Kaspersky ICS CERT

kaspersky



Cybersecurity of over-the-air update mechanisms for vehicles

Sergey Melnikov, Stepan Rybakov

15.10.2026 Version 1.0

Contents

Introduction ... 3
How threats to automotive software are evolving ... 5
Attacks on vehicle manufacturers' infrastructure ... 5
Vulnerabilities in digital services for vehicles ... 6
Vulnerabilities in aftermarket electronic equipment ... 7
From infrastructure attacks to attacks on connected fleets ... 8
Update security challenges ... 8
Protecting the update process at the vehicle level ... 11
Security measures ... 11
Root of trust ... 11
Securing connections with backend servers ... 12
Hardware isolation and trusted execution environment ... 13
Protection against physical tampering and side-channel attacks ... 13
Hardware support for unit lifecycle management ... 14
Fault tolerance and critical data redundancy ... 15
Memory management ... 15
Configuration hardening ... 16
Logging ... 16
Authenticity verification and replay attack protection ... 17
Secure boot ... 17
RxSWIN identifiers ... 18
Identification of software modules ... 18
Managing the RxSWIN registry for an individual vehicle ... 22
RxSWIN identifiers are stored on FOTA Target ... 23
RxSWIN registry stored on the FOTA Master ... 23
Secure-by-design control units ... 24
Conclusion ... 24

© 2026 AO KASPERSKY

Introduction

Since the 1980s, when electronic control units (ECUs) first appeared in mass-produced cars, their number has grown substantially: a modern vehicle[1] incorporates more than 100 such units. In the 1990s, automotive manufacturers began using microcontrollers with rewritable flash memory in ECUs. This innovation made it possible to update unit firmware in vehicles that are already in use on public roads.

In the 2010s, automakers began implementing over-the-air (OTA) update features in production vehicles. In the early 2020s, at least 309 models from 23 manufacturers offered this option. According to forecasts by SBD Automotive, over the next 10 years, the percentage of vehicles capable of receiving OTA updates on roads in the US, EU countries, and China will more than triple, reaching 30–40% of the total vehicle fleet in these regions (Fig. 1).

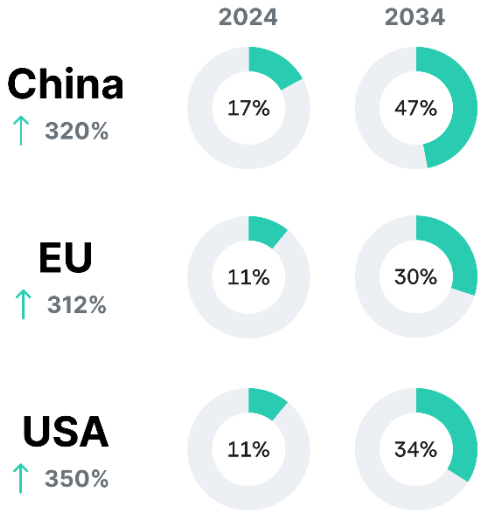


Fig. 1. Current and projected shares of vehicles capable of receiving OTA updates as a percentage of the total vehicle fleet in the US, EU countries, and China

The frequency of updates is also increasing. According to a VicOne study, until recently, an average of 4–5 OTA updates per year were delivered to a single

[1] This article covers vehicles that are subject to UNECE requirements for software update security: passenger and commercial vehicles (categories M and N), trailers (category O), and agricultural and forestry equipment (categories R, S, and T). A detailed list of vehicle categories subject to the requirements of UN Regulations Nos. 155 and 156 is provided in the article "Cybersecurity in the automotive industry: ensuring compliance with UNECE regulations".

© 2026 AO KASPERSKY

vehicle. That frequency is expected to increase to 12–14 updates per year in the near future.

Why are over-the-air updates needed? A substantial part of the features in a modern vehicle, from user interactions via smartphone to controlling transmission modes, relies on software. Software code can contain errors, and over-the-air updates make it possible to promptly deliver patches to all vehicles connected to an update platform. In addition, software developers can refine existing features, introduce new capabilities, or connect the car to new cloud-based digital services while the vehicle is already in operation.

Large-scale computerization across the automotive industry creates fertile ground for an increase in attacks targeting individual vehicles, fleets, and roadside infrastructure. Direct attacks on vehicles are not yet widespread and are predominantly confined to isolated incidents of vehicle theft or non-standard modifications of on-board software commissioned by current owners. Meanwhile, attacks on automakers' IT infrastructure most often result in leaks of confidential data and financial losses caused by service disruptions. Nonetheless, cybersecurity researchers regularly discover vulnerabilities in on-board vehicle software and the backend infrastructure of in-vehicle cloud services[2] that could be exploited in destructive attacks, including those carried out remotely. Such attacks could target vehicle fleets connected to the same service or fitted with identical aftermarket equipment, as well as vehicles of a specific make and model sharing a common architecture and an identical factory-installed electronic unit.

According to our assessment, probable attack vectors against vehicles may involve compromising the infrastructure of an automaker, its technology partners, or digital service providers, as well as exploiting vulnerabilities in factory-installed electronic control units and aftermarket electronic equipment, including devices installed to comply with national regulatory requirements.

This article consists of two parts.

The first part presents examples of attacks on automotive manufacturers' systems, as well as examples of research reports on vulnerabilities identified in

[2] The backend infrastructure of in-vehicle cloud services is generally a cloud solution hosting application, data, and update servers. Infrastructure services may be deployed either within the automaker's environment or on third-party platforms. Some services may be maintained by taxi fleets (telemetry processing), vehicle service stations (keeping electronic service logs), charging station networks (loyalty program support), etc.

© 2026 AO KASPERSKY

vehicles and their supporting infrastructure. This part is intended for a broad audience interested in modern automotive cybersecurity issues.

The second part describes architectural approaches to addressing critical vehicle security challenges: ensuring security across ECU software update campaigns, preventing update package tampering, and minimizing potential damage to connected vehicles if an update platform is compromised. This section will primarily be of interest to developers, update platform architects, automotive software vendors, and suppliers of software components and digital services for vehicle supporting infrastructure.

How threats to automotive software are evolving

Attacks on vehicle manufacturers' infrastructure

Several major incidents took place in the second half of 2025, some of which hit the transportation and logistics sector. In September of 2025, automotive manufacturer Jaguar Land Rover (JLR) reported a ransomware attack that disrupted the company's operations and affected its suppliers worldwide. As a result of the attack, JLR was forced to take several systems offline, including those used in its manufacturing plants. According to media reports, several JLR suppliers faced the threat of bankruptcy due to production shutdowns that lasted several weeks.

JLR representatives also said that some corporate data was compromised as a result of the threat actors infiltrating the infrastructure. The attackers published screenshots of JLR's internal systems and vehicle technical documentation.

Between September 29 and October 9, JLR addressed the attack's consequences and resumed production in stages. Direct losses from the attack on JLR's infrastructure amounted to tens of millions of dollars. The company also secured additional commercial and government-backed loans totaling $4.69 billion to cover losses stemming from production shutdowns and supply chain disruptions. According to estimates by a British incident monitoring center, the attack on JLR affected about 5,000 organizations, causing total losses to the UK economy amounting to $2.5 billion.

Publicly reported incidents involving breaches of automakers' infrastructure and that of their suppliers have, so far, not included attacks propagating either

© 2026 AO KASPERSKY

to connected vehicle fleets or to supporting automotive infrastructure via vehicle compromise.

Vulnerabilities in digital services for vehicles

Researchers regularly identify vulnerabilities in the digital services modern vehicles connect to. For instance, in November 2024, cybersecurity researcher Sam Curry discovered vulnerabilities in the Subaru STARLINK administrator portal, a platform providing smartphone integration, navigation, remote access, and Subaru digital services.

During his research, Sam Curry reset a company employee's account password and obtained platform administrator privileges. The admin panel provided access not only to VINs, license plate numbers, and vehicle location data across the United States, Canada, and Japan, but also to owners' personally identifiable information (PII): full names, addresses, phone numbers, and email addresses. With an owner's personal data, VIN, or license plate number, a platform administrator could add new authorized users with remote vehicle control permissions. Meanwhile, the actual vehicle owner received no notification regarding the addition of new users or remote commands issued on their behalf. To demonstrate the vulnerability, Sam Curry unlocked an acquaintance's vehicle using only its license plate number. According to the researcher, threat actors could similarly start or shut off the engine, as well as access owners' personal data and vehicle information (such as location history from the past year, mileage, previous owners, sales history, etc.).

Digital services for modern vehicles can be provided not only by vehicle manufacturers themselves, but also by third-party companies. Earlier, in January 2023, Sam Curry showed that vehicles could also be attacked through third-party vendors' services used by large commercial fleets. He identified vulnerabilities in the infrastructure of Spireon, a fleet management and telematics provider. By exploiting a basic SQL injection vulnerability, Sam gained access to the service management console and obtained administrative privileges. This gave him potential control over 15.5 million vehicles, including police and ambulance fleets across several US states. He gained the ability to send control commands to vehicles, track their real-time location, and reflash the firmware of installed Spireon telematics hardware.

Reports by information security experts highlight the severe risk posed to millions of connected vehicles. The highest risk is that of unauthorized access to fleet remote management services, including corporate fleets, carsharing

© 2026 AO KASPERSKY

platforms, taxi services, public transit, road maintenance services, police, and ambulance fleets.

Some studies, typically confidential and rarely made public, mention the possibility of compromising safety-critical vehicle functions, potentially enabling attackers to remotely hijack vehicle controls. This could lead to large-scale traffic incidents and severe gridlock on public roads.

Vulnerabilities in aftermarket electronic equipment

Another attack vector targeting vehicles involves aftermarket electronic equipment installed either by fleet operators themselves or by vendors, e. g., to comply with national regulatory requirements. Such systems – designed, in particular, to monitor the condition of commercial vehicle drivers – are often installed without following information security requirements. Consequently, a threat actor could gain access to a channel that provides remote control of critical vehicle systems.

Research assessing the security of aftermarket equipment accessible via wireless communication channels reveals many critical vulnerabilities and architectural flaws that threat actors can exploit to seize control of vehicles.

In February 2024, researchers from Colorado State University published a report on vulnerabilities in the firmware of Electronic Logging Devices (ELDs). These devices are installed in commercial vehicles across the United States, Canada, and European Union countries to automatically record the time drivers spend behind the wheel and collect engine operation and vehicle-movement telemetry required for regulatory compliance in these countries.

The device analyzed in the study featured a comparatively simple architecture: it was connected to the in-vehicle network via the on-board diagnostic port and monitored CAN bus traffic. Additionally, it was equipped with a Global Navigation Satellite System (GNSS) module and Wi-Fi and Bluetooth wireless interfaces for connecting to user applications and mobile devices. The unit used predictable service set identifiers (SSIDs) and default passwords, while also exposing an API that allowed device configuration and over-the-air firmware updates.

The researchers re-flashed the ELD unit with modified firmware over the air, altering the engine's air-fuel mixture parameters. As a result, the test truck slowed down. Furthermore, the researchers succeeded in propagating a worm-type malicious program from the infected unit to a truck parked nearby (they called the program a "Truck-to-Truck Worm"). The second truck was located 12

© 2026 AO KASPERSKY

parking spaces away from the compromised ELD unit—approximately 36 meters. Infecting the target took less than 30 seconds.

In June 2024, Kaspersky ICS CERT published a security analysis of Telit Cinterion modems. Such modems are integrated, among other applications, into ERA-GLONASS emergency response units. The analysis uncovered seven vulnerabilities that require local access to exploit, and one that can be exploited remotely. Together, these vulnerabilities allow an attacker to fully compromise the modem, execute arbitrary code, and pivot the attack to other vehicle ECUs.

From infrastructure attacks to attacks on connected fleets

Attacking ECUs requires a sufficiently high level of technical expertise and preparation from a threat actor: such attacks are considerably more difficult to carry out than attacks on the infrastructure of in-vehicle cloud services. ECU attacks are technically sophisticated, requiring reverse-engineering skills, expertise in identifying and exploiting vulnerabilities in low-level vehicle components, and the ability to bypass built-in security mechanisms.

While low-skilled threat actors may use off-the-shelf exploits – predominantly for unauthorized software modification or targeted vehicle theft – developing their own exploits is typically beyond their capabilities.

The most common attacks today are those targeting the infrastructure of automakers, their suppliers, or third-party services used by vehicle owners. Such attacks result in leaks of confidential vehicle manufacturer and owner data, cause manufacturing facility downtime, and create other operational losses.

Nevertheless, we believe that in the future, threat actors will not restrict themselves to compromising servers and workstations within enterprise infrastructure. Researchers regularly find vulnerabilities in digital services and in electronic control units connected to these services. Attackers can leverage these vulnerabilities to discover and execute attack vectors targeting vehicle fleets already connected to these services.

Update security challenges

Fixing vulnerabilities via over-the-air updates, as well as correcting flaws in the implementation of vehicle functions, is posited as a major advantage of modern software-defined vehicles. In practice, however, automotive manufacturers

© 2026 AO KASPERSKY

must address several non-trivial challenges to update vehicle ECU software securely.

To make maintaining software modules easier, developers compile specialized registries (for example, in the form of SBOMs – Software Bills of Materials) that track all software components and describe dependencies between them. Any vehicle update can change the set of software components and their corresponding dependencies. Consequently, an automaker must not only compile such registries for vehicle models in production, but also maintain and update them whenever updates are released throughout the vehicle's lifecycle. Without maintaining these registries and analyzing applied updates, it is difficult to imagine fulfilling the requirements of UN Regulation No. 155[3], discussed in the article "Cybersecurity in the automotive industry: ensuring compliance with UNECE regulations". Under these regulations, a manufacturer must inform the approval authority of any vehicle type modification that affects its technical characteristics with respect to cybersecurity.

Beyond legitimate updates from the vehicle manufacturer, vehicle owners or service facility technicians occasionally attempt to bypass manufacturer security controls to independently install uncertified electronic equipment or re-flash a control unit with modified firmware. Some automakers' efforts to prevent such tampering are not merely about safeguarding business models that provide subscription-based access to vehicle features. Third-party firmware may contain vulnerable code, cause premature wear of mechanical components, and render the vehicle's behavior on the road unsafe. Automakers seek to avert consequences such as control unit malfunctions, warranty claims, regulatory fines, and the revocation of Vehicle Type Approval (VTA) certificates; however, detecting such modifications across an entire fleet of connected vehicles can be difficult.

Today, there are technologies for over-the-air update platforms that can handle automotive software configuration management and detect unauthorized or modified software installed in vehicles.

In a typical vehicle architecture, one or more control units are designated to manage communication with the update platform: they download installation packages, verify their authenticity, and transfer them to the unit being updated. In some publications, units interacting with the update platform are termed FOTA (Firmware Over-the-Air) Master, while in others they are designated

[3] UN Regulation No. 155: "Uniform provisions concerning the approval of vehicles with regard to cyber security and cyber security management system."

© 2026 AO KASPERSKY

Primary. All units updated via FOTA procedures mediated by such dedicated units are called FOTA Target or Secondary. In this article, we use the FOTA Master – FOTA Target terminology to denote unit roles. The FOTA Master – FOTA Target update procedure is illustrated in Fig. 2.

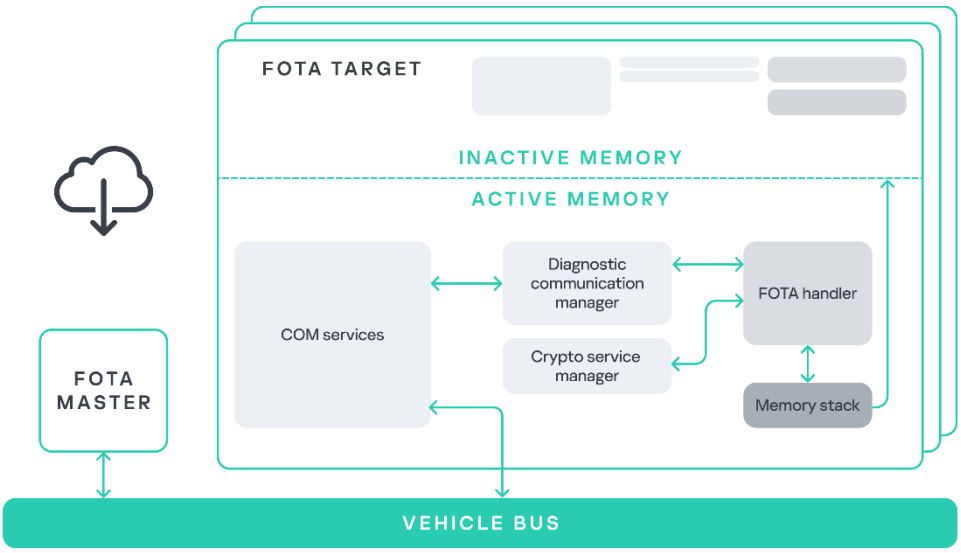


Fig. 2. FOTA Master – FOTA Target update diagram

There are at least two reasons to use a FOTA Master:

1. **Ensuring cybersecurity.** Without a FOTA Master, each unit would have to interact with the update platform over a wireless channel independently, substantially expanding the vehicle's attack surface. However, the FOTA Master is a prime entry point for remote attacks and is therefore subject to more stringent cybersecurity requirements.
2. **Optimizing protection costs.** Engine and transmission control units are subject to stringent response-time requirements. For instance, an engine control unit must process crankshaft position sensor signals and, instantaneously, within a millisecond, issue commands to open or close valves. Any latency or desynchronization in control commands from the unit can degrade engine performance, reduce its service life, or cause it to break down. Therefore, given the nature of the tasks these units perform, integrating security features directly into these units is either impossible or substantially increases their cost.

© 2026 AO KASPERSKY

Protecting the update process at the vehicle level

The update process is protected, and the integrity and authenticity of updates are ensured through security measures deployed across various levels of the vehicle architecture: the in-vehicle network level, the FOTA Master, and the FOTA Target levels. The security mechanisms are selected and implemented based on cybersecurity threat and risk assessments for each of these tiers.

In this article, we present only protection measures for the FOTA Master as the key unit supporting update processes. The following threats are relevant to the FOTA Master:

- Interception of the connection between the vehicle and the update platform with modification of the transmitted data;
- Overloading of the FOTA Master and disruption of its availability (denial of service);
- Rollback to outdated and vulnerable software versions;
- Software retrieval from rogue update servers;
- Switching the unit to debug mode with elevated privileges;
- Exploitation of bootloader vulnerabilities to replace executable binaries and execute arbitrary code;
- Attacks on the unit's configuration parameters (connection and operating parameters);
- Compromise of cryptographic keys, subversion of the root of trust, and attacks on cryptographic operations.

Security measures

Root of trust

Protection of the FOTA Master must be built upon a root of trust (Fig. 3) that provides security services, i.e., a set of basic, immutable security functions. These services reinforce the vehicle electronic control unit protective measures, which are described in the following subsections. The root of trust is implemented using hardware-based security modules – Trusted Platform Module (TPM), or Hardware Security Module (HSM), which can be either an integrated component of the FOTA Master or a standalone module.

© 2026 AO KASPERSKY

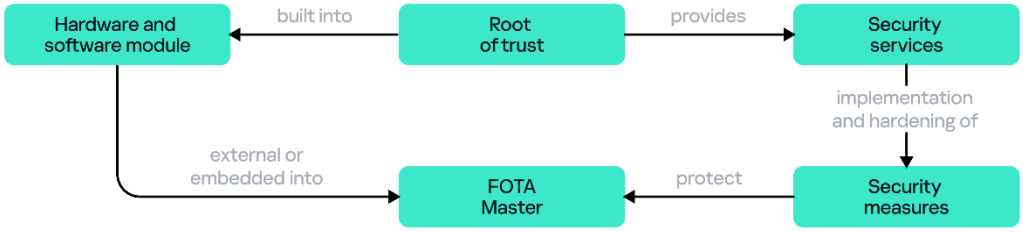


Fig. 3. Root of trust and FOTA Master security measures

Cryptographic keys are stored in a protected memory area and are never transferred outside the hardware-based module. All cryptographic operations are executed inside its isolated hardware-based environment, preventing software from reading the keys. Protection of the module against side-channel attacks prevents attackers from capturing intermediate computation results or analyzing power consumption to derive encryption keys.

Unique device keys (endorsement keys) are generated at the production stage and stored in the security module, cryptographically binding a specific firmware instance to a specific physical module and making it impossible to create an exact copy of the module, i.e., to clone it. Consequently, firmware extracted from a module in one vehicle will not work on an identical module in another vehicle.

A hardware-based implementation of the root of trust protects the FOTA Master against several types of attacks, including attacks on firmware integrity and attempts to extract data from memory.

Securing connections with backend servers

Encrypting wireless traffic using the TLS cryptographic protocol is a baseline measure that guarantees the confidentiality and integrity of commands and files delivered to the FOTA Master from update platform servers.

Mandatory authentication of platform servers by the FOTA Master using X.509 certificates enables unambiguous verification of data sources and prevents man-in-the-middle attacks aimed at spoofing commands and update repository servers.

Additional communication channel protection measures include a private APN and the corresponding security policies that isolate data communication between the vehicle and the update platform from the communication service provider's public infrastructure. APN policies can block connection attempts

© 2026 AO KASPERSKY

originating from atypical geographic locations or routed through unencrypted base stations before data transmission begins.

Hardware isolation and trusted execution environment

Critical operations, such as decrypting and verifying the authenticity of update binaries, must be performed within a Trusted Execution Environment (TEE). For example, telemetry processing runs on general-purpose computing cores, while the authentication of downloaded update packages is performed by a secure coprocessor.

Physical isolation of operations reduces the risk of cryptographic secret leakage if higher-level ECU software is compromised.

Microcontroller memory and peripheral isolation[4] must be enforced via hardware mechanisms, such as Memory Protection Units (MPUs), which regulate transactions across isolated security domains.

Communication between the FOTA Master and the in-vehicle network should be handled by a dedicated microcontroller. This microcontroller inspects and handles ECU messages traversing internal data buses, exposing an interface (API) with a strictly defined list of available commands to other FOTA Master components. The compact firmware of such a microcontroller can be certified to rigorous automotive safety and security standards.

Comprehensive isolation measures counter privilege escalation attempts, block lateral movement across security domains, and prevent unauthorized tampering with vehicle update mechanisms.

Protection against physical tampering and side-channel attacks

Debugging interfaces on the FOTA Master, such as JTAG or SWD, must be disabled or require authentication in every client connection attempt. Restricting access via the debugging interface blocks attempts to extract cryptographic keys and firmware images from unit memory.

Attackers may attempt to bypass FOTA Master security features by subjecting the microcontroller to laser or electromagnetic interference. The following

[4] General-Purpose Input/Output (GPIO) ports; communication interfaces including UART and SPI; hardware timers and counters; analog-to-digital (ADC) and digital-to-analog (DAC) converters, etc.

© 2026 AO KASPERSKY

hardware and software countermeasures help mitigate these fault-injection techniques.

1. Hardware countermeasures:
   - Core redundancy (lockstep cores);
   - Glitch detectors;
   - Hardware memory checking (ECC and parity checks);
   - Protective shielding over the chip;
   - Internal clock randomization (clock scrambling and jitter injection),
2. Software countermeasures:
   - Control Flow Integrity (CFI);
   - Conditional branch duplication (double-checking);
   - Multi-bit state flags;
   - Execution timing obfuscation (software jitter and dummy cycles);
   - Using watchdog timers.

FOTA Master hardware security modules must be hardened against physical tampering and side-channel attacks. Tamper-detection sensors responsive to enclosure breach, temperature fluctuations, and voltage spikes automatically initiate zeroization of secret keys or trigger permanent microcontroller shutdown. Secure module memory featuring continuous on-the-fly encryption and integrity checks prevents unauthorized read operations and arbitrary code execution.

Hardware support for unit lifecycle management

Hardware-enforced separation of ECU lifecycle stages eliminates the possibility of reverting a unit to development or debug mode, as well as unauthorized reuse of a device.

A unit's irreversible transitions between lifecycle phases are achieved using one-time programmable fuses (eFuses). Blowing debug interface fuses during production permanently cuts off access to unit calibration and debugging modes.

Lifecycle transition logic can be implemented via immutable finite-state machines, as specified, for instance, in EVITA Full HSM specifications. A lifecycle stage is encoded not by a mutable variable in memory, but by a hardware logic circuit comprising transistors and flip-flops on the microcontroller die. Based on the state of immutable fuses, this finite state machine determines which memory partitions and peripherals are accessible.

© 2026 AO KASPERSKY

As a result, a threat actor cannot register a decommissioned or stolen unit on the update platform under the guise of a legitimate node. During an authentication attempt, the platform server detects a mismatch in the unit's cryptographic signature derived from immutable fuse states and rejects the session.

Upon decommissioning, the security module must execute a hardware sanitization routine. Master keys, digital certificates, and session tokens stored in secure non-volatile memory must be overwritten with random sequences.

Fault tolerance and critical data redundancy

Attackers may intentionally induce hardware glitches or component failures to corrupt or forcibly manipulate encryption keys, unique identifiers, update metadata, configuration files, etc.

To prevent unauthorized modification of such data, duplicating it across physically and logically isolated regions of secure non-volatile memory is recommended. During read and write operations, data integrity is verified using checksums or cryptographic hashes. If data corruption is identified, an automated recovery routine restores corrupted records from the backup copy maintained within an isolated region.

In volatile and non-volatile memory, hardware-based error detection and correction codes (EDC/ECC) are implemented to mitigate operational faults and targeted fault-injection attempts directed at the control unit.

The control unit must support atomic write transactions and operation journaling. Any changes are first recorded to a journal in non-volatile memory and then applied to system RAM and primary microcontroller ROM. In the event of a fault or power disruption, a recovery mechanism reads the journal from the last consistent checkpoint and executes an appropriate recovery routine: rolling back to the initial baseline or reapplying pending changes.

Memory management

A Memory Management Unit (MMU) leverages hardware- and software-enforced memory isolation to partition processes, ensuring that they execute only within their designated memory regions.

Certain MMUs can also restrict memory access for coprocessors and peripheral devices. For example, a multimedia system GPU can be restricted from accessing memory zones allocated to steering system communications. Memory areas receiving data from external networks (such as LTE and Wi-Fi) are allocated exclusively for data processing.

© 2026 AO KASPERSKY

Even if an attacker succeeds in injecting instructions into a data buffer or the call stack, execution prevention bits configured on corresponding memory pages (NX or XD bits) guarantee that the ECU processor refuses to execute code from those data segments.

Additional hardening mechanisms can include stack canaries and Address Space Layout Randomization (ASLR).

This approach to memory management mitigates attack vectors associated with buffer overflows, unauthorized data reads, and arbitrary command execution beyond authorized memory boundaries.

To counter denial-of-service and resource exhaustion attacks against the control unit, compute and memory quotas are assigned to each process. For example, time-triggered scheduling prevents a single run-away process from monopolizing core resources, reserving deterministic time slots for safety-critical tasks such as processing automated emergency braking system signals.

Configuration hardening

Strong hashing algorithms, such as SHA-256, or checksums are used to prevent unauthorized tampering with unit configuration parameters. The configuration is stored in a protected region of the unit's memory.

Enforcing authentication for configuration-change commands sent from the cloud infrastructure to the connected vehicle minimizes the risk of their unauthorized modification or spoofing.

These measures protect against unauthorized modification of unit parameters.

Logging

Logs must record not only successful installation events, but also intermediate states, explicitly capturing version identifiers of downloaded files and the results of digital signature verification. Suspicious patterns, such as repeated failed authentication attempts or recurring access denials, help detect brute-force attacks and other attack traces. Real-time anomaly detection on board enables prompt execution of defensive playbooks, such as interface lockdowns or isolation of compromised components.

When gaining an initial foothold in the system, threat actors frequently attempt to eliminate evidence by clearing or modifying system log files. To mitigate this threat, event logs must reside in a protected memory area, with their records cryptographically chained.

© 2026 AO KASPERSKY

Authenticity verification and replay attack protection

Over-the-air update download sources must be restricted to trusted update platform repositories[5]. Digital signature mechanisms or message authentication codes (MAC) applied to update platform transmissions mitigate the risks of malicious command injection and vehicle hijacking. Digitally signing transmitted update packages minimize the probability of tampering with or spoofing update packages sent to a connected vehicle. Packages are verified using the public key of the software developer or automaker, stored within a protected hardware module (HSM or TPM).

To counter replay attacks, timestamps based on the TSP cryptographic protocol are embedded into incoming traffic from the update platform. By verifying the timestamp, the vehicle can confirm that specific commands and update binaries were generated and dispatched at the time specified. If discrepancies are detected, the control unit rejects incoming traffic originating from the update platform.

An additional protective measure is anti-rollback protection prohibiting system downgrades to deprecated versions. The system finds the version number within the signed metadata of the installation package and verifies that it is strictly greater than the currently deployed version. This prevents threat actors from rolling back the system to an authentic, but deprecated and vulnerable, firmware release.

Secure boot

Secure boot protects against exploiting vulnerabilities in low-level software that allow tampering with the bootloader and executable binaries, as well as bypassing security mechanisms before executing these files.

The process begins with the execution of verified, immutable code stored in a hardware-based security module. Storing this code in one-time programmable read-only memory (ROM/OTP) prevents a threat actor from modifying or substituting it to circumvent boot-stage checks or subvert the root of trust. The integrity and authenticity of bootloader code must be verified using the manufacturer's public key. The control unit must support public key revocation and update procedures.

This immutable code sequentially verifies each subsequent software component before executing it and transferring control to it. If a mismatch is

[5] The architecture and security measures used on the update platform are outside the scope of this article.

© 2026 AO KASPERSKY

found at any stage of this chain of trust, the bootloader blocks execution of the unverified code.

During an update, attackers may artificially trigger a critical error that sends the system into an infinite boot loop or a vulnerable state. In that case, the bootloader triggers a safe rollback mechanism. It must detect the abnormal boot sequence, isolate corrupted files, and restore system state based on a verified baseline software image maintained in a protected memory area.

RxSWIN identifiers

The security of the update workflow on the vehicle side is enforced using the technical measures discussed above. However, their deployment alone cannot guarantee the functional — and consequently legal — compliance of an individual vehicle with the Vehicle Type Approval under which that vehicle model is authorized for sale and operation.

VTA certification is granted strictly for designated ECU software versions and configurations. Therefore, if a scheduled software update materially alters the functionality of control units, recertification will be required to obtain a new VTA.

Identification of software modules

Until recently, automakers took different approaches toward automotive software modifications: while some, like BMW, declared technological war on unofficial firmware, others adopted a more lenient stance.

Currently, the United Nations Economic Commission for Europe requires automakers to maintain rigorous control over precisely which update packages are dispatched and how they are deployed on vehicles.

UN Regulation No. 156[6] requires that automakers assign a special RxSWIN identifier to each software version whose deployment triggers type-approval certification for affected vehicle types[7].

[6] UN Regulation No. 156: "Uniform provisions concerning the approval of vehicles with regard to software updates and software update management system."

[7] Details regarding Whole Vehicle Type Approval certificates can be found in the article "Cybersecurity in the automotive industry: ensuring compliance with UNECE regulations".

© 2026 AO KASPERSKY

The identifier consists of two parts: Rx (derived from UN Regulation) designates the specific UN Regulation with which the automotive software must comply, while SWIN (Software Identification) indicates the specific version of that software.

The introduction of RxSWIN accomplishes several objectives simultaneously:

- Directly links a software version to its corresponding VTA certificate;
- Provides centralized auditing of control unit software installation and configuration across the entire vehicle;
- Enables the detection of unauthorized ECU tuning ("gray" chip tuning);
- Establishes an evidence base for traffic accident investigations and cyberattacks on vehicles.

UN Regulation No. 156 delegates the responsibility of defining the RxSWIN format to vehicle manufacturers. Automakers are obligated to define and assign RxSWINs such that they meet the following criteria:

- Unambiguously identify software versions;
- Are guaranteed to update with every software update;
- Can be read using standard electronic communication interfaces, for example, at service stations using dedicated diagnostic tools connected to the OBD port.

Additionally, an automaker must provide all information pertaining to RxSWIN identifiers to the competent vehicle type approval authorities and keep that information up to date.

UN Regulation No. 156 does not provide detailed instructions on constructing RxSWIN identifiers. Automotive engineers are currently working on aligning the RxSWIN format with the requirements of UN Regulation No. 156.

For example, Japanese engineers proposed the following approach to structuring RxSWIN (detailed in the "Report of Japanese Test Phase (Software Update)"). Every function of an electronic control unit subject to certification under a UN Regulation is assigned a unique identifier. The RxSWIN includes information on the vehicle model (type) and a system configuration identifier – SysID.

© 2026 AO KASPERSKY

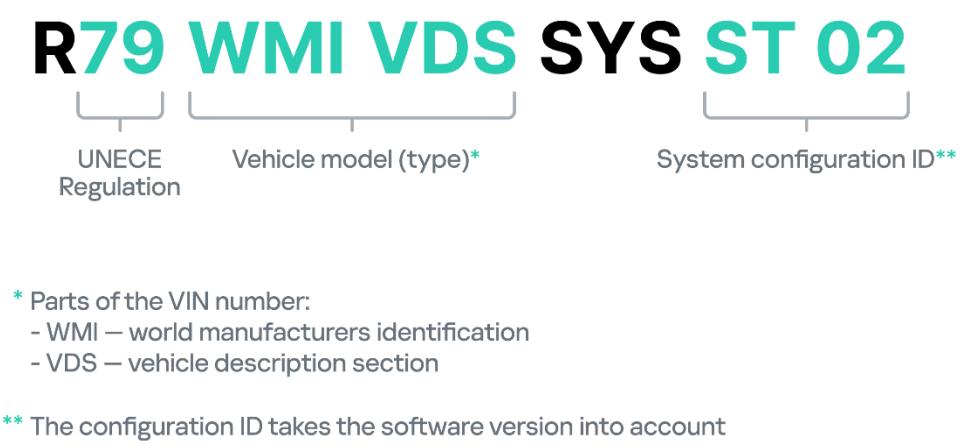


Fig. 4. Example of an RxSWIN structure

Before releasing a new software version, developers must determine whether it will require system recertification. If changes to the VTA are required, developers enter a new SysID, consisting of the system name and the corresponding software version, into the configuration management system.

Now let us examine how the corresponding RxSWIN is updated (Fig. 5). Consider a hypothetical vehicle model that supports software updates for the in-vehicle infotainment (IVI) unit, its anti-lock braking system (ABS), electronic stability program (ESP), and interior ambient lighting units. UN Regulation No. 13-H governs braking systems, UN Regulation No. 39 governs speedometers, and UN Regulation No. 79 governs steering equipment. A single system can fall within the scope of multiple UN Regulations or, as in the case of ambient lighting, remain entirely outside regulatory scope. Consequently, ambient lighting is not subject to certification, and lighting control software does not require an RxSWIN assignment.

© 2026 AO KASPERSKY

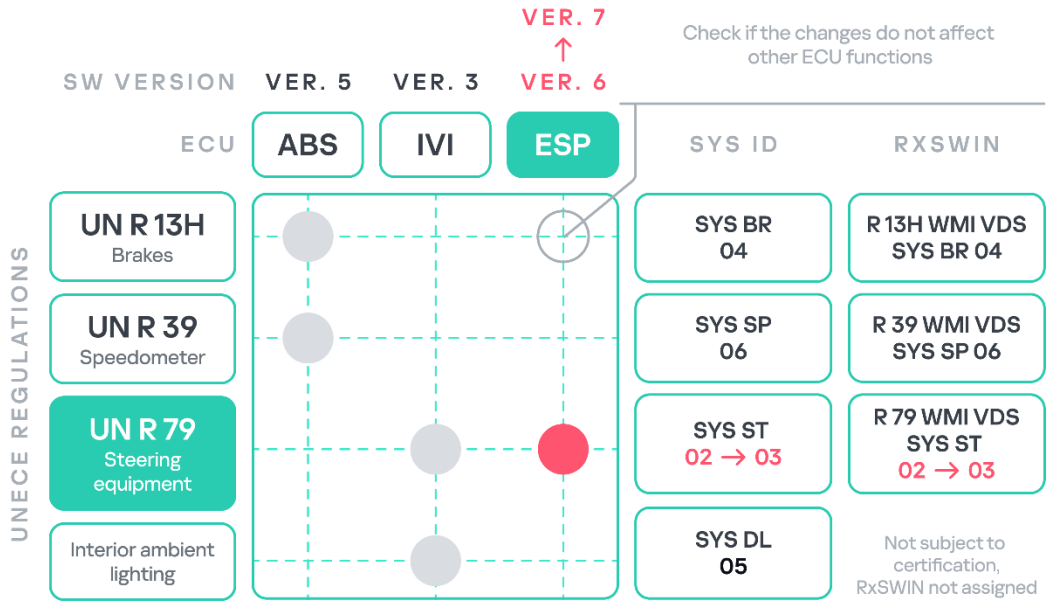


Fig. 5. Software update and update of the corresponding RxSWIN

When releasing a new software version for ESP that introduces new steering control features, the following steps must be performed:

1. Assign a new version identifier to the software;
2. Update the SysID based on the new software version;
3. Determine whether the new software version affects other vehicle functions and whether recertification of other systems is required;
4. Complete system recertification and assign a new RxSWIN;
5. If related vehicle systems are affected, recertify them in accordance with applicable UN Regulations.

To meet regulatory requirements, automakers cannot rely solely on an up-to-date Software Bill of Materials (SBOM) listing software components and their dependencies. An RxSWIN is different from a standard software build or version number because it is required only when software is subject to certification under a specific UN Regulation. Furthermore, the RxSWIN of a new software version must explicitly indicate whether updating to that version will necessitate recertification or an extension of the existing vehicle type approval.

Moreover, UN Regulation No. 156 makes the task more difficult for automakers by requiring them to guarantee protection against unauthorized changes not only to update installation packages, but also to the corresponding RxSWIN

© 2026 AO KASPERSKY

identifiers. In addition, to obtain a certificate of compliance with UN Regulation No. 156 – that is, vehicle type approval – automakers must submit an evaluation report documenting the protective measures implemented for the vehicle. These measures must span vehicle electronic control units and on-board software, as well as the update platform.

Different automakers may use different RxSWIN formats, potentially giving rise to global identifier uniqueness issues. The automotive industry currently lacks a unified conflict-resolution mechanism to arbitrate instances where different automakers declare identical RxSWIN identifiers for different software packages or, conversely, assign different RxSWINs to the same software packages. The RxSWIN synchronization and harmonization challenge points to establishing an industry-wide central RxSWIN registry; however, that issue is outside the scope of this article.

Managing the RxSWIN registry for an individual vehicle

Paragraph 7.1.2.3 of UN Regulation No. 156 requires that, automakers should maintain an RxSWIN registry containing information on software versions mapped to RxSWIN identifiers, along with the associated integrity verification data. At the same time, Paragraph 7.2.1.2.2 permits storing this registry either on board the vehicle or in the automaker's infrastructure.

Automakers may feel that storing RxSWIN registries on the update platform is the most practical approach, because it requires minimal changes to vehicle architecture. Under this approach, the FOTA Master must download update packages and verify their integrity. Additionally, the FOTA Master must report the firmware versions of updated units back to the update platform. The primary burden of campaign planning and identifying update targets is handled by the platform, where outdated configuration data might reside at the time of campaign start. Therefore, backend update servers must continuously synchronize and update information on the connected vehicle fleet.

Let us now examine two opposing approaches to storing RxSWIN identifiers directly on board the vehicle:

1. RxSWINs are stored individually on each FOTA Target unit that receives updates from the FOTA Master over the in-vehicle network.
2. The RxSWIN registry is stored on one or more dedicated units (FOTA Masters) equipped with the necessary security mechanisms and responsible for communicating with the update platform.

© 2026 AO KASPERSKY

RxSWIN identifiers are stored on FOTA Target

Storing RxSWIN identifiers in the firmware of each individual unit, much like storing them on the update platform, requires minimal modifications to vehicle architecture. This model aligns closely with defense-in-depth principles: compromising any adjacent unit, including the FOTA Master, and gaining access to the in-vehicle network does not automatically enable a threat actor to compromise the target unit by modifying its firmware and RxSWIN identifier.

To prevent unit compromise, an automaker must implement more stringent security measures for each ECU. Units must support secure communication protocols and cryptographic mechanisms that provide update package authenticity verification and protect RxSWIN identifiers from spoofing or tampering.

Under this approach, FOTA Target units become more sophisticated and expensive, the range of suitable hardware components narrows, and the automaker must maintain strict oversight of how rigorously its suppliers adhere to cybersecurity requirements for the units they supply. The FOTA Master's role is limited to interfacing with the platform and downloading updates for the vehicle's target units.

In this scenario, an RxSWIN registry does not need to reside permanently on board the vehicle; rather, it is generated dynamically—for example, at the beginning of a vehicle update campaign by polling the target vehicle and storing the resulting identifier list on the cloud update platform.

RxSWIN registry stored on the FOTA Master

When the RxSWIN registry is maintained on one or several FOTA Master units, the FOTA Master must verify that unit firmware identifiers match the RxSWIN registry and notify the update platform of any discrepancies. On the one hand, the vehicle manufacturer can impose less stringent functional requirements on target units being updated. On the other hand, these units must still support authenticity and integrity verification for updates delivered by the FOTA Master over the in-vehicle network.

The FOTA Master must not only monitor the configuration of target units, but also provide secure storage for downloading binaries from the update platform and keeping firmware backup copies for units being updated. In the RxSWIN registry, the FOTA Master aggregates ECU data (unit identifier, unit type, firmware hashes/signatures, the corresponding RxSWIN identifiers, etc.).

© 2026 AO KASPERSKY

Some security functions – such as cryptographic key management, RxSWIN registry integrity protection, update authenticity and integrity checks, and secure boot – require equipping the FOTA Master with a hardware security module (such as a TPM).

More stringent security requirements apply to the FOTA Master because, if it is compromised, a threat actor can pivot the attack to any of the vehicle control units it manages.

Secure-by-design control units

When defining and implementing requirements for electronic control units, automakers must account for at least the following factors, each of which can serve as an entry point into modern vehicles:

- Architectural flaws and implementation errors in units, such as those associated with access to their functions, configuration, in-vehicle network communications, and wireless channels;
- Vulnerabilities in the implementation of cryptographic algorithms and cryptographic key management;
- Vulnerabilities in open-source components, third-party components, and out-of-the-box security components;
- The human factor, manifested in the inclination to prioritize convenience over established security policies and security guidelines.

When designing a new vehicle model, it is unacceptable to develop security measures for individual components in isolation, without considering how components interact, including communication with other systems. Security requirements must be integrated in the design process from the earliest conceptual stages of development, including architectural decisions. The Principles and Approaches for Secure by Design Software is one of the foundational publications on secure by design principles and can be used to develop secure-by-design systems with maximum resilience to vulnerability exploitation during attacks by threat actors.

Conclusion

The challenge of securing remote automotive software updates is recognized by international bodies and industry organizations. More countries will likely seek to regulate OTA update processes through regulatory frameworks, harmonizing their national laws with UN Regulation No. 156.

© 2026 AO KASPERSKY

In this context, automotive manufacturers face several challenges because the number of software and hardware configurations is huge, while the requirements imposed on them are comprehensive and diverse. These include requirements governing the implementation of software update identifiers (RxSWIN). At the same time, each vehicle manufacturer has its own requirements for identification and version control. This can hinder consensus on adopting a unified RxSWIN format among stakeholders in the global automotive market.

Typically, the statutory service life of a vehicle, during which the safe operation of all components is guaranteed, is 10 years. In practice, however, vehicles often remain in service well beyond this period – for 15 or 20 years or more. This raises the question of how to provide updates and protection for a vehicle throughout its actual lifecycle.

Fortunately, or perhaps unfortunately for automakers, neither UN Regulations nor the complementary ISO 24089:2023 standard restrict vehicle manufacturers to a fixed set of implementation technologies. Consequently, when confronted with these and other challenges, an automaker may deploy only baseline security measures to achieve formal compliance with international or national regulations. However, under such an approach, the automaker risks constantly playing catch-up: in cybersecurity, patching holes as they emerge is far from the best strategy.

We believe that in the rapidly evolving field of over-the-air updates specifically, and automated transportation as a whole, it is essential to take a proactive stance. Software and software-defined systems across all production vehicle models should be developed with secure updating capabilities in mind. In a few years, minimalist security measures that formally comply with today's regulations may become inadequate. This means that design solutions that keep pace with technological developments are the way to go.

© 2026 AO KASPERSKY

Kaspersky Industrial Control Systems Cyber Emergency Response Team (Kaspersky ICS CERT) is a global Kaspersky project aimed at coordinating the efforts of automation system vendors, industrial facility owners and operators, and IT security researchers to protect industrial enterprises from cyberattacks. Kaspersky ICS CERT devotes its efforts primarily to identifying potential and existing threats that target industrial automation systems and the industrial internet of things.

Kaspersky ICS CERT ics-cert@kaspersky.com

© 2026 AO KASPERSKY