The PASTA threat model implementation in the IoT development life cycle

Andreas Wolf[1], Dimitrios Simopoulos[2], Luca D'Avino[3], Patrick Schwaiger[4]

Abstract: Recently, IoT usage has grown rapidly. Security risks are rising analogously, though. Our paper introduces an approach to identify and address security threats by applying the PASTA (Process for Attack Simulation and Threat Analysis) threat model to the IoT domain. By adapting PASTA, we optimize the threat analysis based on domain knowledge and specific needs of IoT. With integration of the PASTA results into the development process and the IoT software development life cycle, we reduce security risks. A prototype demonstrates the feasibility of the concept for security vulnerability reduction via an integrated DevSecOps toolchain.

Keywords: Internet of Things; Threat Model; DevSecOps; Cyber-Security; Security Testing

1 Introduction

The Internet of Things (IoT) is becoming an integral part of our lives. With all the negative news about data leaks, privacy violations and device compromises, trust in technology is the key acceptance and success factor for IoT. Even though positive effects of IoT are proven, like resource and cost saving (weather based irrigation control), security (intrusion detection) and even life saving (smoke detectors), it will only be accepted, if we can trust it.

IoT differs from standard applications and systems due to the inherent complexity and the number of involved components. Standard applications usually consist of a few up to 10th of components (back-end servers, back-end networks, Internet, (browser)-client). IoT however starts on a different scale. Even in the smallest Cloud-based home automation scenario we have dozens of sensors on doors and windows, IP-Cameras, sun-blinds with light and wind sensors and motors, an IoT-Hub as gateway between e.g. a ZigBee IoT network, Wi-Fi and the Internet. Industrial scenarios scale that up to multiple on premise IoT networks with hundreds of sensors and actuators, multiple (hierarchical) hubs and concentrators, cloud links and services and interfaces to multiple back-end systems. Therefore, IoT always requires a system perspective.

In the context of this paper, an IoT system covers at least the local IoT network as well as services provided by the Internet. Services are vital for the system, as they provide the

[1] AKKA DSO GmbH, Taunusstraße 36, 80807 Munich, Germany, andreas.wolf@akka.eu
[2] AKKA DSO GmbH, Taunusstraße 36, 80807 Munich, Germany, dimitrios.simopoulos@akka.eu
[3] AKKA DSO GmbH, Taunusstraße 36, 80807 Munich, Germany, luca.davino@akka.eu
[4] AKKA DSO GmbH, Taunusstraße 36, 80807 Munich, Germany, patrick.schwaiger@akka.eu

doi:10.18420/inf2020_111

respective functionalities. Potential existence of threats leads to vulnerabilities, which are weaknesses of a system that make an exploit possible and can be found on network, host or application levels, including operational practices [UM15]. Successful risk handling plays a significant role in the system functionality. Although there are techniques facilitating the general software life cycle, special treatment is necessary for IoT systems, applications, and operating systems.

Due to the variety of components in an IoT system, we have different types of security measures to apply. It starts with the hardware, a Microcontroller unit (MCU) -based board or a System-on-Chip (SOC) that can contain different types of design flaws. It continues with communication links with authentication, authorization, encryption and validation protocols, which due to their computational complexity are often trades for security. IoT systems also include software in the form of firmware, operating systems, services and applications opening them for the standard threats like buffer overflows, especially because of the limited memory resources [MM19] or Random Number Generator (RNG) attacks [Ke98]. Finally, secure boot and component registration absence can lead to undesirable system functionality and malicious software boot.

This paper discusses methods and practices for dealing with potential security risks and complex threats during the IoT system life cycle. We adapt a specific threat model for deriving design security requirements, architecture- and implementation-countermeasures and verification for the Continuous Integration tool chain. A methodology is defined to predict and manage the potential threats that arise in the process of developing and using any such IoT system.

2 Literature Review

Threat models are integral to risk identification. A plethora of such models were created over time. The most mature threat model was proposed by Loren Kohnfelder and Praerit Garg in 1999 under the name STRIDE and was widely used for several years by Microsoft [KG99]. The specific model evaluates the overall design of the system, either through data flow charts (DFD's) or through a repository of known threats. Threat identification is based on its name acronym: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege. Thus, security experts can break down and analyse in depth potential risks, processes and flows. Although it provides the user with checklists and methodologies, it cannot represent any security architectural decisions and it is time consuming.

Another threat model is LINDDUN, focusing on software architecture and data security [WJ15]. LINDDUN stands for Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, Non-compliance. The model uses data flow diagrams to represent the system's overall state. Constant analysis of the system threats helps to identify its weaknesses. LINDDUN achieves that through workflow iterations. A powerful identification tool is the rich knowledge-base and its documentation [WSJ14]. The drawbacks are the high workload and time consumption that the specific model needs.

The National Institute of Standards and Technology provides an additional threat model, the Common Vulnerability Scoring System (CVSS) [SM09]. A final numerical score depicts the severity of specific threats based on the assessment of vulnerability characteristics. An online calculator is available for the results [Ib11]. Three metric groups - base, temporal and environmental, play an important role during the recognition and analysis phase, determining the output score. Due to the calculation method, analysts doubt its accuracy. This is a major reason the final score values may vary, which results in combination with other existing threat models for better results [Ha13].

One of the oldest and most widely used threat models is the Attack Tree. Bruce Schneier presented that threat model in 1999, with the help of extensive tree diagrams [Sc99]. The root represents the purpose of the attack, while the leaves depict different ways. Every major target of an attack is broken down into more discrete roots. So, with the specific threat mode, the user iterates through the problem and breaks it down into more attack trees. This threat model focuses on experienced analysts, assuming they have cybersecurity experience and thus does not provide them with detailed documentation or guidelines.

A recently proposed threat model is the Persona non Grata (PnG) [Cl14]. Its goal is to profile the potential attacker and their skills. This helps to outline vulnerabilities and threats early and understand the attacker's possible motives and skills. While this approach fits into the agile methodology and it is easy to use, the threat identification is limited to a specific set of threat types [Me18].

Quantitative threat modelling is a hybrid method, combining features from attack trees, STRIDE, and CVSS [PMK16]. Analysts build attack trees based on the five STRIDE categories and afterwards they apply the CVSS method to compute the respective scores for the tree components. This method focuses more on cyber-physical systems and risks with complex dependencies among their components.

Trike is an additional threat model, launched as a security audit framework in 2005 [Me18]. It aims more in the defensive side of a system. Analysts need an overview of the system's environment and create a requirement model. For the risk assessment, Trike normalizes the likelihood of the respective risk occurrence. The downsides, however, are lack of documentation and doubt of its calculation accuracy.

In 2003, the CERT division presented its own method, OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) [Al03]. Strategic evaluation of potential organizational risks is its primary focus. As this model fits larger organizations, it assesses operational risks and activities, not processes. Using this model, however, is a long process and its documentation is lengthy.

3 Introduction of the PASTA Threat Model

Threat models create an abstraction of the system to get a better understanding of the complexity of a system in order to find security flaws and potential attack vectors [Sh18]. Provided that, system architects and developers can respond to design failures and improve the system security [MLY05].

PASTA is a risk-centric threat modelling framework, proposed by Tony UcedaVélez in 2012 [UM15]. It includes different layers of abstraction and considers high-level layers, like the business logic, as well as the lower layers, like the concrete attack vectors. Especially in the IoT sector, a good understanding of abstract system architecture and business objectives is needed. The specific model fills the gap between technical and business risk analysis related to cyber threats. It provides analysts with an overview of manageable threats and risks to the addressable application environment. Thus, PASTA is the proposed threat model for IoT projects. The PASTA threat model consists of seven (7) stages, as shown in Figure 1, and can be applied to already existing projects or during the definition phase of a new project. We define the mapping of these seven stages of the PASTA Model to the IoT environment in the following chapter.

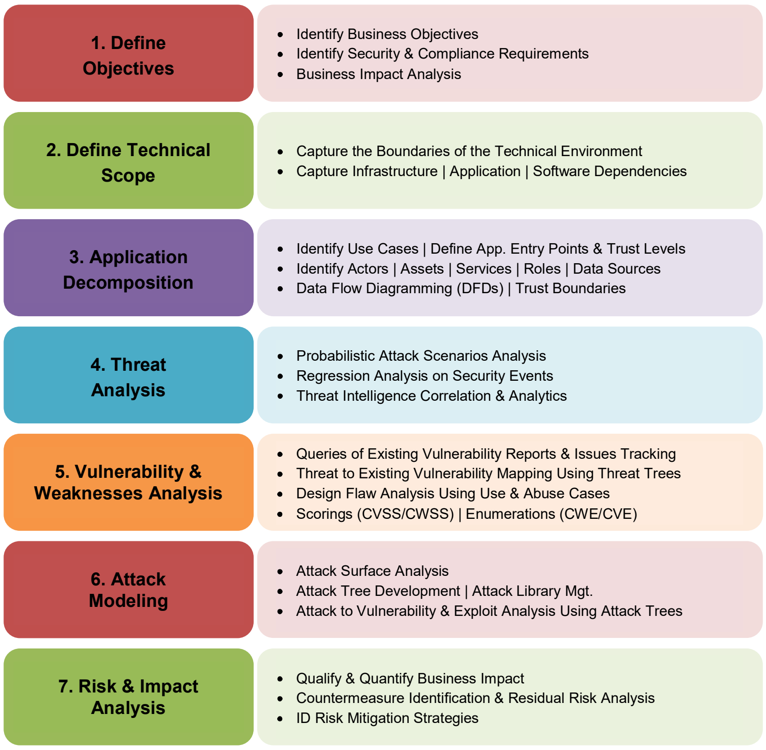
1. Define Objectives
Identify Business Objectives
Identify Security & Compliance Requirements
Business Impact Analysis
2. Define Technical Scope
Capture the Boundaries of the Technical Environment
Capture Infrastructure | Application | Software Dependencies
3. Application Decomposition
Identify Use Cases | Define App. Entry Points & Trust Levels
Identify Actors | Assets | Services | Roles | Data Sources
Data Flow Diagramming (DFDs) | Trust Boundaries
4. Threat Analysis
Probabilistic Attack Scenarios Analysis
Regression Analysis on Security Events
Threat Intelligence Correlation & Analytics
5. Vulnerability & Weaknesses Analysis
Queries of Existing Vulnerability Reports & Issues Tracking
Threat to Existing Vulnerability Mapping Using Threat Trees
Design Flaw Analysis Using Use & Abuse Cases
Scorings (CVSS/CWSS) | Enumerations (CWE/CVE)
6. Attack Modeling
Attack Surface Analysis
Attack Tree Development | Attack Library Mgt.
Attack to Vulnerability & Exploit Analysis Using Attack Trees
7. Risk & Impact Analysis
Qualify & Quantify Business Impact
Countermeasure Identification & Residual Risk Analysis
ID Risk Mitigation Strategies

Fig. 1: The seven (7) stages of PASTA [UM15].

4 Applying the PASTA model to the IoT-ecosystem

Defining goals is the first stage in PASTA. Understanding business and its pain points is the basis for understanding and valuing risk. The system's main goals are set, and key services and target markets are specified, resulting in security and compliance requirements. Stride's security compliance can be a starting point to identify and address threats and risks. Business impact analysis to assess the importance of specific services, applications or devices can be based on ISO 22301 methodology. By defining maximum interruption, recovery time and priority, risk can be classified and prioritized from a business perspective.

The definition of the technical scope is the second stage providing the initial attack surface. Component and network diagrams reveal network design flaws like single failure points. For the respective devices, a detailed definition of accessibility, hardware, firmware, operating systems, and executed tasks, helps to identify device specific security requirements and vulnerabilities. Special attention is paid to unused communication and network interfaces, to identify side channels for an attack. Also, it is important to track hardware extensions, such as additional external modules. Technology enumeration, divided into architectural layers, outlines better the technology used. Specifically, a sub-categorization of the operating systems, programming languages, software libraries and communication protocols, ensures comprehensive technological enumeration.

Application decomposition is the next stage, defining system applications (units of deployment) and services (functional elements).The versatility and complexity of IoT networks carries the risk of not covering all applications and services. To reduce this risk, a detailed use-case diagram is essential to keep the overview of its actors, roles and features. The use case diagram shows unused features that can be eliminated or deactivated to reduce the attack surface. It also shows the intended use of the services to identify patterns to distinguish between intended and unintended use. If a feature has poor security performance the use cases indicate if we must redesign or secure it or if we can just eliminate it. Since correct and trustworthy data is the key to an IoT system, a data flow diagram (DFD) gives an important alternative view of the system and its vulnerability. The DFD in Figure 2, includes all data entry and exit points from sensors and actors via manual data entry screens to interfaces. Colours are used to denote the trustworthiness of an item (green: trustworthy) (orange: less trustworthy) (red: not trustworthy) (black: unknown) [AWT07]. The red-dotted circle wraps the trusted boundaries. This annotated DFD analyses the possible impact of incorrect or manipulated data and gives a data driven indicator for the need to secure a component or validate their data before using to increase trust.

In the fourth stage the threats of the respective system are identified. This threat analysis the cornerstone of the model and leads to a customised threat library [UM15]. Such a library is a collection of threats that may harm the overall system functionality. The first step is to be aware of all attack targets, the attack surface. Targets are the technical and functional components identified in stage 2 and 3. Attack targets are classified by likelihood of an attack at that target which is influenced e.g. by the popularity of the component, the outside

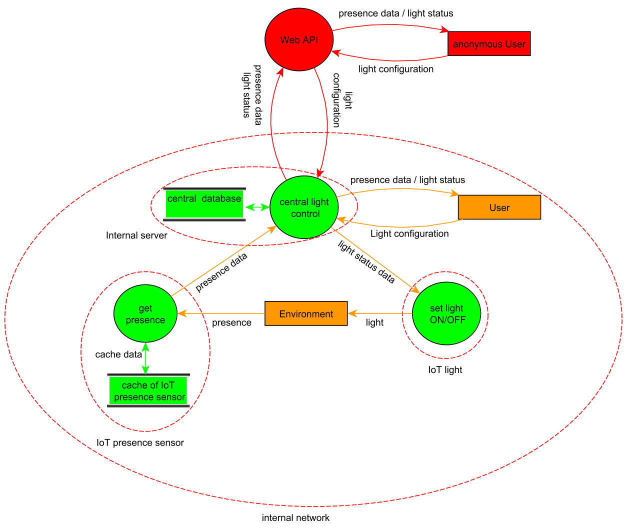

Fig. 2: Representation of an IoT light control system with different trust boundaries.

visibility or a history of known attacks. A second classification is based on business impact analysis. To provide a well-structured threat library, threats are categorized e.g. starting with the categories of the STRIDE model [KG99]. We introduce extortion and physical attacks as important threat categories. In addition to commonly known threats the field of IoT adds new threats like sensor spoofing and battery drain attacks. Typically IoT devices have constrained computational power, less memory and a limited battery capacity. Challenging such limitations is a typical strategy for attackers to find new threats [HFH15]. The list of categories is not static. It must be extended if new IoT system concepts get introduced, that lead to totally new types of threats.

Vulnerability and weakness analysis are the fifth stage. The main task is to enumerate the system vulnerabilities on a technical level using Common Weakness Enumeration (CWE). A comprehensive enumeration requires a vulnerability analysis of any used technology. Public vulnerability enumerations facilitate that task. Existing technologies are often adapted. Thus, security experience with standard Internet technology can often be transferred to the IoT environment. For instance, the Constrained Application Protocol (CoAP) and the Hypertext Transfer Protocol (HTTP) share similar concepts and vulnerabilities [Sh14]. Already created network and data flow diagrams offer insightful information to assess vulnerabilities in system architecture. The abuse-case diagram maps a threat to a component of the system. That extends the normal UML use-case diagram creation with abuse-case shapes, showing how a system is used by an attacker. Abuse-case diagrams create the link between a threat and an applied vulnerability to the system. To adapt it to IoT, IoT devices are considered actors in the Figure 3.

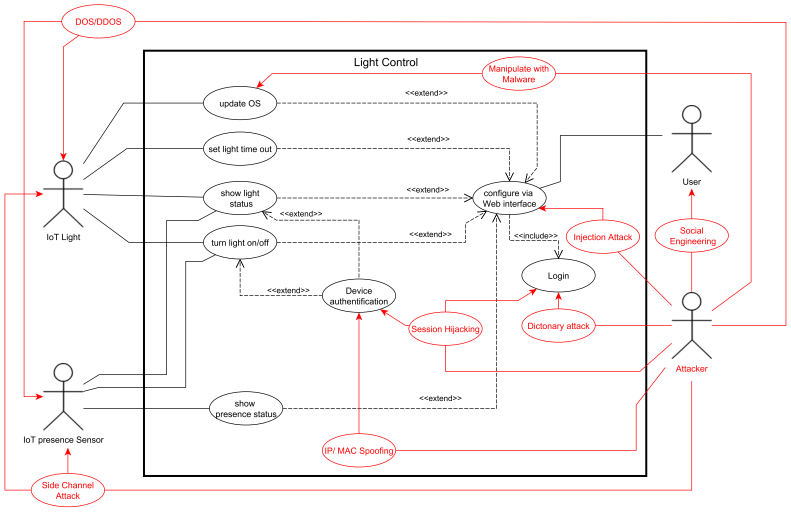

Fig. 3: An abuse-case diagram of a light control system.

The sixth stage, attack modelling, applies vulnerabilities to the system. Attack trees help to visualize discrete vulnerabilities into tree forms [Sc99]. To define the main targets of attack trees, the threats from the threat library are applied to the attack targets. Additionally, the abuse diagrams provide a good overview of which attacks can be executed on the system components. At the end of this task, the attack trees show how different vulnerabilities are used to pose a threat to the system.

The final stage is residual risk analysis. The system's most likely threats and vulnerabilities are evaluated to find countermeasures, mitigate, and detect threats. The use of CVSS (Common Vulnerability Scoring System) or CWSS (Common Weakness Scoring System) helps to assess the relevance and impact of the vulnerabilities in the enumeration. CVSS/CWSS is not optimized for the IoT environment. However, there are promising approaches of IoT-optimized risk assessments like an IoT risk assessment of CyVaR [Ra18]. Based on the threat, appropriate countermeasures must be taken Attack detection, mitigation or prevention. Mitigation is used to reduce attack damage and can be performed automatically by the system after attack detection or manually by isolating or deactivating an attacked or compromised target. Prevention methods often involve a performance trade, but risk analysis provides information needed to decide on it. Specifically, for high-risk threats performance-expensive prevention is acceptable. Countermeasures should be assigned to corresponding threats in the threat library.

DevSecOps is the methodology to improve the software development life cycle by combining development, security and operations. It extends DevOps to integrate security into the agile development process [MC17]. Security analysis in combination with the PASTA threat model ensures high threat identification accuracy. That gives us the ability to verify the correct handling of those risks through testing. Regression testing has is a key component of the DevOps tool chain. Conventional regression tests only ensure the functionality of the business logic and seldom non-functional requirements like response time or throughput. A DevSecOps tool chain should be able to perform evidence-based and reproducible security regression tests, to continuously guarantee robustness and security of the IoT system in an agile development process. Unlike normal unit tests, security tests do not isolate individual system functions. Full system simulation is needed to observe possible side effects and simulate a realistic scenario. Security tests are derived from the security and compliance requirements defined in the PASTA threat model. A security test failure indicates a possible vulnerability and an uncovered non-functional requirement. Like functional tests, the implemented threats of the threat library can be classified according to criticality do detect the frequency in which they are performed.

To prove our concepts, we created a prototype IoT system, based on the RIOT operating system. RIOT, which is the basis for DoRIoT, is well suited because of the native ability to run as a virtual instance. Virtualisation allows us to do industry standard two-stage validation based on Software in the Loop (SIL) and Hardware in the Loop (HIL). The system contains two applications for sensor and actuator. The latter allows the internal LED to be switched on or off, when the sensor triggers it via a message across the network. Networking is based on CoAP over UDP IPv6. We first implemented the prototype, then applied code- and memory-analysis tools to identify code-level weaknesses and then applied the threat model to the prototype. All checks are done in an extended continuous integration pipeline that automatically creates a reference system without attacks and test systems, that include active attackers. Thus, we can compare normal system behaviour with attacking system behaviour. As expected, the threat model revealed system threats and weaknesses. Vulnerabilities were detected in the network and service layers, such as Vulnerability to DOS/DDOS attacks and ping flooding that could bring down the full RIOT OS. Tests could be repeated automatically to verify results and to show the effects of changes in the prototype.

5 Conclusion

IoT opens a wide range of possibilities but also poses new threats. The more IoT devices are integrated into our lives, the greater is the demand for security. In this paper we analysed and evaluated different threat models to determine which one suits IoT best. In our investigations, the PASTA threat model seemed the best approach. We improved the PASTA threat model by applying existing research and industry standards like the ISO 22301 for the Business Impact Analysis. We gained a hardware perspective by adding a network diagram and considering IoT specific properties like accessibility and hardware limitations. We modified the Data Flow Diagram by adapting the colour coding to highlight the trustworthiness of a data source based on location and reliability. To adopt the threat library, we evaluate new threat categories. The technology-enumeration and the network diagram of stage two create the IoT context to the threat library.

The adapted threat model made it possible to identify threats and vulnerabilities of an IoT system, and to assess the evidence-based risks. This methodology improves the security of the development life cycle and of existing IoT systems.

Based on our research, we created a concept for security tests based on threat model theory and data. This concept fulfilled all the conditions for the fast and agile development life cycle of DevOps, contributing to DevSecOps. To prove the concept, we created an IoT prototype system and the implementation of the respective security tests. A future objective is to gain experience in implementing the concept to broaden and improve this methodology.

Test results proved that system vulnerabilities exist and must be detected and fixed. The integration into the continuous integration pipeline proved that security tests can be implemented as automated regression tests. We could show that a threat model, based on a DevSecOps model, improves the security of an IoT system and does not restrict the agility and efficiency of an IoT project.

The presented results are part of the DoRIoT research project [20].

References

[20] Dynamic runtime for organically (dis-)aggregating IoT-processes, Aug. 2020, URL: http://doriot.ovgu.de/.

[Al03] Alberts, C.; Dorofee, A.; Stevens, J.; Woody, C.: Introduction to the octave approach. Pittsburgh, 2003.

[AWT07] Abi-Antoun, M.; Wang, D.; Torr, P.: Checking threat modeling data flow diagrams for implementation conformance and security. In: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering. Pp. 393–396, 2007.

[Cl14] Cleland-Huang, J.: How well do you know your personae non gratae? IEEE software 31/4, pp. 28–31, 2014.

[Ha13] Hanford, S.: Common vulnerability scoring system, v3 development update. In: Technical report, Forum of Incident Response and Security Teams (FIRST). 2013.

[HFH15] Hossain, M. M.; Fotouhi, M.; Hasan, R.: Towards an analysis of security issues, challenges, and open problems in the internet of things. In: 2015 IEEE World Congress on Services. IEEE, pp. 21–28, 2015.

[Ib11] Ibidapo, A. O.; Zavarsky, P.; Lindskog, D.; Ruhl, R.: An analysis of CVSS v2 environmental scoring. In: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing. IEEE, pp. 1125–1130, 2011.

[Ke98] Kelsey, J.; Schneier, B.; Wagner, D.; Hall, C.: Cryptanalytic attacks on pseudorandom number generators. In: International Workshop on Fast Software Encryption. Springer, pp. 168–188, 1998.

[KG99] Kohnfelder, L.; Garg, P.: The threats to our products. Microsoft Interface, Microsoft Corporation 33/, 1999.

[MC17] Myrbakken, H.; Colomo-Palacios, R.: DevSecOps: a multivocal literature review. In: International Conference on Software Process Improvement and Capability Determination. Springer, pp. 17–29, 2017.

[Me18] Mead, N. R.; Shull, F.; Vemuru, K.; Villadsen, O.: A Hybrid Threat Modeling Method. Carnegie MellonUniversity-Software Engineering Institute-Technical Report-CMU/SEI-2018-TN-002/, 2018.

[MLY05] Myagmar, S.; Lee, A. J.; Yurcik, W.: Threat modeling as a basis for security requirements. In: Symposium on requirements engineering for information security (SREIS). Vol. 2005, Citeseer, pp. 1–8, 2005.

[MM19] Mullen, G.; Meany, L.: Assessment of Buffer Overflow Based Attacks On an IoT Operating System. In: 2019 Global IoT Summit (GIoTS). Pp. 1–6, 2019.

[PMK16] Potteiger, B.; Martins, G.; Koutsoukos, X.: Software and attack centric integrated threat modeling for quantitative risk assessment. In: Proceedings of the Symposium and Bootcamp on the Science of Security. Pp. 99–108, 2016.

[Ra18] Radanliev, P.; De Roure, D. C.; Nicolescu, R.; Huth, M.; Montalvo, R. M.; Cannady, S.; Burnap, P.: Future developments in cyber risk assessment for the internet of things. Computers in Industry 102/, pp. 14–22, 2018.

[Sc99] Schneier, B.: Attack trees. Dr. Dobb’s journal 24/12, pp. 21–29, 1999.

[Sh14] Shelby, Z.; Hartke, K.; Bormann, C.; Frank, B.: RFC 7252: The constrained application protocol (CoAP). Internet Engineering Task Force/, 2014.

[Sh18] Shevchenko, N.; Chick, T. A.; O’riordan, P.; Scanlon, T. P.; Woody, C.: Threat Modeling: a Summary of Available Methods. no. July/, 2018.

[SM09] Scarfone, K.; Mell, P.: An analysis of CVSS version 2 vulnerability scoring. In: 2009 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE, pp. 516–525, 2009.

[UM15] UcedaVélez, T.; Morana, M. M.: Risk centric threat modeling. Wiley Online Library, 2015.

[WJ15] Wuyts, K.; Joosen, W.: LINDDUN privacy threat modeling: a tutorial. CW Reports/, 2015.

[WSJ14] Wuyts, K.; Scandariato, R.; Joosen, W.: LIND (D) UN privacy threat tree catalog. CW Reports/, 2014.