




TRUST  SOFT  
ISO/SAE 21434 FROM A  
SOFTWARE DEVELOPMENT  
PERSPECTIVE

---

How sound, exhaustive static analysis can help ensure air-tight automotive cybersecurity while lowering its costs

---

# Table of contents

Executive Summary .....	3
Automotive innovation drives need for greater cybersecurity ....	4
The automotive software security challenge.....	5
ISO/SAE 21434: Road vehicles – Cybersecurity engineering .....	6
Examples of dangerous UBs in automotive SW/FW .....	10
How TrustInSoft Analyzer supports the fulfillment of ISO/SAE 21434 requirements.....	12
How TrustInSoft Analyzer enhances the verification techniques listed in ISO/SAE 21434 .....	12
How TrustInSoft Analyzer exceeds the current state of the art in software analysis tools .....	15
Case studies.....	17
Conclusions .....	18
References.....	19
About TrustInSoft .....	21

## ISO/SAE 21434 FROM A SOFTWARE DEVELOPMENT PERSPECTIVE

# Executive Summary

Over recent decades, several technological trends have converged to reshape the automotive industry. These software-driven trends—which rely on connectivity as well—have transformed motor vehicles from machines that were mostly mechanical into highly complex, cyber-problems physical systems.

Unfortunately, dependence upon software and connectivity has made vehicles inviting targets for malicious software hackers. The resulting growth in attack surfaces has already resulted in some high-profile cyberattacks on motor vehicles and is driving the auto industry to vigorously address this threat. Cybersecurity is now a major concern in the sector.

Automotive electronics systems are especially vulnerable to cyberattacks. For example, the predominant programming language in automotive applications, C/C++, is highly susceptible to coding errors that create undefined behaviors—the types of vulnerabilities hackers most often exploit to penetrate embedded systems like automotive components.

To remedy this situation, the Society of Automotive Engineers (SAE) has collaborated with ISO to develop ISO/SAE 21434: Road Vehicles – Cybersecurity Engineering, an ISO-approved standard aimed at addressing automotive cybersecurity in a systematic way.

ISO/SAE 21434 specifies a cybersecurity risk management process for automotive systems. As a process specification, it provides an excellent framework for achieving cybersecurity in motor vehicles. It does not, however, specify in detail how to prove the absence of vulnerabilities or demonstrate compliance with the standard. Those specifics are largely left up to the manufacturer.

This is especially true when it comes to integration and verification for cybersecurity in general, and how to adequately detect and eliminate the vulnerabilities hackers tend to exploit.

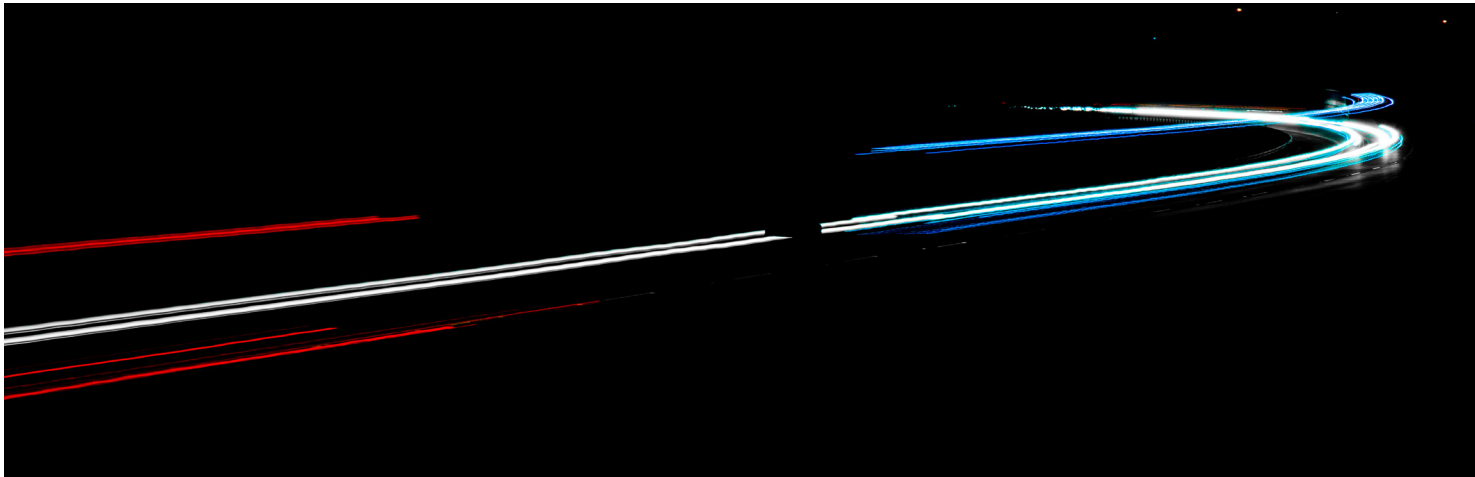
### In this white paper, we'll examine:

- The growing trends and challenges the automotive industry faces today
- ISO/SAE 21434 (what it is, what it addresses, what it does not address)
- The rapid growth of attack surfaces in automotive software and firmware
- The extent to which cybersecurity vulnerabilities impact vehicle safety
- Why the software vulnerabilities known as undefined behaviors (UB) are so dangerous to automotive safety

Finally, we'll examine how certain tools that leverage mathematical formal methods can complement and optimize the verification techniques cited in ISO/SAE 21434. We'll see how these tools can guarantee the detection and elimination of undefined behaviors earlier in the verification cycle, reduce the costs and risks associated with ISO/SAE 21434 compliance, and contribute to air-tight cybersecurity in motor vehicles.

ISO/SAE 21434 FROM A SOFTWARE DEVELOPMENT PERSPECTIVE

# Automotive innovation drives need for greater cybersecurity



The automotive industry is undergoing enormous change. Several megatrends have developed in parallel over the past several years.<sup>1</sup>

One of these trends is **digitization**—the capture of information within the automotive system and its conversion to digital messages. The digital customer connection now affects all aspects of the value chain, starting with product strategy development through customer interaction through sales and after-sales. An example of this trend is the automated notification of customers directing them to a qualified service center when their vehicle self-detects a potential problem.

**Powertrain electrification** is a second trend that is very challenging for the industry for several reasons. The supply chain for electric vehicles (EV) is fundamentally different from that for internal combustion engine (ICE) vehicles. The EV transmission system is decentralized and simplified, using 30% fewer parts than that of an ICE vehicle. A major portion of the cost of an EV is defined by its battery pack, a component supplied by players who, for the most part, are new to the industry. Perhaps most important of all, the competitive differentiators in this market—including electronics, cooling, and smart power distribution management—are substantially defined by software.

**Advanced driver-assistance systems (ADAS)** rely on automated technology such as sensors and cameras to detect nearby obstacles or driver errors and generate appropriate responses. They require extensive, highly reliable command and control communication between components to carry out their safety-critical tasks.

Just about every automotive OEM has prioritized **autonomous driving (AD)** as a strategic growth area. To succeed, however, they will have to overcome some stiff challenges. They will have to guarantee safety and reliability under all circumstances. This could be a tall order in situations where human intervention may not be immediately available. In any circumstance, the human/machine handover is critical to safety, and cybersecurity must be assured to prevent lockout of the human driver and hijacking of the vehicle.

**Car sharing** and other new mobility concepts offer mobility without the burdens of car ownership. These concepts depend on the reliability and integrity of various features, including:

- Self-service pick-up and drop-off, including automated valet parking (AVP)
- Client billing
- Vehicle tracking for maintenance
- Theft prevention

All of these features are heavily dependent on software and connectivity and require the highest level of cybersecurity.

The common thread among all these trends is vehicle **connectivity**. Sufficient mobile connectivity bandwidth is now available to support a variety of complex use cases. Vehicles can be connected to back-end systems and to other vehicles. They exchange large volumes of data.

Unfortunately, there is also a downside to all this software and connectivity. Under heavy time-to-market pressure, automotive OEMs are depending on many new players to bring software expertise to the sector. They're also relying on numerous open-source technologies to facilitate interoperability between systems and suppliers.

These factors have made their latest vehicles ideal targets for hackers looking to exploit cybersecurity vulnerabilities.

MITRE defines a cybersecurity "vulnerability" as

"A flaw in a software, firmware, hardware, or service component resulting from a weakness that can be exploited, causing a negative impact to the confidentiality, integrity, or availability of an impacted component or components."<sup>2</sup>

## The automotive software security challenge

Software security is now a key challenge in the automotive industry. The potential consequences of a cyberattack upon a vehicle have become significantly more severe. These consequences include:

- Sabotage of vehicle
- Injury and loss of life
- Financial loss through either theft or liability
- Widespread vehicle recalls
- Catastrophic impact on the company's image

To meet this new challenge, software developers in the sector must adopt a robust cybersecurity management process. In addition, they must adapt their software verification activities to better identify and eliminate the vulnerabilities that software hackers typically exploit.

To address the first of these needs, the Society of Automotive Engineers (SAE) and the International Organization for Standardization (ISO) have established a standard cybersecurity management process for the industry. It's called: *ISO/SAE 21434: Road vehicles — Cybersecurity engineering*.



## ISO/SAE 21434 FROM A SOFTWARE DEVELOPMENT PERSPECTIVE

# ISO/SAE 21434: Road vehicles – Cybersecurity engineering

### What it is

ISO/SAE 21434 is a set of guidelines for cybersecurity in motor vehicles. Created in response to the growing number of cyberattacks on cars and trucks, it was developed jointly by ISO and SAE to establish cybersecurity requirements for the development of electrical and electronic components in road vehicles.

ISO/SAE 21434 provides a structured approach to define and manage cybersecurity goals and risks. It establishes a rigorous framework to enable organizations to design vehicles that are protected against cybersecurity threats. In addition, it is designed to ensure that cybersecurity is considered at every stage of the product's development, from inception through retirement.

Compliance with ISO/SAE 21434, while mandatory for all ISO members, is technically voluntary for the industry as a whole; ISO/SAE 21434 is a standard, not a regulation. However, automotive OEMs and suppliers must be compliant with UN regulation no. 155 (UN R155).<sup>3</sup>

### Relationship to UN R155

UN R155 requires automotive firms to implement a Cybersecurity Management System (CSMS) and meet other specific requirements related to cybersecurity. For the automotive industry, the requirements of UN R155 are binding and must be complied with in order to obtain type approval and market access. Failure to comply can lead to a sales ban in the corresponding area of application. More than 60 countries have already committed to adopting the regulation.

Regulations like UN R155 often refer to various standards as a thematic point of reference. UN R155 refers to ISO/SAE 21434. A published interpretation document of late 2020 directly relates the requirements of the regulation to the various requirements of the standard.<sup>4</sup>

ISO/SAE 21434 thus provides support for meeting the requirements of UN R155. Put another way, for

companies who must comply with UN R155, ISO/SAE 21434 has become a de facto requirement.

### Minimizing the cost of cybersecurity and compliance

To prove their compliance with ISO/SAE 21434, automotive suppliers must demonstrate their products' cybersecurity at initial delivery and are equally liable for that cybersecurity throughout the entire product lifecycle.

They must define cybersecurity activities that support the initial product delivery and the entire life cycle of the product. They must demonstrate good cybersecurity practices in accordance with the cyber risk analysis. They must identify, assess, and mitigate product vulnerabilities.

Furthermore, they are liable for any damages in the event of a product vulnerability being exploited by an attacker.

To minimize their overall costs, suppliers need to define efficient cybersecurity activities at the beginning of each project to minimize the cost of those activities over the product's lifecycle.



## A process-centric standard

From a software development perspective, ISO/SAE 21434 is a process-centric standard focusing on cybersecurity risk management. It leaves a fair amount of freedom to suppliers in the methods they use to demonstrate the security of their products. The standard enumerates testing and verification techniques, but provides no information or guidance on:

- How to employ those techniques individually
- Circumstances in which they are useful
- How to coordinate and combine them
- What is needed from tools to employ those techniques correctly

ISO/SAE 21434 puts the onus on suppliers to identify potential attack vectors and vulnerabilities and determine how to protect against them. They must do so through their own specific implementation of the mandatory risk assessment process the standard defines.

Again, in seeking to comply with ISO 21434, it is important that suppliers define efficient cybersecurity activities from the very beginning of the process in order to optimize efficiency and minimize costs throughout the product life cycle.

## Attack surfaces in automotive systems

In the context of ISO/SAE 21434, cybersecurity risk analysis involves assessing the attack surface of automotive systems. The term “attack surface” refers to the sum of all potential entry points—direct or indirect—or vulnerabilities within the system that could be exploited by malicious actors.

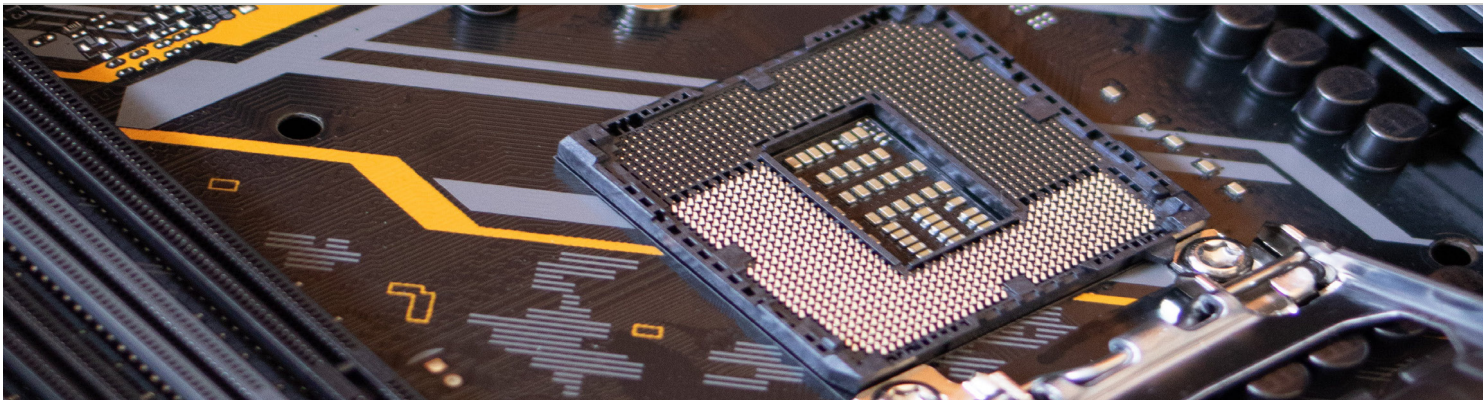
Direct entry points refer to those specific components, interfaces, or interactions within the system that can be directly accessed or targeted by an attacker. Indirect entry points are not directly accessible or exposed to external entities. They often involve exploiting dependencies or interactions between various components or subsystems.

Attack surface analysis includes analyses of hardware components, software modules, network interfaces, external connections, and communication channels.

The recent industry trends discussed earlier are causing a rapid expansion of automotive attack surfaces driven by the immense volumes of data these new technologies require. Examples of the massive new attack surfaces being created and expanded include:

- Internal communication networks (CAN, FlexRay, Automotive Ethernet, etc.)
  - Can be accessed via the OBD-II port <sup>5</sup>
  - Can be accessed remotely through any of the growing number of vehicle ECUs<sup>6</sup>
  - Permits access to all ECUs and most critical functions<sup>7</sup>
  - Permits hackers to realize a vast array of attack scenarios, such as: <sup>8</sup>
    - Spoofing of GPS signals to redirect the car
    - Attacks on the mobile communication between the vehicle and external backend data servers
    - Denial-of-Service (DoS) attacks
    - Hacking user accounts to get access to personal data
    - Attacks on the smart card used for filling the tank
- In EVs
  - Real-time data from GPS and mobile networks for battery charge management
  - Protocols for electricity billing at charging points

- For ADAS and AD
  - Data from sensors (radar, lidar, cameras, etc.) for navigation and warnings
  - Frequent, high-volume exchange of 3D map data
  - Safety-critical procedures for human/machine handoff
- In shared vehicles
  - Direct link between the car and backend servers through a mobile connection
  - Vehicle-to-infrastructure (V2I) communication for automated valet parking (AVP)
- In connected cars, in general
  - Real-time updating of moving maps, traffic, weather, and local service availability
  - Customer services provided by OEMs and third parties
  - Over-the-air (OTA) updates for an ever-increasing volume of applications



### Source code attack surfaces

While much can be done to secure a vehicle at the system level, a vast portion of the overall attack surface just described consists of source code. Software hackers will probe a vehicle's code for vulnerabilities they can exploit. Therefore, automotive ECUs and other components that rely on software and firmware must be secured at the source code level as well.

Much of that software and firmware is written in the C/C++ programming language.

Because C/C++ code can run high-level structured programming on low-level mechanisms, it allows programmers to directly manipulate the hardware on which it runs. This characteristic—along with its flexibility and its extensive support in terms of knowledge and programming resources—has made C/C++ the language of choice for automotive ECUs and for embedded systems in general.<sup>9</sup>

Most of the software and firmware for automotive applications—including ADAS, OTA updating, EV charging and billing protocols, vehicle connectivity, infotainment and many others—are written in C or C++. The services software for networking protocols like CAN and FlexRay are programmed in C. The Automotive Open System Architecture (AUTOSAR) adaptive platform is realized in C++. And all of those applications rely on real-time operating systems (RTOS) or sequencers that are written in C.

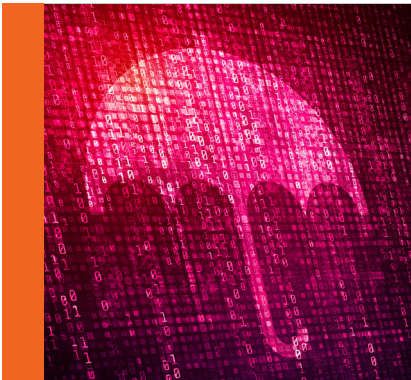
But while C/C++ offers many advantages to automotive software engineers, it also has some drawbacks. Unfortunately, the flexibility C/C++ offers makes it very easy for coders to make errors that hackers can leverage for their nefarious purposes. In fact, a study by IOActive found that automobiles “are plagued by many serious vulnerabilities that malicious actors can exploit to gain access to a vehicle's systems.”<sup>10</sup>



## Undefined behaviors

An undefined behavior (UB) is the result of executing a program whose behavior is prescribed to be unpredictable. UBs can cause programs to crash, produce incorrect results, or permit unauthorized access. Unauthorized access could allow a hacker to inject malicious input data or malware or alter the program's functionality.

Many UBs are what are known as memory-safety defects. The C and C++ programming languages are well known as being "memory-unsafe languages". This means that C and C++ do not prevent programmers from making coding mistakes that can introduce bugs related to how a program uses memory; the C/C++ compiler has no rules to prevent these conditions and thus allows the code to be compiled.



Memory-safety UBs include:

- Buffer overflow
- Integer overflow and underflow
- Use after free
- Null pointer dereferencing

Besides being very common, UBs are also very dangerous.

Undefined behaviors are very subtle and very difficult to detect. They tend to be identified by hackers (white hat or black hat) who probe software through an assortment of automated and highly sophisticated methods. They can't be discovered through rule checking or syntactic analysis. Systematic detection of UBs requires a specialized, purpose-built tool.

Six of Mitre's CWE Top 25 Most Dangerous Software Weaknesses in 2022 were undefined behaviors. CWE stands for Common Weakness Enumeration, Mitre's method for identifying and numbering security flaws in software and a widely accepted reference for vulnerabilities classification. The rankings were similar in 2021.

If we look at the occurrences of CWEs in embedded code and weight the share of each vulnerability based on its CWE "Score" (Mitre's method of accounting for the fact that some vulnerabilities are more exploitable than others), Buffer Overflows (CWE-787, CWE-125 and CWE-119) represent nearly 50% of embedded vulnerabilities, as illustrated in Figure 1.

Undefined Behaviors as a whole constitute nearly 69.6% of embedded vulnerabilities based on Score.

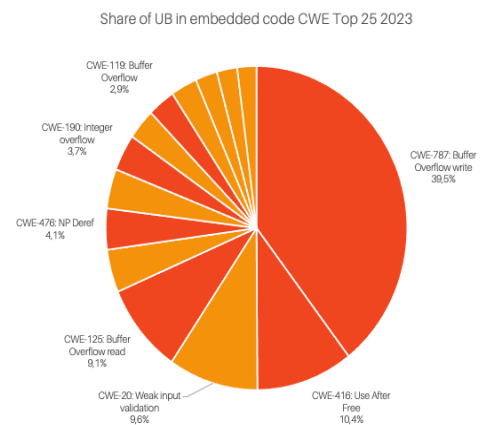
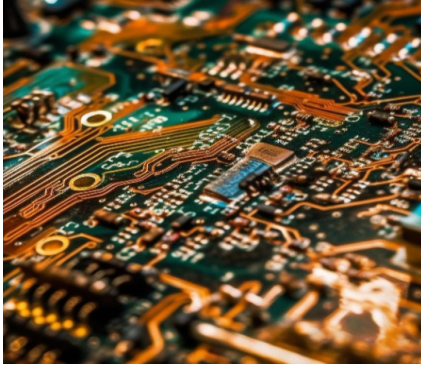


Figure 1: UB share of embedded code vulnerabilities (2023 CWE Top 25)

# Examples of dangerous UBs in automotive SW/FW



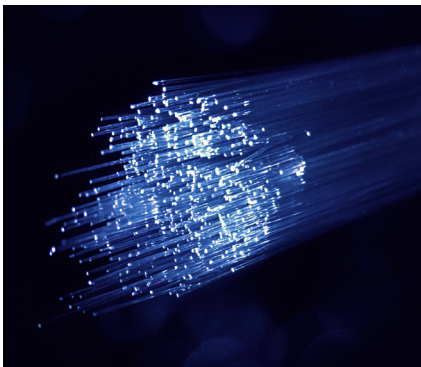
## Continental AG Infineon S-Gold 2 (PMB 8876) chipset

An Improper Restriction of Operations within the Bounds of a Memory Buffer issue (CWE-119, Buffer Overflow) was discovered in the Continental AG Infineon S-Gold 2 (PMB 8876) chipset in July 2017 (CVE-2017-9633).<sup>11</sup> Affected were all telematics control modules (TCUs) built by Continental AG containing that chipset, and various vehicles produced by BMW, Ford, Infinity and Nissan.

NIST said this “vulnerability in the temporary mobile subscriber identity (TMSI) may allow an attacker to access and control memory. This may allow remote code execution on the baseband radio processor of the TCU.” NIST gave this weakness a base vulnerability score 8.8/10 (high).<sup>12</sup>

In an Industrial Control Systems (ICS) Advisory, the Cybersecurity and Infrastructure Security Agency (CISA) said, “Successful exploitation of these vulnerabilities could allow a remote attacker to execute arbitrary code. This may allow an attacker to disable the infotainment system of the vehicle and affect functional features of the vehicle.” They warned manufacturers that this weakness requires a “low skill level to exploit” and that “public exploits are available.”<sup>13</sup>

The primary mitigation plan for this vulnerability was the deactivation of the component. Affected vehicles were no longer connected to telematics services,<sup>14</sup> depriving users of some functionality.



## Marvell 88W8688 Wi-Fi firmware

In November 2019, an Out-of-bounds Write (Buffer Overflow, CWE-787) vulnerability (CVE-2019-13582) was identified in the firmware of the Marvell 88W8688, a highly-integrated, low-cost, low-power, WLAN and Bluetooth Baseband/RF system-on-chip (SoC).<sup>15</sup>

This vulnerability allows attackers to bypass the authentication process and hack into the Tesla Model S/X in-vehicle multimedia system remotely through the Parrot Faurecia Automotive FC6050W Bluetooth module. A stack overflow could lead to denial of service or arbitrary code execution. NIST gave this flaw a base vulnerability score of 9.8/10 (critical). It affects Tesla Model S/X vehicles manufactured before March 2018.<sup>16</sup>

Researchers from Keen Security Lab demonstrated how this weakness could be exploited remotely through over-the-air (OTA) communication.<sup>17</sup>





### QNX Real Time Operating System (QNX RTOS)

In August 2021, BlackBerry publicly disclosed that its QNX Real Time Operating System (RTOS) is affected by a BadAlloc vulnerability (CVE-2021-22156).<sup>18</sup>

NIST characterized this flaw as “an integer overflow vulnerability [CWE-190] in the calloc() function of the C runtime library of affected versions of BlackBerry® QNX Software Development Platform (SDP)... that could allow an attacker to potentially perform a denial of service or execute arbitrary code.” NIST gave this weakness a base vulnerability score of 9.8/10 (critical).<sup>19</sup>

According to CISA, this vulnerability is remotely exploitable. “A remote attacker could exploit CVE-2021-22156 to cause a denial-of-service condition or execute arbitrary code on affected devices. A compromise could result in a malicious actor gaining control of highly sensitive systems.”<sup>20</sup>

Possible consequences of the exploitation of this vulnerability could include the loss of vehicle control due to compromise of safety-critical functions, and injury or death to vehicle occupants or other persons resulting from an accident.



### TrustInSoft Analyzer – the ultimate weapon against undefined behaviors

We mentioned earlier that the systematic detection of undefined behaviors requires a specialized, purpose-built tool. That tool is TrustInSoft Analyzer.

TrustInSoft Analyzer is a hybrid code analyzer that combines static and dynamic analysis techniques together with formal methods to produce existence proofs of properties that cannot be confirmed using static techniques only.<sup>21</sup> It is the formal methods that are key to these existence proofs.

Formal methods are ideal for validating code that needs to be perfect. They use mathematical techniques to “solve” the logic of computer programs or other systems (integrated circuits, for example) to answer questions about their behavior. For example, if you want to know if there is any way a buffer overflow could occur in your program, formal methods can be used to determine that.

**What’s more, a state-of-the-art formal methods tool can answer those questions automatically.**

TrustInSoft Analyzer is a sound formal methods tool designed specifically to detect undefined behaviors. An analyzer is considered “sound” with respect to a specific guideline if it cannot give a false-negative result—if it finds all violations of the guideline within the program.

In other words, when TrustInSoft Analyzer is used to exhaustively analyze a program for specific undefined behaviors, it will find all instances of those undefined behaviors. It will report every UB in your code. Once those instances have been eliminated, you have an absolute guarantee that those undefined behaviors cannot occur in the program.

Thanks to its soundness, TrustInSoft Analyzer provides you, your customer, and any regulatory authorities with proof that no undefined behaviors exist within your software.

What's more, TrustInSoft Analyzer accounts for the configuration of your target hardware. It provides target emulation for embedded hardware platforms that enables testing in an environment that closely resembles your target architecture. Target emulation helps you find vulnerabilities that unit testing in a host environment cannot possibly reveal. And everything that can change from one platform to another is configurable in the Analyzer.

Finally, TrustInSoft Analyzer fits neatly into any software development model. It is compliant with the V-model development life cycle and with Agile methods as well. Plus, it can be fully integrated with continuous integration frameworks.

### How TrustInSoft Analyzer supports the fulfillment of ISO/SAE 21434 requirements

Since TrustInSoft Analyzer was designed to detect and verify the elimination of undefined behaviors in software, it directly contributes to the fulfillment of the requirements of ISO/SAE 21434 section 10.4.2, Integration and Verification. It fully complies with the requirement "testing should be performed in order to confirm that unidentified weaknesses and vulnerabilities remaining in the component are minimized."

TrustInSoft Analyzer contributes to the ISO/SAE 21434 goal of cybersecurity risk management at the software level. It surpasses the baseline techniques cited by ISO/SAE 21434, going beyond expectations of the standard in three specific ways.

First, it focuses on in-depth analysis of undefined behaviors, which are very subtle and difficult to detect. Second, it finds undefined behaviors early in the development process and will not miss any of them (no false negatives). And third, it takes into account the characteristics of both the target hardware and the toolchain.

In short, it confirms that there are no undue risks in your code.

### How TrustInSoft Analyzer enhances the verification techniques listed in ISO/SAE 21434

ISO/SAE 21434 provides a flexible and adaptable framework for organizations to select and apply appropriate cybersecurity measures. It also suggests a number of software verification techniques organizations may choose to use to help ensure cybersecurity. Among the techniques it suggests are the application of CERT-C coding rules, fuzz testing, and penetration testing.

ISO 21434 does not mandate the use of any of these techniques but acknowledges their relevance and effectiveness in certain contexts. Organizations implementing ISO 21434 can choose whether to incorporate these techniques—and how to incorporate them—as part of their cybersecurity measures.

Because of its ability to find and eliminate hard-to-detect undefined behaviors, TrustInSoft Analyzer can be used to enhance all the techniques just mentioned and perform them more efficiently. We'll look, in turn, at how it strengthens each of those techniques.



## How TrustInSoft Analyzer helps achieve the goals of CERT C

CERT C was developed by the CERT Coordination Center (CERT/CC) of the Software Engineering Institute (SEI) because of the inherent security (memory safety) weaknesses of the C/C++ language.<sup>22</sup> It currently consists of a set of 122 rules and 180 recommendations. Cert C rules are meant to provide normative requirements that, when followed, should improve the safety, reliability, and security of software systems.

Cert C was developed with input from a wide range of software experts. The standard is widely recognized and used by government agencies, private industry, and academic institutions.

While these benefits are good, it is important to achieve and verify that the goal of Cert C compliance is also being achieved.

According to NIST, the goal of CERT C is “to develop safe, reliable, and secure systems, for example by eliminating undefined behaviors

that can lead to undefined program behaviors and exploitable vulnerabilities.”<sup>23</sup> This goal echoes the overarching rule 1.3 of a similar standard, MISRA C: “There shall be no occurrence of undefined or critical unspecified behavior.”

As a sound formal methods tool designed specifically to detect undefined behaviors, TrustInSoft Analyzer is ideally suited to helping software development organizations achieve the stated goals of CERT C and MISRA C.

An exhaustive analysis with TrustInSoft Analyzer will identify the location of every undefined behavior within your code. Because of its soundness, TrustInSoft Analyzer will not miss a single undefined behavior. This significantly reduces the number of residual vulnerabilities that must be addressed by other means.

### Following Cert C rules:

- Helps to improve the security of C/C++ code
- Helps to improve the skills of junior programmers
- Makes code easier to understand
- Makes finding and fixing bugs and other issues much easier

## How TrustInSoft Analyzer optimizes fuzz testing

Fuzz testing, also known as fuzzing, is recommended by ISO 21434 (in requirement RC-10-12) because it has been shown highly effective in uncovering cybersecurity vulnerabilities in software. In general practice, fuzzing is a software testing technique that rapidly applies large numbers of valid, nearly valid, or invalid inputs to a program, one after the other, in a search for undesired behaviors (vulnerabilities).

Fuzzing can be greatly enhanced through the use of TrustInSoft Analyzer. The tool can eliminate the need to repeat fuzz testing and reduce verification costs. The mathematical guarantees it provides simplify discussions with customers and regulators.

Fuzzing with TrustInSoft Analyzer will eliminate the vast majority of undefined behaviors in your code. The tool also provides target emulation for embedded hardware platforms, which allows you to fuzz your embedded code in an environment that closely resembles your target architecture.

It is important to remember, however, that fuzzing is not exhaustive. To guarantee the elimination of all undefined behaviors, it is necessary to apply a technique that only TrustInSoft Analyzer provides—a technique called exhaustive static analysis.

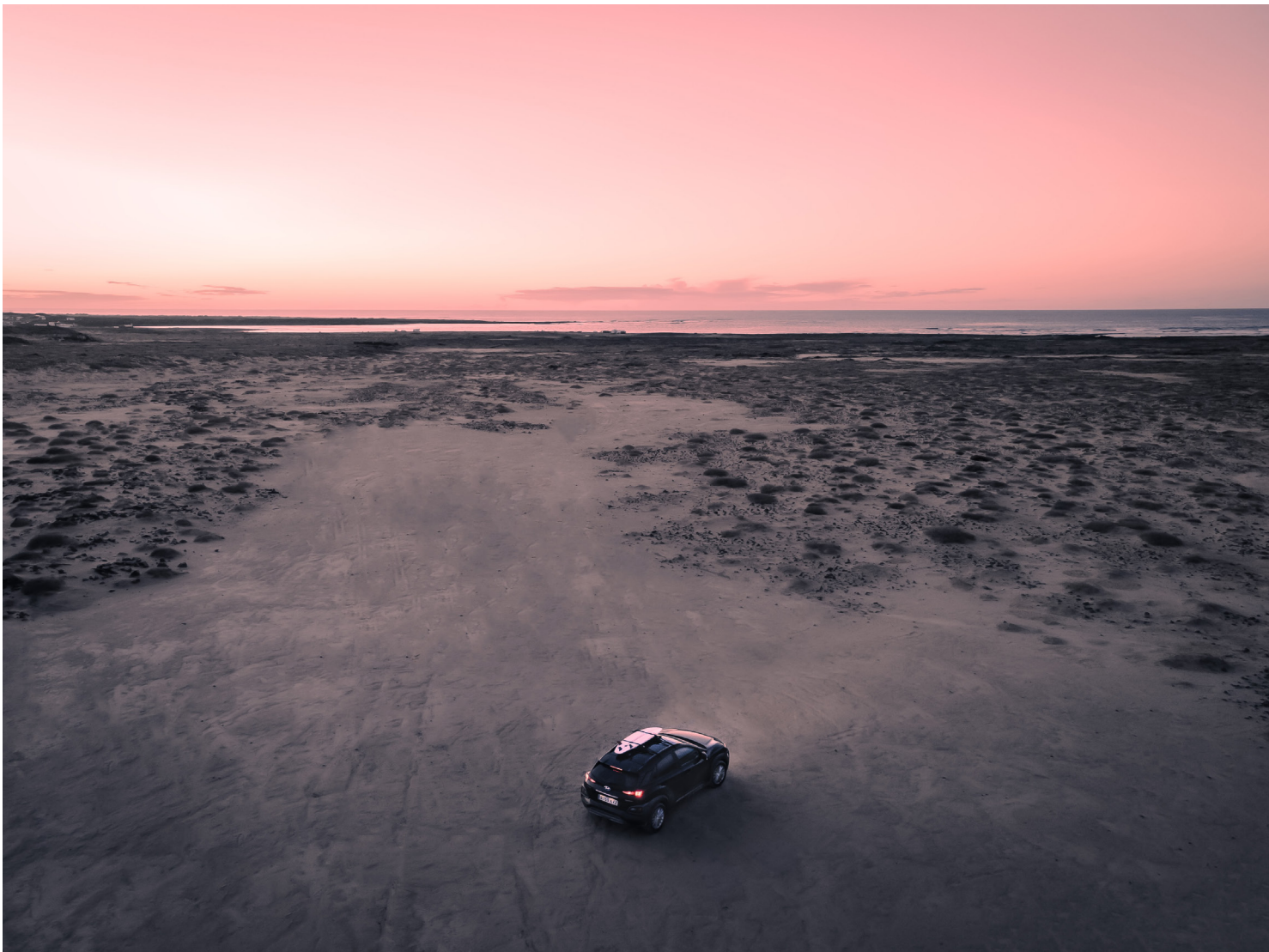
Proceeding to exhaustive static analysis after fuzz testing provides several advantages.

First, it's exhaustive. It removes every undefined behavior from your code.

Second, it provides you with formal proof—a mathematical guarantee—that can be used as evidence in reviews with security specialists, customers, and regulators. Because of the tool's soundness and the mathematical guarantee it offers, TrustInSoft Analyzer increases the level of confidence between suppliers and customers.

Finally, having accomplished exhaustive static analysis once for a given program, you'll find it is much less work than fuzzing when you modify your code. You're now working from a much cleaner baseline. You simply re-run the analyses you've already set up.

For detailed information on how fuzzing with TrustInSoft Analyzer can streamline the elimination of cybersecurity vulnerabilities, please see our white paper on that topic titled [Fuzzing and Beyond](#).



## How TrustInSoft Analyzer optimizes penetration testing

Penetration testing, also known as pen testing, is a software testing method that seeks to resolve system vulnerabilities in the same way an attacker will try to find and exploit them.

Penetration testing offers several advantages in achieving good cybersecurity. It helps identify risks that result from combinations of lower-risk vulnerabilities. It also helps identify risks that are difficult to detect with other methods. In the end, it helps establish trust with customers.

TrustInSoft Analyzer, through exhaustive analysis, proves the absence of all vulnerabilities caused by undefined behaviors within the program, and it helps eliminate those vulnerabilities before pen testing is performed. As a result, it enables pen testers to focus on other aspects of their work and reduce their work load.

Pen testing should be repeated when significant modifications are made to the product. But since TrustInSoft allows you to identify and eliminate vulnerabilities early in the testing process, it helps reduce pen testing iterations and minimizes overall verification cost.

# How TrustInSoft Analyzer exceeds the current state of the art in software analysis tools

## THE FOLLOWING IS A REVIEW OF THE KEY FEATURES AND BENEFITS OF TRUSTINSOFT ANALYZER.

### Exhaustive UB analysis

TrustInSoft Analyzer exhaustively hunts down all undefined behaviors existing in your software. That is its main purpose. Once those UBs have been eliminated, TrustInSoft Analyzer can then provide verifiable, mathematical proof that your software is free of undefined behaviors.

Because of its soundness and the mathematical guarantee it provides, TrustInSoft Analyzer effectively outperforms all of the verification techniques listed in ISO/SAE 21434 section 10.4.2.

So while ISO/SAE 21434 does not require tool qualification, the fidelity of the tool has been demonstrated to a high confidence level in the context of the automotive safety standard, ISO 26262.

### A fully qualified tool for proving UB absence

TrustInSoft Analyzer is a qualified tool for ISO 26262 (Road vehicles — Functional safety) for all ASIL levels. It has been recognized by the TÜV SÜD for its capacity to prove the absence of undefined behaviors in code. As a qualified tool, its results can be used as evidence to demonstrate reliable safety and compliance with ISO 26262.



### Target hardware emulation for embedded applications

In addition, TrustInSoft Analyzer provides a fully representative emulation of the target hardware on which your code will run. It enables you to precisely specify hardware characteristics such as endianness, padding, and memory alignment.

Target emulation allows you to verify your code within its target environment much earlier in the software cycle—without putting the actual hardware in the loop.

TrustInSoft Analyzer also implements a unique feature that faithfully represents the memory mappings between the program variables and chipset-specific memory regions. This feature allows the tool to analyze the software in the exact configuration in which it will run on the final hardware.

Thanks to this accurate representation of physical memory access, TrustInSoft Analyzer guarantees the detection of all memory-related vulnerabilities at the software level.

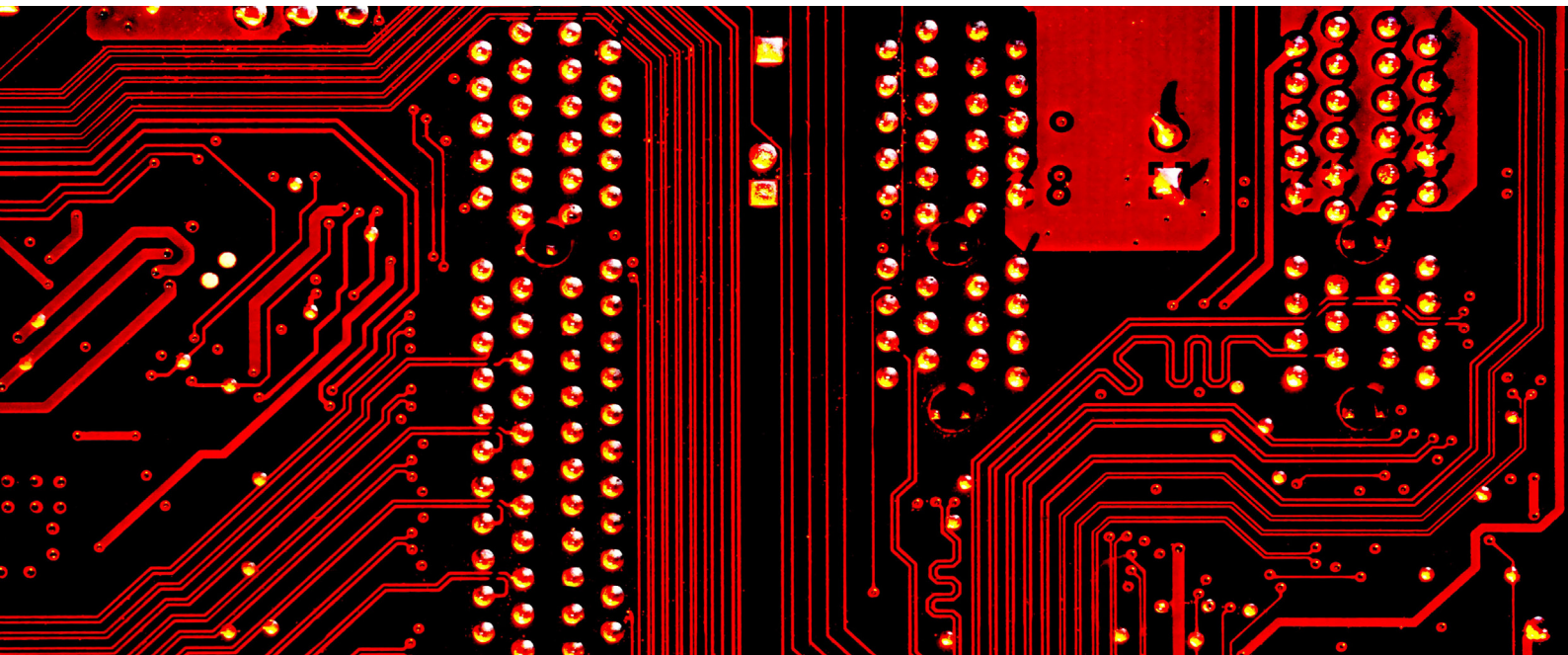
This capability is explained in detail in our white paper [“From Bare Metal to Kernel Code: How Exhaustive Static Analysis Can Guarantee Airtight Security in Low-level Software and Firmware.”](#)

### Faster, more efficient cybersecurity verification

Overall, TrustInSoft Analyzer reduces the amount of effort required to fully test software for cybersecurity vulnerabilities.

TrustInSoft Analyzer makes it possible to do the equivalent of billions of tests with a single, generalized test. It drastically reduces the number of vulnerabilities in your code early in the software development lifecycle, and it reduces the effort and time required to periodically identify and eliminate vulnerabilities that might be added by modifications during the course of that lifecycle. At the same time, it exhaustively detects all vulnerabilities resulting from undefined behaviors.

In the end, you have a mathematical guarantee—for the scope of the analysis performed—that your code is totally free of all undefined behaviors a hacker might exploit.





# Case studies

To illustrate how TrustInSoft increases the efficiency of software testing for cybersecurity, we'll look at three examples of how the tool helped eliminate hard-to-find vulnerabilities in commercial software and firmware products.

## Case study: STMicroelectronics AIS2DW12 Accelerometer Driver Analysis

One of the trends transforming automobiles into advanced cyber-physical systems is the integration of dozens of sensors that provide useful data to a vehicle's ECUs. This sensor integration has created a high degree of coupling among a modern vehicle's sensor, communication, and control layers.

Cyberattacks against sensors can therefore compromise the security of the vehicle. It is imperative that these sensors' software components be totally free of undefined behaviors.

TrustInSoft performed an independent analysis of the device driver of the STMicroelectronics AIS2DW12, a 3-axis accelerometer for the automotive industry. Using TrustInSoft Analyzer, we were able to identify and fix a buffer overflow fault in the driver in less than 1.5 hours. That includes the time we needed to familiarize ourselves with the sensor's datasheet and driver.

After the fault was corrected, we ran the test again. TrustInSoft Analyzer confirmed that no undefined behaviors remained in the driver regardless of the hardware's register contents.

## Case study: A leading autonomous driving platform provider

One of our customers needed a new way to ensure the safety, reliability, and security of their software-driven platform. Their goal was to verify that their ADAS software platform's reaction to the position of the vehicle and any surrounding objects (including living beings) would not cause any undefined behavior that would pose risks to the user's safety or security.

To achieve that goal, they turned to exhaustive source code analysis using TrustInSoft Analyzer.

Their team ran an analysis on a key library, an embedded C++ software stack for vehicle maps and positioning. This collection of C++ classes stores a map and associated objects in the autonomous car software framework. It contains over 300,000 lines of code.

If this analysis were done using traditional testing methods, executing the code on a number of discrete input sets, only a very limited portion of possible vehicle positions on the map could be covered (as illustrated in Figure 2). Even using a test framework or a fuzzing tool, one could cover only a small portion of all possible test cases.

With TrustInSoft Analyzer, however, formal methods extend test coverage beyond the reach of traditional tests. TrustInSoft Analyzer generalizes test inputs through abstraction and exhaustively detects all undefined behaviors present in the code (Figure 3).

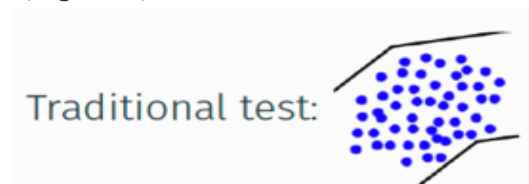


Figure 2: Verification coverage using traditional testing methods



Figure 3: Test coverage using TrustInSoft Analyzer

Through abstraction, TrustInSoft Analyzer was able to test all possible positions of the vehicle and surrounding objects in a given area. The generalized (abstract) test coverage represented the equivalent of more than  $85 \times 10^{36}$  possible positions and discrete tests.

During the initial analysis, TrustInSoft Analyzer identified a number of undefined behaviors. After the software was corrected and reanalyzed, TrustInSoft Analyzer was able to provide a mathematical guarantee of the absence of undefined behaviors in the Maps and Positioning class of the platform for all possible positions of the vehicle and all objects in its environment.

With total coverage, TrustInSoft Analyzer verified that there would be no software runtime errors that could jeopardize safety or security.

## Conclusions

ISO/SAE 21434 is an important standard that provides a good framework for cybersecurity risk management. It is a standard based on the experience of a community of experts and the latest best practices in the automotive industry.

In the context of ISO 21434, suppliers need to allocate resources to cybersecurity (both human and financial) from the outset of every project. Only by managing cybersecurity risk from the very beginning can they avoid the potential negative cost impacts associated with repetitive activities throughout the product lifecycle.

Automotive software providers must exhaustively hunt down all undefined behaviors in their code. These bugs are normally very difficult to detect under standard testing conditions and are typically the type of vulnerabilities exploited by hackers following product deployment.

Sound and exhaustive static analysis, like that provided by TrustInSoft Analyzer, ensures the cybersecurity of automotive software. It guarantees the absence of undefined behaviors and can prove that your software behaves exactly as specified. Exhaustive and sound static analysis tools are a significant, strategic part of the next-generation automotive software toolchain solution.

TrustInSoft Analyzer helps automotive OEMs and their suppliers comply with ISO/SAE 21434. In particular, it can contribute significantly to meeting the requirements of Section 10.4.2., which covers software integration and verification.

TrustInSoft Analyzer contributes to cybersecurity risk management at the software level in several ways:

- It contributes to cost-effective, in-depth analysis of specific cybersecurity risks (undefined behaviors).
- It proves the absence of undefined behaviors and enables suppliers to demonstrate the security of their products at initial delivery by detecting vulnerabilities earlier in the verification cycle.
- It reduces the cost of repetitive software testing activities throughout the product's lifecycle.
- It reduces the frequency of software updates and their associated costs and risks.
- It contributes to reducing corporate risk on a number of fronts, including legal, financial, brand reputation, and time-to-market.



# References

1. Haas, R., and Möller, D., [Automotive connectivity, cyber attack scenarios and automotive cyber security](#), 2017 IEEE International Conference on Electro Information Technology (EIT), Lincoln, NE, USA, May 2017, pp. 635-639.
2. [CWE Glossary: Vulnerability](#), MITRE Corp.
3. Sandler, M., [UN Regulation No 155: What You Need to Know about UN R155.](#), Cyres Consulting, June 2022.
4. [GRVA, Proposal for the Interpretation Document for UN Regulation No. \[155\] on uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system](#), UNECE, November 2020.
5. Miller, C., & Valasek, C., [Adventures in Automotive Networks and Control Units](#), IOActive, October 2014.
6. Miller, C., & Valasek, C., [Remote Exploitation of an Unaltered Passenger Vehicle](#), IOActive, August 2015.
7. Currie, R., [Developments in Car Hacking](#), SANS Institute, January 2016.
8. Haas, R., and Möller, D., [Automotive connectivity, cyber attack scenarios and automotive cyber security](#), 2017 IEEE International Conference on Electro Information Technology (EIT), Lincoln, NE, USA, May 2017, pp. 635-639.
9. Pham, G., [How is C++ Still Used for the Automotive Industry](#), InApps Technology, March 2022.
10. Kovacs, E., [Cars Plagued by Many Serious Vulnerabilities: Report](#), SecurityWeek, August 2016.
11. National Vulnerability Database, [CVE-2017-9633 Detail](#), NIST, July 2017.
12. National Vulnerability Database, [CVE-2017-9633 Detail](#), NIST, July 2017.
13. ICS Advisory: [Continental AG Infineon S-Gold 2 \(PMB 8876\)](#), CISA, July 2017.
14. ICS Advisory: [Continental AG Infineon S-Gold 2 \(PMB 8876\)](#), CISA, July 2017.
15. National Vulnerability Database, [CVE-2019-13582 Detail](#), NIST, November 2019.
16. National Vulnerability Database, [CVE-2019-13582 Detail](#), NIST, November 2019.
17. [Exploiting Wi-Fi Stack on Tesla Model S](#), Tencent Keen Security Lab, January 2020.
18. ICS Advisory: [BadAlloc Vulnerability Affecting BlackBerry QNX RTOS](#), CISA, August 2021.

19. National Vulnerability Database, [CVE-2021-22156 Detail](#), NIST, August 2021.
20. ICS Advisory: [BadAlloc Vulnerability Affecting BlackBerry QNX RTOS](#), CISA, August 2021.
21. Black, P.; Badger, L.; Guttman, B.; Fong, E.; [Dramatically Reducing Software Vulnerabilities: Report to the White House Office of Science and Technology Policy](#); National Institute of Science and Technology (NIST), November 2016.
22. [SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems \(2016 Edition\)](#), SEI, June 2016.
23. [The Bugs Framework/CERT C Coding Standard](#), NIST.



The logo for TrustInSoft, featuring the word "TRUST" in a serif font, a square icon with "IN" inside, and the word "SOFT" in a serif font.

Since our beginnings, TrustInSoft Analyzer has been adopted by industry-leading companies around the world to ensure sound cybersecurity in their low-level code.

Founded in 2013, TrustInSoft developed a game-changing product for software code analysis. TrustInSoft Analyzer is a hybrid code analyzer that combines advanced static and dynamic analysis techniques together with Formal Methods to mathematically guarantee C/C++ code quality, reliability, security and safety. TrustInSoft has customers worldwide in the automotive, IoT, telecom, semiconductor, aeronautics and defense industries. TrustInSoft has received awards and recognition from the NIST, RSA and Linux Foundation.

To learn more about TrustInSoft Analyzer, visit

[trust-in-soft.com/product/trustinsoft-analyzer](https://trust-in-soft.com/product/trustinsoft-analyzer).

If you'd like to speak with a TrustInSoft technical representative about how TrustInSoft Analyzer can meet your organization's specific needs, contact us by email at

[contact@trust-in-soft.com](mailto:contact@trust-in-soft.com)

Phone : +33 1 84 06 43 91 or +1 (408) 829-5882