

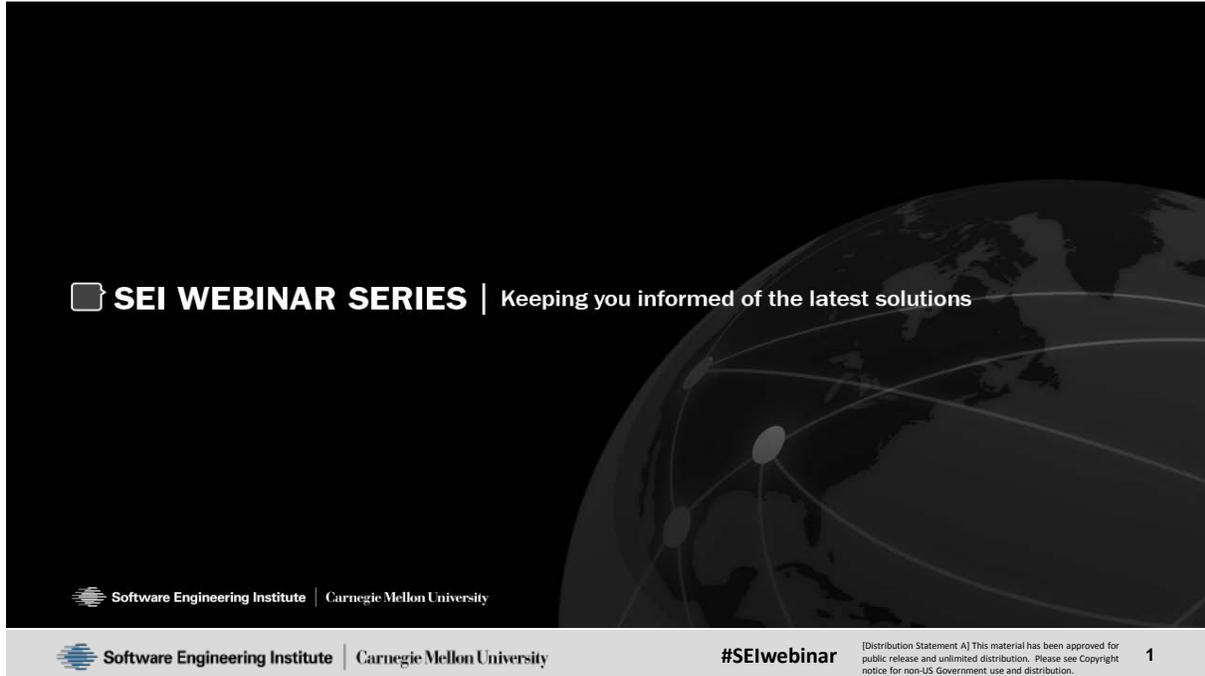
Jeep Case Study and the Automotive Cybersecurity Framework/Secure Software Development Landscape

Part 1

Table of Contents

SEI WEBINAR SERIES Keeping you informed of the latest solutions.....	2
Carnegie Mellon University.....	2
Copyright 2016 Carnegie Mellon University.....	3
Lessons Learned from the Jeep Hack: How to Reduce Software Vulnerabilities in Cyber-Physical Systems	3
What Happened with the Jeep	7
Discussions	8
Lessons Learned from Jeep - Case Study Review	9
What did the Jeep experience illustrate.....	26
Catching software faults early saves money	30
Security is implemented across life cycle	31
Polling Question.....	32
Room for improvement	36
Cross life cycle issues	38
Q&A.....	40
SEI WEBINAR SERIES Keeping you informed of the latest solutions.....	44

SEI WEBINAR SERIES | Keeping you informed of the latest solutions



SEI WEBINAR SERIES | Keeping you informed of the latest solutions

Software Engineering Institute | Carnegie Mellon University

Software Engineering Institute | Carnegie Mellon University #SEIwebinar [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. 1

Carnegie Mellon University

Carnegie Mellon University

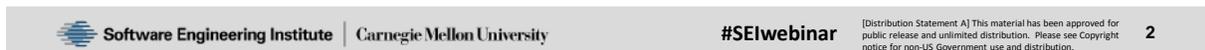
This video and all related information and materials (“materials”) are owned by Carnegie Mellon University. These materials are provided on an “as-is” “as available” basis without any warranties and solely for your personal viewing and use.

You agree that Carnegie Mellon is not liable with respect to any materials received by you as a result of viewing the video, or using referenced websites, and/or for any consequences or the use by you of such materials.

By viewing, downloading, and/or using this video and related materials, you agree that you have read and agree to our terms of use (www.sei.cmu.edu/legal/).

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

© 2016 Carnegie Mellon University.



Software Engineering Institute | Carnegie Mellon University #SEIwebinar [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. 2

Copyright 2016 Carnegie Mellon University

Copyright 2016 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[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.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0003493

Lessons Learned from the Jeep Hack: How to Reduce Software Vulnerabilities in Cyber-Physical Systems

Lessons Learned from the Jeep Hack: How to Reduce Software Vulnerabilities in Cyber-Physical Systems

Mark Sherman
Technical Director, CERT

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

**004 Presenter: And hello from the campus of Carnegie Mellon University in Pittsburgh,

Pennsylvania. We welcome you to Software Engineering Institute Virtual Event, Lessons Learned from the Jeep Hack: How to Reduce Software Vulnerabilities in Cyber-Physical Systems. Depending on your location, we wish you a good morning, a good afternoon, or good evening.

My name is Shane McGraw. I'll be your moderator for today's presentations. And I would like to thank you for attending. We want to make today as interactive as possible. So, we will address questions throughout each discussion and again at the end of each discussion. You can submit any questions you have through our event staff at any time during each presentation by using the questions tab on your control panel.

We will also ask a few polling questions throughout the presentation. And the first one we'd like to ask is, "How did you hear about today's event?" And that will appear as a pop-up on your screen. Another three tabs I like to point out are the download materials, Twitter, and survey tabs. The download materials tab has cybersecurity and software development related work and resources from the SEI that you can download now. For those of you using Twitter, you'll want to be sure to follow @CERT_division. And use the hashtag SEIwebinar. Once again, it's CERT_division with a hashtag of SEIwebinar.

And now, I would like to introduce our presenters for today. Dr. Mark

Sherman will serve as our facilitator, and also as a presenter today. And he's the technical director of the cybersecurity foundations group at CERT within CMU's SEI. His team focuses on fundamental research, on the lifecycle for building secure software and on data driven analysis of cybersecurity. Before joining CERT, Dr. Sherman was at IBM and various startups working on mobile systems, integrated hardware/software appliances, transition processing, languages, and compilers. He's published over fifty papers on various topics in computer science.

I'd also like to introduce Mr. Chris Valasek. Chris will give our keynote talk today. And Chris is the security lead at Uber's Advanced Technology Center, which is in Pittsburgh. He's regarded for his work in the automotive security arena. And most recently Chris was lauded for his remote compromise of a 2014 Jeep Cherokee, whereby he and his research partner obtained physical control of the vehicle. Valasek specializes in reverse engineering and exploitation research. And Chris has a B.S. in computer science from the University of Pittsburgh and is the chairman of SummerCon, America's longest running hacker conference.

And now, I'll turn it over to Mark Sherman. Mark, Chris, welcome. Thank you.

Presenter: Thank you Shane. And welcome everybody to this webinar. I appreciate you spending several

hours of your day with us. And we're going to talk about a very important topic, about the risks that are involved in or emerging on cyber-physical systems. The attacks on IT systems, whether they're the Target system in retail, what happened to the Office of Personnel Management in the federal government, Anthem Health in healthcare, Sony in entertainment. These have all been IT related that released information that was harmful to the organization and individuals.

What's happened now is that this is moving into the cyber-physical space as well. So, we've seen steel furnaces in Germany attacked. We've seen hotel locks being able to be disabled, the electric grid in the Ukraine. Insulin pumps have been disabled, and of course the very popularized Jeep remote control. But when these kinds of events occur, we can get physical damage that causes problems to real people as much as death in some cases like some of these medical devices. So, it's important for us to figure out how to build systems that are more resilient to attacks, to eliminate the vulnerabilities, in order to have a better environment. So, ultimately what we are intending to do today--

What Happened with the Jeep



**005 Is basically not be the next Jeep, that we have learned many things from these various events. And it turns out that many of what's going on, the vulnerabilities that are there, are well understood. And the ways to avoid them are well understood. That if we apply what we already know about building secure systems, we can make the cyber-physical world a lot safer than it is today.

In order to do that, were going to go through--

Discussions

Discussions



Jeep Case Study Review
Secure Software Development Landscape
Secure Requirements Engineering
(Break)
Secure Coding Best Practices
Continuous Integration (Secure DevOps)
Coordinated Vulnerability Disclosure

**006 A range of topics that will help understand better how to build these kinds of systems. Of course what one starts with is analyzing in a little detail how some of the vulnerabilities have already been exploited. So, we'll start, with Chris's help, on the Jeep case study. And we will look at some of the details of how the Jeep was able to be controlled remotely. Then from that, we'll take a look at how one builds secure software and how that would have better enabled the Jeep to be resilient to this.

We'll give an overview of that and then delve in a couple of the specific topics. We'll focus a bit on requirements. We'll discuss a bit about how one actually does the programming of these systems. We'll talk about how one automates this

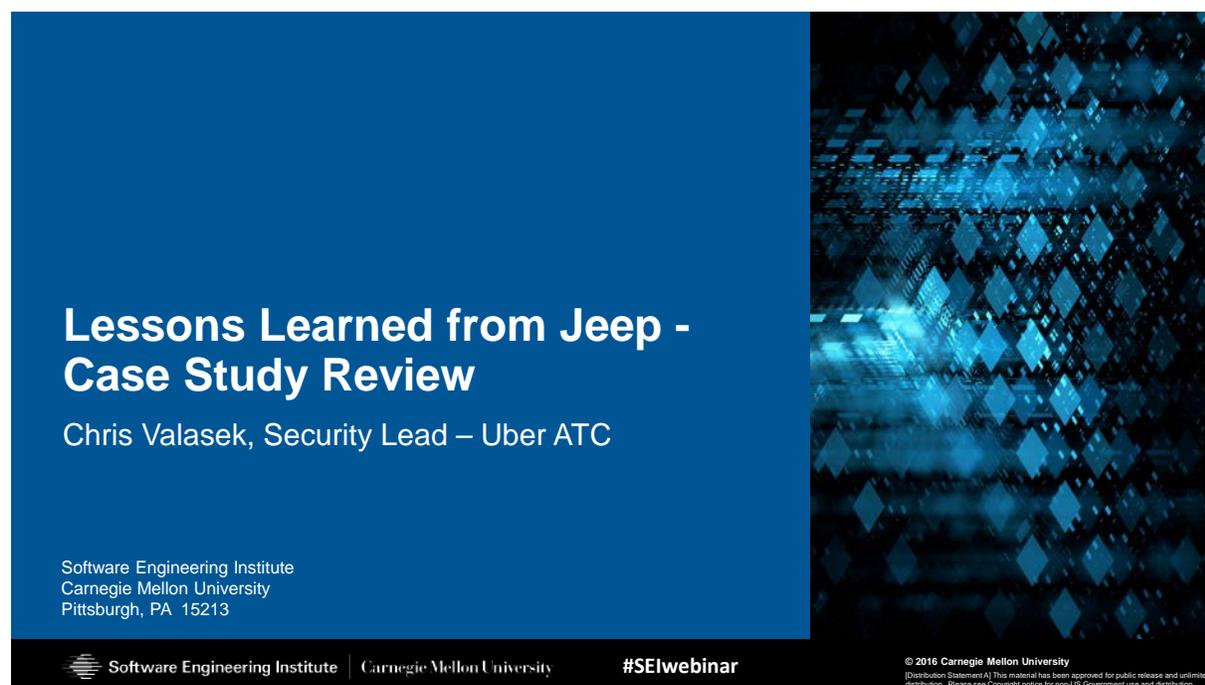
whole process in order to rapidly bring forth better software. And of course, breaches will be inevitable despite our best efforts. So, what is the best way to handle those kinds of events in order to minimize the damage that might occur? So, to start with, I'd like to turn to Chris and have him discuss a little bit about what happened with the Jeep.

Presenter: How's it going Mark?

Presenter: Good.

Presenter: Excellent. Well, I think many people have seen the Wired video and kind of saw us playing around with Andy and thought that was the extent of it.

Lessons Learned from Jeep - Case Study Review



Lessons Learned from Jeep - Case Study Review

Chris Valasek, Security Lead – Uber ATC

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Software Engineering Institute | Carnegie Mellon University #SEIwebinar

© 2016 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.

**007 In reality, it was a year-long effort of analyzing these systems.

Additionally, we released a paper a year prior that actually was a survey of twenty-five different automobiles. That paper was actually us looking for our perfect car that we thought we could compromise for physical control remotely. And the Jeep stuck out because it had a lot of external communications such as Telematics, and Bluetooth, and even in-car Wi-Fi. Additionally, there was no apparent separation between the systems that control things like steering, and braking, acceleration, and those items that were communicated with the outside world. So, we really did this preliminary research to pick the Jeep as our target.

That was our kind of first step in doing all this. And that will show you one portion of security that needs to be considered when designing these things. You need to understand that if you're going to have it communicate with the outside world, what can it touch directly or indirectly? Many times, people will assume just because two systems are on two physically separate circuit boards that they are actually separate when indeed they're not. You could have a serial line, for example, have these two things communicate together. Even if they're not made to communicate in the fashion you've designed them for, people like Charlie and I think about things differently. So, we're using those systems to perform physical control when I think many times people thought that wasn't the case.

Presenter: I saw a presentation that came from the Department of Transportation in where they referenced an analysis of how to get into cars. And at one point, the analysis counted up that there were fifty different antennas. They were a bit encompassing with that inclusion. So, your phone that you kept in your car they also counted in that. But they also were looking at that. Did you see that as a major issue as to number of entry points in?

Presenter: Yeah, absolutely. I mean even things such as your tires have radios in them because after a certain year, it's mandated that you have to have tire pressure monitoring systems. So, there is communications coming from all over. Luckily for us as consumers, it's pretty transparent. And I like that. I like the technology. But you obviously need to be thinking about security when you have external communications.

A lot of these attacks were possible because it was Internet connected devices that were linked up with other control devices that were, to be honest with you, designed before connectivity and security were even a thought. The CAN standard was developed by Bosch in the late '70s or early '80s. And they were for really automated control systems that were self-contained. And this was before you were giving it a cellular modem or Bluetooth connectivity. So, they're not designed to be as resilient to replay attacks or have any level of

authentication built into them. This gets problematic, as you can see, when you have these Internet connected devices touching these things.

Presenter: So, can you tell me the steps that you took in order to get remote control of the Jeep?

Presenter: Yeah, there were a lot of steps. And any one of them could have, I don't want to say prevented it, but would have made it much more difficult. The first thing that we really did was that we understood Wi-Fi. Charlie and I were former pen testers back in our day. So, we figured if this car has Wi-Fi, we at least know how to look at that. That's normal to us. It's just computer Internet connections.

So, lo and behold, we found that indeed it actually had a lot of ports open. And these ports were accepting unauthenticated communications that would pass user inputted data to scripts that they use on the actual head unit of the vehicle. It was Harman Kardon produced this head unit. It was interesting because we didn't think there would be any open ports because it should just be a router. But needless to say, it was our first choice of investigation.

At the same time, we were in parallel looking at how we could do cyber-physical controls of this vehicle. So, when you have a car, you want to look at the type of messages it sends when it's parking itself, or when it's

putting itself into a maintenance mode for mechanics to bleed the brakes, or things like that. So, we were looking at these things in parallel as time went on. And this did take quite a bit of time to figure out.

Early on, we found that there were some scripts that had things like command injection, which is pretty well known in web applications and regular user applications. But it just so happened that it was on an infotainment system in the vehicle. We realized even after looking farther that we didn't need to actually exploit one of these vulnerabilities with command injection because there was a script called execute. And it ran as root on the infotainment system, which permitted us to have super user privileges on this infotainment system. You can imagine sitting down at your Windows or Linux machine and you have administrator or you have root privileges. You can do anything you want.

That being said, that didn't mean that we could send messages out on the CAN bus because it wasn't designed to do that. There was actually a separate microprocessor that was used for that specific application. So, our next portion, and the most time-consuming of this was getting the firmware off the chip that was doing the CAN controller work and figuring out what it did. This actually took us several months to figure out because it's basically just a binary blob that you don't know anything about. And

you have to figure out exactly what it does.

So, we spent forever looking at this, and we finally realized that hey this is where they send messages. And these are how messages are set up in memory. If we could somehow alter this a little bit, then we would be able to reprogram the CAN controller to have our own version of this code on there. Remember, this is all binary. You're literally looking at hex bytes. It took a long time, but we figured out we could actually do that. But we had no way to actually get it on the CAN controller because the head unit was separate. They were literally two different boards. There was an actual air gap in between the two.

Fortunately for us, unfortunately for FCA, they were still connected via serial line, communicated via SPI, S-P-I. We found that there was no concept of code signing on these binaries that were getting flashed to these systems that physically control things. Right? So, we could alter the code to do what we want. And then actually re-flash the chip with a utility from the head unit to the CAN controller, which was discouraging to see in something that should have certain levels of guarantee of code running it. They do a lot of testing in vehicles to ensure things like your safety belt works, and the airbag deploys on time. But then there was no checks to see if the code running on that was actually their code. So, that was the first concerning part.

Even if we were able to do this, which we were, there was no direct method from the head unit to the vehicle where you could say here's a CAN message, send it out, any one you wanted. You could do specific ones, but they didn't do us really much good. We figured out that we altered the code on the CAN control a little bit. So, we would send messages on the serial line, and they would be interpreted to be pushed out to the car's CAN bus. And when we talked about the architecture before, it was that this infotainment system was connected to everything else. So, once we had the ability to send arbitrary CAN messages, that meant we could actually impersonate any other piece of the vehicle that was using those messages, whether sending or receiving. So, at this point, you have arbitrary access to send any type of message. Needless to say, we thought it was a bit alarming.

But at the time, it was limited to Wi-Fi because that's all we knew how to do, which gave the range difference. If I have to drive up next to your car, follow you for some time, that's less of a threat than if I can do anything from say anywhere in the country. We wanted to show that it wasn't a matter of physical proximity but an overall systemic issue. We went back and revisited the code that listened for the certain commands. And found out that indeed it listened on cellular. And the big issue with that is not only the range, but with the Wi-Fi portion, you had to pay. It was like thirty-five

dollars a month or something. Something that I wouldn't think most normal people would do be because that's a pretty high expense. And you already have your phone. So, I wouldn't think many people do that. Maybe they do. Maybe they don't.

But we wanted it to be a kind of mass scale type of thing. And we realized that, indeed, even if you didn't have this package, if you had this specific head unit, it came with a cellular modem that you may or may not have known you had, but we knew that you had. And again, that's another, I guess, concerning part for a consumer is they don't even understand the technology that's in their car.

Presenter: So, is your car constantly connected to the cell towers regardless whether you know it or not?

Presenter: Yeah. Yeah, you can communicate with it. And it makes sense because they have an app for the car that you can unlock the car and start the car from cellphone. So, you would figure there has to be some level of connectivity. And it was just that.

We were able to purchase what's referred to as a burner phone, you pay as you go phone, from Sprint, and were able then to use that as our entry point into the cellular network that held all these cars, amongst other things, whether vending machines, or whatever is using the

cellular technology. Again, this is concerning because you would think that cars would be separated out from say your average twenty-dollar burner phone that you buy at Walmart. It wasn't. So, we were able to get a phone, be it that you couldn't the IP addresses from your Internet, like your cable modem at home. But if you used your phone as the modem, then you could actually touch all these things.

We proceeded to scan the IP addresses that we knew cars were using. For some time, it was really slow because we had a tiny little phone. And we thought there was somewhere between three hundred thousand and four hundred thousand vehicles that may be affected out there. We later found out that it was much higher. And they ended up recalling one point four million vehicles. So, it was a number that was far greater than we ever expected.

From there, even without doing the process as I previously described, which is reprogramming, there were simple mechanisms for getting the GPS coordinates of a vehicle, getting the VIN number of the vehicle. And with those two pieces of information, you could tell what make and model the car was and where it was at any point in time. Less of a physical concern, but a lot of privacy issues stem from being able to say where is this car at and what kind of car is it. So, that's essentially how we went about doing this and researching it.

The lessons that I like to kind of give out the most is that it wasn't just one party involved. This is an ecosystem. FCA took basically all the blame because they are a very highly, well respected-- highly regarded, well respected brand. But at the same, time Harman provided code shipped on an infotainment system that they purchased. And that code was the code that we used to initially get into our vehicle.

Secondly, there was Sprint that let just anyone on their network talk to any other device. Eventually, they blocked that port that we were using to communicate. And that had the highest impact because now we couldn't do this at scale from our house to anywhere in the country. We'd have to actually get close to the car.

And lastly, you have FCA that's doing these types of updates. They're not checking if code is signed by them. Additionally, the biggest issue with this whole research project from FCA's perspective was they had a machine that could communicate over the airwave. But they didn't have a way to update it. So, they had to send out USB sticks and do a full recall. So, I think this would have been probably a different story if they were able to say, "Oh, Chris and Charlie have identified this issue. And we can fix it today or tomorrow without having everyone bring their car in."

Presenter: Although, wasn't one of your ability to exploit was the fact that there was this update capability in those boards. If they hadn't put in that additional wire in order to do a firmware update, you wouldn't have been able to get at that.

Presenter: Yeah, but they had no access mechanism to do over the air updates is what one of the biggest problems was. They are still systemic problems like not code signing that still exist on the cars today. But they stopped the external effort.

Presenter: Right and in other industries like in aviation, the software that goes into the sensors of engines is signed for exactly that kind of reason.

Presenter: Yes, exactly. You want to verify that the party putting code on here is actually indeed you. And it's not an altered form of what you're putting on there, which is exactly what we did. We altered their code slightly to do what we wanted. Yeah, so code signing for these types of applications is highly critical even for your phone. I think Apple learned that pretty quickly as signing all their code. And not only signing their code, they're signing their memory regions. You don't want people putting their own data on these devices for safety reasons, and many times for liability reasons.

Now, it does spring up kind of a conundrum for people that like to tinker with their things, as someone

who enjoys that. But I think many times, especially for this firmware level access, that having verified and cryptographically verified code on them is the way to go.

Presenter: You mention that this was sort of step two in this study. Step one was doing a survey of the various architectures. And then you picked the Jeep as the one that looked the most promising for vulnerabilities. What would you say were some of the better architectures that you saw?

Presenter: Yeah, I mean unfortunately we couldn't-- IOActive, at the time, bought us the Jeep. And apparently, they wouldn't buy us one of every car in the planet, unfortunately, even though we wanted it that way. But yeah, the architectures that we initially shied away from had separation between some kind of gateway unit between things that communicated with the outside world and things like engine controllers. There were even more sophisticated ones that steering was on a different portion that was separated by some kind of gateway than braking and all that stuff. Not to say that's perfect and it was worked as designed, but needless to say, we didn't want to try the hardest thing first.

Yeah, so just like your computer networks that you have in your offices and corporations, you have some level of separation. People from finance shouldn't be accessing code

repositories on different parts of the network. Database servers that are holding highly critical information shouldn't be worldly accessible from the outside. So, we're taking the same kind of concepts because they really do apply to these machines as well.

One of the last, I guess, portions that I do like to mention as well probably just to save more face than anything, I think a lot of people saw the video. And it did prove our point that we wanted this to be real. And I don't think it would have resonated with as many people if it wasn't real, if we were doing it elsewhere. But we worked with FCA for almost nine months prior to this. So, this wasn't us just dumping something on the Internet one day. And we were working with them. We explained to them what we were doing. And we told them that we were doing the Wired piece and everything like that.

So, I do encourage researchers and everyone alike to coordinate with these people. You have to remember that not everyone comes out of the womb being a security expert, as you obviously know. It takes time. And it really takes buy in.

For example, prior to doing automotive research, I was a Windows hacker. And they really didn't get serious about security until Bill Gates' company-wide memo that said, "Hey, we really need to take this seriously if we're going to be in this market." And I think some of the

first big battles that will be fought on these fronts, specifically with cyber-physical-systems is buy in from the top down, where this is a priority, and important, and it needs to be built in as they go along.

Presenter: Does the Wired author still talk to you after being in the car?

Presenter: Yeah. Yeah, Greenberg still talks to us. He's a really nice guy. He asked if we're doing anything new. We said he'd be the first we'd call. I like Andy a lot. He's a great journalist.

Presenter: Is FCA still talking to you?

Presenter: We haven't talked to them much since the events that unfolded. But they were, and I'll commend them on this, they were really cool about it. There was never any legal threatening. They asked us to view things. We felt it would hurt the integrity of the research. And we didn't. And they didn't really have problems with them. So, for all intents and purposes, I'm sure they may not like us. But they were very respectful the whole time. And I think we were, on the opposite side.

Presenter: You mentioned that you did a fair amount of reverse engineering, basically looking at blobs to figure out what they meant. Who would you say has access to that information just as part of their normal job?

Presenter: What pieces of information? You're talking like the firmware code?

Presenter: The firmware code, or the communication message formats, or various other things.

Presenter: Yeah. So, those are probably internal to the company. As for the firmware, if you know the microprocessor, you can open a thing like IDA pro, which is a disassembly tool. That's what we did and spent a lot of time looking at that. As for the messages the car sent, those are internal to the company and not published. So, that was us just reverse engineering. A lot of times making the car park itself, and we wrote our own software to look at the network communications and say this looks like these bytes are changing as the wheel is turning, or this happens when I press the brakes. So, we would spend a lot of time out in the car.

Presenter: So, much of the systems for things like cars have contractors. Harman Kardon's just one of them. In order for all of these various companies to do their jobs if you have all of these units throughout the car, dozens and dozens of units, you have to talk to each other. You have to not step on each other.

Presenter: Yeah.

Presenter: How many people would you sort of stick a finger in the air, would have to know these various

things in order to make sure that it all works together.

Presenter: Yeah, I don't know, lots of people though. There's, I would assume, a lot of groups. I'm only a hacker, so I only know how to break it. But I'm sure building it is much more complicated. Yeah so, there's a lot of people involved in all these steps.

Presenter: So, might a disgruntled employee in any one of them be able to do what you did a lot faster?

Presenter: I don't think so. We're too good. No, I don't know. Yeah, they would definitely have an advantage to having information that took us weeks to derive where they would have an actual document explaining what was happening for sure. We originally, when we first started doing automobile research back in 2013, were trying to figure out what the messages were like. And we thought we could just buy them. Apparently you can, but you have to sign some paper. And it costs an exorbitant sum of money to get those type of databases that have the messages. So, we just figure we'd figure it out ourself. Save ourself a lot of money.

Presenter: So, a well-funded adversary might be--

Presenter: Yeah, right now for a lot of these is well funded is a prerequisite to the whole thing. Now, we had the car because we weren't

going to test against strangers. We only were testing against our car. Additionally, to figure out those messages if you're not going to buy them, you want to purchase a vehicle. So, the barrier of entry to this type of research is very high right now because even that Jeep with the technology packages was somewhere in the realm of thirty plus thousand dollars. So, this isn't a weekend project that most people pick up at home. You actually have to have some funding to get involved in it.

Fantastic. Well, do you have any recommendations at a high level on how you would go about securing these types of systems?

Presenter: Well, just listening to your discussion, and I have to admit I've heard you give this a couple of times already--

What did the Jeep experience illustrate

What did the Jeep experience illustrate



Running out of date software

Wide attack surfaces

Not understanding or appreciating interconnections

Components used in new operational situations

Assumed or misunderstood authentication and authorization needs

Lost opportunity to mitigate damage through disclosure

“Security through obscurity” is not enough



**012 There is a variety of things that I think we saw illustrated by your experience with the Jeep. A lot of the software and some of the components you're talking about were out of date, and there were known issues with those software, and sometimes even fixes for those software that were simply not applied. We already talked about that there's a wide attack surface, that by design they created lots of ways in, which may have been partitionable that we saw in other kinds of situations. Sort of a lack of either understanding or appreciating all the interconnections between the system, the fact that the serial line was placed between these two boards meant it really wasn't air gapped even though it may have been described as an air gap.

Something that you also mentioned, which we actually have seen in the research literature as well is that when you use a software component in a new operational situation, you almost always find a vulnerability. Here, as you said, the braking system was never intended to be connected to a front end that connected to the web. When it did, all of a sudden the path was there. The studies have shown that in general when you introduce a new piece of software that the first defects you find are in functions in the new code. But the first security problems you find are problems in the old code that's now put in a new kind of environment. And this just showed that in many different ways.

Presenter: It gives the old code new context, right?

Presenter: Yes, unexpected context that whoever was building it never thought that, which again led to-- there was sort of an assumed or misunderstood need for authentication and authorization. There's no need for signing. Who's going to do it? Only the mechanic, we trust the mechanics. The CAN bus never has to look at the messages, never has to verify who's coming in with the message because it's all written internally. We know no one from the outside's ever going to do that. Well, someone from the outside now did. And so, all those assumptions have now been changed.

We also saw what I would say was sort of a lost opportunity to mitigate the damage. As you said, you were in constant contact with FCA throughout this process. But it appears from the outside, because I'm not with them either so I don't know, but from the outside it appears that action was only taken after this went public. And so after the article came out, then Sprint did something, then they started to do something.

Presenter: We didn't release anything until there was a patch available. But after the Wired article and our kind of media tour thereafter is when Sprint took action. And it really accelerated.

Presenter: So, that was something that was also something we might learn going forward. And I sort of summarize it sort of with that last comment, which is that security through obscurity is not really the answer. There seems to be a belief, especially by people who are focusing on feature function, that we don't have to worry about this because this is a very complex system. It's very hard to use. You need to know a lot of things. It's just not going to be worth anyone's while in order to go ahead and do this. Well, there are just many examples of which this being the last one and where you just can't rely on the complexity of what the environment is in order to actually solve your security problem.

Presenter: Exactly. We weren't even the first people to do this remotely if

you'll believe that. Researchers from the University of Washington and University of California, San Diego did this on a GM vehicle back in 2011, kind of four years prior. So, even if there was the thought process of this may be impossible because it's super hard, that should have been shattered prior our work.

Presenter: Right. I think several efforts that have done it through the onboard diagnostic port as well. So, again, lots of way in, and any one of them gives you access. The argument on that one however is if you can get physically inside the car to begin with, well all bets are off I think as you know. Take the car away from you.

Presenter: You can go traditional kinetic attack if you have physical access. It might be easier.

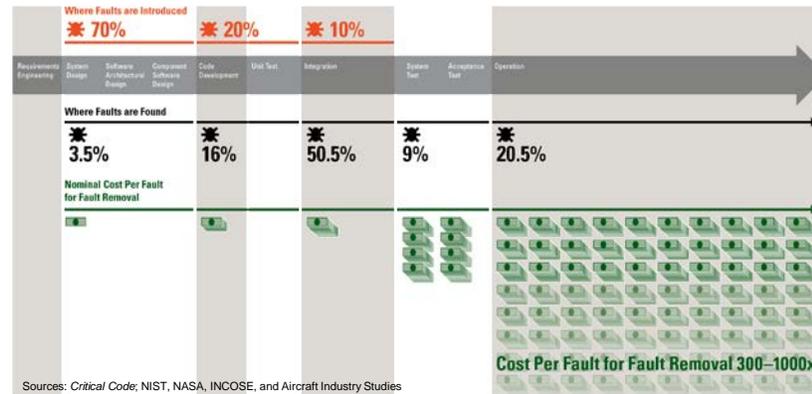
Presenter: But the reason why we're here and why we're trying to focus on this--

Catching software faults early saves money

Catching software faults early saves money

Faults accounts for 30–50% percent of total software project costs

Software Development Lifecycle



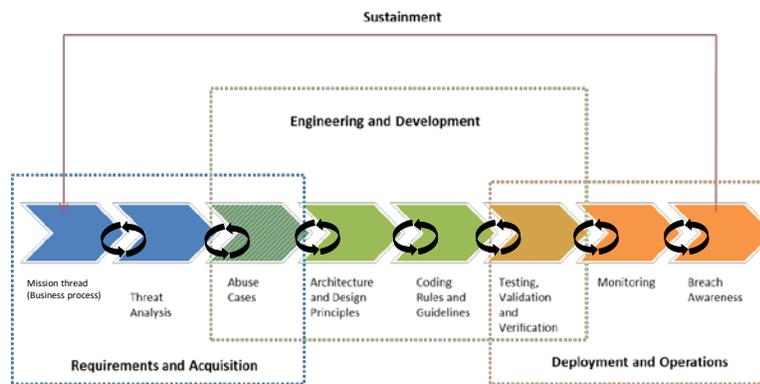
**013 Is that if you see these kinds of things early, it's a whole lot cheaper to fix. That just in general sense, if it costs you a dollar to fix something up front, it can cost you up to a thousand dollars. And that's ignoring any other kinds of implications. So, like in the case of Jeep, they spent hundreds of millions of dollars in a quarter that they got hit once this disclosure came out. Although, not really a security related event, but the Toyota event, which is a software related event, depending on who you're counting, it was somewhere between one and four billion dollars of impact. Yes, it might take another developer. Yes, it might take a couple-- a week impact to the schedule, maybe take a little bit more memory if you want to do some of these things. But the flip side is the cost of not doing it and coming up

with it later is just staggeringly enormous. And it seems to be increasing.

And so, what we're going to do for the rest of the seminar, the rest of the webinar--

Security is implemented across life cycle

Security is implemented across life cycle



**014 Is really focus about how one addresses these in the security life cycle, in the software life cycle with the key message being we know a lot of things that can be fixed right now that, as you mentioned, even in the Jeep example that you gave, there were lots of things that could have been done better that would have made this much, much harder. Not impossible, but we can get a lot better than we are right now. And to do that, we're going to take a look at both the beginning requirements phase, understanding your threats,

understanding the threat models just like we did here with the attack surfaces, the architecture and design phase, looking how to separate things, how you want to actually can code, and then going into operations, how to make sure what you have is working, and when something goes wrong, what to do about it rather than saying oops.

Presenter: I think that's one of the biggest issues that gets overlooked is assuming that if you can make security perfect. It's not something you can make perfect. And you want to be able to remediate when there is an event.

Presenter: Yes. So, with that we've come to our first question.

Polling Question

Polling Question

What tools do you use to support secure development?



Security requirements management tool?

Source code analyzers?

Dynamic fuzz or penetration testing?

Others?

**015 Presenter: So, like I

mentioned in the intro folks, we'll ask a number of polling questions throughout each talk just to get an idea of who the audience and how we can tailor the talk. So, the questions you're going to see popping up now is, "What tools do you use to support secure development? Security requirements management tool, source code analyzers, dynamic fuzz or pen testing, or others. If you use multiple, feel free to type those into the Q & A box as we go along. And while we give you about fifteen or twenty seconds to vote there, we had some questions piling in for you guys. So, maybe I'll address one to Chris here from Brando asking, "In your interaction with Jeep, did you ask if they had done some type of security assessment or threat modeling?"

Presenter: Yeah, I believe when we talked to them, they actually were aware of the vulnerability that we found. And it was fixed in subsequent iterations of those vehicles. It just so happens that when you don't have the ability to update over air, it's hard to retroactively fix previous systems. So, like I said, they weren't completely blind. It's just starting a security program takes time. And it's going to be hard to fix everything backwards.

Presenter: Okay, well this speaks then to Brook's comment. First of all, she says, "Hi, Chris." You must know Brook, but she says, "It sounds like a fairly complete failure to holistically threat model."

Presenter: Yeah, you have to realize that these people are powerhouses in engineering. And you get in a car, and it turns on every time. And it works. And you just have to change the oil, and it will run forever. But figuring out how to secure these things is hard to do. And it's not an area of expertise where there is as plentiful of skilled people to do so. So, while in retrospect, it seems like it was a complete failure. I'm sure they were trying. But it's hard enough, I think as everyone knows, to hire talented security people, nonetheless talented engineers. So, really, we're just hoping for the best, and they can learn from it and move forward.

Presenter: And we had a question from Ed asking, "Any thoughts from an approach to permit self-hacking or extensions to firmware code?" Maybe that's to both of you.

Presenter: I guess I don't understand. Read it one more time.

Presenter: Any thoughts for an approach to permit self-hacking or extensions to firmware code?

Presenter: Oh, so I mean if you want to hack the firmware and it's signed, then you won't be able to load it on the ECUs. Yeah, that's just a trade-off between having verified software and being able to tinker.

Presenter: That's where you have-- there's the classic tradeoff between flexibility and security here. The

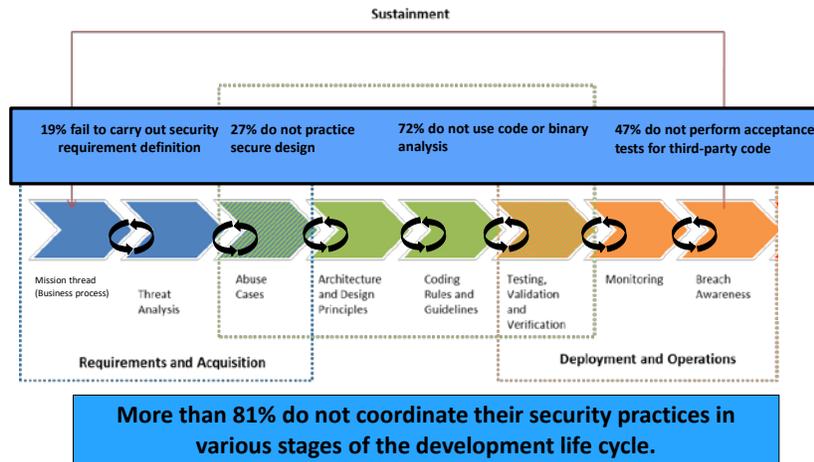
ability to download your own code as a module into a system is a favorite way to extend the functionalities for historically in the whole software world. But with it comes a whole slew of concerns about the things that we talked about, that you're operating in new environments, what kind of authentication is necessary, what authorization, how do you know what it is worth, how do you back it out. Most of the times in these kinds of high-- if you have a system that's high risk of causing physical damage, you want to be really, really careful about those kinds of things.

Presenter: Yeah, unfortunately I think people forget how much testing is done for the most minute code changes on these cyber-physical systems. So, just change itself might have an adverse effect to something else. That being said, I like doing it. But I understand why you might not want it to happen.

Presenter: Right. So, I'll just wrap up the polling question real quick. We had nineteen percent use a security requirements management tool. Forty-one percent used source code analyzers, nineteen percent, dynamic fuzz or pen testing, and other at twenty-two percent.

Room for improvement

Room for improvement



Sources: Forrester Consulting, "State of Application Security," January 2011; Wendy Nather, Research Director, 451 Research, "Dynamic testing: Why Tools Alone Aren't Enough," March 25, 2015

**016 Back to Mark. We've got about ten minutes left of you guys. So, we can--

Presenter: Sure, so just for what it's worth, here's sort of some industry studies that have been done on these kinds of things. And it turns out that about a fifth don't do any kind of requirements analysis in the security area. So, it sounds like that we may be in the same kind of ballpark here. A third really don't look to security issues in doing design and architecture. Surprisingly, nearly three quarters don't do any testing of security. Now, they do lots of other testings, a lot of functional testing. But actually doing static code analysis, or fuzzing, or things of that sort to see whether you have security issues is not well practiced. And especially in these kinds of systems

where you have a lot of third party components, nearly half don't actually consider the security issues of those components. And perhaps more starkly, almost-- no, eighty percent or more don't integrate whatever these things are across. So, even if you have someone looking at the threats, that's not passed on to the people doing the design. And as a result, they're not designing against those threats. So, you don't have-- you have a lack of communication, which makes this very hard to implement.

But again, we know better. And the point is by having this knowledge spread more broadly, we can make these systems better without having to do a lot of new invention.

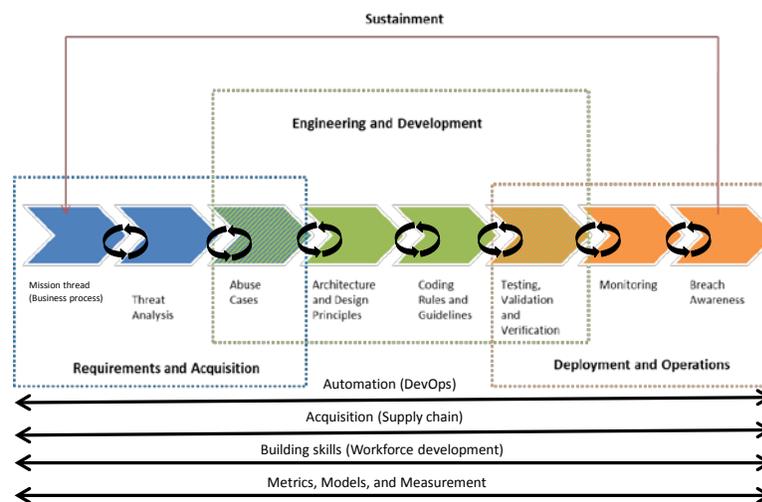
Presenter: Especially too, since sometimes implementing security measures when designing and developing for these things does hinder the design and development, code signing, for example, it's easy to deploy unsigned code quickly. But to deploy signed code, you have to have some kind of key management and self-signing service to make sure that the developer's code can actually be tested and run. So, it really does bring up complications. And I understand that. But at the same time, you want to build these in to your model instead of trying to bolt it all on at the end.

Presenter: Right, so if you understand up front that one of the threats is someone changing code or

having doctored code coming into the system as one of the threats and one of the requirements, then it starts being taken into account all the way down the stream. So, it's not, as you said, an add-on at the end where we're saying oops, we no longer know how we can do testing. We no longer have a way to have our field service do updates. And you don't know how to take away certificates and all the other issues.

Cross life cycle issues

Cross life cycle issues



**017 Now, there's some other concerns that come about in developing secure software that really cross the whole lifecycle rather than just focus on threats or design, things like how you do automation. And we've got a whole discussion of that later on in the webinar. In addition, we've mentioned a couple of times the assembly of these

systems from third-party components. That's managing the whole supply chain. Fiat was on the front end. Well, in most cases, people tend to think of that final integrator as responsible for everything, even though it may have been a Harman issue.

The same thing with airplanes, it's United Airlines that's on the front end. And then it's Airbus. And then it's whoever provided the stuff to Airbus all the way back. But if a United plane falls out of the sky, it's United who's on the hook, not Boeing as the first place that they go or whoever. Sorry, I don't mean to say nasty things about Boeing.

As you mentioned, there needs to be education that people need to understand what's going on. In some cases, you can have a Bill Gates tell the organization, "You shall do this." But in many cases, this is more ongoing professional development throughout the whole development process.

And you need to just keep track of things. You actually need to measure to see if you're getting any better. And putting that in place so that you can keep track of not just defects in quality, which people have already done, but security issues as well to make sure that in fact you've got something that's improving over time.

Q&A



**018 So, with that, we can start to move into the actual discussions on some of the individual pieces that you want to--

Presenter: Great. So, we're getting some audience questions. We're getting a number of questions asking if an archive, or the slides are available. Every presentation today is being archived folks. It's the same registration URL that you got to log in today. You can access it. Most likely the archive will be up by tomorrow. And you can catch the archived version. As for slides and materials, if you look in the download materials tab at the bottom of your console, you'll see a link to walk away with those slides now.

So, just a couple more-- a little bit more time with Chris and Mark here.

So, Mark, when you started at the beginning you talked about a Department of Transportation study. Do you have a link that we can share with the audience? Or can we send that after the webinar?

Presenter: After the webinar-- I was thinking about it. I heard-- this was referred to me from the Department of Transportation. I'm actually not sure that they did the study as opposed to referred to the study. But I do have the link--

Presenter: Okay. So, we'll get that out or we'll add it into the resource sections. Brook wants to know, "Do you know if they met MISRA in their code?" And the comment says, "Not that this would help design issues."

Presenter: Yeah, I have no clue. We never actually saw source code other than some scripts. The rest was just binary format. So, we wouldn't be able to say.

Presenter: Okay. From Brando asking, "Can you address some of the best practices to manage risk when integrating that ecosystem you talked about?" And we'll let both of you.

Presenter: So, actually there's-- the reason why I'm hesitating, I'm trying to figure out how much is going to be discussed later. I know that some of the other speakers are going to address some of those issues. So, there's a general methodology where you look at the supplier, the product, the distribution, and the operational

environments in order to evaluate third-party components. Supplier, are they well equipped to build secure software? Do they know things like a secure software development environment, do they do threat analysis, things of that sort? The product, is it signed. Is it built? Has someone actually validated? Have they done static testing on it? Have they done penetration testing on it or not? Distribution channels, do you know that the code has reliably made its way from the source to wherever it's going to be installed? Do they have an over the air kind of thing? How quickly can things be addressed? And the operational environment, which is what assumptions have been made. Am I assuming that I'm not going to be connected to the Internet? I only have physical access. Or have I taken into account that I'm going to be in these open kind of an environment and as things changed. Those are sort of the general kinds of thoughts one does in general with third party components.

When you get into open source, there's more issues about provenance and making sure that you have a catalog of what components you're using, and how to find them later when you need to update them. So, there's actually a lot of material that's been written up about this.

Presenter: Terrific. And just the last one in the queue here asking Mark, I think you mentioned a figure of hundreds of millions of dollars. Neil

wants to know, "Where did the figure hundreds of millions of dollars to correct the issue come from?"

Presenter: So, in a hundred million dollars that FCA was affected, that they had to set aside in order to do the recalls. And I have the-- I don't recall whether it was in an SEC filing or something of that sort. But I can also give you the reference for that as well later.

Presenter: Okay. So, the queue's now empty. We're running a little bit ahead of schedule. So, we're going to prepare for our next talk. I just want to thank Chris again for participation today. Anything you want to say about SummerCon? We know that's your conference coming up. We'll give you a quick plug there.

Presenter: Yeah, it will be in New York in July. Go on there, fifteenth and sixteenth, head on up. It's a good time.

Presenter: Mark, thank you again. Excellent presentation. Any closing comments before we welcome in Chris?

Presenter: Just a plug, if you like.

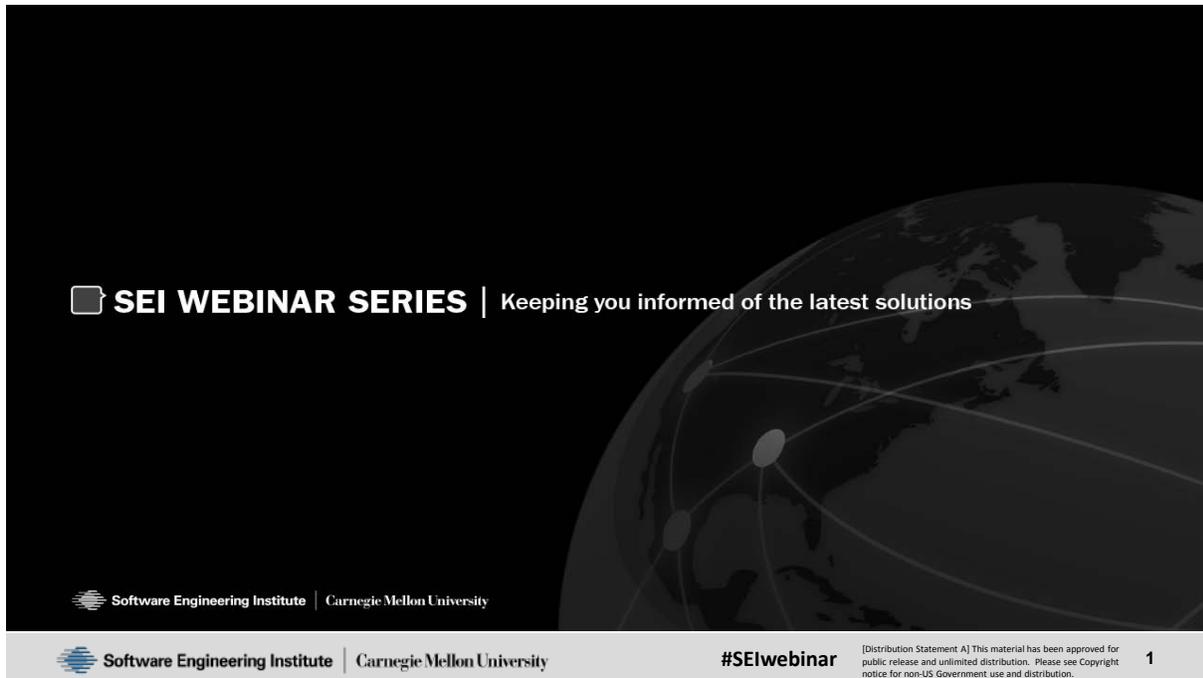
Presenter: Sure.

Presenter: We have put together a white paper on cybersecurity for these kinds of systems in the automotive space.

Presenter: And that is actually in the download section--

Presenter: So, that's one of the downloads. And I know that the reference in particular for the Chrysler comment is in that paper as well.

SEI WEBINAR SERIES | Keeping you informed of the latest solutions



SEI WEBINAR SERIES | Keeping you informed of the latest solutions

Software Engineering Institute | Carnegie Mellon University

Software Engineering Institute | Carnegie Mellon University

#SEIwebinar

[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.

1