Cyber-Vulnerabilities in Aviation Today - Robert F. Behler

Table of Contents

Cyber-Vulnerabilities in Aviation Today

**001 Shane: And welcome back to the SEI virtual event, CERT Alignment with Cyber COI Challenges and Gaps. We're back for the afternoon session. And our afternoon keynote talk is from SEI deputy director and chief operating officer, Mr. Robert F. Behler. Before joining the SEI, he was the president and CEO of SRC.

Prior to his work with SRC, Behler was the general manager and senior vice president of the MITRE Corp. He served thirty-one years in the United States Air Force retiring as a Major General in 2003. During his military career, he was the principal advisor for the command and control intelligence surveillance and reconnaissance to the secretary and chief of staff of the U.S. Air Force and the Deputy Commander for NATO's Joint Headquarters North in

Norway. An experimental test pilot, he flew more than sixty-five aircraft types, including the SR-71 Blackbird and the U2, during his Air Force career.

So, Bob's talk will introduce the idea of heavy reliance on complex software in aviation, how it open avenues for cyber intrusion, and that we need to test for cyber vulnerabilities in aircraft in operation. So, now I'd like to turn it over to Mr. Behler. Bob, all yours.

Presenter: Thank you, Shane. So, my talk will actually bring the conversation up a level to the operational employment of software-defined systems. And I'm going to talk about aviation in general. One because know a lot about aviation; I spent most of my career in that field. But also, I think it's an area that's ripe for investigation and research. I don't think enough attention is being played right now into the aviation world.

There's lots of information in the media. Just yesterday, a major airline was shut down because someone hacked into the aircraft mission planning system and put airplanes on the ground. We had a self-defined cyber security professional that said that he could into aircraft control systems by flying-- while the airplane was flying. So, there's a lot of attention on that particular field.

And I've been giving this talk pretty much around the world. I just came

back from Switzerland last week and talked to a group that's actually testing aircraft for everything from aerodynamics to possibly even software.

So, The first picture I have up there kind of sets the tone for my conversation. This is not a picture taken from 15 miles up where I used to fly, but about 10 thousand feet over the Air Force flight test center out in California. And I have that as a backdrop for this briefing because what I'm going to talk about is really two things. One is how to demystify this whole conversation about cyber-vulnerabilities in software-defined systems and also to show that, in the end, when we have a full-up operational system, it's my feeling and my experience in tests, that we need to test an operational representative system in an operational environment. That's kind of the quote, the definition of an operational test and evaluation. I think we need to know it's time-- Our aircraft have become so software reliant that it's time to start doing that also with aircraft.

So, let me just get this briefing started. The title of this presentation will be Cyber-Vulnerabilities in Aviation. Okay, I picked it because of the reason I mentioned earlier, but that could also be software vulnerabilities in the automobile industry, software vulnerabilities in space, software vulnerabilities in ships, software vulnerabilities in the apartment building I live in in

downtown Pittsburgh because it is software defined. And I'm going give you some examples in aviation. But I'm going to show you a real-world example in the automotive industry later on to demonstrate one of the concepts I want to talk about.

So, In my history in flight tests where I spent a lot of time doing it, when I tested software and aircraft, it was more to test the functionality of the software. At that time, in the early '80s and '90s, we were looking at functionality. We weren't really thinking about the vulnerability aspects. So, my point is--

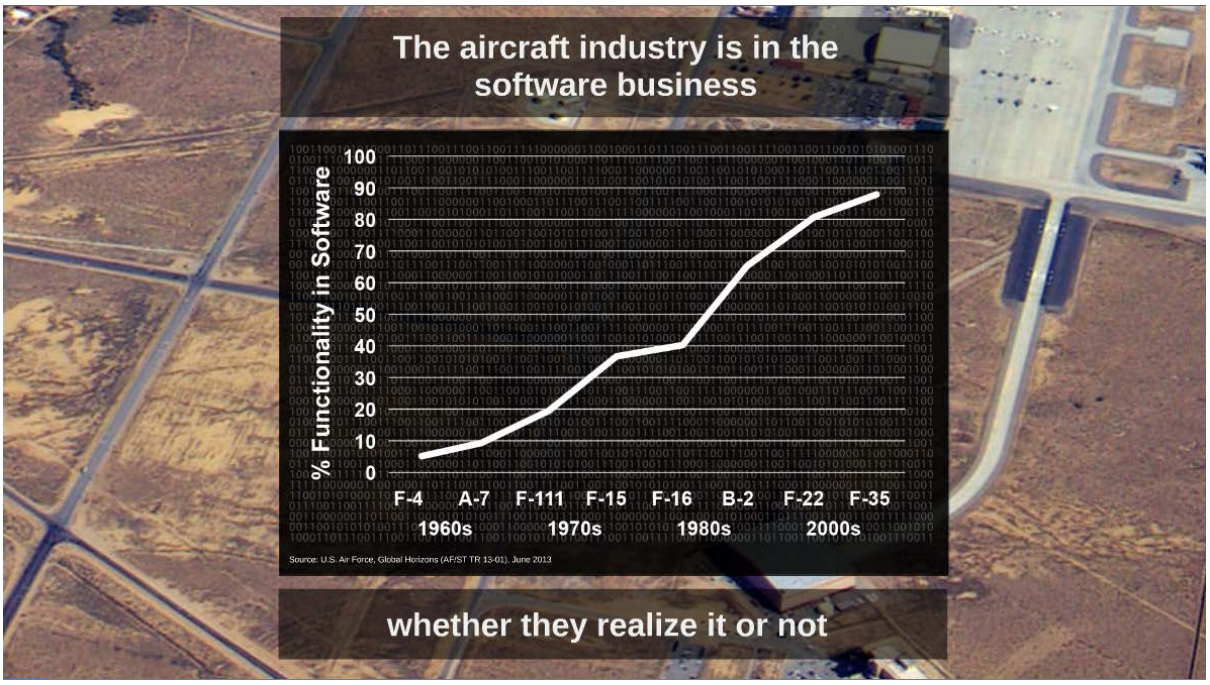
The aircraft industry is in the software business

**002 that the aircraft industry is in the software business whether they realize or not, or whether they want to be or not. And here again, I can add any other industry to that. And

the automotive industry actually has an order of magnitude more software in their vehicles than I have in some of the airplanes I've flown.

So, How did we get here?

Whether they realize it or not



**003 In aviation, look back in the '60s, and these just happen to be representative aircrafts, some of which-- many of which I've flown, not back in the '60s. But I flew them up in the '70s and '80s and '90s. But I flew some of these aircraft that had very little functionality on software. For example, the F-4 Phantom, it basically had about five percent software functionality. It was mostly for navigation.

One of the airplanes I flew, the A7, started getting more software. And that's one of the first airplanes I flew that actually was a

software-reliant aircraft. We had a test airplane that we could put software in it, digital software, and make the aircraft basically define how we want it to fly. And it was an experimental aircraft, but it really led the way all the way up the path here that you'll see.

As the airplanes went up, and The F-16 was the first aircraft that had, what we called in flight tests, relaxed static stability. And what that means is that without the computers, the aircraft just wouldn't fly any safe way. And we thought we really had a high-speed processor when it could process about 30,000 bits of information in a second in the processor. Today, if you look up at the F-35 in the far end, almost 90 percent of its functionality is derived from software. And it's processing about a trillion bits of information a second. So, it is really high speed in many, many ways.

Software is good. Don't get me wrong. I'm not going to say that software is not good. But there are some issues we need to address. And I'll get to that here shortly.

F-35

**004 So, Let's look at that airplane. This is the most advanced fighter aircraft in the world. They call it a fifth generation airplane; It's the F-35 Lightning 2 is what the Air Force calls it. Lots of software code.

Now, I've talked to the contractor. I've talked to the chief test pilot on the program. I've showed them this slide. And fifteen million lines of code on the aircraft. That's an estimate. Nobody really knows. In the software business, we really don't address software lines of code. We talked about function points in software. But in the crowd I talked to, when you compare airplanes to airplanes, software lines of code is probably a good way to address it.

But if you look at that, 15 million lines on the aircraft alone and then

another 15 million lines on the ground support equipment. An airplane like this is not like when you go out to see an airliner, they have maybe some ground support people, people putting fuel on the aircraft. And airplane like this, when you land, it's basically plugged in to look at all its functionality. So, there's lots and lots of software on there.

And one thing I like to show people, it sounds like a lot of code. But let me just hold up this little device in my hand which we all probably have. This has about 15 million lines of software code in it. So, it's pretty neat because it doesn't weigh very much. So, in an airplane, if you want to reduce weight, software is the answer because of the functionality. But it doesn't take up space. And it doesn't take up a lot of weight. And people who build airplanes love that. So, software is good in that aspect right there.

This is a cockpit

**005 Here's another good thing about software. This is the cockpit of that brand new airplane that I'm talking about, the F-35. Software has been able to reduce the complexity in the cockpit. Airplanes that I flew it's just filled with gauges and instruments. And it just really, really complex. By using software, it's really taken out a lot of complexity. As a matter of fact, the airplane is very easy to fly because the software behaves very, very nicely. And this allows it to happen because of all the software.

And let me just add one more point. In software-defined systems, whether it be my apartment building or aircraft or cars, it is the building material of choice today. Back when we first started flying airplanes, across, of course they were fabric covered. And then something was

developed here in Pittsburgh called rolled aluminum, and then All-Clad. And aircraft from that point on were made out of aluminum. Today, it's mostly composites. But beyond that, software today is the building material of choice.

So, let's look at the airplane itself. So, People think airplanes flying around are pretty invulnerable to any kind of software intrusion or cyber-vulnerabilities.

Potential Cyber-Vulnerability Pathways

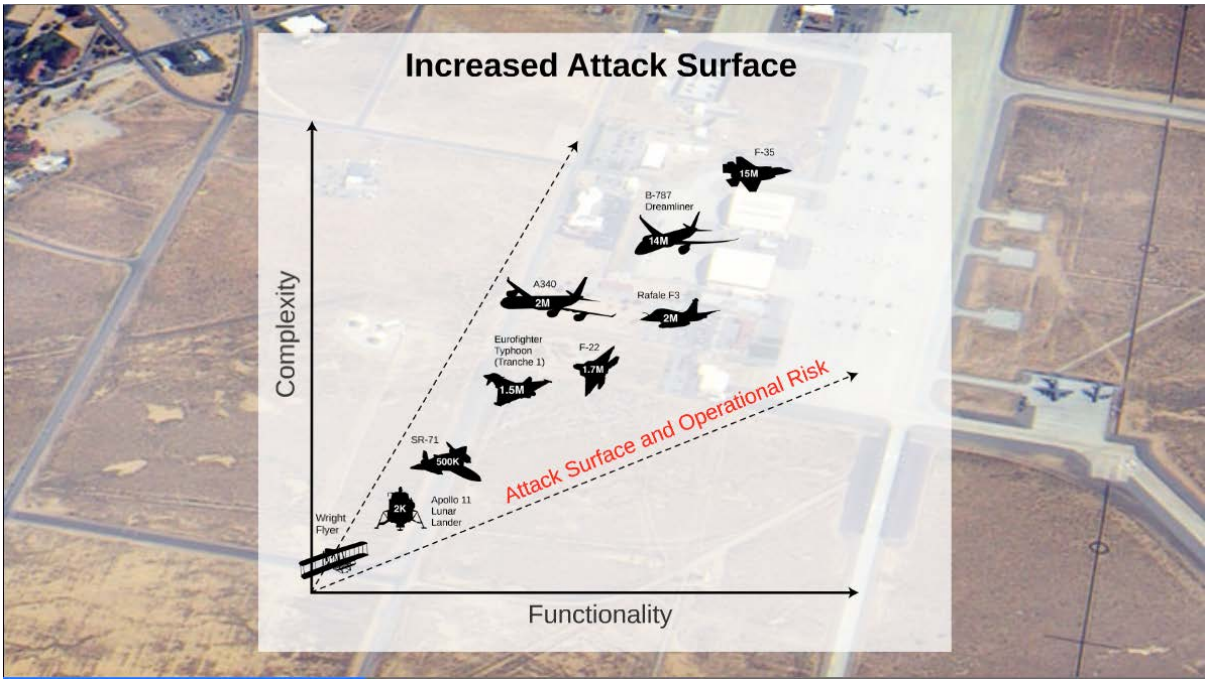
**006 But on this particular airplane, I just picked out a couple places where we have apertures on the airplane that multiple RF signals could possibly do some intrusion, everything from its particular antennas for its radio system, whether it be UHF, SATCOM, and so on. There's an antenna there.

And if there's an antenna, there's an opportunity for extraneous RF signals to get into it.

The whole airplane, by the way, is a distributed aperture system that basically gives the pilot complete situational awareness. That itself is a large aperture, could cause problems. The multifunction advanced datalink, the MADL, has also it.
Those sort of systems are pretty much free from intrusion as we know today. But I'll go a little bit further in this conversation and talk about the existential threats to cyber.

And of course, as I mentioned earlier, the ground support equipment, ground support equipment has lots of people who touch it, lots of contractors that work on it. So, every place there's a touch, there's an opportunity for some sort of vulnerability.

Increased Attack Surface



**007 It'is just a representation of aircraft. And as you go up in the up to the upper right hand corner, really aircraft that has more and more software in it. So, I'm going to start in the beginning of aviation, 1903. The Wright Brothers out in Kill Devil Hills, they didn't have any software in that airplane. I can guarantee you. But it did fly. And it flew very well and not very long, but it did prove that man-powered flight was possible.

And then we first really started using software in aviation in about 1969, '68. And The first lunar lander, Apollo 11, landed on the moon. It had only about 2,000 lines of machine code on the vehicle. And that machine code was to control the descent jet engine, rocket engine, as it was landing on the moon. But I will also add that there also was another computer in this lunar lander. And it

was the ultimate computer. And that was Neil Armstrong. He was able to solve complex problems very quickly. And that's one good thing that humans can do, solve complex problems. Computers solve complicated problems pretty well.

And then in the same period of time, the airplane I flew a lot that Shane mentioned, the SR-71, hard to believe that early '70s, fastest airplane in the world, still the fastest airplane in the world. And it was an analog airplane. And When I flew the aircraft about 15 years after its initial operational capability, there was a problem with vanishing vendors. That means that the old vendors that were supplying the analog parts for the computers on the aircraft were going out of business. So, at that time, Lockheed Martin had to digitize the airplane.

So, When they digitized the airplane, it wound up with about half a million lines of software code. And like I mentioned on the F-16, this had relaxed static stability, which meant if the computers were not working when I was flying three times the speed of sound, it was not going to be a good day. And as a matter of fact, there were three digital computers that controlled the ramjets on the air vehicle, the stability control, and navigation. And if one of those computers [went] out, it was still okay. But if two or more computers were gone, that's when you had to jump out of the airplane. Thank god, nobody had to do that.

I could put the space shuttle there. The space shuttle had about half a million of codes also.

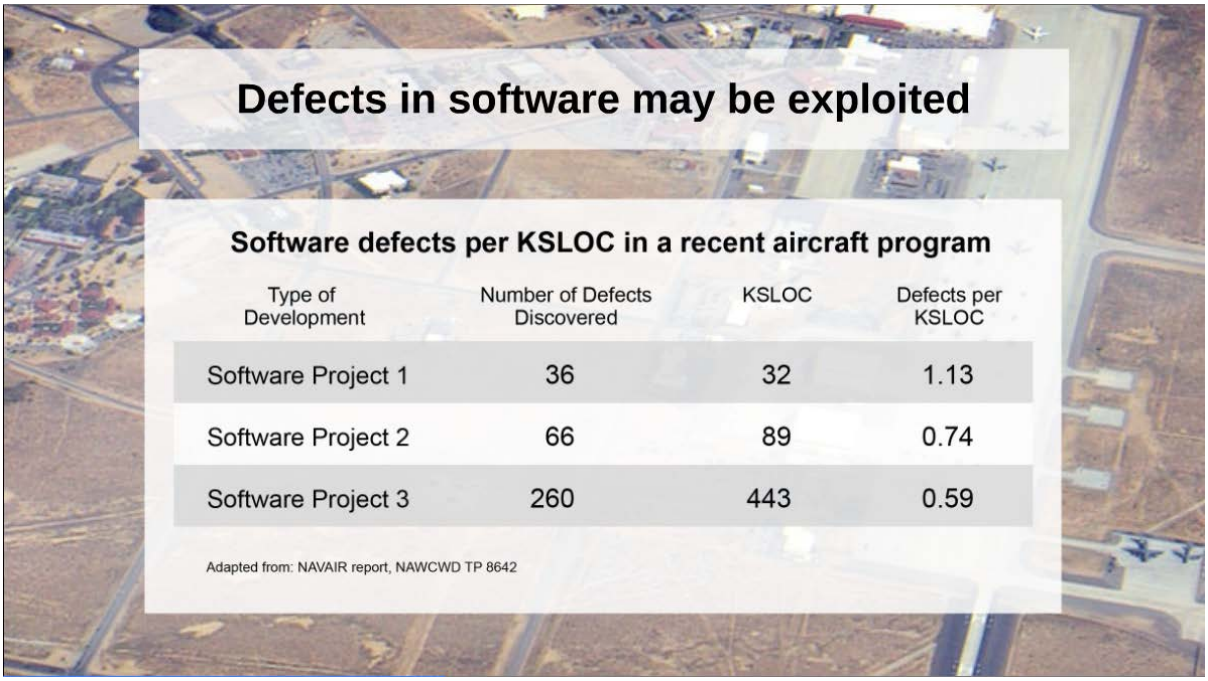
Not a lot, but still enough to be able to control that aircraft.

I've got some other airplanes on here that look-- for example, some Euro fighters, million and a half lines of code. But as you march up to the top, the aircraft with the most software reliance is that new aircraft, the F-35 with approximately 15 million lines of code.

So, what's the big deal? The big deal is, first of all, we like software because it gives us a lot of functionality. But the more code you have and the more functionality you have, the more complexity you have. And that complexity is the area that is bothersome because it actually opens up the attack surface for an intrusion by a cyber-attack. And I'll kind of give you some details about that in a second.

So, the more we march up to the upper right hand corner, the more we open up the attack surface. And that's the area that I think we ought to make sure we test to make sure that the vulnerabilities aren't there. And if the vulnerabilities are there, we know how to protect them. But make no mistake, no software is written without defects.

Defects in software may be exploited



**008 And that causes problems. Here are some examples of some programs that we have worked on here at the Software Engineering Institute. This is a recent naval program that looked at some projects in the operational flight controls of an aircraft. And this software was developed to a gold standard.

If you look at the project number three up there, about point six defects in a thousand lines of software code. Now, that doesn't sound like very much. But if you extrapolate that up to 15 million lines of code, I mean you'll have 9 or 10 thousand lines of defective code. Now, all defects are not exploitable. But many are exploitable.

The one that's been used with a technique that's been used more than anything else in the history of

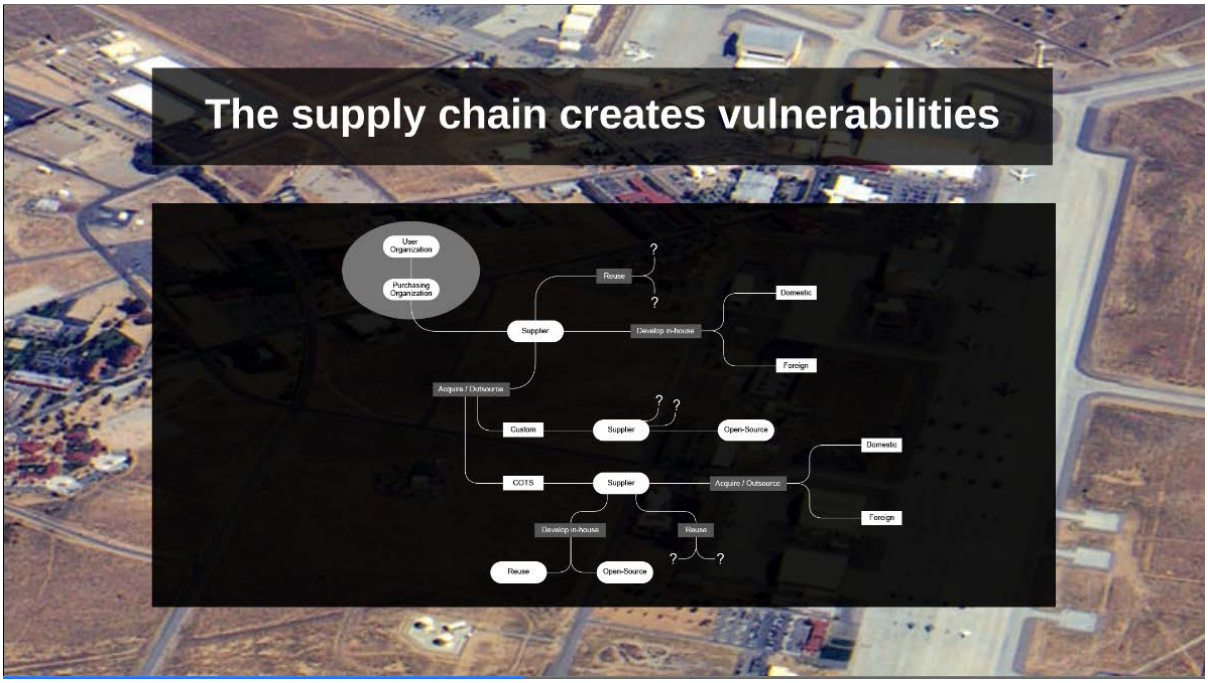
software in weapons systems and any other IT systems is called buffer overflow. Software is getting better. And it's not as used now as much as it was before. But basically, it's allowing the buffers of the system to be overflowed by putting a lot of extraneous information in it. Once you break the system, then you're allowed to put some bad code in it, malware, which causes problems. I will show you a real world example of buffer overflow in a minute, not on an airplane unfortunately, but I will show you a pretty good one here shortly.

So, Let me talk about defects for a second. Defects in a commercial world are called bugs. And those of us who have lots of apps on our smartphones, you'll see that you'll-- every once in a while, it will say you have a new version. Update the new app. And if you ever go on there and look what the versions says, it will say some capability, but it says we're fixing bugs. Well, fixing bugs in the commercial world and our world means they're trying to patch defects in that code. So, that's an important thing to remember.

We had one of our fellows here at the SEI, Watts Humphrey, actually the father of the Software Process Development. And he said, never call a defect a bug. Call it a time bomb because it's there, and somebody can exploit and then cause you big problems.

So, Also we have a problem in the supply chain.

The supply chain creates vulnerabilities



**009 In an example like
the F-35, in avionics alone, there's
approximately 26 contractors
who work on that software. They're
all supposed to comply with a
standard for developing the aircraft [set]
by the prime contractor.

But everywhere you touch old software
and new software, there's a chance
for exploitation. That F-35, by the
way, has about five percent code
that's legacy code off of the F-22, an
older aircraft. And that was written in
a code that we all know and love,
Ada '83. So, that code was written
20 years ago, about five percent
of it in a new airplane. And at that
time, I can guarantee you that the
software was not written to look at
cyber-vulnerabilities, only for
functionality. So, I'm concerned also
with the ability to inject malware in
the supply chain.

Let me give you an order of magnitude of how big this problem is. SEI, this organization has been around for 30 plus years. And about four years after we formed in 1988, a gentleman named Robert Morris released a worm into the nascent ARPA network. Only about 60 computers on the system, but He supposedly said he was just trying to map out the whole thing. But it caused a massive denial of service for most of the system. That was called the Morris Worm.

Today, we keep a repository of malware. And it's measured in the hundreds of millions, approaching a half a billion malware instances that we have cataloged. And Startling is that every week, about a hundred thousand new cases of instances of malware surface. Now, these are all not brand new. They could be variants of things that have been out there before. But that's a lot of malware to try to keep up with. So, I think that's another thing to show some concern for us.

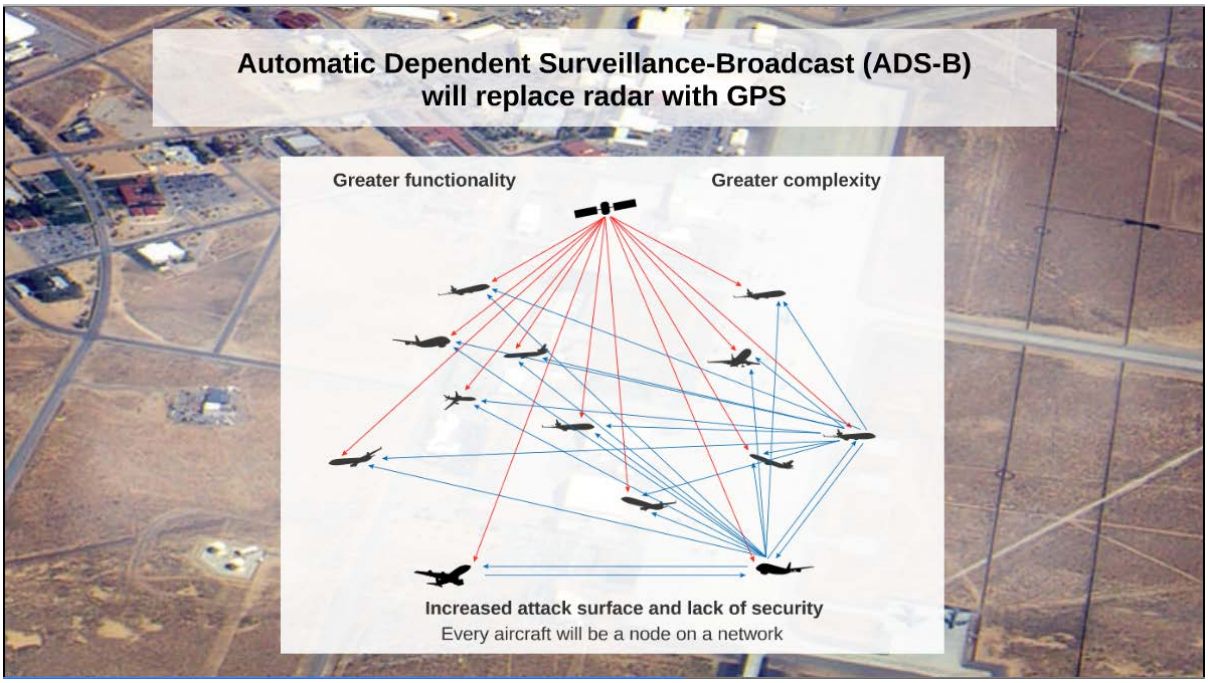
Now, let's go back to aviation a second. Not only are our aircraft complex, the aviation works in a very complex environment.

Aircraft operate in a very complex environment

**010 This is just a representation of the National Airspace Air Traffic Control in the United States, literally tens of thousands of airplanes are flying every day in this national air space. You can see up and down the eastern seaboard where it's most dense. Anybody that's flown through Newark, LaGuardia, JFK, Philadelphia, know that one ripple at that point causes delays across the United States.

So, in order to improve the air traffic control system, FAA is coming up with a new concept that allows aircraft to talk node-to-node. About 80 percent of the airplanes flying in this national air space have a transponder on their air vehicle--

Automatic Dependent Surveillance-Broadcast

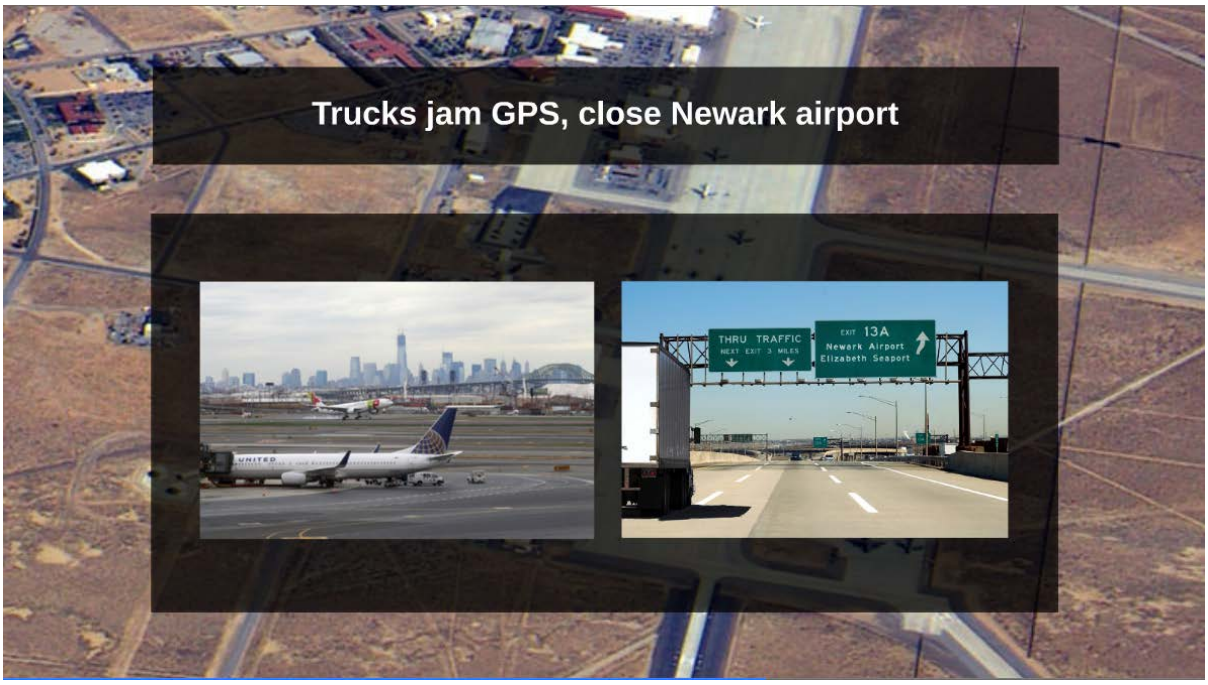


**011 that's called an automatic dependent surveillance broadcast system [ADSB]. It's really just a transponder to allow aircraft to communicate to other aircraft, say where they are, where they're going. And so, they can basically avoid each other.

In the not too distant future, about 2020, all the aircraft in our air traffic control system, if ADSB works properly, will be on this network. And we'll do away with the ground radar systems.
So, remember that picture I showed you about the aircraft and opening the aperture for vulnerabilities? This is really opening up a lot of opportunities for vulnerabilities. The ADSB itself was never designed to be cyber-secure. It was just made for functionality. There's been organizations around the world that have been able to come up with ways

to inject RF signals into this ADSB. So, this is an area that's ripe in a full up system to be able to research it and also to test it.

Trucks jam GPS, Close Newark airport



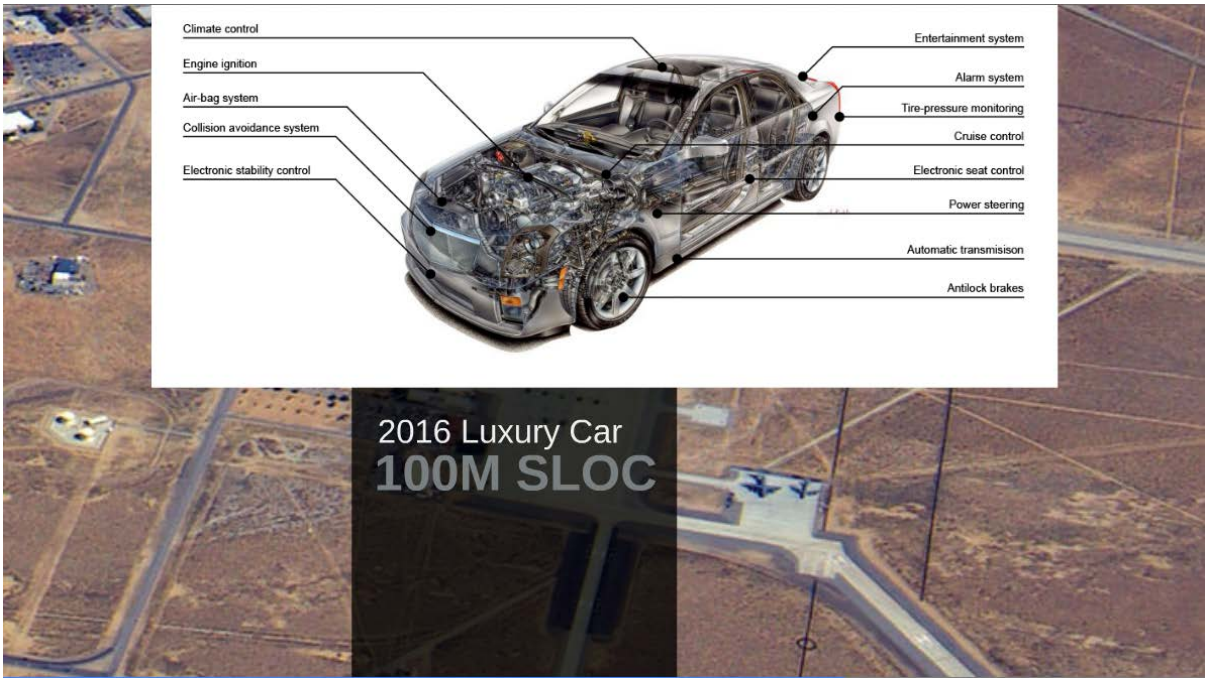
**012 Let me give you an example how easy it is to disrupt that network. This is an actual situation that I got involved in to try to resolve what happened here. In Newark airport, there's a GPS controlled landing system. And pilots were reporting that every once in a while, the system would go off the air. and Flying an approach in the weather as a pilot, any of you who are pilots know, when you lose your ground control for your landing system, you have to take action and go around. It's really disconcerting, especially in the commercial world.

And the LaGuardia maintenance could not really figure out what was causing the GPS to [go] off the air. So, After going out there and doing some research, what we discovered, the picture on the right, is that there was a trucker that basically had a GPS jammer in his truck. And he was going very close to the airport. This highway's very close to it. And it was jamming that GPS transmission signal.

He wasn't doing it to jam the GPS control. He was doing it to jam his own GPS on his truck so his employer could not tell where he was because in the trucking industry, you can only be driving so many hours a day. They have to know where you are. So, he had a jammer that was jamming his GPS. And so, his employer didn't know where he was. Unfortunately, he was also jamming the GPS transmitter on the airfield.

Now, here's the rest of the story. That GPS jammer was about the size of a package of cigarettes. He bought it on the Chinese web for 14 dollars, of course completely illegal. But that's how easy it is to get something to disrupt our entire system in GPS.

2016 Luxury Car



**013 So, let me now go to an example that I can give some real actual situation that occurred. So, This is a new car. I'm not going to tell you what kind it is. But there's about two dozen computers on that car that operate everything from basically running the air conditioning system, power steering, monitoring the pressure in the tires, to controlling safety critical systems like brakes. So, I'm going to show you what a buffer overflow happens in real world and what caused it, what caused it to happen, and the results.

By going through one aperture on this vehicle, that is their cellular communication system, [was] able to get into the car's computer systems and make it do something really bad. So, here's a little clip here.

60 Minutes

**014 Lesley Stahl: This is a
regular new car.

The masking tape is
only there because we agreed to
obscure its make and model.

Kaufman: We'll give them the
illusion they control the car for now.

Lesley Stahl: Kaufman has been
working on this for five years with
multiple research teams.

Kaufman: You want to hit the
fluids?

Lesley Stahl: Oh my gosh.

Kaufman: There we go.

Lesley Stahl: What's that? What's
that?

Woman 1: Yeah, the windshield wiper fluid.

Lesley Stahl: No, wait. So, this is something that a hacker--

Woman 1: That's right, a hacker-- obviously, you didn't turn on the windshield wipers.

Lesley Stahl: I did nothing.

Lesley Stahl: Using a laptop, the hacker dialed the car's emergency communication system and transmitted a series of tones that flooded it with data. As the car's computer tried sorting it out, the hacker inserted an attack that reprogrammed the software, gaining remote control.

Lesley Stahl: Oh my god. They're doing that?

Woman 1: They're doing the horn.

Lesley Stahl: They could control the gas, the acceleration. They could control the braking.

Woman 1: That's right.

Lesley Stahl: And they could do this from anywhere in the world.

Woman 1: So, just stop at the cones here.

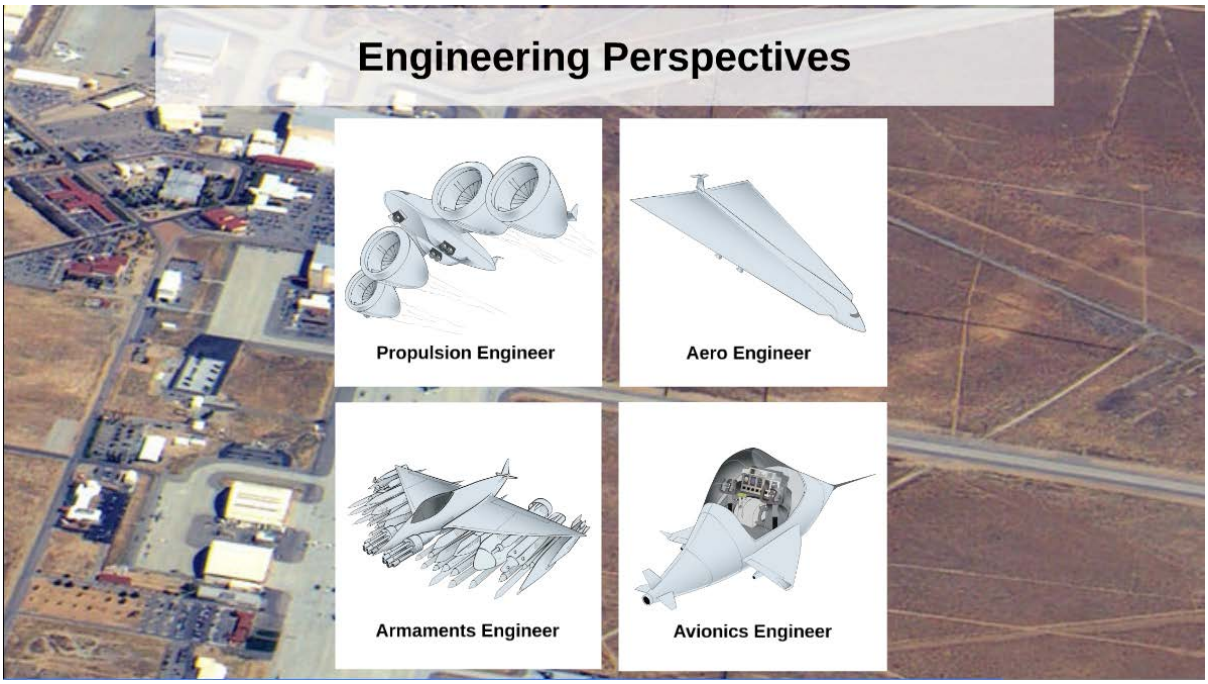
Kaufman: She thinks she's going to be able to stop right at those cones. Let's make sure that she can't. She's going to drive right through them.

We'll have complete control of that brake.

Man 1: All right, here we go.

Lesley Stahl: Oh no, no, no. No, no. I cannot-- oh my god. I can't operate the brakes at all. Oh my word. That is frightening. That is frightening.

Engineering Perspectives 1



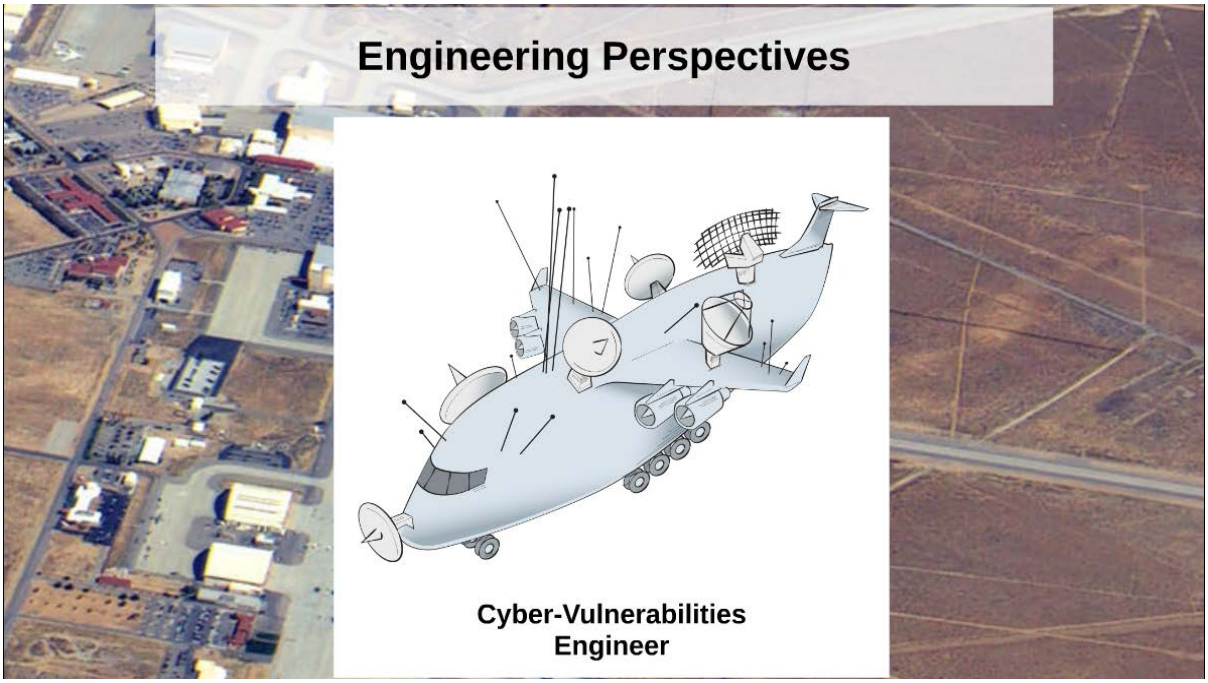
**015 Presenter: Yes, it is. It is frightening. And that's one really simple example of a buffer overflow. And it's done with an RF signal through the antenna into the brake system and disconnect it. So, my extrapolation is that we have a lot of apertures on aircraft. And we have to make sure that we test accordingly.

So, I'm an engineer. And I went through the Air Force test pilot school to learn how to be a test pilot. And

one of the things I remember is that engineers have different perceptions. So, for a propulsion engineer, everything looks like a big engine. From an aero engineer, everything looks slick, armament engineer, all weapons, and for avionics, a big cockpit with lots of instruments.

I added a new one to this. And that is, for a cyber vulnerability engineer--

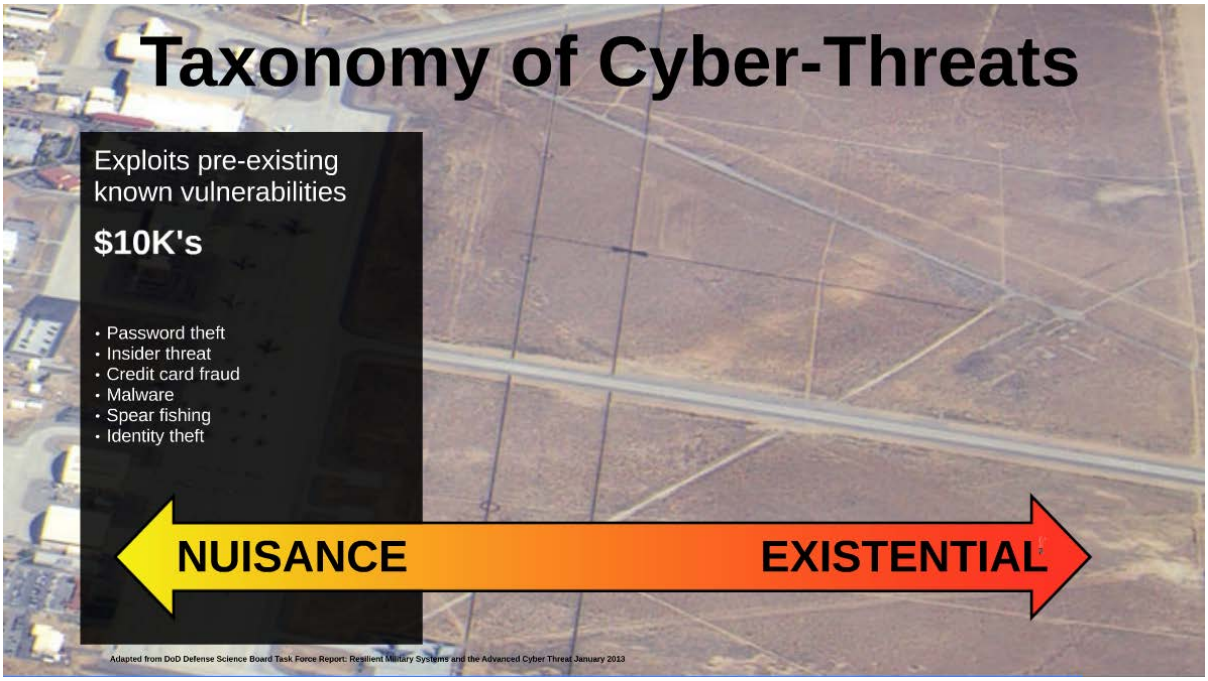
Engineering Perspectives 2



**016 --That's what an aircraft looks like, all those apertures and all those opportunities.

So, I've got a few minutes here--

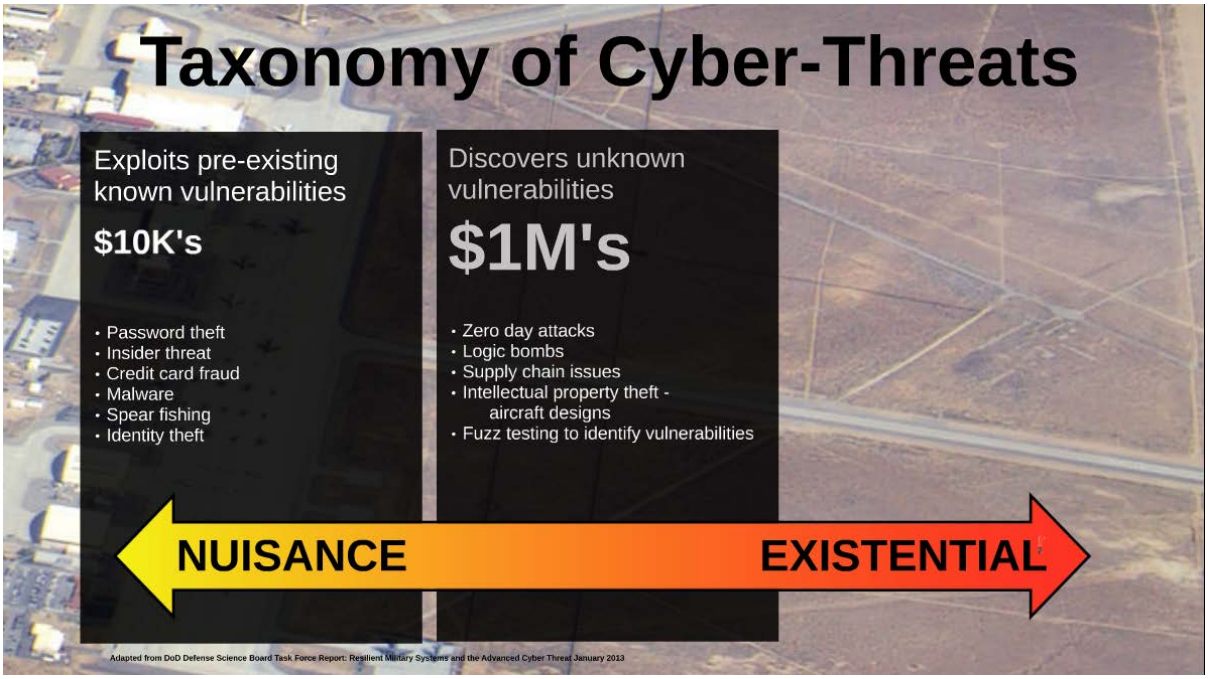
Taxonomy of Cyber-Threats 1



**017 And I want to talk about the taxonomy of cyber threats. They go from a nuisance all the way out to the existential threat, which is the one that gives me most concern. So, at the lowest level, this is the conversation we normally have when we talk about cyber-vulnerabilities, people hacking in, stealing your password, trying to get into your bank accounts, and so on and so forth. This is exploiting known vulnerabilities in software. We know that those defects are out there. And we know they can be exploited. For something as small as a few hundred dollars to about 10 thousand now, you can get a piece of malware to do just about anything you want to exploit those kind of vulnerabilities off of a dark web.

The next level up is they're spending more money.

Taxonomy of Cyber-Threats 2

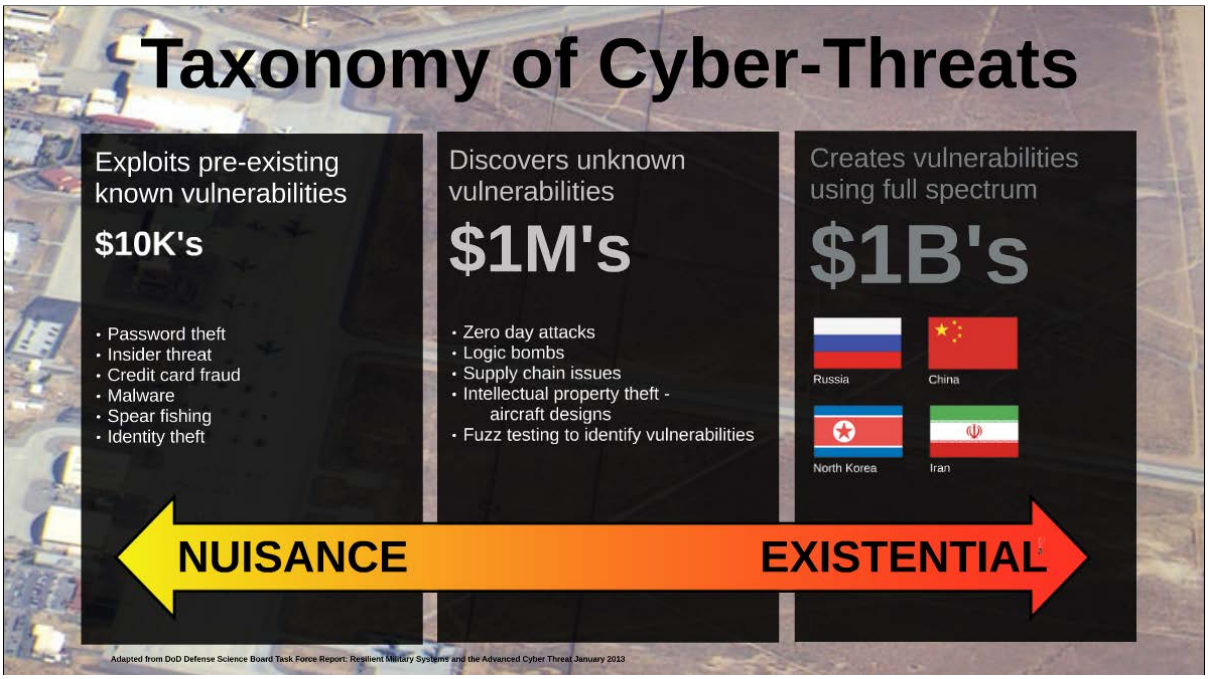


**018 And not only are they looking for the known vulnerabilities, they're discovering the unknown vulnerabilities. This is where the criminals hang out. They're being able to go to exploit software, be able to, with a few million dollars, get into software and figure out what the problem is.

There are tests that we can do. And here's some of the research we're doing at CMU and SEI is to fuzz up the software and find those vulnerabilities and try to patch them.

But the one that's most concerning to me is the existential threat.

Taxonomy of Cyber-Threats 3



**019 There are some that don't believe we have existential threat. But we really do. Nation states are spending billions of dollars, not trying to discover the known vulnerabilities or the unknown vulnerabilities, but they're actually creating vulnerabilities. And that's the ones we have to pay attention to.

And I'll just go here to The director of national intelligence--

James Clapper

**020 Jim Clapper said this. This is a problem. And those are the big actors. And we have to be really careful.

So, as I leave now, I want to just leave you with a thought. And that thought is--

Software-reliant platforms need to be tested

**021 When we have a full-up software-defined systems, cars, airplanes, boats, satellites, we need to test them in an operational environment. And we need to stress them to the limits of the software. And the thing that I always look at, and this is what I learned when I was at test pilot school, is we test to minimize the maximum regret. And we don't want to have maximum regret when we're flying airplanes.

So, Thank you for your time. And I appreciate your interest in this subject.

Shane: Mr. Behler, thank you for your excellent presentation. We appreciate your time today.

Carnegie Mellon University

Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

Copyright 2015 Carnegie Mellon University

CERT | Software Engineering Institute | Carnegie Mellon University
CERT® Alignment with Cyber COI Challenges and Gaps
SEI Webinar
© 2015 Carnegie Mellon University