Karamba Security

December 2018

Autonomous Security

Vehicle Self-Protection against Cyberattacks

Version 2.0

Contents

Executive Summary

In a race to secure connected and autonomous cars, cybersecurity experts have tried to adapt enterprise security methodology to the vehicle computing environment. The results have been suboptimal because vehicles have a unique set of constraints and requirements. Most importantly, passenger safety is at risk, leaving no room for errors.

Even though enterprise data security has been modified for vehicles, it still leads to unwanted ramifications, including: 1) false positives; 2) remediation lag; 3) unpatched zero-day vulnerabilities; 4) slowed performance; and 5) delayed pen testing.

These five safety and security constraints are even more problematic when non-deterministic security products such as anomaly-based intrusion-detection systems are used for attack prevention. By classifying potential threats based on heuristics or probabilities, these solutions lead to false positives. A “better safe than sorry” approach is fine for data security, but in a car, blocking legitimate commands, like braking, could lead to car crashes and to compromised passenger safety. False positives are not an option when driver, passenger and pedestrian safety are at stake.

Collectively, these hazards present a mandate to vehicle cybersecurity protection that ensures passenger safety with a negligible performance impact. To accomplish this goal, deterministic solutions hold the most promise. The “vicious cycle” of detection–analysis–mitigation introduces a detection gap which is unacceptable in mission-critical systems and infrastructure. A more deterministic approach is thus called for.

In this white paper, we first discuss the key differences between enterprise security and vehicle cybersecurity. Based on the distinctions, and using a powerful threat-landscape analysis toolset, we have formulated a set of requirements and strong preferences for the design of vehicle cybersecurity.

In the latter part of the white paper, we introduce the *Autonomous Security*® paradigm, designed to meet these requirements. This model embeds deterministic security into the vehicle’s computer binary code during the software build process, and authenticates in-vehicle communication with no network overhead.
By hardening both the electronic control units (ECUs) and the communication between ECUs based on factory settings, Autonomous Security creates self-defending devices, enabling the car to protect itself from attack attempts.

There is no need for updates, day-to-day management, or cloud access — encumbrances that impede product roll-out, retrofitting, and in some cases, security itself.

Part I: Obstacles that Hinder Vehicle Security and Safety

To create an effective cybersecurity strategy, software architects must examine all obstacles that limit vehicle security options. For starters, there is a distinction between protecting data and protecting lives. To deliver proactive defense without putting a strain on limited-resource ECUs, in-vehicle threat protection must avoid:

1. **False positives**
2. **Remediation lag**
3. **Zero-day vulnerabilities**
4. **Slowed performance**
5. **Delayed pen testing**

False Positives

False positives cannot be tolerated in the context of vehicle safety, given the possible consequences when a legitimate command is mistakenly cancelled. Similar to heuristic-based solutions, signature-based security, which matches signature files to a continually-updated database, also generates false positives. Even if heuristics and signature-identification techniques were to vastly improve, false-positives cannot be completely avoided in heuristic models. This is an unacceptable scenario when lives are at risk.

Furthermore, there is a “false-positive paradox,” meaning that the odds of a false-positive can, ironically, be greater than the odds of a true threat.[1]

The bottom line: heuristic cyber protection is not the right solution for securing in-vehicle systems and embedded systems. Even if there is a driver override option, it could not be triggered fast enough when split-second decisions are critical.

Remediation Lag

"Decision lag" is a term used by economists to refer to a government's slow reaction to economic events. In a similar fashion, "remediation lag" refers to a delayed time-to-detection, the window of time it takes to discover a security breach or vulnerability. Cyberattacks exploit hidden security vulnerabilities in the target's software. In vehicles and fleets, when an attack is detected, it could take weeks to identify the vulnerability that was exploited, remediate it, run tests, and deploy a security update in all affected vehicles. According to Gartner, the average time to detect a breach in the Americas is 99 days, and the average cost is $4M.[2]

The biggest problem resulting from that remediation lag is the security gap created when ECUs with known vulnerabilities remain open to attacks while waiting for a fix. Hackers know they can take advantage of this delay and launch repeated attacks exploiting the same vulnerability. A bombardment of attacks during this period of time can create irreversible damage to a brand.

Zero-Day Attacks

Preventing exploits of unknown vulnerabilities (i.e. zero-day attacks) in the ECU code is key to avoiding this lag. Enterprise zero-day attack-protection strategies typically include statistical and behavioral heuristics plus signature identification, which is functionally heuristic.[3]

The problem with using heuristics to combat zero-day threats is not false positives but false negatives. Heuristic- and signature-based security solutions too often fail to detect and block unknown malware. They are more effective when used in combination, but given their margin of error, they cannot always identify novel attacks. Further, enterprise zero-day protection technologies are compute-intensive. They require strong servers or server farms for the quarantine, sandboxing and analysis process.[4] We claim that such technologies conflict with the vehicle performance requirements, if they are executed on the cloud, and that they cannot run locally on the car's systems, due to limited CPU and memory resources. Zero-day attacks present an extreme challenge to vehicle cybersecurity.

Slowed Performance

When carmakers deploy cybersecurity in resource-constrained networks and ECUs, performance is always a concern. It is important to determine network overload and computational and RAM footprints to understand how much strain the security processing will put on the bus, the CPU, and the memory. Significant overhead may dictate that hardware must be added to the vehicle's networks and ECUs in order to meet performance requirements. Physical segregation can also add to the hardware footprint, further draining limited resources and increasing the cost of the car's bill of materials.

Delayed Pen Testing

Penetration testing to identify vulnerabilities is typically done when the ECU software is ready to be deployed on a target system. By the time a flaw is found, time-to-market can be increased and substantial costs involved.

Avoiding the Security Constraints

The obstacles that hinder vehicle cybersecurity must be addressed to achieve the level of safety and performance needed in connected and autonomous cars. With this goal in mind, Karamba Security has identified three primary defense strategies:

1. **Whitelisting of network messages, ECU applications and control flow**
2. **Embedding native security**
3. **Zero-day protection: Eliminating the need for security updates**

Whitelisting of Messages, Applications and Control Flow

To overcome multiple challenges specific to vehicles, the backbone of cybersecurity should be deterministic, not heuristic. When filtering incoming data or messages, predictive algorithms will lead to false positives, remediation lag, and failure to prevent zero-day attacks. For these reasons, deterministic security built on the principle of multi-dimensional whitelisting is ideal.

The National Highway Traffic Safety Administration (NHTSA) cybersecurity guidelines state, “NHTSA is focusing on solutions to harden the vehicle’s electronic architecture against potential attacks…” The guidelines specifically recommend “strict whitelist-based filtering of message flows,” as a means to harden the vehicle computer environment.”[5] Karamba Security has taken that concept further, to whitelist all network messages, legitimate runnable binaries and in-memory function call sequences.

Embedding Native Security

A second mitigation is to strictly follow the SAE International guideline to install cybersecurity programs during the build process of the vehicle's computer systems.[6] Embedding cybersecurity in the ECU firmware enables the car to protect itself from cyberattacks, without relying on cloud connectivity, or cloud-to-car response time.

Native security can also be more proactive, since it prevents the attack before the ECU is compromised, i.e. before the attacker succeeds to hack the car and issue malicious commands over the car's network.

Eliminating the Need for Security Updates

Vehicle cybersecurity should be designed to run continuously and effectively without relying on software updates. As known, it can take weeks, even months, to isolate a software vulnerability and run a fix through the entire process, from development to deployment. If repetitive attacks target a specific car model and affect passenger safety during this vulnerable period, the car's brand will suffer severe damage that may lead to irreversible consequences.

To avoid these consequences, OEMs and Tier-1 suppliers need to deploy vehicle security protection that is not dependent on updates. In the following sections, we will explain how this can be achieved.

In addition, if penetration testing can be carried out on virtualized instances of vehicle ECUs, much of the preventive measures can be taken during the system design phase, eliminating vulnerabilities in advance.

Part II: Turning Strategies into Best-Practice Protection

To this point, we have discussed the challenges of securing vehicle networks and computer systems from a theoretical perspective. In the following pages, we will look at how the conclusions drawn above can be put into practice. We will demonstrate a new Autonomous Security model that effectively removes the security constraints to create self-defending cars.

More importantly, you will learn how this type of security can autonomously protect vehicle networks and ECUs against hacking attempts, including zero-day attacks. Autonomous Security removes the cybersecurity burden from developers, while meeting design-build best practices and vehicle requirements.

Embedding Native Security in ECUs

Unlike enterprise network and server systems, vehicle systems are not user-changeable. This means that the network messages and binary code in an in-vehicle ECU can be sealed to prevent message spoofing or unauthorized changes, i.e., attempts to exploit security vulnerabilities. Only the OEM would be able to make modifications to the car network and ECUs, when needed.

In this way, cybersecurity remains stable over the life of the vehicle, eliminating the need for continuous malware signature updates and security patches.

As long as performance impact is negligible, this approach also fits together with the best practice of installing security software in the original components.

Hardening ECUs to Factory Settings

As previously noted, multi-dimensional whitelisting is an effective approach to vehicle cybersecurity. NHTSA also advises, “Developer access should be limited or eliminated if there is no foreseeable operational reason for the continued access to an ECU for deployed units.”[7] As manufacturers strive to limit post-deployment modifications, hardening the vehicle networks and ECUs offers the added benefit of a more stable environment that is easier to secure over the life of the vehicle. Karamba Security has developed a blueprint for this purpose. This model for in-vehicle security incorporates the following features:

- In-memory validation
- Whitelisting of executables
- Authentication of network messages
- Enabling OEM-sourced updates
- Protection on hypervisors

In-Memory Validation

In-memory validation monitors the program execution flow to ensure it remains within the expected execution points. If a function call or a return pointer deviates from the execution flow which was automatically mapped during the ECU build process, the call is blocked — before any damage occurs.

When such binary function-call mapping is embedded into the ECU’s software, security decisions are made locally.

The ECU is no longer a potential attack surface; It becomes a self-defending device, impervious to in-memory attacks like buffer overflows and heap overflows. There is no need for malware signature updates since the in-memory security policy is generated based on factory settings.
VM In-Memory Validation: In the case of a virtual machine (VM), this protection blocks guest attacks from exploiting in-memory vulnerabilities at the hypervisor level. In-memory validation blocks any attempt to perform illegitimate functions or download malware.

Whitelisting Executables

ECU-hardening should also include a whitelist enforcement component that integrates with the OS program-loading and file-access services.
All executables can be checked against the whitelist including files (operating system and applications), shared objects (libraries), and scripts. Each time any binary is loaded, its signature can be calculated and compared to a database of approved application signatures.[8]

If the binary is on the whitelist, it is permitted to run. If a binary’s signature is not on the whitelist, it is not a legitimate component originating within the ECU's factory settings. As soon as malicious code attempts to be loaded to memory, the security filter stops the binary from loading. This includes protection against attacks that drop malware onto the ECU flash storage.
VM Executable Whitelisting: In the case of a VM, whitelisting blocks guest malware from a drop-and-execute attack on the hypervisor.

Authentication of Network Messages

At the inter-ECU level, a new approach is called for that overcomes technological and safety constraints and protects the vehicle by hardening in-vehicle networks against unauthenticated CAN activity.

It is essential that this is done without adding to the network load.

In addition, the technology cannot entail custom implementations on the part of OEM developers which would add complexity and additional expenditures.

Enabling OEM-Sourced Updates

Cybersecurity “overkill” is not practical and can decrease security. If the protection mechanism is designed to block all changes blindly, it would block legitimate ECU software updates made by the OEM. When a feature is added or enhanced, the security solution must be flexible enough to allow these updates and generate corresponding policy changes. Only then would protection continue without false positives.

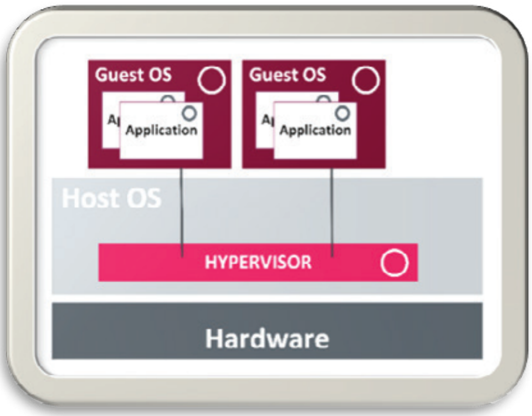
The OEM update mechanism should be able to incorporate new validation rules seamlessly any time the ECU software is updated, so that new components are whitelisted in the same secure manner as they were during the original build.

Protection on Hypervisors

Hypervisors allow VMs to be partitioned in a way that enables defining safety certifications for each VM independently. This means non-critical components do not require the same level of certification as critical components, which can take a heavy toll in terms of time and expense.

Hypervisor configurations also need protection for their unique environment. For starters, vulnerabilities in the hypervisor itself can cause breaks in the segregation that it was designed to provide.

Adding to the complexity, software programs run on various operating systems on the same hardware via the hypervisor’s connected VMs. These software applications are often subject to attacks involving unauthorized access resulting from faulty drivers or maliciously-altered permission settings.[9]



Hypervisor Configuration – Protection Needs

As shown by Intel's Meltdown vulnerability [10] – an attack on the ECU kernel in a virtualized stack – a hacker could execute an API call to the hypervisor from a spoofed VM that masquerades as a legitimate VM.

To combat these threats, cybersecurity is needed on individual applications, on guest operating systems, and on the hypervisor, as illustrated by the circles in the diagram.

Threat Landscape Analysis

Custom-designed "sensor" nodes in a wide variety of in-vehicle ECU deployments around the globe – both real and virtualized – enable the following actions:

- Sniffing the ECUs and their networks (CPU, control logic and memory activity as well as messaging)
- Collecting real and potential threat data
- Processing the findings to create the knowledge and strategies needed by OEMs and Tier-1 suppliers to prevent the observed exploits.

Automating Protection for the Life of the Car

With the parameters of an effective vehicle security solution defined, the next step is to ensure the build process meets the security requirements — without placing a burden on OEMs or Tier-1/Tier-2 suppliers. ECU developers in all tiers should not be required to learn how to deploy, configure, and manage the cybersecurity solution; nor to supply the cybersecurity developer with ECU code.

It is also important to deploy security that is lightweight, since most resource-constrained ECUs are overloaded. Any security process that significantly increases the usage of the ECU RAM or significantly degrades CPU performance will have an effect on ECU operation and may result in compromising safety. In low-cost ECUs, heavy validation processing could cause message delays and collisions.

For these reasons, Karamba Security recommends a light-weight, embedded solution that automatically generates the security policy during the software build process. Ideally this security policy is then encrypted and signed with a private key to prevent malicious tampering. The signed security policy and the public key verifying its authenticity should then be embedded in the ECU.

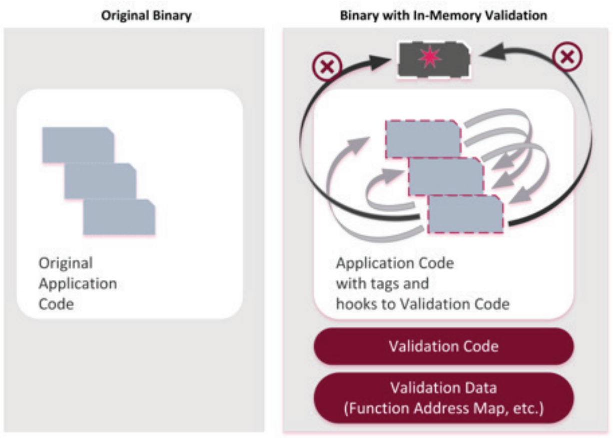
As an added benefit, embedding native security into the binary code minimizes overhead so there is negligible performance penalty.

To accomplish this, the security program must trace the ECU's factory settings in multiple layers. In this way, each layer of protection seals the vehicle's ECU software against different types of attacks.

In the following sections, we will examine the process layer by layer.

First Layer: In-Memory Hardening

The first layer should generate the security policy used by the In-Memory Validation engine. A static analysis engine is used to analyze all binaries (not the source code) of the vehicle's ECU. The engine then automatically maps all valid function-call sequences and call locations in the system. With this call graph, the In-Memory Validation engine ensures in run-time that only legitimate function calls are executed.
It also blocks any attempt to load malware directly into memory.


Original Binary
Original Application Code
Binary with In-Memory Validation
Application Code with tags and hooks to Validation Code
Validation Code
Validation Data (Function Address Map, etc.)

In-Memory Validation

To illustrate, if an attacker were to inject malicious code into the memory of a process, the stack would show a call or a return to a location that is not part of the original call sequence. If the program tries to switch to an unknown location or address, In-Memory Validation would detect this deviation and block the execution — before the hackers could cause any damage to the vehicle or its embedded systems.

Second Layer: Whitelisting Executables

The second step is to generate the security policy used by an application whitelisting enforcement engine All binaries that are part of the vehicle's software are automatically scanned. These binaries provide a complete list of all the programs and scripts that are allowed to run on the ECU.

For each binary file in the system, the security program then creates a unique signature, based on the content of the file. These binary signatures ensure a closed authentication test during runtime.

Third Layer: Authenticating In-Vehicle Messages

Finally, an ultra-light network security solution authenticates communication between ECUs—without slowing performance.

From message-type analysis to in-place message encryption, the Authentication-Encryption (AE) mechanism has been designed to maximize protection while minimizing the burden on the vehicle's limited-resource systems.

The result is zero network overhead in the vehicle, and a negligible impact on latency.

Measuring Performance

Vehicle cybersecurity protection cannot be added at the expense of slowing performance. Any proposed solution must be tested for acceptable levels of added processing associated with validation tasks. There are also additional memory requirements for data structures accessed by the validation code.

For in-vehicle-network protection, latency is the key indicator of performance impact. For ECUs, impact can be estimated by a set of performance indicators including CPU-utilization rate changes, an increase in the root-file system size, and a decrease in available RAM. In actual system-performance tests, however, it is essential to ensure measurements remain within the product performance specification's allowed limits after security is added.

Incident Response and Forensic Reporting

Autonomous Security mechanism automatically issues instantaneous threat alerts when an attack is detected. These alerts identify which system is being attacked to inform the incident response team.

In accordance with NHTSA best practices, Autonomous Security solutions record any anomalous activity or attempts to access ECUs. These incident logs are then sent to forensic experts for analysis.[11] This data should also be shared with the Auto-ISAC community.[12] A heuristic defense system helps ECU developers better understand the types of attacks targeting their programs, so that they can patch any vulnerabilities in future iterations of their product.

With this goal in mind, the following elements are processed and analyzed to create a detailed threat analysis report:

- File system operations
- Network operations
- Peripheral resource operations (e.g. CAN bus, DVD, CD-ROM, and USB ports)
- Internal process communications
- Process and thread operations
- Debugging attempts

This information is used to create analytic reports that include all forensic data that was collected on the system around the time of the attack, including:

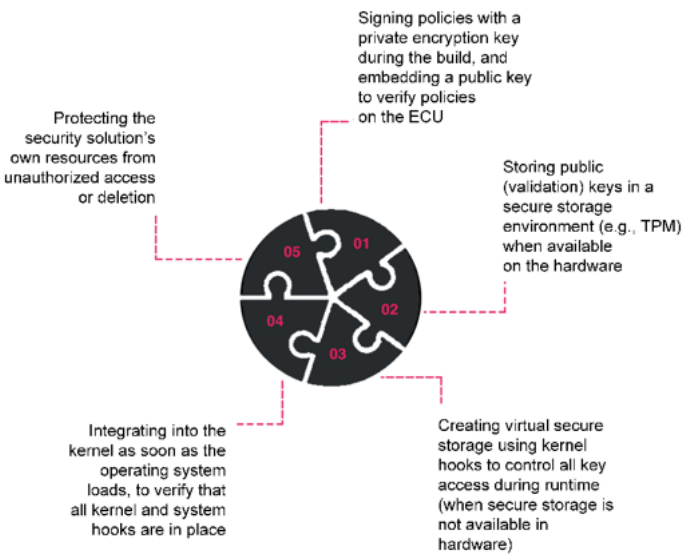
- The exploited process
- External connections involved (i.e. a peripheral port or network communication)
- The type of attack (malicious application, in-memory code injection, etc.)
- The malicious binary trails in the file system

This type of data enables ECU software developers to identify and fix the vulnerabilities that leave vehicle systems exposed to potential threats. Records of malicious attacks are kept for future reference and for cross-reference with other attacks.

Anti-Tampering

Vehicle cybersecurity must empower the security software to protect itself against any attempts to modify its policies, remove enforcement engines, or hide malicious activities. This type of anti-tampering can be achieved through a combination of software and, when available, hardware capabilities that verify the integrity of the policies and protection mechanisms.

As detailed in the illustration, tampering protections ensure that every ECU is a self-defending device.


Signing policies with a private encryption key during the build, and embedding a public key to verify policies on the ECU
Protecting the security solution's own resources from unauthorized access or deletion
Storing public (validation) keys in a secure storage environment (e.g., TPM) when available on the hardware
05
01
02
04
03
Integrating into the kernel as soon as the operating system loads, to verify that all kernel and system hooks are in place
Creating virtual secure storage using kernel hooks to control all key access during runtime (when secure storage is not available in hardware)

Anti-Tampering Measures

Ease of Deployment

Another practical aspect of vehicle Autonomous Security relates to the time and budget constraints under which software developers operate. When a cybersecurity solution automatically develops customized security policies, there is no prerequisite training for the ECU development team. By the same token, the cybersecurity development team does not need to learn the ECU implementations.

Upload of an ECU image is all that is needed on the part of an OEM or Tier-1 supplier in order to participate in and benefit from shared threat-landscape analysis data.

Advantages of Hardening the ECUs

By following these security guidelines, developers can achieve unparalleled protection of vehicle computer networks and ECUs, with negligible performance impact and without adding to the ECU developer's workload.

A deterministic solution built on these design principals provides many advantages:

- Stops foreign code and invalid function calls
- Stops network spoofing attempts
- Eliminates any risk of false positives
- Supports all ECUs as an embedded solution that’s OS and CPU agnostic
- Minimizes network, CPU and memory overhead to ensure optimal performance
- Seals the ECU binaries
- Protects software running on hypervisor VMs to prevent unauthorized access attacks
- Automates the security development process, reducing the time to market
- Expedites retrofitting
- Secures embedded systems over the lifetime of the vehicle
- Installs and operates without the need for developer resources or ongoing administration
- Operates 24/7 without any human intervention or Internet connectivity
- Provides immunity regardless of unpatched vulnerabilities
- Blocks hacking attempts instantaneously, eliminating time-to-detection delays
- Delivers detailed threat data for comprehensive forensic analysis

The Big Picture: Autonomous Security

Always-On Threat Protection

Implicit in the Autonomous Security model is the peace of mind that comes with 24/7 protection, with no updates or manual inspections required. It is essential to give drivers and passengers the confidence of continuous security, regardless of their mobility choices.

Many Facets of Autonomy

For Karamba, Autonomous Security also means a simple process to embed security into the vehicle. Deep insights on ECU threat levels based on intelligence services during the development phase are translated into seamless image-level hardening. The protection of the ECU and in-vehicle communication is added automatically during software builds, with no developer intervention or impact on the production process.

Award-Winning Innovation

Recognized for its safety-related initiatives, Karamba Security received a *Best Cybersecurity Product/Service Award* from TU-Automotive in both 2017 and 2018, and was selected as a *Gartner Cool Vendor* 2018 for IoT security.

Karamba's portfolio of offerings has been designed to assure Autonomous Security.

Carwall®
Embeds vehicle ECU hardening within the image, assuring runtime integrity with zero false positives and negligible performance impact

SafeCAN®
Seamlessly secures In-Vehicle Networks against unauthenticated CAN commands from compromised ECUs or OBDII dongles, with no network overhead

ThreatHive™
A unique, cost-reducing tool for exposing ECU vulnerabilities early in the design-and-development process, by leveraging the intelligence gained from analyzing real cyber attacks

Research & Consulting
Enables OEMs and Tier-1 suppliers to minimize the security risks of their products with threat assessment, defense analysis, and penetration testing

References

1. en.wikipedia.org/wiki/False_positive_paradox

2. The Gartner Group, “The Gartner IT Security Approach for the Digital Age”, June 12, 2017.

3. Kaur, R.; Singh, M., "Efficient hybrid technique for detecting zero-day polymorphic worms" (2014) in Hammarberg D., “The Best Defenses Against Zero-day Exploits for Various-sized Organizations”, Sept. 2014, p.3.

4. Yao, et al, “Hopf Bifurcation in an SEIDQV Worm Propagation Model with Quarantine Strategy” (2012) in Hammarberg, op cit.

5. NHTSA, “Cybersecurity Best Practices for Modern Vehicles”, Oct. 2016, p.19, Sect. 6.7.7.

6. Boran, L., SAE International J3061, “Overview of Recommended Practice; Cybersecurity Guidebook for cyber-physical vehicle systems”, January 2016.

7. Ibid, p.17, Sect. 6.7.1.

8. A cryptographic hash is like a signature for a text or a data file. Hash is a one-way function; it cannot be decrypted. This makes it suitable for password validation, challenge hash authentication, anti-tampering, and digital signatures.

9. CVE Details – for example, KVM-related vulnerabilities: https://www.cvedetails.com/google-search-results.php?q=linux+kvm&sa=Search.

10. https://www.us-cert.gov/ncas/alerts/TA18-004A; https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2017-5754: “Systems with microprocessors utilizing speculative execution and indirect branch prediction may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis of the data cache.”

11. National Highway Traffic Safety Administration. (2016, October). Cybersecurity best practices for modern vehicles. (Report No. DOT HS 812 333). Washington, DC: Author. p.20, Sect. 6.7.9.

12. Automotive Information Sharing and Analysis Center, https://www.automotiveisac.com.

Karamba Security

Karamba Security provides Autonomous Security, the industry-leading end-to-end preventative solutions that are designed, from the ground up, for resource-constrained environments.

This award-winning self-protecting solution provides automotive cybersecurity that is built into the embedded software. Karamba prevents zero-day cyberattacks with zero false positives while assuring negligible performance impact. Karamba technology has proven seamless integration into the software development lifecycle, which fits modern architectures in vehicles and IOT devices.

Karamba Security is led by a balanced executive team of cybersecurity experts and seasoned entrepreneurs.

Karamba Security USA
Tel: +1 248 574 5171

Karamba Security Germany
Tel: +49 151 1471 6088

Karamba Security Israel
Tel: +972 9 88 66 113

www.karambasecurity.com | contact@karambasecurity.com