# Three Software Innovations that DoD Needs Now

## Table of Contents

Three Software Innovations that DoD Needs Now

Jeff Boleng, Sam Procter, Nathan VanHoudnos, Lena Pons, Robert Schiela

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Carnegie Mellon University
Software Engineering Institute

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

**001 Facilitator: And hello, from the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania. We welcome you to Virtual SEI. Our presentation today's a panel discussion on "Three Software Innovations that DoD Needs Now." My name is Shane McGraw. I'll be your facilitator for the discussion, handling audience questions. So feel free to type those questions in at any time on whatever platform you are joining with us today.

So now I'd like to introduce our moderator for today. It's going to be SEI CTO Jeff Boleng. Jeff, I'm going to turn it over to you to introduce our panelists.

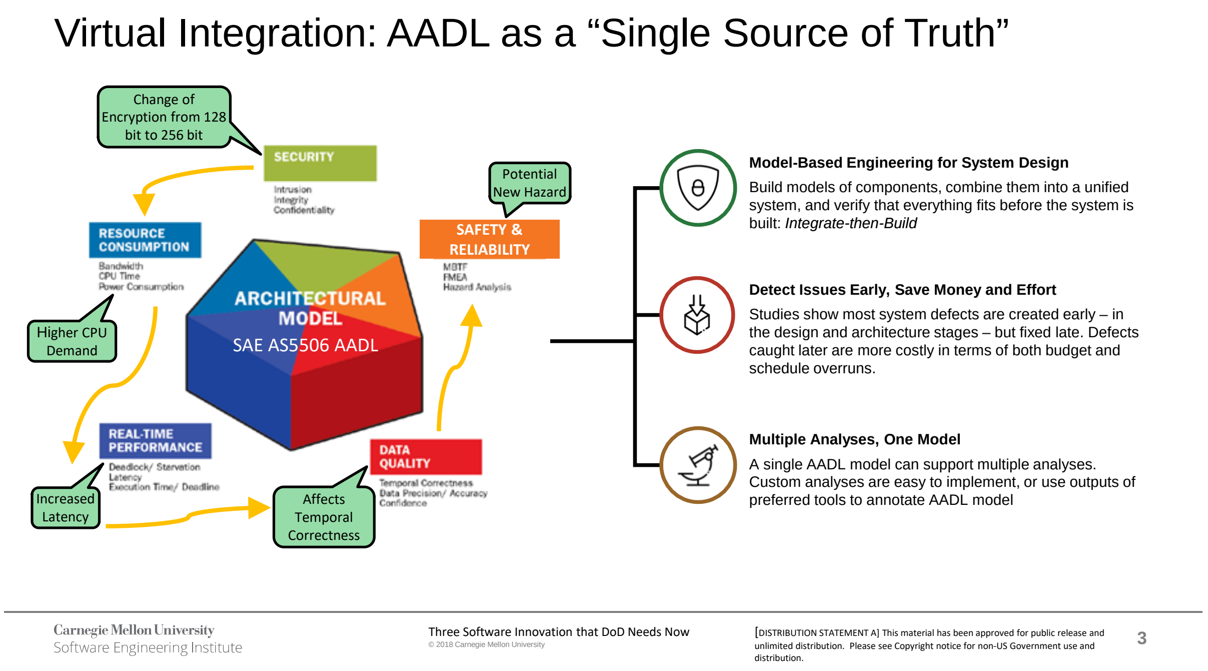
Moderator: Welcome. Thanks, Shane. Thanks for joining us today. We're going to--we were trying to be a little provocative with the title, so hopefully we deliver with three software innovations that DoD can--needs and can use now.  I'm going to start with introducing--I'll start on my left here with Bob Schiela.  Bob's the Tech Manager for our Secure Coding Division, and so one of the things he'll be talking about is automated code analysis, which is one of the three.  Next one over is Sam Procter. Sam's an architecture researcher, and he'll be focusing on something we call virtual integration, which we'll describe in more detail there.  Next to Bob is Lena Pons.  She's one of our machine learning research scientists in the CERT, our cybersecurity division, and next to her is Nathan VanHoudnos, which I got the name right.

Speaker 1: Yeah, good job.

Moderator: Yeah.  Thanks.  I've been coached, and he's also a machine learning research scientist, that we'll have them cover essentially data science is the last of the three innovations I think Do--needs now and needs more of, and so we took a little bit of a liberal license with that one.

So I'm going to start with Sam Procter talking about virtual integration, and he'll give an overview of kind of what we mean with virtual integration, where it fits, and sort of why it's important.

## Virtual Integration: AADL as a “Single Source of Truth”



Virtual Integration: AADL as a “Single Source of Truth”

**Model-Based Engineering for System Design**
Build models of components, combine them into a unified system, and verify that everything fits before the system is built: *Integrate-then-Build*

**Detect Issues Early, Save Money and Effort**
Studies show most system defects are created early – in the design and architecture stages – but fixed late. Defects caught later are more costly in terms of both budget and schedule overruns.

**Multiple Analyses, One Model**
A single AADL model can support multiple analyses. Custom analyses are easy to implement, or use outputs of preferred tools to annotate AADL model

Carnegie Mellon University
Software Engineering Institute

Three Software Innovation that DoD Needs Now
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

3

**003 Speaker 2: Yeah. So model-based engineering is one of sort of the classic ideas in engineering. It's pretty well accepted and it's used in a number of domains. So those domains really range from aeronautics, if you're building a wing. You may be testing models of that wing, either virtually or in some sort of computer system. Same thing with, you know, flames, rocket engines, all over the place. Model-based engineering is well accepted.

So virtual integration is really the application of model-based engineering to system engineering, where you build a model of the components of your system and then you plug them together. So this is sort of a radical departure from the traditional system engineering methodology of building the components and hoping that when you plug them all together, everything fits.

So like a lot of things, a lot of project design and system acquisition both within the DoD and outside of it, cost is a driving factor behind this research.  A lot of, in fact, the majority of system errors are introduced when people are architecting their system or designing their system, and that's also when it's cheapest to fix those problems.  But it gets more expensive and more likely that problems will be detected later on in the development lifecycle. So that can mean that when it comes time to plug your system components together, that you're discovering errors then or even once the system has been deployed, and at this point these errors that were introduced early in the system development lifecycle can be very expensive to fix.

So one of the main goals is to bring these costs down by integrating the system before you build it.  This is the integrate-then-build approach that we advocate here at the SEI. Then a final benefit of AADL, the architecture analysis and design language that we use to study virtual integration here at the SEI, is that it supports a number of different analyses.  So one, a critical weakness with a lot of model-based engineering tools, is that they really only address one area.  So you build a model of your system or of your component or whatever and it supports one analysis that then runs on that model, and if you want to do a different analysis, you have to build a different model.

So in the best case, this leads to a lot of repeated work, but because systems, as they get built, inevitably have requirements change or have to be updated in some way, you then have to update all of your models, and keeping those in sync is laborious and error-prone.  So AADL is a rich language that supports a number of different analyses, and you can have this one model that we like to call the single source of truth for your system.  So whether you're a system integrator looking at how everything plugs together, you're a security person looking at how you can defend against possible attacks, or a safety guy, all of these analyses can be run on one model, and so real quickly I want to walk through a cool scenario that can come about with this single source of truth model.

So your security people might decide that they need to change the key size of your encryption from 128 to 256 bit.  Now, in the status quo, a lot of times this means you're just going to change key size and hope for the best, but with a single integrated model you can see the impacts of that on the resource consumption.  So with a larger key you may need more CPU time, which may increase the latency through some critical paths in your system, which can affect the temporal correctness of your data in a real-time system, and this could even mean a potential safety problem, and these issues are going to be a lot easier to discover if you, all your analysts and all your different design aspects, are relying on one model as the single source of truth.

Moderator: So just a quick question before we move to the next topic.  So with AADL on some of the modeling tools that we've got, what's the--the Eclipse plugin we have.

Speaker 2: So we have a--we have a tool environment called OSATE, the Open Source Architecture Tool Environment, built on top of Eclipse.

Moderator: Yeah, there's an Eclipse plugin.

Speaker 2: Yeah.

Moderator: But you can model down to such a fine level of granularity that you can actually simulate these changes in the system.

Speaker 2: Yeah.  Absolutely.  So AADL supports modeling at a number of levels of abstraction.  So when you first start out, you would say, "Oh, I have a device here.  I have an entire system here," and you would just sort of define the inputs and outputs. But then exactly as you're saying, you can model at successively lower levels of abstraction, you can refine that model down to really the low-level hardware, the processors, the memory banks, and even the buses that are used for the network communication.

Moderator: Yeah, and the fine-level timing constraints for--yeah.

Speaker 2: The fine-level timing constraints. Absolutely.

Moderator: Embedded real-time, safety critical.

Speaker 2: Yeah, yeah. The power consumption. All sorts of things.

Moderator: Okay. Thanks, Sam. We're going to--

Speaker 2: Yeah.

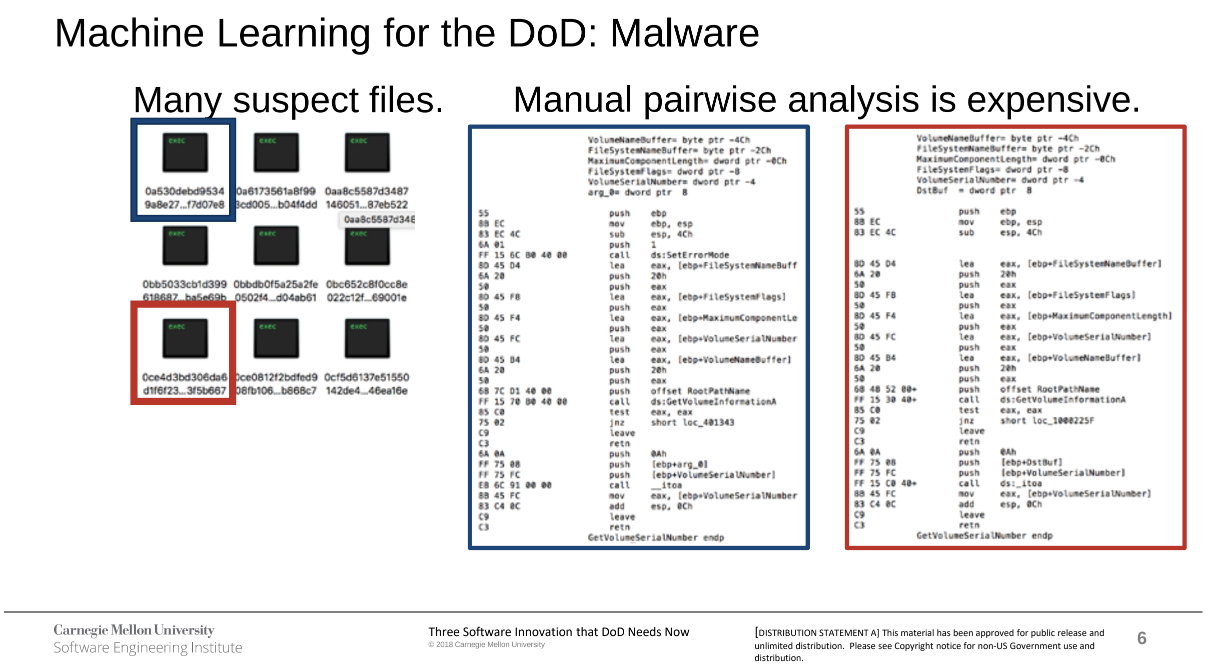
Moderator: --move--we're going to bounce around a little bit. We'll get an overview of the three cores, three or four core things we're talking about today, and then we'll do a little more deep dive and I'll remind everybody that we've got audience questions and Shane's going to be watching for audience questions to interject those too, so I encourage people to do that.

Next we're going to go to Nathan, and he's going to give an overview of data science and not just data science but how we're applying data science here at SEI and in some other contexts, so--

Speaker 1: Thank you, Jeff. So when we think about data science at the SEI, we don't think of like a single person that is a data scientist. We think about data science as more of a set of three roles. The first role is the statistical machine learning role, which is what often people think about when they think about data science, but another role that is sometimes overlooked is the subject matter expert role, right.  This is the person that has the data, this is the person that has the questions that we want to answer, and so not only do we need to have the expertise of the subject matter expert, we also need the expertise of the data scientist.  But there's a third role as well, which is the engineer.  This is the person who makes sure that the data that we get from the subject matter expert can be understood in a machine-learning framework or can even be run at scale, for example, at internet scale or very large scales.

One of the ways that we have applied this methodology at the SEI is with a product that we call malfaces.

## Machine Learning for the DoD: Malware



### Machine Learning for the DoD: Malware

Many suspect files.

0a530debd9534 9a8e27...f7d07e8
0a6173561a8f99 3cd005...b04f4dd
0aa8c5587d3487 146051...87eb522
0aa8c5587d348
0bb5033cb1d399 618687...ba5e69b
0bbdb0f5a25a2fe 0502f4...d04ab61
0bc652c8f0cc8e 022c12f...69001e
0ce4d3bd306da6 d1f6f23...3f5b667
0ce0812f2bdfed9 08fb106...b868c7
0cf5d6137e51550 142de4...46ea16e

Manual pairwise analysis is expensive.

```
                VolumeNameBuffer= byte ptr -4Ch
                FileSystemNameBuffer= byte ptr -2Ch
                MaximumComponentLength= dword ptr -0Ch
                FileSystemFlags= dword ptr -8
                VolumeSerialNumber= dword ptr -4
                arg_0= dword ptr  8

55                    push    ebp
8B EC                 mov     ebp, esp
83 EC 4C              sub     esp, 4Ch
6A 01                 push    1
FF 15 6C B0 40 00     call    ds:SetErrorMode
8D 45 D4              lea     eax, [ebp+FileSystemNameBuff
6A 20                 push    20h
50                    push    eax
8D 45 F8              lea     eax, [ebp+FileSystemFlags]
50                    push    eax
8D 45 F4              lea     eax, [ebp+MaximumComponentLe
50                    push    eax
8D 45 FC              lea     eax, [ebp+VolumeSerialNumber
50                    push    eax
8D 45 B4              lea     eax, [ebp+VolumeNameBuffer]
6A 20                 push    20h
50                    push    eax
68 7C D1 40 00        push    offset RootPathName
FF 15 70 B0 40 00     call    ds:GetVolumeInformationA
85 C0                 test    eax, eax
75 02                 jnz     short loc_401343
C9                    leave
C3                    retn
6A 0A                 push    0Ah
FF 75 08              push    [ebp+arg_0]
FF 75 FC              push    [ebp+VolumeSerialNumber]
E8 6C 91 00 00        call    __itoa
8B 45 FC              mov     eax, [ebp+VolumeSerialNumber
83 C4 0C              add     esp, 0Ch
C9                    leave
C3                    retn
                GetVolumeSerialNumber endp
```

```
            VolumeNameBuffer= byte ptr -4Ch
            FileSystemNameBuffer= byte ptr -2Ch
            MaximumComponentLength= dword ptr -0Ch
            FileSystemFlags= dword ptr -8
            VolumeSerialNumber= dword ptr -4
            DstBuf  = dword ptr  8

55                push    ebp
8B EC             mov     ebp, esp
83 EC 4C          sub     esp, 4Ch


8D 45 D4          lea     eax, [ebp+FileSystemNameBuffer]
6A 20             push    20h
50                push    eax
8D 45 F8          lea     eax, [ebp+FileSystemFlags]
50                push    eax
8D 45 F4          lea     eax, [ebp+MaximumComponentLength]
50                push    eax
8D 45 FC          lea     eax, [ebp+VolumeSerialNumber]
50                push    eax
8D 45 B4          lea     eax, [ebp+VolumeNameBuffer]
6A 20             push    20h
50                push    eax
68 48 52 00+      push    offset RootPathName
FF 15 30 40+      call    ds:GetVolumeInformationA
85 C0             test    eax, eax
75 02             jnz     short loc_1000225F
C9                leave
C3                retn
6A 0A             push    0Ah
FF 75 08          push    [ebp+DstBuf]
FF 75 FC          push    [ebp+VolumeSerialNumber]
FF 15 C0 40+      call    ds:_itoa
8B 45 FC          mov     eax, [ebp+VolumeSerialNumber]
83 C4 0C          add     esp, 0Ch
C9                leave
C3                retn
            GetVolumeSerialNumber endp
```

Carnegie Mellon University
Software Engineering Institute

Three Software Innovation that DoD Needs Now
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

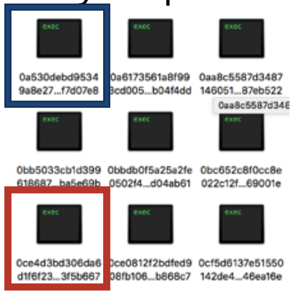
6

**006 And the problem that we're trying to solve here is let's say that you're a malware reverse engineer and what you're given is you're given a pile of suspect files and your task is to figure out which of these files are related to each other. In sort of the traditional ways of doing this, what you often are reduced to is opening up a file in a disassembler, looking at all of that disassembled language, and then opening up another file and comparing them side by side, and this manual pairwise analysis is very expensive to do. So what if instead we could use those three different roles and give ourselves something better to work with? So this is what we did.

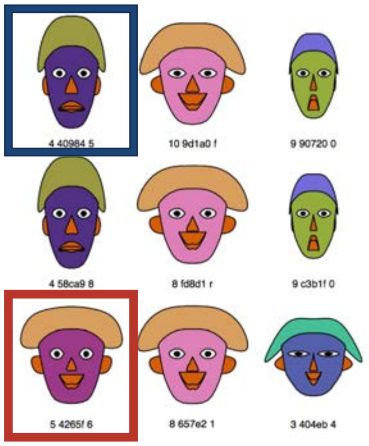
## Machine Learning for the DoD: Malware

### Machine Learning for the DoD: Malware

Many suspect files.


exec
0a530debd9534 9a8e27...f7d07e8
0a6173561a8f99 3cd005...b04f4dd
0aa8c5587d3487 146051...87eb522
0aa8c5587d348
0bb5033cb1d399 618687...ba5e69b
0bbdb0f5a25a2fe 0502f4...d04ab61
0bc652c8f0cc8e 022c12f...69001e
0ce4d3bd306da6 d1f6f23...3f5b667
0ce0812f2bdfed9 08fb106...b868c7
0cf5d6137e51550 142de4...46ea16e

Statistical visualization lowers costs.



**007 Instead of relying on the subject matter expert to do their analysis, we had an engineer take those suspect files and break them down into a way that a machine learning algorithm could understand, and then we had someone like myself come along and perform different similarity comparisons and make a statistical visualization so that now when you just sort of stare at this thing on the screen, you can see that those suspect files that appear on your operating system, that first one and that third one are actually relatively similar to each other because we've expressed them in a way that you can immediately see them as faces. And by using these sort of key components, we have the subject matter expert that says, "Oh, yeah. Those two files are actually quite similar to each other," so we can check that. We have the engineer that can make sure that we can break this down at scale, and then we have someone like myself, who's a machine learning research scientist, to sort of tie it all together.

Moderator: So it sort if, when I first was exposed to this, I was like, "What are we making cartoon faces for?" but--so I just wanted to highlight that.

Speaker 1: Sure.

Moderator: That this is a thing called turn-on faces, which allow you to actually express multidimensional data in two dimensions with different size, shape, colors of faces, right? Is that reasonably accurate?

Speaker 1: That is the idea.

Moderator: Yeah, okay. And then you--one of the things I think I know you used in the background was principle coded analysis? And--

Speaker 1: Yes.

Moderator: --what were--is that one of the primary techniques? Did you use some others?

Speaker 1: So to dive into the details a little bit, what happens is we take the binary file and then we disassemble it so that now what we're left with is a representation of the file in assembly language, and then what we do is we convert that assembly language file into essentially a--trying to think of the right level of detail for this audience. Because I don't want to go in, too

into the weeds. Basically we take
that assembly language and we
summarize it so that the words in the
assembly language now become a
series of keys, and the question is,
"Oh, this file has these keys and this
file has these keys," and then we look
at that matrix and break it down in
something, using a technique called
principle components analysis, that will
then allow us to say, "Okay. Well, this
file needs big eyes and this file needs
a small nose," and those sorts of things.

Moderator: Right. So you--they
don't have to be exact matches.
They can--this really does pull out the
similarities in the binary files so that
you can know, even when those
small changes happen, you can still
pull out the similarities.

Speaker 1: That is correct.

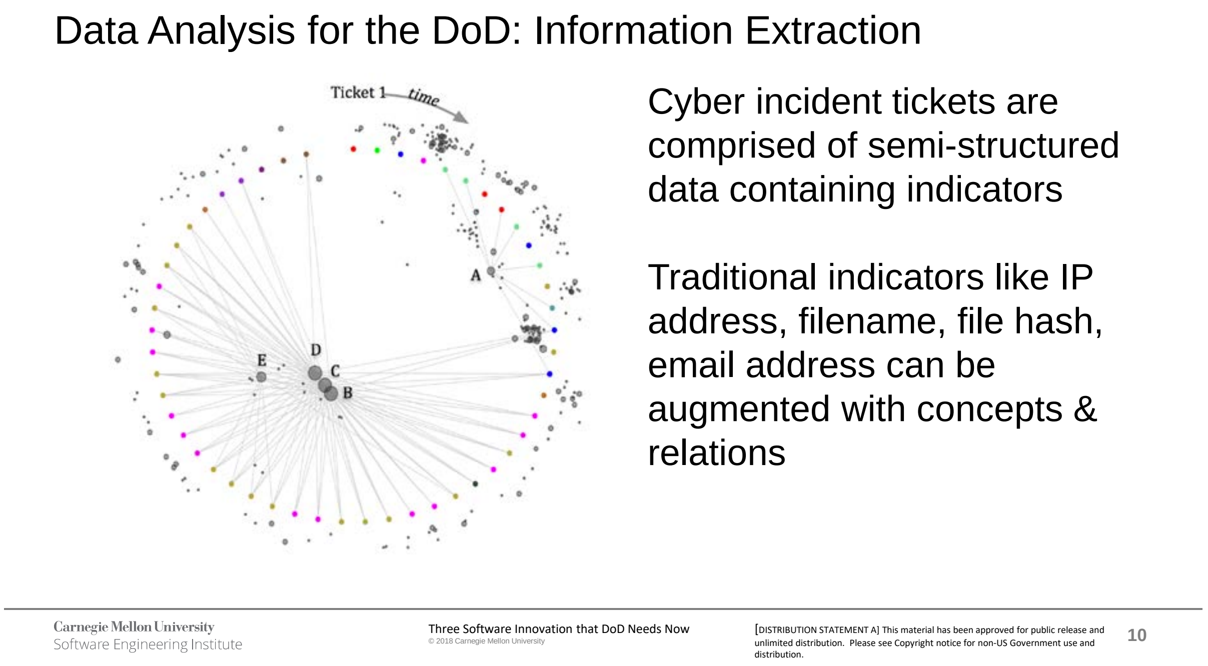
Moderator: Awesome. Yeah. So
much better than signature-based.

Speaker 1: Yeah.

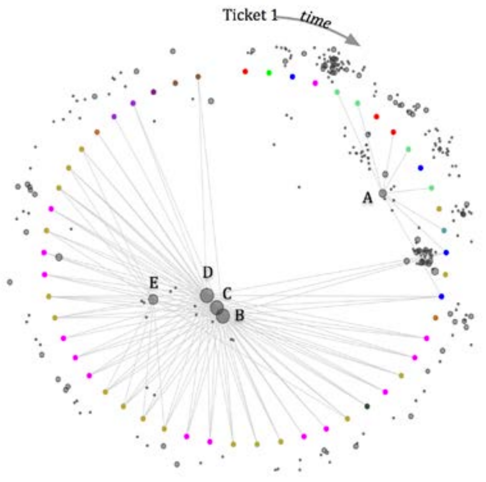
Moderator: Okay. Now make sure I
get my order right. We're going to
go to Lena. Lena's going to talk
about, I don't want to spoil it, but
one of the problems that we have is,
I make the argument, we're
generating more data than we can
consume at this point, and so we
need smart ways to try to handle that
data at creation time, either through
labeling or metadata tagging or some
of those things. So I'm going to turn
it over to Lena to talk a little bit about
that, give an introduction there.

Speaker 3: Sure.  So I'm also a machine learning research scientist, but I come into the data science field from a slightly different background. My background is in information extraction, information retrieval and natural language processing.  So in these techniques, in these ways, I think about the data that the DoD is collecting and particularly in a cybersecurity context in terms of what it contains and what we can mine out of it, what we can identify and how we can build intelligence, and so one of the ways that we do that is through information extraction techniques.

## Data Analysis for the DoD: Information Extraction



Data Analysis for the DoD: Information Extraction



Cyber incident tickets are comprised of semi-structured data containing indicators

Traditional indicators like IP address, filename, file hash, email address can be augmented with concepts & relations

Carnegie Mellon University
Software Engineering Institute

Three Software Innovation that DoD Needs Now
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.  Please see Copyright notice for non-US Government use and distribution.

10

**010 So broadly, information extraction is the process of taking a set of typically we think of it as documents.

But it could be any data set,
and identifying the pieces of
information that are distinguishing an
important for gaining knowledge and
insight into that data set.

So my previous experience was
working in the biomedical research
space, which has a very developed
system of logical language
representations to support things like
metadata, like automated metadata
generation, and cataloguing these,
these large sources of biomedical
research and also clinical data.

So one analogous case that I see
coming from biomedical space into
the cybersecurity space is for
electronic medical records you have
what you could think of as sort of
semi-structured documents which
contain an amalgamation of, like, key
value pair data that has stuff about,
like, "Here are, you know, patient
data," or in the case of cybersecurity
it could be like a tracking number. It
could be an incident type. It could
be some specific sort of key value, a
specific malware signature. These
would be sort of your structured data
types within that data collection, but
then you also have a description
about the incident that says, "At this
time we noticed this behavior.
There's--intrusion detection system
sent us a signal. We did these pieces
of investigation and we were able to
discern this information."

When you take a collection of these
documents, then you start to have
the basis for doing some intelligence

and some pattern recognition and some tracking of actors like across a series of incidents, and so what we have done here is build an information extraction system that will combine extracting from the structured and the pros portions of an incident ticket, and then compute similarities between incidents to start to develop clusters, to say, "Here are incidents that share a lot of similar characteristics."

The advantages to that are that you can identify types of patterns of behavior, which can help you to build intrusion detection methods that will identify similar types of attack paths, but also you can identify incidents that occur in time sequence and identify an attack as it moves through a network of people who may share some data or some similar network characteristics or are just people who are connected to each other in like a broader professional or social network, and so that kind of intelligence is tremendously valuable for getting ahead of the cyber incident timeline.

Moderator: So I was at a workshop just on Wednesday about applying data science and machine learning to the logistics enterprise in the DoD and one of the--so I have couple--then I'm going to play the naove role on purpose a little bit. So some of what I saw was, "Well, we've got all this data. It's--we've got tons of data. We dump it all, all this disparately formatted data into a big data lake." That's the new--right?

That people love that phrase now, and I am just going to pour some ML on there and it's going to answer all my questions. Can you comment on the accuracy of sort of that thinking there?

Speaker 3: Right. Right. So information retrieval is kind of, is kind of fishing in the lake, right? Like, you are trying to, you know, you have a big lake that's full of things and it's hard to know what they are if you don't have any kind of metadata to help you sort of identify, like, where the things that you're looking for are, and so information extraction, information retrieval, information extractions is the process of taking a data structure and mining out the specific pieces of information that you might want to store, and information retrieval is the process of saying, "Within the lake, you know, I would like to find all the fish that are trout." Like--

Moderator: What's relevant to my question of interest?

Speaker 3: Yeah, what's relevant--what's relevant to me, my question of interest, and identifying the--not just the things that you might find through a traditional Boolean query, but actually, like, building a logical hierarchical representation of the relationships between pieces of, like, structures, right. So, like, again, I frequently think of it in terms of documents, like, within your sort of like larger data set, and those kinds of information retrieval questions are very important when you start to think about, "How do I manage a data store like this?"  You know, because just collecting and storing all of these, like, large amounts of data can be very expensive and also it can bog down an analyst trying to make sense of what's in there and what's relevant and what's going to be important and timely for answering the type of question that they're answering, and so yeah.  Those kinds of information retrieval techniques are really critical for identifying, "What do I need to store?  How do I need to store it, and how can I quickly find the things that I need to answer a particular question of interest?"

Moderator: Right.  And so I see--tell me if I'm going to describe this properly.  I really do see this, you've got a subject matter expert with one of the things they need to know is, "What do I need to know that I don't know now?"  Or they need to know the questions they want to answer from their data.  Either they want to answer it faster, more accurately, or actually answer some questions that they don't have answers to now, but it seems like it's very much an iterative process of I propose a question to a data scientist and I work with a data scientist and I say, "Hey, can my data answer this question today?" and you--and then maybe you figure out how to do that, but then we iterate again and say, "Oh.  Well, that's interesting.  Now that I know that, I want to know this."

Speaker 3: Yeah.  Yeah.  So, I mean, there's actually a process within information retrieval called question analysis, where you identify whether the question that you're asking is actually giving you back the answer that you're looking for and expanding out, and so one technique in question analysis is you start with the question that somebody asked and then you ask sort of more probing questions to say--to narrow things down and also to identify, like, what path the question is taking for the end user, and one of the things that you can do there is what's called query expansion, right?  So you've expressed a question one way but there could be two or three or many different ways to pose that same question because of sort of linguistic ambiguity and--

Moderator: Well, a nuance.

Speaker 3: --and, you know, synonymy.  Like, you know, people use different words that mean the same thing to express the same concept, and so--and so this is where having not enough logical language representation within sort of like the cybersecurity and software realm really hurts our ability to leverage some of these techniques like query expansion.  It's very difficult to do automatic query expansion if you don't have, like, a computer readable set of linguistically similar representations of the same question, and so it's much more laborious to do that type of question analysis.

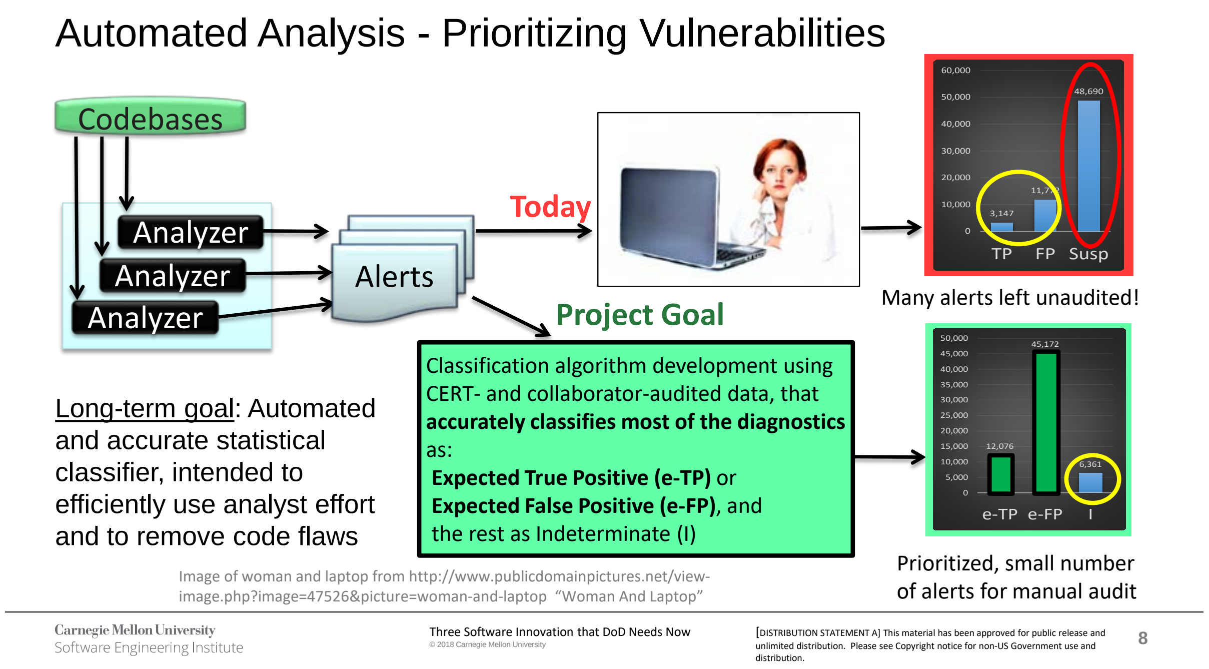
Moderator: Yeah, thanks.  So we're going to move to our final panelist right now, Bob Schiela, and he's going to talk about automated code analysis.  A little different, but we still, there's a big application of some data science machine learning in the automated code analysis nowadays as well, so--

Speaker 4: Yeah, absolutely, absolutely, so--but I think to talk about where we are today and where we're going, it might be worth a minute to talk about where we've been, and looks like I have 40 minutes, so I'll try and finish.

Moderator: Sure, yeah, yeah.  The rest of the time's yours, Bob.

Speaker 4: Within the 40 minutes, so thanks for everyone being very quick.  So about 15 years ago, CERT started identifying common defect types in source code and trying to identify and codify rules to avoid defects that lead to security vulnerabilities.  About 10 years ago, we developed the source code analysis lab, which the purpose was to help organizations test their source code for conformance to this, the CERT standards, the secure coding standards.  So.

## Automated Analysis - Prioritizing Vulnerabilities



Automated Analysis - Prioritizing Vulnerabilities

Many alerts left unaudited!

Classification algorithm development using CERT- and collaborator-audited data, that **accurately classifies most of the diagnostics** as:
**Expected True Positive (e-TP)** or
**Expected False Positive (e-FP)**, and
the rest as Indeterminate (I)

Long-term goal: Automated and accurate statistical classifier, intended to efficiently use analyst effort and to remove code flaws

Prioritized, small number of alerts for manual audit

Image of woman and laptop from http://www.publicdomainpictures.net/view-image.php?image=47526&picture=woman-and-laptop "Woman And Laptop"

Carnegie Mellon University
Software Engineering Institute

Three Software Innovation that DoD Needs Now
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

8

**008 That was about 10 years ago we developed this, and the way that it worked was that it used static analyzers on source code to try and find these problems and we learned through our own efforts that using multiple static analyzers actually made the tools better and the outcomes better, so--

Moderator: So just to interject a little bit.

Speaker 4: Sure.

Moderator: The space here is not like we're hocking our static code analyzers. It's there's a--

Speaker 4: That's actually what I'm going--

Moderator: --really rich commercial.

Speaker 4: That's exactly right.

Moderator: Okay.

Speaker 4: So that's--I'm going back 10 years.

Moderator: Right, okay.

Speaker 4: So this was 10 years ago. So this is not the innovations of today.

Moderator: This is a long time ago.

Speaker 4: Right.

Moderator: In computer.

Speaker 4: So I'm going to get to the innovations of today. So 10 years ago we developed this source code analysis lab and we've been adding features, functionalities, and trying to improve it. So I'll get to where we are today with it. As you said, so 10 years ago we were using static analyzers, some commercial and some open-source, and in the last 10 years the area of static analysis has grown and improved greatly on the commercial area. There are now also aggregator and correlation tools to try and help identify problems and so, you know, that's where a commercial and where a state of practice is now, at least with regard to tools. From a DoD point of view in innovation, what the research shows though from adopting these is that there are still many programs that are not using static analysis, and so now we're into, "Why?"

Moderator: Right.

Speaker 4: All right. "Why today?" Even with the tools and how, you know, the performance levels that these tools are available, why are the DoD and DoD contractors not using it? And there's a few reasons. One reason that some of our teams here at CERT are researching, are integrating the toolsets into the development stack and so into the IDEs and the whole development and testing. You know, there's a lot of advancement and continuous development and so trying to integrate these tools as part of that, that process, is one area of research that we're doing.

Another area of research that our team is specifically focusing on is the problem of false positives, and so one of the issues with static analysis is that it can create lots of alerts, and going through the alerts is a large manual process and it turns out that many of them, many of the alerts, are actual problems that might lead to a security issue, and so here looking at this diagram that we have here, the area today is that organizations can take their code bases, put them through analyzers and they get lots of alerts, and then those alerts have to manually be looked at and you see the red box on the left, which is the arrow--sorry--the yellow circle is basically which alerts get manual inspection, and you can see it's a very small percentage, and the red are suspicious, and they're left that way largely because the analyst just ran out of time and couldn't get to them.

So what our project, one of the projects that we're working on now, is to improve this, and we're improving it by I think your term was spilling machine learning on top or pouring machine learning--

Moderator: Sprinkle.  We sprinkle. We're seasoning it.  We're seasoning with the...yeah.

Speaker 4: --on top of it.  However, we are working with data scientists to help us identify and interpret the information and categorize it correctly so that it means something. But the goal is to use machine learning classifiers with the data that we've collected to try and predict whether these alerts are true positives or false positives without any manual inspection at all, and the goal is that if 80 percent or 90 percent of them can be predicted to be true positive or false positive with high confidence, then there's a small number of alerts that need to be inspected by a person, and that's shown by the small yellow circle in the bottom right.  And so that's one area of work that we're trying to improve this is trying to reduce the amount of work and manual labor that it takes to improve the security or software through source code analysis.

Moderator: So that research has been ongoing for a while, a couple years, and we've already got some positive results there, right?

Speaker 4: It has.  We've published some results as well.  I think at the end of this we have some links to some of those reports.  Just in the time that we had here, I just wanted to talk a little bit--

Moderator: Sure, absolutely, yeah.

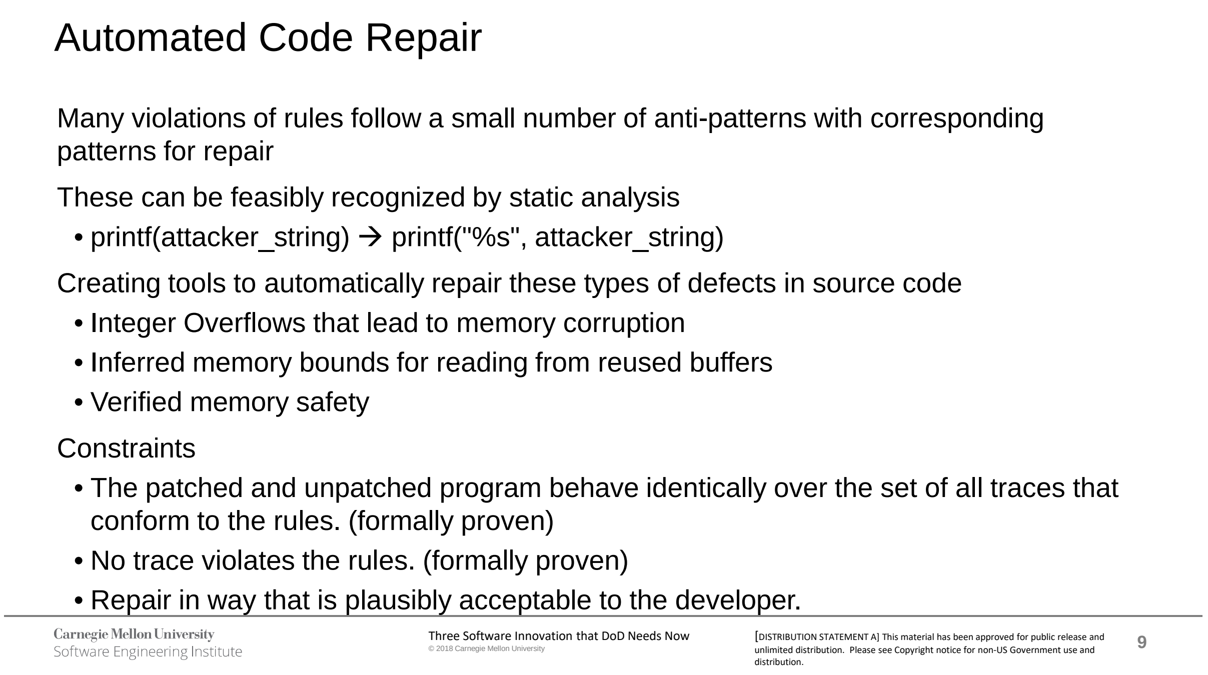
Speaker 4: --and give some teasers on the work.  But we do have some references to show some of the results that we have.  It is also still though ongoing.  It's not complete.

Moderator: Absolutely.  But we can always get better.

Speaker 4: But we can get better, and we're always looking for organizations that are interested to help us with data.  One of the early findings that we had was that finding data was a challenge and in particular finding data with the correct components, attributes, to help us define the right factors or features to help with the machine learning was a challenge, and because lots of people identify or audit their code with using various terms and terminology and that it's very subjective.

So that's one area of work.  Now, this helps with what, you know, I noted here as automated analysis, but so future work that we're doing that will hopefully help even more in the future is automated code repair.

## Automated Code Repair



### Automated Code Repair

Many violations of rules follow a small number of anti-patterns with corresponding patterns for repair

These can be feasibly recognized by static analysis

- printf(attacker_string) → printf("%s", attacker_string)

Creating tools to automatically repair these types of defects in source code

- Integer Overflows that lead to memory corruption
- Inferred memory bounds for reading from reused buffers
- Verified memory safety

Constraints

- The patched and unpatched program behave identically over the set of all traces that conform to the rules. (formally proven)
- No trace violates the rules. (formally proven)
- Repair in way that is plausibly acceptable to the developer.

Carnegie Mellon University
Software Engineering Institute

Three Software Innovation that DoD Needs Now
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

9

**009 So that's kind of the next step as opposed to just minimizing the amount of labor for inspection of these alerts. But what if we could just remove the alerts and address them entirely? That's what we're working on now. So the idea is that many violations to the secure coding roles follow specific anti-patterns and that--so they're easy to find and we know what the correct code should look like, and so here's an example of a printf with just an input parameter where it should be--so that's, that is vulnerable to injection attacks. It's not using, in particular, format string attacks, and so the correct would be using the format string identifier present_s to help make sure that a string doesn't have any format string vulnerabilities within it.

So we know it's easy to find the error and it's easy to fi--or to identify what the fix should be. So we're looking at different types of errors that are in those categories. This is also work that has been going on for a couple years, and so we have some results with regard to integer overflows, identifying and automatically repairing integer overflows that lead to memory corruption and particular buffer overflows, as well as inferred memory bounds, and so this is largely for reused buffers where you might write data into the same buffer multiple times and if you don't clear the previous data that was there, you might give access to someone data that was sensitive to a previous user, and so we try and identify when that vulnerability or weakness is there and to fix it automatically, and right now we have a multi-air project working on verified memory safety.

And so we are working and so I have here some of the constraints with regard to, you know, proving that the patch doesn't break anything. Obviously, we'll want to make sure it doesn't break anything and that it doesn't violate any of the other secure coding rules. We don't want to make not just break functionality but also not add any new software or security vulnerabilities to the code, and so one area in particular for DoD need, that we see this as we're going forward, is the legacy code that's out there for the DoD. Even though we showed on the previous slide for automated analysis, that we're trying to improve that, a lot of the legacy code that's out there, nobody's going to start analyzing that code and looking for vulnerabilities or defects in the source code, and so this might be a way--

Moderator: Except the bad guys.

Speaker 4: Except the bad guys. Yes, exactly.  If they can get their hands on the source code or they might be attacking it, you know, through--

Moderator: Right.  Fuzzing it or something else.  Yeah.

Speaker 4: --fuzzing and other dynamic analysis techniques.  But the idea is that without much manual interaction at all, we might be able to remove a whole class or multiple classes of vulnerabilities in legacy code without using manual labor to do that.

Moderator: Completely automatically, yeah.

Speaker 4: That is the goal.

Moderator: Yeah.

Speaker 4: Yes.
~~~
Moderator: And I think we're, we've already shown, that there's techniques that have been successful for some classes or errors already.

Speaker 4: That's right.  Yeah.

Moderator: Yeah.
~~~

Speaker 4: So in particular, like I mentioned, the integer overflow and the inferred memory bounds are past projects we have published and some tools available for collaborators to try.  It's still in research phase, meaning that it's not, you know, a production quality type of tool, but it does work and it does identify and correct code.

Moderator: Well, at the end of the day we want to get these things in the compiler tool change, right?

Speaker 4: That's right.  That's right.

Moderator: So you would get nice compilator warnings or even set the right compiler flag and it would fix it for you.

Speaker 4: Yes, yes.

Moderator: And I think one of the other, just to point out to the audience, one of the other emphasis for this team is that any automatic code repair that happens remains--maintains human readability and human maintainability as well.

Speaker 4: Yeah.  That's actually a great point.  So there are currently automated code repair tools out there.  Most of them either repair at the binary level or they generate code that can be in the source code that is easily machine readable but hard for humans to read with regard to specific execution statements and such, and so a goal of this is to

maintain readability and
maintainability with the code edits so
that it's still human readable after the
fixes are put in, and then to your
other point, as I mentioned, there is
other research that I don't have time
to go into, but both at the SEI and
other tool vendors, where one whole
area of research is trying to improve
the integration into the development
chain and continuous development,
and so that is also a goal is to get
these into the--get these type of
tools and functionality into the
development pipeline so that it's not
an extra tool that people have to go into.

Moderator: More mainstream, yeah.

Speaker 4: That's exactly right.

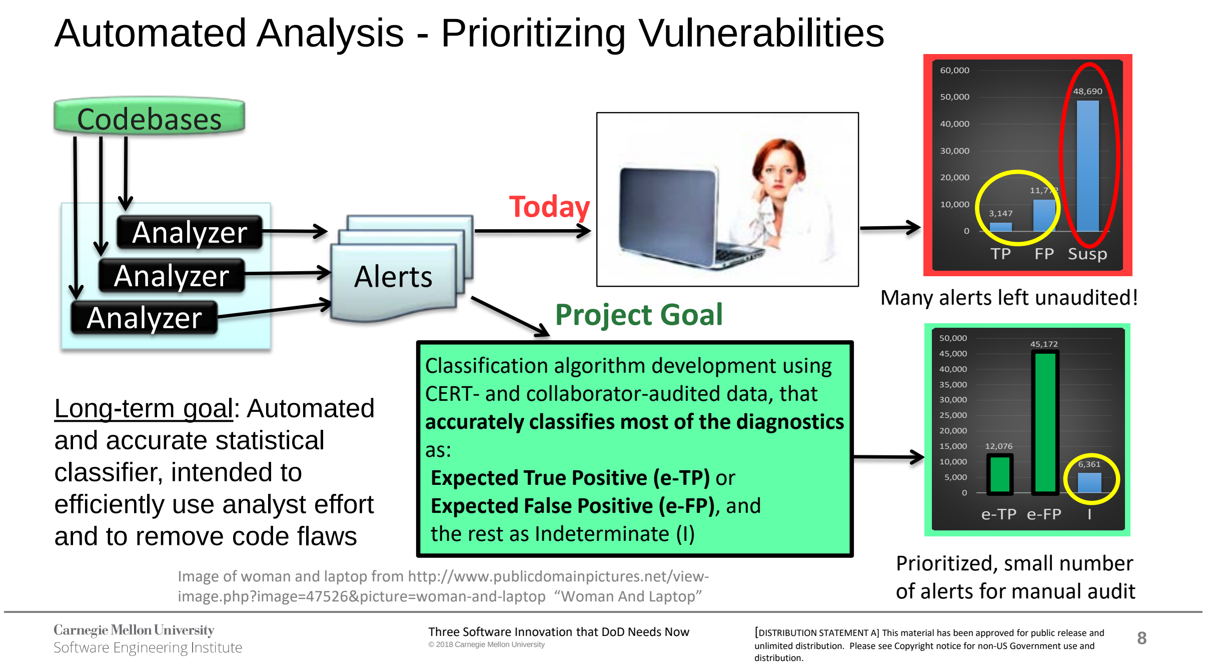
Moderator: Yeah.  Remember the
first time I ran a static analyzer
against some of my code, I think it
was Valgrind.  It just completely
overwhelmed me.

Speaker 4: Yeah.

Moderator: And it wasn't even a
very complex code set, and I was
like, "Oh.  I'm just going to ignore
this and hope I'm okay."

Speaker 4: That's--and to that, you
know, to go to the previous slide--

## Automated Analysis - Prioritizing Vulnerabilities



### Automated Analysis - Prioritizing Vulnerabilities

Many alerts left unaudited!

Classification algorithm development using CERT- and collaborator-audited data, that **accurately classifies most of the diagnostics** as:
**Expected True Positive (e-TP)** or
**Expected False Positive (e-FP)**, and
the rest as Indeterminate (I)

Prioritized, small number of alerts for manual audit

Long-term goal: Automated and accurate statistical classifier, intended to efficiently use analyst effort and to remove code flaws

Image of woman and laptop from http://www.publicdomainpictures.net/view-image.php?image=47526&picture=woman-and-laptop "Woman And Laptop"

Carnegie Mellon University
Software Engineering Institute

Three Software Innovation that DoD Needs Now
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

8

**008 That is largely, even when people are looking at it, the point at the upper-right column there in the red circle, is that most of the things are ignored.

Moderator: Right. Half of them I didn't even know what to do with. Honestly.

Moderator: Didn't know what to do with. All right. So let's pivot back to virtual integration a little bit.

Speaker 2: Okay.

Moderator: I wanted to talk, as much examples as we can give--

Speaker 2: Yeah.

Moderator: --about some places we've actually applied virtual integration and assisted some programs.

Speaker 2: Yeah.

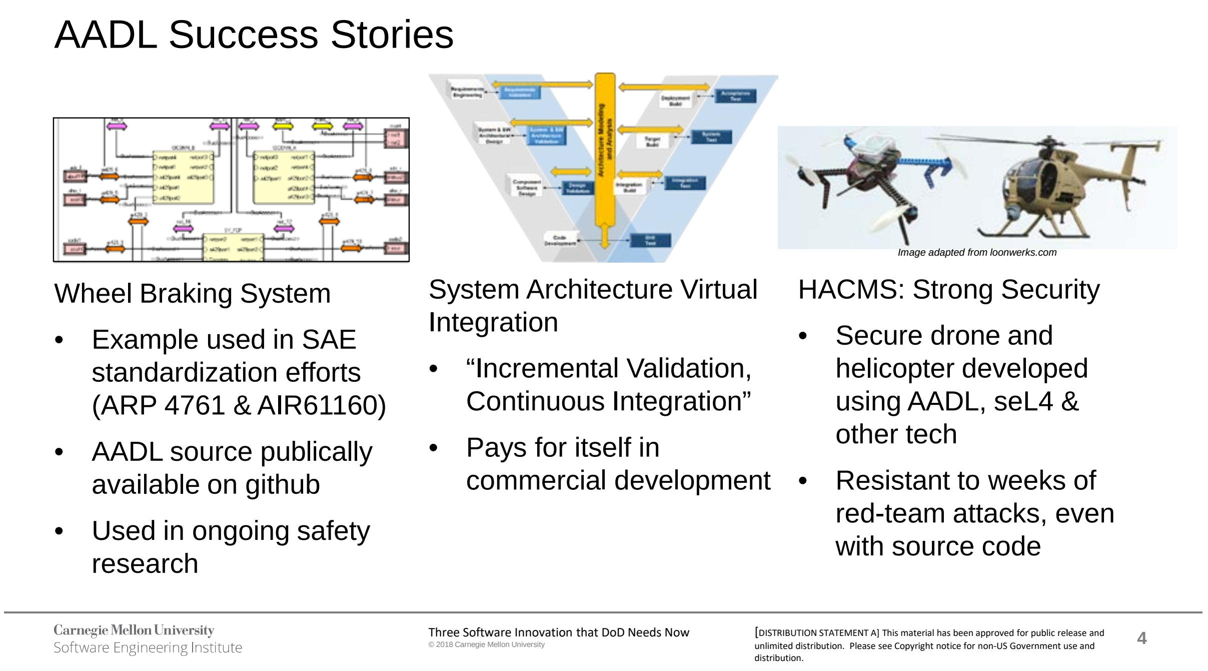
Moderator: And I think we could talk about future vertical lift.

Speaker 2: Yeah, yeah, yeah.  So--

Moderator: And maybe even-- maybe we get to HACMS after a while if we can talk about that too.

Speaker 2: Yeah, absolutely.  So.

## AADL Success Stories



### AADL Success Stories

Image adapted from loonwerks.com

Wheel Braking System
- Example used in SAE standardization efforts (ARP 4761 & AIR61160)
- AADL source publically available on github
- Used in ongoing safety research

System Architecture Virtual Integration
- "Incremental Validation, Continuous Integration"
- Pays for itself in commercial development

HACMS: Strong Security
- Secure drone and helicopter developed using AADL, seL4 & other tech
- Resistant to weeks of red-team attacks, even with source code

Carnegie Mellon University Software Engineering Institute

Three Software Innovation that DoD Needs Now
© 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.  Please see Copyright notice for non-US Government use and distribution.

4

**004 There are a number of projects that have successfully applied AADL and OSATE, the tool environment that we build here.  Jeff mentioned the JMR and FVL, which are--so these are-- JMR stands for Joint Multi-Role.

It's, as I understand it, sort of a technology demonstration/look at better ways to build the next generation of helicopters and vertical lift aircraft, and that, it feeds into, the larger project of future vertical lift, and so that is an ongoing project where AADL is being used right now on a pretty significant scale, and then there are some success stories that I have up on the slide here.

So these range from pretty small to pretty big.  The small one, the wheel braking system, is not trivial by any means, but it was built as an example to be used in some standardization efforts.  So AADL is not just an SEI product, it's this international standard and we are also using it in some other standards to build this example of how things can and I believe did at one point go wrong in a wheel braking system.

So this is a publicly available example, and it's actually been used in ongoing safety research.  So other academics have picked up this example and pointed out ways that we could improve what we've done and how safety analysis could be improved.  Which, coming from academia, I really like this idea that SEI is creating examples and then those examples are picked up by the community and used as conversation points.

Then there's this larger effort, the system architecture virtual integration, which really drives this idea of integrate-then-build, that much the same way you have a V-model of software development that many of you are probably familiar with where you go from requirements down to building the code down to-- or all the way back up to acceptance and deployment, that you should also have a V-model of testing, and so when you are building your requirements there should be some way to verify those requirements, and one of the cool up-shots of the SAVI, the S-A-V-I effort, is that it has been shown to pay for itself in commercial development.  That this isn't something that costs a ton of money and, you know, maybe incrementally improves your software development, but this actually avoided enough errors that it paid for itself.

Moderator: And some of the SAVI members are things, people, like Boeing and Airbus, right?

Speaker 2: Yeah, yeah.  These are big--

Moderator: Yeah.  So--

Speaker 2: --commercial--

Moderator: --good acceptance in the avionics and safety critical real-time areas.

Speaker 2: Absolutely.  So AADL is usable to describe a range of systems.  It's primarily right now, see most use, in the avionics world.  But, you know, not necessarily.  When I was in grad school, we were looking at applying at medical devices and found some good benefits there, and then the last project that you mentioned is HACMS, which is a really neat project where the DoD said, "Okay.  We recognize that we need some sort of foundational shift to build really strong security into our systems.

Moderator: So just for the audience. It was DARPA program.

Speaker 2: Oh, yeah, yeah.

Moderator: Yeah.  So high assurance, cyber military systems I think is how it expands?

Speaker 2: I think so.

Moderator: Yeah.

Speaker 2: And it was--and correct me if I'm wrong here, but there were sort of phases where they first wanted to build this little drum, the small quadcopter on the left, and then a full-size one that would carry people, on the right there.

Moderator: Yeah.  The ultimate demonstrator was a full-size autonomous helicopter.

Speaker 2: Yeah.

Moderator: So a full-size helicopter autonomously flown.

Speaker 2: So--

Moderator: And not remotely flown but autonomously flown, yeah.

Speaker 2: But yeah.  So the security problems or potential problems are pretty apparent here that you have an autonomously flying drone, and so the effort was to see if we could develop these in such a way that they would be resistant to attack, like, really resistant to attack. So a number of technologies were used, various formal methods, but AADL was used to specify the architecture because it has strong semantics that really don't leave a lot of ambiguity that could be exploited, and it was. It worked. It was resistant to several weeks of red-team attacks. They couldn't take control of it. I think at one point they found they could, you know, disable logging temporarily or something. So pretty low-level stuff, and that was even with the source code, the full design schematic. It's really a great success story of all sorts of cool software engineering research, including using AADL to get this rock-solid architecture specification.

Moderator: Very cool. Yeah. I'm very impressed with that project.

Speaker 2: Yeah.

Moderator: I've referred to it a lot lately. So let's--I'm going to reattack over here on the--with our data scientists. I'm going to make a comment, and you guys please comment on this after the fact if you want, is I make it a point always to talk about the discipline or the area of data science and not call it just blind application of machine learning, because I think machine learning and artificial intelligence is a bit of a buzz word that we have going on today. But in my mind, machine learning is one of the algorithmic tools in your toolkit as a data scientist, right. It's not really a field of its own. Well, some people think it might be a field of its own but--

Speaker 1: I'll push back real strongly there.

Moderator: Please do. Please do.

Speaker 1: Machine learning is a field of its own.

Moderator: Good.

Speaker 1: Yeah. I'll get a PhD in Machine Learning.

Moderator: Yeah, yeah, yeah.

Speaker 1: Mine's in Statistics, but so we kind of like fight with each other about, you know, sort of who--

Moderator: Statisticians versus machine learning experts or--

Speaker 1: Yeah. One comes from-

Moderator: --yeah, machine learn--

Speaker 1: --a computer science background and one comes from more of an experimental design background, and I come from more of an experimental design background instead of a computer science background.

Moderator: Right.

Speaker 1: But at the PhD level, we read each other's papers, we go to the same conferences.  There's not a lot of differ--

Moderator: So I made this statement in the past and you agreed with me that all ML is statistics.  Is that too controversial?

Speaker 1: ML is its own field that has its own methods of knowledge in attacking problems.

Moderator: I hope that the questions are going to go crazy here in that answer.  Tried to be provocative.

Speaker 1: Versus--and statistics is different.  Like, for example, a broad brush of machine learning is to look at a problem and then write down something like a loss function and then optimize it, right.  Versus, like, a statistician might look at a problem and say, "Okay.  What are the experiments that we might perform to do some causal inference or something?  Machine learning people are often primarily focused around problems that have to do with prediction because prediction is important.  Statisticians are often focused around problems that have to do with causal inference because knowing why something is, is important.

Moderator: So explanation and prediction sort of things.

Speaker 1: And we work together and sometimes I do ML stuff, sometimes my ML colleagues do statistics stuff.

Moderator: Right.  Tools in the toolkit of the data scientist.

Speaker 1: Yeah.  Well, this part of the data scientist, right?

Moderator: Okay, yeah.

Speaker 1: We've still got the engineers and the subject matter experts.

Moderator: And subject matter experts.  Absolutely, yeah.  So that bigger collaboration, as I were-- so Lena, you have any comments?  I saw you making a couple of faces at me there.

Speaker 3: I, I mean, I think, you know, this is again why, like, communication is pretty important. Because when you're trying to span a couple of domains and describe a problem where there might be a little bit of, like, domain mismatch in how people are using terminology, like, that is where having logical language models really helps to resolve a lot of that difference, because everybody can speak the same language and then we can resolve it, you know, sort of by machine, like, through a logical language model to say, "This way that statisticians talk about this problem is getting at the same question that this way that machine learning researchers talk about the same problem," but there's a little bit of miscommunication that comes just from like domain-specific language.

Moderator: Interesting. Okay. So I want to come back and we talked about this just a little bit before, Lena and I did, about what can we do--when I say "we," I mean, well, society at large, but the DoD, even specific programs, what can we do with our data now as we create vast amounts of more data that could greatly aid using that data most effectively in data science to answer questions.

Speaker 3: So I think that when you are starting a data collection there's a strong responsibility to think about how you might want to use it and what the implications of the decisions that you make about what you collect are at that time. Right? Because one of the big problems that you see with trying to answer a question is, "Can I identify some data that speaks to that question, and then with the features and the specific data types that I'm collecting and the specific elements that are in that data set that I have collected or will collect in the future, what kinds of questions will that data support answer?" and so one of the things that I have run up a lot in my work is trying to identify opportunities to influence some of the--based on some analysis that I've done on existing data that we have, what are features that I can't currently extract from that or infer from that that might help me understand my problem so much better?"

Speaker 1: Right.  And just to build on that, like, there's that--

Moderator: I want to ask a question about the feature thing.  I'm sorry.

Speaker 1: No, please.

Moderator: On the--when you say--if the data doesn't have the features you need, how can it get the features?  Do you have to modify the sensor?  Do you have to modify how the data's collected?

Speaker 3: Well, it kind of depends on what the question you're asking is and what things you can measure in your environment, right?  And so, I mean, but that's why sort of involving--involving data science in a data collection decision can be really powerful in terms of controlling what kinds of questions you might be able to answer and just that sunk cost of once you started data collection, you have something that has a value and if you had additional pieces of information you might be able to answer more questions more cleanly or more quickly than you would be able to with what somebody thought was the right thing to collect when they started, so--

Moderator: All right.  So I'm going to go out on a limb and then I'm going to let Nathan comment, is we've got this--that--so what you said, I think, underlines that team approach to data science where you've got a subject matter expert, a strong subject matter expert, some engineering support to help realize the system, and the data scientists. So I'm going to coin the term called data science DevOps.  How's that sound?  Because that's sort of like the DevOps component, right.  You bring in the people that have to operate the system at the end. You've got a subject matter expert and a user with the developers and as a core team.

Speaker 4?: Can I interject?

Moderator: Absolutely.

Speaker 4?: Data science secure DevOps.

Moderator: Data science secure DevOps, there we go.  So we're--

Speaker 4?: Yeah.  Because we're moving toward secure DevOps.

Speaker 1?: How can we make this longer?

Moderator: Yeah, right, yeah.  Well, we have secure--

Speaker 4?: Secure DevOps is a thing now, so we might as well not lose it.  Might as well just add it in there, so--

Moderator: Right.  All right. Nathan, please, I'm sorry I had to interrupt your--

Speaker 1: No, I like that a lot.  I do like that a lot.  The only thing that I was going to add to Lena's was that there's that joke about a man outside of a bar looking for his keys under a streetlight and someone comes up to him and says, "Hey, did you lose your keys under the streetlight?" and he says, "No.  But that's where the light is."  Right.

Moderator: Yeah.

Speaker 1: And with all of the sensors and all of the data that we're collecting, we're essentially picking where our streetlights are, and if people who are not involved with the kinds of questions that are going to be asked, are involved in those, "What are we going to collect?" things, we might end up pointing our lights at places that make it really hard to find the answers that we really want.

Moderator: Well, and the adversary, which ultimately the Department of Defense has--builds, has to build systems that are robust to adversary action, they're going to try to infer where our streetlights are pointed and operated in the shadows.

Speaker 1: Right.

Moderator: That's, I mean, there's a whole field of malicious--presenting malicious data to a machine learning algorithm specifically to trick them or break them.

Speaker 1: Right.

Moderator: And they can be quite brittle sometimes, so--

Speaker 1: Right.  You can put on specially formatted sunglasses that make you look like Tom Cruise to a learning system.

Moderator: Yes.  Lujo Bauer at Carnegie Mellon.  That's some of his work, actually.

Speaker 1: Right.  Right.  Exactly.

Moderator: So yeah.  I actually try to look like somebody else.  No.  No. Okay.  So let's come back to automated code analysis.  Where do you want to go here with this one?

So a lot of what you do is static analysis.

Speaker 4: Right.

Moderator: Can you talk a little bit about dynamic analysis, how that plays in and how maybe--I know there's another team at SEI that's doing a lot more dynamic analysis, but can you talk about some of the applications of dynamic analysis as well?

Speaker 4: Sure, sure.  So dynamic analysis--

Moderator: And maybe actually give, explain the difference, between the two for the audience.

Speaker 4: That's what I was going to do.

Moderator: Yeah.

Speaker 4: So they come from things, from different perspectives, kind of like statisticians and machine learning experts.  So dynamic analysis is analyzing software while it's running, and so often--there's many types of dynamic analysis just like there's many types of static analysis, but often it's trying to find inputs to software while it's running to break the software, and sometimes the breakage of the software is security-related, sometimes it's not, but often if you find something that, you know, some behavior of the software that is unexpected, then you might be able to adjust that to get to a security issue, and so that's a whole different area of research, as you were mentioning.  The benefit of dynamic analysis is that you have basically every incident is a true positive, right?

Moderator: Oh, right.

Speaker 4: That something breaks.

Moderator: That really broke.
Yeah.

Speaker 4: Right.  We know it broke.  So there must be something wrong with the software that allowed it to break.  Whereas with static analysis, as we mentioned, a significant concern or challenge, is the amount of false positives, right.
We get lots of alerts that maybe something in the code isn't right, but it might actually be mitigated somewhere in the code so you can't

actually attack it and there's no real vulnerability.  So basically every security incident or security finding that you have in dynamic analysis is a vulnerability of some type, and from the--

Moderator: Or at least an error.

Speaker 4: Well, what--

Moderator: That could become a vulnerability, yeah.

Speaker 4: Well, as you say, from the information assurance model of confidentiality, integrity and availability, often it might just be availability.  We crashed it and you don't want it to crash, and maybe that's all it is, but it crashed, and so they're coming from two different angles, but as research has kind of continued, they're starting to merge, to try and get the benefits of both worlds, and so we have, you know, different types of analysis that are merging different types of things like execution or other methodologies that are trying to use, again, depending on the direction you're coming from, either first use binary--or dynamic analysis, to find the vulnerability and then go back and look at source--or static analysis, to try and find where it is.

Moderator: Right.

Speaker 4: Right.  Because--

Moderator: And that's the happy scenario where you've got the binary and the source code.

Speaker 4: Exactly.

Moderator: And at least some mapping between the two.

Speaker 4: That's right.  Because the--what I didn't mention was the challenge with dynamic analysis.  The challenge is fixing it, so now we know that there's something in the code that breaks and this input that causes the unexpected behavior.  But we have--we don't necessarily know where in the code, you know, from the source path, the execution path, where it failed, and so trying to find what the fix should be is a problem.

Now, some researchers are, as I mentioned, just trying to fix the binary and they don't care about the source code, because it might be hard to find or because they don't have access to the source code.  But that then hurts the ability for maintainability and making sure that you don't have a regression error the next time you update that code or your contractor updates the code.  So the happy medium is then using static analysis and source code analysis to try to find where a lineup of weakness in the code to the vulnerability, and similarly, if you're starting with static analysis, people are using dynamic analysis methods to try to identify whether or not that particular weakness will lead to a vulnerability.

Moderator: Is a true positive.

Speaker 4: Yeah.

Moderator: Yeah.

Speaker 4: Yeah.

Moderator: Nice.

Speaker 4: Whereas where, you know, where the project I talked about is using machine learning and data to try and predict whether or not something is, there's other methods that are actually trying to do execution paths, and try and tell, "Did this really lead to or will this lead to some sort of a vulnerability?"

Moderator: Right. Yeah, so one of the techniques I try to use is I try to imagine the ideal instinct for things. So if you're massively successful in all your research, where would we be, you know, 5 and 10-so years from now? So that's what I want to--want to go around the table. We got about 10 minutes left.

Actually, questions? We have any?

Facilitator: We do have a couple audience questions. We could sneak them in before the--

Moderator: Go ahead. Sneak those in, and then we're going to do--

Facilitator: --future states.

Moderator: --the idealized end state. I--

Facilitator: Okay.

Moderator: Okay.

Facilitator: One that came in during, I believe, Sam's section, was, "Many DoD customers--" and this is from Brian, asking, "Many DoD customers are concerned with time-to-mission rather than design activities. How do you convince government/non-engineers MBSE values, a model-based systems engineering value?"

Speaker 2: Well, I think that you would look at the success of model-based system engineering in bringing down costs, which typically are a function of manpower. So when we say that this is paying for itself in commercial development, that means that you are saving enough time to save money. I also think that, you know, I recognize that this is a driving need, but there are competing needs, like the need for a secure system, and I would make that case to the acquisition director or manager.

So yeah. I would say that, you know, you are going to be saving quite a bit of time, that the cost savings from these efforts stem largely from development time savings, and then also the fact that you would be producing higher-quality software in, you know, in a insecure cyber world. Insecure software is extremely costly both in terms of development delays and even human lives.

Moderator: And not just insecure but correctly functioning software, right?

Speaker 2: Right.  Right.

Moderator: And so my argument honestly to this one is, and very sensitive to the whole time-to-capability argument, that we want to spend capability as quickly as we can to stay ahead of or at a minimum at pace, but ahead of adversary development, but you really want to have not time-to--shorter time-to-capability.  You want to have the minimum amount of time to correct capability.

Speaker 2: Right.

Moderator: And my argument with model-based systems is very in doing things, having more discipline, earlier in the lifecycle.

Speaker 2: Yeah.

Moderator: You're going to get, you're a lot--you're assured a bunch, much more, correct capability, even on the same or less time, than you would overall be.  So you have to take care to--sometimes if you spin too quickly, you field things that are very fragile or break.

Speaker 2: Yeah, and I would also just add sort of as an under view, that this is a weakness of research in all aspects of software engineering research.  You know, whether--particularly formal verification, but model-based engineering as well, that, you know, cool techniques are developed but then don't work for whatever reason in the real world, and so--but I would loop back to the state of the art even if you find that the current technology doesn't meet your needs.  I'd loop back in a few years because the work is progressing and so whereas before you had these individual models for each sort of analysis you wanted to do with something like AADL that is the single source of truth for multiple analyses.  You know, there really are time savings that weren't there a technology generation previously.

Moderator: Yeah, couldn't agree more, and to sum the point out, that the SAE standard for architectural analysis and design language is like 18 years old now.

Speaker 2: The first version, yeah, yeah.

Moderator: So quite mature.

Speaker 2: Yeah, yeah.

Moderator: And has been successfully used in big-system integrations.

Speaker 2: Absolutely.

Moderator: And my argument, it's one of the three things DoD needs more of right now.

Speaker 2: Yeah.

Moderator: So--

Facilitator: Another one, one for Bob actually asking, this is from Jack, asking, "Are your analyzing--are you analyzing executable level code instead of just source code?"

Moderator: I can give an intro there but--

Speaker 4: Well, I guess it depends on the "you," but let me answer and then you can add.

Moderator: Yeah.  Sure.

Speaker 4: So my specific team largely focuses on source code, but there are other groups at CERT that are working on executable binary analysis, both for--well, largely for correctness and/or for, you know, finding vulnerabilities to be able to patch them or whatnot and that goes back to, like we said--

Moderator: And about more of the identifying malware.

Speaker 4: Yeah.

Moderator: Malware analysis, yeah.

Speaker 4: And also identifying and categorizing malware to understand, you know, what type of malware is happening.  So there's, there's a lot of executable binary analysis going on for various purposes as well.

Moderator: Yeah.  One of the really compelling areas that I think, neat things that I think SEI's been doing, is so Bob's team is up here at source code and they actually, a lot of times, will take that source code to the

intermediate representation and
actually do analysis there or
correctness and security
vulnerabilities and actually even
make changes there, which helps
make them language agnostic, right.

Speaker 4: Right.

Moderator: So you can go to the--
fix it in intermediate representation
and go back to whatever source
language needed, and that's a pretty
good terministic process, right?

Speaker 4: That's right.

Moderator: But we got another
team at the other end that they've
largely made their living doing
hardcore binary analysis where that's
the only artifact they have is the
binary, and they've actually created
techniques to hoist that binary not
just to assembly language but even
to higher-level object
representations, data structure
representations, and I really think the
two teams in the two outlines of
research are getting pretty close to
meeting.  Now, I don't think I would
make the claim that we can go from
source code to binary in a
deterministic fashion.  We can, if you
know all the switches.
The other way's a lot more, there's a
lot more non-determinism going the
other way.  But I think there's a lot of
progress being made with both the
binary analysis and the source code
analysis.

Speaker 4: Yeah, yeah.

Moderator: And they collaborate a lot also.

Speaker 4: Unfortunately I think you just took my 5 to 10-year answer.

Moderator: Yeah. That's your--that's what's also look like.

Speaker 4: But that's okay. I'll--but yes.

Moderator: Before we tie those to the--that's okay.

Speaker 4: Yes, that's fine.

Moderator: We only have four minutes, so--

Speaker 4: That's okay. So yeah. So as a detail, as I was mentioning, that those two directions, we are actively working towards connecting those and one model to think of that is connecting through the whole compiler process that whether you're looking at the input or the output, that we can go by directionally through that whole compilation process.

Moderator: Yeah. So I think this might be--I apologize in advance. I hope it's not a hard question for--because I think these are pretty specific technologies when we did the provocative title of "What Three Things Does DoD Need Now?" Data science was a lot more of a more general area, but I think it's--we need more of that. What does awesome look like for a data scientist 5 or 10 years from now?

Speaker 1: Superpowers.

Moderator: Superpowers.  Like that answer.

Speaker 1: But a, somewhat a serious answer in terms of superpowers.  Like, a lot of my work has to do with there is a subject matter expert with a problem and there's all of this stuff that a machine might be able to help them with, but they just don't know how to do it yet. So like with the malware thing, let's break up that malware into different pieces and let's use the machine to give context.  Now that reverse engineer has a superpower where they look at all this stuff and say, "Oh, these groups are related; these groups aren't.  I'm going to reverse engineer that, that, that and that," and now I understand the whole pile, instead of having to reverse engineer all hundred things.  So more things like that.  That's what I mean by superpowers.

Moderator: So data science gives people superpowers.

Speaker 1: Indeed.  That's the idea.

Moderator: That's awesome.  That's awesome.

Speaker 1: We're like the Green Lantern.

Moderator: Yeah.

Speaker 1: Little rings.

Moderator: Well, he's one of my least favorite of the superheroes, I guess.  Just joking.  All the Green Lantern fans are going to get mad at me now.

Speaker 1: Batman's a scientist.

Moderator: Lena, do you want to talk to that one?  What does awesome look like for you?

Speaker 3: Yeah.  I mean, very similarly, like, better capacity to carry and understand the data sets that we have and better visibility to sources that are creating and storing data to the kinds of data that have the highest value to them, and the kinds of data collections that will empower, like, the best kinds of insight, and so just really thinking about the, like, the data curation and maintenance parts of sort of the data science process in the sense that, like, as Nathan and I have discussed a few times about, you know, enabling the relationship between a subject matter expert who understands the domain that you're working in, to building some kind of a model, to engineering a way of processing the real-world data to build that model, and so just really being able to understand what it is that you're collecting and storing.

Moderator: Yeah, and I think the, one of the elephants in the room we didn't probably spend enough time talking about, we don't probably have time, is data curation I think is just a humongous task for data scientists and for machine learning experts in general, that a lot of times you're given literally a pile, and I mean that in a very negative sense, of data, that is a pile of data that you have to make sense of, and it's not curated well.

Speaker 3: Yeah.  I mean, one big problem for a data scientist is that a lot of their time can be spent just trying to, like, make the data usable and then just try to understand what's even in it before they can even make any inferences, and so anything that we can do to improve the tools for curating and understanding the data sets so that we can make the data scientist time be better spent on the actual analysis portion of the task is definitely a need.

Moderator: Yeah, and this is where my sort of push, I hope it came through a little bit, was get your local data scientist involved early in your data creation so that they can help you create it in a way that makes it much more effective, and I think we're--

Speaker 3: Absolutely.

Moderator: --out of time.

Facilitator: Yeah.

Moderator: Are we going to wrap up?

Facilitator: We're fine.  Yeah.

Moderator: Okay.

Facilitator: We can--if you want to answer, you know, Bob and Sam.

Moderator: Well, I just--we can wrap up, but buy, you know, go to your local software engineering store near you. Buy some--get some data science and some data scientists. Go get some virtual integration, and we didn't get to talk about design space exploration. I wish, but maybe the next time, and go get yourself some automatic code analysis and repair and I think a lot of it's mature enough now to apply industrial scale problems and I think we'd be better off if we all just used a little bit more, or seasonings or ML and virtual integration and automatic code repair seasonings.

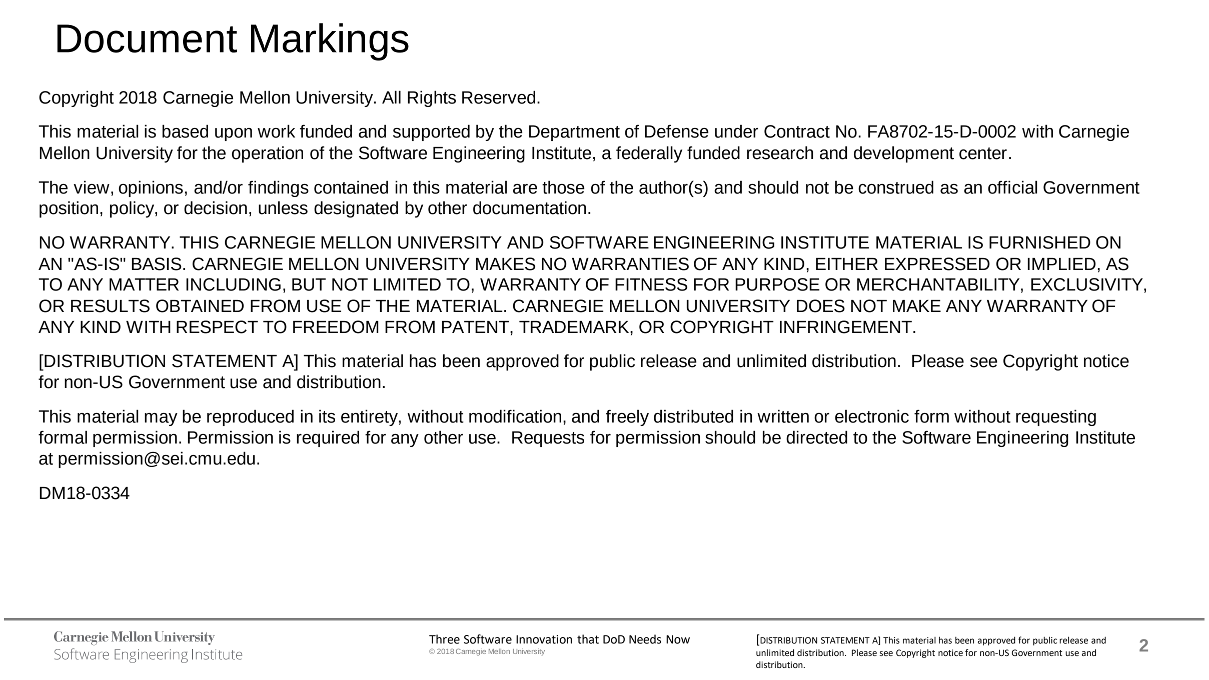
Facilitator: Looks like a good way to end. Thank you, Jeff. Thank you, panelists, for--

Moderator: Yeah. I want to thank all the panelists and the folks behind the scenes too that make all this happen.

Facilitator: Yeah. Great. Great discussion today. We'll wrap up. Just want to invite everybody to their upcoming Software and Solutions Symposium 2018 that's going to take place at our Arlington office in--March 27th. This event--I'll read the tagline here. Says, "Attend this exciting event to learn to tailor acquisition and development programs to avoid common software and cyber pitfalls and position your programs for success."  So it's free to anybody with a .mil or .gov address, and we'll send this information on this event. Hopefully you can attend there.

Our next virtual event will be March 23rd, and the topic will be "Five Ways to Boost Cybersecurity with DevOps." Again, thanks everyone for attending. We had a number of comments we didn't get to but maybe we'll share these with our panelists and try to get back to everybody in the future. But have a great day, everyone. Thanks for attending.

## Document Markings



### Document Markings