacts, providing the threat modeler with an accurate and workable run-time inspection view of the system. We then outline and motivate the potential of adopting threat analysis models as a basis for holistic and adaptive threat management through integration of adaptive security and privacy technologies. This will enable systems to autonomously respond to emerging threats by dynamically activating dedicated controls or via run-time reconfiguration.

***Index Terms*—Threat modeling, threat analysis, threat management, security, privacy, run-time reflection, architecture-centric adaptation**

I. Introduction

Threat modeling [1], [2] involves the systematic analysis of potential security and privacy threats in the context of a specific system under design. It is considered an important cornerstone in the secure software development life-cycle (SDLC) [1], and it contributes to attaining the widely-advocated principles of *security* and *privacy by design*: instead of addressing problems and issues whenever they arise, a proactive and thorough assessment of security and privacy requirements in the early development stages will avoid costly, invasive engineering activities in later development stages.

Current threat modeling approaches [3], [4] are heavily shaped by their role in the development life-cycle: (i) they are to be performed manually, by analysts/requirement engineers in a workshop/brainstorm setting, (ii) they act upon an abstraction model of the system under analysis (such as Data Flow Diagrams (DFDs)) that is created in isolation and thus decoupled from implementation or run-time artifacts, (iii) they involve manual risk assessment which involves estimation of risk factors that both cannot be assessed precisely and quantitatively beforehand, and which rely on implicit information and background system knowledge[1], and (iv) if they also include support for selecting appropriate mitigations, threat modeling approaches mainly focus on design-level proactive mitigations such as architectural security or privacy design patterns [6], [7], design strategies and architectural tactics [8], [9].

While existing threat modeling approaches are suited for determining the most relevant security and privacy threats in an a-priori, constructive context, their disconnect from further software development activities and the operational context of software-intensive systems has a number of significant disadvantages. In practice, when threat modeling is conducted, it is done as a one-shot operation, and the analysis is rarely revised in later development stages as it is perceived too time-consuming and costly.

In this article, we provide an in-depth discussion about what is required to integrate threat modeling and threat analysis into the operational context of a system, in two incremental steps.

In a first step, we discuss the opportunities and challenges related to the automated construction of system models through reflection and inspection, an approach that we call '*reflective threat modeling*'. We envision the (semi-)automated derivation of integrated and rich input models in which different threat sources such as design flaws, vulnerabilities, and actual incidents can be distinguished and potential threats enumerated and assessed. These inputs models are architectural in the sense that they distinguish explicitly between system design structure, deployment, and run-time events and interactions.

In a second step, we discuss the possibilities and challenges related to implementing '*adaptive threat management*', in which the system itself performs automated threat analysis to identify emerging threats (e.g., when the system or its environmental context has drastically changed), and makes decisions

[1]For example, to assess the likelihood of a certain threat, the threat modeler must consider aspects such as attacker incentives and capabilities, the trust relations to third parties, the perceived strength of existing countermeasures, etc. In threat modeling practice, these elements and such rationale are kept implicit [5].

to proactively or reactively address threats, for example by employing adaptive security and privacy technologies [10]–[12] or enacting run-time system reconfigurations.

While approaches and technologies to adaptive security and risk assessment enjoy practical adoption [10], [13], they address specific threat types (e.g., focusing exclusively on fraud, anomaly or intrusion), and lack the holistic end-to-end system perspective that is inherent to threat modeling. As such, they are not suited for monitoring threat and risk evolutions across wider threat landscapes.

This article is structured as follows: Section II presents the necessary background, whereas Section III motivates this work. Next, Section IV and Section V respectively propose and discuss the research challenges of reflective threat modeling and adaptive threat modeling. Finally, Section VI concludes.

II. Background

This section provides background information on threat modeling, on the integration of security in a development and operations context (SecDevOps), and on adaptive security approaches and enabling technologies.

A. Threat modeling

Threat modeling approaches have emerged in the early 2000s [1] as a means to systematically identify and evaluate security requirements. The most prominent elicitation approach is Microsoft's STRIDE [2], yet different survey studies [3], [4], [14] show the range of available threat modeling approaches. Recently, the Threat Modeling Manifesto [15] was established to create a common understanding among practitioners and to articulate the core values and principles of threat modeling.

Figure 1 provides a generic overview of the different activities involved in threat modeling, starting from (activity 1) the establishment of an abstraction model of the system. Based on this, (activity 2) systematic threat elicitation approaches involve enumerating all theoretically viable threats, based upon reusable threat taxonomies, trees, vulnerability databases, or threat categories. As this step leads to large bodies of threats to be investigated, the subsequent step (activity 3) involves prioritizing and filtering, by determining which threats pose a significant risk to the system. For this, risk assessment models (e.g., FAIR [16]) can be applied [17] that take into account a wide array of risk components. Finally, for the most critical threats, countermeasures can be selected to mitigate the identified threats (activity 4).

Threat modeling has been successfully adopted in a wide variety of application domains, in cyber-physical systems (CPS) [18], automotive and mobility systems [19]–[22], large-scale enterprise environments [23]–[25], e-health systems [26] and even to validate the complex privacy-preserving architectures that are used in COVID-19 exposure tracking and contact tracing systems [27].

While DFDs [1], [2] are the most common system representations, there is in effect a wide range of system abstractions used in the context of threat modeling (e.g., Process Flow Diagrams [28], Information Flow Diagrams [27], DFD extensions [29]–[31]), or tool-specific representations [32]. For a more elaborate coverage of different secure design notations, we refer the reader to the literature [33].

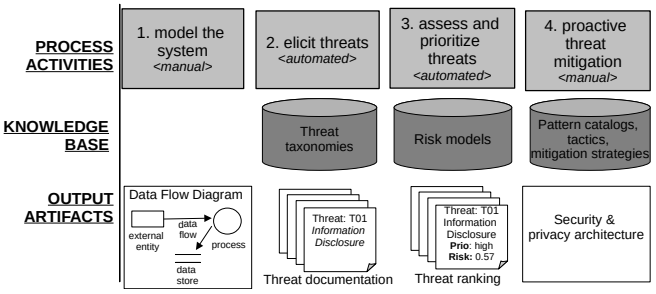

Fig. 1. The four core activities involved in threat modeling, with their inputs and outputs. Here depicted is a DFD-based approach which involves explicit threat documentation and risk-based prioritization.

Threat modeling is predominantly a manual task [15], [22] and in practice is used in the context of (i) educating and raising awareness about threats and common threat sources, (ii) increasing confidence in system concepts and early architectures, and (iii) performing fine-grained analysis at the level of DFD elements or interactions. Due to its heavy reliance on manual effort and the required degree of expertise, continuous threat modeling or frequent iteration of threat analysis activities are considered cost-inefficient [34].

Recently, advances have been made towards automation of threat modeling and threat analysis activities [3]. These efforts are focused on automated threat elicitation (activity 2) and automated threat prioritization and risk assessment (activity 3), e.g., as in SPARTA [30] and the work of Tuma et al. [29]. In addition, complementary attempts to bring threat modeling closer to the implementation phase have led to approaches such as pytm [35], with a code-based model, and threatspec [36], which allows developers to include annotations in code, from which system models can be derived for threat analysis.

B. SecDevOps

DevOps refers to the tight integration of development activities with the operational management of systems [37] and is associated with practices of agile development and continuous integration and deployment (CI/CD). These techniques involve extensive process automation, for example by embedding testing activities in production environments, quality assurance, automated and fine-grained deployment of individual updates or product features. By informing development activities with metrics and results obtained in a production or operational environment, problems can be identified and prioritized more accurately, and the time windows of iterative development can be shortened substantially.

Secured DevOps or SecDevOps [38], [39] involves dealing with novel cyber security risk that emerge in the DevOps context. In its strictest interpretation, it refers to addressing the security implications caused by automated and frequent deployment [40], for example by properly configuring the

target environment such as securing containers or virtual machines [41]. In other work, it is also used to refer to the integration of automated security development and operational practices into the actual CI/CD cycles, e.g., integration of automated security testing or scanning tools [42], [43].

C. Adaptive security and privacy

Approaches that are adaptive in nature are capable of identifying security- or privacy-related issues and reacting appropriately [10], [11], [11]–[13], [44]. Without aiming for exhaustiveness, the term refers to technology classes such as: adaptive firewalls [45], [46], adaptive incident or intrusion detection systems (IDS) [47], [48], adaptive fraud detection systems [49], [50], adaptive cryptography, and adaptive access control systems [51], [52].

For example, when an access request comes from outside of the corporate network or from a new device, an adaptive access control system can decide to require two-factor authentication, or when there is uncertainty about the credentials of a user, the adaptive system can decide to ask for extra credentials, such as biometric identifiers.

Other technology classes of relevance are not uniquely dedicated to security or privacy but represent key enablers for implementing and enacting of adaptations in a more generic sense. These include technologies such as virtualization, dynamic container orchestration [53], [54], software-defined networking (SDN), and policy enactment frameworks [55].

III. Motivating case: maritime system

Figure 2 presents a (strongly simplified) DFD of the system architecture of a maritime system, more specifically with emphasis on the communication subsystems deployed on a vessel used for navigation, management, and interaction with external systems (e.g., tracking of shipping containers).

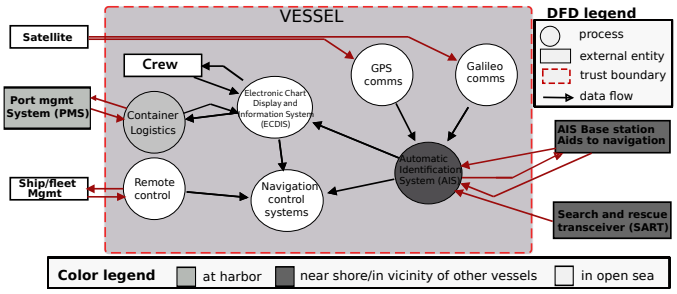

Fig. 2. A DFD of a maritime system, showing the subsystems deployed on a vessel for navigation, management, and interaction with external entities.

Adopting threat elicitation approaches at the basis of the depicted DFD will yield a number of possible and relevant system threats, for example the threat of a malicious agent deliberately steering the vessel off course by tampering with the satellite communications or by spoofing the on-shore management systems to which the vessel remains in contact.

However, the different threats and their risk profiles (likelihood and impact) will in reality depend strongly on the operational context of the system. For example, the attack surface that involves access via Port Management Systems (e.g., the Information Disclosure threat that involves a third party that wants to identify specific containers in the cargo in, for example, smuggling scenarios) only becomes relevant when the vessel is docked, whereas such threats are not applicable when the vessel is in open sea. Conversely, the threat of a malignant external entity falsely emitting distress signals via the Search and Rescue Transmission Systems (SART) to deviate the vessel (a spoofing threat) will not be applicable when the vessel is docked.

When performing threat analysis in an a-priori design context, the analyst is required to enumerate and anticipate the different operational contexts of the system, and as these contexts may change significantly over time (as in the motivating example), so will the corresponding threat landscapes. This is especially problematic in systems for which exhaustive enumeration of these operational context is cumbersome or even infeasible, and as a consequence, relevant threats may not be identified, prioritized nor mitigated correctly.

In addition, when the system itself changes or evolves over time, novel threats will not be identified nor mitigated unless the entire threat analysis exercise is repeated. Change scenarios are: new subsystems are activated that were not initially present in the analysis (e.g., the installation of on-deck wireless networks to enable personal communication of crew members), when new external entities emerge (e.g., new means to communicate to base stations, or new types of base stations), or simply, as the capabilities of attackers evolve over type (e.g., new vulnerabilities have been published or new types of attacks have been detected) so that threat types that were previously considered impossible have become actual issues.

IV. Reflective threat modeling

In this section, we motivate and outline a first necessary innovation in threat modeling related to how the system models that are used as the basis for threat analysis are obtained. Our motivation is based on two key problems with the current state of the art: (i) system models used in threat modeling lack expressiveness, (ii) system models created in early stages only convey design information which can become outdated once the system is in operation. We elaborate on both problems below.

a) Lacking expressiveness in system models: Data flow (DFDs) modeling involves a relatively simple and intuitive notation consisting of only five element types (*processes*, *data stores*, *external entities*, *data flows* and *trust boundaries*). The simplicity ensures easy adoption by diverse stakeholders and as such facilitates the creation of DFDs in workshop settings. The downside however is that it leads to system models that easily omit relevant information, e.g., DFD models typically model data flows but omit how exchanged information is encoded or which communication protocols are used. In addition, it leads to overloading of element types in terms of the type of system information that is conveyed. For example, a trust boundary may represent an organizational boundary, a communication boundary, a security control boundary, or a

physical boundary, but it may also be used to delineate a unit of computation (a system component), or a logical grouping of elements (e.g., to represent system elements deployed in a virtualized environment such as a distributed cloud).

The correct interpretation of DFD elements[2] as such depends heavily on human disambiguation.

b) Low accuracy of system models: When threat modeling is performed in the early stages of requirements engineering, the system model is *prescriptive* in nature, i.e. it is indicative of how the system is expected to be materialized, but in practice, such a holistic view on the system is incomplete in these stages, especially in contemporary development practices that involve drastic and continuous evolution. When relying exclusively on early models, the final product often deviates (or has evolved drastically away) from the planned system.

A. Concept

In the core concept of *reflective threat modeling*, we advocate extensively relying on methods and techniques that involve automated derivation of more expressive system models that leverage this information to create and maintain accurate and rich system models.

These system models are architectural in nature, in the sense that they encode information of distinct architectural viewpoints. We adopt the classification in viewpoints of Bass et al. [8] in terms of (1) a module view, (2) an allocation or deployment view, and (3) a process or client-server view, which we consider a valuable distinction to differentiate between design flaws, physical and infrastructural attacks, and run-time incidents respectively.

- **Module views** represent the design and development structure of a system. Contemporary practice is increasingly based on expressive models that encode a wealth of information about design structure and remain accessible at run time. Some examples are: data model or schema specifications, workflow descriptions or business process specifications. These can be complemented with techniques of static code analysis (e.g., dependency analysis) and the dedicated annotation-based approaches discussed above [35], [36]. Finally, methods of software architecture reconstruction (SAR) [56], [57] also take into account other development artifacts, such as code comments, information drawn from versioning systems, etc. to retroactively derive the development structure of the system.
- **Deployment and allocation views**: Distributed systems commonly rely on middleware platforms that require developers to define deployment abstractions (e.g., applications, (micro-)services), and this is the level of granularity at which distributed applications are deployed. This information is available through reflection and via platform-specific deployment descriptors. In addition, virtualization and orchestration systems [58], [59] employ techniques such as Software-defined networking (SDN) [60], [61] and involve expressive deployment descriptors from which valuable deployment information can be obtained.
- **Client-server/process views** represent run-time interactions, for example between active objects, threads and processes. These views contribute information about frequency of interactions, provide concrete examples of data elements being processed, run-time user sessions, etc. The construction of these views can leverage upon information provided in audit trails, system logs and session management, but also may rely on dynamic analysis of application execution (e.g., based on call graphs to identify run-time interactions, or taint analysis [62]). In addition, intrusion or fraud detection systems are capable of detecting security incidents in an operational system, and these represent threat instances that may trigger system-wide threat re-evaluation.

B. Research agenda

We define an approach that leverages the aforementioned sources of architectural information to construct or refine system models as a *reflective threat modeling* approach and discuss the required innovations and main research challenges.

- **Architecture description languages** We argue that data-flow-centric abstractions (DFDs) lack the expressiveness to document and maintain the architectural distinction between (i) design structure, (ii) run-time interactions, and (iii) deployment configurations. As such, we envision adopting or tailoring existing software architecture specification or description languages [63], [64] for the specific purpose of threat analysis, as these allow defining a system from multiple orthogonal and complementary viewpoints [65]. A first problem inherent to adopting more expressive architectural models is related to internal consistency, i.e. ensuring that different views on the system remain consistent [66].
- **Heterogeneity of architectural information sources.** The potential information sources listed above are heterogeneous in terms of (i) the type of architectural information provided, (ii) their technology context, (iii) how accessible they will be through run-time reflection and inspection. To deal with heterogeneity, we envision a reflection framework that is extensible and capable of integrating these different sources into a comprehensive and complete system model. Also required are facilities to assess the extent to which automatically-generated system models can be considered complete or suited for the purpose of threat analysis.
- **Derivation logic.** Casting the obtained system information obtained via system inspection and reflection into abstractions suited for threat analysis is not straightforward, not in the least because these target abstractions are not yet well-known or -understood. From a technical perspective, dedicated translation mechanisms and algorithms will be required to raise the level of abstraction or granularity, e.g., by employing clustering algorithms in call or execution graphs obtained through system inspection to identify more coarse grained 'components' or 'processes'. Model-to-model transformation techniques [67] will be required to select and convert the relevant system information encoded

[2]While we focus predominantly on the DFD notation in the above for the sake of simplicity in the argumentation, the same applies to different design notations such as those discussed in Section II-A.

in descriptors and domain-specific languages into the target abstractions. Although these mechanisms will be heterogeneous in nature, they should all contribute to a coherent system model that serves as an appropriate input for system-centric threat analysis.

- **Systemic impact of reflection and inspection.** Continuous inspection and reflection is costly in terms of system performance and the operational disruption of extensive monitoring and inspection systems becomes prohibitive [68]. As such, we argue that an approach that involves gradual and event-based co-evolution of the reflection system model (a digital twin) in parallel with the operational distributed system will be most realistically feasible. However, propagating run-time events and changes between a run-time system and its model-based representation from a variety of sources (run-time state and interactions, versioning systems, software updates, etc.), is a non-trivial problem, for three main reasons: (i) divergence has to be avoided between the run-time state of the system and the model, (ii) the constraints of large-scale distributed systems such as partition tolerance and lack of global time synchronization have to be taken into account, (iii) when the distance between the source abstractions and targets abstractions is substantial [68] translations become complex, difficult to engineer, and verify.

V. A HOLISTIC ADAPTIVE THREAT MANAGEMENT PERSPECTIVE

Traditional threat modeling approaches are mainly used in a constructive, by-design context, and these approaches support design-centric decision-making in the sense that they offer guidance towards addressing identified threats in design or development. For example, Shostack advocates tight integration with security testing and validation approaches and emphasizes overall quality assurance [2], whereas LINDDUN [5] provides taxonomies of architectural patterns and strategies for privacy.

These are less relevant in an operational context, as it is impossible to switch between architectural tactics or design strategies at run time. As such, the key challenge is to mitigate emerging threats by instrumenting and dynamically adapting operational systems.

Security or privacy mitigations always incur a cost and involve essential trade-offs with other system concerns. For example, extensive authentication reduces the usability of a system and an approach to encrypt all communications will incur higher resource consumption (bandwidth, processing power) and increase architectural complexity (e.g., for cryptographic key management). Instead of incurring such costs by default (e.g., enforcing multi-factor authentication by default), adaptive security approaches and techniques have the intrinsic capability to change according to the system's needs.

For example, when there is uncertainty about the credentials of a user, the system could dynamically decide to ask for extra credentials (adaptive access control). Or, when the system detects that it is near untrusted devices and thus its attack surface increases, it could dynamically reconfigure its security controls. For example, the maritime system could switch to stronger encryption mechanisms for communication when docked in a port but use weaker mechanisms at plain sea (through adaptive cryptography).

A. Concept

We envision an approach in which the threat analysis model becomes the cornerstone of a threat management framework which is capable of installing, activating, and reconfiguring a wide range of adaptive security technologies, such as those described in Section II-C, a concept that we refer to as '*adaptive threat management*'.

As the main advantage, such a holistic analysis framework will maintain an end-to-end system perspective in terms of the different threats that may emerge over time, and it can reason about and decide upon mitigations and reconfigurations in terms of their expected implications on the entire threat landscape. These mitigations may both be preventive and reactive in nature.

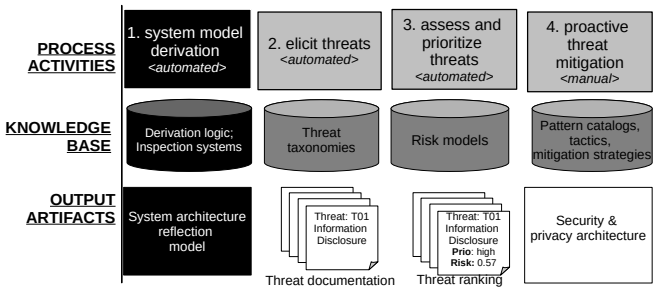

Fig. 3. Graphical depiction of the proposed *adaptive threat modeling* approach, in which the threat model is used to embed and steer system-wide adaptations of adaptive security techniques, by dynamically activating or tuning such approaches (required innovations highlighted in black).

B. Research agenda

The proposed approach of *adaptive threat management* builds upon the vision of automated reflective threat modeling presented in Section IV, in which the threat model itself acts as the analysis model from which reconfiguration actions can be motivated:

- **Engineering a base platform.** Approaches to determine common framework requirements for the purpose of establishing a system base architecture that integrates a sufficient set of adaptive security and privacy controls to enable mitigation of the most plausible threats that will emerge in the context of the system are required.
- **System model.** An additional challenge is related to finding a representation of these adaptive technologies in the system model, in such a way that the model can be used both for analysis (e.g., for *what-if* analysis to estimate the impact of potential reconfiguration actions) and to steer and orchestrate actual reconfigurations.
- **Risk events:** Recognition of conditions in which risk significantly changes, at the basis of observed run-time events (e.g., vicinity of new entities, new data types being

processed, ...) requires new threat taxonomies (in which a threat type not only refers to a design flaw or a design weakness but also to vulnerabilities and incidents) and novel risk models and dedicated approaches that act at the level of these run-time events.
- **Technical integration:** Allowing the threat management framework to activate or tune specific security controls in response to identified threat and problems raised by analysis of individual events (reactive threat mitigation instead of proactive) requires access to these enabling technologies (e.g., orchestration or policy systems such as those described in Section II-C).
- **Novel threats:** Establishing an adaptive threat management framework that (semi-)autonomously reconfigures the system at the basis of its internal threat analysis results itself opens up a entirely novel attack surface. Especially when the system is capable of reconfiguration actions (e.g., lowering defenses), malicious entities will be strongly incentivized to gain access and manipulate these models and reconfiguration facilities.

VI. Conclusion

In a model-based analysis activity, the quality of the inputs greatly impacts the overall effectiveness. This is certainly true for threat modeling: when performing a threat analysis activity at the basis of system model that is outdated, incomplete, ambiguous or inaccurate, relevant security and privacy threats may not be recognized (low recall), or conversely, the reported threats will not be realistic (low precision). In addition, threat modeling approaches currently focus on design-level threats and are less suited to identify threats that emerge at run time, i.e. as the operational context of a system changes.

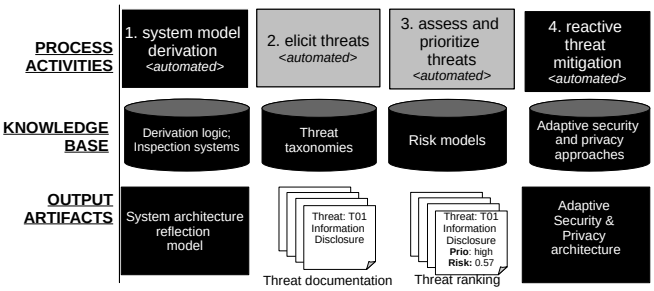

Fig. 4. Graphical depiction of the proposed threat modeling innovations (in black), in which (i) the system model is constructed through system reflection, and (ii) the analysis model is used to express and enact system security-oriented adaptations.

To alleviate these shortcomings, we explore and discuss the requirements and research challenges related to adopting more expressive system models, the construction of which is automated by means of system inspection and monitoring, in a novel threat modeling approach called '*reflective threat modeling*'. We complement this approach with a vision on how this can lead to integration of adaptive security and privacy technologies, which are then controlled and informed from this threat-centric perspective, an approach we refer to as '*adaptive threat management*'.

This vision aligns well with the canonical MAPE architecture for constructing self-adaptive systems [69], [70], which involves monitoring the system, performing analysis and planning, and enacting changes at the basis of dedicated and concern-specific analysis models. In that regard, this article discusses and considers the feasibility of adopting traditional threat modeling techniques for the main analysis and planning in support of holistic self-adaptive security and privacy approaches.

Acknowledgements. This research is partially funded by the Research Fund KU Leuven, and also supported by the European H2020-SU-ICT-03-2018 Project no. 830929 CyberSec4Europe (https://cybersec4europe.eu)

References

[1] M. Howard and S. Lipner, *The Security Development Lifecycle*. Microsoft Press, 2006.
[2] A. Shostack, *Threat Modeling: Designing for Security*. Indianapolis, Indiana: John Wiley & Sons, 2014.
[3] W. Xiong and R. Lagerström, "Threat modeling–a systematic literature review," *Computers & security*, vol. 84, pp. 53–69, 2019.
[4] N. Shevchenko, T. A. Chick, P. O'Riordan, T. P. Scanlon, and C. Woody, "Threat modeling: a summary of available methods," Carnegie Mellon University Software Engineering Institute, Tech. Rep., 2018.
[5] D. Van Landuyt and W. Joosen, "A descriptive study of assumptions made in LINDDUN privacy threat elicitation," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 1280–1287.
[6] C. Steel, R. Nagappan, and R. Lai, *Core Security Patterns: Best Pratices and Strategies for J2EE, Web Services, and Identity Management*. Prentice Hall Ptr, 2005.
[7] E. Fernandez-Buglioni, *Security patterns in practice: designing secure architectures using software patterns*. John Wiley & Sons, 2013.
[8] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.
[9] M. Colesky, J.-H. Hoepman, and C. Hillen, "A critical analysis of privacy design strategies," in *2016 IEEE Security and Privacy Workshops*, 2016.
[10] E. Yuan, N. Esfahani, and S. Malek, "A systematic survey of self-protecting software systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 8, no. 4, pp. 1–41, 2014.
[11] I. Omoronyia, L. Cavallaro, M. Salehie, L. Pasquale, and B. Nuseibeh, "Engineering adaptive privacy: on the role of privacy awareness requirements," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 632–641.
[12] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh, "Requirements-driven adaptive security: Protecting variable assets at runtime," in *2012 20th IEEE international requirements engineering conference (RE)*. IEEE, 2012, pp. 111–120.
[13] G. Tziakouris, R. Bahsoon, and M. A. Babar, "A survey on self-adaptive security for large-scale open environments," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.
[14] K. Tuma, G. Calikli, and R. Scandariato, "Threat analysis of software systems: A systematic literature review," *Journal of Systems and Software*, vol. 144, pp. 275 – 294, 2018.
[15] Z. Braiterman, A. Shostack, J. Marcil, S. de Vries, I. Michlin, K. Wuyts, R. Hurlbut, B. S. Schoenfield, F. Scott, M. Coles, C. Romeo, A. Miller, I. Tarandach, A. Douglen, and M. French, "Threat modeling manifesto," http://www.threatmodelingmanifesto.org/, 2021.
[16] J. Freund and J. Jones, *Measuring and managing information risk: a FAIR approach*. Butterworth-Heinemann, 2014.
[17] L. Sion, K. Yskout, D. Van Landuyt, and W. Joosen, "Risk-based design security analysis," in *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment*. ACM, 2018, p. 11–18.
[18] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "STRIDE-based threat modeling for cyber-physical systems," in *IEEE PES Innovative Smart Grid Technologies Conference Europe*, 2017.

[19] A. Karahasanovic, P. Kleberger, and M. Almgren, "Adapting threat modeling methods for the automotive industry," in *Proceedings of the 15th ESCAR Conference*, 2017, pp. 1–10.

[20] A. Almulhem, "Threat modeling of a multi-uav system," *Transportation Research Part A: Policy and Practice*, vol. 142, pp. 290–295, 2020.

[21] Z. Ma and C. Schmittner, "Threat modeling for automotive security analysis," *Advanced Science and Technology Letters*, vol. 139, 2016.

[22] W. Xiong, F. Krantz, and R. Lagerström, "Threat modeling and attack simulations of connected vehicles: Proof of concept," in *International Conference on Information Systems Security and Privacy*, 2019.

[23] R. Stevens, D. Votipka, E. M. Redmiles, C. Ahern, P. Sweeney, and M. L. Mazurek, "The battle for New York: a case study of applied digital threat modeling at the enterprise level," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 621–637.

[24] C. Möckel and A. E. Abdallah, "Threat modeling approaches and tools for securing architectural designs of an e-banking application," in *Sixth International Conference on Information Assurance and Security*, 2010.

[25] A. Yeboah-Ofori and S. Islam, "Cyber security threat modeling for supply chain organizational environments," *future internet*, 2019.

[26] M. Cagnazzo, M. Hertlein, T. Holz, and N. Pohlmann, "Threat modeling for mobile health systems," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2018.

[27] A. Gangavarapu, E. Daw, A. Singh, R. Iyer, G. Harp, S. Zimmerman, and R. Raskar, "Target privacy threat modeling for COVID-19 exposure notification systems," *arXiv preprint arXiv:2009.13300*, 2020.

[28] ThreatModeler, "ThreatModeler," https://threatmodeler.com/, 2020.

[29] K. Tuma, R. Scandariato, M. Widman, and C. Sandberg, "Towards security threats that matter," in *Computer Security*. Springer, 2017.

[30] L. Sion, D. Van Landuyt, K. Yskout, and W. Joosen, "SPARTA: Security & privacy architecture through risk-driven threat assessment," in *IEEE International Conference on Software Architecture Companion*, 2018.

[31] T. Antignac, R. Scandariato, and G. Schneider, *A Privacy-Aware Conceptual Model for Handling Personal Data*. Springer, 2016.

[32] Foreseeti, "SecuriCAD," https://www.foreseeti.com/securicad/, 2020.

[33] A. van den Berghe, R. Scandariato, K. Yskout, and W. Joosen, "Design notations for secure software: A systematic literature review," *Software & Systems Modeling*, vol. 16, no. 3, pp. 809–831, 2017.

[34] K. Wuyts, D. Van Landuyt, A. Hovsepyan, and W. Joosen, "Effective and efficient privacy threat modeling through domain refinements," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1175–1178.

[35] OWASP, "A pythonic framework for threat modeling." https://owasp.org/www-project-pytm/, 2021.

[36] threatspec.org, "Threatspec: continuous threat modeling, through code," https://threatspec.org/, 2021.

[37] L. Bass, "The software architect and DevOps," *IEEE Software*, vol. 35, no. 1, pp. 8–10, 2017.

[38] V. Mohan and L. B. Othmane, "SecDevOps: Is it a marketing buzzword?-mapping research on security in DevOps," in *2016 11th international conference on availability, reliability and security (ARES)*, 2016.

[39] V. Mohan, L. ben Othmane, and A. Kres, "BP: security concerns and best practices for automation of software deployment processes: An industrial case study," in *2018 IEEE Cybersecurity Development (SecDev)*, 2018.

[40] A. A. U. Rahman and L. Williams, "Software security in DevOps: synthesizing practitioners' perceptions and practices," in *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*. IEEE, 2016, pp. 70–76.

[41] M. Mattetti, A. Shulman-Peleg, Y. Allouche, A. Corradi, S. Dolev, and L. Foschini, "Securing the infrastructure and the workloads of linux containers," in *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2015, pp. 559–567.

[42] C. Schneider, "Security DevOps-staying secure in agile projects," *OWASP AppSec Europe*, 2015.

[43] S. Cash, V. Jain, L. Jiang, A. Karve, J. Kidambi, M. Lyons, T. Mathews, S. Mullen, M. Mulsow, and N. Patel, "Managed infrastructure with ibm cloud openstack services," *IBM Journal of Research and Development*, vol. 60, no. 2-3, pp. 6–1, 2016.

[44] C. Tsigkanos, L. Pasquale, C. Ghezzi, and B. Nuseibeh, "On the interplay between cyber and physical spaces for adaptive security," *IEEE Transactions on Dependable and Secure Computing*, 2016.

[45] S. K. Majhi and P. Bera, "Designing an adaptive firewall for enterprise cloud," in *2014 International Conference on Parallel, Distributed and Grid Computing*. IEEE, 2014, pp. 202–208.

[46] L. K. Strand, "Adaptive distributed firewall using intrusion detection," Master's thesis, 2004.

[47] W. Lee, S. J. Stolfo, and K. W. Mok, "Adaptive intrusion detection: A data mining approach," *Artificial Intelligence Review*, 2000.

[48] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Real-time multi-agent system for an adaptive intrusion detection system," *Pattern Recognition Letters*, vol. 85, pp. 56–64, 2017.

[49] W. Y. Moon and S. D. Kim, "Adaptive fraud detection framework for fintech based on machine learning," *Advanced Science Letters*, 2017.

[50] A. Yeşilkanat, B. Bayram, B. Köroğlu, and S. Arslan, "An adaptive approach on credit card fraud detection using transaction aggregation and word embeddings," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2020, pp. 3–14.

[51] Y. Yang, X. Zheng, W. Guo, X. Liu, and V. Chang, "Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system," *Information Sciences*, vol. 479, pp. 567–592, 2019.

[52] S. Kandala, R. Sandhu, and V. Bhamidipati, "An attribute based framework for risk-adaptive access control models," in *2011 Sixth International Conference on Availability, Reliability and Security*. IEEE, 2011.

[53] A. Khan, "Key characteristics of a container orchestration platform to enable a modern application," *IEEE Cloud Computing*, no. 5, 2017.

[54] E. Casalicchio and S. Iannucci, "The state-of-the-art in container technologies: Application, orchestration and security," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 17, p. e5668, 2020.

[55] A. Tabiban, S. Majumdar, L. Wang, and M. Debbabi, "Permon: An openstack middleware for runtime security policy enforcement in clouds," in *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2018, pp. 1–7.

[56] D. Guamán, J. Pérez, J. Diaz, and C. E. Cuesta, "Towards a reference process for software architecture reconstruction," *IET Software*, 2020.

[57] R. Koschke, "Architecture reconstruction," in *Software Engineering*. Springer, 2007, pp. 140–173.

[58] D. Weerasiri, M. C. Barukh, B. Benatallah, Q. Z. Sheng, and R. Ranjan, "A taxonomy and survey of cloud resource orchestration techniques," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–41, 2017.

[59] E. Casalicchio, "Container orchestration: a survey," *Systems Modeling: Methodologies and Tools*, pp. 221–235, 2019.

[60] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[61] N. F. S. De Sousa, D. A. L. Perez, R. V. Rosa, M. A. Santos, and C. E. Rothenberg, "Network service orchestration: A survey," *Computer Communications*, vol. 142, pp. 69–94, 2019.

[62] D. Jönsson, P. Steneteg, E. Sundén, R. Englund, S. Kottravel, M. Falk, A. Ynnerman, I. Hotz, and T. Ropinski, "Inviwo—a visualization system with usage abstraction levels," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 11, pp. 3241–3254, 2019.

[63] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: A survey," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 869–891, 2012.

[64] P. Lago, I. Malavolta, H. Muccini, P. Pelliccione, and A. Tang, "The road ahead for architectural languages," *IEEE Software*, 2014.

[65] A. A. Júnior, S. Misra, and M. S. Soares, "A systematic mapping study on software architectures description based on ISO/IEC/IEEE 42010: 2011," in *International Conference on Computational Science and Its Applications*. Springer, 2019, pp. 17–30.

[66] R. Eramo, I. Malavolta, H. Muccini, P. Pelliccione, and A. Pierantonio, "A model-driven approach to automate the propagation of changes among architecture description languages," *Software & Systems Modeling*, vol. 11, no. 1, pp. 29–53, 2012.

[67] A. P. F. Magalhaes, A. M. S. Andrade, and R. S. P. Maciel, "Model driven transformation development (MDTD): An approach for developing model to model transformation," *Information and Software Technology*, vol. 114, pp. 55–76, 2019.

[68] W. De Borger, "Middleware for the inspection of complex software systems," 2014.

[69] D. Garlan, B. Schmerl, and S.-W. Cheng, "Software architecture-based self-adaptation," in *Autonomic computing and networking*, 2009.

[70] Y. Brun, G. D. M. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, "Engineering self-adaptive systems through feedback loops," in *Software engineering for self-adaptive systems*. Springer, 2009, pp. 48–70.