Constructing Tool-based Security Test Sequences for Vehicles as High-tech Data-rich Systems

Alexandr Vasenev[1], Stelios Karagiannis[2] and Roland Mathijssen[1]
[1]*Joint Innovation Centre ESI (TNO), Eindhoven, The Netherlands*
[2]*Beyond Vision, Ilhavo, Portugal*

Keywords: Test Sequences, Method, Reference Architecture, Pen Testing, Open Source.

Abstract: Vehicles, as a prime example of high-tech systems, get increasingly connected and data-centric with the need to process personally identifiable information. Often, companies that develop such systems act as integrators and need to comply to adequate data protection requirements. For instance, GDPR requires securing personal data. Yet, testing security of data (including, but not limited to personal data) is challenging. Penetration testing often starts from the outside of the system and take place at the end of the development lifecycle. This may be insufficient to adequately test for potential errors hidden within system boundaries. Having methods to design, execute, and reuse (automated) security test cases on a 'white-box' system is desirable. This positioning paper proposes an approach to design tool-based security test sequences. We structurally approach high-level data storing, processing, and communicating functionality in connection to the system boundary. We suggest to use pen-testing tools and sequences for testing the functionality of the vehicle's (sub)system, before test-enabling interfaces are removed. This paper intends to contribute to discussions how to test layered defense implementations. The proposed approach is undergoing extensions and validations.

1 INTRODUCTION

Automotive companies develop their high-tech systems (HTS) by acting as integrators who combine components provided by suppliers. Improper re-use, misconfiguration, or non-compliant components can lead to undesired functionality and, therefore, system vulnerabilities.

Testing security as an integral non-functional system quality is an complex task. It commonly involves simulating attacks and employing other kinds of penetration testing where testers play the role of a hacker trying to attack the system (vehicle) and exploit its vulnerabilities (Felderer et al., 2015). This approach serves its purpose, but is commonly used at late development stages and often such tests start from the outside of the system. As a result, potential errors and vulnerabilities within the system boundary can stay hidden after product release.

Modern-day solutions are often designed with the defence-in-depth concept in mind, where multiple layers of security are placed through a system. Yet, the question remains how to test these inner security measures properly, easily, and securely, without introducing extra weak spots.

Arguably, automated test cases conducted on a 'white-box' system during system integration can assist in providing the proper support and reasoning for security and privacy compliance claims towards the authorities. Test interfaces can provide access to inner components of the vehicle. Clearly, these interfaces are expected to be fully removed before releasing the system.

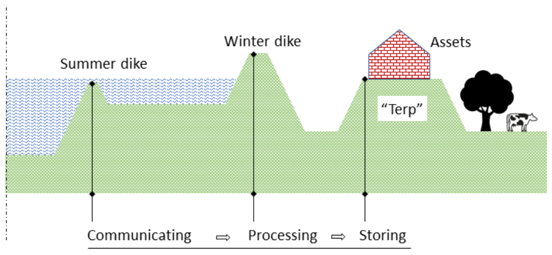

Figure 1: The flow of the analysis.

To approach white-box testing, automotive, similar to other high-tech domains, can borrow ICT concepts. In particular, data handling can be approached as data storing, processing, and communicating functions. For instance, an attacker needs to bypass the communication, processing, and storage functionality to access the GPS history before being able to retrieve the data.

Figure 1 shows an analogy how elements of the data handling functionality triad can be related to each other through the idea of Dutch river dikes. The water represents the environment external to man-made structures. Next to the river is the first protection layer, called the summer dike. Once the summer dike is compromised, the river water can overflow in the retention base, where the water has room to flow, and strong waves and changes are diminished. The retention base is again surrounded by a second protection layer, the winter dike. As it can also be breached, the most valuable houses or villages are built on a higher hill, called the Terp, to provide the last layer of defense.

Similarly, a data asset is typically located behind several perimeters. *Communicating* functionality is located on the system boundary prior to *Processing*, which in turn precedes *Storing*. If an intruder gets into a system, their actions shall be dampened out, another defense layer should exist before the more critical parts of the system, and if that layer is breached, valuable assets need to have yet another defense layer (e.g. extra encryption or access control). Thus, it might take separate steps to access Storing. To ensure data containment (e.g., personal data or algorithms), these steps should also tested.

This high-level functionality triad can be implemented by different in-vehicle (sub)systems with different components depending on the data in focus. Communicating functionality can be mainly included in Gateway or Telematics unit. Yet, while those ECUs can include their own storage and communicating, some critical data may be located deeper in the system. Consequently, the testing approach outlined in this paper deals not with individual components responsible for communication, processing, and storage, but with specific subsets of ECU functionalities structured as communicating-processing-storing of particular data.

This paper investigates how recent ICT security advancements on threat actor modelling can provide insights needed to design test cases. We explore how to apply available security tools in sequences based on available security (pen)testing tools and methodologies. For this, we consider:

- The concept of kill chains (sequences of steps) needed for an adversary to achieve a goal, e.g., to access data;
- A subset of penetration testing tools available (for longer lists see, e.g., (Software Testing Help, 2019), (ProfessionalQA, 2019));
- High-level decomposition of high-tech systems in terms of data storing, processing, and communicating functions.

This paper takes a connected vehicle case to illustrate how to design tool-based test sequences. These sequences aim to test the ‘functionality triad’ in a manner independent from the system topology.

2 BACKGROUND

2.1 Introduction to Security Testing

Security tests are often performed on complete systems and focus on simulating situations when the system is attacked from a particular external interface. For instance, (Mouratidis and Giorgini, 2007) describes an example testing over available interfaces.

The reasons behind testing nearly completed systems are often justified. First, testing from the outside simulates adversary activities when one needs to exploit interfaces to proceed. Second, the testers often have a multitude of other aspects to consider, not to mention the functionality of the barrier itself. At the same time, testing external interfaces can be insufficient, because it could limit the test coverage within the (sub)system itself. It was repeatedly shown that most barriers can be breached by brute force, social engineering, phishing, lost passwords, and other means. Additionally, although penetration testing often takes place late in the system design lifecycle, testing can be beneficial throughout the development life cycle and on units (Arkin et al., 2005).

The question remains how to test what an intruder can do, once access is gained. Although layered defenses are becoming more common, it is often unclear how to (automatically) test such defense-in-depth constructions.

A possibility can be to test systems before testing interfaces are removed, i.e., when testers have full access to the system. Testing in such conditions simulate a situation when the adversary already obtained a foothold in the system. Therefore, designing tests cases that focus on subsystems and from within the system appears to be a promising direction. However, methods to design such tests are not so well documented, or at least not many strategies can be found on this topic.

Interest to bring testing earlier in the development cycle led to research on testing with the use of simulators. One example is to employ hardware-in-the-loop testing. Another possibility concerns validation platforms like (Ming et al., 2016) to benefit from OMNeT++ as an event-based network simulator, and SUMO (a road traffic simulator). Simulated attacks can include Message spoof, malicious fault of devices, physical disturbance. Such solutions help to reduce testing and validation efforts, avoid creating dangerous road situations and lower validation costs. Yet, they do not cover testing defence-in-depth aspects related to data security.

This paper investigates whether pen testing tools commonly used at a rather late system development stage can also provide benefits at an earlier stage (when test interfaces are still available). We wish to contribute to discussions on practical strategies for subsystem testing and full system testing in contexts when the outer layer of defence is breached.

2.2 Automotive Security Concerns

The automotive domain provides a suitable example of a complex software-intensive high-tech system available in large numbers all over the world. In this aspect it is similar to many other high-tech systems, e.g., from medical, production, and financial domains, that are readily accessible over the internet. In principle, remote access to them is only limited by well-defined interfaces.

New vehicle functions that rely on connectivity do attract customers, but they also open up the system to many external attack vectors. As a result, vehicle (sub)systems can be accessed by Internet, GSM, or Wi-Fi in addition to traditional on-board diagnostic ports (OBD) and Controller Area Network (CAN) communication buses. Accessing the latter can imply physical interactions with the vehicle (e.g., by simply drilling a door of the car). But it can also be done in the aftermath of remote attacks, as the following cases demonstrate:

- A buffer overflow error can grant unauthorized access to an in-car media player. This breach of *processing* allowed the adversary to propagate through a CAN bus and reach multiple Electronic Control Units (ECU) (Wired, 2015).
- A dongle used for insurance purposes that is plugged into the dashboard can be remotely hacked through external *communication* channels. This allows attackers to access the vehicle's essential system actuators, such as brakes (IBS, 2015).
- An adversary can take control of an electric vehicle remotely by using an app. This results in accessing the vehicle’s *storage* with GPS history, which can be exfiltrated through *communication* channels (IBS, 2016).

These examples demonstrate that relying only on system boundaries can be insufficient to prevent remote access to in-vehicle components. Developers need to consider protecting the in-vehicle functionality as well. For this, data storage, processing, and communicating functionalities shall be adequately designed, implemented, and tested.

2.3 Security Testing Approaches

Security professionals examine high-tech systems by using specialized tool-based approaches like vulnerability assessment or offensive methods. Such manual and automatic security testing approaches have their advantages and disadvantages (Wallingford et al. 2019) and can provide valuable insights for designing security tests and sequences.

Vulnerability assessment as a process to identify and rank vulnerabilities can be compared to risk assessment. This activity provides significant insights, but can be time consuming with too many attack vectors to control and simulate.

Offensive methods support more focused system examinations. In particular, penetration testing as series of tasks positively impacts the enhancement of system security (Applebaum et al., 2017) by considering specific attack vectors. Such testing often uses the attack trees as conceptual diagrams representing actions of the adversary in connection to the asset. For different attack vectors, specific penetration tools like Nmap, Hydra, Msfconsole and Dirb can be used.

Using (automated) test sequences can help to overcome some difficulties of manual assessments, as noted in (Miller et al., 2018). Created tests can be executed systematically and, in some cases, continuously. Benefits of (semi-) automated penetration testing methods are described in (Samant, 2011). As an example, a semi-automatic security evaluation helps to enumerate automotive security issues with severity ratings on each step of the attack chain (Cheah et al., 2018).

Adversary emulation is an approach that assists to structure potential (inter)connected actions of an adversary. Centered on scenarios, it helps to effectively utilize threat intelligence, discover advanced persistent threats (APTs), and anticipate and better describe possible attack vectors. Cybersecurity community has accumulated a number of methods for describing attacks, such as threat models or attack trees. A prime example is an ATT&CK model created by MITRE researchers (Strom et al., 2017) based on real-world observations. This represents a knowledge base of adversary tactics and techniques. The *Cyber Kill Chain* framework describes cyber-attacks as sequences of typical stages including: reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objective. This approach was used to describe Havex and Stuxnet cases (Assante and Lee, 2015) and is a subject of extensive research.

The described data functionality triad constructed as a sequence resembles a simplified kill chain. This similarity provided us with an aspiration to reuse security methodologies and tools for testing layered security.

3 PROPOSED APPROACH

As mentioned, this positioning paper outlines possibilities to envision test sequences by employing high-level data handling operations described as (nested) Communicating, Processing, and Storing functionality related to the system boundary (Figure 1).

The proposed steps of identifying and sequencing tools for 'white-box' testing of systems during the integration step include:

1. Construct a high-level functional decomposition of (sub)systems involved in the Communicating-Processing-Storing triad based on the focus of the test sequence. For instance, components that deal with GPS-related data.
2. Set up initial assumptions which software can implement the functionality of the triad elements. This step is particularly relevant for embedded systems. For instance, an ECU may have limited resources to run general-purpose operating systems, while a vehicle's gateway can suffice. Such ECU capabilities can limit usage of, e.g., TCP/IP tools or web applications.
3. Relate available tools to the functional decomposition from step 1. Specifically, to identify groups of tools for:
   a. Setting up test management environment and the testing context;
   b. Testing Communicating functionality (located on the system boundary);
   c. Testing the Processing functionality of the system;
   d. Testing Storing functionality.

4 ILLUSTRATIVE EXAMPLE

This section illustrates how relevant groups of tools can be identified for a connected vehicle case.

Step 1: Figure 2 shows how a vehicle use case can be described using the high-level Communicating-Processing-Storing triad. We assume that a connected vehicle interacts with a traveler, a traveler's device (e.g., smartphone), and external entities (infrastructure or other vehicles). This section denotes them as Vehicle, TD (Traveler's Device), and OtherS (Other Systems). The latter include, for instance, cloud-based services for renting a vehicle or advanced authentication schemes. Human machine interface (HMI) represents a specialized subset of Communicating functionality to interact with the traveler. Actuating and Sensing functionality provide inputs for Processing and act on Processing commands.

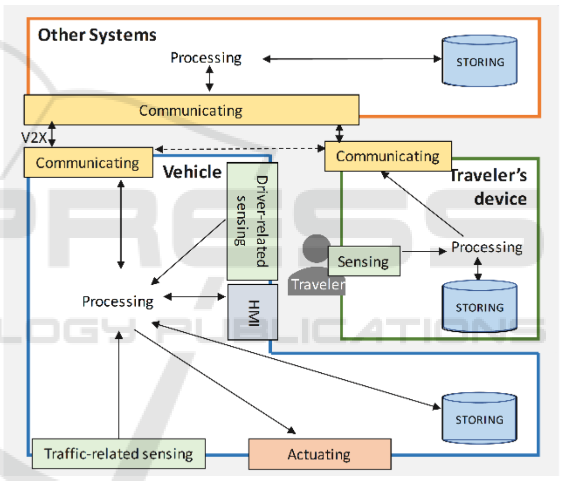

Figure 2: High level architecture of an automotive system.

Step 2: We can set up assumptions on capabilities of components implementing the Figure 2 functional elements as follows:

- Processing capacity of Vehicle, OtherS, and TD systems can support Linux and TCP/IP tools;
- Vehicle.HMI (infotainment) and OtherS.Processing can incorporate webservers;
- OtherS.Processing can support honeypots;
- OtherS.Communicating can employ Zigbee.

Step 3: Existing pen-testing tools can be used to test components that deal with high-level functionality shown in Figure 2. Table 1 shows a (non-rigorously constructed) list that can illustrate the approach. The relation between the tools to the high-level functionality is not exhaustive and used here for illustrative purposes. We refer both to a system (Vehicle, TD, OS) and functionality element. For instance, '[Vehicle/TD].HMI' indicates that the tool can be used to test Vehicle's HMI or TD's HMI. Another example, "All.Storing" indicates that the tool can be related to the Storing functionality of all systems from Figure 2.

Table 1: An illustrative mapping of existing tools to high-level system functionality.

Tool name	**Applicability for testing:**	
	Figure 2 functionality	***Comment***
Ail-Framework	All.Storing and All.Communication	On captured data contents
BeRoot	All.Processing; Vehicle.HMI	Privilege escalations
CAN-alyzat0r	Vehicle.Processing	Car protocols
CANToolz	Vehicle.Processing	CAN
Can-Utils	Vehicle.Processing	CAN, car network
Cloakify	Prepare All.Storing	Steganography
DefectDojo	--	Management tool
EvilLimiter	All.Processing (links to .Communicating); Vehicle.HMI	Linux and network. ARP spoofing
Infection Monkey	OtherS.Processing	SSH, WMI, passwords
KillerBee	OtherS.Communicating	Zigbee
Lnx Smar Enum	All.Processing; Vehicle.HMI	Checks for run-ning environments
MitmAP	All.Communicating	Simulates an Access Point
Net Creds	All.Communicating (incl. links to All.Processing); Vehicle.HMI (incl. link to .Processing)	HTTP; FTP; Telnet; SMTP; Kerberos; NTLM
Nmap and Zenmap	All.Processing	Scanner
OWASP Honeypot	OtherS.Processing; Vehicle.HMI	Honeypot; web server
P4wnP1	All.Communication; [Vehicle/TD].HMI	Emulates another device behavior
Poisontap	[Vehicle/TD].HMI; All.Communicating (USB)	Sniffer. Provides an entry point
RedHunt-OS	All.Processing; Vehicle.HMI	Aggregator of tools
SheepL	OtherS.Processing (incl. link to .Communicating)	Simulates benign PC users' network behavior
ShellSum	OtherS.Processing; Vehicle.HMI	Detects web shells, Web-server
SSH Key Auth	[Vehicle/TD/ OtherS].Storing	Manage SSH keys
Suricata	All.Processing (Gateway)	Intrusion detection

Tools to set up the test environment can be used to (1) simulate scenarios how high-tech systems are utilized and (2) assist security testers in managing the test process. Afterwards, tests can be executed in connection to a particular functionality. For instance, MitmAP as a Communication-related tool can simulate a rogue access point; or Ail-Framework can assist in testing Storing. Complementary to individual tests, test sequences can be organized based on the steps within Communicating-Processing-Storing structure. The testers can assume moving from the system boundary to the data storage or in the opposite direction (e.g., during data exfiltration).

5 DISCUSSIONS

As integrators, high-tech companies need practical tools and methods for testing their systems before release. This paper proposed an approach that can be used as a reference for designing tool-based test sequences. Specifically, this paper argues in favor of extending the use of automated pen testing approaches to earlier integration stages. Testers could identify relevant vehicle components in connection to the data functionality triad.

The described approach suggests how to operationalize a simplified kill chain sequence. While high-level functionality can be deployed to several system components (e.g., Storing and processing can be done at different locations within the system), the tools still stay relevant. The system topology and functional decomposition do not directly impose requirements to this method, but can further elaborate test sequences.

The outlined approach can be applied recursively and form a basis for creating automated testing infrastructure. E.g., after a system component is added during the system integration stage, designated groups of tools can be applied. This could help ensuring continuous testing during system integration.

The system under test must have dedicated test interfaces for testing the security of the system. These dedicated interfaces must give access to the system behind the initial security barrier, however these test interfaces must also provably be fully removed in the final system. Any remnants of this interface might lead to vulnerabilities or forgotten backdoors.

The steps mentioned in the Proposed Approach section are to be tailored according to the case at hand. E.g., storage of particular data can be located in a specific part of the system. Correctly identifying relevant data handling functionality is essential.

Assumptions of ECU capabilities can impact the future mapping between tools and ECUs. Furthermore, the same tools can be used for different types of functionality and in connection to different components. While the structure of the approach would remain the same, it is the tailoring of a specific system that might require inputs from experts.

This paper illustrated the approach, but didn't go into interdependencies of tools and testing infrastructures. Ensuring proportionality of tests, tools, and test outcomes is a subject for future research. Future mapping and applying will provide further insights on their appropriateness and sequences of their use. Test dependencies is another topic for future investigation.

The illustrated list of tools is not exhaustive, rigorously constructed, or undisputable linked to the high-level functionality. To scope the list, we focused on technologies relevant to connected vehicles. The mentioned tools were identified based on their relevance to automated security testing, adversary emulation, vulnerability assessments, attack simulations, and identifying threats, such as threat management tools and attack trees. Yet, this is borrowed significantly from network and web testing. Future research could focus on more comprehensive studying how these and other tools can be used to devise advanced strategies of testing and simulated adversary activities.

ACKNOWLEDGEMENTS

The research is carried out as part of the SECREDAS project, which is co-funded by the ECSEL Joint Undertaking of the European Union under grant agreement number 783119 and the Netherlands Organization for Applied Scientific Research TNO.

REFERENCES

Applebaum, A., D. Miller, B. Strom, H. Foster, and C. Thomas, 2017, "Analysis of automated adversary emulation techniques." In Proceedings of the Summer Simulation Multi-Conference (p. 16). Society for Computer Simulation International, July 2017.

Arkin, B., S. Stender, and G. McGraw, 2005. "Software penetration testing. Security & Privacy, IEEE, 3(1): 84-87.

Assante M.J. and L.M. Lee, 2015, "The industrial control system cyber kill chain". SANS Institute InfoSec Reading Room, 1

Cheah, M., S.A. Shaikh, J. Bryans, and P. Wooderson, 2018, "Building an automotive security assurance case using systematic security evaluations," Computers & Security, 77, pp. 360-379.

Felderer, M., M. Büchler, J. Martin, A. Brucker, R. Breu, and A. Pretschner, 2015, "Security Testing: A Survey", 10.1016/bs.adcom.2015.11.003.

IBS, 2015, "International Business Times. Hackers disable Corvette brakes by texting dongle meant to lower insurance risk", http://www.ibtimes.co.uk/hackers-disablecorvette-brakes-by-texting-dongle-meant-lower-insurance-risk-15151253, Last accessed on Dec 20, 2019.

IBS, 2016, "Hacker takes control of Nissan electric vehicle from other side of the world through Leaf app", http://www.ibtimes.co.uk/hacker-takes-control-nissan-electric-vehicle-otherside-world-through-leaf-app-1545808. Last Accessed on Dec 20, 2019.

Miller, D., R. Alford, A. Applebaum, H. Foster, C. Little, and B. Strom, 2018, "Automated adversary emulation: A case for planning and acting with unknowns," 2018.

Ming L. et al., 2018, "A General Testing Framework Based on Veins for Securing VANET Applications," IEEE SmartWorld, Guangzhou, pp. 2068-2073.

Mouratidis H. and P. Giorgini, 2007, "Security Attack Testing (SAT)—testing the security of information systems at design time". Information Systems. 32. 1166-1183. 10.1016/j.is.2007.03.002

ProfessionalQA, 2019, "Penetration Testing Tools: Top 55", July 16, http://www.professionalqa.com/penetration-testing-tools. Last accessed on Dec 20, 2019.

Samant, N., 2011, "Automated penetration testing"

Software Testing Help website, 2019, "19 Powerful Penetration Testing Tools In 2020 (Security Testing Tools)", Dec 14, https://www.softwaretestinghelp.com/penetration-testing-tools/. Last accessed on Dec 20, 2019.

Strom, B.E., J.A. Battaglia, M.S. Kemmerer, W. Kupersanin, D. Miller, C. Wampler, et al. 2017, "Finding cyber threats with ATT&CK-based analytics," Technical Report MTR170202, MITRE.

Wallingford, J., M. Peshwa, and D. Kelly, 2019, "Towards Understanding the Value of Ethical Hacking. In International Conference on Cyber Warfare and Security", pp. 639-XIV. Academic Conferences International Limited.

Wired, 2015, "GM Took 5 Years to Fix a Full-Takeover Hack in Millions of OnStar Cars", https://www.wired.com/2015/09/gm-took-5-years-fix-full-takeover-hack-millions-onstar-cars/#. Last accessed on Dec 20, 2019.