Assuring Trustworthiness in an Open Global Market of IIoT Systems via Structured Assurance Cases

Authors:
Robert A. Martin
Senior Principal Engineer
The MITRE Corporation
ramartin@mitre.org

Ken Modeste
Director
Connected Technologies, UL LLC
Ken.Modeste@ul.com

INTRODUCTION

This paper proposes key elements of a process for supporting an open global marketplace of trustworthy Industrial IoT (IIoT) Systems. We offer that in such a marketplace, creating, exchanging and integrating components that are trustworthy as well as entering into value-chain relationships with trustworthy partners and service suppliers will become common if we can provide a method for explicitly defining what is meant by the word "*trustworthy.*" The approach in this paper leverages Structured Assurance Cases[1,2,3] to explicitly identify the detailed requirements "about what is needed to know about something for it to be worthy of trust based on the risk associated" and to do that in a methodology that is scalable to differing sets of hazards and environments; and is applicable to most sectors, domains, and industries.

Organizations Will Require a Mechanism to Measure Trust in Their Supply Chain

Questions about trustworthiness that need to be addressed are, what does it mean to those involved, how can they measure and specify the different aspects and requirements of the trustworthiness[4,5,6], and how can the fulfillment of those requirements be captured and conveyed to others and then combined into systems, components and value-chains.

SOFTWARE-ENABLED CONNECTED MICROELECTRONICS

With the advent of the internet of things and the continued progression of micro technology and software-enabled connected microelectronics (SECM), addressing the security assurance of the individual components of a system is becoming more and more prevalent.

In 1976, for example, there was no software in a Chevy Vega because there were no microelectronics. However, over the subsequent years many of the critical functions of the car moved from physical connections to software and networked

[1] ISO/IEC 15026-2 Assurance Case, 2011, https://www.iso.org/obp/ui/#iso:std:iso-iec:15026:-2:ed-1:v1:en

[2] Open Group Dependability Through Assuredness (O-DA), 22 Jul 2013, https://publications.opengroup.org/c13f

[3] OMG's Structured Assurance Case Metamodel 2.0 (SACM 2.0), March 2018, https://www.omg.org/spec/SACM

[4] Industrial Internet Consortium, "Industrial Internet of Things Volume G4: Security Framework*," IIC:PUB:G4:V1.0:PB:20160926,* (2016), https://www.iiconsortium.org/IISF.htm.

[5] Industrial Internet Consortium, "Industrial Internet of Things Volume G8: Vocabulary," IIC:PUB:G8:V2.0:PB:20170719, (2017), https://www.iiconsortium.org/vocab/.

[6] NIST Interagency Report 7755 Toward a Preliminary Framework for Assessing the Trustworthiness of Software https://www.nist.gov/publications/toward-preliminary-framework-assessing-trustworthiness-software

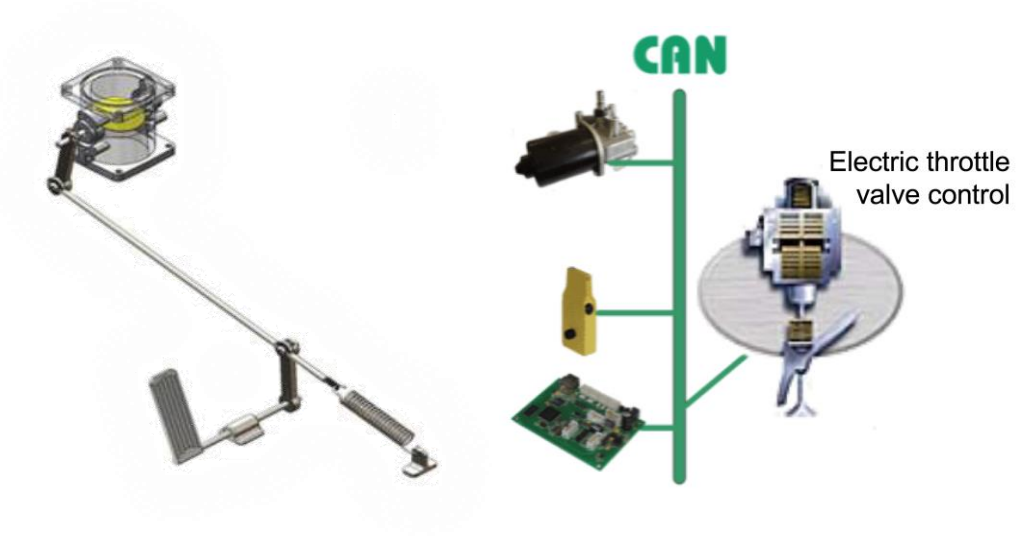

Figure 1: Physical to Software-enabled and Connected Throttle-to-Accelerator Pedal Transition

ones. For instance, in 1976, the connection from the accelerator pedal to the carburetor was a physical rod or cable.

Now, as illustrated in Figure 1, it is a sensor in the pedal that is then connected through a communication bus to an electric throttle valve controller that interprets and conveys the actions from the pedal to the throttle. There is no longer a physical connection.

That same trend of moving from physical to software-enabled connections was paralleled by a change in the way vehicles wired their devices as shown in Figure 2. Back in the 1976 time-frame, there were actual physical wires going between each item when any two needed to communicate, control one another or provide power.

A bus, a communication network, is actually a couple of connected computer or microelectronics parts running software. As shown in Figure 3 below, these pieces of microelectronics running software are needed for signaling, with memory and the CPU for processing of instructions: A significant advancement that now creates new opportunities for measuring and qualifying trust.

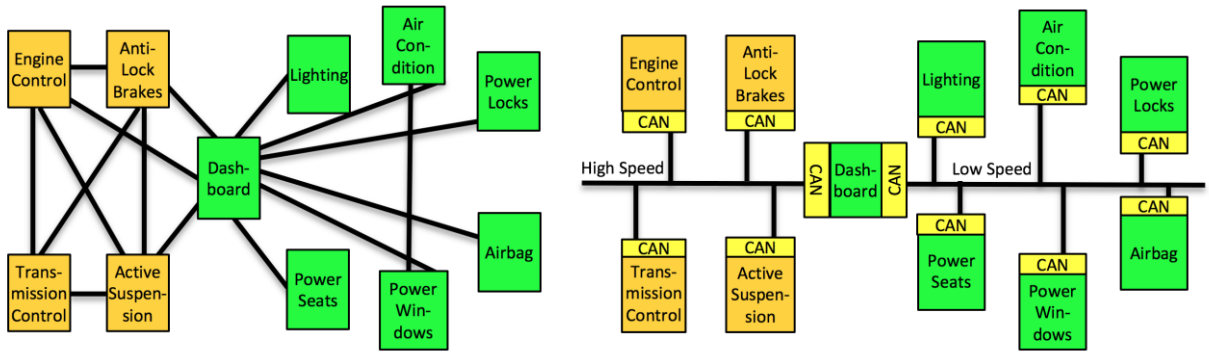

Figure 2: Point-to-Point Wiring to Bus Connectivity Transition

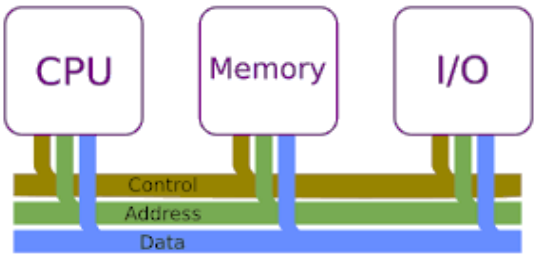

Figure 3: Connected Software and HW (microelectronics)

The same migration of physical connection and control to a bus structure has taken place for all of the critical functions in vehicles. As shown in Figure 4, most of an automobile controlling functions are all now enabled through software, over buses.

Additionally, there are many software-driven networks, both for high speed, low latency type of activities, and for more relaxed timing requirements and things that are focused on the occupants of the car. The Controller Area Network (CAN), the Media Oriented Systems Transport (MOST), Ethernet, the FlexRay Consortium's FlexRay, and Local Interconnect Network (LIN) are examples.[7]

With the controls now being software enabled with network connectivity as shown in Figure 5, there are several external attack surfaces where the car's systems are now accessible from outside the vehicle. If we look to the future, where vehicle to vehicle communications telematics and other enhanced capabilities are coming[8], that attack surface will grow even further.

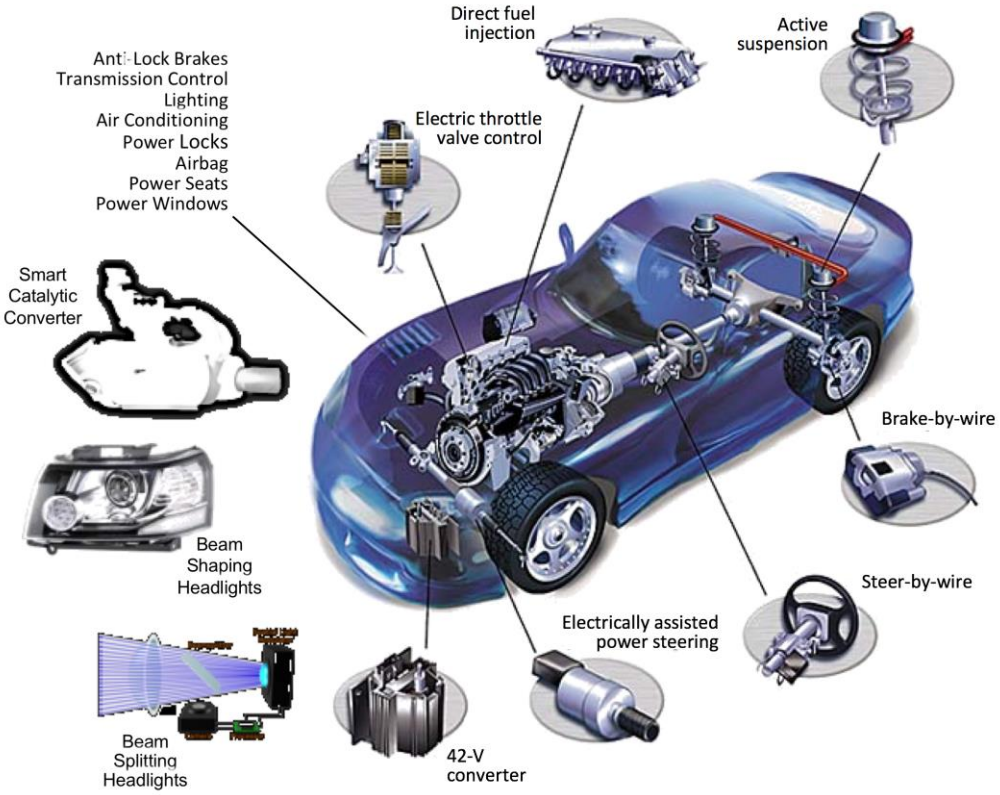

Figure 4: Critical Functions Have Migrated into Connected SW/HW

[7] Renesas In-Vehicle Networking Solutions https://www.renesas.com/us/en/solutions/_automotive/technology/networking-solutions.html

[8] Siam Ahmed, Geotab Inc., "Get to Know Connected Vehicle Technology: V2V, V2X, V2I," February 9, 2018, https://www.geotab.com/blog/connected-vehicle-technology/

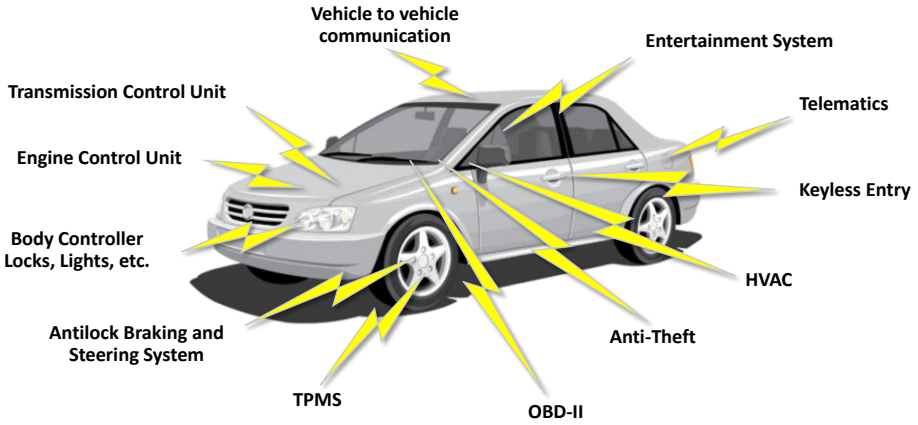

Figure 5: Multiple Attack Surfaces

In addition to having all these critical functions dependent on software and networking, as shown in Figure 6, the software that drives these abilities needs to be updated and sustained over the long term, both to fix flaws and to add functionality.[9]

When we look beyond current vehicle functions, as shown in Figure 7, we see connectivity with GPS; connectivity with other vehicles; connectivity with fuel management systems; connectivity with the highway and city infrastructure; and connectivity with traffic systems. All of these

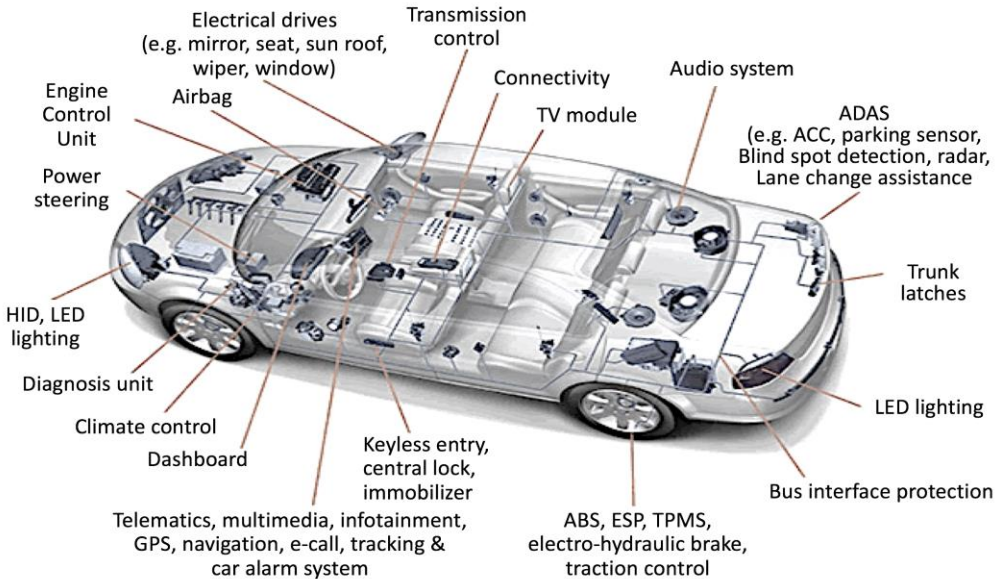

Figure 6: Exploding Need for Being Updatable and Sustainable

[9] Connector Supplier.com, "Automotive Connectivity Evolves to Meet Demands for Speed & Bandwidth," http://www.connectorsupplier.com/evolution-automotive-connectivity-autonomous-vehicle-technology-drives-need-speed-bandwidth/

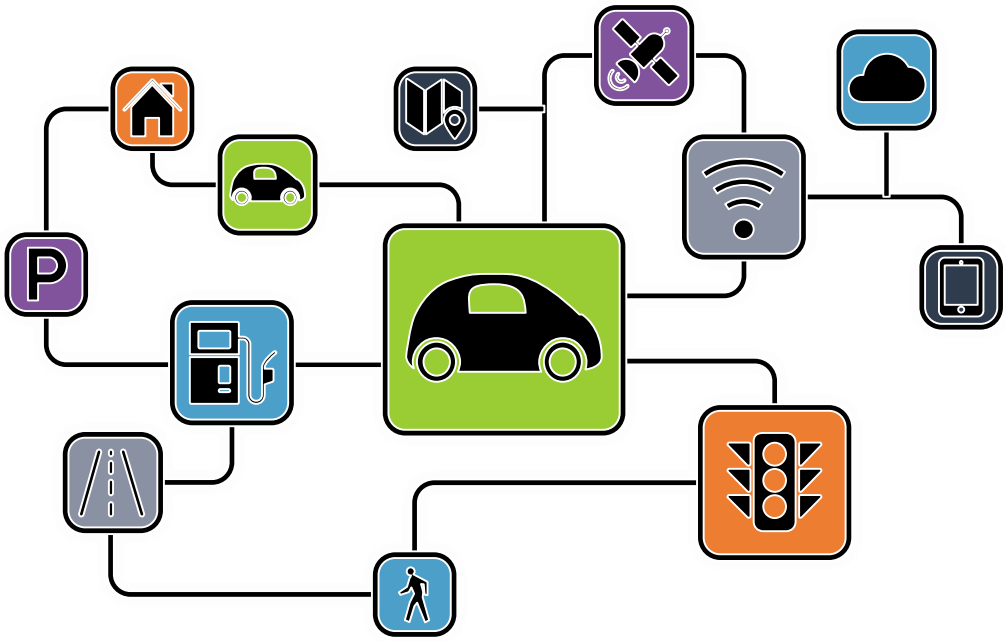

Figure 7: Connectivity and Complexity of Connected Software Systems is Still Expanding

new connections bring additional functionality, safety features, traffic management enhancements; but, they also bring risks and need to be done in a way that maintains the overall safety and security and reliability of the systems being connected.

As we connect this industrial internet of things, either directly or through workflows, the question of whether or not a specific system is secure and safe needs to focus on the interaction with that system. Did the owners of those other IIoT systems address issues about patching and vulnerabilities, configuring software correctly, and addressing weaknesses that could lead to unsafe or insecure operation of their IoT systems?

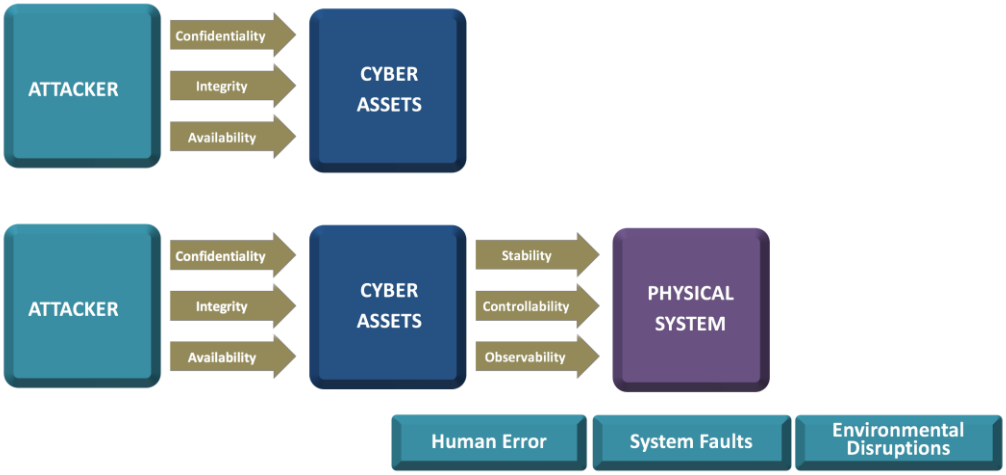

Figure 8: Risks and Impacts Expand with Physical Systems

The examples outlined so far in this paper focused on automobiles, but the trends and market forces that motivated and enabled those changes are equally applicable to other industries, such as commercial transportation in general, healthcare systems, critical infrastructure, retail systems, and building security and automation.

PERVASIVE SECM'S RISKS AND FAILURE IMPACTS

With the pervasive presence of software-enabled capabilities attackers can now focus on cyber physical assets via their cyber elements, as illustrated in Figure 8. Safety now involves risks associated with connectivity based on the innovation occurring around IoT. The traditional safety elements expand beyond fire, electric shock or physical harm, to the exfiltration of data, process implications that can affect systems (reliability and productivity), and the overall impact of newer technology to the intended use of products and systems.

As many have pointed out over the past few years [4, 10, 11], we must evolve from just an IT risk world view, where we're worried about the loss of information or loss of a service, to an operational risk view, where we consider loss of safety (the expanded concept of safety) and reliability or loss of life and property. Within the IIC work [4], as shown in Figure 9, we have put this together under the rubric of trustworthiness, where the safety, privacy, resilience, reliability and security behaviors of a system, while not always the same in proportion, are all interacting.

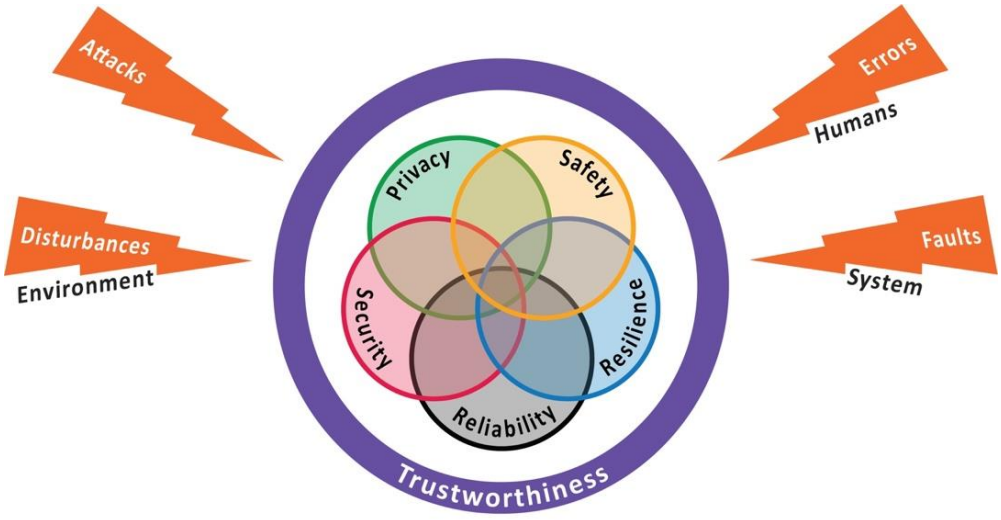

Figure 9: Risks and Impacts Expand with Physical Systems

[10] SANS Institute, "Security in a Converging IT/OT World," November 2016, https://www.sans.org/reading-room/whitepapers/analyst/security-converging-it-ot-world-37382

[11] Gartner, "IT and Operational Technology: Convergence, Alignment and Integration," March 2011, http://www.gartner.com/resId=1548729

THE STRUCTURED ASSURANCE CASE

Deriving a methodology for trustworthiness across a marketplace, we believe, requires building assurance cases. One of the key ideas with assurance cases is to develop and gather all the evidence that is going to be used to convince the stakeholders that the system properties, the system claims and requirements are being fulfilled with risks that are acceptable or known.

There are two main prerequisites in developing assurance cases:

a) An explicit statement(s) of the assumptions for the assurance
b) Claim of system trustworthiness and its sub claims

Figure 10 shows a more realistic assurance case illustration, which is an assurance case of assurance cases. In this illustration there are 28 different assurance cases shown. Each of them can be constructed independently. If your assumptions are complete and you can argue that the assumptions of each case are being fulfilled by the encompassing system and its assurance case, then you can compose the safety, reliability, security, and functional requirements of your subsystems and their assurance.

You can find standards defining the process and activities of creating an assurance case[1], exchanging assurance cases[2] and using assurance cases to hold the current composite state of the systems key behaviors[4]. Within the IIC Industrial Internet Reference Architecture [12] and Industrial Internet Security Framework[4], we discuss

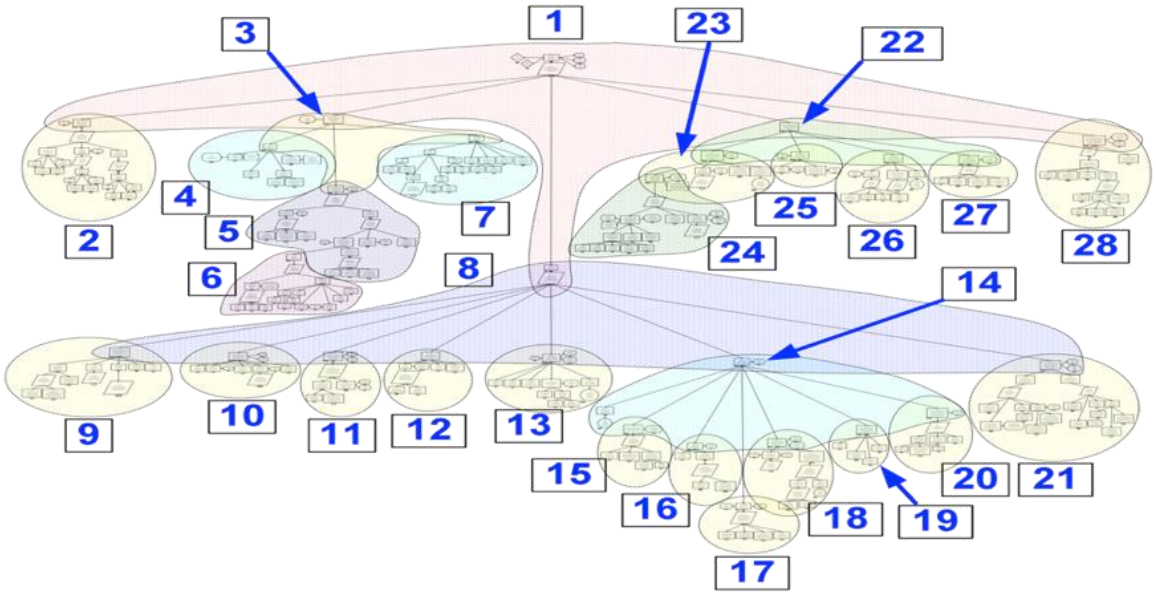

Figure 10: A Composition of Assurance Cases

[12] Industrial Internet Consortium, "Industrial Internet of Things Volume G1: Reference Architecture," IIC:PUB:G1:V1.80:PB:20170131, (2017), https://www.iiconsortium.org/IIRA.htm.

utilizing assurance cases as has NASA[13, 14, 15], the FDA[16], NIST[17], and projects going on in the EU[18, 19].

The key idea is that assurance cases can gather all the required information (including evidence of meeting system trustworthiness claims) about the systems characteristics and organize it for assessment across the life-cycle of the item and now that there is a standard for exchanging assurance cases [3], we as a marketplace can compose assurance cases leveraging others' work.

SUPPLY CHAIN AND SOFTWARE DEVELOPMENT ARTIFACTS

The other part of the life cycle of a market is the supply chain where, especially in software elements, there may be no visibility into the source of the software and its components and how they were created. Without that information you may incorporate software from sources that, you as the recipient, do not trust. One concept that should be part of your assurance is a software bill of materials with the similar intent and requirements as a hardware bill of materials. When an organization creates a hardware bill of materials (BOM), it is from trusted sources that have been validated through standard practices for the components listed in the BOM for longevity, performance and environment sustenance for the intended use. A software bill of materials (SBOM) should carry the same level of weight. For the trustworthiness of a system, its components, software, firmware, etc. should be validated for the source, responsibility of the providing party and vulnerability potential.

The design of software is ongoing from concept, to deployment and maintenance. In software design projects there are actually many artifacts (i.e., CONOPS, design documents, control flow, etc.) that are created early in the life cycle that can be examined to see if you are on track to meeting your goals about security, safety, resilience, reliability and privacy.

[13] National Aeronautics and Space Administration (NASA), "NASA System Safety Handbook, Volume 1, System Safety Framework and Concepts for Implementation," NASA/SP-2010-580, Version 1.0 November 2011, https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/ 20120003291.pdf

[14] National Aeronautics and Space Administration (NASA), "Understanding What It Means for Assurance Cases to "Work"," NASA/CR–2017-219582, https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20170003806.pdf

[15] National Aeronautics and Space Administration (NASA), "Dynamic Safety Cases for Through-life Safety Assurance – NASA," https://ti.arc.nasa.gov/publications/21593/download/

[16] Food and Drug Administration (FDA), "Infusion Pump Improvement Initiative," https://www.fda.gov/medicaldevices/productsandmedicalprocedures/generalhospitaldevicesandsupplies/infusionpumps/ucm202501.htm

[17] National Institute of Standards and Technology (NIST), "NIST SP 800-160 Vol. 1, Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems," 21 March 2018, https://nvlpubs.nist.gov/nistpubs/ SpecialPublications/NIST.SP.800-160v1.pdf

[18] CITADEL, Critical Infrastructure Protection Using Adaptive MILS, http://www.citadel-project.org/

[19] Dependability Engineering Innovation for Cyber Physical Systems (CPS), http://www.deis-project.eu/

The assurance case for software and software-enabled systems can include details about the SBOM, why each meets its respective trustworthiness needs, and what the evidence is to support that claim. Additionally, as the software elements are updated and revised, the assurance case for the system can be updated to reflect the current state of the assurance of its trustworthiness.

GATHERING AND SHARING EVIDENCE BASED ON NORMS & STANDARDS

Another key aspect about the software and software-enabled components of a system is the need for a focus on the software's intended use in supporting its objective and the need to actively try to determine whether the software can be influenced by hazards and attacks and threats in a way that impacts that purpose which the software is supporting or delivering. There are known attack patterns and hazard structures that can be executed by attackers or happen in the physical world and are applicable to the software. Mitigation strategies for those attack patterns can be code reviews, design reviews, dynamic testing, fuzzing communications and interfaces, attack surface analysis or pen testing. This can provide confidence that either the vulnerabilities are not there or that their impact to the operations has been mitigated. This process is illustrated in Figure 11 below.

One of the ways that industry uses to articulate potential software vulnerabilities, so that they can understand when others are talking about the same thing, is through the use of the Common Vulnerabilities and

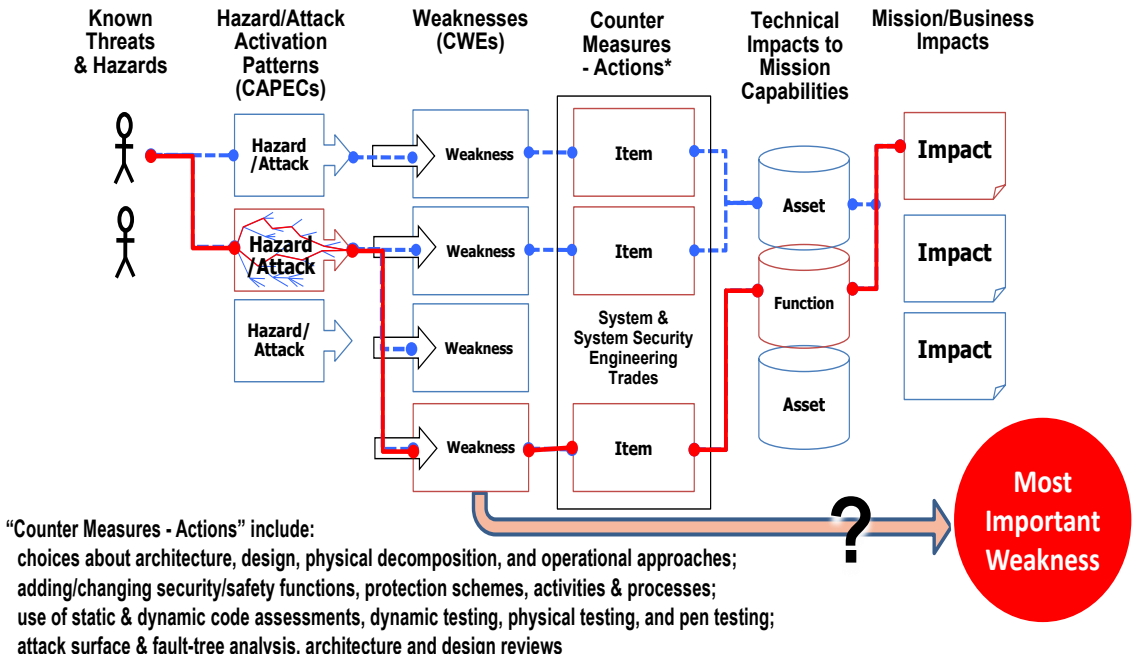

Figure 11: Hazards & Attacks that can impact Mission/Business Functions

Exposures (CVE) initiative[20], which started in 1999 and is now used throughout the industry of software-based systems. There is also the Common Vulnerability Scoring System (CVSS) [21], which is a risk scoring mechanism for prioritizing those vulnerabilities. Another mechanism used in the software community to discuss the types of vulnerabilities is the Common Weakness Enumeration (CWE) [22], which is the actual weaknesses that manifest as vulnerabilities. There is also a scoring system for prioritizing and focusing on the weaknesses that matter, called the Common Weakness Scoring System (CWSS) [23], and a standard way of referring to the attack patterns, Common Attack Pattern Enumeration and Classification (CAPEC) [24]. They are all described in international standards, part of the International Telecommunication Union's Technical Standard Series, X.1500 [25, 26, 27, 28, 29].

[20] MITRE Corporation, "Common Vulnerabilities and Exposures (CVE®)," https://cve.mitre.org/

[21] FIRST, "Common Vulnerability Scoring System (CVSS)," https://www.first.org/cvss/

[22] MITRE Corporation, "Common Weakness Enumeration (CWE™)," https://cwe.mitre.org/

[23] MITRE Corporation, "Common Weakness Scoring System (CWSS™)," https://cwe.mitre.org/cwss/

[24] MITRE Corporation, "Common Attack Pattern Enumeration and Characterization (CAPEC™)," https://capec.mitre.org/

[25] International Telecommunications Union Standardization Sector (ITU-T), "X.1520: Common vulnerabilities and exposures," 2011 & 2014, https://www.itu.int/rec/T-REC-X.1520

[26] International Telecommunications Union Standardization Sector (ITU-T), "X.1521: Common vulnerability scoring system," 2011 & 2014, https://www.itu.int/rec/T-REC-X.1521

[27] International Telecommunications Union Standardization Sector (ITU-T), "X.1524: Common weakness enumeration," 2012, https://www.itu.int/rec/T-REC-X.1524

[28] International Telecommunications Union Standardization Sector (ITU-T), "X.1544: Common attack pattern enumeration and classification," 2013, https://www.itu.int/rec/T-REC-X.1544

[29] International Telecommunications Union Standardization Sector (ITU-T), "X.1525: Common weakness scoring system," 2015, https://www.itu.int/rec/T-REC-X.1525

USING APPROPRIATE TESTING AND ASSESSMENT METHODS

Evaluating and assessing software is all-encompassing. Reasonable real-world solutions require using multiple techniques that are suited for specific scenarios and getting wide coverage instead of a one-size-fits-all model. The diagram shown in Figure 12 shows a large number of test cases of weaknesses for C and Java[30], where several tools were run on the test cases to see which of the tools could find the weaknesses.

As you can see in figure 12, there is a lot of white space shown in these two plots of tool coverage of the test cases for C and Java, which means that the tools did not find the things that were in those test cases. Identifying the right testing capability for the problem is ideal. The work that the Institute for Defense Analysis did for the Department of Defense [31] in their State-Of-the-Art-Report, looked at testing methods beyond just tools and the finding was the same, a lot of white space.

As shown in Figure 13, the appropriate tool or detection technique is matched with the artifact so that the weaknesses you care

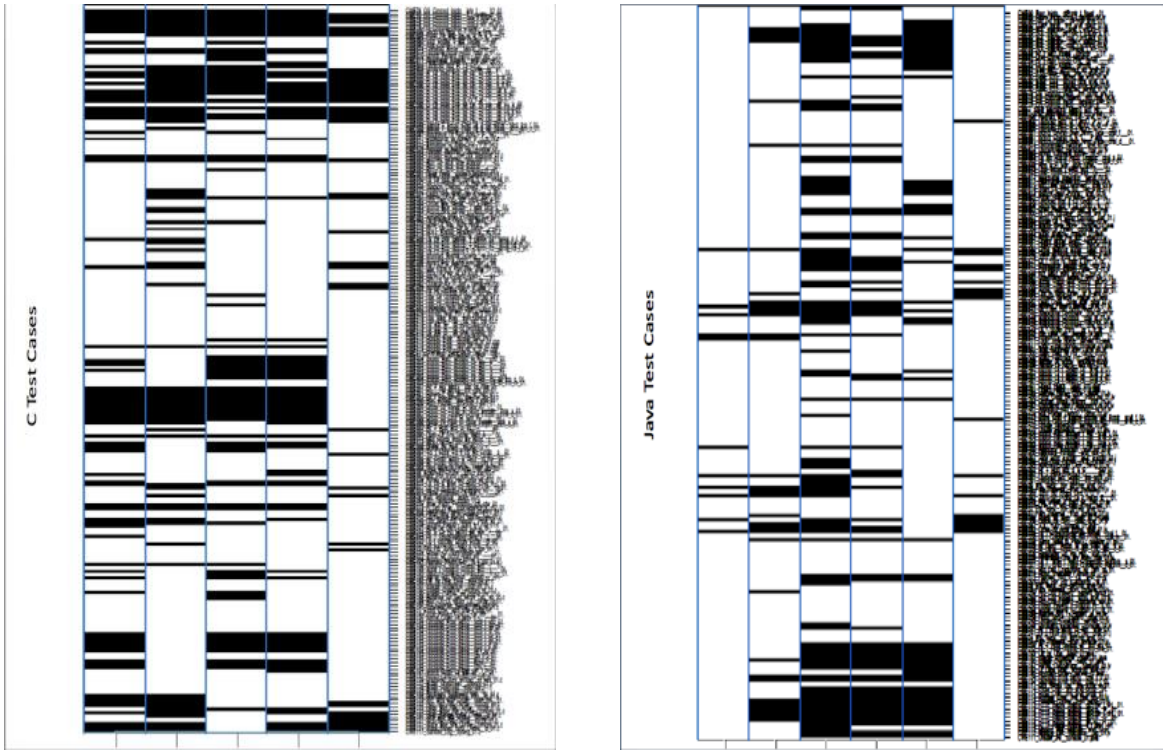

Figure 12: Coverage of Software Weakness Assessment Tools

[30] National Institute of Standards and Technology (NIST), "Software Assurance Reference Dataset (SRD)," https://samate.nist.gov/SRD/

[31] Institute for Defense Analyses, "State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation," 2016, https://www.acq.osd.mil/se/docs/P-8005-SOAR-2016.pdf

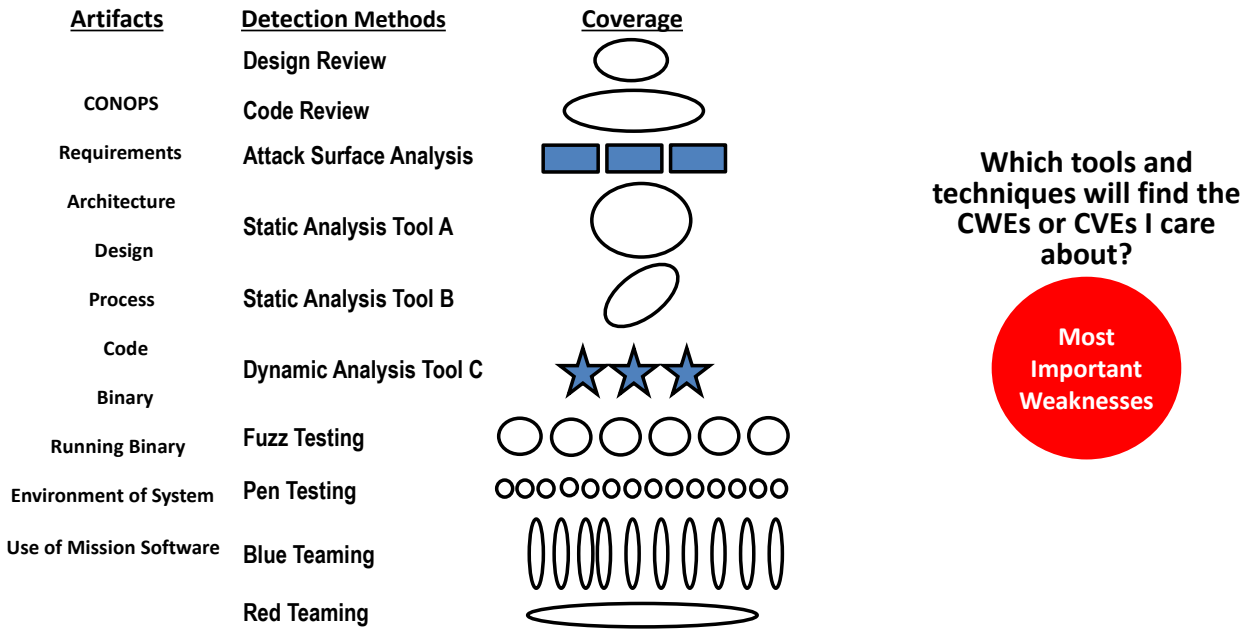

Figure 13: Matching Coverage to Weaknesses of Interest

about are addressed/detected. Identifying multiple techniques instead of one for a multitude of weaknesses will provide greater coverage.

When Assurance Activities Are Not Aligned

In an organization, different parties that are responsible for security, safety and reliability should be aligned along similar principles. For example, using CWE and CAPEC can be thought of as a list of those attacks that should be considered and the different kinds of weaknesses that should be looked for to help make sure the analysis is as complete as possible, whether it is an analysis of the software's architecture, design, code, or the way the code is deployed.

As an example, if we revisit our focus on automotive and think about the attack surface for cars, there is a constrained number of approaches for attacking a car[32]: You can attack through the services by trying to interrupt them and perform a man-in-the-middle attack; you can exploit software vulnerabilities in the actual commercial and open source software being used; you can try to retrieve data through sniffing attacks; you can go after the mobile devices or compromise physical components to infiltrate software; and you can try to attack the updates and download malware or malicious software.

For an assurance effort, we need to figure out how to address these threats and remove as much of the attack surface as

[32] Kaspersky Lab, "Connected cars: Secure by design," June 2017, https://www.kaspersky.com/blog/connected-cars-secure-by-design/16947/

possible but make sure the solutions fit together. The Miller-Valasek "Jeep Hack" attacks from back in 2015 [33] and 2016 [34] demonstrates how alignment is necessary.

Figure 14 is an illustration of the bus structure similar to the one in the Jeep. On the far right there is a square labelled "RAD." That is the radio/entertainment system also

the car. Unfortunately, that approach was eventually figured out by Miller and Valesek and they also figured out how to apply a software update of their own creation to the bus gateway (BCM) through the head-unit. This bus gateway was supposed to arbitrate the connection between the CAN bus and the bus with the head-unit, but after the

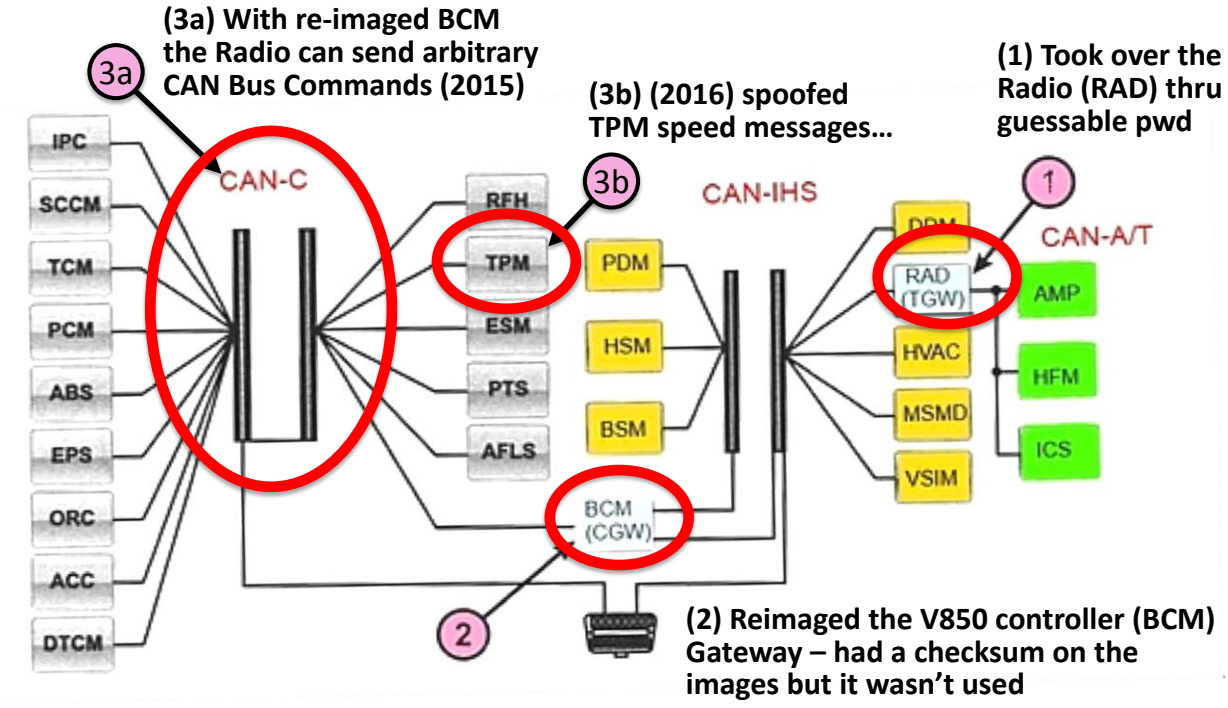

Figure 14: Hacking a Vehicle

referred to as the head-unit, an externally facing device that talks to the world. What Miller and Valasek found was that the head-unit used a guessable password.

This makes sense for convenience for the dealer, service people or manufacturer, who might need those passwords to do service on

update, Miller and Valesek had access to all of the devices on the internal CAN bus – those that control the operation of the car.

Unfortunately, while the update applied to the gateway through the head-unit was supposed to require a signed checksum, in practice it did not and was accepted as a

[33] Dr. Charlie Miller & Chris Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," August 2015, http://illmatics.com/Remote%20Car%20Hacking.pdf

[34] Dr. Charlie Miller & Chris Valasek, "Advanced CAN Injection Techniques for Vehicle Networks," 2016, https://www.youtube.com/watch?v=4wgEmNlu20c

legitimate update without that signed checksum. Once they had access to the CAN bus from the head-unit they could issue all the commands they wanted to the others on the bus, including opening windows, closing the window, changing speeds, turning the wheels, turning on the blinkers and windshield wipers and so on.

Now the actual attacks in 2015 did not work at highway speed, because it was based on a hack to the diagnostic system that does not allow changes to be made above 5 MPH, but it turned out that the tire pressure monitoring system was the source of the information about what speed the vehicle was going. So, in 2016, the Jeep Hack was evolved to spoof the tire pressure monitor messages to tell the car that it was going slow when it really wasn't. This was possible because the protocol for the bus discarded duplicate messages. Once they figured out how to get illegitimate message and message numbers onto the bus before the actual tire pressure monitoring systems messages through a spoof attack, they could go at highway speeds. Now when the tire pressure management system put its message out, it was discarded as duplicative and the car paid attention to the spoofed messages that it was going slow when in fact it was not.

A more structured review of the possible weaknesses in the design, architecture, code and deployed configurations of the software-enabled capabilities in the Jeep ecosystem, guided by a broad understanding of how software can be attacked through its inherent weakness could have identified the mistakes leveraged by the Jeep hacks. Expecting engineers and developers to think of the myriad of ways software can be influenced and attacked to do things that were not intended by its creators without standard methods of identifying what to look for and proving a rigorous method for collecting and analyzing what is done is one of the most common mistakes enterprises make. Through knowledge-bases such as CVE, CWE, and CAPEC, an organization can leverage the expertise of the world's software security and assurance experts and apply it into their specific type of software-enabled capabilities. All of the different testing techniques introduced and discussed in this section can be brought together as part of the evidence supporting an assurance case.

The DEIS (dependability engineering, innovation for cyber physical systems) project [35] is one example of an effort applying assurance cases. DEIS is exploring the idea of a digital dependability identity, which basically has all the information about the dependability characteristics of the cyber physical system. We offer that you can swap the word trustworthiness for dependability. Thus, the idea they are investigating is to have the vehicle itself, from its creation on, carry in digital form, the assurance case for why it is trustworthy and under what conditions it is trustworthy. Then, as it goes out into the world it can offer up to others in its ecosystem an explicit machine process-able document that

[35] Dependability Engineering Innovation for Cyber Physical Systems (CPS), http://www.deis-project.eu/

explains under what operating conditions it is trustworthy. This provides it a continuously assessable live assurance case about its operation that it is continuously reevaluated to determine whether it is still in a trustworthy state.

TRUSTWORTHINESS MODEL

Models of trustworthiness are element to bringing repeatable and scalable approaches that can be used across a marketplace or sector. Trustworthiness Models are a challenge in an interconnected world and revolve around describing the acceptable risks for a system and its context and these in turn drive the definition of trustworthiness for that system and its context. A trustworthiness model needs to define the required confidence level for the assurance level of the overall system, its individual components and connectivity. Using a trustworthiness model to define physical equipment trustworthiness has historical metrics. Currently, to validate and use a trustworthiness technique for physical equipment revolves around wear and tear for the environment, with a lifetime of usage cycles. For an overall system it can be based on the composition of the trustworthiness of the individual components. Using the automobile example, you can measure the individual components, such as a tire pressure sensor operating in cold and hot environments, water and humidity. The metrics around the cold and hot environments, water and humidity can be defined by the intended use of the physical equipment. Therefore, a trustworthiness model would be typically applied to general environments during multiple seasons where the average user of automobiles would drive. This is harder for software. Software's metrics would revolve around the requirements, design and development of the software – ultimately ending up with the software coding and deployment and then its use in its intended configuration and for its intended use. With hardware, you can look at the individual components, and expand the encompassing component structure. The tire pressure sensor can be looked at as one component, then the sensor and its housing as another component, then the sensor, housing and wiring. With software, to do this, you have to define all the components (SBOM). By utilizing the assumptions portion of an assurance case and being rigorous about capturing the things that need to be true/available in order for the rest of the assurance case to be true we can decouple components from the system of which they are components. As long as the encompassing system can make sure the assumptions are met, we can take trusted components and put them together into a system whose trustworthiness is assured. The core item is to capture, as assumptions, the things that will make the software reliability, resilience, safety, security and privacy possible.

Trustworthiness Criteria

A traditional model around trustworthiness in the safety world focuses on:

a) Reliability of the components and the system.
b) Availability of the components and the system

c) Safety of the components and the system

Newer trustworthiness criteria in the IIoT world for software and the security of systems also includes:

d) The integrity and authenticity of the components and system
e) The confidentiality of the data used by the components and system
f) The reputability of the data from the components and system
g) The privacy of the data used by the components and system
h) The maintainability of the components and system
i) The ability of the components and system for easy and modifiable configuration
j) The resilience of the components and system to an attack or misuse
k) The usability of the components and system for its intended use

Trustworthiness Assurance

Finding ways to measure the trustworthiness criteria can provide techniques to measure and qualify the trustworthiness of a system. Finding ways to measure (a) to (k) is the new challenge for defining trustworthiness in the IIoT world. The first steps are to define the trustworthiness scale needed for the intended application. Is it a mission critical system, a safety system or a general business application? Defining the expectations for (a) to (k) is the first step. If the first metric of integrity and authenticity is investigated, then looking at the ability for users that are authorized to use the system, the mechanisms used to secure data in transit and rest, software updates and validating the source, are the next major components.

Finally, if we had a market ecosystem where hardware, services and software components – whether at the part level or system level – all had assurance cases for themselves and that set of claims and assumptions was available to the market, we could understand what the value of the item was not only from its functionality and price, but also from the level of assurance and integrity it offered to those leveraging it and how well it could be leveraged and composed into a system that met their trustworthiness goals.

CONCLUSIONS

In an ever-increasing connected world with exponential software components, ensuring that systems designed today can provide trustworthiness in security, safety, privacy, resilience and reliability is necessary. Understanding how to define what trustworthiness is by focusing on a definition of a system's trustworthiness and assumptions made can be used to develop assurance cases. These assurance cases allow the ability to specify and then measure the trustworthiness. Using internationally recognized techniques to build on assessment capabilities for trustworthiness, such as CVE, CWE and CAPEC, provide a robust global standard that would allow an easy method to communicate the trustworthiness measure. Recognizing also that identifying the trustworthiness criteria that is applicable to the system, in the environment that it is operating, provides

clarity for the assessment. This can then lead to leveraging of standardized Structured Assurance Cases to enable a marketplace of trustworthy internet of things systems, components and related capabilities.

ACKNOWLEDGEMENTS

We would like to thank all of those who provided the work that we have leveraged to put together the thoughts in this paper. Without their work over the past decade we would not have been able to address the topics and their management as concisely and completely.

➢ Return to IIC Journal of Innovation landing page for more articles and past editions.