escrypt

SECURITY. TRUST. SUCCESS.

Whitepaper

Putting automotive security to the test

Security testing as acid test for connected vehicles

The increasing connectivity of modern vehicles brings new risks from a security standpoint. To safeguard connected systems against unauthorized access and attacks by cybercriminals, the new ISO/SAE 21434 standard “Road vehicles – Cybersecurity engineering” embodys a security-by-design-based approach. It provides for comprehensive testing at the component and system level. This will make security testing an acid test for security measures and an integral part of type approval. Performing and evaluating the tests requires specific know-how. This white paper presents the main test procedures, identifies typical findings and points of attack, and demonstrates the necessity of a holistic security strategy.

Contents

Security moves in a highly dynamic environment

Two major automotive standards were published in 2021. The first was in June: UNECE WP.29, which addresses optimized vehicle security and effective anti-theft protection. ISO/SAE 21434 "Road Vehicles – Cybersecurity Engineering" followed in August and defines processes and requirements for the development of secure vehicles. A major driving force behind both sets of standards is the increasing connectivity of modern vehicles: connectivity with the outside world enhances comfort and safety, but also offers more scope for hackers to target connected systems. The technical standards offer manufacturers guidelines for the systematic and effective safeguarding of vehicle systems against unauthorized access, attempted manipulation, and cyberattacks.

But how can they successfully minimize risks without breaking the timetables and financial constraints of vehicle development? And how can they test the effectiveness of security measures? Especially when some of the attack scenarios and strategies they are considering are not yet known at the time of vehicle development and may even lie far in the future. The lifecycles of modern vehicles can encompass two decades. A look back illustrates the scale of things: in the year 2000, only five percent of the global population had access to the internet, smartphones had yet to be invented, and connected vehicles were a pipe dream.

This makes it clear that the requirements of IT security change and develop in a highly dynamic fashion – and particularly when it comes to the automotive sector, which opens its vehicle networks up to infotainment, updates and upgrades over the air (OTA), and vehicle-to-X communication. This digital connectivity makes the protection of systems a formidable challenge. And not only because digitally connected criminals are able to access increasingly more powerful computers and continually proliferating new attack patterns, often online, but also because security requires the mastering of organizational challenges. For example, cryptographic techniques and algorithms can become outdated during the multi-year development phase and even more so during the subsequent lifecycle of the vehicle. Communication can also be tricky in development processes that often contain teams from different companies scattered across various international locations. If security requirements are not communicated well enough, then problems may arise during implementation and testing. Another consequence of communication difficulties is that third-party system components remain "black boxes" for the other market players. Sometimes they don't know what security measures are already implemented in the components and how robust these implementations are.

Testing becomes an acid test

Looking at the ISO/SAE 21434 standard, we see what methods it puts forward to attain cybersecurity engineering with long-term effectiveness. The standardization bodies have embraced security by design. In the future, security engineering is set to follow a V-model, just as ISO 26262 does for functional safety. The left-hand arm comprises the following tasks: defining the system requirements, elaborating the system design, and deriving the security specifications from the above. This is followed in the right-hand, climbing arm of the V primarily by comprehensive testing. It begins at the component level and continues with tests at the system level. Only once these tests have been evaluated and then the connected systems have been accepted, can and should they be operationally deployed.

As such, testing becomes the acid test. At the latest by the time the new standard is published, the type approval of new models will be fully contingent upon testing: following the methods set out there will be mandatory. At the same time, manufacturers can use the corresponding test documentation in the approval process as proof that they have followed the technical regulations and the security-by-design approach. Security testing provides evidence that their development process complies with the security regulations for type approval.

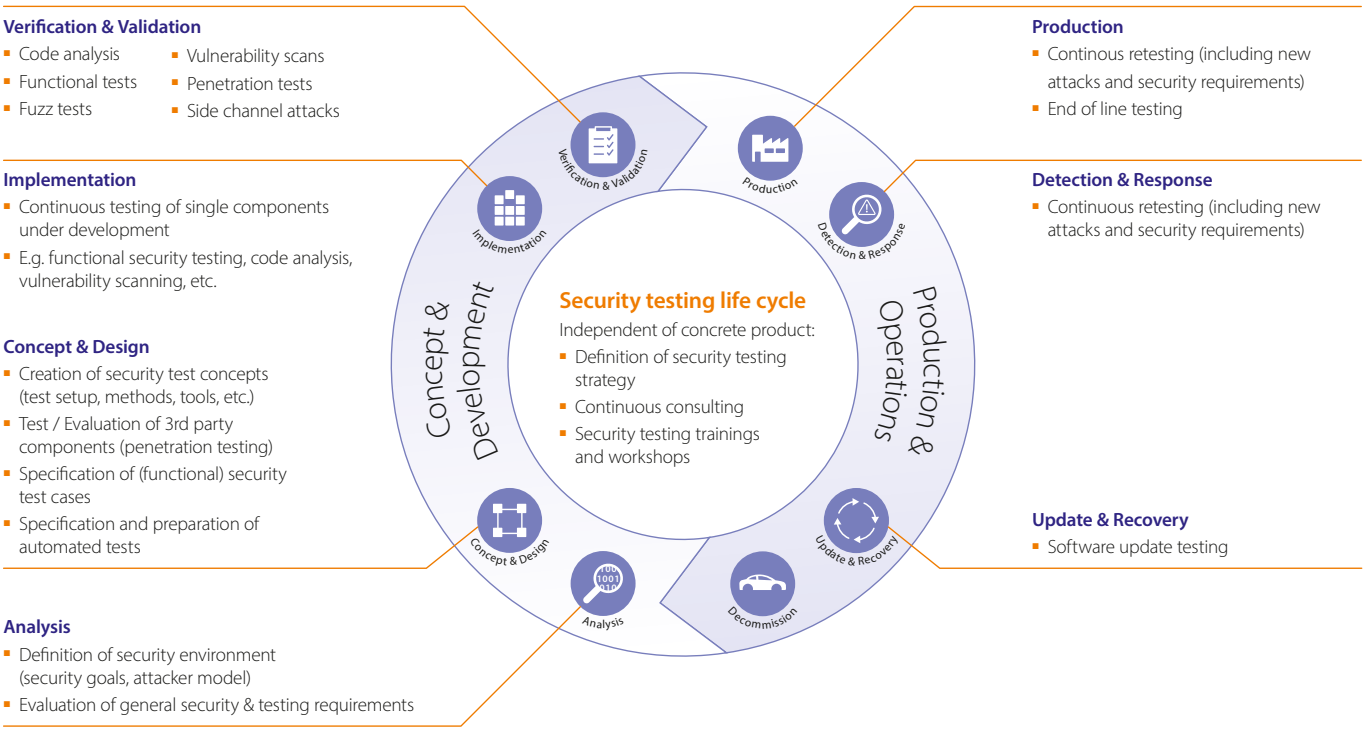

Figure 1: Security testing life cycle

Testing becomes a matter of organization

In view of this importance and the organizational challenges of security in distributed development processes, the concrete implementation of security testing throws up a series of urgent questions: What tests are needed at the component and system level? Which test procedures are suitable? What expertise is needed to plan, implement, and evaluate the tests? And how can a corresponding test strategy be meaningfully embedded in a holistic security concept that reaches beyond the start of production (SOP)?

When considering these questions, devising a well-founded test strategy is the best place to start. After all, it is a matter of identifying suitable test procedures and the respective scope of testing. Only then can manufacturers make an informed estimate about what infrastructure is needed for which tests, and when. Equally, they need to determine whether they have specialists with the appropriate security testing know-how available in-house. And because security testing doesn't end with SOP, the test strategy must take into account the entire vehicle lifecycle. It requires ongoing consideration of attack scenarios and strategies, which must result in protection measures that are effective in the long term.

Even a casual survey of the topics and tasks that underlie the test strategy makes clear that they require specific experience, strategic foresight, and well-founded analyses of existing human resources and technical infrastructure. OEMs and suppliers have to determine to what extent they can handle the testing in-house with their own experts and own infrastructure. And if they can't cover everything in-house, they need to determine in the course of this strategic preliminary work to what extent and for which specific security testing tasks they have to bring in external know-how. The underlying goal is to protect the vehicle and all connected components against unauthorized access and manipulation as efficiently as possible and for as long as necessary. Testing is designed to ensure that the protective measures undertaken actually protect the connected ECUs, vehicle computers, and other components in an effective manner – and that all known security gaps are closed in advance of SOP. To address security risks that arise after SOP, a holistic security concept is required, which we will discuss further below.

Security testing methods

Putting protective measures through their paces: penetration testing

Penetration testing, often called pentesting for short, is one of the most common and most effective methods for testing the effectiveness of protective measures and identifying undiscovered security gaps.

Pentesters deploy a broad arsenal of hacking strategies to simulate cyberattacks and unauthorized access to the system. As such, you could say they are hacking for a good cause. They discover weak points and possible implementation errors, which may be caused by faulty technical implementation, deviations from the security concept, or problematic interactions between system components or third-party components.

Fundamentally, such pentests can take the form of white box, gray box, or black box tests. To achieve maximum security, white box tests are the best option. In this case, testers have full knowledge of the internal system information, right down to the source code. In gray box tests, by contrast, this information is only partly known. In black box testing, the starting point of the experts is like that of real hackers, with access only to the system to be tested and publicly available information. Evidently, the less information the testers have, the longer the tests take, the less systematic the probing, and the less scope there is for automated hacking methods. As a rule of thumb, the deeper the insight a tester has into the system, the sooner they will be able to identify any security gaps and the more easily they will be able to obtain results within a constrained budget.

The pentest itself comprises a multi-stage process, beginning with the setup of the system, moving on to the analysis and exploration of potential points of attack, and culminating in the actual penetration attempts themselves. Identified weaknesses are discussed with the developers so that they can be fixed and the system re-tested if required. Based on the overall system development, a process has become established whereby pentesters typically start off by probing individual ECUs, network components, interfaces, apps, and services, before going on to test ECU networks, and only then testing the IT security of the vehicle as a whole.

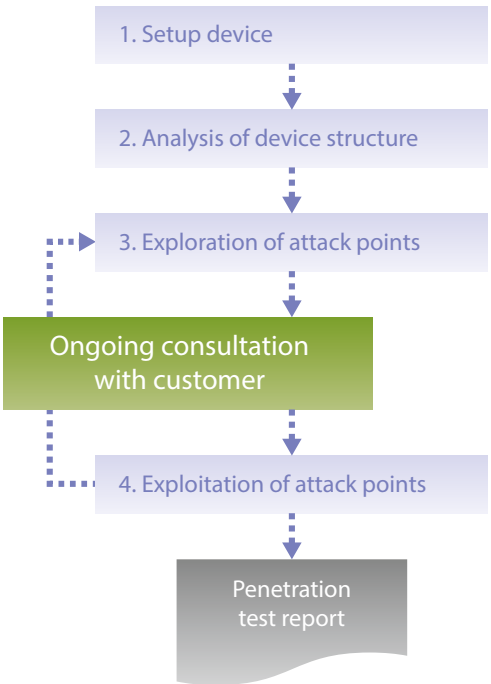

Figure 2: In penetration tests, the tester plays the role of the hacker and tries to get into the target system

Blackbox 	Greybox 	Whitebox 
Public information only	Partial specification and documentation	Specifications, documentation, configuration, source code, shell access
Low Efficiency Time is spent on reverse engineering instead of testing ► Might miss deeper attack paths	Medium Efficiency	High Efficiency Time is spent on testing and not on reverse engineering

Figure 3: The more that is known about the system to be tested, the more efficient the testing is

Code analysis

In code audits, security testers search at the source code level for programming errors or security gaps that hackers could exploit. They pay particular attention to the correct behavior of implemented security measures and to code that might possibly be processing hostile input from potential hackers, such as parsers, crypto-implementations, or communication stacks (e.g. for network, radio, user interface). In addition, code audits can detect implementation errors, such as the incorrect validation of inputs or storage problems (e.g. buffer overflows).

For the purposes of high efficiency and coverage, automated code audits are a recommended tool – for example, by means of static code analysis. It should be noted, though, that the focus here is more on the robustness of the code and compliance with best practices (e.g. MISRA-C, CERT-C), than on logical or functional problems during implementation. In the automotive environment, code audits are especially worthwhile for domains that contain particularly high IT security risks. They are often used to test communication interfaces as well as the security functions themselves.

Functional security test

Functional security tests are used to verify whether the security mechanisms used are working correctly and are fully implemented. These mechanisms include cryptographic methods, secure booting, firewalls, communication protocols, secure software updates, and much more. To this end, the testers define positive and negative cases in advance in order to test the correct behavior of the function. Subsequent execution of the tests is generally done on an automated basis. It is vitally important that a high level of security know-how goes into defining the test cases, so that attack scenarios can be realistically simulated. Know-how is also required to validate correct integration on the target platform, as it often behaves differently than the development system. In vehicle environments, integration tests are complex and demanding. Taking the example of typical bus protocols such as CAN, sometimes no direct answer messages are sent, making it hard to ascertain whether the test messages are being correctly processed. Thus, it is often necessary to generate and monitor several signals in a certain sequence on various vehicle buses simultaneously – for example, to test the effectiveness of security protocols or gateway filter functions.

Fuzz testing

Fuzzing is a powerful testing technique. It is often used in conjunction with functional security tests to check the robustness of tested systems. Fuzzers are pieces of test software that are deployed to generate a high number of untypical or invalid inputs. This allows testers not only to run quickly through a wide variety of internal system states, but also to provoke malfunctions, anomalies, and unforeseen information disclosures that hackers could exploit for a cyberattack. A good fuzzer is protocol-aware, meaning it speaks to the target system's protocol and modifies only single aspects of the protocol, such as individual data fields or aspects of the message flow. This "smart fuzzing" is particularly efficient, as it homes in on the fine line between invalid and valid input.

Fuzzing tests call for comprehensive monitoring to be able to recognize malfunctions in the object of the testing. This is by no means a trivial task, as the tested object's reactions are not always recognizable solely on the interface being tested, but can sometimes reveal themselves only in the object's behavior or via output on other interfaces.

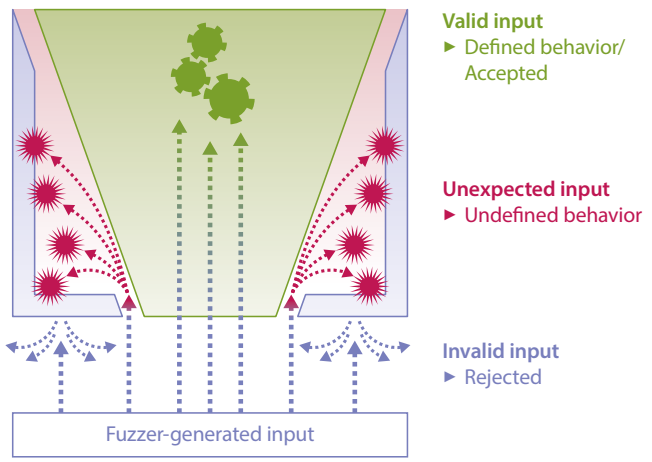

Figure 4: Fuzzing targets the fine line between invalid and valid input

Good fuzzing tools today cover virtually all automotive-relevant protocols: CAN, ISO-TP, UDS, USB, Bluetooth, Wi-Fi, and many Ethernet-based protocols (IP, TCP, UDP, FTP, TLS, etc.). The method is considered a highly effective tool for uncovering a huge variety of weaknesses and their consequences – from complete hard failures or crashes to subtle bugs such as timing issues or memory leaks.

Vulnerability scan

In vulnerability scanning, the target systems are tested for known vulnerabilities, exposures, and security gaps. The testers generally utilize a database in which the weaknesses currently known for the test object have been stored. With the input from the database, the scanner "reads" through the system; in the ECU environment, for example, it scans the Unified Diagnostic Services (UDS) protocol for typical weak points, including seed values that are too low or key calculation algorithms that are too weak. The quality of such vulnerability scans is particularly dependent on the scope of the database used and how up to date it is. While there are numerous comprehensive and up-to-date collections for classic IT vulnerabilities (e.g. the Mitre CVE database), automotive-security-specific databases with the relevant protocols and technologies are mostly still in development. Their utility depends on them being constantly updated. In addition, vulnerability scanning is made more complicated by the low level of standardization in the automotive sector. For the reasons mentioned above, this method requires a high degree of specialization and plenty of resources for database maintenance.

Side-channel attacks

Side-channel attacks are a technique for attacking components involved in the physical implementation of the system. We distinguish between passive and active side-channel attacks.

In a passive side-channel attack (also known as side-channel analysis), the testers seek to draw conclusions about internal data processing by measuring so-called side channels, i.e. physical characteristics of the target system (such as time behavior, power consumption, and electromagnetic emissions). In this way, for example, fluctuations in the power supply of a microprocessor when processing a cryptographic algorithm can provide indications about the keys used.

In active side-channel attacks, the testers try to deliberately manipulate the system. A typical example is fault injection attacks, where testers attempt to provoke processing errors in a microprocessor by

means such as temporarily interrupting the power supply or electromagnetic injections, and thereby trigger a behavior that a potential hacker would seek to bring about.

In security testing, specialization is the name of the game

Our overview of common test procedures shows that automated and manual approaches are intertwined in many cases. Automated tests have the advantage of fast implementation, high reproducibility, good integration capability into test routines, and – last but not least – high cost efficiency. And all the more so when they identify weaknesses at an early phase of the development process, as the sooner security gaps are discovered, the easier and more cost-effective it is to close them. This advantage can be exploited by security testing in the loop (security XiL), which performs automated security tests (functional security, fuzzing, and vulnerability scans) during ECU development in order to identify any security gaps at an early stage.

However – and this is the flip side – despite the scope for automation, security testing remains a job for specialists. Even a brief survey of test procedures reveals that the planning, implementation, and monitoring of the various tests require comprehensive know-how, precise knowledge of hacking strategies, and in many cases specific infrastructure, such as constantly updated databases and test equipment for virtualized security XiL testing. But another aspect also favors the inclusion of security specialists: although automated tests quickly find known weaknesses in widely-used services and protocols, as soon as there are individual customized adaptations, they soon reach the limits of their capability. And customizations are no rarity, specifically in the automotive sector. On top of this, automated testing tools are blind to logical errors in the tested program code or the linking of several weaknesses, each of which are harmless when viewed in isolation, into complex paths of attack. As such, there is a need to complement automated methods with manual tests by security experts. This also applies when access to information about system components and source code is blocked for legal or other reasons – and the testers are forced to do black box testing.

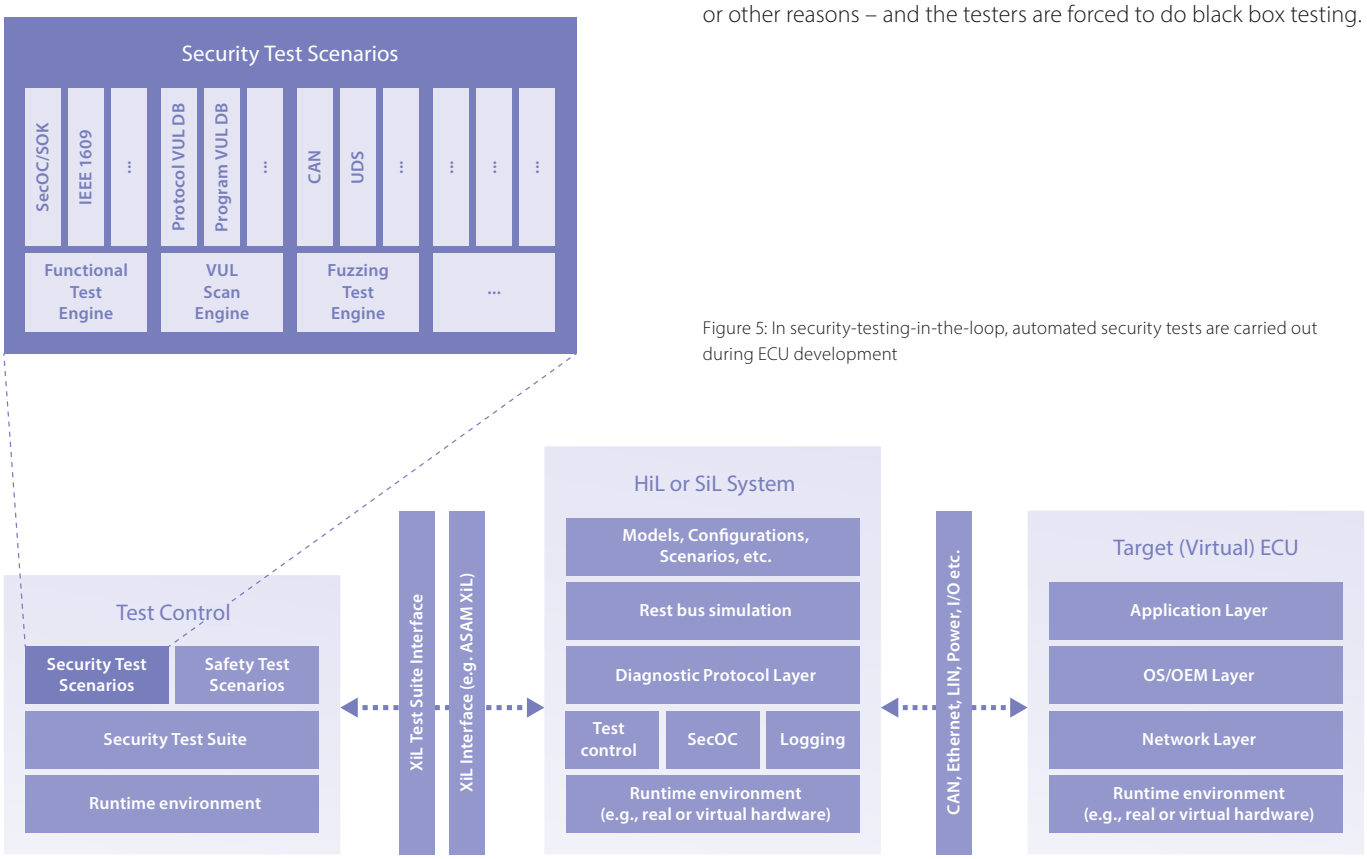

Figure 5: In security-testing-in-the-loop, automated security tests are carried out during ECU development

Interplay of individually adapted testing tools and manual pentesting techniques

It follows then, that in addition to coarse-grained automated testing, highly targeted manual tests are needed to be able to actually cover all test cases. This is where it pays to use testing tools that can be individually adapted to the specifications of vehicle platforms from specific OEMs or even adapted to individual ECUs. For example, ESCRYPT's security testers use fuzz testing tools specifically designed to meet the requirements of certain manufacturers, allowing them to obtain much more accurate test results in a much faster process. Based on the anomalies this brings to light, the actual analysis then begins. To this end, security experts bring their entire know-how and accumulated experience to bear on interpreting the findings in the underlying data, the data traffic, the investigated processes, and the source code. Working in a team, they can trawl through databases for attacks on similar targets, draw comparisons, and develop optimized protection strategies. Experienced specialists are also able to recognize the potential danger in small errors that may seem harmless in isolation, but that could be used in complex attack chains. As valuable as suitable testing tools are, ultimately it is only when they are combined with manual testing by security experts that they yield a comprehensive understanding of IT security, which is indispensable for the effective safeguarding of systems against targeted and elaborate cyberattacks.

To do this, the testing experts proceed on an iterative basis in close cooperation with the manufacturers. First, they identify possible weaknesses and points of attack. Next, they present the manufacturers with these security gaps and the resulting risks, and lay out possible solutions and effective countermeasures. The spectrum ranges from small implementation errors and changes to individual lines of code, to massive IT security flaws, which in the worst case require a complete redesign of the system that was tested. As soon as the gaps have been closed, re-tests are carried out to check the success of the measures taken and identify any need for additional corrections. The iteration loops continue until the agreed scope of testing has been completed in full. These tests can take place both at the component and the system level.

Typical pentest findings

In fact, the testing of development projects regularly uncovers weaknesses of a kind that hackers could potentially exploit for manipulation and cyberattacks. Although the pentesting of ECUs, infotainment systems, and ECU networks always follows an individually adapted strategy, experienced automotive pentesters start off by focusing on typical points of attack and findings.

- **Open security access:** The vehicle diagnostics allow access to a development mode that has remained in the final product as a "backdoor."
- **Open hardware debugging port:** An unlocked debugging port (e.g. JTAG) allows read/write access to the ECU memory as well as analysis and modification of the programs being executed.
- **Insufficient robustness / outdated software:** Insufficient robustness or lack of compliance with coding standards (e.g. MISRA-C, CERT-C) can cause serious errors. For example, if the target system crashes completely when receiving a faulty CAN frame, an attacker can investigate the error in more detail and exploit the underlying problem (e.g. buffer overflow) for an attack. Furthermore, it can happen that although integrated software modules from third-party providers have received corrections over time, these updates are not necessarily installed on the device in question. A good example of this is the Heartbleed security bug in the OpenSSL cryptographic software library in versions older than 1.0.1f.
- **Faulty security functions:** The security mechanisms are not correctly implemented or configured, e.g. a certificate check only ever comes back as "true" due to a logical error during implementation, or a firewall lets all traffic pass.
- **Faulty random number generator:** A faulty random number generator (RNG) compromises access control, allowing an attacker to reuse recorded authentication passes. This has even greater consequences if an RNG is used for cryptographic purposes; in extreme cases, an attacker is able to calculate secret keys.
- **GPS time for validating certificates:** Sending an incorrect GPS signal and manipulating the GPS time to a specific date will invalidate certificates, which can result in functions being permanently restricted.
- **USB support:** The system supports USB functionalities that are not needed, which unnecessarily increases vulnerability. For example, a computer keyboard or an additional network adapter can be used on an ECU.
- **Bluetooth issues:** The Bluetooth protocol has grown to become very complex, resulting in many points of attack. Examples include the use of static default PINs like 1234 or the BlueSmack denial of service attack. In the latter case, large L2CAP echo request packets crash the target system, which needs a power cycle to continue its work. Such an attack can be carried out without much hardware or software and can lead to dangerous traffic situations.
- **Use of non-secure protocols:** In the case of protocols with outdated security mechanisms, message content can be read or manipulated using its encryption algorithm.
- **Cell phone connections:** These can be attacked by forcing a fallback to an outdated GSM; this allows the attacker to manipulate the application protocol with a fake base station. If the application does not have its own protection, the intended remote control functions (e.g. door opening) are vulnerable to abuse, or malware can be installed via other gaps.

At the end of the testing, customers receive a detailed test report, which not only lists the identified points of attack, applied test procedures, and all findings in detail, but also puts forward possible solutions to close the identified IT security gaps. With the publication of the ISO/SAE 21434 standard, these test reports will become an indispensable prerequisite for type approval.

Security testing as part of a holistic security philosophy

As already alluded to several times, security testing doesn't end with type approval and SOP. Rather, connected vehicles require holistic protection throughout their lifecycle. ESCRYPT offers appropriate solutions, ranging from the safeguarding of development and manufacturing processes, to hardware-based in-vehicle security, to cryptographic solutions, to an intelligent active intrusion detection and prevention system (IDPS) for vehicles and vehicle fleets in the field. An IDPS makes it possible to collect all known anomalies of all vehicles in the field into a central cloud-based event database, evaluate them, and immediately initiate countermeasures when the findings point to new risks or attack patterns. With every vehicle that is connected to the network, the IDPS becomes smarter and affords stronger protection. Previously invisible attacks or hacking attempts repelled by firewalls are incorporated into a continuous situational evaluation, in which security measures can be adapted at any time to the current risk situation quickly and in a targeted manner, even after SOP. In this way, the connected vehicle acquires an immune system whose defenses are strengthened by every attempted hack – and that can safeguard the IT security of modern vehicles up to the end of their lifecycles.

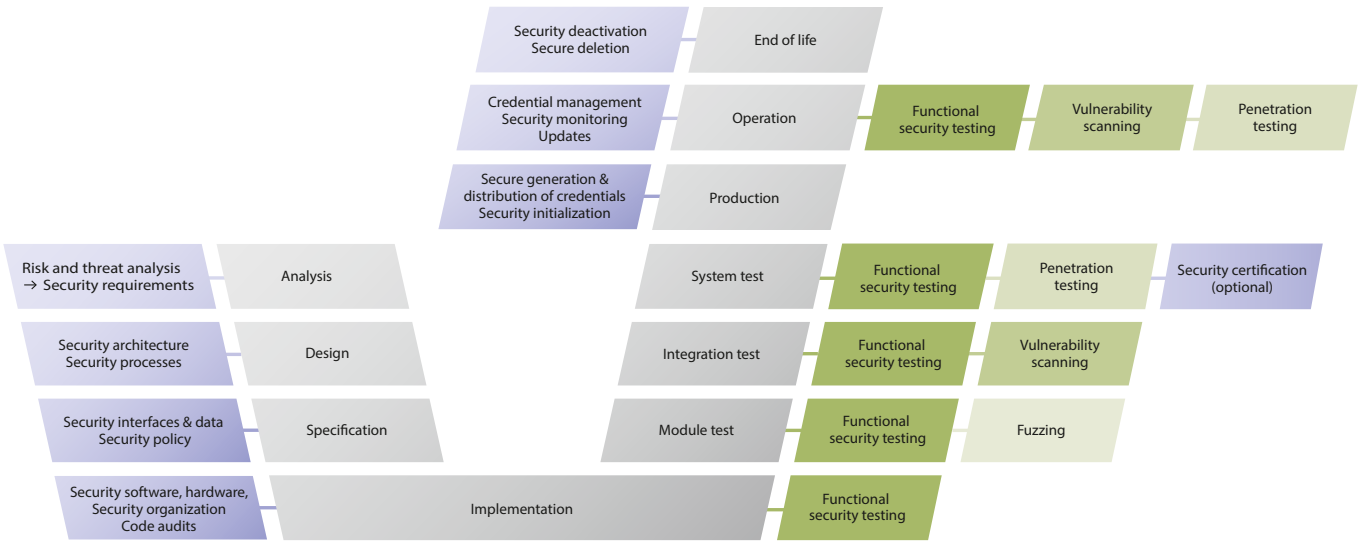

Figure 6: Security testing throughout the complete product lifecycle

Conclusion

Connected vehicles need systematic all-round security protection from the cradle to the grave. Moreover, the upcoming publication of the future ISO/SAE 21434 standard "Road vehicles – Cybersecurity engineering" calls for well-founded and methodical approaches for truly comprehensive security testing at the component and system level. These tests and their documentation will become a prerequisite for the type approval of new vehicle models. Automated and manual test procedures deployed in tandem offer the opportunity to systematically correct weaknesses and fix possible points of attack on vehicle networks at an early stage. This comprehensive testing can be implemented only with specific know-how and the corresponding technical infrastructure. Manufacturers and suppliers now face the challenge of devising test strategies, meeting the personnel and technical requirements for the tests – and embedding them in more comprehensive holistic protective measures for their increasingly connected vehicles. ESCRYPT recognized this trend early on – and over the past few years, the company has built up the relevant know-how and a solutions portfolio that guarantees highly effective, long-term IT security.

Authors & contact details

Dr. Martin Moser
Head of Consulting & Testing Munich
Lead Service Owner Security Testing
Martin.Moser@escrypt.com
+49 (89) 35 64 78-191

M.Sc. Tobias Brennich
Security Consultant
Tobias.Brennich@escrypt.com
+49 (89) 35 64 78-166

Dipl.-Math. Thomas Enderle
Lead Security Specialist
Thomas.Enderle@escrypt.com
+49 (89) 35 64 78-133

ESCRYPT GmbH
Ridlerstraße 57
80339 München
Germany

www.escrypt.com

Last updated:: 1/2022