The Threat Modeling Naturally Tool: An Interactive Tool Supporting More Natural Flexible and Ad-Hoc Threat Modeling

Ronald E. Thompson*, Madison Red*, Richard Zhang*, Yaejie Kwon†, Lisa Dang*,
Christopher Pellegrini∓, Esam Nesru‡, Mira Jain*, Caroline Chin* and Daniel Votipka*
*Tufts University; †Swarthmore College; ∓ Northeastern University;
‡ University of Maryland, Baltimore County

Abstract

Threat modeling is an important process in achieving secure-by-design software systems. While some tools have been developed to aid system architects in building threat models, many of these do not support the more flexible ways that threat modeling occurs in practice. We present the Threat Modeling Naturally Tool as the first step in providing architects with a tool that allows for a more natural threat modeling process that is modular in design. This tool consists of a threat modeling Domain-Specific Language and a series of modular components that allow users to specify their system and assign threats and mitigations without disrupting their brainstorming. We describe the design and implementation of our tool using a mock medical device as a case study as well as discuss how the tool can be used for future work supporting threat modeling research.

1 Introduction

Threat modeling is a structured process that helps system architects to brainstorm possible threats to their system and determine mitigations to identified threats [8, 19, 31]. As they review a system, architects might look at each element part-by-part [34], consider various user workflows [36], follow formal processes such as STRIDE [19] or LINDDUN [41], and/or utilize various other techniques to build threat models. For example, in a robotic surgical system, shown in Figure 1, an architect could analyze the workflow for a Surgeon (C) remotely guiding a Surgical Robot (1) to operate on a patient (B) in a different hospital. As the architect considers this interaction, they might realize an attacker could pretend to be the remote surgeon by spoofing their identity, thus allowing the attacker to control the robot and harm the patient. The architect would then decide to add two-factor authentication to prevent this type of threat.

USENIX Symposium on Usable Privacy and Security (SOUPS) 2024.
August 11–13, 2024, Philadelphia, PA, United States.

Threat modeling provides a formal process to help architects navigate these systems to ensure a thorough review. The US Cybersecurity and Infrastructure Security Agency (CISA), along with 17 other international partners, encourages threat modeling as a key aspect of mitigating the potential impact of exploiting software and cyber-physical systems [8]. Additionally, global regulators of medical devices and other security agencies are adding requirements for threat modeling activities [6, 7, 11, 15, 28].

There have been several tools and domain-specific languages that have been previously developed to aid system designers with threat modeling [3, 9, 12, 17, 18, 21, 23, 37]. However, prior research by Thompson et al. observed that system designers do not use these existing tools [36]. We expect this is because existing systems do not support the ad-hoc, flexible ways that Thompson et al. found architects review systems when threat modeling in practice [36].

We propose the Threat Modeling Naturally Tool (TMNT) to address this gap. TMNT is an interactive threat modeling tool designed to support the various ways architects perform threat modeling in practice. In particular, it is designed to allow free-flowing threat modeling on systems of varying stages of completeness. TMNT comprises five configurable components: A domain-specific language, a threat and mitigation knowledge base, a threat and mitigation identification engine, a natural suggestion engine, and a user interface. Figure 2 gives each component and relevant interactions. Together, these components support the representation of systems at varying specificity, threat, and mitigation identification, multimodal user interaction, and threat and mitigation suggestions tailored to match the users' focus. In this paper, we describe the design (Section 3) and implementation (Section 4) of TMNT, describing how it could be used in a specific case study scenario, a remotely operated surgical robot (as described in Section 2).

In addition to being a useful tool for practitioners, we believe TMNT will be a valuable resource for researchers

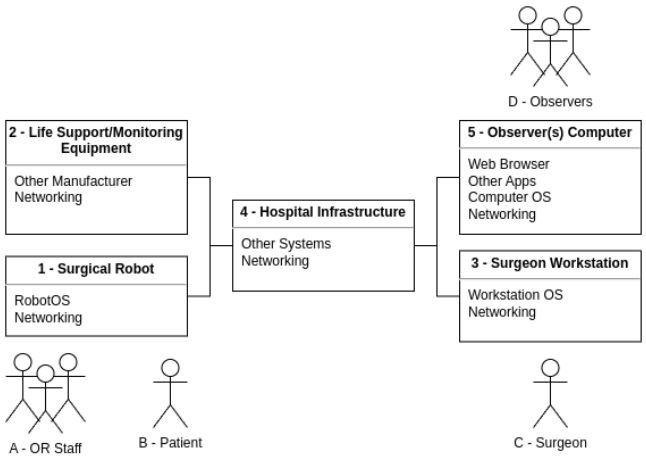


Figure 1: Surgical Robot system diagram. Numbers indicate system components. Letters indicate user groups.

studying the human factors of threat modeling. We also present directions for future research using TMNT. The tool, including the source code and examples, can be found at `tsp.eecs.tufts.edu/tmnt`.

2 Example Use Case: Surgical Robot

Before providing details of our tool, we present an example use case, which we will use as a motivating example throughout our discussion. Specifically, we consider a remote-operated surgical robot, drawn from Thompson et al.'s prior work, which was developed based on extensive discussions with professional medical device architects, who regularly perform threat modeling tasks [36]. A system diagram for the surgical robot is shown in Figure 1.

This example was developed to represent similar connected surgical systems, such as Intuitive's DaVinci [38] and the Auris Monarch [14]. These devices allow surgeons to perform surgeries remotely. Their complexity and connectedness introduce various network-based threats. Because exploitation of these devices can lead to patient harm, the FDA requires manufacturers to submit a threat model of these medical devices for regulatory review before they can be approved for patient use.

This device is made up of five key components. The Surgical Robot (Fig 1.1) is the device at the center of the patient interaction. It is located in the operating room (OR), contains the remotely controlled surgical device, which performs the physical operation, and sensors for real-time data collection. The data is then streamed to a central hospital server within the Hospital Infrastructure (Fig 1.4), which the surgeon (Fig 1.C) can connect to through their Surgeon Workstation (Fig 1.3) to control the robotic system remotely and receive a live video stream of the operation. The video stream and other data can also be shared with one or many other observing computers (Fig 1.5) where others (Fig 1.D) can observe the operation for learning purposes. Data from the surgical robot is also transmitted through the hospital infrastructure to patient monitoring equipment (Fig 1.2) and integrated with the hospital's Electronic Health Records (EHR).

Some potential threats to this system include the following:

- **Insider threats** could be at the primary hospital or any of the remotely connected sites. Potential threats would include manipulating system operations due to their role in the hospital, leading to patient harm or leaking of sensitive data.
- **Human error** could occur within all parts of the system. This would include users incorrectly manipulating surgical controls, providing unverified surgery observation access, or taking down the network for routine maintenance.
- **Malicious actors** could be located remotely or internal to any of the connected sites. For example, malicious actors might tamper with remote commands to the surgical robot to cause the device to malfunction during surgery or observe telemetry data transmitted between components, which could reveal sensitive patient data.
- **System failures**, especially those impacting the hospital infrastructure, could occur at any point. Possible threats would include power or network outages, which could degrade patient monitoring during critical care periods.

3 Tool Design

In prior work, Thompson et al. conducted threat modeling sessions with 12 medical device security architects [36]. In these sessions, Thompson et al. asked developers to develop threat models for two mock systems, similar to the surgical robot in Figure 1, and observed the order in which participants navigated system components and how they approached threat modeling for each component (e.g., threats first, mitigations first, etc.). They observed their participants did not follow a single process but instead varied their approach as they considered different elements of the system. They also found architects consider several potential system configurations as they attempt to plan for possible ways their system could be deployed in practice throughout its lifecycle of use. To provide a more natural threat modeling experience, we designed TMNT to support these practices. Specifically, we considered three primary design goals: supporting varied approaches, uninterrupted brainstorming, and threat modeling over incomplete information and alternative configurations.

Support varied approaches (D1). Because architects' navigate the system and consider threats and mitigations in several different ways, TMNT should support all possible approaches. This includes considering threats component-by-component or interaction-by-interaction, as suggested by some threat

modeling guides [29, 31, 35]; focusing on specific system use cases [36]; or considering centers of gravity [33]. The user should also be able to switch between these approaches freely. Naturally, the UI should be designed to allow users to focus on relevant elements of the described system. Additionally, TMNT should be able to identify potential threats and mitigations through the lens of different system views. For example, we would want to be able to simulate attacks both starting from or leading to the Surgeon Workstation (Fig 1.3), as opposed to only considering a single direction of attack.

Uninterrupted brainstorming (D2). While TMNT should be able to provide threat and mitigation suggestions to support a wide variety of threat modeling approaches, it must be strategic in how these threats are presented. Many current tools present the full list of suggested threats and mitigations to users. This produces overwhelmingly long lists that are hard for users to parse. Instead, we propose filtering suggestions to the users' current focus, engaging their natural tendencies and avoiding breaking their focus, which is common practice in supporting brainstorming [25, 27]. For example, if the architect reviewing our surgical robot is focusing on the use case of the surgeon (Figure 1.C) directing the robot (Figure 1.1) during surgery, then TMNT would only provide suggestions relevant to components involved in this workflow, i.e., the robot (Figure 1.1), hospital infrastructure (Figure 1.4), and surgical workstation (Figure 1.3).

Incomplete information and alternative configurations (D3). TMNT must allow the system model to have flexible representations. The architect might know there are many ways the system could be deployed or not know the exact nature of its deployment [9, 36]. When analyzing the system model for potential threats and mitigations, TMNT should consider these different possible system representations. This also has the added benefit of allowing architects to produce threat models as the system is being designed, supporting secure-by-design from system inception [8].

In addition to these primary threat modeling design goals, we considered two general design goals to support tool usability: a multi-modal interface and component modularity.

Evolving Interface (D4). Given that there is a learning curve to threat modeling and that users may have different levels of expertise, the interface should allow for "graceful evolution" [30, pg. 41]. Features like drag-and-drop components and the ability to click and toggle through various system views help build an "intuitive" design that architects can learn as they use [30]. Novice users will be presented with more specific instructions that will evolve to become interpretative with their experience [20]. As architects become more familiar with TMNT and threat modeling, they should also be able to switch to a view with more fine-grained control over the system representation.

Component Modularity (D5). To allow practitioners in different domains to tailor the system to their particular context, components should be developed modularly to allow users to swap in alternative versions [26]. For example, medical device architects might prioritize different types of threats than power suppliers, necessitating different threat suggestion engines. An architect can also be given the option to use modules for formal threat modeling processes like STRIDE [19] and LINDDUN [41]. Additionally, as more advanced AI methods become available, it may become useful to incorporate these techniques to identify novel suggested threats and mitigations. Finally, the knowledge base of threats and mitigations should be easy to update and extend to give architects access to the quickly growing security knowledge base as they threat model.

4 Implementation

We built TMNT with our five design goals in mind. TMNT consists of five components: a system representation in our domain-specific language (DSL), Fig 2.A; threat and control knowledge bases Fig 2.B; Assignment Engine, Fig 2.C; Natural Suggestion Engine Fig 2.D; and User interface (UI) Fig 2.E. Each component was designed for modularity, both for ease of development and to allow users the flexibility to tailor TMNT to their needs [26]. For each component, we describe its functionality, how it addresses our design goals, and describe how it would be used in the context of the running example from Section 2.

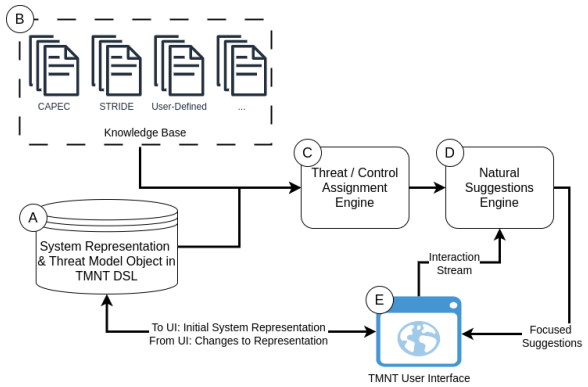


Figure 2: TMNT System Diagram.

4.1 Building a Threat Modeling DSL

As part of building TMNT, we needed a robust, flexible DSL to allow a complete representation of all objects in the threat model. This DSL should support the ability to analyze and verify a threat model and allow threat modeling components to be reused in later models [22]. While there has been prior work establishing DSLs for threat modeling [1–5, 18], none fully met our design requirements. Many of these DSLs only support a specific set of rules to apply threats and controls [1–3, 18], violating *D1*. Others only support a limited

set of threat annotations [4, 5], which does not lend itself to modularity (*D5*). Some require significant additional software development before they can be used [18], making it difficult for novices (*D4*). Rather than trying to modify one of these tools to fit our needs, we created our own DSL guided by our design goals and inspired by ideas from these existing DSLs.

In our DSL, a threat modeling object consists of a set of *Components* representing the system. These components can be *Elements*, which are objects in the system representation, *Threats*, or *Controls*. To allow for varied approaches (*D1*) and alternative configurations (*D3*), we included more specific types of *Elements* that help more readily differentiate between them, using the components of Data Flow Diagrams as a starting point, i.e., datastore, process, data flow, external entity, and trust boundary. We then added more levels of specificity that are associated with more complex system representations that DFDs cannot necessarily capture [10, 32] and are found in other threat modeling DSLs [3]. One of these was to create a *Flow* abstraction that is not necessarily a data flow, allowing users to capture more complex workflows that are relevant to the system [36]. Additionally, we allow the user to specify parent-child relationships between the elements, allowing them to "zoom" in and out depending on their needs. Expert users can directly manipulate the DSL representation of their system to allow fine-grained edits (*D4*), and objects in the language can be sub-typed, or other classes can be added, supporting modularity (*D5*)

Representing a system in the DSL (Fig 2.A). The TMNT DSL provides an overarching framework to describe a system on a macro level while having the ability to focus on specific aspects of the system in question and how they relate to one another. Establishing these relationships allows users to highlight important system workflows and see how threats might be connected [18]. As the user creates the various assets and information flows, they can specify the security properties (e.g., confidentiality, integrity, availability, non-repudiation, authorization, and authenticity) relevant to that specific component and the associations it has with other parts of the system.

The surgical robot example in the DSL. Next, we partially describe how the surgical robot example (Section 2) would be represented in our DSL. For reference, Figure 3 shows the associated YAML file produced in our DSL. The robot would be considered an *Asset* located within the operating room (OR) trust boundary. The user can also indicate the security properties the asset must maintain, which can be used by the Natural Suggestion Engine (Fig 2.D) to prioritize threats/controls based on these properties. This example prioritizes integrity and availability while setting the bar for confidentiality guarantees slightly lower. This information is useful as we assess potential threats against the system, helping architects determine whether they might accept those threats if they do not violate a high-priority property. In this

```yaml
name: Surgical Robot
type    : Asset
trust_boundaries : OR
security_properties :
    confidentiality  : LOW
    integrity : HIGH
    availability : HIGH

name: Surgical Procedure Execution Flow
type    : Workflow
src :
    name : Surgeon Workstation
dst :
    name : Surgical Robot
path :
    name : Hospital Infrastructure
controls :
    authentication : Certificated-based
    multifactor : true
```

Figure 3: Example for how the surgical robot and the surgeon's workflow can be specified for the DSL using YAML.

case, the user has not specified everything about this asset. However, this is sufficient for TMNT, which does not require a complete specification and will provide suggestions based on the limited information given (*D3*). The user then would go on to specify the other components.

Users can also specify associations of assets, such as a workflow. In this example, the user wants to consider the workflow between the robot and the surgeon's workstation through the hospital infrastructure. This is defined through a Workflow type object (the bottom item in Figure 3), which includes each associated asset, security control, and multi-factor certificate-based authentication.

4.2 Enumerating Threats and Controls

The DSL also includes *Threats* and *Controls* abstractions but does not dictate specific examples, such as Spoofing. Instead, TMNT uses a knowledge base of threats and controls and rules that assign these threats and controls based on system specifications, such as the component type.

Building a Knowledge Base (Fig 2.B). Many other tools hard-code the threats and controls into the application itself, relying on the creator's knowledge and potentially leveraging databases such as MITRE's Common Attack Pattern Enumerations and Classifications (CAPEC) [24]. In our initial prototype of TMNT, we used CAPEC in addition to custom threats and controls, which we found for our examples. However, users can add their own threats and controls references specifying relationships and requirements based on the DSL.

Threat and Control Assignment (Fig 2.C). Using the knowledge base(s) selected by the user, TMNT assigns these threats and controls based on the system specified by the user and as

the system's representation is updated, keeping with our goal to allow for incomplete information (*D3*). For our prototype, we are using deterministic rules from CAPEC and our own custom threats and controls. Still, TMNT can leverage probabilistic rules based on historical examples of threat models and/or additional rules from alternative sources. Again, this allows TMNT to be customized based on the user's needs (*D4*) and ensures modularity (*D5*).

4.3 Defining a Natural Suggestion Engine (Fig 2.D)

Once a set of threats and controls have been assigned, TMNT must present this information to build on what the user has already done (*D1*) and not interfere with the user's workflow (*D2*). Other tools provide a list of threats based on the system representation once the user indicates they are done, which produces a long list of suggestions that can be hard to navigate. TMNT's goal is to minimize this by instead asking the user what they would like to work on next, whether that is thinking of more threats for the component they are focused on or applying the same control to other parts of the system. This filters the total list of suggestions produced by the Threat/Control Assignment Enginge, showing only the partial list that is relevant to the users' focus. As the user progresses through the threat modeling session, more of the total suggestions list will be presented as their focus changes.

To do this, TMNT uses the list of all the threats and controls that have been identified by the Assignment Engine (Fig 2.C) in conjunction with the user interaction stream that comes from the UI itself (Fig 2.E). Using various heuristics based on the user's prior interactions, the system's representation, and the potential assignments, TMNT will dynamically recommend potential paths for the user.

Recommendations for the surgical robot. We show the potential threats and controls that would be suggested for the hospital's server from our example system. The user can see the potential threats flagged based on what is known (Fig 4a), or they could review the controls without considering threats initially and select controls they would expect to use (Fig 4b). As the user is doing either of these actions TMNT updates the threat model to ensure that controls are not suggested for threats that do not apply and threats are considered mitigated if an appropriate control is selected.

4.4 Designing a User Interface (Fig 2.E)

The final part of TMNT is the UI, which is how users primarily interact with the tool. The UI design requires it to receive suggestions from the Suggestion Engine and present these in a way that does not disrupt the user's workflow (*D2*). Users can use a menu interface (seen in Fig 4b and on the left side of Fig 4a) and direct manipulation via drag-and-drop, depending on their preferences. The menu provides a more verbose, structured view, which can help guide novice users through the interface [20]. Conversely, the drag-and-drop direct manipulation option allows users already familiar with the tool to move quicker through the tool as it reduces the number of clicks necessary for common tasks. [20]. The drag-and-drop interface uses icons found in other threat modeling tools to provide a consistent experience [12, 17, 21]. We used the four questions for threat modeling [31, 43], part of a framework developed by industry experts, as a way to group the different menus of activities. Each is described below.

What are we building? The user can add elements to the system representation through menu options or direct manipulation. The user can also upload a YAML file specifying the system and any identified threats or controls, as partially shown in Figure 3. We expect users could develop other parsing tools for direct integration into the architect's workflows.

What could go wrong? The user will see a list of potential threats for a selected element, see Fig 4a; if no element is selected, the user is presented a selection menu for elements they would like to review. The user then selects which threats are relevant. If the user wants to address the threat, they can select a control to apply, and the threat will be marked resolved in the summary bar at the bottom of the UI (Fig 4c).

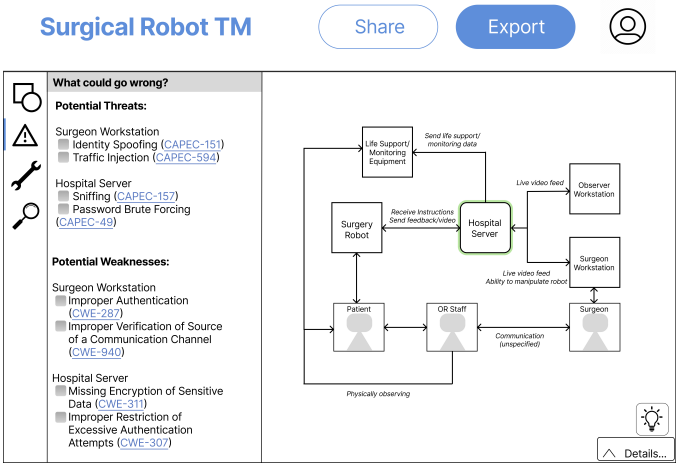
What are we going to do about it? The control assignment mirrors threat assignment, whereby the user select the controls they want for a specific element. They then confirm which threat it addresses, which will help draw out implicit assumptions being made, which are common in threat modeling [40]. The user can select other elements using the same control, add additional controls to that element, or switch views.

Have we done a good enough job? In this last menu option, the user can map controls to threats. The user will evaluate the residual risk for each finding, which maps a threat(s) to a control or a set of controls to a threat. This is done using the questions from NIST 800-30 [16], but it could also be updated based on the user's risk assessment framework.

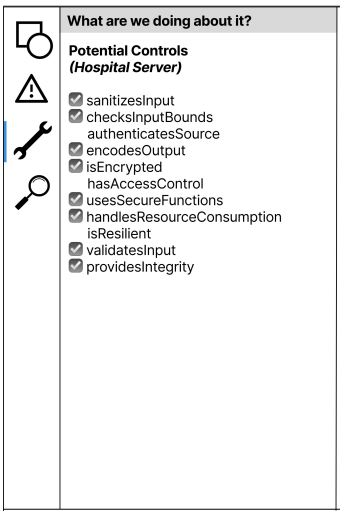
As the user makes threat model edits, the system representation (Fig 2.A) will be updated, and the user will be able to save and export the threat model as YAML files linking system components to identified threats and controls. Additionally, user interactions are logged by the UI and given to the Natural Suggestion Engine so it can provide recommendations based on user focus, as well as what they historically have focused on (and not focused on) to provide suggestions that allow the user to create a robust threat model.

5 Discussion

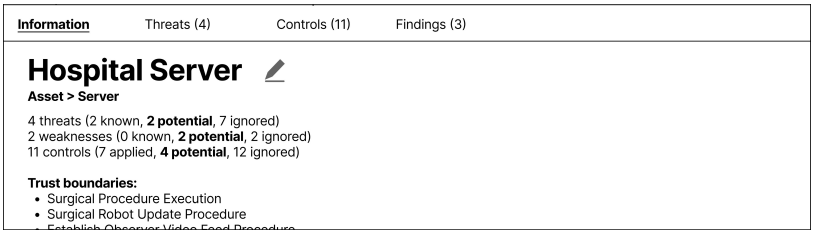
There have been several tools developed to support threat modeling [3, 12, 17, 18, 21, 23, 37], but none were developed to match the process real users follow when threat modeling [36]. TMNT represents a first step toward a human-centric threat modeling tool, giving practitioners the ability to threat model



(a) Suggested potential threats to the hospital's server



(b) Suggested potential controls that could be applied to the server



(c) Current information known about the specific element selected

Figure 4: The UI for TMNT when analyzing the surgical robot. The threats (4a) are both for the server and other components that could affect the server based on the workflows specified. The controls (4b) could be used on a server and would address some of the potential threats suggested. The server summary (4c) is shown below the menus and drag-and-drop.

in a way that better fits their actual practice. Through its flexible DSL and natural suggestion engine, TMNT supports tractable navigation, threat and mitigation suggestions that do not break user focus, and multiple and incomplete system representations. The tool is designed to be more usable with multiple modes of interaction to support users at different levels of expertise and uses a modular architecture to allow user customization.

5.1 Future Work

However, TMNT is not an end-state human-centered threat modeling tool. We have designed the tool based on prior works' findings, but that work was on a small number of participants in a specific domain (i.e., medical devices). The question remains whether those results generalize and whether TMNT is actually usable and useful in practice. We plan to conduct the expected future usability testing to assess TMNT. However, we note that TMNT offers an additional benefit to the research community as a platform enabling the measurement of user behaviors during threat modeling. This creates new opportunities for threat modeling research using TMNT beyond the evaluation of TMNT. We conclude with a short research agenda discussion of our plans for future work using TMNT to understand *how people threat model in practice*.

Research Agenda: How do people threat model? TMNT allows fine-grained tracking of user behaviors. We can measure which objects users add, how they update them, what suggestions they consider, and many more interactions with the interface. Because this is built into the tool and can be collected whenever TMNT is used without researcher supervision, we can scale user observation beyond the small sample in prior work [36]. This will allow us to determine whether the process model we built TMNT on actually generalizes. For example, we can determine what types of objects users consider most often, what navigation methods are most common, and what events precipitate a change in method.

Additionally, we expect threat modeling processes will likely differ between experts and beginners. We can identify the differences by collecting usage data from members of both groups. This will allow us to understand what "professional vision"—the idea that experts know where to look and develop unique mental structures of information based on their experience [13]—looks like in threat modeling. Once

we know how experts differ, we can develop education to help train beginners' "professional vision" and add suggestions or visual guides in the tool that help users develop expertise [42]. Conversely, it is also possible experts may develop blind spots as they focus on certain aspects of a system based on prior experience, skipping other components that historically are not an issue [39, pg. 98]. By comparing experts and beginners, we can identify challenges for each group and update TMNT to help them address their biases.

References

[1] threagile: Agile Threat Modeling Toolkit. https://github.com/Threagile/threagile.

[2] AT-AT. https://github.com/yathuvaran/AT-AT, March 2024. original-date: 2021-10-13T21:20:29Z.

[3] pytm. https://github.com/izar/pytm, May 2024. original-date: 2018-05-14T23:16:07Z.

[4] threatcl. https://github.com/threatcl/threatcl, May 2024. original-date: 2021-09-14T23:15:12Z.

[5] threatspec. https://github.com/threatspec/threatspec, May 2024. original-date: 2019-06-16T21:45:19Z.

[6] Agence nationale de la sécurité des systèmes d'information. SecNumedu-FC/EBIOS Risk Manager – Lancement d'un nouveau référentiel pour la formation continue dédiée au management du risque numérique, 2019.

[7] Center for Devices and Radiological Health, Food & Drug Administration. Cybersecurity in Medical Devices: Quality System Considerations and Content of Premarket Submissions, 2023.

[8] CISA, NSA, FBI, ACSC, NCSC-UK, CCCS, BSI, NCSC-NL, CERT NZ, and NCSC-NZ. Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-by-Design and -Default. Technical report, 2023.

[9] Shamal Faily and Claudia Iacob. Design as code: Facilitating collaboration between usability and security engineers using cairis. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pages 76–82, 2017.

[10] Shamal Faily, Riccardo Scandariato, Adam Shostack, Laurens Sion, and Duncan Ki-Aries. Contextualisation of data flow diagrams for security analysis. In *Graphical Models for Security: 7th International Workshop, GraMSec 2020, Boston, MA, USA, June 22, 2020, Revised Selected Papers 7*, pages 186–197. Springer, 2020.

[11] Federal Office for Information Security, Medical Engineering Division of the German Electrical and Electronic Manufacturers' Association, and Federal Institute for Drugs and Medical Devices. Cyber Security Requirements for Network Connected Medical Devices, 2018.

[12] OWASP Foundation. OWASP Threat Dragon. https://owasp.org/www-project-threat-dragon/, 2020.

[13] Charles Goodwin. *Professional Vision*, pages 387–425. Springer Fachmedien Wiesbaden, Wiesbaden, 2015.

[14] Chauncey F Graetzel, Alexander Sheehy, and David P Noonan. Robotic bronchoscopy drive mode of the auris monarch platform. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3895–3901. IEEE, 2019.

[15] Health Sciences Authority. Regulatory Guidelines for Software Medical Devices – A Life Cycle Approach, 2022.

[16] Joint Task Force Transformation Initiative. Guide for Conducting Risk Assessments. Technical Report NIST Special Publication (SP) 800-30 Rev. 1, National Institute of Standards and Technology, September 2012.

[17] IriusRisk. Threat Modeling Platform. https://www.iriusrisk.com.

[18] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. A meta language for threat modeling and attack simulations. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–8, 2018.

[19] Loren Kohnfelder and Praerit Garg. The threats to our products. *Microsoft Interface, Microsoft Corporation*, 1999.

[20] Kai H. Lim, Izak Benbasat, and Peter A. Todd. An experimental investigation of the interactive effects of interface style, instructions, and task familiarity on user performance. *ACM Trans. Comput.-Hum. Interact.*, 3(1):1–37, mar 1996.

[21] JGraph Ltd. draw.io.

[22] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, dec 2005.

[23] Microsoft. Microsoft Threat Modeling Tool overview. https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool, 2022.

[24] MITRE. CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC™), 2007.

[25] Bernard A. Nijstad and Wolfgang Stroebe. How the group affects the mind: A cognitive model of idea generation in groups. *Personality and Social Psychology Review*, 10(3):186–213, 2006. PMID: 16859437.

[26] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12):1053–1058, dec 1972.

[27] Paul B. Paulus and Vincent R. Brown. Toward more creative and innovative group idea generation: A cognitive-social-motivational perspective of brainstorming. *Social and Personality Psychology Compass*, 1(1):248–265, 2007.

[28] Pharmaceuticals and Medical Devices Agency. Guidance on Ensuring Cyber Security of Medical Devices, 2018.

[29] Nataliya Shevchenko. Threat modeling: 12 available methods. Carnegie Mellon University, Software Engineering Institute's Insights (blog), Dec 2018. Accessed: 2023-Mar-21.

[30] Ben Shneiderman and Catherine Plaisant. *Designing the user interface: strategies for effective human-computer interaction*. Pearson Education India, 2010.

[31] Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[32] Laurens Sion, Koen Yskout, Dimitri Van Landuyt, Alexander van den Berghe, and Wouter Joosen. Security threat modeling: Are data flow diagrams enough? In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, page 254–257, New York, NY, USA, 2020. Association for Computing Machinery.

[33] Rock Stevens, Daniel Votipka, Elissa M. Redmiles, Colin Ahern, Patrick Sweeney, and Michelle 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)*, pages 621–637, Baltimore, MD, August 2018. USENIX Association.

[34] Microsoft Software Development Lifecycle Team. The stride per element chart. https://www.microsoft.com/en-us/security/blog/2007/10/29/the-stride-per-element-chart/, 2007.

[35] Microsoft Software Development Lifecycle Team. The stride per element chart, 2007. https://www.microsoft.com/en-us/security/blog/2007/10/29/the-stride-per-element-chart/.

[36] Ronald Thompson, Madeline McLaughlin, Carson Powers, and Daniel Votipka. There are rabbit holes i want to go down that i'm not allowed to go down: An investigation of security expert threat modeling practices for medical devices. In *33rd USENIX Security Symposium (USENIX Security 24)*, Philadelphia, PA, August 2024. USENIX Association.

[37] ThreatModeler. ThreatModeler. https://threatmodeler.com/.

[38] Shawn Tsuda, Dmitry Oleynikov, Jon Gould, Dan Azagury, Bryan Sandler, Matthew Hutter, Sharona Ross, Eric Haas, Fred Brody, and Richard Satava. Sages tavac safety and effectiveness analysis: da vinci® surgical system (intuitive surgical, sunnyvale, ca). *Surgical endoscopy*, 29:2873–2884, 2015.

[39] Barbara Tversky. *Mind in motion: How action shapes thought*. Hachette UK, 2019.

[40] Dimitri Van Landuyt and Wouter Joosen. A descriptive study of assumptions in STRIDE security threat modeling. *Software and Systems Modeling*, 21(6):2311–2328, 2022.

[41] Kim Wuyts and Wouter Joosen. Linddun privacy threat modeling: a tutorial. *CW Reports*, 2015.

[42] Yu-Chun Grace Yen, Jane L E, Hyoungwook Jin, Mingyi Li, Grace Lin, Isabelle Yan Pan, and Steven P Dow. Processgallery: Contrasting early and late iterations for design principle learning. *Proceedings of the ACM on Human-Computer Interaction*, 8(CSCW1):1–35, 2024.

[43] Zoe Braiterman, Adam Shostack, Jonathan Marcil, Stephen de Vries, Irene Michlin, Kim Wuyts, Robert Hurlbut, Brook S.E. Schoenfield, Fraser Scott, Matthew Coles, Chris Romeo, Alyssa Miller, Izar Tarandach, Avi Douglen, and Marc French. Threat Modeling Manifesto.