# Title

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

# This presentation will cover Attack Surface Analysis.

# 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, and additionally, that you've completed the

# Requirements Taxonomy,

# Security Requirements,

# Secure Design Principles,

# Apply Secure Design and

# Threat Prioritization Plan training.

# This training will cover Attack Surface Analysis.

# Introduction

When the subject of attack surface analysis is mentioned in many people's minds, the metaphor of certain elements of space opera come to mind,

<pause>

but we don't have to go to some a far-off place and time to understand what attack surface analysis is about.

<pause>

This is the USS Gridley DDG 101.

It's a destroyer class vessel and you'll note that it has a lot of technology which allows it to observe to the greater world around it. It also has very visible defensive mechanisms.

There's an ability for it to communicate. There're ingresses and egresses from the vessel and all of this is to be expected.

<pause>

Now, when we think about attack surface, we look at this and we say, “yeah, I can see what there is, and I know what it is that I'm going to focus on if I wish to attack this.”

Now, let's look at a different vessel also from the United States Navy, also a destroyer.

<break>

This is the USS Zumwalt DDG 1000.

This vessel performs the same type of mission as the Gridley.

It also has ingresses and egresses.

It also has defenses systems.

It also has communication systems.

When we look at it, we can see very plainly that its attack surface is dramatically different from that of the Gridley.

<pause>

When we're looking at cyber physical systems and considering the attack surface,

our desire is to be far closer to the Zumwalt then to the Gridley.

We know the consequences of having attack surfaces that are available because it doesn't matter what we want an attacker to focus on,

the attacker will focus on everything and anything,

and their starting point is the attack surface.

<pause>

So, let's consider what it means to have an attack surface,

what things we want to focus on,

and how we're going to a document this information.

# Terminology

In any technical discussion, if you can't agree on terms, the discussion doesn't get very far, very fast.

And so here is the terminology as used within the context of the AVCDL.

This terminology is based on the threat modeling glossary proposed by Steven and Migues

You'll notice that it has two distinct halves.

We're going to concentrate on the left half.

Specifically, let's focus in on that attack surface piece, and look at the relationship it has with the things around it.

We can see that an attack surface

<pause>

and a boundary

<pause>

interact with each other and that

<pause>

an attack

<pause>

interacts with

<pause>

the attack surface.

All this makes sense.

Continuing, a threat

<pause>

Traverses

<pause>

a boundary.

So, when we're talking about a threat, we're talking about some boundary traversal as presented by an attack surface and it becomes an issue when there is a vulnerability.

So, an attack surface

<pause>

which possesses

<pause>

a vulnerability

<pause>

is a defect.

<pause>

It’s key to be able to identify these attack surfaces because that allows us to not have to deal with the right half at all.

If there is no attack surface, there can be no attack.

If there is no attack surface, there can be no threat because there is no traversal of a boundary.

<pause>

Now let's get more concrete with this and start with an anchoring example.

# Simple System – Block View

There's a tendency when using an anchoring example of making it overly complicated.

This is not that type of anchoring example,

nor is it in the typical domain of cyber-physical systems.

This is a web service.

So, can we learn about attack surface analysis from this?

Absolutely.

All the elements we’ll need are there.

We have on the gross scale, the thing which is of interest to protect

and the things that interacts with the thing of interest.

If we look deeper, we have elements within each of those contexts which are interacting with each other.

This is a perfect example.

There is sufficient detail for us to be able to speak about all the different areas we need to cover.

It's relatable in that everyone has experience with these types of systems

and it allows us to bridge various domains without having to get into needless levels of specialization.

Now, let's talk about what we care about when we're dealing with attack surface analysis.

Areas of Interest

When we're performing an attack surface analysis, we are going to focus on three fundamental areas.

Three Ps if you will,

<pause>

The first are ports.

We tend to think about these a lot. These are the holes in the box.

<pause>

The second are protocols.

These are the things that carry the information that go through the holes in the box.

<pause>

And finally, there are the processes.

These are the drivers of that same information that pushes things through the holes in the box.

<pause>

A question may come up in your mind that's

“Wait, this is about holes in the box, right? Why do we care about processes?”

We'll get to why we care about processes in a bit.

And you should think about why processes are important.

If it's not immediately obvious, you need to step back and say,

“How am I considering processes? And what do they mean in the context of attack surface?”

# Physical Ports

Our first, simplest, and most obvious area within the realm of attack surface is that of physical ports.

Here we can see the physical ports on various laptops.

You'll note that the, the vast array of ports

* USB,
* Serial,
* video,
* audio,
* PCMCIA,
* Ethernet,
* and S/PDIF.

Moving on, we have the ODB II connector which will be present in just about every vehicle known to mankind in recent history.

And finally, we have the JTAG interface for those of you who deal with things that occur on the board level.

All of these physical ports are the hole in the box.

Now, there's one more hole in the box that isn't pictured here because there's no point in putting a picture of it.

And that's a radio antenna because if we're communicating via some form of electromagnetic radiation, that's a hole in the box, and we need to take that into consideration.

# Logical Ports

The next type of port that I'd like to examine is the logical port.

These are the ports that software folk tend to think about.

These are technically abstractions, as they only exist as numbers in the data being transported.

But they’re also the domain in which many cybersecurity attacks focus.

Logical ports are encoded as an unsigned 16-bit value which means we have from port number zero to port 65535. Logical ports which are used in shared environments are registered and this registry is managed by the IANA.

<pause>

Here's a portion of the current well-known port list.

Starting with, of course, port zero which is reserved. No one can use it.

And you'll note that we cover two transport protocols here,

<pause>

TCP and UDP.

This is important and these are the two most common transport protocols used. And for that, we have this list.

<pause>

There's a link to the full list in the references.

<pause>

Let’s focus in on port 666, which is registered for use by Doom.

Yes, that Doom, the game.

You'll notice that they're using both TCP and UDP.

So, if we were to look at the traffic and see that we're getting packets from on port 666 using either of those two protocols that we would presume that those must be from Doom because that's what's registered.

And if that's what we intend to have happen, then that's all well and fine.

<pause>

However, you'll notice that there is a recorded unregistered pair being used by MDQS.

So, we have the issue here that there is both known and expected traffic, and also traffic which should not be there.

This is a problem.

Additionally, if we see that we have other types of traffic, then we need to ask if we expected it to be there.

<pause>

It's very important to not complicate matters by randomly utilizing ports that are not registered. To quote the IANA,

“Assigned ports both System and User ports SHOULD NOT be used without or prior to IANA registration.”

Additionally, when an organization needs more than one port, that port multiplexing be applied.

The discussion of a diversity of different types of information needing to be carried for a given registered port is outside the scope of this video.

Now let's move on.

# Common Protocols

In our second area of consideration, we have protocols

<pause>

and here we can see a list of common protocols that you're going to find in any number of systems.

Now, if your favorite protocol isn't up here, then I apologize.

But I made a list, these are the common protocols.

And the thing about these protocols is that we have expectations that when you're doing best practices-based cybersecurity that, where possible you, will be using standard protocols.

If you're using custom protocols, we need to know about those.

And when someone says they’re using a custom protocol, that in and of itself should raise flags because writing protocols is difficult and writing secure protocols is very difficult.

# Operating Systems – one size doesn’t fit all

Our third area of consideration focuses on processes.

Now we can’t talk about processes without first considering the operating system.

So, let's talk for a moment about operating systems.

When we're considering operating systems, one size does not fit all.

Let's look at operating systems, divide them into categories, talk about what makes those categories interesting, and provide examples as appropriate.

<pause>

Our first category is going to be the tiny operating system model.

The common name for this category is bare metal.

Now, there aren't any examples because the code that you have is literally all that exists.

There is no separate operating system.

If you want to accomplish a task, you have to write it.

If you want to talk to ports.

If you want to do anything, it is being done by the single simple application running on this system.

This is the most basic way of approaching dealing with a system, where you are so deeply embedded that you own and control every resource.

There is absolutely nothing between you and the machine.

<pause>

Our second category, we'll call it small, more commonly referred to as embedded in this category.

We are talking about things like FileX, NetX, ThreadX, QPano.

What distinguishes this type of operating system is that it has static linkage to our main executable.

It has a threading model because there can be multiple processes, not necessarily, but there can be and they exist using a co-operative multitasking should that be the case.

We have no user levels.

Since you have these processes and they own the environment, there's no isolation to speak of.

<pause>

At the next level up in sophistication, we have our medium class of operating systems.

Here, we're talking about RTOSs - real-time operating systems.

Some examples are AUTOSAR, QNX, VxWorks.

These can have dynamic linkages.

That is, the operating system is not bound-to or built-into the executable of interest.

They have a more interesting kernel which allows for pre-emptive multitasking.

They have rudimentary and sometimes more sophisticated distinctions between user levels.

Typically, though, either you're the system or you're a user.

They'll have minimal isolation.

<pause>

Finally, we have what most people think of when they're thinking about operating systems, the large scale.

These are PC grade and above. We're talking about operating systems like MacOS or Linux or Windows where you again have dynamic linkages, and pre-emptive multitasking.

Additionally, you have full ACL-based user configuration and full isolation between processes is possible.

So now that we've established that you have these different configurations, let's talk about what makes the operating system, the operating system.

# Operating System Abstraction

One of the reasons that we can have this distinction between sizes of operating system is related to the natural evolution of the computer and software that we run on the computer.

Initially, regardless of how big the computer was, it was running only a single program. That program needed to contain all the different components that it would need to operate.

Eventually, we ended up creating subroutines which were attached to the program that allowed it to do these common functions.

The first of which that becomes interesting is going to be

<pause>

the file system as a separate piece.

This allows us to have some sort of reasoned storage and structured storage.

The next piece that comes into play is going to be

<pause>

the process manager because we fundamentally need a way to start and stop processes.

Once we have processes that are being managed, we added

<pause>

the inter-process communications manager to allow the processes to pass data back and forth between them in a way which wasn't just I write a file; you look at the disk and see if a file is there and then you open the file and get the date out of it.

The last major piece added was

<pause>

the network manager because computers were not initially connected to networks. They were standalone devices.

And this was just as true for when PCs in their early days.

And if we look and considered the previous embedded system example, we talked about FileX, NetX, and ThreadX.

FileX is a filesystem manager, NetX is a network manager, and ThreadX is a process manager.

Those are all separate items. If your system doesn't need process management, you don't include it.

So, you have the capability of custom OS configuration. The term for this is Just enough Operating System (JeOS).

Now let’s talk about the way that processes see the world.

# Process-centric Worldview

In a process-centric worldview, the world looks like

<pause>

a poker chip. At its center is the process.

<pause>

It accesses the four fundamentals that the operating system provides and several more in a modern operating environment; be those

<pause>

crypto engines

<pause>

or serial drivers

<pause>

or database managers

<pause>

or memory managers.

These support elements, provided by an operating system, allow the process to interact with all of

<pause>

the entities that are external to it.

And so, the attack surface here exists between the process and the operating environment and the operating environment and all of the external entities to which the process wishes to interact.

So it's not as simple as saying, we have a serial port on a computer and therefore if we fill it with epoxy, then we're good to go.

Depending where we're considering our attack surface, we're going to have different elements that are going to have to be enumerated and assessed.

Let's look at what that process looks like.

# Attack Surface Analysis Workflow

Here's our overall attack surface analysis workflow within the workflow.

We have three individual activities,

<pause>

model creation,

<pause>

analysis,

<pause>

and threat candidate triaging.

Let's examine each of these activities in detail.

# Model Creation

In the model creation activity,

<pause>

the development SME

<pause>

and the attack surface analysis SME

<pause>

take the system documentation

<pause>

and create the element model.

The element model may also be known as the list of ports and protocols in some domains.

This model is very skeletal. It only shows us what is known and serves as documentation of those ports, protocols, and important processes which exist within the system.

# Analysis

In the analysis activity.

<pause>

The element model is taken by

<pause>

the ASA SME

<pause>

and analyzed. Any deficiencies that are identified are put into

<pause>

a list of threat candidates.

# Excessive Exposure

By deficiency of what we mean in this context, the attack surface analysis is excessive exposure.

We have an attack surface. Are we exposing too much,

<pause>

are exposing ports that we don't need to?

<pause>

Are we exposing protocols that we don't need to?

<pause>

Is there a reason that you would have excess services?

Say you have a web server. Why is your sensor running a web server?

We can go deeper and we can talk about whether there is excessive use

<pause>

of data structures,

<pause>

of accounts,

<pause>

of diagnostics,

<pause>

of remote execution.

<pause>

We can talk about excessive connection lifetime.

Do you have a process that you've got sitting out there that opens a port, establishes a connection via a protocol and then lets it persist.

Now, you could consider things like a diagnostic lifetime where there’s a diagnostic process and it's listening on a port waiting for input to tell it what to do,

or it might be presenting outbound data that all it requires is that someone plug an appropriate connector in and data spews out. A JTAG port could be considered in this case.

<pause>

We have also the concept of features, right?

Do you need a feature that allows you to delete entries in your log files in a system where log files may be the thing that's required for post mortem analysis.

<pause>

Do you have processes running that are going to exert undo memory pressure on the system?

Are all the things in the system essential to its operation?

Triaging

Our last activity in the attack surface analysis workflow is that of triaging

<pause>

Using the threat candidates as inputs

<pause>

the attack surface analysis SME

<pause>

and the development SME

<pause>

review each of the candidates and determine whether or not they are actual valid cases or whether there needs to be

<pause>

an update to the element model to account for the use and necessity of a particular port, protocol, or process.

Those elements which are deemed as excessive exposure, go into

<pause>

the list of triage threat candidates which will make its way

<pause>

to the standard threat ranking system. This is described in the threat prioritization plan.

<pause>

An attack surface analysis report is generated which tells us what the disposition of the system is.

Threat Candidate Information

As discussed in the threat prioritization plan, the AVCDL generally recommends that information exchanged between various parts of the cybersecurity activity pipeline be done using a JSON encoded SARIF file.

<pause>

As a reminder SARIF is the static analysis report interchange format. It’s both highly flexible and supported by numerous applications.

<pause>

In the case of the attack surface analysis.

<pause>

When we look at it in the context of a SARIF element, the context and the run are the common usage within the context of SARIF.

<pause>

Our test is basically a type.

Is this a port?

Is it a protocol?

Is it a process?

<pause>

When we talk about the finding,

<pause>

the location is the particular port, protocol, or process that we're dealing with.

<pause>

And the issue is the excess exposure that that particular port, protocol, or process presents.

Layers

You might have found that during this presentation, that there was a lot of jumping around in terms of the level of abstraction that I was operating at, it went everywhere from physical ports to individual processes.

And that was by design.

The intent of an attack surface analysis is not to start with a huge system and delve down deep into its innards when done properly.

An attack surface analysis is a compositional activity which allows you to take various components and determine what their attack surface is.

If you have a system being provided by a third party, which happens to be a library, then its attack surface is the totality of interfaces it presents. Do they all need to be there?

If you're dealing with a component, then you might say, my attack surface happens to be the physical interfaces that exist as well as the ports and protocols that are being used at the software level.

But there's also a concern as stated earlier that you don't have processes running that don't need to be running.

There's no point in advertising attack surface by saying, “I'm building an extension to my house and I only have this blue tarp because the door hasn't come in yet. Don't pay attention to the blue tarp.”

That doesn't work.

When doing an attack surface analysis, it's important to recognize that this is not a check box activity that there are layers that exist and may need to be explored.

And that simply because you have done an attack surface analysis at one level of abstraction does not mean that that's all that is there.

Keep in mind that within the realm of cybersecurity, one of our principles that we use in order to establish cybersecurity of a system is that of defense-in-depth.

That is layers.

Overall Dependencies

Returning to our diagram on dependencies from the security requirements training.

<pause>

We have the functional requirements

<pause>

and the cybersecurity requirements global catalog

<pause>

and all of the products and systems that are used to hold those products.

Now, let's consider how the attack surface analysis materials fit into that.

Downstream Use

Focusing for a moment just on the cybersecurity elements.

<pause>

All of our element models are aggregated together

<pause>

into an attack surface database.

This database can be used to generate reports that provide a list of the attack surfaces present within the system.

We can then take that database and use it

<pause>

as input to the cybersecurity penetration tests so that we're not simply presenting a system to the penetration testers and saying, “Off you go do a penetration test. You can spend two weeks working on it.”

By having the attack surface database available, we can more effectively utilize the efforts of the penetration testers because they will know where to point their tests.

Verification

Here we can see the AVCDL framework. As you can see,

<pause>

the attack surface analysis takes place

<pause>

in the design phase and is used to inform activities

<pause>

in the implementation phase.

We also need to verify that those deficiencies which were identified have in fact been mitigated.

And so

<pause>

in the verification phase, we do

<pause>

an attack surface analysis review shown here as ASA review.

Now let's look at that workflow.

Attack Surface Analysis Review Workflow

Here's the workflow that we're going to use to do the attack surface analysis review. Again, we have three activities:

<pause>

the model update,

<pause>

the analysis

<pause>

and the mitigation verification.

Let's look at each.

Model Update

In the model update activity,

<pause>

the development SME

<pause>

and the attack surface analysis SME

<pause>

take the as-implemented element documentation

<pause>

and the element model

<pause>

and use them to update the element model.

<pause>

The output of that being the updated element model so that we have a current state of the system.

Analysis

As with our attack surface analysis,

<pause>

the attack surface analysis SME will take

<pause>

the updated element model,

<pause>

perform an analysis, and generate

<pause>

a list of threat candidates.

Mitigation Verification

Finally,

<pause>

the attack surface analysis SME will take

<pause>

the threat candidates generated in the analysis

<pause>

and the list of uncontrolled threats from the initial attack surface analysis

<pause>

and verify that the issues that were shown in the uncontrolled threats have in fact been mitigated.

If they have not, then

<pause>

a list of unmitigated threats is generated and that goes into

<pause>

the threat ranking system.

Otherwise,

<pause>

an updated element model annotation is done so that we can note what the changes were to the model that eliminated these.

And finally, we produce

<pause>

an attack surface analysis report showing the activities that have been done.

Summary

To summarize, when done properly, an attack surface analysis

<pause>

is lightweight.

It's not an activity like a threat model analysis where we're looking at data flows. We are looking at just what our attack surfaces are and documenting them and then reasoning as to whether it's necessary for all of the surface holes to be present.

<pause>

It aids us in our downstream cybersecurity activities, specifically in the activities of the penetration tester.

<pause>

It may be a regulatory requirement to have documentation of known ports and protocols.

<pause>

And lastly, it's important to consider when a layered analysis is necessary.

Further Reading

Here are some documents suggested for more in depth information.

They are

<pause>

the attack surface analysis report,

<pause>

the updated attack surface analysis

<pause>

and the threat prioritization plan.

These are all available in the GitHub repository.

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 other trainings at this level.

These are:

* Threat Modeling
* Penetration Testing

and

* Vulnerability Identification

Additionally, if you’ve completed Secure Coding and its prerequisites you can also proceed to

* Static analysis
* Dynamic analysis
* Fuzz testing

and

* Secure Code Review

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.