# Title

Good afternoon, everyone. I'm Charles Wilson, technical fellow for cybersecurity engineering at Motional. This presentation will cover cybersecurity requirements taxonomy-based threat modeling.

# Introduction

The bulk of the information I present is complete, documented, and has a bow on it. This talk is different. The material I’m going to present will cover emerging technology that we're working on. It's important enough that we’re sharing it as covers a conceptual framework applied in a way that hasn't been done before.

The talk is divided roughly into two parts, the first will cover what a cybersecurity requirements taxonomy is, and the second how it can be used as the basis for threat modeling.

Before that, let’s take a moment to establish some terms.

Terminology

The AVCDL uses terminology consistent with this threat modeling glossary diagram; created by Steven and Miguez. You'll note that it that it has two distinct halves to it, which are separated by the attack. On the left, we have the structural form. On the right, we have the implication area. When considering threat modeling, what we deal is fundamentally in the upper left region. We have the threat traversing a boundary, interacting with an attack surface. That's what I'm focusing on today. If we were to continue downward, we’d see that the attack surface may possess vulnerabilities which are defects. And that's fine. But let's stay focused on threats traversing boundaries.

With the terminology out of the way, we’ll dive right into the cybersecurity requirements taxonomy (or CRT).

Assets

The first dimension of the taxonomy covers the assets we're going to be dealing with.

Asset Classes

All the assets we care about fall into a set of classes. These classes can be assigned to a one of three states.

The first state is data at rest. In the context of data at rest we have executables, which is any binary data that can be run in the system. It doesn't matter if it's software or firmware.

Then we have configuration data. This is data used to establish the personality of the system. This metadata tempers how our system behaves.

Next, have the two types of data stores. The first are databases which is structured data, typically managed by specialized engines, and the second is unstructured data. This is everything that's not a database.

There are two very specific types of data we're going to call out regardless of how they’re embodied. One is credentials. This is the data used to establish and manage identity of any entity within the system. The other is logs. This is data used to record system events. Logs obviously decompose into an assortment of different types. And we tend to care about security logs because we're security focused, but other types of logs may also be consideration here.

In the category of data in motion, we care about PII and packets. We care specifically about PII because we want to ensure that we don't leak that information. As for packets, they’re what we use to move data around the system.

And finally, we have data in use for those of you who are watching side channel attacks. This is broadly speaking memory. This could be registers. It could be stack. It could be RAM. Its memory. It's the data actively being manipulated within an executing system.

Cybersecurity Properties (title)

The second dimension of the taxonomy is that of cybersecurity properties.

Cybersecurity Properties Timeline

We've been talking about cybersecurity properties for a really, really long time. When asked what these are, everyone says the CIA. If you press for details, you’ll probably hear mention of the Saltzer and Schroeder work from 1975.

This is in no way the first discussion of cybersecurity properties. We can trace things back at least to 1964 in a paper entitled, “On Distributed Communication, Security, Secrecy and Tamper-free Considerations.” And just because we have the Salter and Schroeder’s paper doesn’t the discussion ended there. In fact, the CIA isn’t even mentioned in the Department of Defense's Trusted Computer Systems Evaluation Criteria (AKA the Orange Book).

Throughout the early 2000s, we have an ongoing discussion of how the CIA is necessary, but not sufficient. In fact, we can look at the end of the 2010s and see that we're still debating these.

Cybersecurity Properties

In 2001 NIST SP 800-33 introduced an underlying technical model for information technology security. From 800-33, we can identify seven cybersecurity properties which are called out in the GRVA material that went into the making of UN R155. In that material these are referred to collectively as the extended CIA. They are confidentiality, integrity, availability, non-repudiation, authenticity, accountability, and authorization. Together, I would argue that these are orthogonal. I don’t know how many times I’ve seen people jumping through hoops of fire to try to make everything fit into the CIA alone. It doesn't work.

Resource Access Working Model (1)

This is the resource access model that I've put together to provide a place to hang this discussion. At the center of the diagram going from top to bottom, we have a requester making a request to a resource owner. The resource owner performs an operation on the resource and then returns information to the requester. Additionally, the transaction may be subject to logging.

Flanking this transaction on either side, we can see schematic of the data flows. A request comes in, has source, destination, there's payload. There's a check that may be there. That payload is a command with optional data. The command will either perform a write operation, in which case there's a value in, or a read operation. This comes back as a write status or read data that goes into a payload which forms a response. This is the working model for resource access.

Now let’s consider how cybersecurity properties might be applied to this diagram.

Resource Access Working Model (2)

If we just applied the CIA. We would cover these elements here. The confidentiality, integrity, and availability are shown on the elements where controls would be placed. Note the number of places where we’d normally want to place controls that we're not covering.

Resource Access Working Model (3)

When we include the other properties (non-repudiation, authenticity, authorization, and accountability), we see that the previously unaddressed control points are now covered.

Combining Asset and Property

If we bring asset and property together, we get a visualization like this. Now work similar to this has been proposed in the past, but we’re not quite done. So, what’s left?

Simple System

To answer that, we need to know what a threat model is and how we decompose a system.

# What is a Threat Model?

Let’s start by addressing what a threat model is.

A threat model can be considered a representation of a system’s data flows, data stores, and interactors. Alternately, as a collection of views of a system. And finally, as an engineering design document. This may be the most important function it serves.

Fundamentally it’s a model of a system we can reason on with respect to cybersecurity.

Simple System – Block View

So, here's a block view of a simple system. You'll note this is not an automotive system. But it is a system that anyone can relate to. I would argue that it manifests all of the behaviors that we care to model. On the right, you have an interactor, an administrator who is interacting with the system via two modalities, the first is a web browser and the second a terminal. On the left (the system of interest), we have a web server and a console interface that the administrator is interacting with. Both are communicating with the core service. That core service and terminal interface have configuration data and application data. We have multiple processes. We have read-only and read-write data stores. And we have different types of data flows.

Simple System – DFD

We can create a data flow diagram from the block view. All of those entities have a one-to-one correspondence within the DFD realm. Our boundaries can be simplified to just two. There's the core service boundary between the core service, the web server and the console interface. And then we have a machine boundary between the web server browser pair and the console interface terminal pair. One could argue that there's also a network boundary, but it's subsumed by the machine boundary.

Now if you look at this, you can say yes, this is correct, but actually there are pieces that are implicit but missing. Anyone who’s done extensive threat modeling will appreciate the problem this creates.

Layers (title)

In order to understand the problem, let’s explore our DFD more.

Simple System – DFD (Level 2)

Missing from our original DFD was the operating system itself. These are the new elements in red, the file system, database manager, Interprocess communications manager, network manager, and serial driver. We tend to forget about these processes, all of which our first diagram conveniently ignored. You'll note that we have now have a many more boundaries.

But this diagram isn't a complete view either.

Simple System – DFD (Level 3)

The database manager has to work through the file system. The network manager has to talk to a network driver. We also have additional boundaries.

The point is that as we create our models, if we don't take these elements into consideration, we're going to miss things. By the same token, if we only look at issues that exist at this highest resolution, we're going to miss issues in the end-to-end traffic.

Layers

This is where layers come into play. We have four distinct layers.

We have the physical layer, which is the actual hardware interface. The network layer, which includes all system-mediated transports. The protocol layer contains custom data transports. And finally, the application layer which is where data handling occurs within executables.

Now from top to bottom, you go from no control to complete control. With network protocols you don't get to decide what types of cybersecurity controls are in place. They're pre-defined and you work with them. When you're doing custom data transport, you have pretty much full control over how that data goes back and forth. You may not know what to do with the data that you're sending, but you control how that data is moved from point A to point B. Finally, within an application you have full control over the data, how you're storing it, and all the controls applied.

Taxonomy Space

So, when we put these things together, you end up with the full taxonomy space.

Cybersecurity Requirements

So now let's see how the taxonomy informs the cybersecurity requirements.

Identifying Requirement Needs – Application Layer

We can create cybersecurity requirements based on the desired cybersecurity properties of an asset in the context of a particular layer. So, for instance, within the application layer, we don't make any assertions about packets because packets don't exist in the application layer. Nor do we make assertions about authenticity or accountability for unstructured data. We do make assertions about confidentiality of configuration data.

For each layer within the taxonomy, we create a set of assertions. From these, we can create requirements. Those requirements are not backward looking. They're not considering historic attacks and attempting to assert requirements to protect against them. The requirements are positive assertions of how ensuring cybersecurity properties.

For instance, we can say that when have an executable, we want to assert this cybersecurity property is ensured. That requirement will never change because it's a fundamental. It’s future proof.

Cybersecurity Global Requirements Catalog

Once you go through this exercise, you have a global requirements catalog. All possible requirements are here. This is a portion of the catalog provide in the AVCDL. On the left-hand side are the ID and description. These are followed by the property, asset, and the layer that each apply to. The catalog is only about 70 entries. Compare that to the MITRE ATT&CK framework. Additionally, all requirements are INCOSE compliant.

Tailored Cybersecurity Requirements

Now we can take these global cybersecurity requirements and for each of the functional requirements, create tailored requirements from them and then create the augmented functional requirements. These are then used to drive development stories and development tasks. Now, you may say, “you only have 70 requirements. You don't want to constantly be a situation where you are applying dozens of requirements to individual functional requirements.” And I agree absolutely.

Macro Cybersecurity Requirement

What you do is create macro requirements which are then attached to the functional requirements. A macro requirement implements multiple tailored requirements in the same way that a functional requirement implements multiple technical requirements.

Macro Cybersecurity Requirement Example (SecOC)

So as an example of this, we can look at SecOC. There are four requirements from the global catalog that SecOC provides coverage for. Credentials crossing trust boundaries are encrypted. Communication crossing trust boundaries ensures data integrity. Communication crossing trust boundaries ensures authentication. And custom protocols use current best practices for authentication and key exchange. We wrap those together and create a macro requirement named SecOC. Specifically, SecOC security profile 1. Not all of the SecOC security profiles conform to all four of these requirements, but profile 1 does.

Applying the Taxonomy

So that's the requirements taxonomy and how it’s used to build out our requirements.

Now, let's look at how we apply the taxonomy. We're going to use the Microsoft threat modeling tool as a basis because it's both freely available and configurable.

Threat Model Element Attributes

What we've done is develop a new set of attributes for each of the standard DFD entities (resource, process, interactor, boundary, and data flow). That allow us, using these attributes, we can build up standard pieces for our use case, which will generally speaking, be automotive. Other use cases are certainly possible.

Threat Modeling Tool Rules

We then took our requirements and created rules. Within the context of the Microsoft threat modeling tool, rules are applied when a data flow crosses a trust boundary. Rules can reason on the trust boundary, data flow, source, and destination information.

Not every requirement can be applied to threat modeling. In fact, only about one quarter of the global requirements lend themselves to a treatment of this kind. It's important to recognize the limitations of threat modeling tools. They aren’t magic boxes.

Quality of Results

The important thing is that what we get out of this customized threat modeling tool is the ability to have a one-to-one correspondence between a rule and its mitigation. Typically, when you're using something like the Microsoft threat modeling tool, you're going to get generic mitigation recommendations that says you are subject to some type of attack. That’s nice, but how does that help a developer make an informed decision as to what to fix?

Since we use our requirements as the basis for the rules, any violation of the rule points back to a single requirement. So, you don't get false positives. The only way that you would get a false positive is if you didn't put all the data in that you need. The results you get are very clear and very definitive.

Because our categories use our cybersecurity properties as the classification, instead of using STRIDE, it’s much clearer what we are concerned with.

The question becomes, is this useful? The last thing that we need is something that that doesn't give us useful information. And for individual tests that we have run, we see that when compared to the baseline from the Microsoft threat modeling tool and then using the cybersecurity requirements (CRT) as the basis, we get much higher quality output from the latter.

Now, you might say well, you're applying a much smaller number of rules, so you're not covering all cases. The thing to remember is that threat modeling does not address everything. Threat modeling is not performing attack surface analysis. Threat modeling cannot reason on metadata. For instance, if you're using encryption, it can't tell you that you're that you're using the correct cipher suite. You may be using a null cipher suite; in which case your encryption is completely pointless. That's something caught elsewhere.

STRIDE vs CRT

Finally, let’s look at STRIDE versus the CRT. The focus of STRIDE is an attack-based system. The focus of CRT is property based. STRIDE values are not unique. They're not unambiguous. They're not complete. They're not grounded. There is no fundamental underpinning for STRIDE. It’s an arbitrary collection of problem types. As anyone who's done threat modeling using STRIDE knows, it does not scale well. And it's only actionable some of time.

Even with its smaller ruleset, the CRT provides results which are always unique. They always unambiguous, because all of our requirements are INCOSE compliant. It is complete in that it covers all of the elements. It's well grounded. We have documentation providing a basis for all of our choices. It is scalable. Its behavior is very well defined and it's always actionable because all of the violations of the rules point to an actionable requirement, that can to be attached to a functional requirement.

Where to Go from Here

Where to go from here? As I said, this is a work in progress and as we get further along, we'll release the materials that we have on this topic. As you can imagine, testing a set of rules and then also providing those tests so that we can assert that it's not just a magic box takes time. We want people to be able to have a well-founded understanding of their threat modeling system and have assurance that the tests which verify the rules are correct. That’s the goal.

# AVCDL on YouTube

The video for this talk will be up on our YouTube channel alongside other that relate to the cybersecurity requirements taxonomy and cybersecurity requirements themselves.

AVCDL on GitHub

Up on GitHub you can find source and distributions versions of all the AVCDL materials, include the sources used to build this talk.

# References

Here are some references that were used within the context of this presentation for those interested in more detailed information.

Questions

And now I'd like to open for questions.