# Title

# I'm Charles Wilson, Cybersecurity Engineering Technical Fellow at Motional, responsible for cybersecurity development lifecycle practice.

# This presentation will cover threat prioritization within the context of the AVCDL.

# Training Path

# This diagram shows the overall AVCDL training path.

# If you’re taking this training, it’s assumed that you’ve already completed the AVCDL overview training.

# This training covers threat prioritization.

# Introduction

Unlike many aspects of product development, cybersecurity is traditionally treated like a one-man band.

They're responsible for bringing their own instruments, playing their own music, and dealing with everything related to the function they try to accomplish.

A classic example is that of penetration testing.

You can penetration test anything.

You don't need input from any other group.

You just need the device and you can perform a penetration test and produce a penetration test report.

This in and of itself is fine.

<pause>

But if we then go and look at something like threat modeling, we discover that it’s the same kind of animal.

If we can have the device or the design or the requirements, we can synthesize a DFD which we can then reason on and generate a report from.

<pause>

Or we could do static analysis and take all the source code and run it through a separate program, generate output, and then produce that set of results.

<pause>

Or perhaps we could do something like fuzz testing where we actually are slightly more intrusive, but we still bring all the other pieces to the table ourselves. And we again generate a report.

<pause>

Now it's not desirable for us to treat all of these activities and several more in isolation, but that's the way that they have traditionally been treated.

<pause>

What we would like to have is not a collection of one-man bands, but rather a symphony orchestra where discrete elements can be composed and work harmoniously with each other.

That's what the threat prioritization aspect of the AVCDL is intended to accomplish.

# Threat Prioritization and Feedback

In the development of any complex system, there's a strong motivation to have processes and procedures which can both be shared and require few changes to existing workflows.

The threat prioritization process was designed with these goals in mind.

One way the workflow achieves its light touch is by having a single simple mechanism for feeding changes back into earlier processes.

<pause>

To illustrate, we’ll use this diagram.

It’s a bit of an eye chart, so we'll look at it in stages,

but basically, what we're showing here is that activities providing feedback follow a common path.

<pause>

Although they don’t generate feedback, the foundation and requirements phases receive feedback from other phases, specifically in the area of requirements.

<pause>

In the design phase, what we see is that both the attack surface analysis and threat modeling activities can filter into the threat prioritization process.

We’ll cover this in detail shortly.

As we can see, the threat prioritization process sends feedback through the issue tracking system into either the element design review or the element requirements review.

<pause>

In the implementation phase, we can see four activities which are feeding into this threat prioritization process.

Those being fuzz testing, static analysis, dynamic analysis and security code review.

The feedback can go either directly to the element implementation process of the implementation phase or,

depending on the scope of the change required,

into either the element design review of the design phase or element requirements review of the requirements phase.

<pause>

Similarly, in the verification phase, we see that penetration testing, attack surface analysis review and threat modeling review feed into this same threat prioritization process, feeding back into the stages that we've mentioned already.

<pause>

And finally, in the operation phase, cybersecurity monitoring also feeds into the threat prioritization process.

The advantage that we have is that regardless of the activity that we undertake within the AVCDL,

we have a single, simple, consistent set of activities that we do, regardless of the threat’s source.

# Threat Candidate Sources

Now let's look at all of the areas at the same time.

<pause>

For this, we're going to refer to the AVCDL framework diagram.

We can see that **secure design review**, **attack surface analysis**, and **threat modeling** in the **design phase**;

<pause>

**static analysis**, **dynamic analysis**, **fuzz testing**, and **secure code review** in the **implementation phase**;

<pause>

**penetration testing**, **threat model review**, and **attack surface analysis review** in the **verification phase**;

<pause>

and **identify and confirm vulnerabilities activities** in the **operation phase**

all serve as threat sources feeding into a common threat prioritization process.

Threat Prioritization Workflow

Here's our overall workflow.

The methodology used is based on guidance from NIST SP 800-30.

Within the workflow, we have three individual activities, threat candidate ranking, ranked threat candidate risking, and threat candidates slicing.

Let's examine each of these activities in detail.

# Threat Candidate Ranking

The first activity in the threat prioritization workflow is threat candidate ranking.

In this activity, we're ranking in order to establish the relative exploitability or likelihood of each threat candidate.

As mentioned earlier, these candidates can be sourced from several activities including threat modeling, incident response, and attack surface analysis.

The ranking itself should be done using a standard methodology such as the common vulnerability scoring system or CVSS.

There are additional methodologies and we'll be discussing those later when we cover quantization.

The ranking system used should consider factors such as:

• Mechanism

• Locality

• Maturity

• Scope

• Required privileges

Ranking lends itself to a continuous system and once ranked a threshold may be applied to reduce the number of candidates to be considered in the risking activity.

If the threshold is applied, the dismissed candidates must be documented, along with the dismissal rationale, in the candidate’s source tracker.

This is seen here in the notification that goes back from the ranking activity.

<pause>

Generally speaking, the minimum inputs for this process are going to be the what, the where, and the worst case of each threat candidate.

<pause>

And the output will include the likelihood of that threat candidate.

# Threat Candidate Information

Generally, the AVCDL recommends that information exchanged between various parts of the cybersecurity activity pipeline be done using a JSON-encoded SARIF file.

SARIF, the Static Analysis Report Interchange Format, is highly flexible, and is supported by numerous applications.

<pause>

Here’s a table showing a number of the processes which generate threat candidates and how they can represent their unique information in SARIF.

The processes in the table are listed in the order they appear within the AVCDL.

<pause>

If we start with static analysis, we can see that it has a test which is a checker.

It has a location being a file and a line.

And it has an issue which is a rule violation.

<pause>

Notice that **static analysis**, **fuzz testing**, and **source code review** are very similar in that all are using checkers and have location information defined by a file and line combination.

In the case of **secure code review**, the checker is a manual list of things that are being reviewed by the person doing the secure code review.

In the case of **fuzz testing**, the issue is a specific fault, because generally speaking, fuzz testing operates by running tests and reporting failures.

<pause>

If we look at **threat modeling**, we see that rather than a checker, we have a rule;

and rather than a location, we have a graphic, because threat modeling is traditionally done using data flow diagrams.

The issue we get is a violation of that rule.

<pause>

Similarly, with **dynamic analysis**, we have a checker, with the issue being a failure of that checker, similar to the way fuzz testing works.

In this case, the location is a variable resolution location.

This is to say that depending on the amount of information we have, we're going to be able to identify the location of the fault with various degrees of precision.

<pause>

For penetration testing and incident response, what we have are tests with the location being test steps.

Both have issues which represent failures that have been identified. These are very high-level, very process-oriented in nature.

<pause>

The last thing for consideration is attack surface analysis.

This one is a bit more abstract in that we are looking at types of attack surfaces. Instead of the location, we're considering the exposure that is expressed by the surface.

And our issue is an excess of access via the exposed surface.

<pause>

No, these are not 1-to-1 to each other.

However, from a conceptual standpoint, SARIF allows us to convey information in a way which allows us to have a uniform mechanism for processing the data.

# Threat Candidate Risking

The second activity in the threat prioritization workflow is threat candidate risking.

Here a risk SME takes the ranked threat candidates and assigns a risk impact to each.

<pause>

As a reminder, our ranked threat candidates contain the information of what the threat is, where it's located, what the worst-case scenario is, and what the likelihood as determined by the ranking activity was.

<pause>

And the output augments the input information with the impact information.

# Threat Candidate Slicing

The third activity in the threat prioritization workflow is threat candidate slicing.

We take the threat candidates which have been both ranked and risked and slice them.

That is, we bisect the space they exist in to determine which present an uncontrolled risk.

These are designated as uncontrolled threats and forwarded to the issue tracking system.

If an issue is determined to be controlled an update dismissed candidate notification is sent to the source tracker document in order to record this.

Additionally, a threat report will be generated for audit purposes.

<pause>

It's important to note that it's the responsibility of the standard issue management process to disposition issues by applying the standard risk treatments of avoid, reduce, share, or retain.

Additionally, it's necessary to document the reason for this disposition.

<pause>

And as a reminder, our inputs are what the threat is, where it is located, what the worst case is, what the likelihood is, and what its impact is.

<pause>

The output list is truncated, in that we're only including the uncontrolled threats.

And so, rather than indicating whether the threat is controlled or uncontrolled.

We simply provide a priority for the threat.

The reason for this is that all the threats being entered into the issue tracking system are uncontrolled threats.

# Slicing

Now let's take a moment and look at how we're going to actually do the slicing operation itself.

<pause>

Here's a diagram that shows the way that the slicing or determination of controlled and uncontrolled threats is achieved.

The diagram is adapted from the FDA’s **Post-market Management of Cybersecurity and Medical Devices** document.

We have the values from the ranking and the risking activities, the likelihood and impact values.

We're showing them here as being continuous.

Similarly, the range from controlled to uncontrolled is also being shown as continuous.

The line bisecting the exploitability-severity space, shown here in red, is the risk appetite.

We need it not only to establish those threats we consider controlled or uncontrolled, but also to prioritize them.

Above the line, the threats are deemed to be uncontrolled, that is, in need of mitigation.

We will cover the mechanism used to make the determination of controlled versus uncontrolled and risk appetite in a bit.

# Prioritization

One of the questions that comes up a lot is given a set of uncontrolled threats, which ones are the most important to deal with.

If we just do a strict bisection and say everything above the line is uncontrolled, that doesn't really help all that much.

<pause>

One way to address this issue is to take a set of points in the space and consider the magnitude of the normal from the risk appetite line to the individual plotted threats.

<pause>

We can then assign an ordering to them based on this magnitude of the normal.

Now, this is a wonderful theoretical approach and although it is possible to apply the math behind it, this is not typically how organizations address this particular problem.

So, let's look at how it’s typically handled.

# Quantization

The typical way that organizations deal with the likelihood and impact information is through quantization.

Quantization is built into most of the tools used to determine the impact and the likelihood.

And also because of this, they are baked into various standards.

One reason for the adoption of quantized rather than continuous values is that people believe it reduces decision-making complexity.

<pause>

So, let's look at how the likelihood and the impact are quantized.

# Likelihood Quantization (CVSS)

Probably the most well-known likelihood ranking system is CVSS.

CVSS is a system which has a score ranging from 0.0 to 10.0 in increments of 0.1.

It's a system based on the ordering of one hundred unique threats into a particular order, but we can quantize it into twelve buckets, as shown here.

CVSS also specifies a rating which quantized to only five levels of interest.

Technically four, but a true zero or none is added so that we can actually talk about a zero.

Those being none, low, medium, high, and critical.

These values map directly to the requirement of ISO/SAE 21434 which has very low, low, medium, and high.

# Likelihood Quantization (CCSS / CMSS)

CVSS works reasonably well for many things. but not all.

Two other systems worth mentioning stem from work done by NIST.

These are the common configuration scoring system and the common misuse scoring system.

They’re intended to complement CVSS.

Where CVSS focuses on the vulnerabilities of a system, CCSS focuses on its configuration. The problem lies not with the design of the system, but rather the metadata tailoring its behavior.

In the case of CMSS, the focus is on system misuse. Both the design and configuration may be correct, but the system is being leveraged for an unintended use case.

Both these scoring systems calculate their rankings using mechanisms similar to those used in CVSS.

Additionally, these ranking systems have the same range of values.

# Impact Quantization (Safety Domain Comparison)

For quantization in the impact space, let's take a look at the safety domain.

There’s ISO 26262, which is the safety standard for automotive.

We also have ISO 14971, the safety standard for medical devices.

ISO/SAE 21434, which specifies a numeric quantization standard for use with automotive cybersecurity.

And finally, FIPS 199, which is the IT standard addressing risk.

<pause>

You'll note that these don't line up perfectly as you can see 26262 has a three-value raking system.

As with the likelihood quantization, a zero or no impact value has been added in order to have a true zero.

< pause>

In order to go from the ISO 26262 space to the ISO/SAE 21434 space, you actually require more resolution than the 26262 numeric value provides.

<pause>

What we have in the impact space is far less granular, when compared with what we see in the likelihood space.

This is vastly different from the continuous space presented earlier.

<pause>

I’ve made the point of adding a true zero value twice now.

Does it really make that much of a difference?

Yes and no.

If we only consider true positive cases where the threat candidate has both a non-zero likelihood and impact, then we would never need a true zero. Mostly.

If we never use the values for anything other than in a text field in a report, then we wouldn’t need a true zero. Mostly.

But, if we need to store the value in a database or perform computations using numeric values or need to assign a value of zero likelihood or no impact, then we absolutely need a true zero.

Quantized Slicing

Now let's see what happens when we combine our slicing diagram from earlier with the concept of quantization.

<pause>

Let's start with our most granular level, that being the CVSS twelve bucket score and six bucket ISO/SAE 21434.

<pause>

Now let's compare that with the CVSS five bucket rank and ISO/SAE 21434 mapping.

There are far fewer buckets to consider.

<pause>

Let's look at how this compares with using CVSS rank against ISO 26262.

<pause>

And finally, here’s CVSS versus FIPS 199.

<pause>

You can get a sense of the impact of quantization, but it’s still a bit abstract at this point.

So, let's look at what happens if we shade in the boxes using our risk appetite line as a basis for slicing.

Quantized Slicing (Shaded)

We’ll apply a simple 51% rule as to whether or not we include a particular bucket, and use the risk appetite line to decide where to slice.

We now can see more clearly how that affects the buckets that we can consider uncontrolled.

As you can see here, the high granularity case is going to give us, the greatest number of discrete items.

<pause>

as we go through the previously mentioned combinations,

<pause>

you'll see that each of the choices made for decompose of the space

<pause>

has an impact on what values you're going to pick up for a given choice of risk appetite.

Prioritization - Continuous

This brings us back to the problem of prioritization.

Let's look once again at the continuous representation.

As covered earlier, we take the magnitude of the normals and use them to determine what the prioritization is going to be.

<pause>

Now, this methodology can be used, but it's not really workable for the quantized form because we would have to end up using the center point of each bucket for any value that it contained.

So that's not really a workable strategy.

Let's look at a way to look at the buckets and come up with this prioritization.

Prioritization – Quantized [CVSS / ISO/SAE 21434]

Here, what we've done is taken the numeric values for the twelve bucket CVSS and six bucket ISO/SAE 21434 and multiplied them together.

<pause>

We can see here is that these are the floor values of the CVSS buckets, which is why the value of 0.1, since it’s the lowest non-zero CVSS value.

<pause>

And then we go from zero through five for the 21434 values.

<pause>

Now we could do all sorts of mathematical gymnastics to ensure that each value is unique.

For instance, you could put weightings on them to make things that are exactly the same be different based on their weighting from the impact space versus the likelihood space.

But in terms of a rough value, these are reasonable.

These can stand in as a good proxy for doing all the hard math of figuring normals to spaces and centers of the buckets.

Multiple Risk Domains

Typically, when we think about risk in safety-critical systems, we're thinking about safety risk, but in reality, there are multiple domains of risk.

<pause>

So, let’s look at four of those domains which are specified in EVITA D2.3 security requirements for automotive onboard network based on dark side scenarios

and talk about each domain and how we’ll address them collectively in the context of threat prioritization.

<pause>

The first domain is safety.

We're considering here the functional safety of the vehicle occupants and the road users.

<pause>

Our second domain is finance.

How do we prevent fraudulent transactions from taking place via the electronic systems within the vehicle and its connected infrastructure?

And how do we prevent the vehicle itself from being stolen?

<pause>

The third domain is privacy.

The focus is on the privacy of both the vehicle users and the IP of the OEM and suppliers.

This gets down to things like the code that's running in the vehicle or the data stores that are being held in the vehicle.

<pause>

And our fourth area is that of operation.

The concern is the operational performance of the vehicle and things which can perturb that operational performance.

<pause>

Together all of these domains of risk have to be considered for any threat candidate.

That's wonderful. Now we have four risk domains instead of just one.

How do we deal with that?

Multi-domain Workflow

In order to accomplish this multi-domain workflow, let's go back to our original workflow diagram, which you can see here where we have the ranking, the risking and the slicing.

Now let's take it and break it up so that we can address those multiple domains.

<pause>

Here, we see the risking and slicing activities are replicated.

Specifically, there's going to be one of these per risk domain.

Additionally, we have a new activity that needs to take place.

That's the consolidation of the risk domains.

This comes between the uncontrolled threats and the issue tracking system.

Let's look at that particular additional step.

Multiple Risk Domain Consolidation

Here we have the consolidation activity.

<pause>

We take the uncontrolled threats from each of the risk domains that have been risked and then sliced using their respective criteria for slicing.

<pause>

Because the appetite for each of these risk domains is going to differ, we’ll consolidate them in such a way as to de-duplicate any items which come from multiple risk domains.

When we de-duplicate, we do so maintaining the traceability back to all contributions for the consolidated threats entered into the issue tracking system.

<pause>

Finally, we generate a consolidated threat report for audit purposes.

Summary

So, to summarize

<pause>

the threat prioritization approach used within the AVCDL allows us to have uniform treatment of all threat sources.

<pause>

It specifies a common data interchange format that being SARIF.

<pause>

The approach works with either continuous or quantized metrics.

<pause>

Finally, the approach works with both single and multiple risk domains.

<pause>

Hopefully it's appreciated that this methodology is far closer to our desired orchestration than it is to a herd of one-man bands.

And implementation of it gives you a lot more leverage dealing with cybersecurity throughout the product development lifecycle.

AVCDL on GitHub

All AVCDL materials, both in source and distribution forms, are available on our GitHub site, as shown here.

Because of the size of the repository, it's recommended that you either clone the repository or download a ZIP archive of it, if you're not familiar with using git.

Instructions for downloading a ZIP archive are linked to on the repository’s front page.

Next Steps

With this training complete, you can proceed to one of the more specific areas.

These are:

* Attack surface analysis
* Threat modeling
* Penetration testing
* Vulnerability identification
* Static analysis
* Dynamic analysis
* Fuzz testing

and

* Secure code review

Some of these may have other prerequisites and so be sure to check as to whether all of the prerequisites have been attended to for any of the areas that you would like to pursue.

References

Here are references to the source material used in the creation of this presentation.

They'll also be included in the video description.

Additionally, this presentation’s source material will be provided on the AVCDL GitHub repository.