Introducing model-based tool support for applying zero-trust security for microservices at a bank

Donald Baldwin, Martin Henkel* and Erik Perjons

Stockholm University, Borgarfjordsgatan 12, 164 25 Kista, Sweden

Abstract

Zero-trust security involves designing, coding, and deploying applications, assuming that threats may exist both inside and outside the application environment. Developing applications using a zero-trust design is complex since it requires internal development teams to understand and apply zero-trust principles throughout the development process. This is especially crucial for microservice architectures, where many independent teams develop services. However, enforcing and teaching security principles may lead to a formal process, focusing on documentation and auditing rather than agile development. In this paper, we describe a pragmatic use of a modeling tool that is tied to a knowledge repository and contains means for team communication. The tool supports a systemic way of developing zero-trust architectures, catering to both programming needs and the desire to improve the overall development process. The paper concludes with lessons learned from a bank case study where the tool has been developed and utilised for microservices development.

Keywords

Zero-trust architecture, Modeling tool, STRIDE analysis, VSM

1. Introduction

In the domain of cybersecurity, applying zero-trust (ZT) principles marks a paradigm shift from the traditional perimeter-centric security models to a more holistic, omnipresent security. Traditional perimeter-centric security is a data security strategy that focuses on protecting the outer boundaries of a network. The idea is to establish a strong "perimeter" around the network to prevent unauthorised access and external attacks. Using ZT principles, on the other hand, operates on the principle that trust is an omnipresent vulnerability [1]; hence, no distinction is made between internal and external threats. This approach necessitates continuous verification of identity, and other contextual factors before granting access to resources [2].

The need for ZT stems from an increasing sophistication of cyber threats and the recognition that breaches often occur due to the exploitation of overly trusted networks and systems. This is especially critical in distributed architectures, such as architecture using microservices, where the security of each discrete service is important to prevent a domino effect of vulnerabilities.

The adoption of ZT principles necessitates a departure from conventional security approaches, particularly in the development and management of systems. It demands not only a technical reconfiguration but also a comprehensive understanding of its principles across the organisation's teams. A major challenge that organisations face in this regard is the dichotomy between heavy formal security processes and the agility required by development teams. Formal processes, with their exhaustive checklists and protocols, are perceived as burdensome, prompting teams to engage in informal practices that, while expedient, inadvertently circumvent established security measures. Thus, the problem addressed in this paper is the challenge to

BIR-WS 2024: BIR 2024 Workshops and Doctoral Consortium, 23rd International Conference on Perspectives in Business Informatics Research (BIR 2024), September 11-13, 2024, Prague, Czech Rep.

* Corresponding author.

don.baldwin@dsv.su.se (D. Baldwin); martinh@dsv.su.se (M. Henkel); perjons@dsv.su.se (E. Perjons)

0000-0003-3712-7454 (D. Baldwin); 0000-0003-3290-2597 (M. Henkel); 0000-0001-9044-5836 (E. Perjons)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

CEUR-WS.org/Vol-3804/short20.pdf

support the adoption of ZT principles within teams used to traditional security approaches while using agile development approaches.

In the paper we examine a solution in the form of a modeling tool that allows the modeling of microservices and their associated threats, but also provides features for communication and process support that allows the teams to develop knowledge about security principles. The tool is based on Data Flow Diagrams and STRIDE analysis. In this paper, our focus is mostly on how the tool is situated in an organisation that is using it, rather than the syntax of the used diagram. While the modeling tool contains basic modeling features, it has some features that make it especially suited for ZT microservices architectures. For example, the collaborative features of the tools enable scaling across multiple development teams, which is essential for supporting microservices architectures. Moreover the tool also contains a component library - making sharing and pushing for ZT design principles among teams possible. Thus the tool's benefits lie only partially in the core modeling support, equal importance is in the organisation support.

We examine how the tool supports the organisation by recognising the organisation as an organic entity, akin to a living organism, which requires a balance between its operational functions and strategic imperatives. Inspired by the Viable System Model (VSM) [3], our method underscores the importance of systemic thinking. VSM is an approach that views an organisation as an organism, encompassing both the tactical day-to-day operations and the overarching strategic vision. It advocates for a symbiotic relationship where different teams work in concert, ensuring the security integrity of the organisation at every level. We use VSM to analyse the potential effect of using the modeling tool.

The paper is structured as follows. The main concepts and related research is introduced in section 2. Section 3 covers the research approach. Section 4 is devoted to describing the case company, the tool features, and lessons learned from applying the tool. In section 5 we analyse - based on VSM - how the tools helps the organisation.

2. Background

Even though ZTA is fairly new concepts for protecting IT systems, there are ample amounts of papers that describe its technical implementation, but far less that describe how to use models to shift an organisation's way of working to build systems using ZT principles.

Technical implementations to uphold ZTA include measures such as continuous verification [4] and monitoring to ensure that security policies are upheld [2]. Even when discussing technical means for ZTA implementation, there has been a discussion about the effort needed for shifting to ZTA, mentioning both the cost for tools [5] and the analysis needed before migrating [6]. The migration to ZTA entails 1) identification of current resources (IT systems), 2) risk assessment and prioritisation, and 3) deployment and review [6]. The modeling tool presented in this paper supports the three steps - identification and description of resources, risk assessment, and security reviews. However, there is currently no support for real-time monitoring.

Modeling as a way to understand and take security measures can be undertaken in several ways. One way is to make use of an existing modeling framework. For example, the TOGAF framework may be used for security analysis [7]. While this has the benefit of using a well-known model and/or method as a foundation, there is a risk of obscuring the problem at hand—dealing with security. Another approach is to use tailor-made models for security. For example, the CORAS model focuses on modeling risk by creating relations between risks and harmful outcomes [8]. CORAS models are similar to goal models in that they convey a cause-effect view of actions taken. Another example of a tailor-made model is the Microsoft Threat Model tool [9]. The threat modeler uses the same basic concepts employed in the tool presented in this paper - its foundation is the architecture of the system modeled as data flows. While the tool presented in this paper also uses data flow diagrams (DFDs), it has some additional features that set it apart

from the Microsoft Threat Modeler. Most prominently, it includes a component library, making it easier to get started with. Moreover, it also has team collaboration features, which are essential for continuously using the tool and keeping the models and software updated.

3. Research methodology

This study employs a Design Science Research (DSR) [10] methodology, which involves the creation and evaluation of artefacts designed to solve identified organisational problems. The DSR approach ensures the practical relevance of the solution and its contribution to the knowledge base for both research and general practice. In this paper, our focus is on the demonstration part of Design Science Research. Specifically, we present the application of the modeling tool and share the lessons learned from its application. We base these lessons learned from first-hand experience working in the case organisation, and an interview with personnel using and developing the tool within the case organisation.

4. Case study at a bank

The modeling tool has been deployed at a multinational bank, which is offering online payment solutions to other businesses (B2B). The bank distinguishes itself by providing a wide array of software deployment options tailored to its clients' needs. Operating in a highly regulated financial sector, the bank is committed to stringent compliance with relevant financial regulations and security standards. This commitment ensures the integrity and reliability of its services and is crucial for upholding trust among its business clients. The bank's adherence to these regulatory requirements is integral to its operations, as it needs to handle the complexities of providing secure, efficient, and compliant payment solutions in a global marketplace.

The work with the bank's software solutions is divided into several types of teams. A team typically consists of 8 to 12 team members. Each team is using the modeling tool to support their work, or in the case of the architecture team, has the potential to use it:

- The *security team* is conducting modeling, audits and reviews. The team uses the tool as a base for modeling, auditing, and reviewing security functions before and during the development of microservices.
- *Security team for penetration testing*. The team is using the tool to gain insight into the vulnerabilities of the microservices, which helps the team take action to improve their security posture.
- *Development teams model, design and document the security functions of the microservices*. The teams are using the tool to model, design and document new microservices and the maintenance of existing services, including security functions.
- *The architecture team describes business cases and designs the overall solution*. The team plays the role of mediator between the business and development teams. So far, the team has not used the tool. However, the current plan is that the architects in the future could use the tool for architecture auditing of the microservices.

When the tool was introduced, it was used solely by the security team to model, audit and review the microservices and their security functions. However, the use now, when the tool is fully introduced, is that the development teams use the tool to create models of the microservices and their security functions, and then the security team uses the model as a foundation for the auditing and reviews. Hence, as will be discussed in Section 5, the tool is now much more integrated with the organisation.

4.1. Description of the tool

The tool is designed to assist in conducting a thorough security STRIDE threat analysis, a key component in identifying and mitigating potential security threats in system design. It is open source (Apache License), built with TypeScript, using a Postgres database, Node.js webserver React front-end. The tool is composed of several modules and functions that facilitate a model-based and partially automated security assessment.

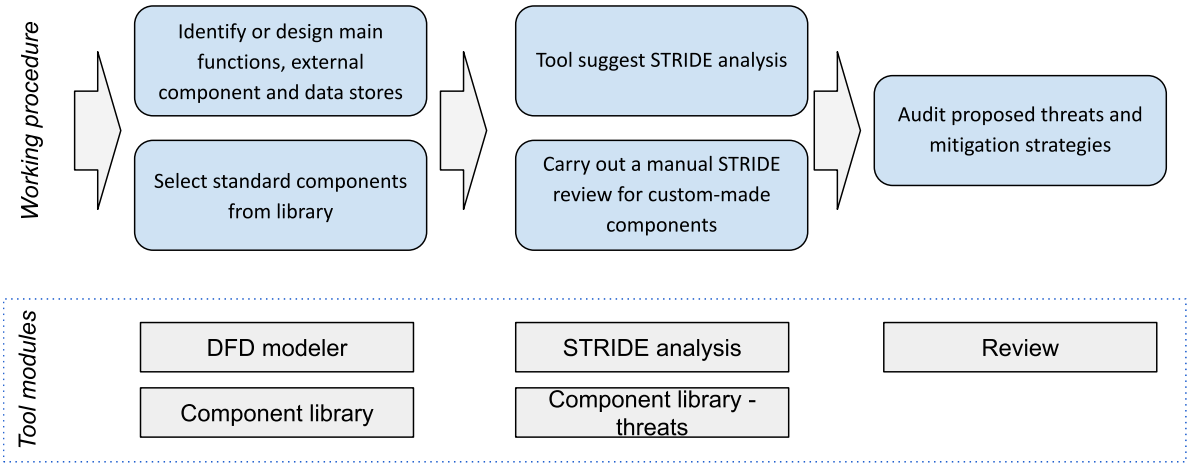

Figure 1: Overview of the work process and tool support

In practice, users begin by employing the modeling function to draft a data flow diagram (DFD), which maps the flow of data and identifies critical processes within the microservices to be constructed. Concurrently, users leverage the library of components. The library contains various elements such as cloud services, open-source libraries, and organisational-specific components. Each component is accompanied by a dedicated STRIDE threat analysis, ensuring that potential vulnerabilities are not overlooked. The areas covered by STRIDE are Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege, and are a common way of identifying IT security threats.

Given the DFD, the tool performs a partially automated STRIDE analysis. This process generates a list of identified threats based on the employed components. The tool then proposes mitigation strategies to be integrated into the system's design and deployment. The mitigation strategies are denoted as *Controls*. To ensure comprehensive coverage, the analysis is supplemented by a manual STRIDE review for custom-made components, enabling the combined analysis of both custom-made and off-the-shelf software components.

When a first draft of the model and analysis has been done, it can be sent for review. The review process begins with an architect/developer marking a model for review, initiating a collaborative platform for direct communication between security reviewers, architects, and developers. This collaborative approach ensures a unified understanding of security threats and mitigation strategies in the form of available controls. Key to the collaborative process is the setting of action items, where specific tasks are assigned to address identified threats, enhancing accountability and ensuring effective implementation of security measures.

Additionally, the tool supports follow-up reviews, allowing for the verification of completed action items and the documentation of changes, crucial for maintaining security. A notification system complements this process by sending email alerts about significant events like review initiation, action item assignments, and task completions. This ensures all stakeholders are informed and can respond promptly, fostering continuous engagement with the security improvement process.

In the following, we describe the main modules of the tool.

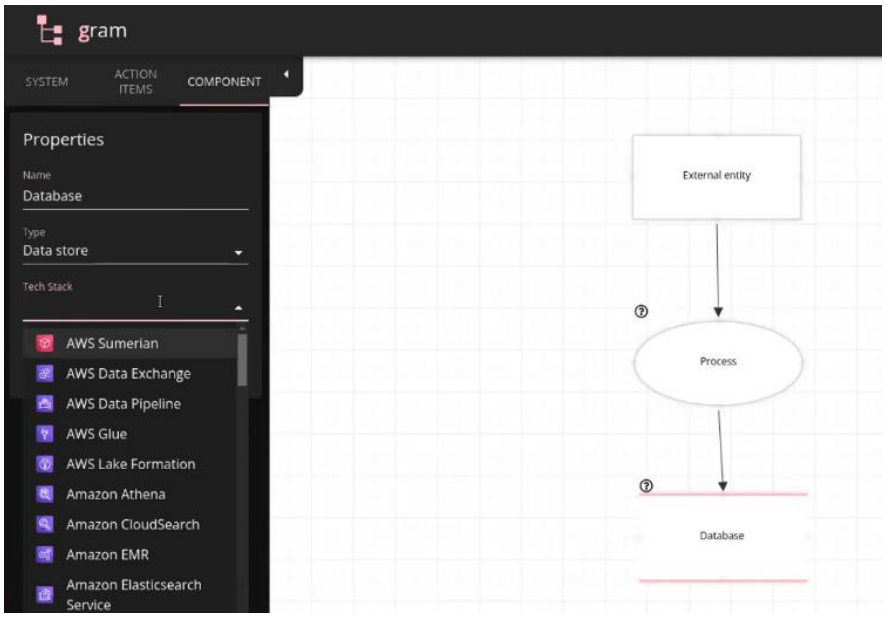

Figure 2: The DFD modeler, and component library (left side)

Modeler for Data Flow Diagrams (DFDs): This module enables users to visually represent and analyse the flow of data within their systems. It's crucial for understanding how data moves and where vulnerabilities may exist. DFDs use standardised symbols to represent the flow of data within a system, highlighting where information is processed and stored (see Figure 2). The primary graphical elements include: External entities that are external sources of data (rectangles), Processes functions or activities that transform data (ovals), Data stores - repositories where data is held (two parallel lines), and Data flows depicting the movement or transfer of data (arrows). These elements work together to provide a visual understanding of the system's data handling, facilitating analysis and design.

Library of Components: The tool includes a library of components and services that can be used within the organisation. This library encompasses infrastructure/platform elements like AWS services, open-source libraries such as Docker, and reusable components developed internally. A component is referred to as a technology stack. Each Process and Data store can be associated with several components. For example, a service can be build using both using Java (one component) and run on Apache (another component). An example of components can be seen on the left hand side of Figure 2.

STRIDE Analysis: The tool provides STRIDE analysis functions. This feature is instrumental in identifying and assessing potential security threats in six key areas: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. The STRIDE analysis could be both manual and automatic:

Automated Analysis: Based on the DFD model and the included component names, the tool can automatically generate a list of identified threats, assess their risks, and suggest mitigation strategies. For example, if a PostGreSQL database is used, the tool will suggest SQL injection as a threat (Figure 3).

Manual Analysis Support: For custom-made components in a project, the tool supports a manual STRIDE analysis, ensuring comprehensive coverage of all system elements.

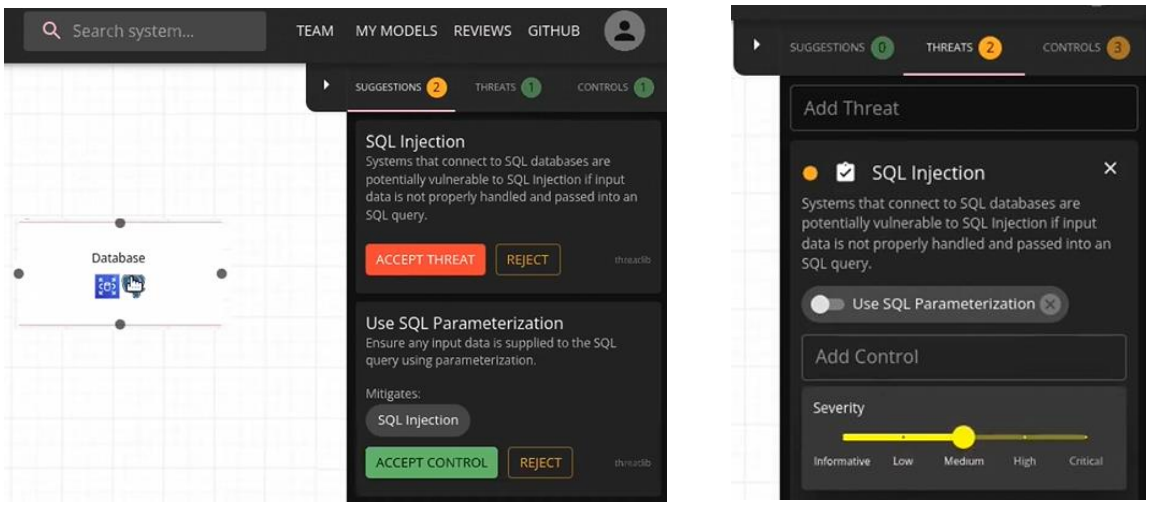

Figure 3: STRIDE analysis, suggested threat and controls (left) and setting severity (right)

STRIDE review: When the architecture requires a review, there are several features that enable the architect and security reviewer to communicate. Unmitigated threats (threats that have no controls assigned) will be shown as warnings in the DFD diagram. This makes it possible to see any issue at a glance.

Collaborative features: There are also collaborative features built into the tool. This includes marking a model as in need of review. Once the review has commenced, the reviewer can set action items for developers and architects to follow up on. These action items are handy when performing a follow-up review to observe and document the actions that have been taken. All relevant events can be sent as mail to notify the review team and developers. The modeling tool is also fully real-time multi-user - meaning that during a review session both the auditor and developer can change the model.

While the tool itself is quite straightforward, its implementation in the organisation is more complex. The next section will discuss the implementation in this case.

4.2. Lessons learned from the tool implementation

The implementation of the tool in the case highlighted several key enablers for its successful adoption. Some positive and negative aspects were also uncovered.

Tool learning curve. Initially, the learning curve presented a significant challenge, as many developers were unfamiliar with modeling techniques. To mitigate this, special training sessions were introduced, supplemented by the availability of support bookings for tool use. Auditing sessions also served as a learning platform, providing feedback on model implementations and fostering continuous improvement. The prerequisite knowledge of STRIDE emerged as a critical enabler for tool utilisation. Recognising the gap in widespread STRIDE familiarity, efforts were made to educate the user base, emphasising the importance of understanding this framework to leverage the tool effectively. Gradually the developers learn the analysis and the tool, and now most models are created by developers themselves.

Tool design. Flexibility in the tool's design proved beneficial, allowing teams to incorporate project-specific components and guidelines in the form of templates. This adaptability was further enhanced by filtering mechanisms that excluded irrelevant data, such as third-party integrations, thereby streamlining the focus on central security responsibilities.

The adoption of the tool brought several *positive aspects*. Adopting the tool yielded significant benefits, notably reducing the burden on the security team by minimising the time and personnel required for auditing. It facilitated a more structured approach to security, generating models, graphical presentations, and documentation for applications and microservices. The tool also

enabled a structured security review process, compelling developers to document their adherence to for example security protocols and the usage of encryption standards, thereby identifying and rectifying outdated practices.

Several challenges were also encountered in the case. *Engaging teams* to initiate tool usage was difficult; however, this was effectively addressed by replacing self-guided instruction with facilitated training sessions, which enhanced understanding and engagement. The complexity of some models posed another obstacle, discouraging early adoption. This issue was similarly overcome through guided sessions, which provided direct support and guidance, enabling more effective model simplification from the outset. A guide was also built into the tool, explaining the basic concepts of the model.

A noted drawback with the use of the tool was the *increased complexity* introduced into the development process. The requirement for detailed security documentation, while beneficial for security oversight, was initially perceived as an additional burden by developers. This underscores the need for balancing security rigor with development efficiency, a challenge that will inform future tool enhancements and training approaches.

5. Systemic effects of using the tool

In this section, we analyse the systemic effects of implementing the ZT modeling tool within the case, guided by the Viable System Model (VSM) [3]. VSM provides a framework for understanding the organisation as an integrated, living system, balancing both operational needs and strategic objectives. By mapping the tool's functions to the components of VSM, we can elucidate how it supports and enhances the organisation's ZT security posture.

The Viable System Model (VSM), developed by Stafford Beer, is a framework designed to understand and manage complex organisational systems. Rooted in cybernetics, the study of systems and their regulatory mechanisms, VSM provides a way to diagnose and design organisations to ensure their viability. At its core, VSM conceptualises any viable system, whether an organism, a machine, or an organisation, as consisting of five essential functions or subsystems. These subsystems are responsible for operations, coordination, control, intelligence, and policy. Table 1 gives an introduction to VSM subsystems 1-5 (column 1), what the subsystems need to handle when it comes to ZT and microservice development (2), the problem each subsystem exhibits (3), and how the presented tool helps each subsystem (4) based on its functions (5).

Notable, adding the collaborative functions to the tool enables the tool to also support the higher level subsystem of VSM. While the table outlines the benefits for the subsystems, some drawbacks can also be noted. The modeling tool, while beneficial in enforcing security protocols and aiding coordination (System 2), could due to the learning curve, temporarily disrupt daily operations (System 1), due to increased burden on developers, potentially slowing down routine tasks. Furthermore, if the tool is used for unneeded rigid implementation of ZT principles, which is not the case at the studied organisation, it might create resistance within development teams, affecting overall organisational coherence (System 5).

Table 1

Tool functions, mapped to VSM system 1-5

Viable System Model	VSM applied to ZT development.	ZTarchitecture issues/problems	How does the Tool help?	Tool functions
System 1 (Operational Units): Operational elements that carry out primary activities. Handles day-to-day operations, ensuring tasks are completed.	Development teams and architects develop microservices. Services are tested and integrated.	Implementing ZT can disrupt daily operations. Requires constant validation, potentially slowing down routine tasks and increasing operational overhead.	Tool helps to analyse the microservice design, ensuring adherence and timely detection of ZTdeviations.	Modeling of software components in DFDs. Support with a pre-defined list of standard components Support the identification of threats and mitigation controls.
System 2 (Coordination): Manages conflicts and coordinates between different System 1 units. Provides stability and short-term regulation.	Teams collaborate, share experiences and coordinate.	Difficult to keep consistent ZT policies across different teams. Inconsistencies complicate communication and coordination between teams.	Facilitates consistent security policy application across teams, ensuring unified ZT practices. The consistent policies allow for better communication between teams.	Standardised models (DFD) allow for the comparison and communication between teams. The use of the standard component library fosters a common understanding of threats.
System 3 (Control): Monitors and controls System 1 activities. Ensures accountability. Manages performance and optimisation.	The security teams oversee the development and instruct the development teams to solve security issues.	Monitoring complexity increases. Ensuring all development teams adhere to Zero Trust policies requires more sophisticated control mechanisms, which can become resource-intensive.	Automates oversight, flagging non-compliance. Streamlines the resource used for security reviews. Enables specification of Zero Trust standards to be upheld.	The fulfilment of ZT principles can be monitored. Allows continuous control and follow-up by assigning actions to be carried out by the development teams. By changing the component library and associated threats it is possible to instruct the teams to address new threats.
System 4 (Development/Planning): Focuses on the future. Plans for change, adaptation, and sustainability in the evolving environment.	Planning of new micro-services. Assessment of changes that need to be done.	Micro-services are quite complex, thus it is difficult to get an overview of the existing services. This makes it difficult to motivate and plan for changes.	The tool provides a good overview of the existing microservices and their security levels, which provides a good foundation for future changes.	Gives a list of ongoing and existing micro-services projects. Give the current status of the security efforts.
System 5 (Policy/Identity): Highest level of the system. Sets the organisation's purpose, values, and oversees all other systems	Forms a unified culture and understanding of ZTdevelopment.	ZTsecurity, due to the extra effort required, may create resistance within development teams.	Helps foster a shared and streamlined way of working according to ZTprinciples. Creates a culture of continuous awareness.	The tool provides a holistic and coherent set of functions. Thereby embodying the organisation's purpose of being a secure partner for transactions.

6. Conclusion

In this paper we presented a multi-user modeling tool designed to support the implementation of ZT security principles within a microservices architecture at a bank. The tool integrates data flow diagrams (DFDs) and STRIDE threat analysis, offering both semi-automated and manual security assessments. Through a case study at a bank, we demonstrated the tool's potential impact on various organisational levels, guided by the Viable System Model (VSM).

While the tool effectively structures the security team's work, and provides a structured approach to security, it also introduces challenges. The initial learning curve and the complexity of detailed security documentation need an initial effort in training. Despite challenges, the tool's ability to enforce consistent security policies and facilitate coordination across teams is a significant advantage. A sign that the advantages outweigh the initial learning curve is that the examined case organisation has continued using the tool, and its use is even widened as more and more of its software services are making use of the tool.

References

[1] N.F. Syed, S.W. Shah, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, Zero trust architecture (ZTA): A comprehensive survey. IEEE, 2022.

[2] A. Kerman, O. Borchert, S. Rose, and A. Tan, Implementing a zero trust architecture, National Institute of Standards and Technology (NIST), 2020.

[3] S. Beer, The Heart of Enterprise, Chichester: John Wiley & Sons, 1994.

[4] A. Wylde, Zero trust: Never trust, always verify, In 2021 international conference on cyber situational awareness, data analytics and assessment (cybersa), IEEE, 2021.

[5] Z. Adahman, A.W. Malik, and Z. Anwar, An analysis of zero-trust architecture and its cost-effectiveness for organisational security, Computers & Security, 122, 2022.

[6] S. Teerakanok, T. Uehara, and A. Inomata, Migrating to zero trust architecture: Reviews and challenges, Security and Communication Networks, 2021(1), 2021.

[7] N. Mayer, J. Aubert, E. Grandry, and C. Feltus, An integrated conceptual model for information system security risk management and enterprise architecture management based on Togaf, In The Practice of Enterprise Modeling: 9th IFIP WG 8.1. Working Conference, PoEM 2016, Skövde, Sweden, 2016. Springer International Publishing, 2016.

[8] R. Wirtz and M. Heisel, Model-based risk analysis and evaluation using CORAS and CVSS, In Evaluation of Novel Approaches to Software Engineering: 14th International Conference, ENASE 2019, Heraklion, Crete, Greece, 2019. Springer International Publishing, 2020.

[9] Microsoft, Threat Modeling Tool, Accessed 2024-05-12 URL: https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

[10] P. Johannesson and E. Perjons, An Introduction to Design Science, 2nd ed., Springer International Publishing, 2021.