PLAYBOOK FOR THREAT MODELING MEDICAL DEVICES

November 30, 2021

MITRE
SOLVING PROBLEMS FOR A SAFER WORLD®

MDIC
MEDICAL DEVICE INNOVATION CONSORTIUM

MITRE Sponsor: FDA

MITRE Contract No.: 75FCMC18D0047

MDIC Sponsor: FDA

MDIC Contract No.: 75F40119C10116

Contents

Acknowledgements

This playbook has benefited significantly from contributions and feedback made by numerous individuals and organizations since 2019. The authors are grateful to these contributors for their willingness to share their expertise and invest their valuable time to ensure that this playbook will be useful to the industry.

Organizational affiliations for individuals are provided based on the date of their contribution and may have changed by the time of this publication.

Playbook Authors (MITRE): Elaine Bochniewicz, Melissa P. Chase, Steve Christey Coley, Kyle Wallace, Matt Weir, Margie Zuk

Bootcamp Planning, Logistics, and Hosting (MDIC): Zafar Azam, Jithesh Veetil, Helen Wickstrom

Bootcamp Lead Trainer and Framework Author: Adam Shostack (Shostack & Associates)

FDA Leadership: Kevin Fu, Matthew Hazelett, Linda Ricci, Aftin Ross, Suzanne Schwartz, Jessica Wilkerson

Bootcamp Facilitators and Observers: Joan Adams-White (FDA), Seth Carmody (FDA), Julie Connolly (MITRE), Ronnie Daldos (MITRE), Stephanie Domas (MedSec), Brian Fitzgerald (FDA), Nicholas Gerteisen (Smith+Nephew), Sean Harrington (Abbott), Tyrone Heggins (BD), Daniel Heppner (Roche), Iacovos (Jake) Kyprianou (FDA), Tara Larson (Abbott), Ashley Mancuso (Johnson & Johnson), Charles Martinez (Beckman Coulter), Andrea Matwyshyn (FDA), Ju-Lie McReynolds (FDA), Colin Morgan (Apraciti), Chris Reed (Medtronic / Eli Lilly), Sudar Shields (Boston Scientific), Lisa Simone (FDA), Daniel Speck (Roche), Scott Van Eps (Beckman), Eugene Vasserman (Kansas State University), Charles Wilson (Motional), Beau Woods (I am the Cavalry)

Interview organizations: Abbott, Apraciti, BD, Beckman Coulter, Boston Scientific, MedISAO, Eli Lilly, GE Healthcare, Johnson and Johnson, Medtronic, Roche, Smith+Nephew

Finally, the authors thank the bootcamp participants who provided valuable suggestions to better understand the playbook's target audience and better tailor the contents accordingly.

1. Introduction

The opening of the "Threat Modeling Manifesto" [1] provides a succinct definition of threat modeling and why it has become a recognized cybersecurity best practice:

> *Threat modeling is analyzing representations of a system to highlight concerns about security and privacy characteristics. At the highest levels, when we threat model, we ask four key questions:*
>
> - *What are we working on?*
> - *What can go wrong?*
> - *What are we going to do about it?*
> - *Did we do a good enough job?*
>
> *When you perform threat modeling, you begin to recognize what can go wrong in a system. It also allows you to pinpoint design and implementation issues that require mitigation, whether it is early in or throughout the lifetime of the system. The output of the threat model, which are known as threats, informs decisions that you might make in subsequent design, development, testing, and post-deployment phases.*

Medical devices are increasingly complex and connected systems existing in complex connected ecosystems of healthcare delivery. Although standard lists of controls such as the National Institute of Standards and Technology (NIST) Special Publication (SP) 800-53 and ANSI/AAMI/IEC 80001 can ensure some baseline security capabilities, they fail to address the myriad of ways that medical devices are used, interface with the healthcare ecosystem, and most important, how security risks could result in unacceptable safety issues. Instead, for several years, the Food and Drug Administration (FDA) has recognized the value of threat modeling as an approach to strengthen the cybersecurity and safety of medical devices. To increase knowledge and understanding of threat modeling throughout the medical device ecosystem, FDA engaged with MITRE, the Medical Device Innovation Consortium (MDIC), and Adam Shostack[1] to conduct a series of threat modeling bootcamps and develop a playbook based on the learnings from those bootcamps.

The playbook is not prescriptive in that it does not describe one approach to be used when threat modeling medical devices. It is intended to serve as a resource for developing or evolving a threat modeling practice. The playbook is agnostic about specific methodologies, and instead focuses on the values and principles articulated in the manifesto and illustrates how different methodologies can be used, alone or in combination, to answer those four key questions.

The playbook provides a foundation that can inform an organization's threat modeling practices. The playbook provides insights on how an organization can develop or evolve an approach to creating threat models in a systematic and consistent way to achieve those objectives.

1.1. Playbook Development

The playbook was developed through learning from a series of threat modeling bootcamps designed by FDA, MDIC, MITRE, and Adam Shostack as the expert trainer, along with additional interviews of organizations that helped in the execution of the bootcamps.

The goal of the bootcamps was to scale existing threat modeling training to the medical device ecosystem by training large groups, with facilitators leading breakout activities. The bootcamp participants could then become ambassadors for threat modeling in their organizations. Even with these larger bootcamps, the number of people who could participate would be limited, and so this playbook was developed to serve as a threat modeling resource that would be widely available for use throughout the ecosystem.

[1] Adam Shostack is President of Shostack + Associates, a member of the Threat Modeling Manifesto Working Group and author of *Threat Modeling: Designing for Security* [2], a comprehensive overview of the subject.

The bootcamp series hosted by MDIC began with a “Train the Trainers” session in February 2020, where a group of 20 experienced threat modelers were trained to be facilitators for follow-on bootcamps for larger groups. The facilitators participated in a two-day face-to-face training session and a third day hotwash to critique the class and propose adjustments to better serve an audience of medical device manufacturers (for example, to include discussions relating threat modeling to medical device cybersecurity risk management).

The second bootcamp was conducted virtually in February 2021. In addition to the materials presented in the first bootcamp, a supplemental session was added to each day that focused on the medical device example. Chapter 2 of this playbook presents the end-to-end threat modeling process using the fictional example of a monitoring device used in a home setting that was used during the supplemental sessions. The exercises done in the supplemental session followed the homework of the previous day, so the learnings would be operationalized in a medical device-specific application. Following feedback, two additional fictional devices were developed that are featured in 4.Appendix A to demonstrate the application of threat modeling techniques to a diagnostic device in a hospital environment and a therapeutic device.

In addition to the bootcamps, the MITRE team conducted stakeholder interviews with the facilitator organizations to better understand the state of threat modeling in the medical device industry. Manufacturers described their current practices and provided insights into strategies for implementing threat modeling into business practices, tools, and methodologies, and scaling threat modeling throughout the enterprise. Chapter 3 captures the individual opinions and feedback of participants, including discussions during the bootcamps and complementary activities.

1.2. Using the Playbook

The playbook can be used as a resource for threat modeling training within an organization. Individuals can work through the examples, filling in the details left to the reader, applying the different methodologies discussed in the playbook to those gaps, and researching additional approaches using the references in the playbook as starting points. An organization could develop its own training using the playbook as a basis.

The playbook can also be used to educate stakeholders on threat modeling: what it is, its role in improving product safety and security, and how it fits with quality processes. For example, the playbook may help:

- product line managers understand how threat modeling fits into existing processes;
- systems engineers to understand how threat modeling informs design requirements;
- design engineers and architects understand how threat modeling informs design choices;
- design verification and validation (V&V) engineers understand how to use threat models in designing test strategies;
- regulatory specialists understand how to present and document threat models; and,
- contract manufacturers and consultants who may not be experienced in threat modeling.

Each of these stakeholders can select the portions of the playbook that can help them fulfill their roles and responsibilities in making their devices safe and secure.

2. Threat Modeling Overview

A well-developed threat model will document how a system is intended to function, justify trade-offs made in the design process, identify remaining threats to the system, and explain what mitigations are in place against them. This encompasses both the premarket development phase of the system and the postmarket maintenance phase.

A challenge with developing a mature threat modeling process is that there is no one-size-fits-all approach to threat modeling. Identifying the threat modeling techniques that fit an organization's team and products is a learning process that only improves with practice. As stated earlier, this Playbook is not meant to be prescriptive. Instead, the goal of Chapter 2 is to introduce concepts and techniques to aid in identifying threat modeling approaches that may be helpful. This content is not meant to be read passively, but instead serve as a starting point for additional research. The playbook provides three fictional medical devices to threat model (see Section 2.1, 4.Appendix A), but its concepts are applicable to other systems. As organizations adopt proactive threat modeling techniques, more questions will arise. When they do, this playbook and the concepts it details will provide a basis for selecting more advanced resources and tailoring the techniques for individual organizations' needs and products.

2.1. Fictional Example: The Ankle Monitor Predictor of Stroke (AMPS)

This section contains a high-level feature overview document for the Ankle Monitor Predictor of Stroke (AMPS) system. This system is a home use medical device for a fictional stroke diagnostic technique. While this example was fabricated to avoid focusing on specific clinical treatments, the overall scenario was constructed to highlight typical features encountered when threat modeling medical devices. This example will be used throughout Chapter 2.

The Ankle Monitor Predictor of Stroke System:
AMPS is a home use medical device worn at night (or when resting) by patients considered at risk for a stroke. The AMPS system gathers medical readings that can be later analyzed by a medical professional. While the system can help predict a patient's risk of experiencing a stroke, it does not alert—and is not intended to alert—if a stroke is imminent or occurring.

- Period of expected use: One to three months
- Medical capability: Diagnostic only
- Device invasiveness: Low (easily removable, like a wristwatch)

AMPS Core Use Case:
Alice has been informed by her doctor, based on her family history and several other risk factors, that she is at increased risk of experiencing a stroke. To gain further insight and determine a treatment plan, her doctor has instructed her to take the AMPS system home and wear it when she sleeps to take readings. She is also directed to install a companion app on her phone that will connect to the AMPS system (via Bluetooth) and upload the readings every day to the AMPS cloud service, where they will be analyzed by an automated algorithm. Alice's doctor will check the results after the first week to identify any immediate causes of concern, and they will schedule a follow-up consult in two months.

AMPS Core Technology:

- A Bluetooth Low Energy (BLE)-enabled ankle monitor that takes physiological measurements from the patient
- A phone/tablet application (app) for patients to pair with their ankle monitor that will display readings and communicate with the cloud services
- AMPSCS: The Ankle Monitor Predictor of Stroke Cloud Service

AMPS device:
AMPS is a health monitoring system worn on a patient's ankle when they are resting. It has the following specifications and capabilities:

- Weight: 0.13kg
- Power source: Lithium-ion battery recharged via universal serial bus (USB) C cable. Provides up to 96 hours of usage under normal circumstances
- On/off switch
- Physical Bluetooth pairing button
- Proprietary stroke-predicting sensor. Note: This is a fictional sensor that requires contact with a patient's skin.
- Heart rate monitor
- Body temperature sensor
- Bluetooth Low Energy (BLE) connectivity
- Onboard computer and flash storage that can store up to two weeks of patient data for later transmission

Patient App:
There are two different versions of the patient app, one for Apple iOS, and another for Android devices. Both apps contain the following functionality:

- The app is downloaded by the patient via Google Play or the Apple app store.
- It can pair with the AMPS device via Bluetooth.
- It contains an interface for a patient to create an account with the AMPS cloud services, register an AMPS device, and authorize clinicians to view their data.
- If the patient gives permission to the app, it will automatically connect to the AMPS device once a day and upload readings to the AMPSCS. If the patient does not give it permission, the app will store the data retrieved from the AMPS device until a manual upload is initiated. The amount of data transferred per upload is typically less than 1 megabyte a day.
- The app will display status information to the patient, including the last time the app synced with the AMPSCS, a log of the days the app was able to pull data from the AMPS device, and a log listing if the AMPS device was successfully collecting data.
- There is a device management screen that primarily focuses on diagnosing Bluetooth connection problems, and common issues that may prevent the AMPS device from collecting data. In addition:
    - The app can wipe patient data from the AMPS device.
    - The app can check for and update the firmware of the AMPS device with new versions.
    - The app can revert the AMPS device to factory default settings.
- If the device does not successfully sync to the cloud services once every 24 hours, an in-app notice will appear directing the patient to sync their data. After 72 hours have elapsed since a successful sync, the patient will be emailed an automatic reminder.

AMPS Cloud Service:
The AMPSCS is a collection of virtual machines hosted in a cloud infrastructure. It consists of the following functionality:

- An application gateway server to inspect and limit traffic going into the AMPSCS systems
- A set of backend services that perform analysis of the patient data
- A collection of patient-facing services that communicate with the patient app, provide a web portal for patients to register their AMPS device, and authorize clinicians to view their data
- A collection of health delivery organization (HDO)-facing services that provide a web portal for clinicians to create an account and access a patient's data

- Clinicians' access to the portal using a web browser.
- Authentication is provided via username and password.
- Clinician service identifiers that clinicians can provide to patients so the patients can authorize them through the app.
- The clinicians can view a summary of the patient's raw data and the analysis performed by the AMPSCS backend algorithms.
- The ability for clinicians to download a patient's data via an encrypted zip file.

2.2. The Four Questions (Overview)

The Four Questions Framework is a set of questions developed by Adam Shostack to add structure to the threat modeling process [2]. The Four Questions are as follows:

- "What are we working on?"
- "What can go wrong?"
- "What are we going to do about it?"
- "Did we do a good job?"

Keep in mind that fully answering each of these questions will likely be an iterative process. While the following sections will tackle each question in turn, when performing threat modeling on a real system, it is natural to skip back and forth between these questions, using the answer from one question to modify previous answers, and incorporating previous threat modeling efforts to skip past some of the work of other questions. Compounding this, rarely do the devices being threat modeled remain static. Systems are patched, new vulnerabilities are discovered, and new features are added. Even if the system itself does not change, a product team's knowledge of it may become more complete over time.

2.3. Question 1: What Are We Working On?

To help answer the first question, this section focuses on structured brainstorming and modeling techniques for diagramming medical devices that are being threat modeled. While unstructured brainstorming can certainly be productive, using a more structured approach can help identify gaps in an understanding of the system that might not be apparent with a more ad-hoc approach. More importantly, using a structured methodology can make it easier to share results with others who are also familiar with these techniques. This can give others confidence in the findings of an organization's threat modeling process and help them identify gaps that may exist. It is important to think beyond transferring this knowledge to external auditors. This knowledge is also useful for future work involving the product, new team members, and other teams inside an organization. Moreover, threat modeling is very rarely a singular activity that occurs once. Using a formal technique allows for later reviews and updates of threat models as new features are added and products evolve.

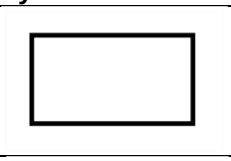
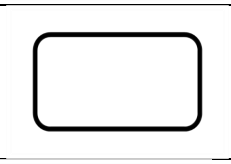
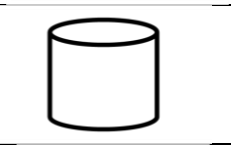
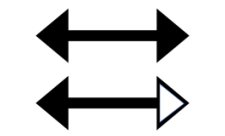
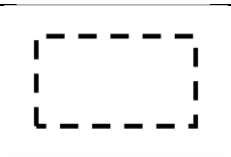
2.3.1. Structured Modeling with Data Flow Diagrams

Data flow diagrams (DFDs) are one way to help visualize the system that is being threat modeled. At a high level, DFDs are a way to represent the entities involved with the functioning of the medical device, how those entities are related, and the assumed trust boundaries between them. There are many different standards for designing DFDs, but for the purposes of this playbook, the DFD3 notation documented at https://github.com/adamshostack/DFD3 [3] will be used.

2.3.1.1. DFD Objects

One advantage of the DFD3 standard is the limited number of icons it contains, which can help to facilitate brainstorming sessions. In the core DFD3 standard, there are just five icons. Their description in Table 1 is taken from the standard at [3].

Table 1: DFD3 Icons

Element	Symbol	Discussion
External Entity		**Object:** A sharp-cornered rectangle. **Represents:** Anything outside your control. Examples include people and systems run by other organizations or even divisions.
Process		**Object:** A rounded rectangle. **Represents:** Any running code, including compiled, scripts, shell commands, Structured Query Language (SQL) stored procedures, et cetera.
Data Store		**Object:** A drum. **Represents:** Anywhere data is stored, including files, databases, shared memory, cloud storage services, cookies, et cetera.
Data Flows		**Object:** A double-headed arrow. **Represents:** All the ways that processes can talk to data stores or each other. If a conversation is only initiated by one side, you can represent the initiating side as an empty arrow.
Trust Boundary		**Object:** A closed shape drawn with a dashed or dotted line. **Represents:** A way to display different trust levels between objects.

While the DFD3 standard includes only five icons, DFDs may include additional icons if they help with brainstorming and conveying information. Below are some tips for constructing a DFD.

- Because DFDs may be printed out in physical copy, best practices suggest that DFDs not depend on the use of color. Color can be used to convey additional information, but the goal is to have diagrams that remain understandable if printed out in black and white.
- To enhance readability, it is useful for all elements to have a label.
- Objects can be grouped together and nested inside each other. For example, there may be several processes and data stores inside a single entity.
- For data flows, it is useful to use a double-headed arrow, as traffic usually flows in both directions. If it is useful to indicate which side initiates the connection, this can be represented as one arrowhead filled, with the initiating side open. This will still indicate that responses are passing back to the initiating side. If the traffic is truly one way, for example using the User Datagram Protocol (UDP) transport network protocol, use a one-way arrow.
- Trust boundaries will be described further in Section 2.3.1.3, but best practices suggest that they be drawn as a complete shape and not a line or an arc. The shape will clearly highlight what lies inside the trust boundary and what is outside of it.

2.3.1.2. (Example) Initial Brainstorming and Modeling the AMPS device with DFDs

Using the AMPS device as an example, one method to start brainstorming is to quickly identify the major components of the device. Referring to the original specification of the AMPS device in Section 2.1, we can see that three main components were referenced: the AMPS device, a companion app that resides on the patient's cell phone, and the AMPSCS. This is depicted in a DFD in Figure 1. The initial brainstorming does not have to be complicated. For example, Figure 1 captures that the AMPS device does not directly talk to the AMPSCS.

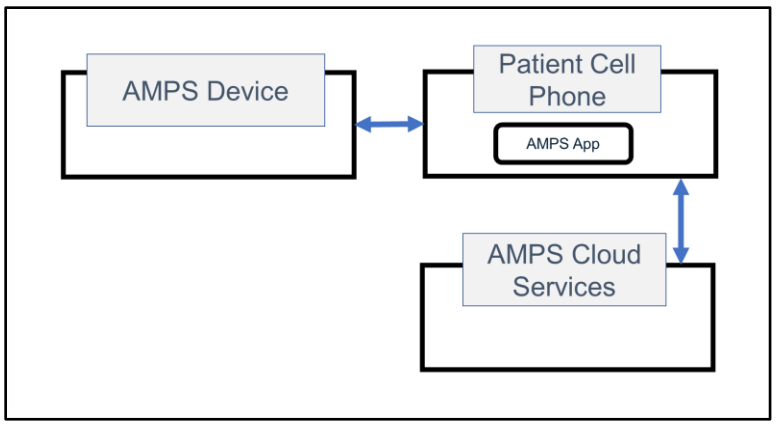


Figure 1: Example High-Level Diagram of the AMPS Device

The next step is to continue to build out the DFD by adding information to the individual components, as well as identifying other entities that may be relevant to the threat modeling process. Figure 2 is one example of a more detailed DFD for the AMPS system. Notice that the clinician's computer and the phone app store were added to the DFD. While these devices are likely outside the control of the product team, recording external entities such as these can be useful for later stages of threat modeling, since they interact with the device being analyzed.

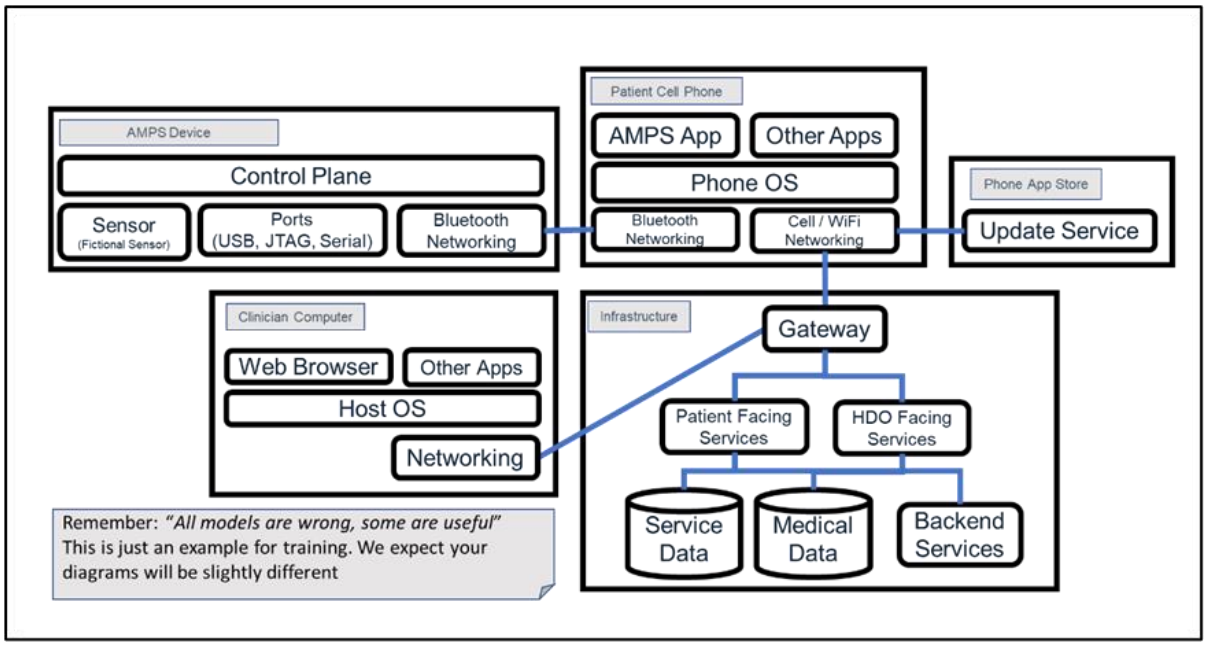


Figure 2: Example Mid-Level DFD for the AMPS Device (No Trust Boundaries)

When it comes to identifying the individual processes and modeling them in the DFD, there is no set procedure or hard rule for the techniques. As organizations continue to mature their threat modeling processes, they will be able to identify processes and entities missing from initial DFDs which may be helpful for identifying and understanding threats. Likewise, organizations may identify design decisions of DFDs which do not add significant value. Therefore, the more experience organizations gain in threat modeling, the easier creating DFDs becomes. Likewise, initial diagrams may always be improved as knowledge about relevant systems increases. Some of the considerations behind the design process of the DFD in Figure 2 are listed below:

- Different communication paths between the devices in the AMPS ecosystem were captured and labeled. While network communication paths were initially highlighted, and lines were drawn between the components, other non-standard interfaces were identified. For example: Joint Action Test Group (JTAG) and serial ports. Examination of the figure may reveal that physical ports do not have any lines connecting them to other components. One future improvement to this diagram may be to include and document external entities, such as a medical technician or a factory technician/process, that uses those communication paths. That way, they are not accidently overlooked in a later stage of the threat modeling process.
- Labeling how the communication occurs (e.g., Bluetooth, Wi-Fi, etc.) can also be useful for later stages of threat modeling.
- It can also be helpful to diagram how the different communication options interact with each other, especially with regards to medical devices. This is because questions such as "if the Bluetooth networking component crashes, what does it impact?" often come up in later stages of the threat modeling process.
- It is often helpful to reference external standards and guides when figuring out how to group and label different components. For example, much of the infrastructure labeling in Figure 2 was written to match terminology found in NIST SP 1800-30, "Securing Telehealth Remote Patient Monitoring Ecosystem" [4]. However, it remains important that organizations' models represent how their systems are architected and designed.
- The system being threat modeled may have multiple development teams. For example, the cloud infrastructure team may be different from the mobile app team, which in turn may be different from the team developing the physical medical device. A reasonable approach is to start the threat modeling process at a top-level system view with key participants and engineers participating, with follow-on sessions focusing on individual teams and their components. Documentation may follow a similar approach, with detailed threat modeling results residing in component-specific documentation (such as a design history file), while also maintaining system-level threat modeling records that reference those subcomponents. It is important to develop procedures that meet the needs of the organization performing threat modeling, while ensuring the objectives of threat modeling are accomplished.
- Threat modeling is an iterative process. When doing additional threat modeling activities, gaps will likely be identified in DFDs that can then be refined.

2.3.1.3. Trust Boundaries

Trust boundaries are a slightly different concept in a DFD than the other components such as databases, processes, and entities. That is because trust boundaries do not physically reside in a given organization's system, but instead represent ideas and assumptions being made by the threat modeling team about how different entities interact. Trust boundaries help in later stages of the threat modeling process by identifying areas that require enhanced investigation. They also help capture the thought process of the threat modeling team and can be used to help convey that information to external reviewers. It is important to note that trust boundaries are used in many modeling techniques other than DFDs and represent a more universal concept. While the descriptions here primarily focus on applying trust boundaries to DFDs, the idea of trust boundaries will be referenced throughout this Playbook.

Like many other aspects of threat modeling, knowing how to draw trust boundaries is a skill that improves with practice, and there is not one correct answer for how to do it. Below are a few helpful tips:

- Trust boundaries reside between entities where there is a desire to enforce controls on how they interact. These entities can be processes, devices, users, or systems.
- The interaction between entities may not be part of a standard business process. For example, physical fences are a type of control that can regulate traffic. Legitimate users may not be expected to cross a fence, but it is still important to document these controls as trust boundaries.

This is because an adversary may scale or bypass them. Remember, adversaries do not have to follow normal processes.

- As a rule of thumb, where control mechanisms exist—such as a firewall, access control list, lock, or login—it may represent a good place to draw a trust boundary.
- If code or data goes to an external entity's device, or their code or data comes to a modeled device or system, it can be useful to draw a trust boundary between them.
- When drawing trust boundaries, it is helpful to completely encircle what is inside the boundaries to clearly differentiate what resides inside and what remains outside. Do not use arcs or single lines to draw trust boundaries, since they can hide information flows and add confusion about what resides within a trust boundary and what is external to it.
- The act of drawing a trust boundary raises concerns about how entities on either side of the boundary interact through it. If there is not a defined communication path across the trust boundary documented in a diagram, but there is a trust boundary, that implies that there is communication path (either approved or unapproved) that may benefit from analysis in later stages of the threat modeling process. Examples of these communication paths include someone gaining physical access to a component, a sensor collecting data from an untrusted source, or someone accessing the ports and peripherals of the device.

2.3.1.4. (Example) Adding Trust Boundaries to the AMPS Device DFD

There are many ways to draw trust boundaries for the AMPS device's DFD. For example, a trust boundary could be drawn between every single component. That certainly is a valid approach, but it may not be scalable. Another approach is to draw trust boundaries between items that an organization controls, versus those that are controlled by external entities. This approach is shown in Figure 3. Note that for the purpose of this example, the trust boundaries are highlighted in red, but that color is not required to distinguish them. As stated earlier, trust boundaries are drawn using a dashed rectangle so that they remain recognizable if the diagram is printed out in black and white.

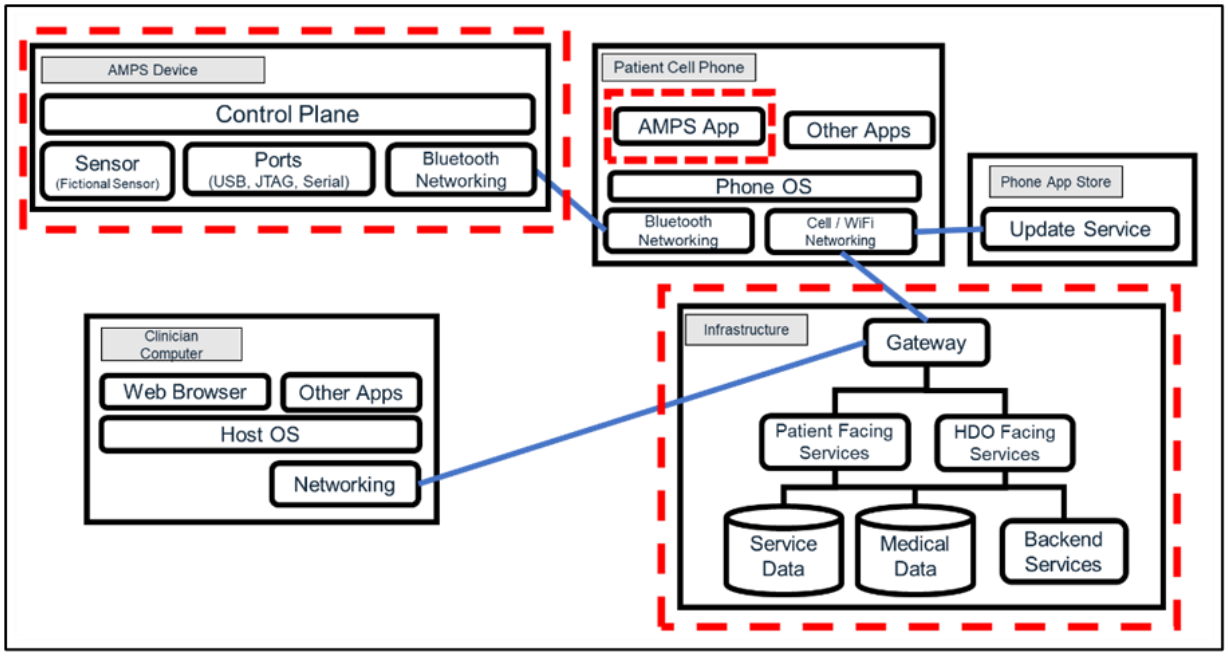


Figure 3: Example High-Level Trust Boundaries of the AMPS Device DFD

This example highlights that the patient's cell phone, the clinician's computer, and the phone app store are external entities whose interaction with other items inside the trust boundaries likely need to be

managed via security controls. These security controls will need to be evaluated in later steps of the threat modeling process.

More detailed DFDs may be created for individual components of the system being threat modeled. An example focusing on the AMPS device can be seen in Figure 4.

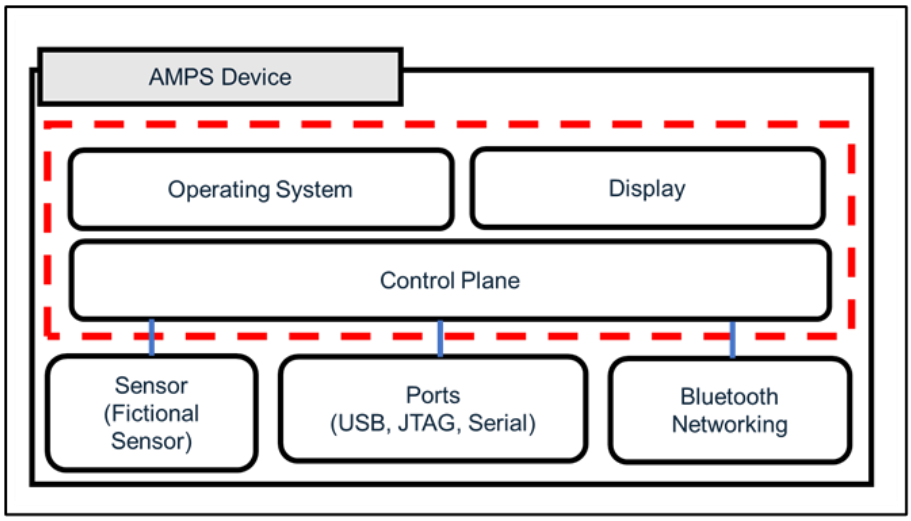


Figure 4: Example Device Trust Boundary

Notice that the external interfaces are in a different trust boundary than the control plane and core operating system. Placing the threat boundary in this manner raises questions such as how the device will continue to function if, for example, the Bluetooth network stack becomes corrupted or crashes. This can be a feedback loop as well. If during the threat modeling process your team raises the concern about how one component misbehaving could impact another component, it is helpful to ensure that those components are in different trust boundaries. If they are not, a good practice is to document a new trust boundary to separate them.

2.3.2. Structured Modeling with Swim Lane and State Diagrams

The strength of a DFD lies in the ability to lay out components on the macro level to understand how various pieces of a system fit together. However, there are times when other models might be better suited to capture certain aspects of the system—particularly when documenting how the system state changes or when evaluating dynamic interactions such as communication protocols. For those types of modeling tasks, Swim Lane and State diagrams may be better choices.

A Swim Lane diagram helps capture when multiple components work in tandem to achieve some goal. Each participant in the overall process being modeled is assigned its own “lane” to perform its steps before passing any outcomes to another lane. Swim Lane diagrams do not have to reference computational processes; they can also be used to capture decision flowcharts. Figure 5 is a Swim Lane example showing a generic high-level authentication session.

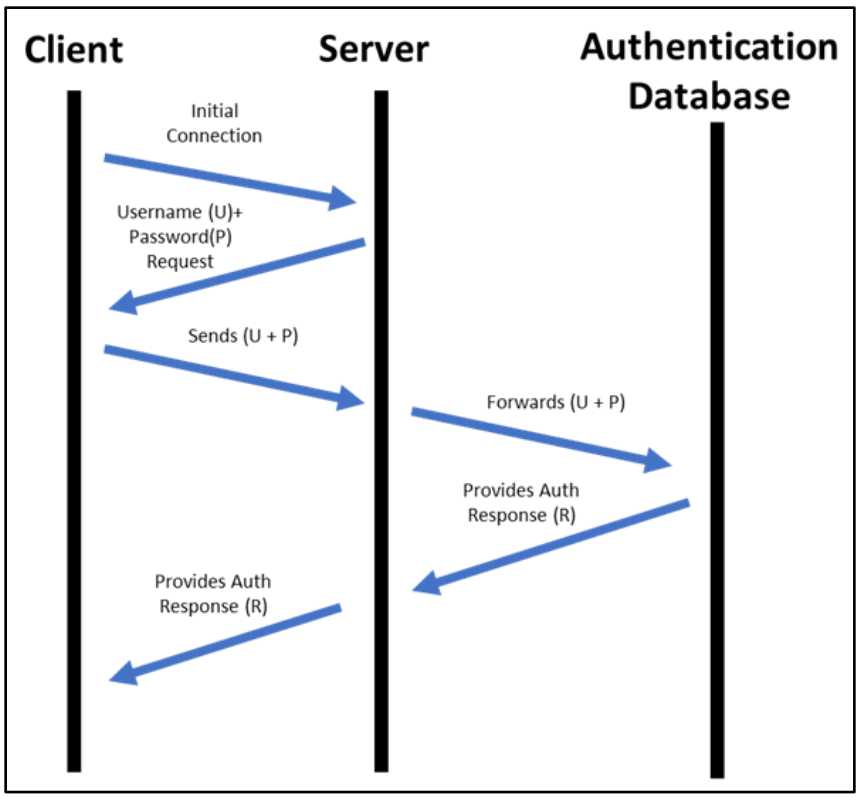


Figure 5: Example Swim Lane Diagram Illustrating a Generic High-Level Authentication Session

The Swim Lane diagram format shown in Figure 5 is also commonly referred to as a Sequence diagram. There are many ways to draw such diagrams, but a defining feature is that the entity names are displayed either at the top or left, and time flows downward or rightward, respectively. Abbreviations are often used to represent the data being passed back and forth, although the abbreviations tend to depend on the subject matter being diagrammed. Building on the earlier idea of trust boundaries in DFDs, implicit trust boundaries exit between each entity in a Swim Lane diagram. These will come into play in later stages of the threat modeling process. Sequence Swim Lane diagrams can be drawn in more formal ways, for example, using Unified Modeling Language (UML) sequence diagrams. More information about UML sequence diagrams can be found here [5].

Another type of Swim Lane diagram is a cross-functional diagram, with one specific type being a Rummler-Brache diagram. These diagrams tend to focus on business logic modeling and may be better suited for representing processes that involve branching or looping decision paths. Rather than depicting time flowing downward, components are presented horizontally with arrows describing the logic and communication flow between them. Figure 6 is an example of this type of diagram.

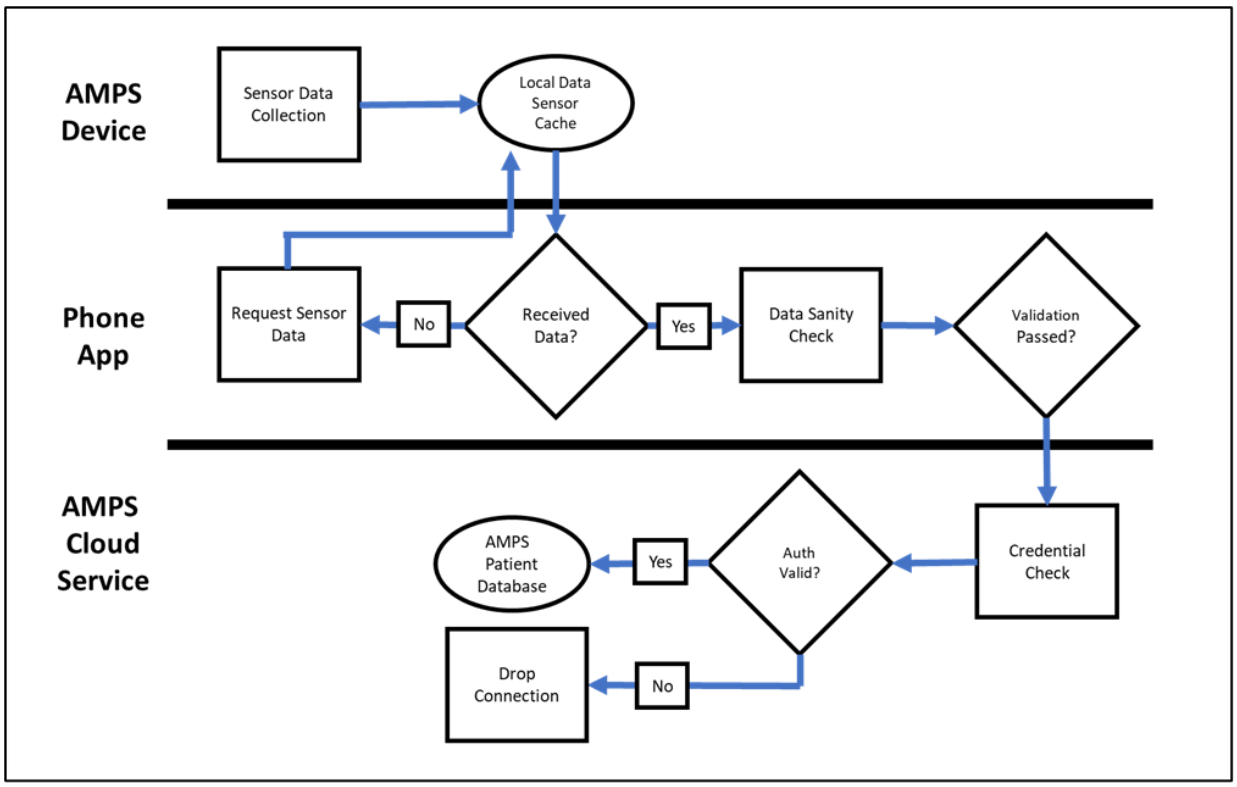


Figure 6: Cross-Functional Swim Lane Diagram

A more generic approach to a Cross-Functional diagram is a State Machine. Traditionally, in a State Machine, the states of the system are depicted as circles or boxes, with labeled arrows depicting the conditions required to transition from one state to another. The difference between this and a Cross-Functional diagram is that in a State Machine, the "state" represented in each circle describes the state of the system being modeled as a whole, with external inputs represented as the arrows transitioning the system from one state to another. An example can be seen in Figure 7.

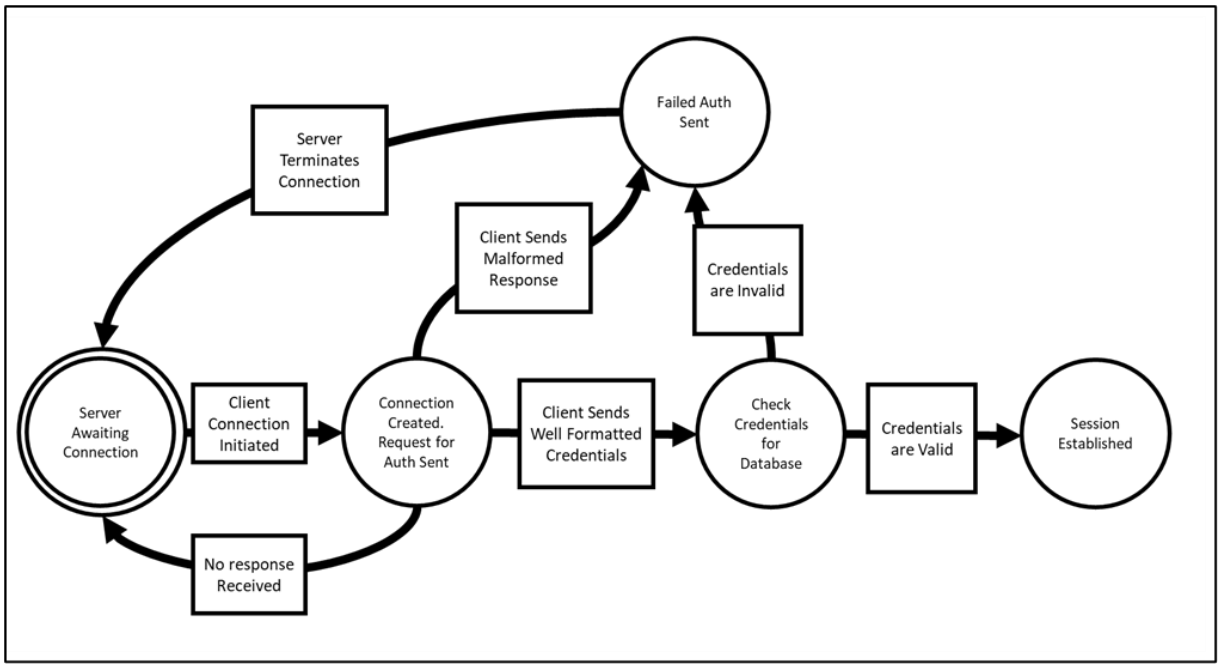


Figure 7: Example of a State Machine

It is important to note that the "system" in question may be one individual component, multiple components, or even entire business states and processes.

The modeling techniques described depict the system as dynamically changing over time, which is hard to capture in a more static DFD model. These views of system state can be helpful to describe how an organization expects the system will be used, the different modes it may find itself in, and how the system handles error states and invalid input. For example, a medical device may behave differently if it is in a treatment mode versus a maintenance configuration mode. Also, each of these different dynamic modeling techniques may be useful for highlighting different aspects of the system. For example, a Sequence diagram could be used to document network protocols (e.g., authentication sessions). State Machines, in turn, can be very useful for diagramming modes of operation. There is no single "best" diagram, and organizations may find that using multiple different diagraming techniques as part of the threat modeling process produces the most useful result.

2.3.3. Leveraging Multiple Modeling Techniques

When attempting to answer the question of "What are we working on?", it is very common to use multiple different modeling techniques. There is no universal "best" modeling technique. Instead, organizations may select what works best for their needs and the systems they are attempting to understand. It can also be helpful to remember that these modeling techniques all describe a common underlying system. Therefore, different modeling techniques may highlight different aspects of the system in unique ways that can be helpful when identifying threats. For example, in a DFD, trust boundaries can be explicitly drawn and highlight where entities of different levels of trust may interact with each other. In process flow diagrams, the idea of trust boundaries still exists, because there is an implicit trust boundary for every transition in state the system makes. Even though a trust boundary might not be shown on the diagram, controls may be in place to ensure the state transitions depicted in a state machine occur in the manner depicted. For example, a buffer overflow might cause an authentication session to be bypassed. Likewise, a cleaning crew might unplug a device in the middle of a transaction, causing the system to fault into an unexpected state. An attacker could connect a keyboard and a monitor, then access a root command prompt locally rather than via the remote terminal where authentication would normally occur. An explicit trust boundary in a DFD might highlight the threat of an attacker accessing the system using

an unexpected keyboard and monitor, while the implicit trust boundary in a process flow diagram might highlight the threat of a cleaning crew unplugging the device.

Listing these examples demonstrates an unstructured threat modeling technique; structured approaches will be detailed later. But it is useful to consider how the diagrams created can guide analysis about where systems can fail. These models are designed to help organizations understand and threat model devices. Attackers do not have to respect organizations' diagrams or assumptions. Therefore, as these models are built out, organizations can frequently revisit them, make revisions, identify gaps, and confirm or challenge assumptions that may have been made.

As mentioned earlier, static diagrams such as DFDs struggle to capture the idea of "system state." System state information can be imposed on top of a DFD, either by including descriptions of connections between entities or using external documentation. However, DFDs excel in identifying the connections and the entities that interact with each other. Process Flow diagrams (e.g., Swim Lanes and State Machines) are good at modeling system state, but they suffer a common pitfall. They tend to capture how the designers intend the system to behave versus how it might behave in practice.

At a high level, creating diagrams and documentation help answer the following questions:

- What makes up the system?
- Where do trust boundaries exist with external entities?
- Are there trust boundaries that exist within your system itself?
- What are the high-value dataflows that merit detailed analysis?

The last question about identifying high-value dataflows will be revisited in 4.Appendix A, "Additional Fictional Medical Device Examples." As a quick preview, high-value dataflows tend to be high risk and have a high impact if they are compromised. Examples include:

- Authentication protocols
- Programming and configuration commands
- Obtaining and validating software updates
- Sharing patient data with external servers
- Transmitting and silencing of alarms
- Procedures to restore from backups

For these types of dataflows and interactions, detailed diagrams about their specific implementations can be helpful. If a standard protocol is used, such as Digital Imaging and Communications in Medicine (DICOM), be sure to include versioning information as well as specific implementation details.

At this stage of the threat modeling process, organizations will likely have a mixture of DFD-like diagrams to model the components of the system, and state-dependent diagrams such as swim lane depictions of high-value dataflows. Different versions of the same diagram type that depict different views of the system may also exist. Finally, these diagrams are not immutable. Expect to update them as the threat modeling processes continue, which will be much easier if they are saved in a manner that is easy to capture, share, and update.

2.3.4. Tips for Knowing When to Move to the Next Stage

One of the most frequent questions asked when building out a threat model is "How do I know when I'm done?". While a thorough job is important, remember that threat modeling is an iterative process and that none of the Four Questions needs to be answered completely in one go.

Consider the following when determining if it is appropriate to move on:

- Are all the core functions of the system described by the model in some way?
- Could someone unfamiliar with the system learn how it works based on the current documents?
- Are diagrams clear and concise, with minimal overlapping dataflows?
- Are critical dataflows labeled?
- Are trust boundaries established and justified?

When in doubt, it is generally fine to move to the next stage of the threat modeling process. This will often identify gaps in initial diagraming when organizations start to identify "What can go wrong?" and find that certain threats do not easily map back to the models they have developed. In that case, organizations may consider whether it would be wise to step back, update the diagrams and models in question, and then repeat the process.

2.4. Question 2: What Can Go Wrong

When threat modeling, the question "What can go wrong?" springs to mind. There are many ways to answer this question, and opinions of threat modeling experts can vary widely, as do current practices. In this Playbook, the techniques and approaches highlighted are intended to help organizations avoid common pitfalls that occur during the threat modeling process. The Playbook will focus on structured threat modeling techniques and will minimize asking threat modeling teams to "think like an attacker." Thinking like an attacker is a loaded phrase that often comes up in threat modeling discussions and revolves around the idea of defenders trying to get inside the mind of attackers and predicting what strategies those attackers are going to follow. There certainly are benefits to thinking like an attacker, but the challenge is that different teams bring different life experiences and preferences to the threat modeling process, and those may be different from skilled attackers who have built up expertise and invested significant time and resources with the primary goal of targeting a system.

When asking "What can go wrong?" consider what might happen in the absence of security controls that may already be in place. At first, this may seem counterintuitive. Is something really a "threat" if it cannot happen because mitigations are already in place? But consider–what might happen if those mitigations were to fail or be subverted? The "What can go wrong?" stage is the time to identify and document any controls that are already in place. Then, brainstorm how those controls might fail, or how the presence of each control might cause additional threats. The presence of a security control does not automatically imply that there is a lack of threats, or an inability for threats to manifest. Instead, best practices encourage organizations to use these discussions to expand assessments of "What can go wrong?"

2.4.1. Identifying Threats with STRIDE

STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) is one of many methodologies for structured analysis and reasoning about threats against a given system. As always, it is not meant to be a silver bullet that catches or enumerates all potential threats. However, it can help provide a foundation for discussing and brainstorming threats that may not have been considered before.

STRIDE is a mnemonic that articulates six types of potential threats against a system. Table 2 describes each element of STRIDE and provides a brief example that may be seen in healthcare systems.

Table 2: Summary of the STRIDE mnemonic elements with healthcare system examples

STRIDE Element	***Description***	***Example***
Spoofing	Tricking a system into believing a falsified entity is a true entity	Using stolen or borrowed credentials to log on as another nurse
Tampering	Intentional modification of a system in an unauthorized way	Changing patient data to incorrect values
Repudiation	Disputing the authenticity of an action taken	Denying that a prescribed treatment has been provided to the patient
Information Disclosure	Exposing information intended to have restricted access levels	Health data is sent over an unencrypted Bluetooth connection
Denial of Service (DoS)	Blocking legitimate access or functionality of a system by malicious process(es)	A Bluetooth SpO2 sensor is flooded with bad pairing requests, preventing legitimate connections
Elevation of Privilege (EoP)	Gaining access to functions to which an attacker should not normally have access according to the intended security policy of the product	A patient uses a web portal vulnerability to see all patient data, rather than their own

Identified threats may not cleanly fit into only one STRIDE type. The example used for “Elevation of Privilege,” where a user gains access to other patient data, may also be considered information disclosure. Threats may also be the cause of additional threats. For example, Elevation of Privilege may occur due to the utilization of stolen credentials (Spoofing). While using a standardized methodology for categorization can be helpful for sharing results, ultimately the goal is to ensure that potential threats are documented, rather than excluded because they do not appear to fit cleanly in the framework. Likewise, while discussions about which category to assign to a threat may become extensive, it may be more productive to pick a “good enough” category for the identified threat and move on to brainstorming other threats, versus trying to agree upon the perfect category.

Below are several considerations when applying STRIDE:

- When applying STRIDE to a DFD, it is often helpful to apply STRIDE threats to the elements that are impacted by those threats. For example, a dataflow may be tampered with using a person-in-the-middle attack. But a spoofing threat would be applied to a process or external entity that was being “tricked.” The important thing is to capture and record the threats, not where they are categorized.
- Identifying repudiation threats is typically a source of frustration for those new to using the STRIDE methodology. One shorthand to use when identifying repudiation threats is to look for an area where there is monetary or legal impact to subverting the process. Repudiation threats are different from spoofing since the risk is in how different parties interpret an action. Therefore, repudiation threats may also include spoofing threats, such as an attacker submitting false data, but they have an added twist that a legitimate user may instead be deceptive.
- There are many ways to document the output of applying STRIDE to a device. There is no “right” way to capture findings. However, one helpful approach is to create one table that keeps track of STRIDE threats found, including a link or reference to a more detailed description of those threats that may be found outside that table. This allows organizations to quickly identify where threats have been identified while also providing space to have a detailed description of those threats.

There is no universal methodology for applying STRIDE when identifying what can go wrong. Typically, when using STRIDE, the first step is to refer to the models created when identifying "What are we working on?" STRIDE tends to be easiest to apply when analyzing a DFD, but it can be applied to other modeling methodologies as well.

One method to apply STRIDE to a DFD is called the "STRIDE per Element" approach. This method was developed by analyzing which STRIDE threats tend to appear for individual DFD element types. This approach creates a mapping (shown in Table 3), where for a particular DFD element, there will be a list of STRIDE threats commonly associated with it.

Table 3: STRIDE Per Element

Element	Spoof	Tamper	Repudiate	Info Disclosure	DoS	EoP
External Entity	X		X			
Process	X	X	X	X	X	X
Data Store		X	?	X	X	
Dataflow		X		X	X	

There are several items to consider about the STRIDE per Element approach:

- External Entities only have the Spoof and Repudiate threats applied against them. This is because an organization may not have knowledge regarding how external entities are designed. Therefore, organizations may protect their systems from interactions with External Entities by threat modeling the Spoof and Repudiate threats, but modeling of other threats may be difficult and outside the scope of threat modeling activities.

- There is a question mark for the Repudiate threat for Data Stores. This is because some organizations categorize Repudiation threats against the Data Stores, and others instead categorize those threats against the Processes that manage those Data Stores.
- Dataflows only have the Tamper, Info Disclosure, and Denial of Service threats applied to them. That is because the STRIDE analysis of Dataflows focuses on threats against the data being transmitted, not how that data is interpreted. Therefore, an adversary may eavesdrop on the conversation (Info Disclosure), alter the data in transit (Tamper), or interrupt the transmission of the data (Denial of Service). Threats such as Escalation of Privilege and Spoofing may be categorized with the Process that is receiving that data.

An important note about the STRIDE per Element approach is to view it as a helpful tool versus a hard-and-fast rule. Its goal is to stimulate discussion and structured analysis by identifying gaps that may exist in organizations' current assessments of what can go wrong. How organizations categorize and assign those threats can be done with some degree of flexibility. For example, Spoofing in the STRIDE per Element approach is a common threat to External Entities and Processes. Many organizations may instead categorize that spoofing threat as being applied to the dataflow instead. Identifying and documenting the threat is the important part, not categorizing it.

The STRIDE per Element approach can be helpful when trying to identify when to stop brainstorming "What can go wrong?" and move to the next stage of the threat modeling process. For example, Repudiation threats can often be easy to skip over and miss. If organizations have performed an initial pass at identifying what can go wrong and have a process that does not include a Repudiation threat associated with it, it can be productive to spend additional time brainstorming to identify if that threat

applies. That threat may not exist, in which case it may be helpful to document why it doesn't apply to the element under consideration. Likewise, if organizations have identified threats that match up with the STRIDE per Element, and is struggling to identify new threats, that may indicate that this step of the threat modeling process is complete, and the next step may begin.

Another common technique to optimize applying STRIDE to a device is to focus on how different elements interact between trust boundaries. This allows abstraction of much of the inner complexity that may occur inside a trust boundary. This also allows prioritization when performing a STRIDE analysis. This is partially an acknowledgement that DFDs for many modern medical devices may have hundreds of elements in them, and organizations have limited resources with which to threat model. Therefore, the elements that are most likely to come under attack are also the elements that have been identified crossing a trust boundary. However, given sufficient time, it is also helpful to perform STRIDE on individual elements inside a trust boundary.

For more information about STRIDE, one reference is "Threat Modeling: Designing for Security" by Adam Shostack [2].

2.4.2. (Example) Applying STRIDE to the AMPS System

One potential way to apply STRIDE to the AMPS system is to initially prioritize investigating all STRIDE threats that cross trust boundaries. In Figure 3, three trust boundaries were drawn: one around the AMPS device itself, one around the AMPS app on the phone, and a final one around the APMSCS.

Each of those trust boundaries has various other components they interact with, as well as data flows going into and out of them. To capture STRIDE threats to various components, it may be helpful to form a summary table to enumerate both the components of a system and whether that component has a threat for a particular STRIDE element. Table 4 shows a completed summary after one round of brainstorming. Note that the number in each row is a reference to a more detailed explanation of the threat and is not a count of the threats found. The numbers also do not have to be contiguous. They may instead be treated as links, since new threats may be identified during the modeling process, and previously identified threats may be deemed to not exist upon further analysis.

Table 4: Summary of STRIDE Categories with Identified Threats against the AMPS System Components

AMPS Component	***Spoof***	***Tamper***	***Repudiate***	***Info***	***DoS***	***EoP***
AMPS Device	1	2			3, 34, 35	4
AMPS App	5, 36	6		7	8	9, 37
App Store	10, 38	11		12	13	14
AMPSCS	15, 39, 40	16,41	17, 42, 43, 44	18, 45	19, 46, 47, 48	20, 49, 50
Clinician Computer	21, 51, 52	22		23	24, 53	25
Dataflow: Bluetooth				26	27, 54	
Dataflow: Cell/Wi-Fi Network		28		29	30	
Dataflow: Clinician Computer Internet		31		32	33	

For brevity, this example section focuses on applying STRIDE only to the AMPS device itself, based on the DFD shown in Figure 3. Table 5 summarizes the initial results. Note that one round of brainstorming did not initially yield any threats for repudiation and information disclosure. Going back to the STRIDE per Element approach detailed in Table 3, this implies there is still likely work to be done to identify threats for those elements.

Table 5: Description of STRIDE Threats against AMPS Device

Reference ID	STRIDE Type	Description
1	Spoof	An attacker could pretend to be an authorized phone app to obtain readings from the device
2	Tamper	Control plane could be attacked and given incorrect readings
3	DoS	Invalid input could cause device to crash
4	EoP	Device could be hacked, and software could be installed to perform other actions (such as make it part of a botnet, enable lateral movement, etc.)
34	DoS	Software could be corrupted
35	DoS	Battery could be drained more rapidly than normal

2.4.3. Identifying Threats with Attack Trees

One challenge with threat identification approaches such as STRIDE is that they can struggle to tell a story, and it may be hard to prioritize threat findings because certain threats may be mitigated by controls elsewhere in the system. Attack Trees attempt to address this problem by adding structure to make it easier to walk through exploitation paths.

There are many ways to draw Attack Trees, with the divide usually being if they flow top-down or bottom-up. Top-down Attack Trees tend to model an attacker's actions and are sometimes called threat-driven Attack Trees. Bottom-up approaches more closely resemble Fault Tree methodologies that are often seen when performing safety analysis.

A top-down Attack Tree is a multilevel diagram with the first stage of the attack drawn on the top as a root node, and children of that node drawn below that node to include possible outcomes and attacker actions. This process is repeated at subsequent levels. A very basic example of a top-down Attack Tree applied to the AMPSCS is in Figure 8.

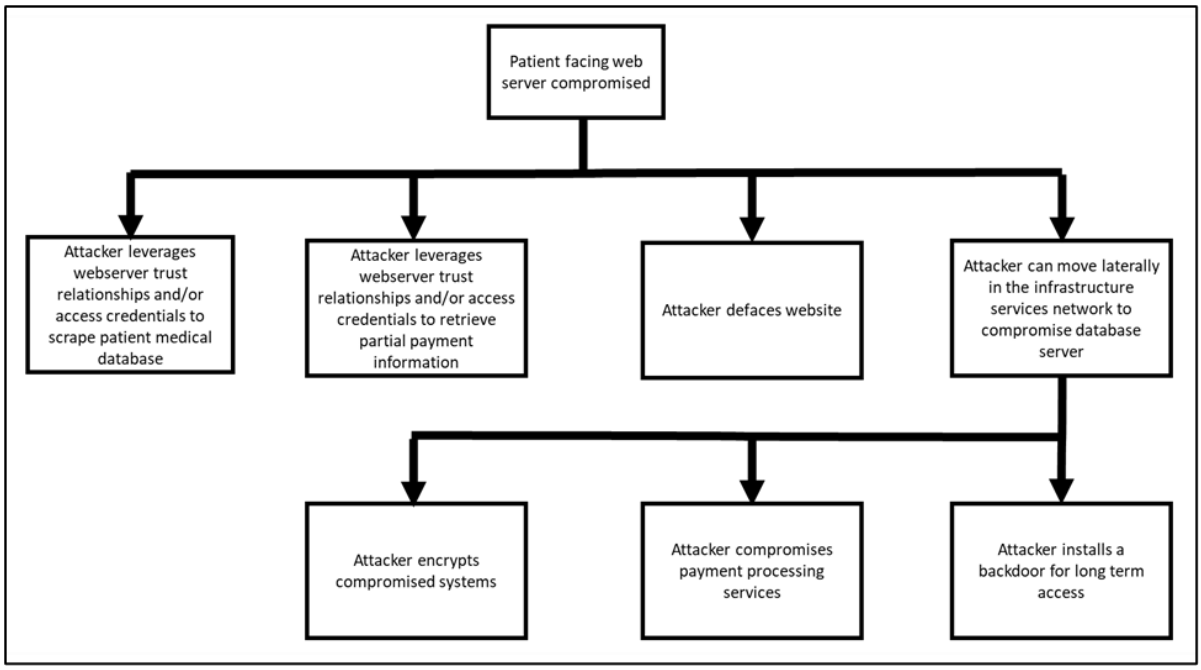


Figure 8: Example top-down, threat-driven Attack Tree against AMPSCS

Despite its name, a bottom-up fault analysis Attack Tree is also often started by drawing a root node at the top. The difference is the root node represents an end state that the system designers are trying to

avoid. Then Boolean logic (AND and OR gates) is constructed to illustrate the preconditions that are expected to be required to enable the resulting state. Therefore, the diagram grows from the top down, but then is read from the bottom up when following the path an attacker is expected to take. An example is in Figure 9.

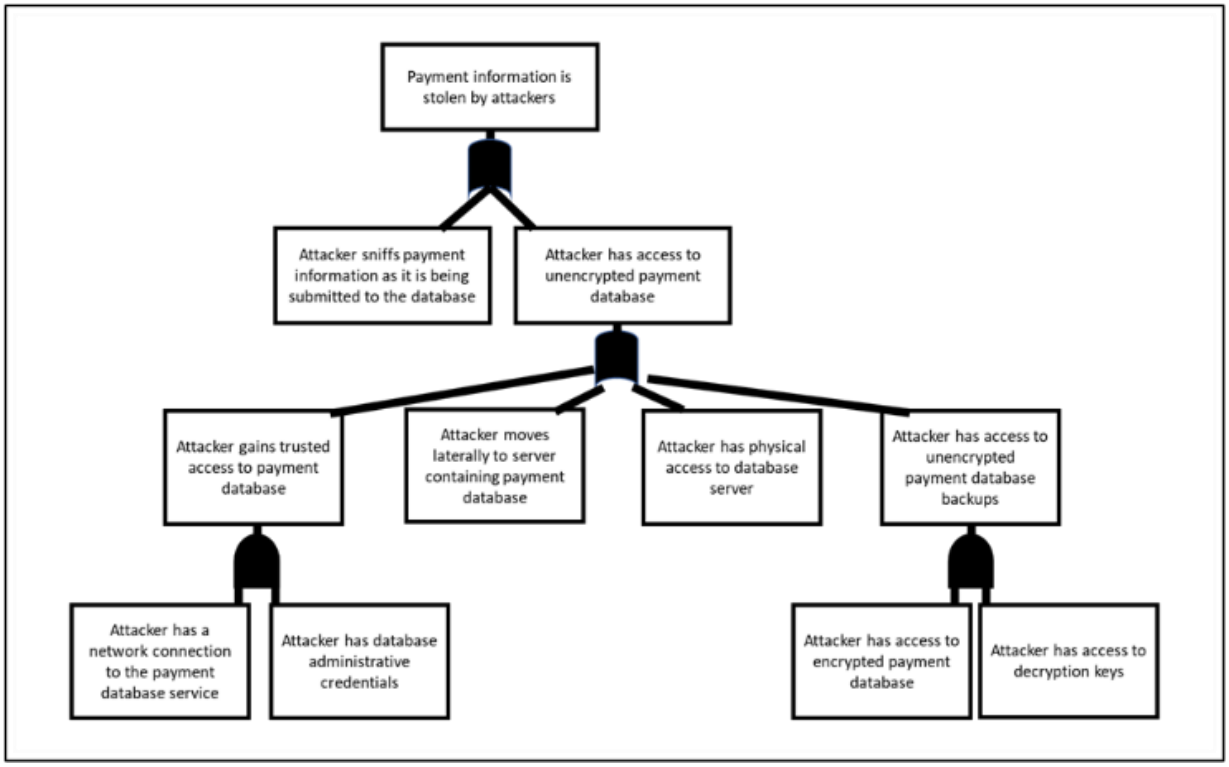


Figure 9: Example of bottom-up fault analysis Attack Tree against AMPS payment information

Below are several considerations for designing Attack Trees:

- As a general rule, when developing Attack Trees, ensure that there is always at least one path across any identified trust boundary. The reasoning behind this rule is that the trust boundaries are permeable by their nature. An attacker may be on one side of that trust boundary attempting to leverage their access. Therefore, if this movement across a boundary is not captured, the Attack Tree is missing a potential threat.
- It is often useful to separate Attack Trees into different diagrams for each trust boundary, and then connect them together. An attacker may be able to gain access through an undocumented connection, or an attacker may be a trusted user or administrator. Not all attackers may start from public-facing servers and move their way into the networks being modeled. Being able to highlight the fact that attackers may be able to “skip” steps can be very important to make sure security controls can handle those situations. This also helps because many times different Attack Trees share nodes, as an attacker may gain access to certain systems via a multitude of means. Being able to “link” trees together can help reuse work and keep Attack Trees manageable.
- It is often helpful to diagram paths through an Attack Tree that are not believed to be possible due to existing security controls. Security controls fail, and often there may be undocumented trust relationships that impact the security of the system. For example, a firewall policy may not be correctly applied, or a developer could have left a password stored in a script.
- Attack Trees tend to become more useful as they mature over time. As new attacker techniques are discovered, organizations may return to their Attack Trees and identify gaps that may not have been initially captured. This discovery process may include open-source research based on

adversaries targeting other products, or real-life experiences gained though testing, red-teaming, or actual attacks against products.

- It is often useful to combine Attack Trees with other threat identification techniques such as STRIDE. For example, if a threat is identified during STRIDE analysis, organizations may then attempt to develop an Attack Tree that encompasses that threat. In this way, Attack Trees can help provide a story around STRIDE results. On the other hand, if a STRIDE-like analysis is performed after an Attack Tree has been developed, the identified threats in that analysis can be used to check the Attack Tree for gaps.
- For fault analysis-style Attack Trees, it often is tempting to assign probabilities to events to try to calculate the overall risk of an adverse event occurring. While this can be effective in traditional fault analysis, it is usually not recommended for attacker-driven threats, because there are no known effective techniques for calculating probability data for human-driven attacks.
- Attack Trees tend to require significant subject matter expertise to provide useful coverage. For initial threat modeling efforts, it may be more productive to start with a more structured threat identification technique such as STRIDE.
- One obvious weakness of the Attack Tree approach is that an attacker can carry out many types of actions once they achieve a certain level of access to the system. This can make diagramming and identifying these actions time-consuming, but more important, there is a good chance that organizations' threat models may have missed several techniques. It is important to keep in mind that an Attack Tree is only a representation of what an attacker will do. Attackers do not have to follow the models that organizations create.

For more examples about using Attack Trees, a good resource is an article by Bruce Schneier [6].

2.4.4. Identifying Threats with Kill Chains and Cyber Attack Lifecycles

The idea of a Cyber Kill Chain was originally developed by Lockheed Martin to enumerate the steps an attacker takes when compromising a system or network [7]. The basic premise is if any of the "rings in the chain" (steps) can be disrupted or broken, a defender can prevent the attacker from obtaining their end goals. When it comes to modeling a cyber kill chain, there are several different variations of this approach beyond the original Lockheed Martin design, such as the Cyber Attack Lifecycle depicted which is depicted in Figure 10.

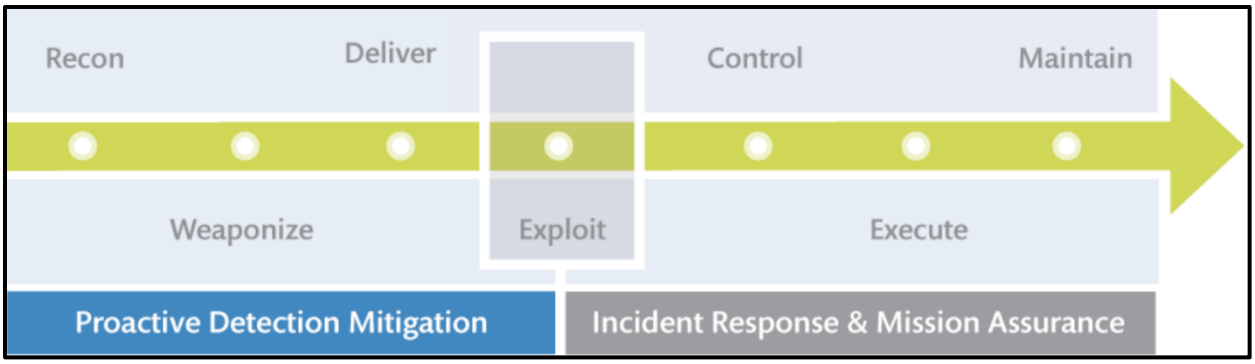


Figure 10: Cyber Attack Lifecycle (Diagram from [8])

Modeling an attack as a lifecycle versus a kill chain can be useful since attacks may not be static events. If an attack is mitigated at any stage, the attacker may modify their techniques and try something else. Therefore, while these frameworks often depict an arrow going forward, there may be loops as an attacker responds to defensive mechanisms.

In the traditional Cyber Attack Lifecycle, there are usually seven different stages. These stages may have different names depending on which framework is being referenced, but at a high level, they generally follow the basic stages as described in [8]:

1. **Recon**: In this stage, the adversary identifies, selects, and investigates a target.
2. **Weaponize**: This stage describes an adversary's effort to acquire tooling around a vulnerability and develop it into a form that can be executed on the target's computer or network. This may involve finding a tool or exploit created by others or creating it themselves.
3. **Deliver**: This stage describes how the initial attack is transported to the target's computer or network. This could be a phishing email, a network connection to a vulnerable web server, or a USB drive dropped in a parking lot.
4. **Exploit**: This stage describes what occurs when the initial attack on the target is executed.
5. **Control**: This stage covers how the adversary employs mechanisms to send commands to software running on the target's network. This stage is common in many attack lifecycles but does not have to occur. For example, a DoS attack may not require the exploit to be controlled after its initial delivery.
6. **Execute**: This is a catch-all category to highlight the steps an adversary may take to achieve their objectives, whatever those objectives may be.
7. **Maintain**: This describes the actions an adversary takes to maintain their access to the target's computer or network.

Cyber Attack Lifecycles can help in the threat modeling process by placing additional structure around approaches such as Attack Trees and providing ways for the organizations carrying out threat modeling to consider what an attacker must achieve to execute their objective. To illustrate this, consider Figure 8, which was the Attack Tree built out previously for the AMPSCS. Figure 11 shows the same diagram through the lens of the Cyber Attack Lifecycle. By using the Cyber Attack Lifecycle framework, this approach highlights that our original model did not consider the recon, weaponizing, and delivery stages an attacker would have to go through to exploit the vulnerable web server. Likewise, once an attacker is inside the system, our original Attack Tree did not consider the actions that an attacker would need to maintain control of their tools as they traversed the internal services of the AMPSCS.

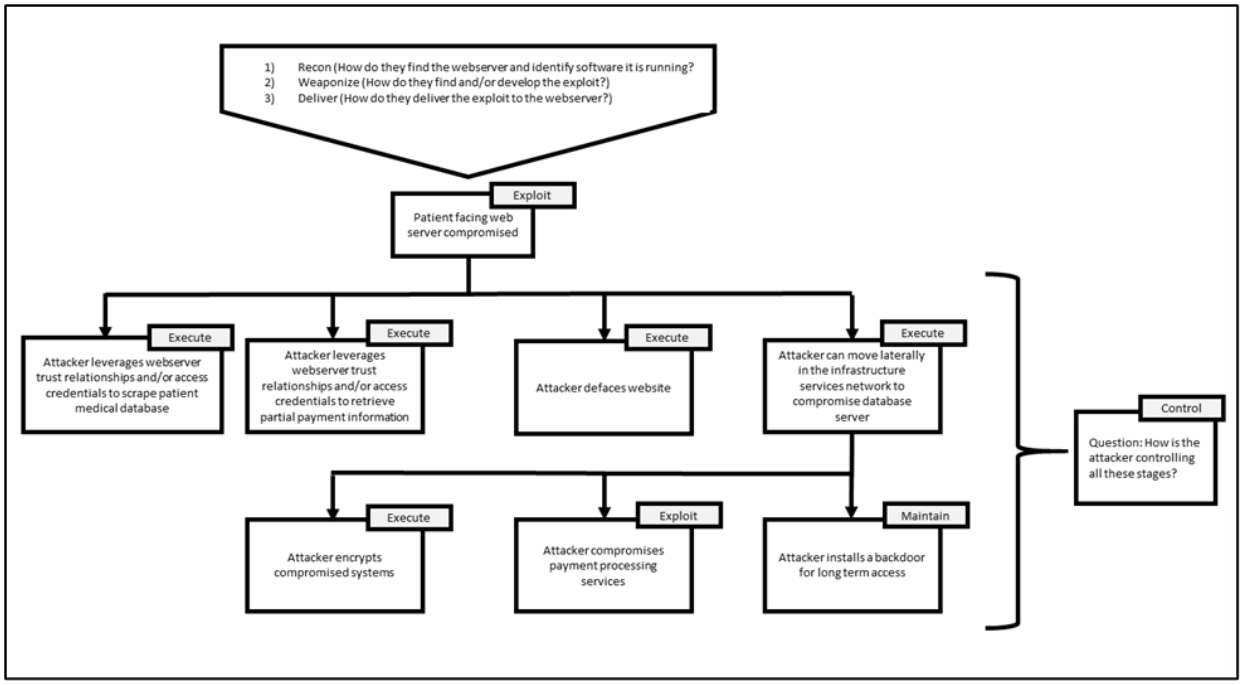


Figure 11: Revisiting the Attack Chain Example Using the Cyber Attack Lifecycle

Instead of representing the Cyber Attack Lifecycle as an Attack Tree, another way to represent the different stages is in a table. Table 6 shows the Cyber Attack Lifecycle applied to a portion of the Attack Tree shown in Figure 11.

Table 6: Example Cyber Attack Lifecycle for the AMPSCS

Lifecycle Stage	**Description**
Recon	Attacker runs a scanner against public-facing pages of the AMPSCS. They discover that the web server is outdated and has a known vulnerability.
Weaponize	Attacker finds a publicly available exploit for the website vulnerability.
Deliver	Attacker connects to the website through a virtual private network and sends an exploit payload to the vulnerable website.
Exploit	The exploit compromises the web server and installs an initial "dropper" executable on the server.
Control	The "dropper" downloads a more fully featured remote access trojan (RAT) to the web server from an external host under the attacker's control. The RAT then beacons out to a different computer under the attacker's control to receive commands.
Execute	Attacker uses credentials stored on the web server to download patient data from a backend database.
Maintain	Attacker sets their RAT to reload if the web server is rebooted by adding it to the normal system start-up scripts and processes.
Recon	Attacker uses their access on the web server to scan for other systems on the internal AMPSCS network.
Exploit	Attacker uses shared credentials to log in to a different server inside the AMPSCS.
Control	Attacker installs their RAT to the new server they now have access to.
	… And the process continues

Below are several considerations for incorporating the Cyber Attack Lifecycle into threat modeling processes:

- For the initial recon and weaponize stages of the Cyber Attack Lifecycle, the defender often has limited insight into what the attacker is doing, and almost no control. Therefore, those stages are often simplified or abstracted out of the generated models.
- The initial delivery stage may also be abstracted out, but it can be important if the exploit requires user interaction. For example, if the exploit is a phishing email that tricks a user into downloading an executable, the delivery stage may involve defensive mechanisms, such as antivirus running on the mail server, that can disrupt the delivery of the exploit. On the other hand, if the attacker is connecting to a public-facing web server to deliver an exploit, there are fewer defensive mechanisms a defender can use at this stage to disrupt the delivery of the exploit.
- Once an attacker is inside a system, the recon, weaponize, and delivery stages of the Cyber Attack Lifecycle can become more actionable for a defender. For example, network scanning may be detected and alert security teams that an internal system has been compromised.
- From a threat modeling perspective, it is often helpful to consider the exploit stage to be successful, even if no known vulnerability exists for the service or system under attack. No software is free of vulnerabilities; new vulnerabilities may be discovered, and assuming the exploit stage is successful allows organizations to capture what may happen afterwards, as well as identify controls that can help detect and mitigate the attack.

- Common patterns will often emerge in the Cyber Attack Lifecycle as an attacker performs the same actions repeatedly while moving through a system. It is acceptable to condense these techniques and reference them versus listing them all out for every single system. However, when looking at the effectiveness of mitigations, the same approaches an attacker takes may be detected or deterred differently depending on where they are. For example, an intrusion detection system may only be listening to a subset of network traffic. Therefore, it can be helpful to capture the targets of an attacker's actions in documentation such that the effectiveness of mitigations can be evaluated against the different places an attacker may be.

2.4.5. The ATT&CK Framework

One challenge with applying the Cyber Attack Lifecycle is that it often assumes that organizations have detailed knowledge of what an attacker may do. For example, an attacker could exploit a server using SQL injection, install a dropper that would then download a remote exploit trojan for control, and then enumerate user accounts and attempt to steal passwords. To reduce the amount of expertise an organization needs to carry out threat modeling, several public repositories and frameworks exist for capturing and describing what attackers have done based on real-world data.

One such repository is the MITRE ATT&CK framework [9]. The ATT&CK framework is a globally accessible knowledge base of adversary tactics and techniques based on real-world observations. There are several variations of the ATT&CK framework, including ones for enterprise computing, mobile, and industrial control systems (ICS).

At a high level, the ATT&CK framework is broken down into tactics, techniques, sub-techniques, and procedures. The following is taken from the ATT&CK Frequently Asked Questions (FAQ) [9]:

- Tactics represent the "why" of an ATT&CK technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action. For example, an adversary may want to achieve Credential Access, which is one of the tactics represented in ATT&CK.
- Techniques represent "how" an adversary achieves a tactical goal by performing an action. For example, an adversary may utilize the "OS [Operating System] Credential Dumping" technique to achieve the Credential Access tactic. This technique describes obtaining credentials from the operating system itself, usually in the form of a password hash or clear text password.
- Sub-techniques are a more specific description of the adversarial behavior used to achieve a goal. They describe behavior at a lower level than a technique. For example, an adversary may perform OS Credential Dumping by accessing the Local Security Authority Subsystem Service (LSASS), which manages login access and credentials in Microsoft Windows operating systems.
- Procedures are the specific implementation the adversary uses for techniques or sub-techniques. For example, a procedure could be an adversary using PowerShell to inject into lsass.exe to dump credentials by scraping LSASS memory on a victim. Procedures are categorized in ATT&CK as the observed in-the-wild use of techniques in the "Procedure Examples" section of technique pages.

Figure 12 shows an example of the MITRE ATT&CK for Enterprise layout with tactics displayed in the top row, with associated techniques for those tactics listed in the rows below. When viewed on the ATT&CK website, the techniques may be clicked on to display the sub-techniques and procedures associated with them.

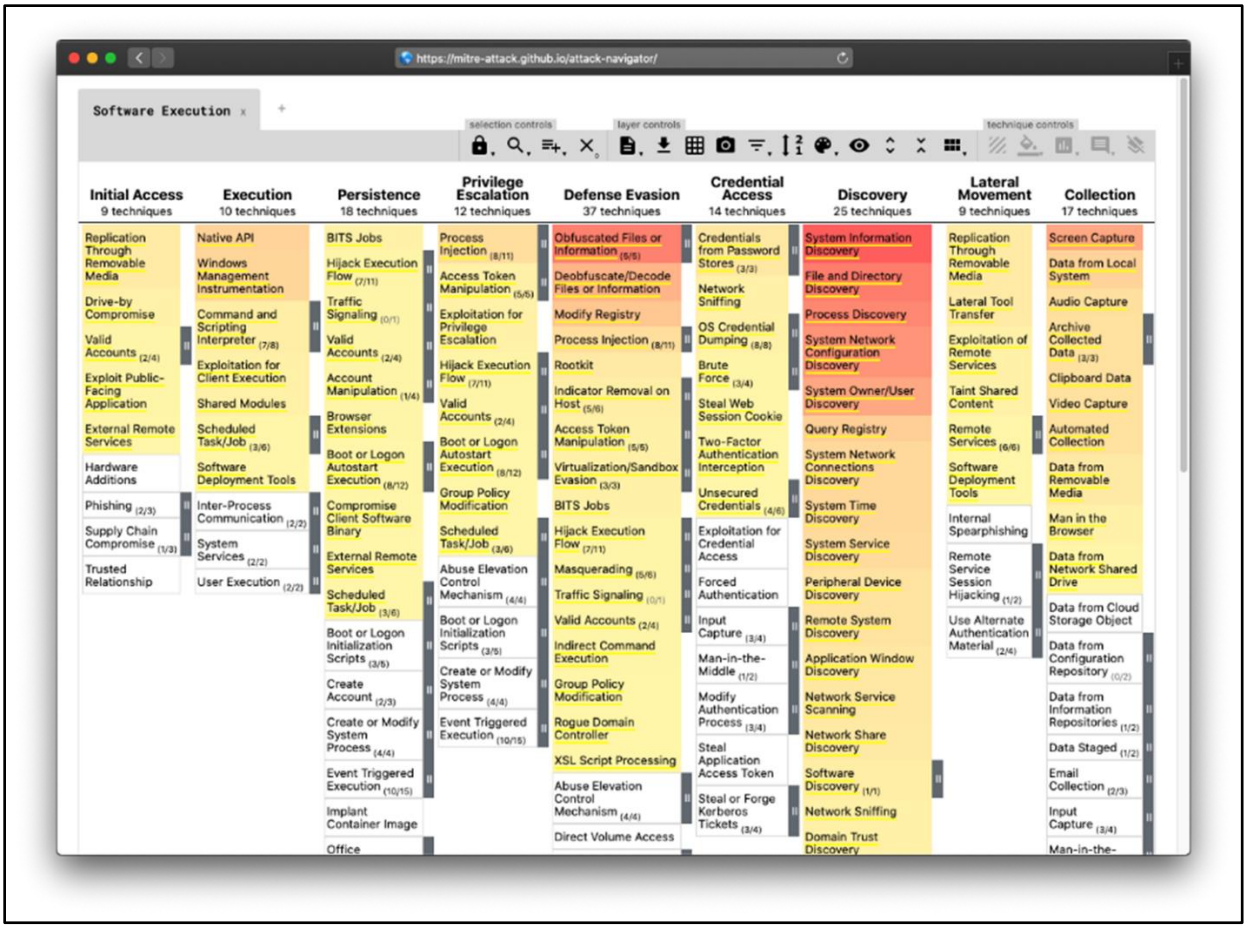


Figure 12: MITRE ATT&CK for Enterprise Framework

The ATT&CK for Enterprise tactics roughly resemble the stages of the Cyber Attack Lifecycle but have been further broken down to better differentiate the different tactics an adversary may employ. Another difference is that the ATT&CK tactics don't depict an in-order flow, and attackers can jump between them multiple times during a campaign. The nine ATT&CK for Enterprise tactics are:

- **Initial Access**: This roughly corresponds to the initial Recon/Weaponize/Delivery/Exploit stages of the Cyber Attack Lifecycle and describes how an adversary gains an initial foothold in a network or system. For example, they may have physical access, they may use a supply chain compromise, or they may target an external-facing web server.
- **Execution:** This tactic focuses on how the adversary runs their code on a local or remote system. Examples could be using a PowerShell script or hijacking a native Application Programming Interface (API). Despite the name similarity, this does not directly map to the Execute stage of the Cyber Attack Lifecycle. This ATT&CK tactic focuses on the specifics of how the attacker code is executed, whereas the Cyber Attack Lifecycle stage focuses on the attacker's end goals. ATT&CK focuses on how the code is run because that can highlight how different defensive approaches can identify malicious code execution.
- **Persistence:** This maps to the Maintain stage of the Cyber Attack Lifecycle and describes how an adversary maintains access to a system across restarts, changed credentials, and other interruptions that could cut off their access. Examples could be installing a malicious browser extension or creating a new attacker-controlled account on the system.
- **Privilege Escalation:** This tactic does not directly map to a stage of the Cyber Attack Lifecycle. It describes how an adversary can leverage their existing access to gain higher level permissions

on a system or network. For example, an attacker could perform a password cracking attack against stored administrator credentials, or they could exploit a vulnerability to gain enhanced privileges.

- **Defense Evasion:** This tactic loosely maps to the Maintain stage of the Cyber Attack Lifecycle. It describes how an adversary avoids being detected by defenders. Examples could be disabling antivirus software on infected computers or encrypting their network traffic when inside the defender's network.
- **Credential Access:** This tactic describes different ways an attacker can gain access to user and system keys and credentials. It can loosely map to the Execute stage of the Cyber Attack Lifecycle and can occur in many other tactics of the ATT&CK framework as well. For example, Credential Access can occur to facilitate the Initial Access technique and/or the Privilege Escalation tactic. Credential Access was given its own tactic in the ATT&CK framework to better highlight the range of ways attackers obtain credentials for the system.
- **Discovery:** This tactic describes how attackers gain additional knowledge about the system and internal networks. It roughly maps to the Recon stage of the Cyber Attack Lifecycle. Examples could be listing all the local accounts on a compromised system, or port scanning for remote servers.
- **Lateral Movement:** This tactic describes how an adversary moves through your environment. This can be used in the Recon/Weaponize/Deliver stages of the Cyber Attack Lifecycle. Examples could be using stolen credentials to move to other systems or installing malware on removable USB drives that are inserted into a compromised system.
- **Collection:** This tactic describes how an attacker gathers data of interest to their end goal. It doesn't directly map to a stage of the Cyber Attack Lifecycle, but it highlights how adversaries often need to figure out how to access the data they want to steal. Depending on the amount of data they are stealing, they often need to employ techniques to archive that data to make it easier to exfiltrate. For example, an attacker may use a keystroke logger to collect data from multiple systems and store the results in an encrypted file that they periodically collect, versus having every single keystroke sent back immediately to their command-and-control server.
- **Command and Control (C2):** This tactic describes how an adversary communicates with a compromised system to control it. This tactic roughly maps to the Control stage of the Cyber Attack Lifecycle, with the difference being that this ATT&CK tactic focuses on the attacker-generated network traffic. For example, techniques include using web protocols to send commands, or the types of data encryption adversaries use to encrypt their traffic
- **Exfiltration:** This tactic describes how attackers copy and transfer data from a defender's systems. This ties to the Collection ATT&CK tactic and focuses specifically on how adversaries move stolen data back to systems they control.
- **Impact:** Unlike the other tactics, this ATT&CK tactic focuses on the attacker's end goals. For example, data destruction or data encryption may be used to enable ransomware attacks. This roughly maps to the Execute stage of the Cyber Attack Lifecycle.

Defenders may drill down into these different tactics, techniques, and procedures documented in the ATT&CK framework to see how current security controls perform. Since the ATT&CK techniques and procedures are also based on known adversarial actions, it can reduce the burden on organizations performing threat modeling as they can instead model attacker behaviors described in the ATT&CK framework.

As an example, Table 7 highlights selected procedures from the ATT&CK website [9] for the Lateral Tool Transfer technique, which falls under the Lateral Movement tactic. It lists specific procedures along with references to real-world examples of these procedures being used.

Table 7: Example ATT&CK Procedures from the Lateral Movement Tactic, Lateral Tool Transfer Technique

ID	NAME	DESCRIPTION
G0050	APT32	APT32 has deployed tools after moving laterally using administrative accounts.
S0190	BITSAdmin	BITSAdmin can be used to create Background Intelligent Transfer Service jobs to upload and/or download files from Server Message Block (SMB) file servers.
S0106	cmd	cmd can be used to copy files to/from a remotely connected internal system.
S0366	WannaCry	WannaCry attempts to copy itself to remote computers after gaining access via an SMB exploit.

At the time of this writing, there are three different versions of the ATT&CK framework: ATT&CK for Enterprise, ATT&CK for Mobile, and ATT&CK for ICS (Industrial Control Systems). At the tactics level, all three frameworks are roughly the same, with the main difference being that the Impact tactic was broken into additional tactics in ATT&CK for ICS to highlight trends that attackers have been following in the ICS field. The main substantial difference in these frameworks occurs at the procedures level, as attackers will use different tools and techniques depending on whether they are targeting an enterprise server, a mobile phone, or a building air conditioning system. All three of these frameworks can be useful for performing threat modeling on medical devices, but generally speaking, when modeling server software, ATT&CK for Enterprise tends to have the most relevant procedures to reference. When evaluating a remote healthcare solution involving phone apps, ATT&CK for Mobile can be a good reference. Finally, when evaluating medical devices themselves which are running embedded operating systems, ATT&CK for ICS can be a useful resource.

2.4.6. Tips for Documenting Assumptions and Results

There are several different goals with documenting results for “What can go wrong?” Documentation can ensure that in the next stage of “What are we going to do about it?” all identified threats are addressed. Documentation is also important for capturing and transferring knowledge to external partners, whether those external partners are other organizations, or partners in other business units within a single organization. Documentation provides a record to aid in future threat modeling exercises as products evolve. Finally, documentation can provide your team a tool to identify coverage of the threats identified, which can be helpful to gauge when it is time to move on to the next stage of the threat modeling process.

The tooling used to capture this documentation will vary based on an organization’s existing setup, the maturity of its threat modeling process, and the organization’s goals. In general, documentation can be categorized in the following ways:

1. **Document Describing the Threat Modeling Process:**
    - This document focuses on the threat modeling process itself. How are threats identified, how are they documented, what is in scope, what is out of scope, and what assumptions are being made?
    - Even if this information is captured in the output of the threat modeling process, it is useful to summarize it for external collaboration, as well as internal quality control. Specifying that insider threats are in scope provides an additional check to see if insider threats were identified in the threat modeling output. Clearly specifying that certain threats are out of scope or not considered can help highlight that gap to external partners.

2. **Ticketing Systems:**
    - Ticketing systems can be very useful for more mature products where a base-level threat modeling activity has already been performed, and an organization is focusing on identifying threats against new features or a changing environment.
    - Ticketing systems excel at creating an approval chain and documenting both the initial threat and decisions about how to address the threat.
    - A challenge with ticketing systems is that they can struggle to deal with the initial threat modeling of a new device, as the number of newly identified threats is high.
    - Another challenge is that ticketing systems often need additional documentation or processing to convert their output to a format that is suitable for sharing with external partners.
    - Finally, unless a customized reporting system has been incorporated into the ticketing system, it can be difficult to view which components of a system have had robust threat modeling applied to it, and where there are gaps, based on open and closed tickets. Therefore, it helps to pair ticketing systems with other forms of documentation. For example, organizations can include custom tags in tickets that refer back to components in diagrams created for the threat modeling process. This could be a particular flow diagram for protocols, or an entity in a DFD model.

3. **Diagrams:**
    - Diagrams often take the form of models of the system with threats applied to various components, steps, or stages. Diagrams also can be graphical representations of Attack Trees.
    - Diagrams can be very useful to identify where threats occur in the overall system, and to follow branching paths in a documented attack. Rather than reading a paragraph of text or following multiple lines in a spreadsheet, seeing the path an attack takes can be a very powerful tool.
    - Perhaps the biggest challenge with diagrams is they can get very crowded; they are not ideal to display a large amount of information and assumptions that an organization may have collected during the threat modeling process.
    - One strategy is to create unique identifiers for threats and/or components of threats that are captured in the diagrams. As a result, more information can be provided about those threats in an external document. Likewise, by providing a unique identifier, organizations can track what mitigations are applied to each threat.
    - Another strategy is to consider diagrams at high levels of abstraction and at lower, more detailed levels. Organizations will also likely have different types of diagrams that represent specific aspects of their system. For example, DFDs may be used to provide an overall view of the components, and a Swim Lane diagram to document network protocols. Organizations may also create different diagrams to represent the network view of a system versus the physical wiring.

4. **Spreadsheets and Databases:**
    - Often the data an organization captures about threats is stored in a spreadsheet or database. This is due to the amount of data necessary for storage, and the fact that many threats may be repeated with different variations throughout the system. It provides a convenient manner to determine the coverage of the threats that have been identified.
    - Usually there are multiple different tables, spreadsheets, views, and/or reports for that data to better understand the current coverage. For example, organizations may have a STRIDE per Element view of the data, linking to another spreadsheet with detailed information about each identified threat.
    - When many Attack Trees have been developed, they often share steps with slight variations at different junctures. A spreadsheet or database can be helpful to capture this situation and reduce the number of diagrams that need to be produced.

- Many spreadsheets and databases provide a way to generate diagrams from their data. This can aid in keeping documentation up-to-date, as they may be used to dynamically generate new diagrams and documentation from the data collected in the spreadsheets.

It can be useful to view the artifacts produced through the lens of how a third-party user would interpret it. This can be hard to objectively validate, as organizations may "fill in the gaps" in existing documentation with their own experiences and knowledge. One approach to address this is to bring in an external reviewer. For example, existing documentation may be used when onboarding a new team member. Another use of documentation can be to trace how threats have been managed over time. Having a history of when threats were identified, how they were identified, what was done about them, and how they have evolved can help third-party reviewers understand the system.

2.5. Question 3: What Are We Going to Do About It?

The terms "threat" and "risk" are often used interchangeably, but in a threat modeling context they have very different meanings. Threat is the possibility of a negative event happening. Risk is a much more complicated and subjective concept that weighs the probability of that negative event happening, as well as the potential impact if it does happen. At this point in the threat modeling process, the question of "What are we going to do about it?" is better understood as "how do we manage the risk from the threats we have identified?"

It is tempting to focus on defensive techniques. For example, if there is a threat "X", then apply solution "Y" to mitigate it. But threats are continuously evolving, and the systems being developed all have their unique features. Therefore, this Playbook will instead focus on the broader problem that frequently pops up in threat modeling: the silent acceptance and transferal of risk.

Overall, there are four main strategies for addressing threats:

1. Eliminate
2. Mitigate
3. Accept
4. Transfer

Each of those strategies will be discussed later, but an important goal of threat modeling is to apply one or more of those strategies to each threat, document the decisions made, validate strategies, and if necessary, re-evaluate threats. When that process is not followed, or threats are missed or "accepted," that threat does not go away. If a strategy for addressing that threat is not documented, that threat is silently accepted or transferred to the stakeholders of that medical device, which may be the threat modeling organization itself, customers, caregivers, or patients. The silent aspect of this risk acceptance and transferal is the key aspect that needs to be avoided. There will always be some level of risk acceptance and transferal. The important point is to document that risk and provide actionable information to all the relevant stakeholders.

As an example of this silent acceptance: if a device uses a code library with a known flaw in it and there is no documentation of the threat, the risk associated with it is being silently accepted.

Silent transference can be an even more insidious problem, and there are many ways this can occur. Often it can manifest when a threat model makes overly broad assumptions about the threat landscape. For example, a threat model could assume that a medical device must be connected to a secure network, and that it is the HDO's responsibility that the device never be exposed to any threats. An assumption like this transfers all responsibility to the HDO without giving them the information about which threats they need to defend against. Additionally, it is a poor assumption that a HDO's network will be secure, given the amount and types of threats an HDO network is exposed to. Making this type of assumption is likely to result in designs with insufficient security controls for realistic use environments.

Once again, it is the “silent” portion that threat modeling best practices seek to avoid. For example, an administrative web page for a device might not have a rate-limiting policy applied to it for failed logins to defend against brute-force password-guessing attacks. Also, the medical device manufacturer of the device likely does not have control over what passwords users pick or how the users protect their passwords. Finally, the HDO may ultimately control access to the web page through use of network segmentation and firewall policies. Therefore, some risk needs to be transferred to the HDO when they install the device. The HDO will be responsible for managing user password strategies and controlling access to the admin web page to limit attacks against it. But highlighting these threats, as well as recommended actions, can make HDOs aware of the specific risks they are accepting. This is a much better approach than simply saying that the device should only be connected to a secure network.

The rest of this chapter will focus on the four main strategies to address threats and risk, and then cover several techniques to attempt to identify risk based on known threats.

2.5.1. The Eliminate Approach

From a security perspective, eliminating threats is the most desired outcome. If the threat doesn’t exist, there is no risk from it. It also makes things simpler from a threat modeling perspective. Organizations may document that the threat no longer exists, why it no longer exists, and move on. The challenging aspect with eliminating threats is that it does not remove or eliminate adversary actions or behaviors. Instead, the process focuses on eliminating and/or changing system features or processes that the threat applies to. To take this to the most extreme example, if a medical device is never fielded, then no threats would exist against it. This is obviously an unrealistic outcome. Therefore, when leveraging threat modeling to eliminate threats, the goal is to identify and eliminate features that may provide limited value compared to the risks associated with them, and/or transforming the processes or features in a way that eliminates its exposure to certain threats.

An example of eliminating threats would be to avoid collecting data that is not relevant to providing care to patients. If data is not collected and stored, the threat of it being stolen is eliminated. Eliminated threats can also target certain components of a system versus the system as a whole. In the AMPS example, it may make sense to avoid storing patient contact or billing information in the phone app. Threats still exist regarding that data being stolen from the AMPSCS, but the threat of it being stolen from the phone app can be eliminated.

A related approach is to transform a process or feature to still provide the desired functionality while eliminating the identified threat. In the AMPS example, the AMPSCS include a feature that will email patients if their devices have not uploaded any new data to the cloud infrastructure. Various threats may have been identified against this email feature, so the decision could be made to send a push alert to the patient’s phone app instead of using email. Thus, the email feature and its associated threats are eliminated. When such transformation occurs, the key aspect to keep in mind is that new threats are being transferred to the new features or architectural changes. Using the previous example, under the updated design, the phone app may have new threats against it involving messages being pushed to it, and the overall architecture may have additional failure modes. The advantage of taking a transformational approach can be that these new threats may be easier to address than those identified in the original architecture.

Keep in mind that eliminating threats may involve the removal of code or functionality. Without it, threats may only be mitigated. Connected devices and systems will always face cybersecurity threats—they cannot all be eliminated. The key is to apply one of the other strategies (mitigate, accept, and transfer) and document the results.

2.5.2. The Mitigate Approach

Mitigating threats involves identifying, adding, and improving controls to stop attacks. This may involve the evaluation of existing features of a system and potentially creating new features to add, or settings to

configure. These security controls may already be in place, and an identified threat may already be mitigated by the time a threat modeling process identifies them. The key point is that both the threats and current security controls that mitigate them are documented. The reasons for this are twofold. First, when future threat modeling is performed, that knowledgebase will be preserved and will be informative. The second is that systems evolve over time. Even if new features are not added or changed, users of the system may apply additional configuration changes to it. Having documentation for the security controls can help identify what may happen if they are disabled, modified, or the architecture changes in a way that undermines the protection they provide—or if those controls are discovered to have unresolvable vulnerabilities in the future.

The mitigation control to apply will be situational for a system. At a high level, mitigation controls are often grouped into the following four categories, outlined in the NIST Cybersecurity Framework [10]:

- **Protect:** Protection controls make the threat less likely to have an impact. Protection could be regularly applying security patches, creating network segmentation, locking down user permissions, or employing encryption. Protection controls can be highly effective and are important to protect patient safety and privacy. At the same time, it is important to note that protection controls are rarely 100% effective. New vulnerabilities can be discovered and exploited before the developer is aware of them; attackers can get past firewalls and network segmentation; a user may still be able to utilize their locked-down permissions in unexpected ways; and encryption keys can be stolen. This isn't to criticize protection controls. To allow your system to provide value, most of your threat management strategies will likely center around protection controls. It simply is an acknowledgement that the protection they provide is rarely perfect, so it is important to document your assumptions around these controls, and to realize they will likely leave some residual risk that will need to be either accepted or transferred.
- **Detect:** Detection controls focus on identifying unwanted behavior of the system. This is useful to minimize the impact of an attack and can be thought of like audible alerts that a system generates when it starts to fail. Since no system can be perfectly secure, and no threat modeling process can identify all possible threats, having the ability to detect unwanted behavior can enable the system owner to take remedial action. Two important aspects of documenting detection controls are identifying who or what will be receiving and acting upon the alerts, and what the recommended response to those alerts is. If the detection control is to write a message to a log that nobody ever reviews, the threat mitigation value of that control will be limited. Another useful feature for detection controls is a way to test if they are working. Often, one challenge for a detection strategy is that usually nobody notices if it breaks, which lessens or eliminates its value. Having this testing functionality significantly improves the value of a detection control.
- **Respond:** Respond controls often are combined with detection controls. They are a way to react to unexpected states of the system. A respond control could be the ability for a system to work with degraded capabilities. For example, an "offline" mode could be entered where the device can continue to provide value if its network connectivity is interrupted. It could also take the form of a failover mode. Sometimes respond controls may not even need to know the system is in an unwanted state. An example of this could be a system that goes back to a known-good version of its operating system every time it is rebooted with no persistent memory, or a container that is routinely destroyed and recreated. Finally, respond controls may be based on a human element. Security operations center (SOC) personnel may counter an identified threat, or operators of the device itself act based on provided guidance.
- **Recover:** Recover controls focus on getting back to a known good state. There is often overlap with respond controls, but the goal of recover controls is to return to a state where additional actions to counter the current threat are no longer needed. An example could be moving from an offline or degraded operations mode back to having all systems online. Often, several different steps and processes are needed to do this. For example, data generated when operating in a degraded mode may need to be incorporated and saved. Systems may need to be patched to

mitigate the threat that caused the original issue. Also, verification may need to be done to ensure the system is working correctly before using it to provide clinical care. All these steps, processes, and procedures can be categorized as recovery controls.

Rating how strong these controls are is a decision to be made by an organization as it proceeds through the threat modeling process. One consideration to note: if threat modeling a mitigation control proves to be difficult, that control is more likely to be weak. As an example of this, the security of human-generated passwords can be difficult to threat model. What if someone chooses a weak password? What if they reuse that password across different sites? What if they fall to a phishing attack? Going through all these failure modes and attempting to understand the amount of threat mitigated by human-generated passwords is a difficult problem to threat model. This in turn implies that in general, human-generated passwords are a weak mitigation control without other policy or technical controls in place. For example, if a system were to require two-factor authentication instead, the threat modeling aspects may get easier. It remains true that the two-factor authentication process itself could fail, but its failure modes tend to be easier to define, which also implies that two-factor authentication is a stronger mitigation.

With respect to mitigation controls for various threats, one general resource that may be helpful is the MITRE D3FEND Knowledge Graph [11]. According to its FAQ, “D3FEND is a knowledge base, but more specifically a knowledge graph, of cybersecurity countermeasure techniques. In the simplest sense, it is a catalog of defensive cybersecurity techniques and their relationships to offensive/adversary techniques.” More specifically for what it is not, “D3FEND does not prescribe specific countermeasures, it does not prioritize them, and it does not characterize their effectiveness.” The top-level knowledge-graph can be seen in Figure 13.

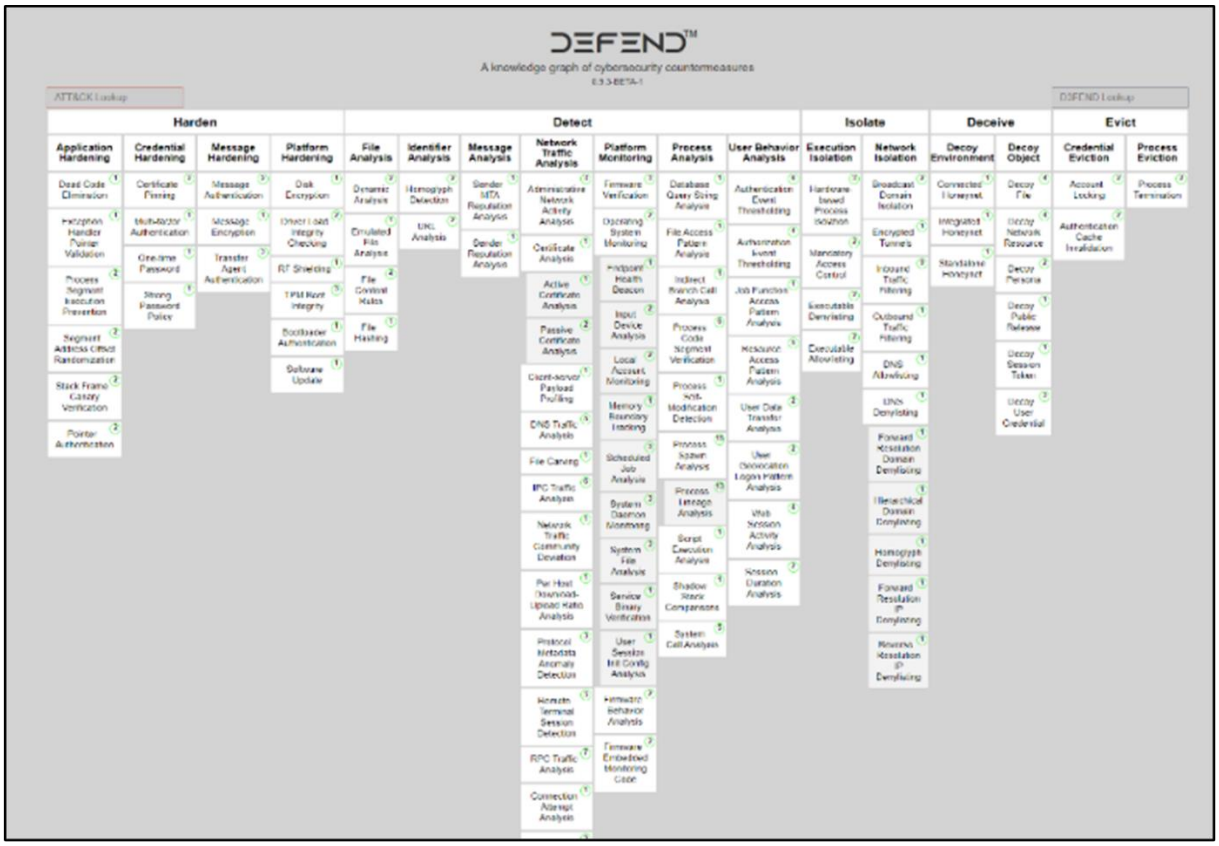


Figure 13: MITRE D3FEND Knowledge Graph

D3FEND can be useful to the threat modeling process by allowing users to search for ATT&CK techniques (described in Section 2.4.5) that may already have been identified as threats, and it will list various common mitigation strategies that can be applied to them. Perhaps more important, the D3FEND framework allows teams to use standard terminology for describing mitigation strategies, which helps with sharing knowledge and creating documentation. For example, the “Harden” tactic includes countermeasures such as application hardening (e.g., pointer authentication, process segment execution prevention, and dead code elimination); credential hardening (including certificate pinning, multi-factor authentication, and strong password policy); message hardening (e.g., message authentication and message encryption); and platform hardening (e.g., disk encryption, radio frequency shielding, and bootloader update). Another tactic, “Isolate,” includes countermeasures such as execution isolation (e.g., executable allow listing or hardware-based process isolation) and network isolation (e.g., encrypted tunnels, inbound traffic filtering, and broadcast domain isolation).

Another resource is NIST SP 800-53 “Security and Privacy Controls for Information Systems and Organizations” [12]. Chapter 3 of NIST SP 800-53 lays out a set of 20 control families, ranging from Access Control to Supply Chain Risk Management. Each of these families has a unique identifier to make it easier to share and reference. For example, the Identification and Authentication family of controls can be referenced as “IA.” Specific controls can then be given a follow-up numeric identifier. So, the “Authentication Management” control of the Identification and Authentication family can be referenced as “IA-5.”

What makes NIST SP 800-53 stand out is it contains a discussion of these controls, as well as tips and guides for how to implement and enhance them. For example, the guide for IA-5 contains descriptions of what they mean by the control, which includes verifying an authenticator such as a password, provisioning the authenticator, changing default authenticators after first use, updating authenticators as group memberships change, and so on. Furthermore, it lists enhancements and discussions. Continuing with IA-5, for password-based authentication, NIST 800-53 suggests maintaining a list of commonly used passwords as a block list to prevent those passwords from being selected by users of your system.

Since there are no standard categorizations of mitigation techniques that are specifically intended for medical devices, manufacturers might choose to define their own set of mitigations and use other sources to resolve any gaps. For a simpler informal model, commonly used mitigations can be grouped into broad categories such as access control, authentication, cryptography, environment hardening (e.g., in the OS), build/compilation techniques, using “least privilege,” isolation techniques (e.g., sandboxes), automated detection mechanisms (e.g., overflow protection), attack surface reduction, logging, monitoring/alerting, and others.

2.5.3. The Accept Approach

There will always be some level of residual risk that cannot be eliminated or mitigated. The key points are how the decisions to accept risk are made, and how the outcomes of these decisions are tracked.

Accepting risk is most straightforward when an organization is the one that is impacted and accepts the risk. When this is the case, having a policy in place to approve and track the risk is important. Scoring mechanisms and bug bars (to be discussed in Section 2.5.6) can be useful for this, as risks of certain scores or types may need different approvals based on their severity.

Accepting risk on behalf of customers is a much more difficult undertaking. Usually it takes the form of silent risk transference, which, as mentioned earlier, is probably the most impactful outcome threat modeling seeks to avoid. This may be avoided by being explicit and, instead of marking something as risk acceptance, approach it as a transfer of risk instead.

2.5.4. The Transfer Approach

Many products will be deployed by users and individuals who are not directly affiliated with the manufacturer of that product. This means they will be accepting risks that the product may possess. These users are putting the product on their network or in other untrusted environments where it may allow attackers in. They will be providing clinical treatments in which the product may fail and impact patient care. Often this transference of risk is handled through license agreements and terms of service. Other times this transfer of risk may be silent, with no indication that a threat against the system remains unmitigated. This transfer of risk may even be hidden in well-meaning but overly broad advice from medical device manufacturers (MDMs) to HDOs, such as, “this device must be installed on a secure network.” The goal of threat modeling processes is that this transfer of risk happens in a way that provides actionable information to users. Risk transference is an important reality: because products are meant to be sold and used by those other than their manufacturers, at least some level of risk transference will need to happen.

Effective risk transference involves documentation, approval processes, and liability.

From a documentation standpoint, it helps to provide any security guides, installation guides, and maintenance guides, as well as follow-up documentation that contains threat modeling outputs. The guides may document preferred strategies for the users to install and secure the device in their network, with the audience being a typical user or system administrator. The threat modeling output, including diagrams, identified threats, and their mitigations, are useful for more advanced users and organizations that may have non-standard setups or are trying to provide enhanced security.

Another aspect of documentation can be user interface warnings. Users can have insight into how the system is being used. For example, a user may know that a website’s security certificate is expired and choose to browse to that website regardless. It remains a balancing act to present threats and risk to a user in a useful way, but this real-time feedback and documentation can be a very powerful tool.

The approval process also becomes critical for risk transference. Having a documented procedure, following it, and tracking it for accepting and transferring risk to others is important so that the appropriate stakeholders can be involved in the process.

Finally, there are legal and liability aspects of risk transference. This Playbook does not discuss laws and strategies, but it is important to keep in mind who is responsible for certain aspects of the security lifecycle of the product. What is a company’s responsibility to remediate new security threats as they are discovered? What is the notification policy when this occurs? If a threat is exploited and a system is compromised, who is held liable? These are aspects of risk transference that need to be identified and documented.

2.5.5. Tracking, Documenting, and Assessing Risk Reduction Strategies

Some risk reduction strategies may apply to multiple threats, and they might require different amounts of time to implement. It can become difficult to track and understand the status of their adoption and implementation. It may be useful to maintain a separate list of individual strategies and use a unique identification scheme, such as “M-09,” “bug-1234,” etc. This can make it easier to link multiple threats that have the same strategy. It can also help to spot strategies that can apply to multiple threats, which may help with prioritization. Additional commentary for each strategy can also be centralized, such as the strengths and limitations of the selected strategy, and possible options for stronger strategies in future designs. In some cases, technology may have been purchased that offers a certain type of mitigation that can be used throughout several components and/or products.

When deciding which strategies to adopt, the following factors may be considered, especially for mitigations:

- **Impact to patient safety**: Many mitigations that make sense within an information technology (IT) environment may need special consideration before being applied in a medical device or clinical environment. For example, password-based authentication might not be appropriate in environments or use cases in which the patient's memory or dexterity might be reduced, or when access time is critical.
- **Human factors and usability**: Any technical, process, or workflow change that improves the security of the device requires consideration of how easily (and how safely) people can use the device. Examine the users' abilities (cognition, strength, dexterity, language, etc.); the device use environment (lighting, noise level, maneuverability); and aspects of the device user interface (interaction logic, indicators, buttons, knobs) [13].
- **Complexity**: Some mitigations might offer some significant benefits, but they might be so complex that it can be difficult to understand how the mitigation works and how it might interact poorly with other components or functionalities.
- **Difficulty of update**: Some mitigations might be powerful when considered in isolation, but they might be too difficult to update regularly, or they could be too brittle. It may be possible to build in some flexibility so that new technologies can replace aging ones with minimal modification, but such flexibility will need to be carefully weighed.
- **Technical or environmental limitations**: Devices might have limitations that are not easy to address, such as compute power in embedded devices with limited power supply, processor speed, or memory. Nonetheless, best practices suggest that the best available mitigations be adopted.
- **Relative technical immaturity**: Some mitigations might be available that are relatively new or immature from a technical perspective. These might be difficult to analyze or subject to rapid change.
- **Insufficient understanding or details** about the strength of the mitigation, especially if it depends on a third-party product. New frameworks and capabilities are frequently introduced to the market, which might provide significant benefits when used in the protection of medical devices. However, these capabilities might not be well understood or well documented due to intellectual property concerns by their vendors—and so it may not be appropriate to trust these vendor claims implicitly. Recognition by device manufacturers as to when a component is effectively a "black box," and might introduce some dependencies, is important. This is especially true if the capability is in an inherently trusted role, such as an enhancement that modifies the operating-system kernel to introduce another layer of protection or monitoring. The power of the mitigation might make it worthwhile to adopt, but it remains important to note the dependency and to revisit it over time.
- **Other concerns** not related to performance or protection of the device include such as difficulty of integration, limited maintainability, etc.

It is also important to be as realistic as possible about the capabilities of a mitigation. Some mitigations might not be as strong as advertised, or they might have certain dependencies that cannot be guaranteed during operation. It can be difficult to fully eliminate a risk unless changes are made to the design or architecture in a way that avoids the risk entirely, or the associated functionality is removed from the device. In many cases, a mitigation can only serve as a delaying tactic by introducing barriers to an attacker that are more difficult or time-consuming (but not impossible) to overcome.

Do not rely too heavily on mitigations that do not directly address the threat or limit the impact. For example, code obfuscation methods might be considered to make reverse engineering more difficult. These methods might make it more difficult for attackers to figure out how the software works, but they do not protect against any vulnerabilities that the code implements. In other cases, techniques such as Address Space Layout Randomization (ASLR) can make it more difficult for attackers to construct reliable buffer-overflow exploits, but ASLR can be defeated by using other minor bugs to infer the underlying

memory layout—and even in the best case, the usual response is to have a controlled exit from the code, enabling a denial of service.

When choosing to transfer the risk to other parties, be mindful of the ability of the other party to implement the desired countermeasures. While risk transfer may simplify the manufacture of the device, the goal is to ensure that risk to patient safety is minimized.

Organizations would optimally choose to use the strongest risk reduction strategies available, but they may not be immediately feasible due to factors outlined previously. For planning purposes, stronger mitigations may be included as requirements or features for future versions of a device. Attacks will only get better over time; it is important that defenses do the same.

If you choose not to adopt a mitigation that would obviously reduce risk, be sure that an explanation is documented that is acceptable to external parties who may have different priorities, such as HDOs, patients, vulnerability researchers, other business units, and regulatory authorities.

Finally, best practices encourage organizations to resist the influence of market pressures on the analysis of strengths of mitigations or artificially understate residual risk, as this is not in the best interest of patient safety.

2.5.6. Determining and Ranking Risk from Threats

There often is a need to rank threats based on likely impact. This can help prioritize development of mitigations for internal development teams, but also to convey estimations about how much risk is being transferred to other stakeholders. This section will focus on several different techniques that can be used to perform this ranking and/or scoring of threats.

2.5.6.1. Quality Gates/Bug Bars

One common approach in software development programs is the idea of quality gates. As the name implies, quality gates focus on ensuring a minimum level of quality of the product, and they tend to cover many different aspects of the software beyond its security. Examples of quality gates might be that all patches for a piece of software must pass pre-established regression tests; all compiler warnings of a certain type(s) must be eliminated; or certain insecure functions must not be used. Best practices suggest that quality gates also contain a standardized process for approving the code to be released and/or deployed that may contain exceptions to those quality gates. For example, violations of a certain type may require a project manager to approve accepting that violation, while more severe violations may require increasingly senior leadership to sign off.

Bug bars are a similar idea and focus on categorizing bugs or threats in a standardized and repeatable manner. An example can be found in the Microsoft Security Development Lifecycle (SDL), where bugs and threats are categorized if they apply to a server or client, and then a categorization of Critical, Important, Moderate, or Low is applied depending on details of the STRIDE threat that it is describing. More information about Microsoft's approach is available at [14]. As an example, assume identification of a denial-of-service threat against a server; following the Microsoft bug bar example, the next two main questions ask whether DoS is temporary or persistent, and whether it was caused by an authenticated user or an anonymous user. If it was a persistent DoS caused by an authenticated user, the threat would be categorized as "Moderate" by the bug bar. If it was a persistent DoS caused by an anonymous user, the threat would be categorized as "Important." This different level of categorization can then be paired with other program management processes to prioritize mitigating high-level threats, requiring additional sign-off to accept or transfer the associated risk if they are not mitigated.

2.5.6.2. Scoring Mechanisms (DREAD, CVSS Rubric, and NIST SP 800-30)

Several scoring mechanisms attempt to calculate the risk from an identified threat using a predetermined approach. One example is the DREAD scoring methodology. DREAD is a mnemonic for performing threat modeling against a given system, originally developed by Microsoft, although Microsoft no longer employs

it. While STRIDE provides a framework for loosely categorizing threats at a high level, DREAD provides a framework for discussing the potential risk of individual threats. The different categories of DREAD are listed in Table 8, and the "Example" column considers each DREAD element and scoring in the context of a clinical device designed to gather sensor readings over time.

Table 8: Summary of DREAD Mnemonic Elements

DREAD ELEMENT	DESCRIPTION	EXAMPLE
DAMAGE	How badly would a successfully exploited threat affect the target system?	*The threat could cause important medical sensor readings to be missed. Medium*
REPRODUCIBILITY	How easy is it to reproduce the attack against the target system?	*The threat requires physical access to exploit. Low*
EXPLOITABILITY	How much work is needed to launch the attack against the target system?	*The threat requires interfacing with a proprietary port. Low*
AFFECTED USERS	How many people will be impacted by a successfully exploited threat?	*The threat will impact one user per device. Low*
DISCOVERABILITY	How easy is it to discover that a threat has been used against a system?	*Exploitation is indicated by missing data. May require manual review. Medium*

Despite superficial similarities between DREAD and STRIDE, they remain different methodologies, and in fact both may be applied as part of a larger threat modeling effort to supplement each other. While STRIDE focuses on identifying threats, regardless of likelihood or impact, DREAD is a framework for scoring threats and prioritizing which ones may take precedence for remediation. DREAD is typically considered to be semi-quantitative in that a qualitative assessment is made (e.g., High, Medium, Low) and then a score is assigned. A qualitative/quantitative scoring rubric is not defined, so these criteria are up to the MDM to establish for their own organization. The lack of a pre-specified scoring methodology makes DREAD implementations variable and subject to different audiences (both internal and external to the MDM) potentially disagreeing with how the rubric was defined.

Once a score for a particular threat is assigned for each DREAD element, the resulting data can be utilized in many ways. One option would be to prioritize certain elements for enhanced review during later stages of the threat modeling process. Another option might require more detailed strategies to mitigate or eliminate risks based on their DREAD element scores.

Another scoring framework is the Common Vulnerability Scoring System (CVSS). CVSS is managed by FIRST.Org Inc. (FIRST), and the official documentation can be found at https://www.first.org/cvss/ [15]. As described in FIRST's mission statement, "The Common Vulnerability Scoring System (CVSS) provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity. The numerical score can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organizations properly assess and prioritize their vulnerability management processes." To make CVSS easier to apply to the specific patient safety concerns found in a clinical environment, MITRE created a rubric for applying CVSS to medical devices [16]. A rubric is a set of scoring criteria and guidance, and the rubric for applying CVSS to medical devices takes the form of a branching series of questions to identify a score for each portion of CVSS. Figure 14 shows an example rubric for identifying a score for the CVSS Attack Vector metric.

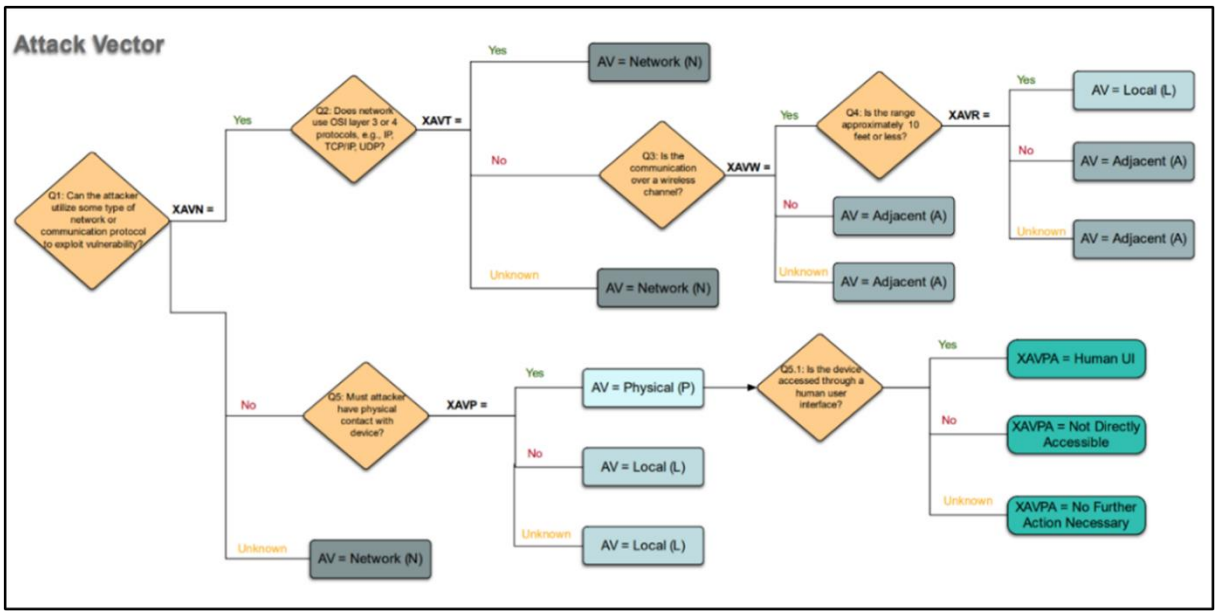


Figure 14: Example Rubric for Scoring the CVSS Attack Vector. Taken from [16]

Tools for applying CVSS to medical devices can be found in [17].

One final scoring mechanism is the NIST SP 800-30 “Guide for Conducting Risk Assessments” [18]. In appendices D through I, it contains a list of scoring criteria for determining risk. These appendices are briefly described below:

- **Appendix D Threat Sources:** This appendix contains assessment scales that categorize adversarial capability, intent, targeting, and effects. Categorizing the impact of a threat being actualized, as shown in Table D-6 of SP 800-30, can provide a useful metric, but keep in mind, its focus is on number of systems being compromised versus the impact of those systems being compromised.
- **Appendix E Threat Events:** This appendix focuses on different threats that can occur and can be useful for performing structured brainstorming during a threat modeling exercise. From a scoring perspective, Table E-4 of SP 800-30 describes the categorization of threats as either “Confirmed,” “Expected,” “Anticipated,” “Predicted,” “Possible,” or “N/A.” In practice, some of these categories can get difficult to distinguish, with a focus on whether a working proof of concept for the threat has been developed.
- **Appendix F Vulnerabilities and Predisposing Conditions:** This appendix attempts to evaluate the severity of the vulnerability, with Table F-2 of SP 800-30 providing a vulnerability severity score.
- **Appendix G Likelihood of Occurrence:** This appendix attempts to quantify the likelihood of a threat event occurring. One important point is that it contains two tables, one for adversarial and one for non-adversarial threats. For non-adversarial threats, it may be possible to estimate the likelihood of that error or accident occurring. It is much more difficult to predict the behavior of human threat actors.
- **Appendix H Impact:** This appendix focuses on the different types of impacts. Table H-3 provides an assessment scale of the impact of threat events.
- **Appendix I Risk Determination:** This appendix attempts to combine and organize the various assessments from previous steps.

For more information about the actual assessments, scores, and how to record them, please refer to NIST 800-30 [18].

2.5.6.3. Avoiding Misuse of Scoring Mechanisms

While these scoring mechanisms and others attempt to define a repeatable and consistent process for establishing a score, there is still a risk of “bad faith” manipulation of the process so that the resulting scores fall below some desired minimum or threshold. Best practices suggest that a security assessment score not be the sole criterion for deciding whether a risk is acceptable. Rather the security score can be utilized as an input to the safety risk assessment for determining risk acceptability. The process that was followed to generate the score will likely highlight key considerations that can assist in communicating with others and provide crucial context for more informed decision making.

2.6. Question 4: Did We Do a Good Job?

The question “Did we do a good job?” serves two purposes. The most immediate impact of this question is to ask, “Is the current threat modeling task complete?” While threat modeling is a continuous process and is most effective when updated and refined over the lifetime of a product, there is a time to finalize documentation. The second, longer term goal of this question is to ask if a threat modeling process is working as desired, or whether there can be improvements made. The challenge involves identifying where and how to receive feedback on the effectiveness of threat modeling processes.

One important note. It's tempting to point to a lack of compromises as a sign that a system is secure. Unfortunately, the lack of previous compromises by itself does not provide useful and actionable information on how a system will perform in the future. Similarly, a lack of reported vulnerabilities does not necessarily indicate that a system is secure, as undiscovered vulnerabilities may exist.

2.6.1. Sources of Feedback for a “Good Job”

There are several sources of feedback that can be helpful to identify the effectiveness of threat modeling processes.

The first source of feedback can be internal checks performed by an organization against the threats identified during this process, as well as the strategies developed to address them. As mentioned in Section 2.4.1, organizations may use the STRIDE per Element approach to detect gaps in the threats identified. Likewise, the list of high value dataflows laid out in Section 2.3.3 and check if they are being used by the system being threat modeled, and whether they have been evaluated. For reference, here is a list of example high-value dataflows that were mentioned earlier:

- Authentication protocols with external servers
- Programming and configuration commands
- Obtaining and validating software updates
- Sharing patient data with external servers
- Transmitting and silencing of alarms
- Procedures to restore from backups

This list is not definitive, and a system might have its own high-value dataflows that are unique to its architecture or intended use.

Trust boundaries in a system also provide a convenient check to see whether threat modeling has been performed across them. In addition, checking to see whether a strategy has been identified and documented for every threat listed may be useful.

Another source of feedback about the effectiveness of threat modeling strategies may be processes, artifacts, and design decisions made in the development process for a system that are influenced by the threat modeling effort. Has the design of the system changed based on findings? If the development team for a system has an issue tracker, how many of those issues are tagged “Opened due to threats identified during the threat modeling process?” When accepting and transferring risk, how is the approval process working, and do the different levels of the approval chain provide actionable feedback and guidance?

Were concerns raised that were later validated or rediscovered in the quality and assurance process? These questions can provide useful feedback regarding how effective threat modeling processes are.

The final source of feedback can be based on security assessments and outside events. Assessments can take many forms. For example, network scanning may regularly be performed against devices to identify undocumented services, code auditing tools may be leveraged to identify unsafe practices, or full red team assessments may be performed. In addition, external researchers may proactively discover and report new vulnerabilities in products. In such cases, it may be useful to return to the vulnerable product's threat modeling documentation to see if it was predicted and captured. Even if that vulnerability or attack doesn't target a specific system, it may be useful to look at what happens to additional systems to evaluate whether the security controls in place may be effective. Likewise, feedback may be available to evaluate whether security controls do work. Logs and records of unsuccessful attacks can help identify which security controls are providing value.

2.6.2. Checklist for Evaluating a "Good Job" in Documentation

When evaluating whether a threat modeling process has been effective, the following checklist may be helpful for analyzing documentation and other artifacts that are produced. The answers may vary widely as the threat model evolves over time.

- **Completeness**: Is all information within the STRIDE analysis (or other analyses) complete? Within each aspect of the threat model, are all sub-elements complete? For example, Attack Trees can be more dependent on ability and expertise of the threat modeler to come up with sufficiently comprehensive set of attacks that inexperienced developers might not think of. Were the strongest possible mitigations identified, and for each component? If there are gaps, is it well understood why those gaps exist?
- **Clarity**: Is each threat, mitigation, etc., clearly explained? Does it make sense for the intended audience(s)? Ideally, clarity would emerge naturally with the iterative nature of threat modeling across a variety of participants within the organization, but there would still be external consumers—and some issues of clarity could be explained in meetings or email without being updated within the documentation itself.
- **Specificity**: Is each element appropriately specific and at the right level of detail for the intended audience(s)? For example, giving the name of a protocol might be sufficient in some cases, whereas the protocol, version number, and configuration mode might be necessary for others.
- **Traceability**: Are interrelationships between different components of the threat model easily traceable? If identifiers are used, are they consistent and correct? Are there any identifiers that are cited but not defined? Are there multiple entries or descriptions for the same concept within the same context (i.e., duplicates)?
- **Consistency**: Is there consistency between the intended design and the implementation of the manufactured device? Documentation can change frequently due to the iterative approach of threat modeling, so inconsistencies can arise.
- **Roles and responsibilities**: When applicable to the threat model or risk analysis, are the roles and responsibilities of the HDO, patients, and other parties clearly identified, particularly when discussing risk?
- **Assumptions**: Are all assumptions clearly identified? Are these assumptions "reasonable"?
- **Rationales**: Are rationales for decisions included in sufficient detail, especially for cases in which risks are accepted or transferred; is there some ability to revisit why one mitigation is selected over another; etc.?

Threat modeling processes are most effective when they evolve over time. There will always be areas to improve. The key is trying to identify quality sources of data and feedback to make those changes.

3. Considerations for Implementing Threat Modeling

Chapter 2 introduced threat modeling concepts and techniques, applied these techniques to a fictional medical device example, and discussed some of the key considerations when developing a threat model for a medical device. Chapter 3 examines how to integrate threat modeling with product safety risk management processes, how to implement it within an organization, how to scale, and finally, discusses some existing best practices.

3.1. Threat Modeling and Risk Management

This section explores how threat modeling can be integrated with product safety risk management processes. “ANSI/AAMI/ISO 14971:2019 Medical devices — Application of risk management to medical devices” is a key standard for the safety risk management processes recognized by many regulatory authorities. It defines a process for assessing safety risks in medical devices based on identifying hazards and hazardous situations, and the sequence of events that might lead to harm. The risk of each hazardous situation is estimated as a combination of the severity of harm and the probability of occurrence of harm. Assessing cybersecurity risk is similar, but not identical. Estimating probability, or the likelihood that a cybersecurity risk can lead to harm, can be difficult and misleading due to inability to assess probabilities for a malicious actor. Instead, FDA’s postmarket guidance [19] suggests assessing the exploitability of the cybersecurity vulnerability. Threat modeling affords a fruitful approach to systematically identify the cybersecurity vulnerabilities, weaknesses, and threats to a medical device and the networked clinical environment in which it operates. Threat modeling can also help assess the exploitability and in turn, the need for urgency. Manufacturers can use the threat model as an input into the overall risk management of the device: identifying cybersecurity risks, determining the best approaches to controlling these risks, deciding when a risk might pose a safety concern, and assessing any potential risks introduced by security and safety controls.

MDMs have quality processes that manage safety risks throughout the product lifecycle, from concept through launch, maintenance, and decommission. Both “AAMI TIR57: Principles for medical device cybersecurity – Risk management” [20] and the HSCC Joint Cybersecurity Working Group’s “Medical Device and Health IT Joint Security Plan” [21] (hereafter referred to as “JSP”) suggest frameworks for applying the risk management principles of ANSI/AAMI/ISO 14971 to cybersecurity risk management and risk assessment. Risk management processes generally begin with risk analysis, followed by an evaluation of the risk and, if needed, implementing control measures. An important part of this process is ensuring that the risk control measures do not introduce new sources of risk. Other risk management processes include validation and verification of the design requirements and postmarket monitoring.

In the following discussion, these frameworks will be used to introduce the vocabulary to describe how threat modeling can provide input into and interact with a manufacturer’s quality processes.

3.1.1. Threat Modeling and Cybersecurity Risk Modeling

The JSP creates a framework for incorporating cybersecurity into a device manufacturer’s existing quality processes throughout the development lifecycle from concept through commercialization. Figure 15 illustrates the alignment: the top row represents the product lifecycle phases, the center depicts the cybersecurity processes and activities, and the bottom rows show the quality processes. The risk management processes described in the JSP ensure that “cybersecurity risks identified during design, development, or post launch complaint handling are properly analyzed, evaluated, and documented.” [21]

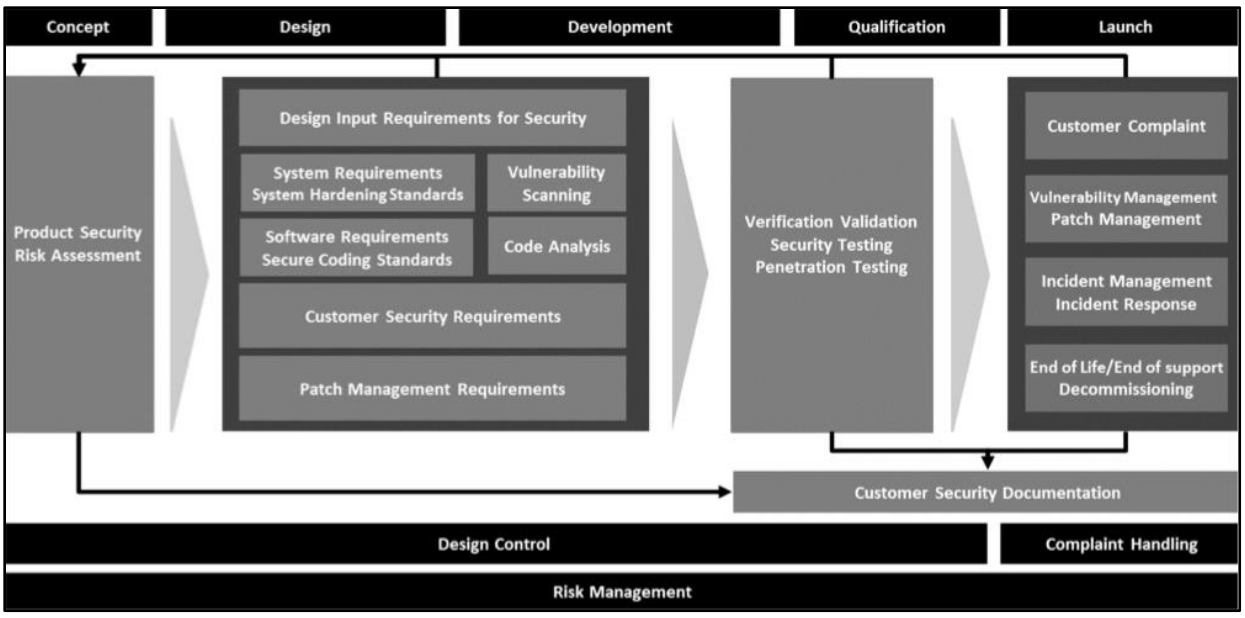


Figure 15: JSP Product Security Framework. Reprinted with permission from [21].

As seen in Chapter 2, threat modeling is an iterative process. The risk management processes in the JSP framework include built-in feedback loops that support iterating on the product's threat model. This allows issues discovered through these processes to be addressed. Ideally, threat modeling begins during the concept, requirements engineering, and early design phases. During product development, vulnerability scanning, code analysis, penetration testing, and other security testing processes can identify additional vulnerabilities, weaknesses, and potential threats that need to be included in the threat model and may require additional controls. After product launch, existing risk management processes including complaint handling and product monitoring may also identify new risks that may benefit from additional threat modeling and mitigation.

The central elements in the JSP framework are the security activities and processes for managing cybersecurity risk. The JSP discusses these activities in the context of where they align with quality processes: cybersecurity risk assessments, design control, and complaint handling and reporting. Threat modeling informs and provides inputs to the activities described in the JSP.

3.1.1.1. Cybersecurity Risk Assessment

Cybersecurity risk assessments are conducted throughout the product development lifecycle to determine the impact of known vulnerabilities or potential cybersecurity risks on patient safety, business operations, patient privacy, regulatory compliance, etc. Assessments also support prioritizing cybersecurity risks. Cybersecurity risk assessments are most effective when they reflect the operational environment and intended clinical use cases of the product.

Threat modeling helps identify vulnerabilities in the medical device and its components. In some cases, these are known vulnerabilities, perhaps in third-party components. In other cases, these may be vulnerabilities or weaknesses in the design or implementation of the system. Scoring methodologies, such as those described in Section 2.5.6, among others, can be used to assess these vulnerabilities and weaknesses. Best practices suggest that cybersecurity risks with a safety impact are also assessed in the safety risk assessment, and conversely, that safety risks with cybersecurity impacts are assessed in the cybersecurity risk assessment. Cybersecurity risk assessment processes is most effective when the relevant risk-related information is included in the manufacturer's safety risk process as appropriate, such as Preliminary Hazard Analysis and Failure Mode and Effects Analysis, etc.

3.1.1.2. Design Control

The JSP states that "[d]esign controls are an interrelated set of practices and procedures that are incorporated into the design and development process…that make systematic assessment of the design an integral part of development." [22] Design control principles are applied during product development and each new release; they are also applied to third-party components used in the product. The central part of Figure 15 shows the various JSP framework activities that can be applied from product concept through product qualification: design input requirements for security, system hardening and secure coding standards, code analysis, penetration testing, and other V&V and testing procedures.

The process of threat modeling identifies both potential threats and one or more risk control strategies (eliminate, mitigate, accept, or transfer) that can be applied for each threat. These threats and risk control strategies may provide additional design input requirements for security, and may inform system hardening, secure coding activities, and testing strategies. On the other hand, some of these later stage security activities (e.g., testing and V&V) may provide input back into the threat modeling process to help make the model more complete.

3.1.1.3. Complaint Handling and Reporting

After product launch, the complaint handling system provides a mechanism to obtain feedback from customers, including on cybersecurity issues. New vulnerabilities may be discovered. Best practices encourage manufacturers to have incident response plans in place, including coordinated vulnerability disclosure programs and processes, to support timely and accurate actions and communications. When new vulnerabilities are discovered, the cybersecurity risk management processes described above will be used to address them. Sharing information through DHS/CISA alerts and Information Sharing and Analysis Organizations (ISAOs) about vulnerabilities, especially those discovered in third-party libraries that may be widely used in medical devices, helps strengthen the overall cybersecurity of medical devices.

The issues that are discovered through the complaint handling system and other post-product launch activities (e.g., vulnerability monitoring, periodic security testing, product field monitoring, etc.), as well as those discovered in devices across the medical device sector and conveyed through the threat sharing activities mentioned above, may then be threat modeled to assist with their remediation. If this occurs, best practices suggest that the threat model be updated to reflect these new risks. It may also lead to additional design controls.

3.1.2. Mapping Threat Modeling to Security Risk Assessment

Another way to think about the relation between threat modeling and the quality processes is to map the four questions against the stages of the risk management process. The JSP is agnostic regarding whether a medical device's cybersecurity risk assessment is carried out separately or as part of the safety risk assessment. AAMI TIR57 recommends that these assessments be separate processes to ensure that the security harms not relevant to safety are appropriately managed, but as shown in Figure 16, these risk management processes are closely coupled to ensure that the risk control measures introduced in one process don't lead to unacceptable new risks in the other.

Figure 16 annotates the risk management processes from AAMI TIR57 with the four questions on threat modeling. The four questions related to threat modeling may be mapped to the JSP security activities aligned with the quality processes and the security risk process articulated in AAMI TIR57:

What are we working on? The JSP cybersecurity risk assessment and AAMI TIR57 security risk analysis step both include conducting a product or asset inventory, respectively. AAMI TIR57 suggests that the asset inventory includes information about information that is stored in, used by, or transmitted by the device; the device itself; and the software and hardware components included in the device. Thus, the asset inventory can provide input for developing the system models described in Chapter 2, to

understand the entities that comprise the system, the interactions between them, and the trust boundaries.

What can go wrong? Another activity during the JSP cybersecurity risk assessment and the AAMI TIR57 security risk analysis step is the identification of threats and vulnerabilities. AAMI TIR57 focuses on identifying threats from threat information that is shared by customers, found in government alerts, and communicated through information sharing and analysis organizations. The JSP focuses on design control activities described above, including vulnerability monitoring, vulnerability scanning, code analysis, security audits, and customer complaints. In addition to the sources identified by the JSP and AAMI TIR57, one or more threat modeling techniques can be used to perform a structured analysis of the product to identify additional potential threats. Answering this question may yield additional design input requirements for security as described in the JSP's "Design Control."

What are we going to do about it? The security risk control step in the AAMI TIR57 security risk process involves implementing appropriate security risk controls. AAMI TIR57 recommends using a risk control hierarchy that parallels ANSI/AAMI/ISO 14971 where risk control options are pursued in priority order: inherent security by design, protective measures, and information for security (e.g., documenting security requirements placed on the customer and/or patient). These risk control options align with the strategies described in Chapter 2: inherent security by design may eliminate the threat, while protective measures may mitigate the threats, and providing information for security can eliminate silent acceptance or transference. Evaluating residual risk and performing a risk/benefit analysis contributes to answering "What are we going to do about it?" by helping inform the choice of strategy to address the threats and risk. In the context of the JSP's "Design Control," answering this question may initiate a new round of design to manage the threats (i.e., adding the selected risk control options to the product design).

Did we do a good job? "Did we do a good job?" in fact asks two questions: 1) "Did we do a good job of identifying threats?" and 2) "Did we do a good job of identifying and implementing risk controls?" Many of the security activities described in the JSP "Design Control" (e.g., vulnerability scanning, security testing, and penetration testing) can help answer the first question from product concept through qualification. Both the JSP and AAMI TIR57 highlight the postmarket feedback loop (from customer complaints and monitoring devices in the field) and ongoing information sharing activities that identify potential threats that might be missing from the current threat model. The activities to evaluate overall residual risk acceptability described in AAMI TIR57, which also focuses on security testing, may help answer the second question about controls. But be wary of relying on testing. Passing a test does not mean your device is secure; it only means that it is not vulnerable to the threats that were tested for in the way that testing was performed (e.g., testing may impose time restrictions that prevent complete analysis from taking place, or insufficient documentation may have been made available, forcing testers to spend valuable up-front time to learn how the system works instead of figuring out how things can go wrong). Another way to evaluate the threat model is to involve experts outside the threat modeling team to evaluate the model. Answering "Did we do a good job?" will often lead to revisiting the threat model and possibly new design activities to address any additional findings.

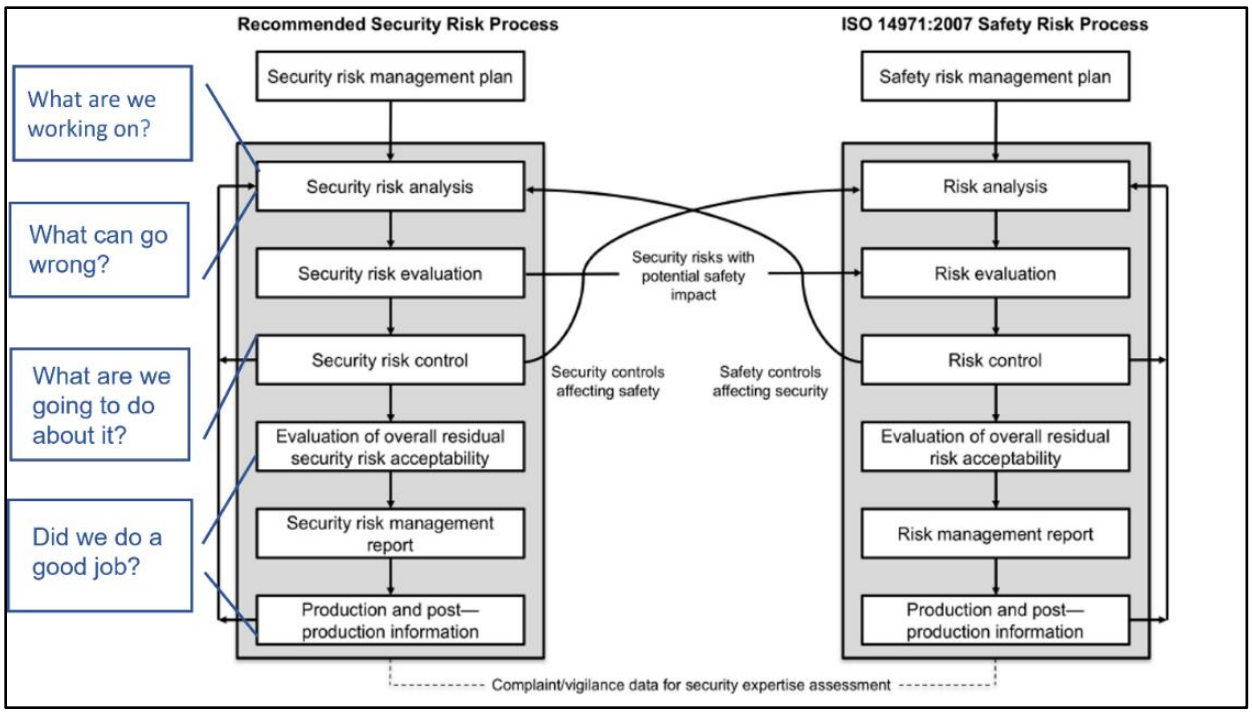


Figure 16: Threat Modeling and the AAMI TIR57 Risk Management Processes. Adapted from [20]

3.2. Organizational Adoption

In addition to understanding how threat modeling interacts with the quality processes, there are other considerations for manufacturers when implementing threat modeling. This section reflects activities initiated by MITRE and MDIC, which included facilitator interviews, discussions during the bootcamps, and the post-bootcamp surveys to discuss different organizational approaches, challenges in adoption, and different methodologies and tools.

3.2.1. Organizational Approaches

Through engagement with the bootcamp facilitators and participants, it was found that manufacturers have implemented different approaches as to who performs threat modeling within their organization: centralized security team, product teams, hybrid, or consultants.

Smaller companies may have a single threat modeling group or no in-house threat modeling staff at all, while larger companies may have multiple centralized threat modeling groups, one for each business unit. These centralized groups may perform the actual threat modeling for product teams or may provide consultants to assist the product teams. Many of the companies with centralized threat modeling groups are just starting to implement threat modeling and have only a handful of experts. Companies with sufficient security expertise may have each product team develop their threat models. Distributing threat modeling throughout the product teams provides a clear value: that threat modeling is being conducted by people who understand the product. Other companies have developed hybrid approaches where a group of threat modeling experts may be embedded in the product teams while still part of the central cybersecurity group. This allows the central group to provide broad-based input, advice, and services to the product teams who create the threat models. The centralized threat modeling team might maintain responsibility for a number of security-related activities for the entire company or business unit, such as developing a threat catalogue (of threats and mitigations) that the product teams can use as they identify “what can go wrong?”, providing templates to help produce more consistent documentation, helping the product teams integrate with the quality processes, qualifying threat modeling tools that can be used

enterprise-wide, and serving as independent reviewers of the threat models produced by the product teams.

3.2.2. Challenges and Current Practices

Interviews and surveys identified challenges and practices to address these challenges.

One key challenge is scaling threat modeling within an organization. Having a small team attempting to produce the threat models for an enterprise is not sustainable. Several companies are taking the hybrid approach described above to spread threat modeling throughout the organization. The central team serves as subject matter experts and provides resources as described. Initially, the central team may develop the threat models, but the goal is to train members of the product teams to perform threat modeling themselves. Organizations that do not have a core threat modeling team may hire consultants to engage in similar threat modeling bootstrapping activities when the organization does not have internal resources and/or expertise. That brings its own set of challenges: it can be time-consuming to bring the consultant up to speed on the system.

Another challenge is getting buy-in: the business units understand the safety risk management processes of ANSI/AAMI/ISO 14971 but may be less familiar with security risk management and the role that threat modeling can play. In addition, these business units may have different threat modeling needs because of the differences in device architectures, operating environments, intended use, modularity, etc. Finally, the members of the product teams and supporting teams have a variety of roles (e.g., systems engineers, design engineering, security risk analysts, safety engineers, design V&V engineers, and regulatory specialists), and they may need to understand how threat modeling fits into their purviews. An organization could stand up a threat modeling community of interest to share experiences and learn from one another.

Organizations that have not made threat modeling a part of their quality processes can introduce threat modeling processes that will facilitate subsequent integration. Some of the processes mentioned above can be introduced to ensure a more systematic and consistent threat modeling practice: document threat modeling processes, define the artifacts that will be produced, and develop approaches and artifacts to support traceability to track status of risks and map the risks to new requirements. Threat models can be validated through the organization's V&V processes during development (through penetration testing and other security testing) and by bringing in external groups, who have sufficient distance from the product, to validate the model. In turn, threat models can provide inputs into these design controls. Finally, it is important that threat models be updated through feedback from testing during development and field monitoring/complaint handling after product launch.

3.3. Methodologies and Tools

Chapter 2 described several threat modeling methodologies in detail—STRIDE, Attack Trees, ATT&CK, and kill chains—discussing some pros and cons of each, and how multiple methodologies can be used together. These methodologies are widely used, and the tips for using these methodologies provided in the playbook are included to help an organization get started with threat modeling. There are other threat modeling methodologies that organizations could consider as they become more experienced. Some of these additional methodologies are described in the Open Web Application Security Project's (OWASP) Threat Modeling Cheat Sheet [23] and the Software Engineering Institute's paper on threat modeling approaches [24].

Although threat modeling can be carried out with whiteboards and paper, it is valuable to create digital artifacts that can be stored in risk repositories and other databases. Various tools can be used to create system diagrams and manage the threats that are identified. These tools can be as simple as PowerPoint for drawing diagrams and Excel spreadsheets for tracking risks. An organization that is already using tools like Visio or Enterprise Architect can use them to create the diagrams that help answer "What are we working on?" But there are several tools, some open source, some commercial (sometimes with a

limited community version), and some free, that can be used to help construct threat models supporting diagramming, generating, and assessing threats based upon different methodologies, and suggesting mitigations. Some of these tools are listed in Table 9.

Table 9: A Selection of Free Threat Modeling Tools

Tool	Methodology	Type	URL
Microsoft Threat Modeling Tool	STRIDE	Free	https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling
IriusRisk Threat Modeling Platform	Various	Commercial and community	https://www.iriusrisk.com/threat-modeling-platform https://github.com/iriusrisk/IriusRisk
OWASP Threat Dragon	STRIDE, LINDDUN[2]	Open source (desktop and web apps)	http://docs.threatdragon.org/

[2] LINDDUN is a privacy-centered threat modeling methodology developed by the DistriNet research group at KU Leuven [37]. It stands for Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, Non-compliance.

4. Summary

This Playbook introduced threat modeling through the Four Questions Framework. It described several threat modeling methodologies and how they can be used within the framework. It presented an end-to-end fictional medical device example, along with additional examples used to highlight specific considerations when threat modeling medical devices. Finally, it discussed how threat modeling interacts with security and safety risk management and provided some strategies for integrating threat modeling into business processes based on current stakeholder practices.

When threat modeling medical devices, it is important to consider harms beyond those of physical injury and patient harm that could be directly caused by the device when a risk is realized through an attack. Since medical devices operate in networked clinical environments, an attack against a device may lead to a broader set of harms, including data breaches, disruption of clinical operations, and opportunities for pivoting into a hospital's IT infrastructure.

Threat modeling is a "team sport" that is most effective when conducted by cross-disciplinary teams bringing together expertise in traditional medical device development (safety perspective) and cybersecurity product development (security perspective).

Threat modeling takes place throughout the lifecycle. Threat modeling begins at concept/early design when the greatest flexibility exists to impact the product's design to improve its safety and security. But threat modeling does not stop there – it is a process that is carried throughout development and deployment. Best practices encourage organizations to continually update their threat models based on what is learned from development testing and validation, and then from feedback, field monitoring, and complaint handling postmarket. Further, threat modeling is not just for new products, but may also be useful when carried out on legacy devices.

Threat modeling identifies threats that could adversely impact the safety and security of a medical device. Threat modeling is an information-generating process that informs quality processes activities. Creating a threat model is not a paperwork exercise to check a compliance box. Instead, the threat model informs decisions about design, development, testing, and postmarket activities. It serves to document those decisions for internal stakeholders, customers, and regulatory reviewers.

Appendix A. Additional Fictional Medical Device Examples

This Appendix introduces two new medical device examples to build on the techniques discussed in Chapter 2. As with the AMPS system, these two additional medical device examples are ***fictional*** and have been created for the purpose of instruction. While this playbook cannot cover every possible device or scenario, these examples are designed to focus on common aspects that show up when threat modeling medical devices. The goal of introducing these devices is to demonstrate how threat modeling techniques can work together and highlight several challenges that can arise in threat modeling processes. Therefore, this Appendix will cover only a subset of threats against the full devices and will leave a vast majority of the threat modeling process as an activity for readers to conduct on their own.

Keep in mind that this Appendix is not intended to be an answer key. As these examples are fictional, there is a considerable amount of room for development of the finer details on how these devices are designed and operate, as well as the threats and controls that would need to be identified and addressed. Consequently, the discussion points brought up in this Playbook represent only one of many views on how these types of devices can be modeled. Furthermore, several diagrams and images in this Appendix contain intentional errors or are left incomplete for the purpose of discussion.

A.1. Fictional Example: The Stroke Nerve-Affective Photography System (SNAP)

The following is a feature overview for the "Stroke Nerve-Affective Photography" system. This represents a hospital clinical workflow with a fictional new diagnostic imaging device. This example was fabricated to avoid focusing on any specific clinical treatment technology. Note that this is not written to be an exhaustive description of the device. Readers are encouraged to bring their own knowledge to this example and fill in the details where necessary. Note: while it is discussed in this writeup for context, the AMPS device is outside the scope of this threat model.

The Stroke Nerve-Affective Photography system:
The Stroke Nerve-Affective Photography system (SNAP) is a diagnostic medical device, designed for use by patients who are either at risk for or recovering from a stroke. The SNAP system is in the same device class as medical imaging devices, such as magnetic resonance imaging (MRI) or X-ray machines. As such, it is expected that SNAP will be deployed in a dedicated space located inside a medical facility. Using an array of fictional sensors, SNAP can render full three-dimensional models of a patient's nervous and circulatory systems that a doctor can use for deeper analysis of underlying health conditions. The system cannot predict if a stroke is imminent. However, it can help visualize areas where a stroke has occurred or is likely to occur.

- Period of expected use: Continual (with different patients) in five-minute sessions
- Medical capability: Imaging

SNAP Core Use Case:
After a recommendation from her doctor, Alice has been wearing an AMPS device for three weeks. During this time, readings from the AMPS device have been uploaded to the cloud and reviewed by Alice's primary care physician. During her next visit, her physician informs Alice that the data indicates she is at a high risk of stroke. To gain further insight, her doctor has scheduled her to come in for a SNAP scan to identify potential areas of concern. Alice's doctor will analyze the scan results and will recommend additional monitoring or treatment options if deemed necessary.

SNAP Core Technology:

- The SNAP scanning base. Pictured in Figure 17 (component 2), the base is a large platform mounted to the floor with three vertical guidance rods with linear motors to mount and move a scanning ring (component 1).

- The SNAP scanning ring. Pictured in **Error! Reference source not found.** (component 1), the s canning ring contains a fictional sensor on a motorized track that can traverse the ring. Meanwhile, the ring itself is mounted on the SNAP scanning base and can move up and down with a patient in the center of it.
- The SNAP control station. Pictured in **Error! Reference source not found.** (component 3), the c ontrol station serves as a dedicated management terminal for configuring scanning parameters and initially managing image data.

SNAP Concept Diagram:

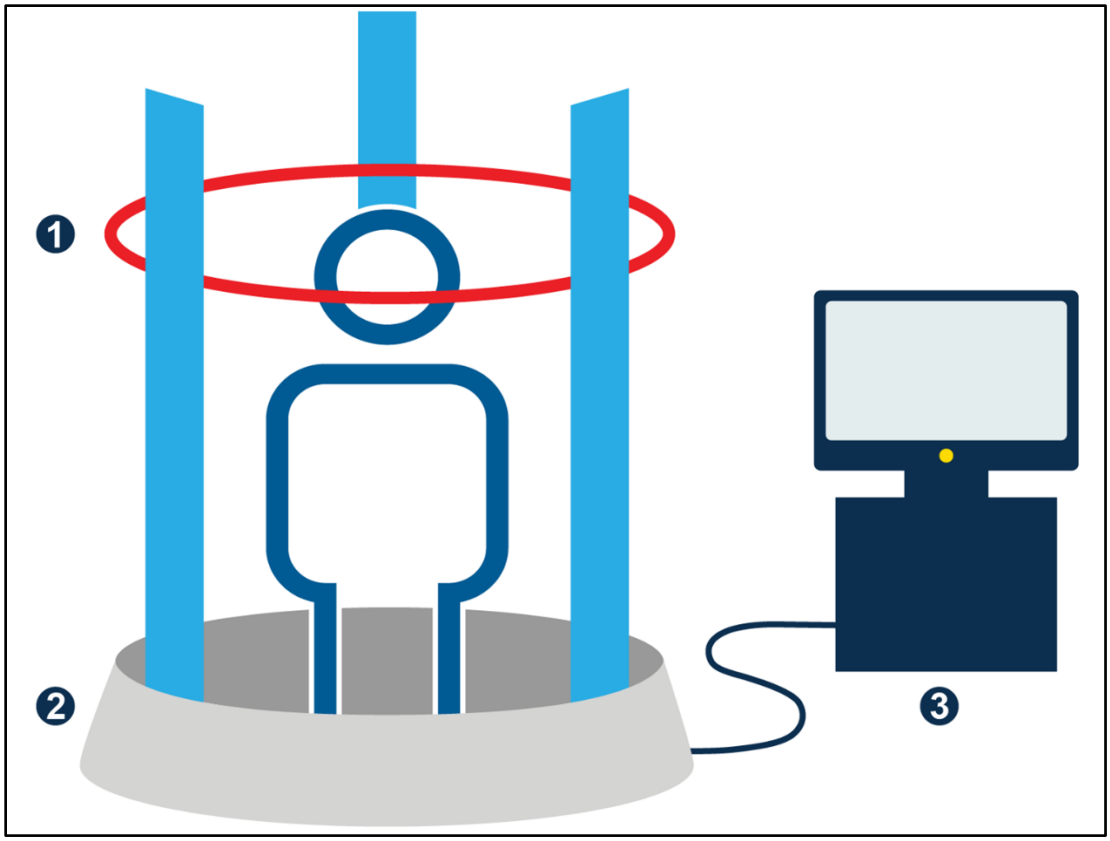


Figure 17: Diagram of the SNAP System. 1) The SNAP Scanning Ring. 2) The SNAP Scanning Base. 3) The SNAP Control Station.

SNAP Scanning Base:

- Weight: 150Kg
- Power source: Direct 120v wall connection
- Multi-cable connection to the sensor ring for data and power
- High-speed wired data connection to the control station
- Three vertical guidance rods, each with an independent linear motor
- Onboard computer and flash storage
    - Can temporarily store data for three patient scans for up to 24 hours
    - In case of connectivity issues, will hold old scans until requested by the control station
- USB-C port for diagnostic access and firmware updates

SNAP Scanning Ring:

- Weight: 7Kg
- Size: 120-centimeter diameter ring, attached to vertical alignment poles
- Power source: Connection to scanning base via power cable

    - Power and data cables can be disconnected from the base to allow servicing and replacement of the ring.
- Proprietary multi-sensor imaging array using the visible light spectrum
    - Note: this is a fictional sensor that allows for internal imaging, like X-ray or MRI technology. For the sake of this example, the sensors are assumed to pose lower levels of risk than that of an X-ray or MRI.
    - The scanning procedure automatically moves the ring up and down several times over the course of five minutes. One full up and down sweep takes approximately 10 seconds. A patient may be seated or standing.
    - The scanning ring is held in place by a set of three vertical guidance rods. Each rod is equipped with a linear motor to raise and lower an anchor point for the ring. Software is responsible for moving the anchor points in a synchronized fashion (position, speed).
    - The sensor itself is on an internal track inside the ring and will rotate around the patient at a maximum speed of 30 centimeters a second. The track and sensor are completely enclosed inside the ring.
- Hardware shutoff switch to disable the scanning ring in case of emergencies. This is located on the outside of the ring, intended for use by the technician.

SNAP Control Station:
The control station is responsible for managing the parameters of the scanning platform, allowing for on-site visualization and facilitating data transfer to the hospital electronic health record (EHR) system and Picture archiving and communication system (PACS). It is a dedicated workstation computer running Windows 10 with proprietary medical software to interface with the SNAP platform. It has a range of deployment options and can be integrated into a hospital network in a similar fashion to a nursing station. The SNAP control station by default sends data to PACS using the DICOM protocol, and it communicates to the hospital EHR using Health Level 7 (HL7). There is a separate software suite developed by an independent vendor that can help translate the HL7 generated by the control station to be compatible with a hospital's EHR.

The SNAP control station has the following capabilities:

- Large amounts of onboard storage for temporarily storing patient models
- Dedicated data cable connecting the scanning base to the control station
- Dedicated ethernet + Wi-Fi card for connecting to the internal healthcare network
- Scans are initiated using the dedicated SNAP program hosted on the control station
    - Scans do not require a connection to the internal healthcare network to be initiated.
    - Scans are automatically deleted from the control station after two weeks.
- Visualization software, allowing for exploration of a patient model
- Ability to export and transmit data in standard formats, such as DICOM or HL7 Fast Healthcare Interoperability Resources (FHIR)

Patient Safety Considerations:
Note: This is not an exhaustive list of potential harms. Other medical devices may have more complex harms or more severe potential for cybersecurity compromises. Failure modes may be more complex in interactions between systems.

- Prolonged or excessive exposure of ring light in one location for longer than 10 minutes will give the patient localized skin damage at the affected area. This is not expected to happen under normal operating parameters.
- Improper maintenance of the vertical guidance rods may lead to inconsistent movement or jamming, which will put unwanted tension on the scanning ring.
    - Excessive tension may lead to ring damage or destruction, with the possibility of ring debris striking the patient.

- Excessive tension may cause the vertical guidance rods to collapse toward the patient, potentially striking them or pinning them under equipment.
- Injuries resulting from mechanical failure could range from bruising, laceration, puncture, or even fatality.

If functionality described above is not documented, feel free to record assumptions about how the device works, and for a threat model, when working through these examples.

A.1.1. (SNAP) Objectives of Discussion

Ideally, threat modeling will be performed early in the design process. This often means that descriptions and understandings of the system may be ambiguous or incomplete. For example, in the write-up for the SNAP system, there was no description of the process a clinician follows to enter a patient's information, perform calibrations, initiate a scan, or abort a running scan. Identifying what information impacts a threat modeling assessment is a skill that improves with experience. In cases where a feature or a workflow hasn't been settled upon, being able to threat model different possibilities can help identify issues early and provide guidance to influence the system's design. Conducting threat modeling early in the design process also helps generate documentation about design decisions that may not be captured otherwise.

A similar level of ambiguity is present in this Playbook's discussion of the SNAP system. Rather than attempting to perform a full threat modeling exercise of this fictional device, this section will focus on common issues and challenges when threat modeling medical devices. The two following areas will be highlighted in the SNAP example:

1. Threat modeling the connection and communication between a PC workstation and a directly connected medical device
2. Threat modeling the communication between a system and an HDO's network

These two areas of focus helped drive the development of this example. The earlier AMPS example described a home telehealth device and thus wasn't being directly connected to an HDO's network, so the SNAP system was designed to interface with other HDO services. The separation of the SNAP control station and the SNAP scanning system was included to mimic many other medical devices that interface with software running on more traditional desktop computers.

A.1.2. (SNAP) What Are We Working On?

Often, starting the threat modeling process is the hardest part. In the case of SNAP, a high-level feature overview explaining how the device is intended to operate has been provided. While this write-up gives an idea of what features are present in the system, it is not an airtight description of how each piece accomplishes its function. In lieu of extensive documentation, there are always a few immediate questions that may assist in building out an initial system diagram.

- What new pieces of hardware and software have been developed for this system?
- What communication channels exist to connect these pieces of hardware and software?
- What external entities do these pieces of hardware and software need to communicate with?

When reviewing the provided documentation for SNAP, several features might stand out to help answer those questions:

(Question): What new pieces of hardware and software have been developed for this system?

- (Initial Assumption): In the documentation, SNAP is described as having three main hardware components – the scanning ring, the scanning base, and the control station.
- (Discussion): At this stage it is not entirely clear if differentiating the scanning ring from the scanning base will provide value for the threat modeling process. The fact that the system designers decided to document them as different components may provide a soft indicator that

they felt there was a distinction. It may be useful to capture that initially and then simplify diagrams at a later point if separation does not add value.

(Question): What communication channels exist to connect these pieces of hardware and software?

- (Initial Assumption): The scanning ring and scanning base are connected via a cable.
- (Initial Assumption): The control station connects to the scanning base via a data cable.
- (Initial Assumption): The control station connects externally to the hospital network via either ethernet or Wi-Fi.
- (Initial Assumption): The scanning base has USB-C connectivity for maintenance.
- (Discussion): One thing to highlight is that the initial write-up did not discuss how that USB-C connection in the scanning base is used. Does a medical technician use a serial cable to hook up a maintenance laptop to the base via the USB-C connection? Is a USB-C external drive attached to the port? Listing out these interfaces can help highlight gaps in understanding of the system.
- (Discussion): Note that additional security controls of the HDO's network were not listed. No firewalls were specified. No intrusion detection systems were listed. Since those controls are largely outside of the SNAP system's purview, it usually helps to perform initial threat modeling activities assuming they are not present.
- (Discussion): Note also that not much information was provided about the data cable from the control station to the scanning base. Is it a USB cable? Is it a proprietary connection? Does it use ethernet? Does it require a proprietary connector installed to the control station? These are questions that may need clarification.

(Question): What external entities do these pieces of hardware and software need to communicate with?

- (Initial Assumption): The control station needs to communicate with an HDO's electronic EHR system using HL7. There may or may not be an intermediary system to help translate the HL7 data to a format the HDO's EHR can ingest.
- (Initial Assumption): The control station needs to communicate with an HDO's PACS.
- (Initial Assumption): A medical technician needs to communicate with the scanning base.
- (Initial Assumption): An operator needs to interface with the control station.
- (Initial Assumption): An operator needs to interface with the scanning base.
- (Initial Assumption): A patient needs to interact with the scanning base and scanning ring.
- (Discussion): Note, the initial writeup did not describe other ways the control station interacts with the HDO's network. For example, is it part of the HDO's Active Directory deployment? Does it rely on the HDO's domain name server (DNS)? Does it utilize the HDO's Network Time Protocol server? How does it get an Internet Protocol (IP) address–Dynamic Host Configuration Protocol (DHCP) or static assignment? Does it need internet access to receive patches from the manufacturer? Asking questions like this and digging into hidden dependencies that may not be captured in initial documentation can be very important in this stage of the threat modeling process.

- (Discussion): It is also helpful to identify how users of this system are expected to interact with it. Capturing these expected workflows can help to highlight gaps in documentation. For example, is there a panic button for patients to use to stop or exit the system? Note: attackers don't have to follow expected operation procedures. Investigation of the ways someone might use the system will be addressed at the "What can go wrong?" stage of threat modeling.

While listing out answers to questions like the ones above can be helpful, often creating a high-level DFD can structure the brainstorming discussion about how the system is designed. An initial DFD may resemble Figure 18.

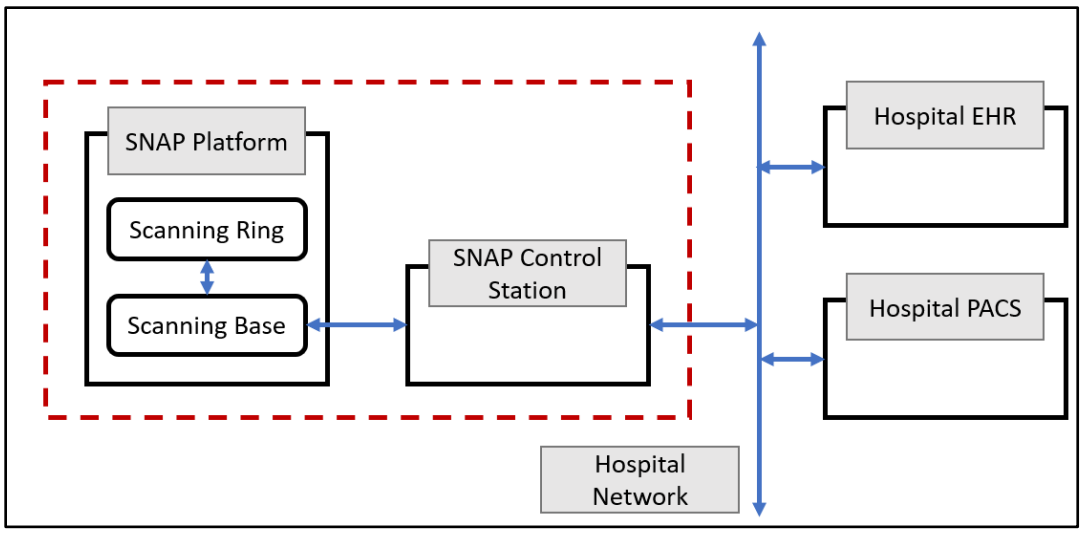


Figure 18: Example First Pass, High-Level DFD for the SNAP System. Only Major Hardware Components Are Outlined.

Despite its modest appearance, Figure 18 provides a solid base for discussion on how to start building a more complex model. Threat modeling is at its core an iterative process, and the iteration often arises from asking questions about what has already been created. Given the content, there exist several immediate locations to iterate further.

What illustrations are most effective to include to show how the SNAP hardware functions under the hood?

- What are some of the critical components that facilitate the hardware working at all?
- What ports, switches, or other input mechanisms are available on each piece of hardware? This can also include internal debug ports, like JTAG.
- Are there mechanisms responsible for actuating parts of the device? What controls them?
- What kind of ports connect each piece of hardware? What protocols do they speak? Could someone walk up and disconnect them, or connect them to other devices?

Do the trust boundaries as currently drawn make sense?

- Is the connection between the scanning ring and scanning base trustworthy?
- Is the connection between the scanning base and control station trustworthy?

What are the critical dataflows for this system? Where is data generated and where is that data eventually stored?

- Given that the control station can connect to the hospital's EHR or PACS, and those systems are out of our control, how deeply does the model need to diagram the internals of these systems?
- In the same vein, a DFD may not be the best tool for representing the flow of patient data from the SNAP platform to the hospital network. This dataflow is core to how SNAP functions, so it benefits from deeper scrutiny.

Based on the above questions and discussion, we can iterate on the current DFD, adding more details to arrive at something like Figure 19.

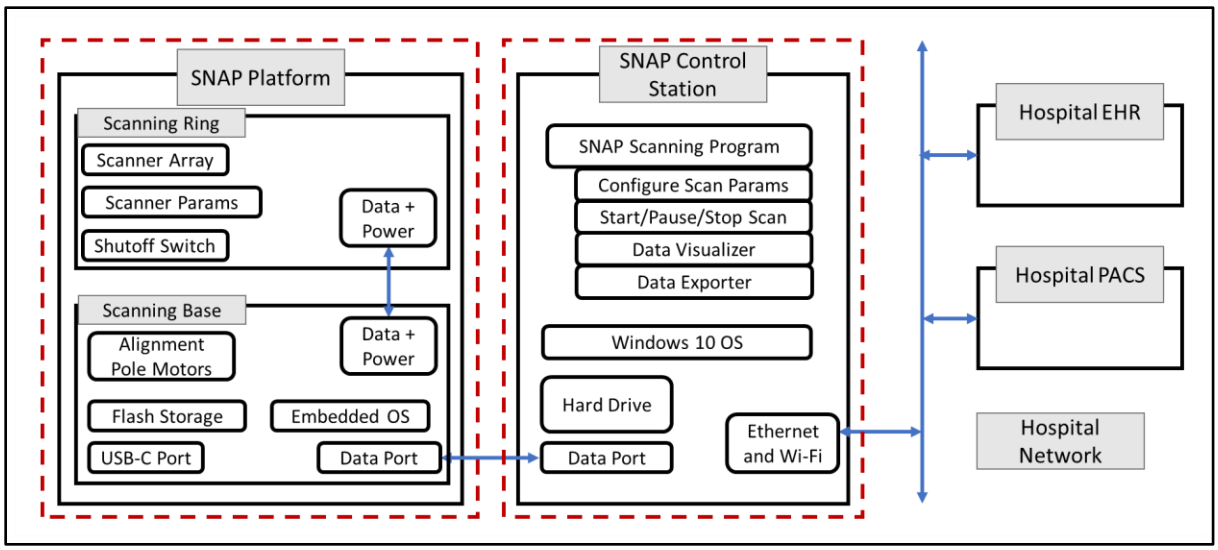


Figure 19: Second Pass at a High-Level DFD for the SNAP System

In comparison to the first DFD, there are a few noteworthy differences:

- Core pieces of internal hardware and software have been added to all three SNAP components. Core, in this case, covers exposed physical ports, storage mediums, operating systems, and connections to other hardware.
- An additional trust boundary has been added separating the control station from the scanning base. However, none has been added between the scanning ring and the scanning base.
    - Not adding a trust boundary between the scanning ring and the scanning base was a prioritization shortcut chosen to highlight the other trust boundaries first. That prioritization may need to be reassessed at a later point during evaluation of "what can go wrong. For example, an adversary may be able to access the scanning base but be unable to cause patient harm due to security controls protecting the scanner in the scanning ring from the Scanning Base. Wherever security controls exist, it implies that a potential need for a trust boundary differentiating what those controls protect.
    - Between the base and the control station, it is assumed that the ports at both ends of the connecting cable are exposed and relatively easy to disconnect. Because of this, the assumed risk is higher, so a trust boundary was added to highlight this.
- The OS for the control station has been specified, while the OS for the scanning base has not.
    - This is a definite gap that needs to be addressed. Even when a proprietary OS is used, capturing that can significantly aid in later threat modeling activities. This information will also aid in postmarket analysis when vulnerabilities impacting that OS are discovered.
    - Conversely, when making custom changes to an OS, be sure to document these changes and add them to the threat model. Vulnerabilities often affect specific versions or version ranges of libraries, and having this information documented can aid when considering threat mitigation or avoidance.

DFDs are a very useful brainstorming modeling technique and a great place to start, but it can be difficult to capture what workflows and dataflows look like only using DFDs. Once a DFD has been created to structure threat modeling discussions, a follow-on step can be to diagram several high-value dataflows and workflows. For example, Figure 20 highlights a potential workflow for a clinician to initiate a SNAP scan using a State Diagram. As a disclaimer, this workflow was not discussed in the initial write-up, so assume data for it was captured through follow-up discussions with the system developers.

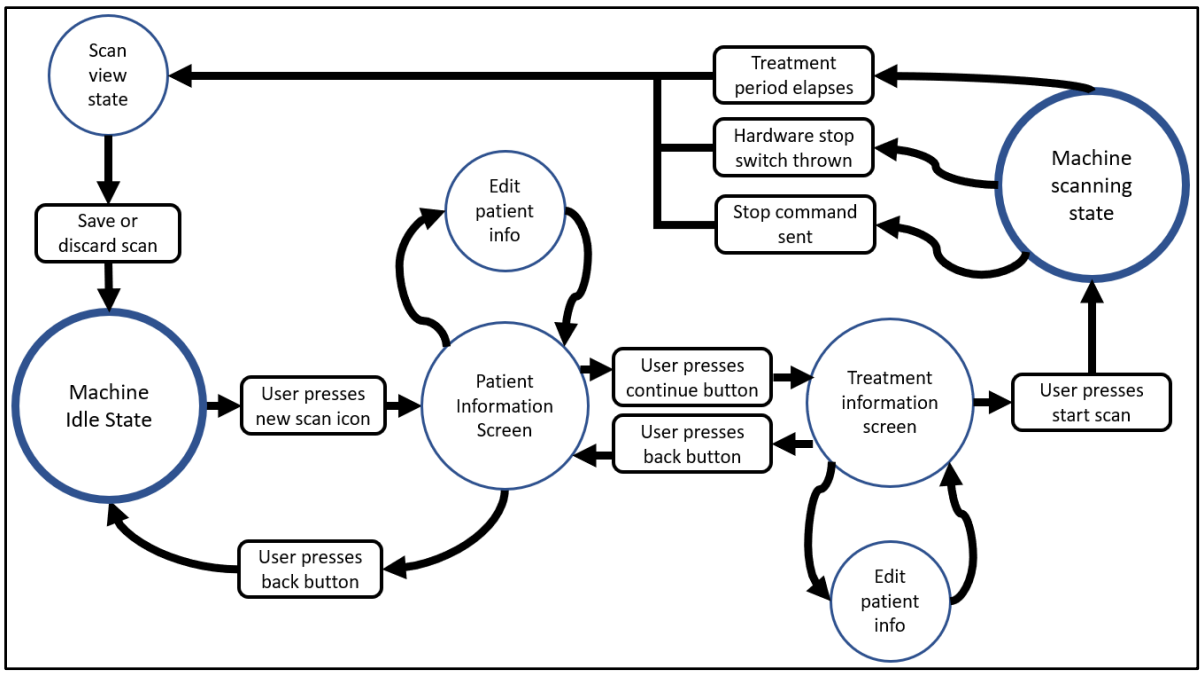


Figure 20: State Diagram for Initiating a SNAP Scan

State Diagrams can be valuable to diagram workflows for how a system is supposed to operate, as well as to help identify assumptions that might not hold up and need further investigation. For example, what happens if the power goes out between the “machine scanning state” and the “scan view” state? This can help structure the “what can go wrong?” activities of the threat modeling process.

Going back to the objectives of this section, we want to focus on the challenges that arise when threat modeling the communication between a system and an HDO’s network, as well as the communication between the Control Station and the Scanning Base. As stated earlier, one common way to do this is with Swim Lane diagrams, but unfortunately there are several common pitfalls that often occur when developing these diagrams. To highlight this, Figure 21 shows an intentionally flawed Swim Lane diagram describing the flow of initiating a scan and sending the results to an HDO EHR.

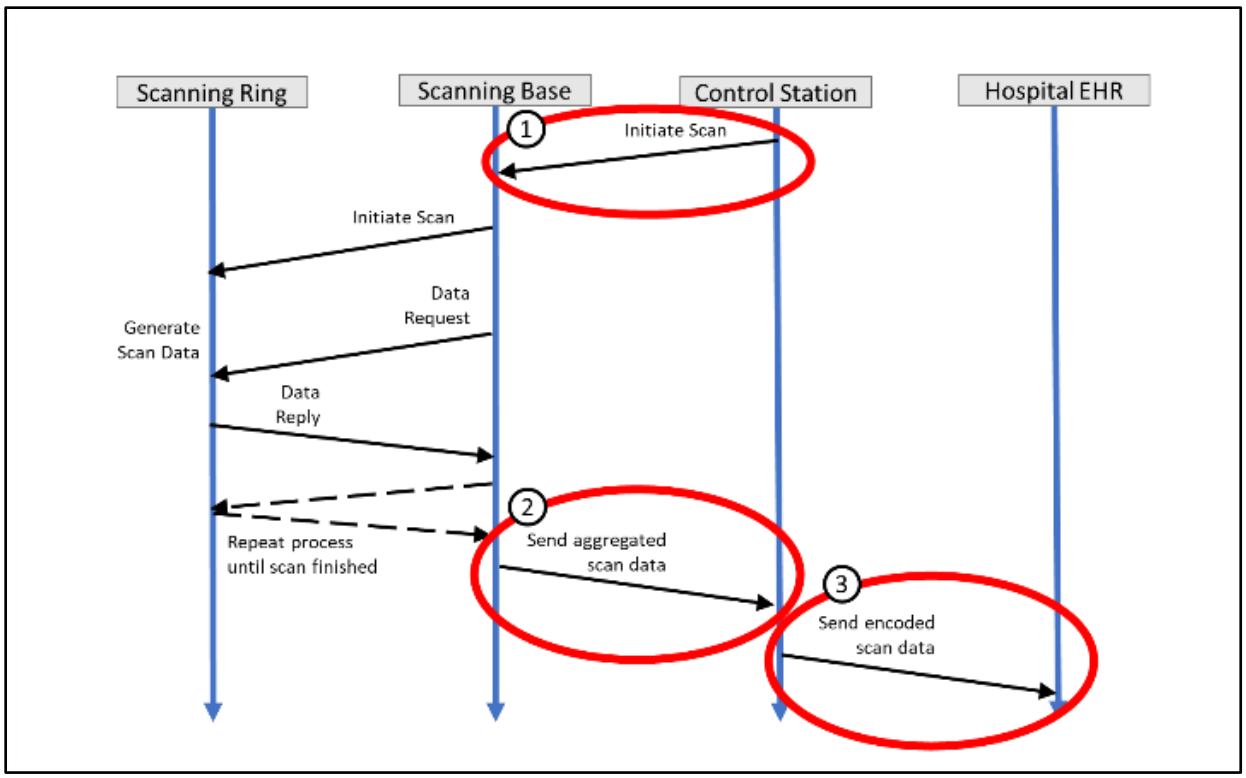


Figure 21: Intentionally Flawed Swim Lane Diagram Describing the Flow of Patient Data from SNAP to the HDO EHR

At first glance, Figure 21 might look sufficient to describe the communication flow, but there are significant gaps that can limit its usefulness when performing threat modeling. To highlight these gaps, below are some questions that might come up when threat modeling these interactions that the diagram does not address.

Control Station to Scanning Base communication (Highlighted Point #1):

- What protocol is being used to initiate the scan? What does the API look like?
- Is there any authentication of the command being sent from the Control Station to the SNAP Scanning Base?
- Does the Control Station perform any authentication to ensure it is talking to the Scanning Base?
- Is there any encryption of the data being sent between the Control Station and the Scanning Base?
- Is there an acknowledgement that the command was received properly?
- Is there an acknowledgement that the command can be successfully executed? For example, what if the SNAP system is in the middle of running a scan when a new command is sent?

Scanning Base to Control Station communication (Highlighted Point #2):

- Is there a listening process running on the Control Station to receive the aggregated results, or is this a continuation of the initial session started in (#1)?
- Is any sort of integrity check performed on the data that is sent back to the Control Station?
- What format is the aggregated results in? Is it a proprietary format? Is it a collection of JPEG or GIF images? Is it using DICOM message format? Is it JSON, and if so, what fields are there?
- What is the approximate amount of data being transferred? Is it 100Mbs? 100Gbs?
- How long is the data transfer expected to take?
- What happens if the data is malformed, or the communication halts unexpectedly?

Control Station to Hospital EHR communication (Highlighted Point #3):

- What is the transport protocol being used to send the data? Is it FTP, MLLP based on TCP/IP, UDP/IP, NETBEUI, or something else?
- What is the formatting of the data? The original SNAP write-up mentioned using HL7, but this diagram does not mention that at all.
- Are intermediary devices expected to collect the messages from the Control Station and translate them into a format that the HDO's EHR can ingest?
- Is there any authentication of the messages?
- What happens when there is a transmission failure?
- What does the diagram mean by "encoded" when mentioning encoded scan data?

Sometimes it may be sufficient to reference other design documents when answering questions such as those raised above. For example, references to API specifications in a Swim Lane diagram could be included. Also, multiple Swim Lane diagrams that describe different paths a protocol may take may be useful. If depiction of the decision logic using Swim Lane diagrams proves difficult, consider a related approach such as modeling the protocol as a Cross-Functional Swim Lane diagram instead. As an example of that, Figure 22 depicts a potential Cross-Functional Swim Lane diagram of the reply from the scanning base being sent to the Control Station, and from there to the HDO EHR's system. Note: This example makes assumptions that were not detailed in the initial write-up, and like Figure 21, contains intentional gaps and areas for improvement.

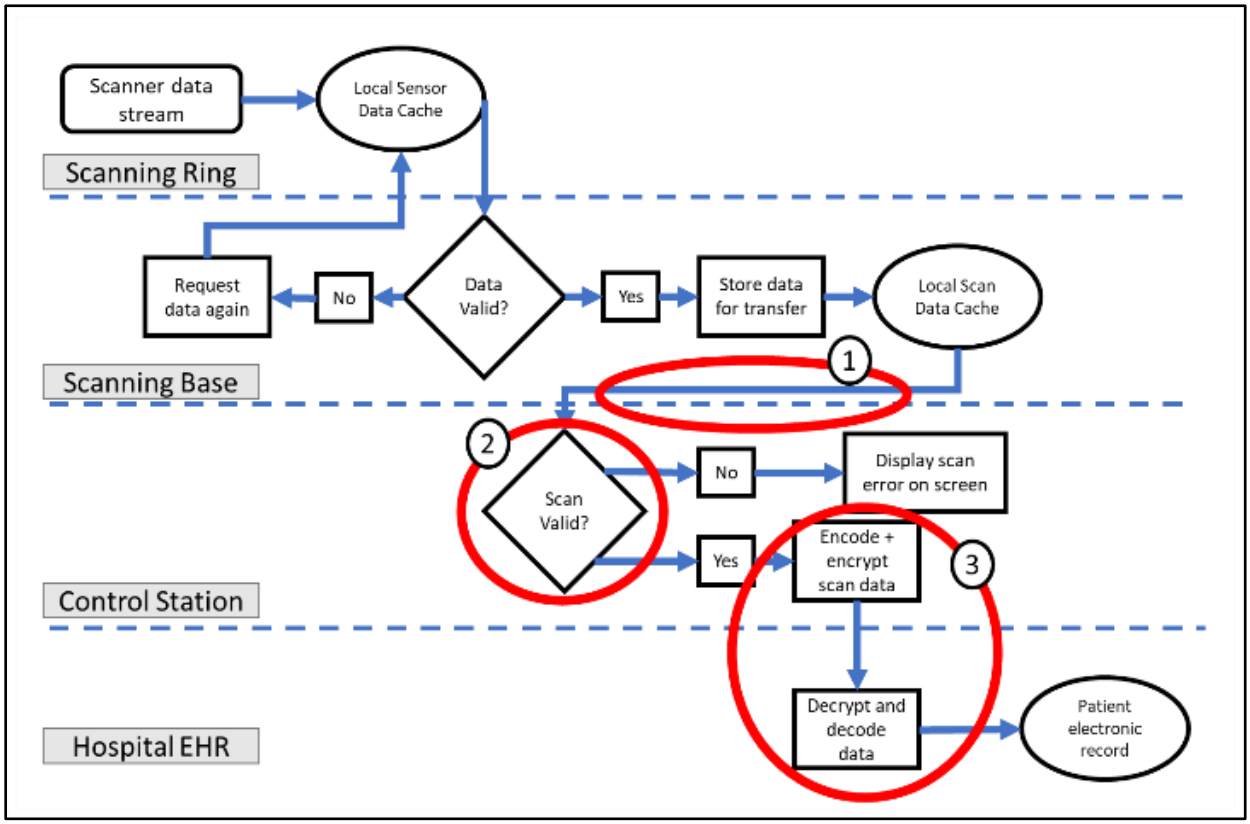


Figure 22: Intentionally Flawed Cross-Functional Swim Lane Diagram Describing the Flow of Patient Data from SNAP to the HDO EHR

Figure 22 starts to answer some of the questions asked previously, such as how the SNAP system handles malformed data, but it still has some serious shortcomings.

Scanning Base to Control Station communication (Highlighted Point #1):

- There still isn't a lot of detail about what the actual communication between the Scanning Base and the Control Station looks like. Yes, there is an arrow, so at least the fact that a communication path exists is documented, but more information about what this protocol looks like will help with the threat modeling process

Scanning Base to Control Station communication (Highlighted Point #2):

- It is worth highlighting that while a validity check is run on the scan results, there are no details about what that check entails. Is it a simple CRC32 checksum? Is it a signed cryptographic hash of the data? Understanding the mechanisms used to validate the data is important to the threat modeling process since different validation schemes vary significantly in the protection they provide against assorted threats. For example, a CRC32 checksum can be an integrity control, but it is not a security control against a malicious actor.

Control Station to Hospital EHR communication (Highlighted Point #3):

- Perhaps the biggest gap in this diagram is the same one that occurred in Figure 21. The actual protocol and encoding when sending the data to the Hospital EHR is not described in this model.
- This is further compounded by the words "encrypt" and "decrypt," which were not present in Figure 21. How is the data being encrypted? Is it a transport layer encryption such as Transport Layer Security (TLS), which is often referred to as "HTTPS" (as seen in the address bar of web browsers)? Is it a message encryption like that found in FHIR? Or is it something else? It is important to specify not only the encryption protocol but also the version and relevant features. For example, instead of just saying something is encrypted with TLS, you could specify TLS1.2 using the TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA cipher suite.

A common shortcoming when modeling a system is the use of terms such as "encrypting," "encoding," "checking," and "validating," without providing supporting documentation or detailed descriptions of what those steps are. This can occur in Data Flow Diagrams, Swim Lane Diagrams, and State Diagrams. As a general rule, when using these words when modeling a system, best practices encourage organizations to think about how to obtain, reference, and provide additional documentation about what those steps entail.

At this point, there is enough detail between the updated DFD and Swim Lane Diagrams to move on to considering what might go wrong with this example. However, this is a good point to step back and ask an important side question–what's missing? So far, the steps completed have been based off the initial feature overview describing SNAP. However, that write-up really is merely an overview. It does not discuss any of the finer details about how a user is intended to interact with the system. To name several items that are missing:

- What is the expected workflow for a technician to initiate a scan? How do they launch the software? What patient data needs to be input for a scan?
- How does a patient need to be situated in the scanner? Does the Scanning Ring default to the top or bottom position? What happens if the patient moves during a scan?
- How does the hardware shutoff switch behave? Can the patient access it during a scan? What happens when it is hit during a scan?
- How does the Scanning Ring behave while a scan is in progress? Can a scan be aborted by the technician?
- How is the Scanning Ring calibrated? Are there ways to know if the Scanning Ring is not properly calibrated?

The above highlights an important thought about modeling in general. There can occasionally be gaps between how a feature is described on paper and how that feature is used in practice. While capturing the

raw mechanics of a system is important, documentation of details like those above are important to capture to fully answer the question of “What are we working on?”

Likewise, sometimes it is helpful to take a step back, look at a high-value dataflow and ask, “What is actually happening here?” A lot of questions arose regarding Figure 19 and Figure 22, and as threat modeling processes continue, the answers may be helpful. What happens if an attacker sends malformed data? What happens if someone trips over the power cable in the middle of a scan? Where is sensitive patient data being sent and who can view it? Who has the keys to decrypt encrypted data? Understanding the answers to these questions requires understanding the system being threat modeled.

A.1.3. (SNAP) What Can Go Wrong?

When looking at what can go wrong, it is tempting to focus on the highest impact outcomes that may happen if a medical device starts to fail. Using an unstructured brainstorming approach, some of the following impacts for SNAP are easy to fixate upon:

- A patient could be afflicted with skin damage if treatment is administered improperly.
- Mechanical failure of the Scanning Platform or Ring could result in the creation of dangerous projectile debris or collapse of the scanner.
- Mishandling of patient data could lead to misdiagnosis or loss of privacy.

These are certainly events that need to be mitigated. The challenge is that human adversaries can have a wide variety of goals that may be hard to predict in advance. They might use access to this medical device as a jumping-off point to target other systems. For example, stored credentials may work on other systems, or firewall policies and network segmentation may make the device a tempting beachhead to target other systems. An attacker may simply be looking to cause a denial of service as part of an extortion attempt. They may just want to mine cryptocurrencies on the device or use it for ad-clicking malware revenue sources. Using a structured technique can help identify not only the high-impact threats, but also otherwise unidentified threats.

There are many ways to perform structured brainstorming, several of which have been discussed in this Playbook. This example begins with a STRIDE analysis. The output of a STRIDE analysis can then be used to define gaps and provide a basis for building out other threat identification techniques.

Rather than going through a full STRIDE analysis of the SNAP system, this example will focus on the communication path between the SNAP Scanning Base and the Control Station, and the Control Station and the HDO’s EHR. STRIDE threats inside the components will not be investigated, and the example will instead focus on where previously identified trust boundaries are crossed.

Table 10Table 10 shows the results of our initial STRIDE analysis. Below are details about how the STRIDE analysis was arranged and documented.

- Reference IDs are used to denote individual threats that have been identified in the top-level STRIDE analysis table. This allows detailed information about those threats to be captured elsewhere without cluttering up the STRIDE table. This example used numbers, but other naming schemes can certainly be used. For example, references to the component or the threat in the ID name to make it easier to understand quickly what it applies to. Instead of referencing a threat with the ID of “1”, it may be referenced as “SB-P1-S-1,” which might map to Scanning Base, Process 1, Spoof Threat, Number 1. Another option is to use a ticketing system to use the ticket ID associated with the threat as the reference in the STRIDE table.
- Often components may be listed, versus processes. An example is that the SNAP Scanning Base may be listed versus the process on the Scanning Base that receives traffic from the Control Station. Processes are useful simply because there may be many different vectors into a component. For example, the Scanning Ring also talks to the Scanning Base, and the Scanning Base contains other interfaces such as a USB-C port. It can be helpful to differentiate all of these

in a STRIDE analysis, and one way to do that is to highlight the processes that manage those communication channels.

- A frequent source of confusion about STRIDE is if a threat is most effectively categorized as a dataflow or a process. The ultimate answer is to do what makes sense for the needs of the organization performing the threat modeling. However, to make it easier for external readers to review STRIDE analyses, threats could be categorized against the bits flowing across the wire under the associated dataflow. Threats on how those bits are interpreted, processed, or received may also be categorized under the process. For example, if someone taps or cuts a cable, that is a threat against the dataflow. If someone sends invalid data that causes a process to crash, that is a threat against the process.

Table 10: STRIDE Analysis for Components of the SNAP System (Intentionally Incomplete)

SNAP Component	*Spoof*	*Tamper*	*Repudiate*	*Info*	*DoS*	*EoP*
Process: SNAP Scanning Base, Control Station communication process	1	2	17	18	3, 16	15
Process: SNAP Control Station, Scanning Base communication process	9	10	19	21	11, 20	12
Process: SNAP Control Station, EHR communication process	4, 22	5	6		7	8
External Entity: EHR	23, 24					
Dataflow: Control Station to EHR		13		14		
Dataflow: Control Station to Scanning Base		10		11		

Table 11: STRIDE Threat Information: SNAP Scanning Base, Control Station Communication Process

Reference ID	**STRIDE TYPE**	**Description**
1	Spoof	An attacker may use credentials gained from the control station to initiate an unapproved scan.
2	Tamper	The listening process' software could be modified by an attacker to behave differently.
3	DoS	An attacker could send malformed data and crash the process.
15	EoP	An attacker could send malformed data that would allow them to gain unapproved privileges (e.g., a buffer overflow attack could allow them to execute their own code on the Scanning Base).
16	DoS	An attacker could continuously send "Scan reset" or "Scan stop" commands, preventing any scans from being completed.

17	Repudiate	If a scan goes wrong and injures a patient, there may be confusion and/or disagreement about whether that scan was initiated correctly by the clinician. There may be a lack of logs in the scanning base to solve this question, or the logs could be tampered with.
18	Info	The process may leak information about past scans when it is not supposed to.

Table 12: STRIDE Threat Information: SNAP Control Station, Scanning Base Communication Process

Reference ID	STRIDE TYPE	Description
9	Spoof	An attacker could spoof scan data and upload it to a patient's profile.
10	Tamper	Process' software could be modified by an attacker to behave differently.
11	DoS	An attacker could send junk scan data packets to crash the process.
12	EoP	An attacker could send crafted data that would allow them to gain unapproved privileges (e.g., a buffer overflow attack could allow them to execute their own code on the Control Station).
19	Repudiate	If a scan goes wrong and injures a patient, there may be confusion and/or disagreement about whether that scan was initiated correctly by the clinician. There may be a lack of logs in the scanning base to solve this question, or the logs could be tampered with.
20	DoS	An attacker could send malformed data or a message that the scan was canceled to trick the Control Station into ignoring the actual scan results.
21	Info	The control station process may leak data that might not be useful or appropriate for the medical device communication link. For example, it might leak its public IP address, the user's Microsoft Active Directory account info or password, etc.

Table 13: STRIDE Threat Information: SNAP Control Station, EHR Communication Process

Reference ID	STRIDE TYPE	Description
4	Spoof	A malicious server pretends to be the EHR and requests patient data or pretends to be the legitimate endpoint when the process sends the patient data.

5	Tamper	The process's software could be modified by an attacker to behave differently. For example, a different application could be installed on the workstation.
6	Repudiate	If a scan goes wrong and injures a patient, there may be confusion and/or disagreement if that scan was initiated correctly by the clinician. There may be a lack of logs in the scanning base to solve this question, or the logs could be tampered with.
7	DoS	An attacker could send malformed data and crash the process.
8	EoP	An attacker could send specially crafted data that would allow them to gain unapproved privileges (e.g., a buffer overflow attack could allow them to execute their own code on the Control Station).
22	Spoofing	An attacker could attempt to log into the Control Station using stolen or guessed administrative credentials.

Table 14: STRIDE Threat Information: External Entity EHR

Reference ID	STRIDE TYPE	Description
24	Spoof	An attacker could create HL7 messages of fake patient scan results to the EHR.
23	Spoof	An attacker could send messages to the EHR that the scans were invalid.

Table 15: STRIDE Threat Information: Dataflow, Control Station to EHR

Reference ID	STRIDE TYPE	Description
13	Tamper	A person-in-the-middle attack could happen on the network, and an attacker could intercept and modify HL7 messages going to the EHR.
14	Info	If encryption is not used, someone listening in on network traffic could view patient scan results.

Table 16: STRIDE Threat Information: Dataflow, Control Station to Scanning Base

Reference ID	STRIDE TYPE	Description
10	Tamper	A malicious inline implant could modify the commands or the scan results. Note: This is a low likelihood event (someone installing a

		malicious tap on the wire), but capturing things like this can still be useful for the threat modeling process.
11	Info	If the data is not encrypted, a malicious process on the control station or a tap on the wire could steal patient scan results. For example, while the Control Station process may not be tampered with, another piece of software could be running on the Control Station listening to network traffic.

To reiterate, *the above does not constitute a full STRIDE analysis*, as several components are left out of consideration (e.g., the Scanning Ring, dataflow between the ring and base, the Control Station software). However, what is there can be used as a first pass to begin discussing some important topics in more detail.

When brainstorming individual threats themselves, here are some considerations based on the above STRIDE analysis:

- There often is some ambiguity on where to categorize individual threats. For example, if an attacker guesses a password, is that a spoofing threat or an escalation of privileges threat? As a general rule, if an attacker is masquerading as something they are not, that is spoofing. If they are leveraging access to gain more access, that is an EoP threat. The categorization is ultimately less critical than ensuring that the threat is captured in some form.
- STRIDE per Element (described in Table 3 in Section 2.4.1) can be a good way to identify gaps in STRIDE analyses. Using it to review the example STRIDE analysis, it implies there may be an unidentified Information Disclosure threat for the SNAP Control Station, EHR communication process.
- One thing that may stand out is that only identified spoofing threats against the external EHR system were identified. Many additional threats likely exist. However, since it is an external entity that is largely out of the control of the organization performing the threat modeling, threats that impact how a system interacts with it are the most critical to identify. In this case an attacker could pretend to be our system to trick the EHR, hence why it is a Spoofing threat.
- Note at this stage that many of the threats are shared across processes/dataflows, and there is little information about how they would be exploited. That is normal, and generally representative of an initial STRIDE analysis. Later, additional detail may be added that will start to answer the "how" question of how a particular threat might be leveraged by an attacker. Other threat modeling techniques such as Attack Trees and Kill Chains can help to provide that detail.
- Reference to previous models and DFDs to get more information about threats may be useful. For example, Threat ID #12, which is a EoP threat against the SNAP Control Station process that communicates with the Scanning Base, is extremely vague in that it only mentions "malformed data." Examining the Cross Functional Swim Lane diagram in Figure 19, there is a validation step that is run on the data. Perhaps the attacker creates malformed data to specifically target that process, so a flaw in the validation logic might lead to an Escalation of Privilege.

STRIDE analysis is a quick way to brainstorm about possible threats. But often organizations will need to create much more detailed information about how those threats may be carried out by an attacker. This is particularly important when moving on to the next stage of the threat modeling process, in which organizations start trying to decide "What are we going to do about it? One approach is to try to build Attack Trees that leverage the threats identified in STRIDE analyses. Figure 23 is one such Attack Tree, with several of the threat IDs from the STRIDE analysis highlighted on it.

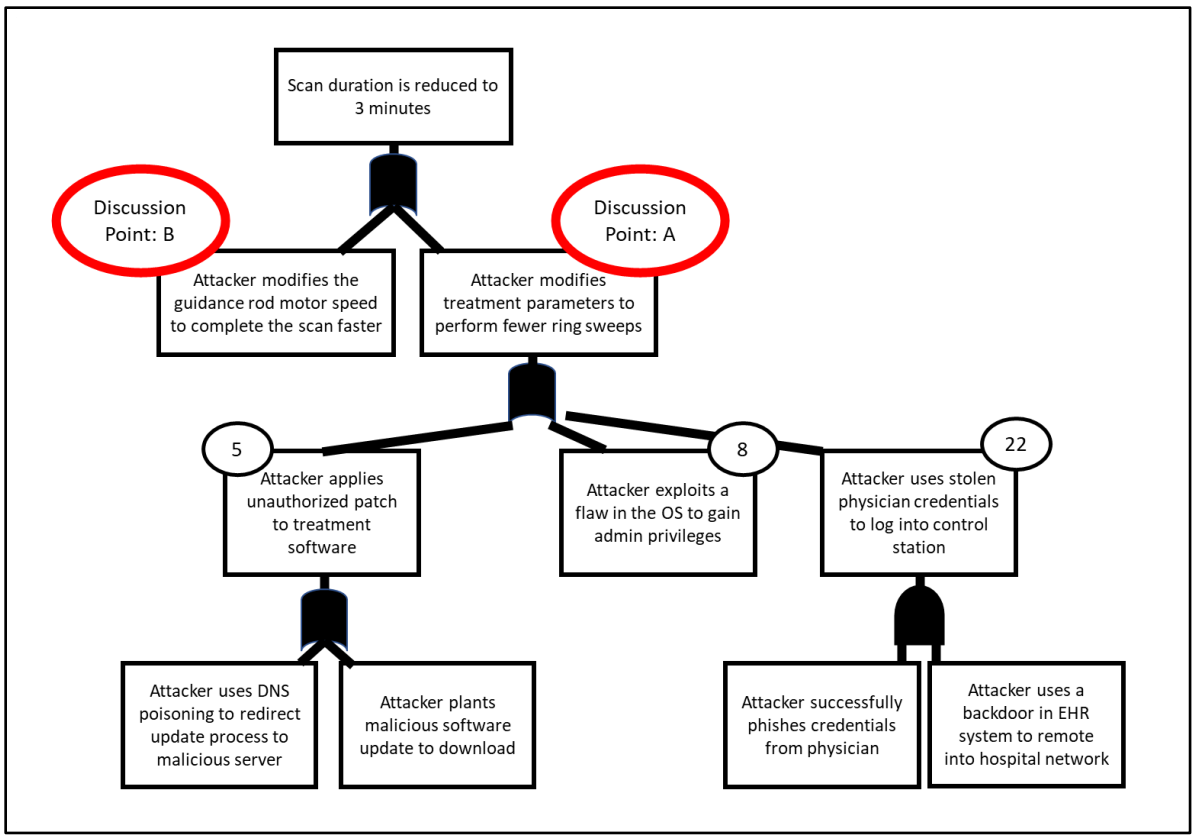


Figure 23: Example Attack Tree Diagram Reducing the Scan Duration to Corrupt the Results

For this example, a top-down approach is used, starting with the targeted negative impact and working backward to determine how an attacker could accomplish those goals. A bottom-up approach can also work, such that it begins from the point of initial compromise and works from there to model what an attacker might do. Which approach is better is largely driven by organizational needs and desires. If the focus is on threats, and examining what can happen if a threat is exploited, a bottom-up approach works best since the end-goal of potential attackers are likely unknown. A top-down approach works better where an end state is known, and the goal is to identify the paths that could lead to it. For example, Figure 21 was constructed with a focus on brainstorming patient safety risks. Changing the scan duration was something that could have a serious impact, so a top-down Attack Tree was created to try and identify paths that would allow an attacker to achieve that.

Looking at the diagram, there are several observations that can be made:

- Threats between the Control Station and the Scanning Base have been highlighted in this Attack Tree. Were STRIDE analyses performed on other sections of the SNAP system, other boxes in the Attack Tree would also likely be highlighted. This is a good example of how different threat identification techniques such as STRIDE and Attack Trees can complement each other. If there is a box in an Attack Tree that does not have a STRIDE threat associated with it, that points to a gap in the STRIDE analysis, which can be remedied. Likewise, if there exists a STRIDE threat that does not show up in an Attack Tree, that may imply there are other paths in Attack Trees that have not been identified yet.
- **Discussion Point A**: A valid question might be if the protocol between the Control Station and the Scanning Base allows for modification of treatment parameters. The answer may not always be obvious, but the simple fact that the question arose can then prompt consultation of the appropriate references to find out. The fact that there is ambiguity to the answer highlights that

there may be different elimination and mitigation techniques when moving on to identifying "What are we going to do about it? For example, this path could be eliminated by hardcoding the scan duration into the scanning base and not making that part of the protocol when communicating with the Control Station. This also highlights the value of performing threat modeling early in the development process, where impacts of different design decisions can be shown, and they can still influence the development of the system.

- **Discussion Point B**: Often there may be stubs in Attack Trees where there exists a general idea of a threat, but the full details have not been fully developed. It remains helpful to document these ideas.
- While it is important to document failures of the system that require immediate attention, other types of failure modes do exist and are important to consider. The targeted failure state for the above example isn't readily apparent as a failure. That is to say, the system doesn't fail in a way that makes the failure immediately known. In contrast, instead of reducing the scan duration, the attacker could prevent scanning of the patient altogether. These two outcomes share much in common, but the result has some differing nuance. In the case of prevented scans, the machine is left immediately unusable, but the machine is also known to be compromised. On the other hand, when the scan duration is reduced, the attacker is introducing inaccuracy into the medical process, which could have impacts during diagnosis.

A.1.4. (SNAP) What Are We Going to Do About It?

For each individual threat, options may be identified to reduce or eliminate the risk, and then the option(s) that will provide the strongest protection selected. Once the options have been selected, the relevant DFDs or other diagrams will need to be updated, along with other associated documentation. This section is focused on the analytical process and will not include additional updated DFDs.

For the purposes of this SNAP example, some mitigations will be identified for the threats to the Control Station endpoint, as covered in Table 11, Table 12, and Table 13.

Note: this list of potential mitigations is for demonstration purposes only; it is intentionally incomplete. Accordingly, an organization's mitigation analysis would be more comprehensive. Organizations may decide how to record the results of threat modeling analyses, especially for rejected mitigations. It is important to balance the amount of effort necessary to track these decisions and the potential benefit of having them easily available for future use.

A.1.4.1. Mitigation Options for Spoofing (ref IDs 4 and 9)

ID 4 (from Table 13): A malicious server pretends to be the EHR and requests patient data, or it pretends to be the legitimate endpoint when the process sends the patient data.

- *Eliminate:* Implement strong authentication of the EHR server (e.g., using certificates with strongest-available cryptography to prevent sniffing of credentials).
- *Mitigate:*
    1. Require password-based authentication from the connecting EHR server
    2. Require the EHR to use a cryptographic signature over Mutual Transport Layer Security to authenticate itself
    3. Require the EHR to use Secure Shell tunnelling to authenticate itself
    4. Contact top EHR vendors for other supported fixes
- *Accept:* Assume that any connection is authorized. Since the risk of Protected Health Information (PHI) loss is unlikely to be acceptable to the HDO, it does not seem feasible to accept this risk without some mitigations in place.
- *Transfer:* Provide a capability for the HDO to specify IP addresses for trusted EHR servers.

One could argue that the use of strong authentication for the EHR server still does not "eliminate" the risk, since there is always the possibility of an attacker compromising the EHR server itself and stealing the private certificates. The primary option to eliminate the risk would be to remove the capability of communications with the EHR server entirely, but this would limit the usefulness of SNAP within clinical environments where connectivity with EHR systems is often required. This could be considered as a strong mitigation instead.

Note that each mitigation can come under threat if it is not implemented correctly. For example, if hard-coded certificates or passwords are used, then any attacker who has such a certificate could launch attacks against all systems using this capability.

As specified, the available mitigation options are not necessarily ideal, especially if communications with third-party product that do not offer the desired capabilities must be supported. If a focused analysis does not find an ideal option, this could be captured as a future requirement for the device and/or other vendors to address.

ID 9 (from Table 12) An attacker could spoof scan data and upload it to a patient's profile.

- *Eliminate*: Require strong authentication of the Scanning Base to the Control Station (e.g., with unique certificates).
- *Mitigate*:
    - Block or prevent all extraneous or unused physical input/output (I/O) ports.
    - Require password-based authentication to the Control Station.
    - Even better, require biometric or smartcard-based authentication to the Control Station that requires minimal interaction by busy clinical staff.
- *Accept*: Assume that any device that connects to the physical data port is trusted. Make sure to document this assumption. However, accepting this risk outright is highly discouraged, as MDMs are increasingly regarding the hospital environment as a hostile environment.
- *Transfer*: Require that the HDO ensure physical security of the room or area in which the Control Station is located.

Notice that clarifying the SNAP architecture might be needed to appropriately define and select the best mitigations. For example, Figure 19 only identifies "data ports" between the base and the Control Station. What kinds of data ports are they? What protection mechanisms or physical access restrictions are built into those ports? If the specific components of the design/architecture of the product have already been selected, then some mitigations might not be feasible. On the other hand, if specific components have not been selected, then the potential mitigations that may be identified could influence requirements for such components.

Hybrid solutions might also be appropriate. For example, risks could be mitigated by requiring (some) form of authentication, while also transferring some of the risk to the HDO by requiring physical security of the room.

A.1.4.2. Mitigation Options for Tampering (ref ID 5)

ID 5 (from Table 13) The process' software could be modified by an attacker to behave differently.

- *Eliminate*: The Control Station software itself cannot be eliminated since it provides core functionality for SNAP. The risk itself cannot be reliably eliminated either, since best practices encourage organizations to presume that all software, including the underlying operating system, has vulnerabilities. However, one could remove the ability to control the Scanning Base from the Control Station (i.e., move key abilities into the Scanning Platform).

- *Mitigate*:
    - Integrity checking of the software during operation
    - Code-signing upon installation and execution (e.g., the OS and drivers for the scanning base)
    - Running the software with limited privileges
    - Performing rootkit detection and other intrusion detection techniques to identify when untrusted code is executing
- *Accept*: Assuming full trust in the software might not be appropriate, as the Control Station might be operating in an environment that cannot be fully controlled.
- *Transfer*:
    - Require the HDO to perform system administration to ensure that the control station is not modified.
    - Require application control (aka “whitelisting”).
    - Require anti-virus.
    - Require regular patching.
    - Require regular audits of the system.
    - Limit physical access to the control station, including all potential I/O ports such as USB.

Note that some of the options to “Transfer” risk could be implemented as mitigations and vice versa. For example, if patches for the device are automatically enforced by an update server, that may be considered “Mitigate.” However, if patches are made available and it is up to the technician to manually install them, that would be considered closer to “Transfer.”

A.1.4.3. Mitigation Options for Denial of Service (ref ID 11)

ID 11 (from Table 12) An attacker could send junk scan data packets to crash the process.

- *Eliminate*: This threat cannot be eliminated, since by design, data packets are sent from somewhere.
- *Mitigate*:
    - Isolate the networking component from the scanning capability in a way that allows the system to perform scans even under heavy traffic load.
    - Implement firewalling and other network isolation so that only authorized/intended systems can perform a flooding attack.
    - Implement rate limiting for networked requests.
    - Use a watchdog-style process to restart the device upon a crash.
    - Use trusted or reliable networking libraries that reject “junk” packets.
    - Closely tie scan results with scan initiations to prevent replay attacks.
    - Use encryption and strong authentication to limit who can spoof results.
- *Accept*: It is not advisable to accept this risk, especially when many potential mitigations exist
- *Transfer*:
    - Require the HDO to perform rate limiting on the network to which the Control Station is connected.
    - Require the HDO to use firewalls to limit which systems can connect to the Control Station

Note that the mitigate/transfer options have similar solutions. If the Control Station software runs on a custom, proprietary operating system that is only intended to be controlled by the manufacturer, then it

may be necessary to expect the responsibility of implementing the capabilities to restrict network access and/or reject invalid packets.

Finally, consider what "junk scan data packets" is intended to mean, as the available options might differ. If packets are malformed in a way that violates specification, then this would be easier to detect and shut down than a series of seemingly legitimate, well-formed packets that would require deeper, more computationally expensive analysis that could slow throughput.

A.1.4.4. Options for Escalation of Privilege (ref ID 8)

ID 8 (from Table 13) An attacker could send malformed data that would allow them to gain unapproved privileges (e.g., a buffer overflow attack could allow them to execute their own code on the Control Station).

- *Eliminate*: There are no real options to eliminate this threat without removing the data-transfer capability.
- *Mitigate*:
    - Ensure that all code is tested for potential security vulnerabilities.
    - Compile/build the code with hardening features such as ASLR, Control Flow Guard, and other features that mitigate the damage from buffer overflows.
    - Isolate the networking component from the scanning capability in a way that allows the system to perform scans even under heavy traffic load.
    - Implement firewalling and other network isolation so that only authorized/intended systems can send malformed data.
    - Implement rate limiting for networked requests.
    - Use trusted or reliable networking libraries that reject "junk" packets.
- *Accept*: Since remote code execution is one of the most significant risks from an IT security perspective, and this device is operating in an IT environment, it is difficult to argue for accepting this risk without attempting to mitigate it.
- *Transfer*: Ensure that HDO operators appropriately limit network exposure

A review of this threat suggests that there might be a gap or simplifying assumption being made within the analysis—namely that "remote code execution" is involved with "network packets." Several questions arise that might require further refining the analysis: (1) Where is the junk scan data coming from? Presumably the Scanning Platform, but that is not stated; and the STRIDE table does not list a process or data flow between the platform and Control Station. Are there other mechanisms or interfaces that provide this scan data? (2) Where are the network packets coming from? Are they only guaranteed to come "from" the EHR system, or possibly other connected systems within the hospital network? (3) This emphasizes a particular attack vector and technique (buffer overflow) – is there a way to express it in a way that is independent of (or ignores) the specific technique?

A.1.4.5. Example Option Analysis from Attack Tree

While the previous mitigation examples used the STRIDE tables, similar analysis can be conducted on the findings from Attack Trees.

Consider the Attack Tree in Figure 23, concentrating on how the "Scan duration is reduced to 3 minutes." Nodes in the Attack Tree consider installing malicious software, exploiting an OS flaw to gain admin privileges, or obtaining a physician's credentials. Since the first two attacks already have equivalent mitigation analyses in previous sub-sections, the analysis could concentrate on cases in which the physician's credentials are stolen. This is effectively a trusted user, so the options might be more limited. For this example, the focus will be on the setting of the parameters for the scan duration.

- *Eliminate*: The parameters could be enforced by hard-coding them within software or hardware, instead of allowing changes by the operator through an interface. However, this could significantly limit flexibility of clinical operations.
- *Mitigate*:
    - Logging mechanisms and/or operator alerts or could be implemented to detect and report when parameters are outside of the recommended window.
    - Key parameters could be limited to be changed by an administrator or maintenance-only mode that requires strong authentication.
    - Require multi-factor authentication for each critical change.
    - The workflow could be modified to ensure that the operator must manually confirm such critical changes.
- *Accept*: Since scanning parameters can affect patient safety, close review would be needed to determine whether this risk can be accepted without some kinds of mitigations.
- *Transfer*: Provide clear documentation, including training, to the Control Station administrators and operators, to warn about setting parameters beyond the window and to respond to any alerts that arise.

Since the Attack Tree has highlighted situations in which the Control Station code itself might not be trustable, it would not be ideal to rely on the Control Station user interface to implement the workflow change because it might be subverted by an attacker-controlled OS. Even multi-factor authentication would not be a perfect solution if the Control Station is attacker-controlled since the attacker might read the operator's credentials from the keyboard as they are entered into the system. For example, an attacker may launch their attack during a time when the operator is expected to enter the credentials anyway, thus reducing suspicion that an attack is underway. A more radical decision could be made to modify the Scanning Platform by introducing a display that shows some minimal collection of critical parameters that are being set, and a physical button could be used to confirm any changes to the settings. This change has the benefit of further separating the Control System from the Scanning Platform, but it's best considered very early in the process before design is complete—further demonstrating how threat modeling can be extremely beneficial early in development.

A.1.5. (SNAP) Did We Do a Good Job?

At the end of the modeling process, it is tempting to conclude that the process has been effectively completed and move on. But what does it really mean to do a good job? Is there an indicator that dictates the difference between a good job and a bad job?

Based solely on the content of this document, it is certainly possible to argue that threat modeling activities completed for SNAP are ineffective by virtue of it being incomplete. Additional diagrams could be created to illustrate how various components or dataflows operate, such as detailed DFDs for the Control Station and Scanning Base. Several components were excluded from the STRIDE analysis and would need to be filled out. Only a portion of the threats were analyzed for potential mitigations, and many threats had incomplete sets of mitigations provided. In a real-world setting, the mitigations would have specific technical solutions (such as protocol names and version numbers).

As an exercise, consider evaluating the SNAP example using the checklist and other parts of Section 2.6, Question 4: Did We Do a Good Job?

A.2. Fictional Example 3: Stroke Kinematic Ankle and Toe Exerciser (SKATE)

This is the feature overview for the "Stroke Kinetic Ankle and Toe Exerciser" system. This represents a hospital clinical workflow with a fictional new patient treatment device. While this scenario does outline a specific clinical treatment methodology, the type of treatment was chosen to be distinct from real-world stroke treatment options. At a high level, the scenario was constructed to highlight typical features encountered when threat modeling medical devices. The following example is intended as an exercise, where the documentation below serves as a starting point for creating a system diagram for use in threat analysis.

The Stroke Kinetic Ankle and Toe Exerciser:

The Stroke Kinetic Ankle and Toe Exerciser (SKATE) is a therapeutic medical device, designed for use by patients who are recovering from a stroke in a healthcare delivery environment. At a high level, the SKATE system provides a boot-shaped apparatus that the patient's foot can insert into, where linear motors will either automatically flex the toes, or provide resistance against a manually flexing toe. Treatment parameters are configured and initiated by the doctor using an attached Control Board (i.e., connected remote). Bend and resistance sensors detect how well the patient is responding to treatment, and treatment data can be exported to the hospital EHR as necessary for further analysis.

- Period of expected use: Twenty-minute sessions, as prescribed
- Medical capability: Therapeutic
- Device invasiveness: Medium (patient will not be able to walk while using the device)

SKATE Core Use Case:

After conducting a scan using the SNAP system, Alice's primary care physician reviewed the resulting model. Based on this data, they have determined Alice has suffered a minor stroke and has localized nerve damage in her left leg. Alice is admitted to the local hospital for treatment and further monitoring of her health. In addition to other recommended recovery therapy, Alice's doctor has instructed her to perform a SKATE regimen for 20 minutes twice daily. Her assigned nurse comes in to initialize the treatment parameters before letting the device work. After three days in the hospital using the automatic setting, Alice is sent home with a reference to a local physical therapist, and instructions to continue her SKATE regimen using the manual resistance setting for another two to three weeks.

SKATE Core Technology:

- The SKATE Foot Apparatus, which places the patient's ankle and toe in a protective enclosure for controlled toe therapy
- The SKATE Control Board, which allows for programming of treatment parameters, starting and stopping treatment, and engaging maintenance mode
- The SKATE Backend Server, which serves as a fleet management point for deploying new devices, collecting log data, and pushing out device updates

SKATE Concept Diagram:

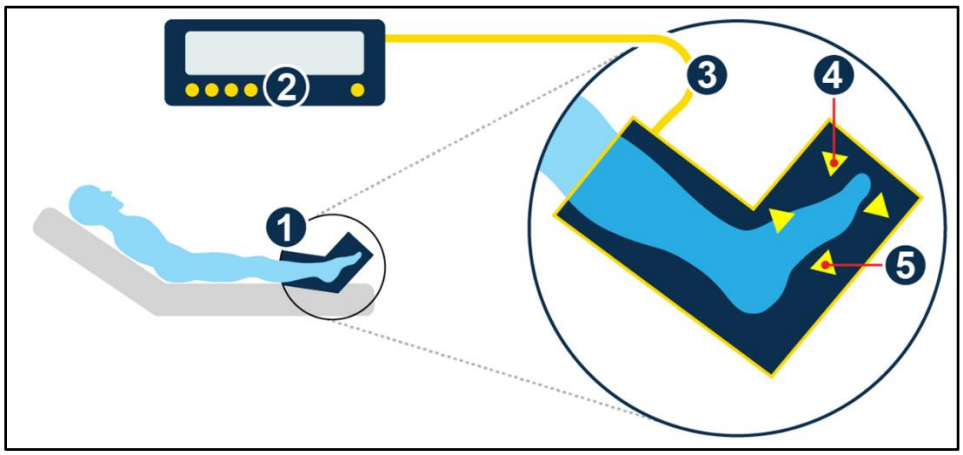


Figure 24: Picture of the SKATE device. The large circle provides a zoomed in version of the SKATE foot apparatus. 1) The SKATE foot apparatus. 2) The SKATE Control Board. 3) Connection between the foot apparatus and control board. 4) Motor and bend sensors for actuating the toe. 5) Motor and bend sensors for actuating the ankle.

SKATE Foot Apparatus:

The Foot Apparatus contains a small motor coupled with a set of actuators and bend sensors. Two modes of therapy can be used—automatic and manual. In automatic mode, therapy parameters are programmed into the SKATE Control Board to gradually flex and relax the toes. In manual mode, the motor and actuators provide resistance to the patient as they flex and relax their toes.

- Weight: ~4.5kg
- Size: Roughly the size of a large boot
- Power source: Onboard lithium battery with USB-C charging
- Motor and actuators for bending limbs or providing resistance
- Bend sensors to detect bend angles
- Real-time OS for managing hardware components
- Onboard flash storage for the firmware and temporary data
- Serial port for connecting to the SKATE Control Board
- Grayscale display and soft keys for viewing treatment information if the Control Board is disconnected
- Hardware switch for entering maintenance mode and applying firmware updates

SKATE Control Board:

The SKATE Control Board is a tablet-like device for managing the SKATE Foot Apparatus. It connects to the SKATE system via a physical serial connection and contains Wi-Fi and ethernet ports for connecting it to the HDO's network so it can interface with the SKATE server. The Control Board has three views that can be presented on the main screen. The default is a patient view, which provides basic information about the treatment plan, such as the time remaining in the current treatment session and a limited set of controls to speed up or slow down the treatment. Those controls are bounded by a set of guardrail values specified in the treatment plan to prevent the patient from selecting an unsafe setting. There is also a "stop" button that is always visible. A second mode is a clinician view, which allows specifying and modifying treatment plans. These plans can be downloaded from the SKATE server to avoid having to enter them manually. To enter the clinician view requires entry of a six-digit passcode. The final mode presents a technician view, which allows performing diagnostics on the SKATE Control Board and attached Foot Apparatus. It also allows specifying the network settings and updating the software and firmware on both the Control Board and the Foot Apparatus. To enter the technician mode requires

selecting a hidden icon in two corners of the screen at once, and then entering in a different six-digit passcode when a prompt appears.

Some basic specifications for the device are as follows:

- Weight: 0.4kg
- Size: 20cm x 10cm
- Power source: Internal lithium battery, charged via connection to the Foot Apparatus
- Serial port connection for connecting to the Foot Apparatus
- Secure Digital (SD) card slot for loading firmware or pre-planned treatments
- Embedded Linux OS
- Flash storage for OS and storing data files
- Ethernet port and built-in Wi-Fi chip

SKATE Server Software:

For environments managing many SKATE devices, the SKATE Backend Server software is also available for use. While a SKATE device can be used in stand-alone mode, the software provides useful functions for deploying and managing device fleets. It can be installed on a dedicated server or alongside other software suites as needed. It provides the following functionality:

- Interact with proprietary management software via a web interface.
- Register and initialize new SKATE devices.
    - Registered SKATE devices will report their status to the server every 15 minutes.
- Check device firmware status and version.
- Make firmware updates available for download to registered devices.
- Retrieve OS-level device logs.
- Retrieve patient treatment logs.
- Connect to an EHR system and upload treatment data.
- Push treatment parameters to a SKATE device.
    - Treatment cannot be initialized remotely.

Patient Safety Considerations:

Note: this is not an exhaustive list of potential harms.

- Improper configuration of the ankle or toe exercising parameters could result in applying more force than desired to the patient's limbs, potentially causing additional joint trauma.
- The range of injury could vary from strain to fracture of the joint, requiring additional hospitalization.

If functionality described above is not documented, feel free to record assumptions about how the device works when working through these examples.

A.2.1. (SKATE) Objectives of Discussion

The SKATE example was designed to share features with many medical devices that can provide interesting challenges when performing threat modeling. For example, it has an interface that patients, clinicians, and medical technicians all interact with. It also has a centralized server residing on the HDO's network that interacts with multiple SKATE systems. In addition, the SKATE server is used to manage and transmit treatment plans for patients.

Much of the threat modeling activity around this system will be left as an exercise to the reader. This section will instead focus on one threat vector: the software update process. There are many ways to update software, and there are many ways for that software update process to go wrong. The reason this

workflow is being highlighted is because providing the ability to update software and firmware is one of the core capabilities for remaining resilient to future threats. Bugs and new threats will likely be found over the lifecycle of a medical device, and the ability to address and mitigate them via applying software patches is an incredibly powerful tool. Unfortunately, the ability to update and modify software is also something an adversary may attempt to take advantage of. Therefore, this is a high-value workflow that merits increased examination when performing threat modeling.

Much of the discussion for this example assumes the reader has already reviewed the discussion surrounding SNAP in Appendix A.1. Rather than repeating many suggestions that also apply to SKATE, this section will instead focus on two things: the unique features of the SKATE example, and the challenges surrounding modeling SKATE system's update process and what can go wrong with it.

A.2.2. (SKATE) What Are We Working On?

Figure 25 shows a high-level DFD of the SKATE system built out using the descriptions provided in the initial write-up. This was designed using a similar process to the one used to create a high- level DFD for the SNAP system in Appendix A.1.2.

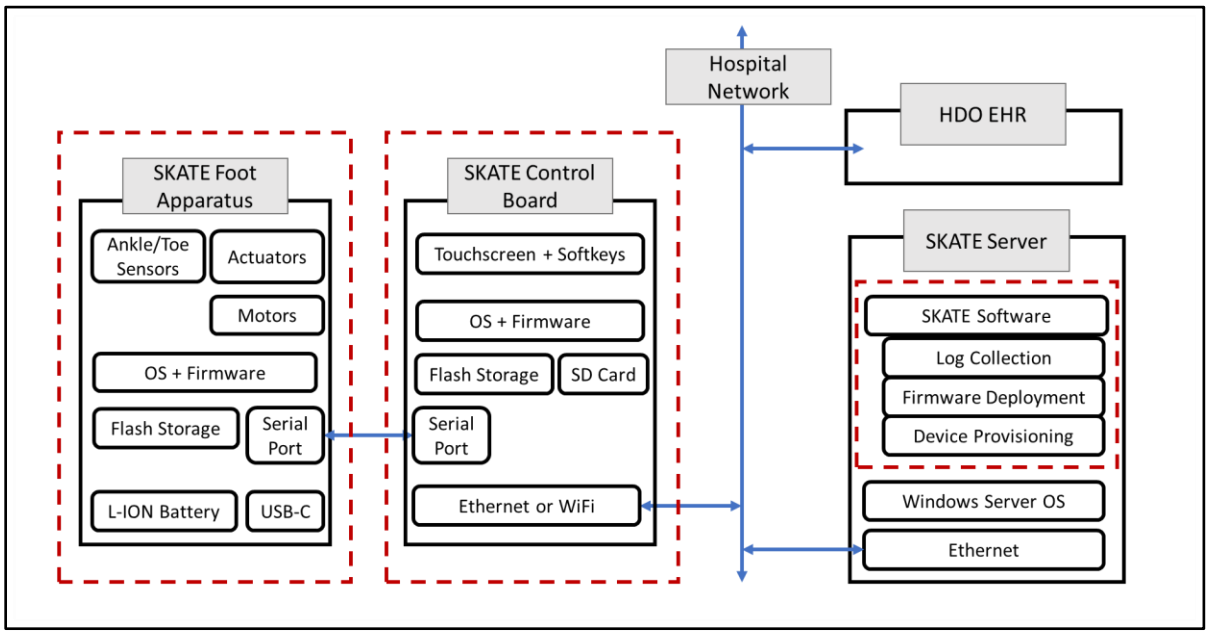


Figure 25: Data Flow Diagram for the SKATE System

Considering the complexity of the SKATE system, additional, more detailed DFDs of the individual components may be useful. These diagrams can handle the finer details on how different hardware components and software processes interact, which can be particularly important for embedded systems where those distinctions can have a significant impact on the threat modeling process. Figure 26 shows one possible example of a lower level DFD for the SKATE Foot Apparatus. While some of this DFD was described in the initial write-up, much of the creation of this diagram relied on outside knowledge of similar systems. Also, much of the naming terminology, such as Physical Layer, OS Layer, and Hardware Layer, were artifacts of the brainstorming process and are not standardized terminology.

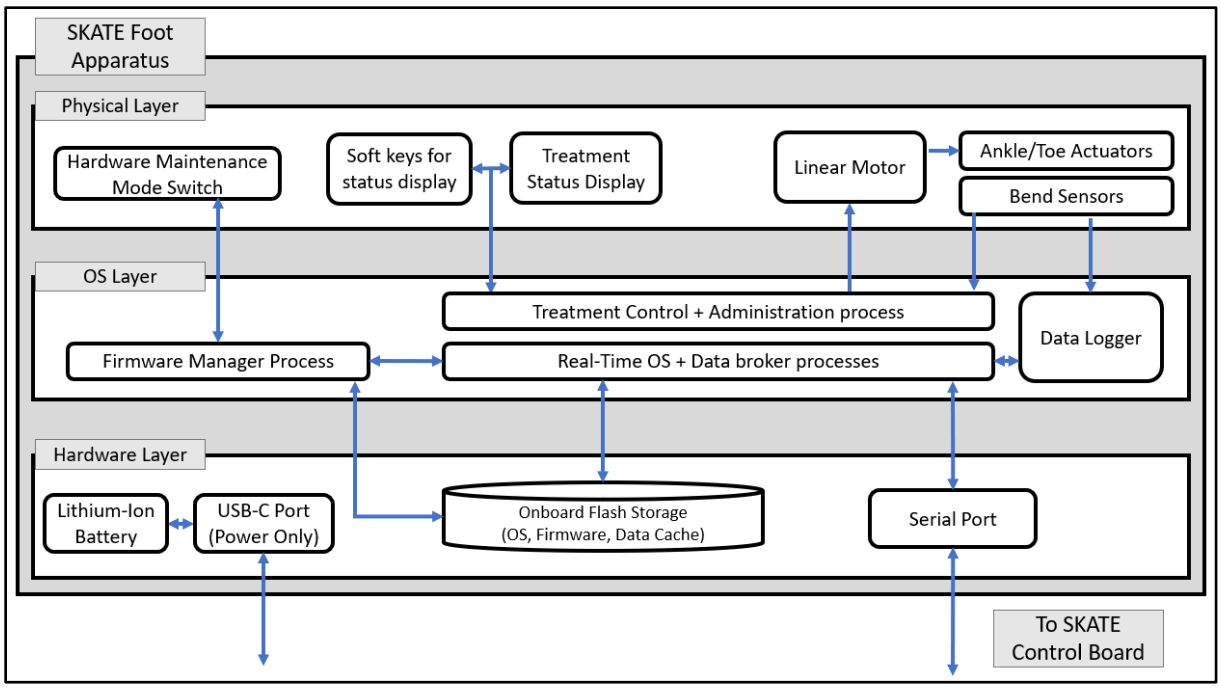


Figure 26: Lower Level DFD for the SKATE Foot Apparatus

Since the focus of this example will be on the software and firmware update process of the SKATE system, it is helpful to have a model of this process. One approach may be to create a Swim Lane Diagram. To highlight some of the common pitfalls when doing this, Figure 27 shows an intentionally flawed Swim Lane Diagram of the firmware update process.

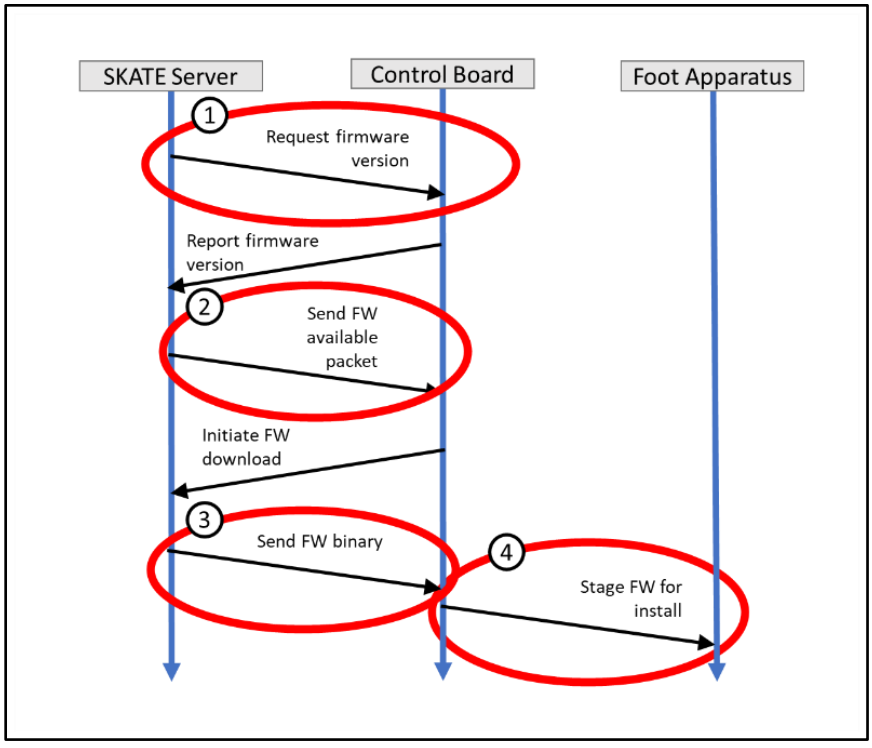


Figure 27: Intentionally Flawed Swim Lane Diagram Describing the Firmware Update Process for the SKATE Apparatus

At a high level, the Swim Lane Diagram does describe a basic workflow of the SKATE Server sending a request to the Control Board about the firmware version running on an attached SKATE Apparatus. If a newer version of the firmware exists, the Control Board downloads a copy of it and stages it on the Foot Apparatus to be installed. This leaves out some important information, however. Below are some questions and comments that might come up when threat modeling this workflow that the diagram does not address.

SKATE Server to Control Board communication (Highlighted Point #1):

- Does the initial request indicate the start of a new session that the SKATE server is initiating, or a continuation of a previous session initiated by the Control Board? The former seems to be implied by this diagram, but that would also imply that the SKATE server requires the ability to initiate connections to the Control Board, and the Control Board may be blocked by network segmentation strategies such as firewalls or network address translation. To avoid confusion, it can be helpful to explicitly label if the start of a Swim Lane Diagram is a new connection or not.
- Is there any authentication between the SKATE Server and the Control Board before a reply to a request is made? Can anyone ask what version of firmware the Foot Apparatus is running?
- What transport protocol does this use (UDP/TCP)? Is there encryption? If so, what is the type, version, and cipher suite of the encryption being used?

SKATE Server to Control Board communication (Highlighted Point #2):

- What happens if the Foot Apparatus is up-to-date? Is the “FW available packet” sent in both cases, and does it contain a flag specifying which case it is? This is where a reference to the actual protocol or adding a short comment can help the reader understand what is going on.
- Note: This may also be a place where Swim Lane Diagram may split based upon the state of the system. If the firmware is up-to-date, best practices suggest that the Control Board not download new firmware. How this split is handled will depend on the situation. It may be possible to simply add a comment saying that the session will end if a condition occurs. Another option is to create separate Swim Lane Diagrams for each path of the full session. Or the diagram could end at the point of divergence and include a link to the separate Swim Lane Diagrams that continue along their respective paths. Finally, it is possible that Swim Lane Diagrams are not the best way to represent this protocol and switch to a new modeling technique such as a State Diagram.

SKATE Server to Control Board communication (Highlighted Point #2):

- What happens if the firmware doesn’t finish fully downloading?
- Is there any verification that the firmware downloaded correctly and was not corrupted in transport? What happens if that verification fails?
- Is there any verification that the software that was downloaded was the software that was requested? The previous question focused on the integrity of the firmware. This question focuses on the security. How does the system ensure the software was written by a trusted source and not tampered with? Does the system check that the software is the expected version, and it is not being tricked by an attacker into instead downgrading the software to a previous version? Is this software written for the device being updated, or is it signed by a trusted source, but for a different device? How is this information being validated? Can this validation be tampered with or spoofed?

SKATE Control Board to Foot Apparatus communication (Highlighted Point #4):

- The lack of arrows going back and forth to the Foot Apparatus implies that much of the communication to it was not captured in this Swim Lane Diagram.
    - How does the Control Board query the Foot Apparatus to ask what version of firmware it is using?

- How does the Control Board negotiate sending the Foot Apparatus new firmware?
- What types of validation does the Foot Apparatus perform on the firmware?

Another way to model the update process is with a Cross-Functional Swim Lane Diagram. These diagrams can be useful to depict branching logic, which a complex process such as a firmware update workflow will likely contain. Figure 28 shows an intentionally flawed representation of a Cross-Functional Swim Lane Diagram.

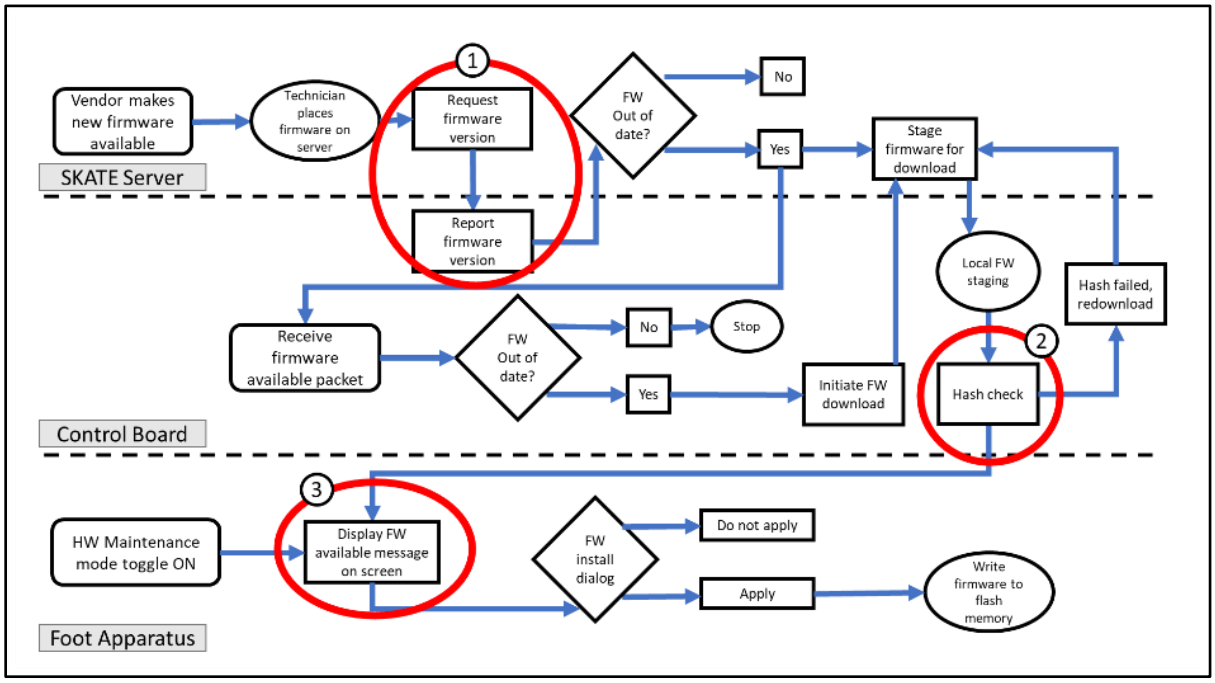


Figure 28: Intentionally Flawed Cross-Functional Swim Lane Diagram of SKATE Firmware Update Process

While more detailed than the earlier Swim Lane Diagram in Figure 27, the new diagram contains many of the same gaps, as well as raising some new issues. For example:

SKATE Server to Control Board communication (Highlighted Point #1):

- Just like the previous Swim Lane Diagram, there is no information about the protocol being used to initialize this connection, if there is authentication, and if there is encryption.
- This diagram implies that the firmware version request and the firmware version download are part of the same communication session. Therefore, the authentication and encryption settings become more important as the threat increases from Information Disclosure of the Foot Apparatus's firmware version to a potential Escalation of Privileges threat if an attacker can spoof or tamper with the firmware download.

SKATE Control Board Hash Check (Highlighted Point #2):

- The ambiguity around this one box, "Hash Check," can cause a significant amount of confusion and frustration, both for outside reviewers and when trying to threat model the system. For such an important step, including as much detail as possible is critically important.
- Beyond including an explicit reference to other documentation that describes the check in more detail, it can be helpful to add even a short description of the goal of the verification step in the diagram. For example, it can instead be labeled: "Corrupted firmware download integrity check." That way, it is possible to highlight that this check is to validate that accidental bit flips on the wire did not occur, and it is not a security check.

- If this is a security validation check, then be explicit about the type of check it is. For example, the box may be labeled “Code signing certificate validation and hash check” with a link to the specific algorithm being used. For example, the signature algorithm (e.g., *sha256WithRSAEncryption*) could be listed, as well as other important features like the timestamping being used; the certificate authorities (CA) it uses; if certificates can be revoked; and/or how versioning is being specified/maintained/checked, etc., in external documentation.
- **File hashing by itself is not a security control.** In order to be considered a security control, best practices suggest that the hashing algorithm would need to be considered secure. For example, the MD5 and SHA-1 hashing algorithms are known to be deprecated and therefore would not be considered viable security controls.

SKATE Control Board to Foot Apparatus communication (Highlighted Point #3):

- What does the negotiation to copy the firmware between the Control Board and the Foot Apparatus look like?
- Is there a separate diagram or model depicting the Foot Apparatus telling the Control Board what its firmware version is? When does that happen?
- Does the Foot Apparatus perform any validation checks on the firmware before it prompts the administrator that an update is available?
- Is there any authentication for the firmware update prompt and initiation? Or can any user update the firmware once it was downloaded?
- Are there any safety checks to ensure the firmware is not updated when the Foot Apparatus is in use?

Much of the documentation for this fictional example was left intentionally vague to highlight the frustration that comes with trying to understand the threats against high value dataflows. When modeling systems, organizations’ familiarity and your subject matter expertise will likely help to fill in the gaps. Part of the purpose of diagraming these dataflows is to help explain them to others who may not have that background knowledge of how a given system works. The goal of this discussion was to help highlight questions threat modeling diagrams may be able to answer. Likewise, the reasoning behind these questions will hopefully become clearer when proceeding through the process of “What Can Go Wrong?”

A.2.3. (SKATE) What Can Go Wrong?

As stated with the SNAP example, even though SKATE is a fictional device, there are a multitude of failure modes that can be considered across the whole system. Performing initial unstructured brainstorming using the diagrams already present, a short list of these failure modes may look as follows:

- Vulnerabilities in the provisioning process or key distribution process
- Remotely programming a treatment that isn’t necessary
- Remotely modifying treatment parameters to provide too much or too little actuation
- Leveraging the firmware update procedures to modify device firmware

As stated earlier, this example focuses on the firmware update procedures. One of the reasons for this is because it is a common dataflow across many types of medical devices, and it can be a challenging process to threat model. Another reason is that this is also a dataflow that can lead to multi-patient harm events. The outcome of a successful attack may lead to an Escalation of Privileges and full control over the system being targeted. Software-based safety settings can be overridden, backdoors can be added, and many of the security controls can be bypassed. Depending on how the update process is managed, an attacker may be able to subvert it to target many systems and HDOs at the same time.

Part of the challenge when identifying threats against software updates is the scarcity of guides detailing implementation recommendations and common threats. To assist with this challenge, below is a list of threats, categorized by STRIDE, that may be considered against each process that handles software

updates. This list is further divided by considering the source of software update and the recipient. In the SKATE example, certain components may be both a source and a recipient. For example, the SKATE Control Board both receives the firmware download and then serves it to the Foot Apparatus. Note, this is not a full list of threats and is meant to promote discussion rather than being an exhaustive list.

Software Update Source:

- **Tamper:** The update process itself can be tampered with to send invalid inputs, or to skip security checks such as verifying the update process is sending the correct software.
- **Tamper:** The software update source can send an update created by the attacker.
- **Information Disclosure/Spoof:** The software update source can send software updates to untrusted sources, allowing it to be reverse engineered. While keeping the software confidential is not a security control best practice, this is still something to consider.
- **Information Disclosure:** If code signing keys are stored on the software update server, they can be stolen. While less common with firmware updates as those code signing keys usually are managed by the system developers, sometimes configuration files need to be cryptographically signed and distributed to remote clients.
- **Denial of Service:** The software update source can be taken offline. This may be done by advanced adversaries to prevent patches from being deployed, but more often it may be a side effect, as many things can go wrong when trying to deploy patches on an already compromised network. Network routes may unavailable, DNS may be offline, Active Directory services may be down, or the update server itself may be hit by ransomware.
- **Escalation of Privileges:** A flaw in the software update process may allow the server itself to be targeted. Whenever parsing attacker-controlled input, there is a chance that malformed data may be sent that subverts processing logic. For example, if a server is expecting a client to send it their software version, a SQL injection may be received instead.

Software Update Recipient:

- **Tamper:** The software download process can be tampered with so that it will not update with new versions, or it will reply that it is up-to-date when it is not. This has traditionally been done by an attacker to ensure the system is not updated, which may interfere with their access to it.
- **Information Disclosure/Spoof:** The software update service could leak what version of software is currently running on the device to an untrusted endpoint.
- **Denial of Service:** A failed software update can cause the system to crash or become unstable.
- **Spoof/Escalation of Privileges:** An adversary can trick the software update process to downgrade the running software to an earlier version that may have known vulnerabilities.
- **Spoof/Escalation of Privileges:** An adversary can send their own software to be installed.
    - There are many ways this can happen. Is the software cryptographically signed? If it is not, then an adversary simply needs to get their software onto the endpoint device.
    - If the software is cryptographically signed, there still are ways that this can fail. The adversary can steal the signing keys. Depending on the certificate authorities that are trusted in the case of a Public Key Infrastructure signature scheme, the attacker may be able to legitimately request a signing key instead.
    - The verification process itself may be flawed. For example, the code verification step may check that the certificate accompanying the code is signed, but not actually check that the code is signed by a trusted CA.
    - While vary rare, there have been cases of the underlying cryptographic functions being attacked as well. One of the more prominent examples of this was the FLAME malware, which leveraged a novel hash collision approach to target the Microsoft Windows update process [25].

- **Escalation of Privileges:** An adversary can find a bug in the software update process to gain access. Whenever processing attacker-controlled input, this possibility always exists. For example, they could find a protocol flaw in how software is downloaded and verified. Perhaps when requesting the newest software version, an adversary instead sends a SQL injection or buffer overflow payload.

Miscellaneous:

These are considerations that don't easily fit into STRIDE categories:

- **Who can update software?**
  - Can any user of the system update the software? Or can only administrators update the software? Can it be initiated remotely, or does it need to be installed locally? Is there an automatic software update process in place?

- **When can software be updated?**
  - Can the software update be initiated when the system is in use?
  - Are there mechanisms in place to ensure the software update isn't done at a time when the system may need to be in use soon? Delays in critical care delivery due to the system updating may not be an acceptable consequence.

- **Are there multiple paths for software to be updated?**
  - Can the software update be initiated locally?
  - Can the software update be initiated remotely?
  - How is software copied over to the device? Via the network, USB, serial interface, CD?
  - Are the security controls in place for these different update paths the same or different?

- **Can software be rolled back?**
  - Sometimes a software update does not go well, and reversion to a previous version is required. As mentioned above, this can be taken advantage of by attackers.
  - Is a roll-back to any previous version of the software possible, or is there a factory reset option that restores the entire system to an initial state? What does this process look like?

- **How do you validate the software update worked?**
  - One challenge, particularly with systems that provide clinical care, is how to validate the system is in an operational state after a software update?
  - Are there any self-test or diagnostic procedures that can be run after the software is updated to validate the correct operation of the system?
  - How does the system ensure this self-test or diagnostic procedure is run after the software is updated?

While STRIDE terminology was used above to describe the threats, this does not mean STRIDE is required. It may be easier to instead use these threats as part of an Attack Tree and use them to go through each component of a system to identify vulnerabilities.

Appendix B. Additional Considerations from the Fictional Medical Device Examples

It is difficult to truly capture the full end-to-end threat modeling experience within the scope of this Playbook. While it is possible to speak in the general sense about what the threat modeling process is and how to walk through it, there is a significant amount of nuance when considering the subjects of the models themselves. Chapter 2 and Chapter 3 described several examples of how the threat modeling process unfolds. Furthermore, they illustrate how to approach several of the unique challenges that may arise when dealing with different types of systems. There exist additional points to consider:

- **Failure modes are not always immediately apparent**. A failure mode is actualization of risk to where the negative impacts of said risk occur. Mechanical failure doesn't always imply a completely broken machine; it could instead mean a malfunctioning or tampered sensor. Leakage or tampering of personal information can lead to long-running ramifications outside the scope of device use, but it could also lead to incorrect treatment being applied. If a patient is interacting with a device, carefully consider what failure of that device entails and how to address it.
    - SNAP and SKATE both face mechanical failure in different forms. With SNAP, patient injury can be a by-product of device failure. A change in scanning parameters can lead to incorrect or incomplete scans. With SKATE, incorrect treatment parameters can lead to a snapped toe. However, SKATE also has a battery that could burst into flames if kept in poor conditions.
- **Avoid silent risk transference**. Furthermore, ensure non-silent risk transfers are carefully considered, justified, and documented. A networked device in a physically secured room still provides opportunities for attackers beyond those within the room. Conversely, the presence of a firewall does not mean that a device is immune to physical attacks. The power can go out, faulty USB sticks might be used, or a frustrated technician may smack the machine for being slow on a bad day. Securing a device is not solely the HDO's responsibility.
- **Pay close attention to connections made with external entities**. External generally refers to anything that is beyond an organization's ability to change or configure (e.g., an HDO's EHR server). For example, when considering how to perform software or firmware updates, is the update server considered a trusted entity despite having ownership of it? What about when updating a mobile app where the app relies on a third-party store? Make sure that not just the identity, but also the data coming from these entities is verified before using it.
- **The cloud is not always exclusively an external entity**. Despite the name, if a cloud is employed as part of a system, it is the responsibility of the employing organization to ensure that the parts it controls are properly modeled. The AMPS example discusses this by modeling the AMPSCS. If the AMPSCS utilizes a cloud provider's storage solution, that does not allow for the transfer of threats to that cloud provider. However, capabilities provided from the cloud may help mitigate discovered threats. For instance, a cloud provider may offer DDoS protection to help keep services running.
- **Identify and document critical dataflows end-to-end**. These are often high-value targets for attackers in terms of return on investment. SNAP and SKATE both have firmware update processes that an attacker could leverage. SNAP produces high-value patient information that eventually comes to rest at an HDO's EHR. AMPS provides patient data access in the cloud. Best practices suggest that all components that touch a critical dataflow be scrutinized and given supporting technical documentation.

- **Trust boundaries highlight where security controls may be most effectively employed**. By attaching two components, a system is effectively exposing those components to new attack surfaces. For example, in SNAP, the trust boundary for the scanning ring and scanning base are joined. This reflects how the physical device is expected to operate in the real world, rather than on paper. However, in doing so, the scanning ring is exposed to the scanning base's attack surface. If this approach is taken, ensure that the use case makes sense and that the justification is documented.

- **Make sure all identified threats are documented**. There are many options to eliminate, mitigate, accept, or transfer threats, and focused analysis may be employed to find the strongest options available. However, just because a threat has been eliminated does not mean that documentation of that fact is not needed. Similarly, just because the possibility of a threat is small does not mean that it may be ignored.

- **Do not overestimate the strength of strategies for addressing threats**. This could cause underestimation of residual risk and leave devices more exposed to adversaries. For example, a proprietary control whose security depends on a degree of randomness could be used by attackers who employ brute force tactics. Alternately, custom cryptographic implementations are often easily breakable by attacks that are well-understood by cryptographers. Finally, while encrypting a data channel communicating with an EHR is a good mitigation against information disclosure, ensuring those keys are secure as well may be needed.

- **Threat model the threat modeling process**. While much of this Playbook focuses on how to initially build up a threat model (i.e., "What are we working on?"), consider what could happen postmarket as well. Ask "What could go wrong?" with threat models after releasing your device. New vulnerabilities or issues may be discovered, and employees with critical knowledge may depart, or documentation may get archived and prove difficult to retrieve. If gaps in threat modeling processes are identified, ask "What Are We Going to do about it?" and ensure that necessary workflow adjustments are made for long-term maintenance.

- **Remember - All models are wrong, but some models are useful**. There is no answer key for what a threat model looks like. The important thing is that by engaging with the process, the safety and security of the system at hand may be improved.

Appendix C. Bibliography

[1] Threat Modeling Manifesto Working Group, "The Threat Modeling Manifesto," [Online]. Available: https://www.threatmodelingmanifesto.org/. [Accessed 22 August 2021].

[2] A. Shostack, Threat Modeling: Designing for Security, Indianapolis: John Wiley & Sons, 2014.

[3] A. Shostack, "DFD3 Standard," [Online]. Available: https://github.com/adamshostack/DFD3. [Accessed 21 4 2021].

[4] J. Cawthra, N. Grayson, B. Hodges, J. Kuruvilla, K. Littlefield, J. Snyder, S. Wang, R. Williams and K. Zheng, "NIST SP 1800-30 Securing Telehealth Remote Patient Monitoring Ecosystem," NIST, 2021.

[5] Object Management Group, "Unified Modeling Language," [Online]. Available: https://www.uml.org/.

[6] B. Schneier, "Attack Trees," Schneier on Security, 12 1999. [Online]. Available: https://www.schneier.com/academic/archives/1999/12/attack_trees.html.

[7] L. Martin, "Cyber Kill Chain," [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html.

[8] The MITRE Corporation, "Overview of How Cyber Resiliency Affects the Cyber Attack Lifecycle," 2015. [Online]. Available: http://www2.mitre.org/public/industry-perspective/documents/lifecycle-ex.pdf.

[9] MITRE Corporation, "MITRE ATT&CK Framework," [Online]. Available: https://attack.mitre.org/.

[10] NIST, "NIST Cybersecurity Framework," [Online]. Available: https://www.nist.gov/cyberframework.

[11] MITRE, "MITRE D3FEND Knowledge Graph," [Online]. Available: https://d3fend.mitre.org/.

[12] Joint Task Force, "NIST SP 800-53 Rev. 5. Security and Privacy Controls for Information Systems and Organizations," NIST, 12 2020. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final.

[13] FDA, "Human Factors Considerations," 23 12 2017. [Online]. Available: https://www.fda.gov/medical-devices/human-factors-and-medical-devices/human-factors-considerations. [Accessed 22 08 2021].

[14] Microsoft, "SDL Security Bug Bar (Sample)," 2018. [Online]. Available: https://docs.microsoft.com/en-us/security/sdl/security-bug-bar-sample.

[15] FIRST.Org Inc., "Common Vulnerability Scoring System SIG," [Online]. Available: https://www.first.org/cvss/.

[16] M. Chase and S. Christey Coley, "RUBRIC FOR APPLYING CVSS TO MEDICAL DEVICES," MITRE, October 2020. [Online]. Available: https://www.mitre.org/publications/technical-papers/rubric-for-applying-cvss-to-medical-devices.

[17] MITRE, "CVSS Rubric for Medical Devices Calculators," MITRE, [Online]. Available: https://mitre.github.io/md-cvss-rubric-tools/.

[18] Joint Task Force Transformation Initiative, "NIST 800-30 Rev. 1 Guide for Conducting Risk Assessments," NIST, 9 2021. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final.

[19] FDA, "Postmarket Management of Cybersecurity in Medical Devices: Guidance for Industry and Food and Drug Administration Staff," December 2016. [Online]. Available: https://www.fda.gov/media/95862/download.

[20] AAMI, "TIR57: Principles for medical device security - Risk management," 2016.

[21] HSCC Joint Cybersecurity Working Group, "Medical Device and Health IT Joint Security Plan," January 2019. [Online]. Available: https://healthsectorcouncil.org/wp-content/uploads/2020/06/HSCC-MEDTECH-JSP-v1.pdf. [Accessed 11 August 2021].

[22] FDA, "Design Control Guidance for Medical Device Manufacturers," March 1997. [Online]. Available: https://www.fda.gov/media/116573/download.

[23] OWASP, "Threat Modeling Cheat Sheet," [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html.

[24] Software Engineering Institute, "Threat Modeling: A Summary of Available Methods," August 2018. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=524448.

[25] swiat, "Flame malware collision attack explained," [Online]. Available: https://msrc-blog.microsoft.com/2012/06/06/flame-malware-collision-attack-explained/. [Accessed 2012].

[26] OpenStack, "OpenStack OSSA-Metrics Wiki," [Online]. Available: https://wiki.openstack.org/wiki/Security/OSSA-Metrics. [Accessed 21 05 2021].

[27] Wikipedia, "Kerchoff's principle," [Online]. Available: https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle.

[28] J. Tirpak, "Find, Fix, Track, Target, Engage, Assess," Air Force Magazine, 1 7 2000. [Online]. Available: https://www.airforcemag.com/article/0700find/.

[29] Microsoft, "The STRIDE Threat Model," November 2009. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN.

[30] Omada Health, "INCLUDES NO DIRT: A Practical Threat Modeling Approach for Digital Healthcare and Beyond," September 2019. [Online]. Available: https://www.omadahealth.com/news/includes-no-dirt-a-practical-threat-modeling-approach-for-digital-healthcare-and-beyond.

[31] Intel Corporation, "Prioritzing Information Security Risks with Threat Agent Risk Assessment," December 2009. [Online]. Available: https://www.researchgate.net/publication/335589639_Prioritizing_Information_Security_Risks_with_Threat_Agent_Risk_Assessment_TARA. [Accessed 20 08 2021].

[32] Intel Corporation, "Threat Agent Library Helps Identify Information Security Risks," September 2007. [Online]. Available: https://static1.squarespace.com/static/5a111571d0e628a8679b6b6c/t/5c379b84b8a045a55b983f3a/1547148165167/Intel+-+Threat+Agent+Library+Helps+Identify+Information+Security+Risks.pdf. [Accessed 20 August 2021].

[33] MITRE, "Common Weakness Scoring System (CWSS)," September 2014. [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html. [Accessed 21 August 2021].

[34] Wikipedia, "DREAD (risk assessment model)," [Online]. Available: https://en.wikipedia.org/wiki/DREAD_%28risk_assessment_model%29. [Accessed 21 August 2021].

[35] R. Munroe, "XKCD: Drop Tables," [Online]. Available: https://xkcd.com/327/.

[36] Wikipedia, "Wikipedia SQL Injection," [Online]. Available: https://en.wikipedia.org/wiki/SQL_injection.

[37] DistriNet Research Group at KU Leuven, "LINDDUN," [Online]. Available: https://www.linddun.org/.

[38] Wikipedia, "All models are wrong," [Online]. Available: https://en.wikipedia.org/wiki/All_models_are_wrong. [Accessed 26 August 2021].

[39] NIST, "NIST SP 800-30 Rev. 1: Guide for Conducting Risk Assessments," September 2012. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final.