Supplementary Materials:
Revisiting Automotive Attack Surfaces: a Practitioners' Perspective

1. More Insights from Interview

We present the other 13 key points identified from the interview, which will offer a broader insight on the automotive cybersecurity industry.

• **KP.8: Non-security groups lack proper security knowledge.** Due to the complexity of modern vehicles, manufacturer companies often consist of a wide variety of groups working together, including the development groups, security groups and others. In our interview, a common challenge identified by ALL interviewees is that non-security groups often lack consensus in security. E.g., as P11 reported: "*Our group knows clearly the meaning of the critical/high/low risks in the TARA results, but other groups do not even know what TARA is.*" (P11). Accordingly, this fact makes it laborious for the security groups and non-security groups to reach a consensus for decisions on particular threats, and it's common that security groups have to show the practical attack results to other groups to present the rationale for security-related requests. E.g., as P11 reported: "*Our development groups actually care so little about security: they totally do not understand why it is necessary to update the system components. For example, the development group thinks that the built-in components in the IVI Android system are safe to use, even when their versions are out-of-date. As a result, we have to craft a practical PoC attack chain to show that the out-of-date codes are vulnerable and the significance of system updates.*" (P11). Overall, as identified by our interview, it is common that the interpretation of the rationale to perform security activities requires a lot of effort, and due to the lack of automatic tools for risk assessment, this process heavily relies on the manual effort and is very inefficient.

• **KP.9: Complex supply chains bring new challenges.** As modern vehicles are becoming increasingly complex in its interfaces and in-vehicle architecture, the corresponding supply chains also get complex and thus bringing new challenges. In particular, as reported by P13: "*We are consistently pushing the security requirements to our supplier, including performing security testing and providing the specific software materials to us. However, it is very common that suppliers are still not attaching enough importance to cybersecurity, and thus they are not able to meet our requirements.*" (P13). Additionally, P14 presented the challenge from the 1st-party OEM: "*It is also very challenging for the 1st-party OEM to ensure the supply chain security: they need to present the very specific cybersecurity requirements to the supplier, and also be capable of reviewing whether the requirement is met. For example, they should give very detailed information about what TLS version and what encryption algorithm should be applied in the specific case, instead of just saying 'follow the best security practice'.*" (P14).

• **KP.10: Conflicts with other groups are common.** As identified from ALL 1st-party interviewees, one fact that they face is that security activities are "costy" and cannot be translated into immediate and direct benefits. As reported by P12: "*We are always compromising with the development groups, and this is inevitable: the design of a strictly secured system requires extra efforts for development group, and often causes a decrease in user experience. As a result, we are always looking for a balance for 'just sufficient' security and reasonable development effort.* " (P12). Additionally, as reported by P15: "*Overall, implementing fancy features is the top priority for development groups, and security does not directly add attractiveness to the product. As a result, we are always trying to reach the sufficient security design and also try to reduce the workloads of development.* " (P15). This key point is also consistent with KP.8: to reach a common security consensus, security groups tend to make a lot of effort to explain the specific threats.

• **KP.11: Information is not transparently shared between groups.** As indicated by KP.8, current 1st-party companies are consist of many different groups, with different responsibilities and team values, even with possible competitions. Accordingly, another intriguing challenge presented by P13 is that the limited information sharing can affect the security activities: "*Information gathering is the very essential stage for our penetration testing, but the information we can access is often very limited, which could affect the efficiency of our testing. For example, other groups may not pay attention to some malfunctions or bugs, but they could be identified as the critical vulnerabilities in our testing. However, other groups may refuse to offer the explicit details in the first place.*" (P13).

• **KP.12: Security activities get inconsistent between various security groups.** It is common that multiple security groups contribute to the cybersecurity of the same car. For example, when 1st-party manufacturers have assembled the vehicle, they might ask multiple security groups to perform testing on the final product. However, the inconsistent testing output from various groups could cause problems. E.g., as reported by P8: "*It is common that multiple testing groups cannot reach a final decision due to the lack of information* *sharing. For example, when other groups have identified the security problems we missed, we might not be able to validate them due to the limited information provided. Vice versa, when we identified a problem that other groups failed to find, we may not be able to locate the relevant responsible party, or further validate whether the problem is fixed in the final product.*" (P8).

- **KP.13: Lack of concrete support for rationales behind security-related CRs.** The development of automotive products is often based on Change Requests (CRs). However, it is identified by all 1st-party interviewees that the development groups often think that the security-related CR (e.g., fixing a bug) lacks rationale or concrete supports. E.g., as reported by P13: "*Development groups are often not willing to accept our CRs to fix certain bugs, because they do not think 'the CR solely based on our inner-group testing' is convincing.*" (P13). Moreover, P14 gave more comments about this challenge: "*Currently, it is a fact that the security activities are short of concrete support, especially from the compulsory regulations. It is common that other groups may challenge security requests, and try to 'lower' the security baseline.*" (P14).
- **KP.14: Reactive TARA overwhelmed Proactive TARA.** Another challenge we identified is that the TARA process tends to be reactive instead of proactive (by P10, P11, P12), which may raise concerns. In particular, as reported by P11: "*I think ISO 21434 would want us to frequently perform TARA in a proactive way, to ensure the cybersecurity consistently. However, how we use TARA is more like a reactive way: we only use TARA when specific events happen, for example, when development groups want to remove some security functions, or add some new functions. In this case, we will use TARA to demonstrate the corresponding risk. But in other cases, we would not do TARA very frequently or proactively.*" (P11).
- **KP.15: What companies care the most is how to pass the test.** One fact we identified from ALL 1st-party OEM is that current regulations, especially WP29 R155e and GB/T series that listed specific threats, are not treated as the *gold standard* or *de facto oracle* to ensure cybersecurity. Instead, they are merely the security baseline that companies are trying to meet, with adequate or even minimal effort, and this fact is consistent with the challenges we identified from KP.4 and KP.9. E.g., as reported by P2: "*The specific descriptions for the threats and attacks are just auxiliary content. For us 1st-party OEM, what we care about the most is how to meet the requirement for each listed clause.* " (P2). As also reported by P3: "*Our company have grown quite mature in cybersecurity, and we have already considered all threats listed in R155e. Accordingly, we never expect to rely on this regulation to ensure cybersecurity, and all we care about is how to pass the standard set up by the regulation.* " (P3).
- **KP.16: Companies are unsure of what level of protection is sufficient.** Unfortunately, companies do not know to what extent the protection is sufficient, as none of current regulations has make this requirement clear, which is identified by ALL interviewees from 1st party. As stated in KP.15, companies are always trying to find the *just sufficient* cybersecurity solution with adequate effort, but current regulations are extremely short of this information. E.g., as stated by P2: "*We are unsure about whether our mitigations are sufficient as the current regulations themselves are not clear about that. For example, when protecting the scenario of using digital keys to open doors, does it mean that attacker should not be able to get in the communication channel at all, or it would be sufficient if we can make sure no damage will be done even if the attacker can inject the channel? Currently no regulations are making these details clear.* " (P2).
- **KP.17: Current regulations lack quantifiable criteria for evaluating threat cases.** KP.17 is also identified from ALL interviewees, and is the main reason leading to KP.16: because no quantifiable criteria is set up, company do not know how to prepare the protection. E.g., as stated by P3: "*We urgently need a very specific and quantifiable criteria, so that we can prepare our cybersecurity solutions accordingly. However, current situation is that, neither we manufacturers nor the certification authority knows how to perform the cybersecurity test.* " (P3). Also as stated by P11: "*We are constantly evaluating our products based on the threats given by WP29 R155e. However, the threats given by R155e are very high-level, and they are often interpreted by the certification authority, and what they say goes. I think it is strange that these specific metrics are explained by the third parties, instead of the regulations themselves, and I think it is one of the most significant weakness.* " (P11).
- **KP.18: Mitigation listed in current regulations is more like remedies rather than high-level solutions.** A majority of interviewees (12/15) agreed that current listed threats seem to focus on discrete remedies instead of high-level cybersecurity solutions. E.g., as reported by P1: "*Currently listed mitigations for specific threats are more like some discrete remedies, rather than some high-level solution that could be considered and applied in the development stage. Although it is challenging to provide detailed and practical high-level solutions, our group is currently working towards this goal and I am expecting such a content in future regulations.* " (P1).
- **KP.19: Clearer guidelines are needed for long-term security management.** We identified that the long-term management of the product cybersecurity is an extremely challenging tasks, due to the insufficient contents of the regulations, and some other difficulties. Particularly, as reported by P14: "*Although ISO 21434 has provided quite detailed guidelines on how to ensure the cybersecurity in the development stage, it currently failed to give clear guide on the long-term security management. The long-term management of the automotive cybersecurity is a very challenging task, and all the automotive companies are exploring how to establish a sound long-term risk management system. I hope future regulations will give more insights on this process.* " (P14).
- **KP.20: There is a lack of an open platform for sharing threat cases.** Another interesting insight we identified is that the sharing of information, especially the knowledge about

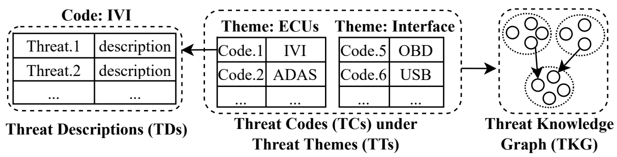

Figure 1: The proposed hierarchical framework to describe automotive cybersecurity threats.

specific threats, is often very difficult. This is often due to the very strict examination process to prevent possible leaks of specific threats. Such a examination process is necessary, but would inevitably hinder the communication between different groups. As reported by P13: "*Although there are various ways to access new knowledge, it is common that many details are still missing in the public document, for example, some vulnerability disclosure documents. As a result, we can only derive some general insights rather than technique details, making it hard to actually try the vulnerability in our own. The situation is the same for us: when we identify the threats which are not so common in the moment, it is also difficult for us to communicate with other groups or to output our content to the industry. As a result, it would be very helpful if future regulations could set up a secure and efficient way to share the identified threat.* " (P13).

Summary: The industry is currently facing a series of challenges in implementing security activities. The reasons for this are multifaceted, including the unique nature of security teams (e.g., difficult to directly generate profits), the complexity of modern vehicle architectures, and the inadequacy of current regulations. We have exposed a range of such issues from our interview, and hope to provide useful recommendations for the improvement of future regulations.

2. Improved Threat Database

2.1. Hierarchical Framework for Automotive Threats

In response to the lack of high-quality automotive threat database, we construct a new threat database that is improved by the collected threats from the interview. Particularly, we use a hierarchical framework to present the automotive-specific threats (in Fig.1), in which the involved concepts are explained as follows:

Threat Description (TD). A threat description (TD) is the smallest element in the framework. It is a set of natural language sentences to describe the details of one particular threat, including the specific Attack Description (AD), the Root Cause (RC) of the threat, the Security Testing Approach (STA) to identify the threat, and the MitiGation (MG) to prevent the threat.

Threat Code (TC) A threat code (TC) is a group of TDs under a particular category. Here the word *"code"* comes from the qualitative analysis methodologies [13], in which the process of *coding* is to give labels to the qualitative data (e.g., interview texts). For example, in Fig.1, *Code.1 IVI* is the code containing the threat descriptions under the in-vehicle infotainment (IVI) ECU.

Threat Theme (TT) A threat theme (TT) is a group of threat codes following a particular high-level classification logic. For example, in Fig.1, the *Threat Theme: ECUs* includes the threat codes representing the in-vehicle *ECUs* (e.g., IVI, ADAS), while *Threat Theme: Interface* includes threats related to vehicular interfaces (e.g., OBD, USB).

Threat Knowledge Graph (TKG) We derive the concept of knowledge graph (KG) [10, 15, 18] to further represent the relations between the threat codes. Specifically, a knowledge graph can be represented by a set of triplets: *(head entity, relation, tail entity)*, meaning that the *head entity* and the *tail entity* has the particular *relation*. In our scenario, the entities are the threat codes, and the triplet *(TC.1, relation, TC.2)* represents the logical relation between the two codes. For example, the triplet *(Code.1 IVI, vulnerable to threats in , Code.6 USB)* connects the code IVI and code USB because the USB interface is a common interface on IVI.

2.2. Detailed Threat Themes

The final result of our threat database is shown in Fig.2, with the following specific threat theme and codes:

T1: General Requirements. The various ECUs can share a set of theats that are general to various implementations, and this T1 describes these common threats from five threat codes: *C1.Hardware, C2.Software, C3.RTOS, C4.Complex OS, and C5.Data*. The advantage of setting up this theme is that *we do not need to repeat these common threats in the specific ECU categories*. For example, secure boot is the de facto mitigation that should be deployed on various types of ECUs. There are 24 threat descriptions under T1.

T2: In-Vehicle Components. T2 describes the threats to specific components in the vehicle, including the threats on various ECUs and on the In-Vechile Network (IVN). T2 contains the following 8 codes: *C6.IVI, C7.Telematics, C8.Sensor, C9.Gateway and Zone Controller, C10.ADAS, C11.IVN, C12.BMS, and C13.Other ECUs*. These codes focus on the threats that are particular to the function of the ECU. For example, the *C6-10: browser threat*, is the very specific threat that exists in the IVI but not on other ECUs, because the browser module has been widely used in the IVI system to support rich infotainment functions. There are 36 threat descriptions under T2.

T3: Outside-vehicle Components. T3 describes the threats for specific components outside the vehicle, but can communicate with the vehicle and affect automotive cybersecurity. Specifically, T3 contains the following 3 codes: *C14.Mobile APP, C15.Backend Server, C16.Charging Pile*. The vulnerabilities in these external components can pose a threat to the vehicle itself. For example, the private data can be leaked through the charging pile. There are 14 threat descriptions under T3.

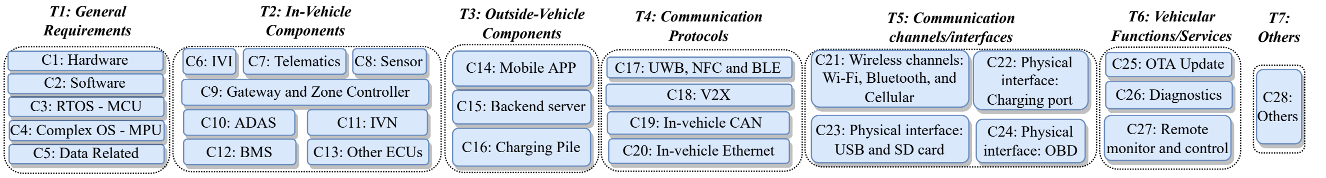

Figure 2: An improved hierarchical threat database derived from the interview study, containing 28 threat codes (TCs) under 7 threat themes (TTs). This database serves as an improvement to existing regulations both qualitatively and quantitatively, and is detailed in §2.4.

T4: Communication Protocols. T4 describes the threats to the communication protocols implemented in the automotive context. Specifically, T4 contains the following 4 codes: *C17.UWB, NFC and BLE, C18.V2X, C19.CAN, and C20.Ethernet*. The unsafe implementation of these protocols can introduce risks. For example, lack of encryption on the data transmitted via the protocol can lead to information leak. There are 16 threat descriptions under T4.

T5: Communication Channels/Interfaces. T5 describes the threats on the communication channels and interfaces on the vehicle. Specifically, T5 contains the following 4 codes: *C21.Wi-Fi, Bluetooth and Cellular, C22.Charging Port, C23.USB and SD card, C24.OBD*. Unsafe implementation of these interfaces leads to threats when these interfaces are exposed to the attacker. For example, the attacker can modify vehicular parameters through the OBD port due to the lack of proper authentication. There are 15 threat descriptions under T5.

T6: Vehicular Functions/Services. T6 describes threats to vehicular function and services, with the following 3 codes: *C25.OTA, C26.Diagnostic, C27.Remote monitor and control*. The implementation of these "trendy" functions can vary for different manufacturers and car models, and can introduce risks when the design is insecure. For example, the unsafe implementation of the secret keys for remote control can be exploited to launch attacks. There are 12 threat descriptions under T6.

T7: Others. T7 includes other threats (e.g., insider attack) that do not fit into other themes. There are 3 threat descriptions under T7.

Several authors have gathered together and performed various rounds of revision on this threat database. The complete database is presented in §2.4. Moreover, we analyzed the relations among the threat codes and presented a TKG which presents the connections of the threats (in §2.3).

2.3. Relation Analysis among Threat Codes

To better illustrate the relations among the threat codes, the 28 codes in Fig.2 are classified into the following two types:

Entity code. An entity code represents a specific automotive *component* carrying functions that can introduce security threats. The entity codes are often the object component in TARA or security testing. Specifically, all 11 codes under *T2: In-Vehicle Components*, and *T3: Outside-Vehicle Components* are the entity codes.

Property code. A property code represents one specific security property residing in one entity code. Specifically,

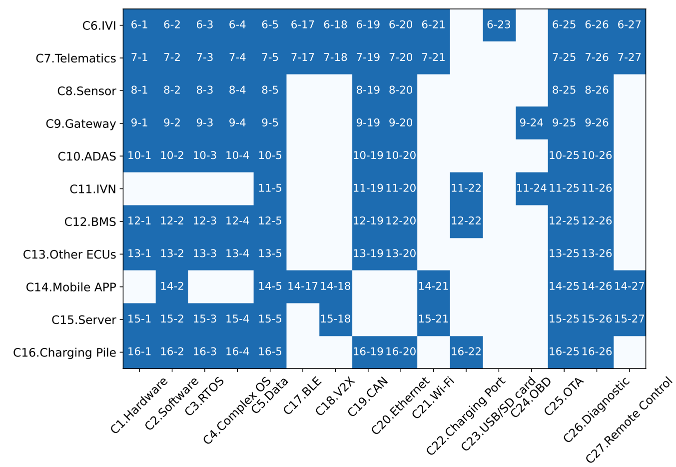

Figure 3: Relation analysis: Adjacent matrix showing the relation between the entity codes (on Y axis) and property codes (on X axis). Marked cell represents that the corresponding triplet is identified between the entity code and property code.

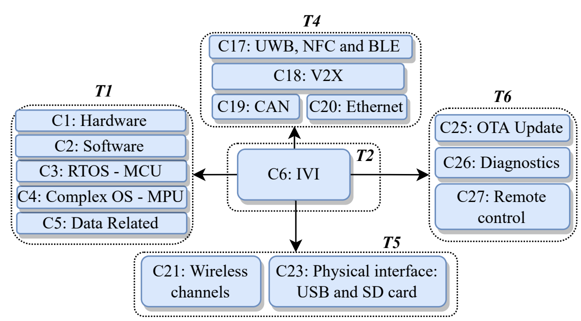

Figure 4: Part of the TKG (14 triplets) showing the property codes related to the entity code: *IVI*. This representation is equal to the first row in Fig.3.

all 17 codes except the 11 entity codes in T2 and T3 are the property codes.

With the above classification, the triplet to build the knowledge graph [10, 15, 18] is further presented as *(entity code, is vulnerable to threats in, property code)*. Finally, we constructed 109 triplets between the 11 entity codes and 17 property codes, and the result is shown in the form of an adjacent matrix in Fig.3. Specifically, in Fig.3, the X axis represents the property codes and the Y axis represents the entity codes. Each marked cell represents that the corresponding triplet is identified. For example, the cell *(C6, C4)* is marked, which means that the threats in *C4: Complex OS* and reside in the entity *C6: IVI*.

Specifically, Fig.4 shows part of the knowledge graph, originating from the entity code *C6.IVI*. This figure represents the threat codes that should be considered when evaluating IVI security. For example, it is very likely that an IVI is equipped with a complex OS (e.g., Linux, Android) to perform various infotainment functions, and thus there is a triplet connecting *C6.IVI* and *C4.Complex OS*. As a result, such a graphical representation can make the TARA and security testing more systematic and comprehensive. Taking Fig.4 as an example, when evaluating the security of the IVI ECU, the threats under other property codes (i.e., the threat codes in T1, T4, T5 and T6 in Fig.4) should also be considered to build a reliable security baseline for this ECU.

These connections among threat codes enable users to understand the interdependencies among automotive threats, aiding them in building CarVal Datalog rules specific to their car models. Furthermore, users have the flexibility to modify and create their own relationships based on different car implementations.

2.4. Detailed Threat Descriptions

The detailed threat descriptions for the 28 codes in Fig.2 are presented from Tab.1 to Tab.28. Each row in the table represent one threat description (TD). The threat code itself represents the Affected Asset (AA), and the other four columns represent the Attack Description (AD), Root Cause (RC), Security Testing Approach (STA), and MitiGation (MG), respectively.

References

[1] Unified diagnostic services (UDS) — Part 3: Unified diagnostic services on CAN implementation (UDSonCAN). https://www.iso.org/standard/55284.html, 2012.

[2] Iso/sae 21434:2021: Road vehicles — cybersecurity engineering. https://www.iso.org/standard/70918.html, 2021.

[3] Aosp: Secure nfc. https://source.android.com/docs/core/connect/secure-nfc, 2022.

[4] Iec 62433. https://webstore.iec.ch/publication/67258, 2022.

[5] Le secure connection. https://www.bluetooth.com/blog/bluetooth-pairing-part-4/, 2022.

[6] Android best security practice. https://developer.android.com/topic/security/best-practices, 2023.

[7] Aosp: Hardware-backed keystore. https://source.android.com/docs/security/features/keystore, 2023.

[8] Information technology — security techniques — code of practice for information security controls based on iso/iec 27002 for cloud services. https://www.iso.org/standard/43757.html, 2023.

[9] Platform security architecture. https://www.arm.com/architecture/security-features/platform-security, 2023.

[10] H. Cai, V. W. Zheng, and K. C.-C. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[11] K.-T. Cho and K. G. Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1044–1055, 2016.

[12] J. Cichonski, J. Franklin, and M. Bartock. Nist: Guide to lte security. Technical report, National Institute of Standards and Technology, 2016.

[13] J. Corbin and A. Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014.

[14] T. Häckel, P. Meyer, F. Korf, and T. C. Schmidt. Secure time-sensitive software-defined networking in vehicles. *arXiv preprint arXiv:2201.00589*, 2022.

[15] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2021.

[16] J. Padgette, J. Bahr, M. Batra, R. Smithbey, L. Chen, and K. Scarfone. Nist: Guide to bluetooth security, 2022-01-19 05:01:00 2022.

[17] M. Souppaya, K. Scarfone, et al. Nist: Guidelines for securing wireless local area networks (wlans). *NIST Special Publication*, 800:153, 2012.

[18] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

TABLE 1: Code 1: Hardware security.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C1-1	An attacker can exploit physical debug ports (e.g., JTAG, USB, UART, SPI) to compromise the target ECU (i.e., IVI and Telematics), allowing them to extract firmware or gain system control	Debug ports are left open for debugging	Conduct penetration testing on the physical debug ports	Manufacturers should disable all physical debug ports or implement security access control on them
C1-2	An attacker can physically analyze the ECU to obtain critical information (e.g., PCB design), or alter the hardware	Absence of physical protection on the hardware design	Testers should physically inspect the hardware design of the PCB from an attacker's viewpoint	Manufacturers should implement measures to hinder such physical analysis (e.g., conceal PINs for critical chips, hide wiring in inner layers, remove readable texts on silkscreen, and apply other hardware obfuscation techniques)
C1-3	An attacker can extract firmware from the chip for further analysis	Absence of physical protection against firmware extraction	Conduct firmware extraction from the attacker's viewpoint	Enable read protection on the flash memory
C1-4	Various physical attacks can be executed on the chips (e.g., fault injection attacks, side-channel attacks)	Absence of best security practices to prevent physical attacks	Conduct penetration testing via potential physical attacks	Manufacturers should implement best security practices according to the SOTA techniques and standards (e.g., Platform Security Architecture levels [9], or IEC 62433 [4])
C1-5	Hardware replacement attack: attackers with physical access to the vehicle can replace specific parts with malicious ones	Absence of authentication for communications between hardware parts	Conduct penetration testing by replacing critical hardware	Manufacturers should implement authentication to detect anomalies when critical hardware is replaced

TABLE 2: Code 2: Software security.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C2-1	An attacker can exploit insecure code in the software installed on ECUs to initiate further attacks	Insecure programming on the ECUs	Conduct vulnerability scanning on the code used in the software systems	Manufacturers should ensure code robustness during development and perform security testing on the software system afterwards
C2-2	The reuse of open-source third-party modules can introduce vulnerabilities that an attacker can exploit	Absence of rigorous security testing before integrating third-party codes	Conduct penetration testing on the third-party code used in the software	Ensure the safety of the introduced third-party code (e.g., always use the latest version and update the code frequently)
C2-3	Software handling incoming messages (e.g., CAN, Bluetooth message) can be compromised by a maliciously crafted message (e.g., triggering a stack overflow attack)	Absence of secure code implementation for handling incoming messages	Conduct penetration and fuzzing testing on security-critical software	Ensure the software implementation properly handles received messages, including authenticating critical messages, preventing DoS attacks, and others
C2-4	An attacker can manipulate the operating system's booting process, leading to information leakage and other subsequent attacks	Absence of protections on the OS booting process (e.g., secure boot)	Testers should conduct penetration testing on the OS booting process to identify vulnerabilities	Secure boot should be well implemented. For example, 1). the Root of Trust (RoT) cannot be overwritten by attackers; 2). the OS should refuse to execute tampered boot code or load manipulated boot image; 3). the secret keys and authentication algorithms should not be easily accessed by the attacker (e.g., using HSM)
C2-5	An attacker can reverse-engineer the extracted firmware to obtain critical information, leading to potential information leaks or product piracy	Absence of protection on the firmware	Conduct penetration testing on the extracted firmware	Manufacturers should enable mitigations to prevent the firmware from being reverse-engineered (e.g., code packing, code encryption)
C2-6	The secret keys protecting the file system and firmware are not securely stored	Absence of secure implementation for storing the secret keys	Conduct penetration testing on extracting the keys	Securely store the secret keys, for example, using the Hardware security module (HSM)

TABLE 3: Code 3: Low-end OS (e.g., RTOS) on MCU.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C3-1	System development that does not adhere to best security practices introduces risks	Absence of a strict and secure development process	Conduct penetration testing based on the best security practices	Manufacturers should strictly follow the best security practices during the development stage (e.g., AzureRTOS best security practice)
C3-2	Outdated OS versions (e.g., AzureRTOS, FreeRTOS) introduce potential risks	Absence of a strict and secure development process	Conduct penetration testing based on the best security practices	Manufacturers should ensure the OS kernel is up-to-date
C3-3	The RTOS responsible for in-vehicle communications (e.g., receiving and sending CAN messages) forwards crafted messages to the vehicle network	Absence of a protection mechanism for possible attacks	Conduct fuzz testing on the communication channel; perform injection attacks on the channel	Enable a whitelist mechanism to block malicious messages

TABLE 4: Code 4: High-end OS (e.g., Linux, Android) on MPU.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C4-1	Outdated OS versions introduce potential risks	Outdated OS version is applied and thus can be vulnerable	Examine the OS version	Manufacturers should ensure the OS kernel is up-to-date
C4-2	Absence of OS-level protection introduces potential risks	Absence of OS level protection	Conduct penetration testing on the related security modules in the MPU OS	Enable the OS-level security modules (e.g., SELinux, AppArmor)
C4-3	Absence of security measures for protecting inter-process communication (IPC) introduces risks	Absence of protection on IPC	Conduct penetration testing on process communication	Enable protections on IPC, including 1). use secure protocols to encrypt essential data; 2). enable access control mechanisms on the IPC resources, and 3). implement user and process isolation.
C4-4	Absence of security measures for protecting the data and resources stored in the OS	Absence of protection on data	Conduct penetration testing on the access control of the OS resources	Enable strict access control for critical applications and data
C4-5	Poorly-secured network settings introduce risks	Improper network settings	Conduct penetration testing on the network interfaces (e.g., attempt to monitor or manipulate the traffic via sniffed ports)	Properly configure the network settings (e.g., a well-configured *iptable* as the firewall)

TABLE 5: Code 5: Data related.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C5-1	Information leakage can occur when private data are not securely collected, stored, or transmitted	Absence of best practices and protection for private data	Conduct penetration testing on the collection, storage, and transmission of private data	Follow best security practices in line with up-to-date regulations (e.g., GDPR)
C5-2	The encryption mechanism can be cracked, leading to various attacks	Insecure encryption mechanism	Conduct penetration testing on the encryption mechanism and evaluate the likelihood of being attacked	Follow best security practices for robust encryption mechanisms, including 1). use strong encryption such as AES or RSA, 2). generate keys with secure random number generators, 3). securely store the keys (e.g., in a hardware security module - HSM), 4). update the keys when necessary
C5-3	There is a lack of comprehensive logging records, making forensic analysis difficult	Absence of necessary logging design in the development process	Extract the system log to evaluate its content (e.g., whether the contents are sufficient)	Enable system log function to provide evidence for forensic analysis
C5-4	System log contains improper information which could be a security threat once leaked to an attacker (e.g., OTA update log)	Improper design of the information collected by the system logs	Extract the system log to evaluate its content (e.g., whether the contents are over-collected)	1). Strictly restrict which data should be recorded and which should not; 2). implement measures to ensure that the log is securely stored (e.g., use encryption)
C5-5	Sensitive information (e.g., secret keys and private information) can be leaked when hardcoded in firmware or program	Developers lack security awareness	Conduct security testing on how sensitive information is stored, and evaluate corresponding risks of being leaked	Use additional methods to protect sensitive information, for example, HSM or encryption

TABLE 6: Code 6: In-vehicle Components: IVI

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C6-1	IVI applications, including the pre-installed ones and the third-party apps from the app market, have security flaws that can be exploited by attackers	Lack of following the best security practice for IVI app development	Penetration testing on the APPs installed on IVI	Best security practice on the development and security testing on the apps that can be installed on IVI
C6-2	"Developer mode" or "engineering mode" can be activated by particular actions (e.g., clicking particular area on screen multiple times), in which case attacker can exploit this function to analyze the system for various attacks	Unsafe backdoors are left in the developing process	Penetration testing the debug backdoors on the IVI system	1). Disable the developer mode or super-user mode function as possible; 2). Add authentication on the access to these high-privilege system modes
C6-3	Attacker sends malicious messages to IVI to trigger particular attacks (e.g., via Wi-Fi, Bluetooth)	Unsafe implementation on the IVI communication interface (e.g., unsafe software or protocol)	Penetration and fuzz testing on the IVI communication interfaces	Best security practice on 1). the communication protocol and interfaces, and 2). software program handling oncoming messages
C6-4	The audio control functions can be compromised by carefully-crafted signals (e.g., the dolphin attack)	Lack of security control on the audio signal received by IVI	Testing the robustness of the audio control system (e.g., by sending signals in various frequencies)	Enable strong authentication to increase the robustness of the audio recognition module
C6-5	The third-party apps that could be installed from the APP market in the IVI have security flaws that can be exploited	Lack of strict supervision of the application published on the APP market	Perform security testing on the applications that can be installed from the IVI APP market	Perform deep security analysis on the apps that can be installed in IVI; Restrict the privilege of apps from the OS level, to reduce risks even when one app becomes vulnerable
C6-6	Attacker can install a malicious app in the IVI via particular interfaces	Lack of protection on the APP installation process in IVI	Check whether IVI allows user to install arbitrary application (e.g., malware)	IVI should enable a whitelist of legal applications, and ban the illegal application from being installed
C6-7	The implementation of the *hypervisor* module on IVI has security flaws that can be exploited	Unsafe implementation on hypervisor	Perform penetration testing on the hypervisor implementation	Best security practice for the hypervisor implementation, for example: 1). enable strict access control between different systems; 2). development process should follow strict security requirements
C6-8	Due to the lack of security access control, high security-level functions (e.g., car-control modules) can be accessed or activated by low security-level ones (e.g., infotainment modules)	The principle of least privilege (PoLP) is not followed	Try accessing the safety-critical functions with low-privilege components in IVI	The resources for *infotainment* functions and *car-control* functions should be strictly isolated (e.g., infotainment modules should not be able to influence the car-control modules)
C6-9	Services opened on particular ports introduce risks	Unsafe implementation on the system ports and the corresponding programs	Penetration and fuzz testing on the open ports and the programs listening on these ports	1). remove unused and unnecessary services and ports on IVI; 2). strict authentication should be applied on the neccessary ports on IVI
C6-10	The browser function, (including an intact browser that can visit arbitrary Internet website, and the browser modules (e.g., WebView) in IVI applications), can be exploited by the attacker	Unsafe implementation on the browser modules (e.g., low version of Webview introduces known CVEs)	Penetration testing on the browser modules used in IVI	Manufacturers make particular effort to ensure the security of the browser module (e.g., make sure the browser kernel is up-to-date and cannot be exploited by known CVEs); Enable process isolation to restrict the resources that can be accessed by infotainment functions (e.g., browser)
C6-11	The communication between the IVI and other clients can be compromised and thus leading to information leakage	Lack of encryption on the communication data	Capture the communication between IVI and other components to check whether the data are properly encrypted	Make sure critical data are communicating via secure channels (e.g., on HTTPS, MQTT, v2x, WIFI, Bluetooth, NFC, etc.).

TABLE 7: Code 7: In-vehicle Components: Telematics

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C7-1	The remote control service is exploited and thus leading to various attacks	Lack of security protection of the safety-critical control messages	Try various attacks for the control interfaces on Telematics (e.g., DoS attacks, spoofing attacks)	Strictly check the integrity and authenticity of the received control messages
C7-2	The remote diagnostic service is exploited and thus the attacker can send malicious diagnostic messages into the vehicle	Unsafe implementation on the remote diagnostic functions enabled on Telematics	Try various attacks for the diagnostic functions on Telematics (e.g., DoS attacks, spoofing attacks)	Strictly check the integrity and authenticity of the received diagnostic messages
C7-3	Attacker sends malicious messages to Telematics to trigger particular attacks (e.g., via Wi-Fi, BLE, SMS message)	Unsafe implementation on the various messages received on the communication channels	Penetration testing on the implementations on the communication channels	Best security practice on 1). the communication protocol and interfaces, and 2). software program handling oncoming messages
C7-4	Information leakage can happen if the Telematics improperly collect and update the user data to the backend server	Implementation does not follow the best practice to protect private data	Check the possible violation of privacy on the data collected and uploaded by Telematics	Best security practice on the data security involved in the Telematics (e.g., GDPR)
C7-5	The network communication on Telematics can be compromised by the particular devices (e.g., GSM fake station) set up by the attacker	Lack of protections against the attacks by particular devices set up by the attacker	Perform various attacks by setting up the GSM fake station (e.g., DoS or Spoofing attack)	Best security practice on the cellular communications and other channels
C7-6	The ports opened on Telematics for Cellular communication can be compromised by the attacker to launch various attacks	Lack of proper authentication on the Cellular implementation on Telematics	Penetration testing on the ports for Cellular services on Telematics	Enable authentication on the ports for Cellular communication services; Implement secure authentication such as two-way TLS

TABLE 8: Code 8: In-vehicle Components: Sensors

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C8-1	The implementation of the perceptions sensors (i.e., camera, LiDAR, and sonar) lacks the robustness and thus can be compromised or cannot meet the performance demand in extreme cases	Lack of implementations to ensure the robustness of the sensor signals	Testing the performance of the sensors under extreme cases	Make sure the implementation is up-to-date and robust; Give warning to the driver when any anomaly (e.g., possible attacks or performance downgradation) is detected
C8-2	The messages communicating with the tire pressure monitoring system (TPMS) can be compromised and thus leading to various attacks (e.g., eavesdropping and spoofing)	Lack of security protections on the communication channels for TPMS	Try various attacks for the TPMS system (e.g., Dos and Spoofing attacks)	Best security practice on the TPMS implementation
C8-3	The GNSS sensors can be exploited and thus the attacker can affect the related vehicular functions (e.g., navigation and autonomous driving)	Lack of implementations to ensure the robustness of the GNSS signals	Try various attacks for the GNSS sensors (e.g., Dos and Spoofing attacks)	Best security practice on the GNSS sensor implementation

TABLE 9: Code 9: In-vehicle Components: Gateway and Zone Controller

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C9-1	Attackers can exploit the improper forwarding rules to perform cross-domain attacks by sending crafted packets to get access control	Security flaw exists in the forwarding process	Penetration testing on the forwarding rules of the gateway and zone controller	Gateway should forward messages based on the strict and correct CAN Communication Matrix (for CAN interface) and Access Control List (ACL) for Ethernet interface, and discard the illegal packets
C9-2	Lack of authentication on critical packets (e.g., control commands) gives attackers the chance to control ECUs by replay attack	Unsafe implementation on the critical commands sent to gateway	Try to send control messages to the gateway to compromise ECUs on other domain	Authentication should be applied in critical commands or services
C9-3	Attackers sends crafted packets to the particular gateway interfaces (i.e., CAN and Ethernet interface) to launch attack on the protocol implementation	Lack of protections on the known attacks for the protocol implementation	Perform penetration testing on the protocol implementation	Security measures should be implemented to prevent attacks on CAN and Ethernet interfaces. For example, for 1). CAN interfaces, replay attacks, eavesdropping attacks, DoS attacks, scanninng attacks should be considered; for 2). Ethernet interfaces, classic attacks including port/IP scanning, ARP/IP spoofing, UDP/ICMP flooding attacks should be considered.
C9-4	The FOTA function of the Gateway can be compromised and thus leading to various attacks	Unsafe implementation for the FOTA function in gateway	Penetration testing on the FOTA function enabled on gateway	Gateway is the essential ECU responsible for the firmware update of other ECUs, and its FOTA implementation should be carefully designed to prevent possible risks, for example: 1). check the firmware integrity and authenticity; 2). transmit the firmware in encrytion

TABLE 10: Code 10: In-vehicle Components: Advanced driver-assistance system (ADAS)

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C10-1	The real-time performance of the ADAS system cannot be met and thus introducing risks	Lack of security implementation to ensure the real-time performance	Penetration testing on the real-time performance	Implementations should be used to ensure the real-time performance of the ADAS system, which is a safety-critical system (e.g., Time-Sensitive Networking)
C10-2	Lack of check on integrity and authenticity of the messages communication with ADAS, and thus attacker can launch spoofing or replay attacks by sending crafted messages	Unsafe implementation on the ADAS communications	Perform various attacks on the communication channels on ADAS (e.g., DoS and Spoofing)	Add mechanism to ensure the integrity and authenticity of the messages, for example: 1). check the timestamp of the control messages; 2). authenticate safety-critical requests (e.g., change system parameter)
C10-3	Adversarial attacks can be launched against the sensors (e.g., camera and LiDAR) on ADAS	Lack of protection to identify adversarial attacks	Perform adversarial attacks on the ADAS systems (e.g., black-box attacks on camera and LiDAR perception)	Mitigations againt such attacks should be considered, for example: 1). use multiple sensors for one type of perception (e.g., multiple cameras); 2). use multi-sensor fusion to reduce the risk of being attacked; 3). use robust machine learning methods
C10-4	The control policy of the ADAS have security flaws which introduces risks (e.g., dangerous driving in complex scenario such as crossroad)	Security flaws exist in the control program of ADAS	Penetration testing on the control programs of ADAS	1). build robust control policy which considers dangerous corner cases; 2). make the human driver take over anytime when ADAS cannot determine current situation
C10-5	The algorithms used in ADAS can be stolen and thus causing product piracy, or attacker can analyze the algorithms to launch further attacks	Lack of protection on the code and algorithms stored in ADAS	Extract the code and algorithms in ADAS	The algorithm code should be safely stored in ADAS (e.g., using encryption or obfuscation)
C10-6	Personal data related to ADAS can be leaked or improperly collected by the manufacturer	Failed to follow best security practice on protecting private data	Extract the private data stored in ADAS	1). Manufacturers should clearly notify the user what data are being collected and why; 2). Personal data cannot be collected without user permission; 3). Personal data should be stored and transmitted with encryption

TABLE 11: Code 11: In-vehicle Components: In-Vehicle Network (IVN)

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C11-1	Attackers can compromise other in-vehicle nodes or network segments once she has controlled one of them, due to the lack of network separation mitigations	Lack of isolation on the IVN design	Penetration testing on the access control of the nodes (i.e., ECUs) on IVN	Security measures should be implemented to securely segment the IVN, for example: 1). using VLAN access control technique; 2). restrict in-vehicle access with passwords (e.g., ban unrestricted *ssh*); 3). avoid using unsafe communication protocols (e.g., Telnet, FTP)
C11-2	Attackers can listen on the bus (e.g., CAN or Ethernet) to steal critical information	Lack of encryption on the data transmitted in IVN	Extract communication data on IVN	Critical data (e.g., secret keys, FOTA firmware, private data) should be encrypted before they are transmitted on the bus
C11-3	Message injection attacks can be launched on the IVN	Lack of protection on the injection attacks on IVN	Perform various injection attacks on IVN (e.g., DoS, spoofing, replay attacks)	Implementation should be used to protect IVN from injection attacks.
C11-4	The network forwarding process has security flaw that can be exploited to reach other in-vehicle nodes	Unsafe implementation on the network forwarding rules in IVN	Penetration testing on the access control in IVN	Forwarding rules should be safely implemented to prevent unauthorized access

TABLE 12: Code 12: In-vehicle Components: Battery Management System (BMS)

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C12-1	The Battery management system (BMS) can be attacked via the charging port (e.g., information leakage, spoofing attacks), due to the unsafe design or lack of protection	Lack of protection on possible attacks launched from charging port	Penetration testing on the charging port to compromise the BMS	Apply best security practice on BMS, for example: apply secure authentication on the messages comming from the charging port

TABLE 13: Code 13: In-vehicle Components: Other ECUs

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C13-1	Other ECUs does failed to follow best security practice and thus leading to possible attacks	Failed to follow best security practice	Penetration testing on other ECUs according to best security practice	Apply best security practice on other ECUs

TABLE 14: Code 14: Outside-vehicle components: Mobile App.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C14-1	APP development does not follow common best security practice	Lack of implementation of app security best practice	Penetration testing on the common best security practice	Best security for APP development should be followed, including ensuring right permissions, safe data storage, secure communication (e.g., best security practice for Android development [6])
C14-2	Communication between the mobile APP and other clients (e.g., server or vehicle) can be compromised to launch further attacks (e.g., eavesdropping, spoofing)	Unsafe implementation on the communication channels	Penetration testing and fuzz testing on the communication channels of the mobile apps	Best security on the specific communication channel (e.g., UWB, NFC and BLE)
C14-3	Attacker can reverse-engineer the App to obtain critical information to launch further attacks	Lack of protection on the app code	Extract the app installation package and perform testing accordingly	Enable implementations to prevent the app from being reverse-engineered (e.g., obfuscation, code encryption)
C14-4	Sensitive data (e.g., private data, encryption/decryption code, secret keys) can be leaked from the mobile app side if not properly handled	Unsafe implementation for data transmission on mobile app	Penetration testing on the data transmission process enabled by mobile app	Enable implementations to prevent the data security (e.g., code encryption and obfuscation; prevent the data from being accessed by other apps)
C14-5	The high-privilege interface or module in the mobile app (e.g., the vehicle control APIs) is exposed to other apps and thus can be exploited to launch various attacks	Failed to follow the principle of least privilege (PoLP) in app development	Testing to access the high-privilege APIs with low-privilege modules	The developer should strictly restrict the safety-critical APIs from being accessed from other process in the mobile

TABLE 15: Code 15: Outside-vehicle components: Backend Server.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C15-1	The server setting does not follow common best security practice	Failed to follow the best security on server implementation	Security testing according to the common best security practice	Best security for backend server security should be followed (e.g., ISO 27017 [8])
C15-2	Communication between the backend server and other clients (e.g., mobile app or vehicle) can be compromised to launch further attacks (e.g., eavesdropping, spoofing)	Unsafe implementation on the data transmission between server and other clients	Penetration testing on the data transmitted from and to the backend server	Best security on the specific communication channel
C15-3	Attacker can launch the DoS attack on the server via particular communication channels	Lack of DoS protections on the particular server interfaces and communication channels	Testers can send a large amount of data on the communication channel to launch DoS attacks	Implementations to prevent DoS attacks, including: 1). setting up firewalls and other filtering mechanisms; 2). detections to identify traffic bursts; 3). a quick recovery plan when DoS attacks happen
C15-4	Attacker can steal the user credentials via Credential Stuffing Attacks on the server	Lack of protections to defend credential stuffing attacks	Try to stuffing the user credentials based on known week passwords	Implementations to prevent credential stuffing attacks, including: 1). enable multi-step authentication and CAPTCHA; 2). implement detectors to detect abnormal traffic and login requests
C15-5	Attacker can perform injection attacks on the server (e.g., SQL injection, cross-site scripting)	Lack of protections to prevent injection attacks	Testers can perform penetration testing on the server interfaces to discover possible injection attacks	Server should carefully check the format and content input data to filter malicious message
C15-6	Personal information can be leaked due to the improper storage of data on server	Data stored on server are in plaintext or not properly encrypted	Penetration testing on the accessibility of the on-server data	Data on server, especially personal data, should be stored and transmitted after encryption
C15-7	Due to the design flaw of the authentication process, the attacker can perform illegal actions on other customers' cars with one valid account (e.g., stealing information or even control other cars)	Security flaw exists in the server authentication process	Perform penetration testing on the authentication process enabled by backend server	1). Adhere to the principle of data minimization, only collecting necessary user data. 2). Ensure data encryption and anonymization. 3). Implement strict access control for data stored on server.

TABLE 16: Code 16: Outside-vehicle components: Charging pile.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C16-1	The authentication implemented on the charging pile has security flaws and thus introducing risks (e.g., allow attackers to steal the electrical energy without payment)	Unsafe implementation on the authentication for charging servcie	Penetration testing on the security of charging service	Apply strong mutual authentication on the charging service
C16-2	The vehicular data is leaked to the third-party charging service provider	Unsafe implementation on the charging access	Penetration testing on the security of charging service	1). restrict the data that can be accessed from the vehicle charging port; 2). the charging pile itself and the corresponding backend server should follow the best security practice

TABLE 17: Code 17: Communication protocols: Short-range communication protocols: UWB, NFC, and BLE.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C17-1	Eavesdropping: attacker monitors the communication channel with particular equipments, and thus stealing essential information	Lack of encryption on the transmitted data on the channel	Perform eavesdropping on the channel trying to steal transmitted data	Enable encryption on the transmitted data
C17-2	DoS or Jamming attacks: attacker spreads a large amount of garbage messages to jam the communication channel	Lack of protection for DoS attack on the channel	Perform DoS attack on the channel	Implementations to increase the robustness of the communication (e.g., frequency hopping)
C17-3	Spoofing / MITM / Replay attack: attacker captures and replay the messages to control the vehicle (e.g., open the door)	Lack of protection to prevent possible injection attacks	Perform the injection and replay attacks on the channel	Enable encryption and strong mutual authentication on the transmitted data
C17-4	Best security practice is not followed	Failed to follow the best security practice for the common protocol implementation	Perform penetration testing according to the best security practice	Perform best security practice for each protocols (e.g., secure NFC [3], LE secure connection [5])
C17-5	The version of the implemented protocol itself is out-of-date (low version of BLE) and thus introducing vulnerabilities on historical versions	Failed to implement the up-to-date protocol	Check whether the version of the implemented protocol is up-to-date	Developers should frequently update the version of the protocols themselves

TABLE 18: Code 18: Communication protocols: V2X.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C18-1	The V2X communication can be compromised and thus leading to various attacks (e.g., DoS, sybil, spoofing attacks)	Failed to follow the best security practice for V2X implementation	Perform security testing according to the best security practice on V2X implementation	The security of V2X messages should be established through the Public Key Infrastructure (PKI), and every client capable of sending messages must be certified and carry a valid certificate, and every message they send must carry a signature from their private key, thereby validating their legitimacy as the sender.

TABLE 19: Code 19: Communication protocols: In-vehile CAN.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C19-1	Eavesdropping: attacker monitors the CAN communication channel with particular device attached on the IVN	Lack of encryption protection on the CAN data	Perform eavesdropping attacks on the CAN bus	Enable encryption on critical data (e.g., when updating a private key) transmitted on CAN bus
C19-2	DoS or Jamming attacks: attacker spreads a large amount of garbage messages to jam the CAN bus	Lack of protections against DoS attacks on CAN bus	Perform DoS attacks on the bus to test the robustness	Implementation to increase the robustness against DoS attack (e.g., IDS and other detection mechanisms)
C19-3	Spoofing / MITM / Replay attack: attacker captures and replay the messages to control particular ECUs on CAN bus	Lack of protection to prevent possible injection attacks	Perform the injection and replay attacks on the CAN bus	Enable encryption on critical data (e.g., when updating a private key) transmitted on CAN bus
C19-4	Attacks exploiting the intrinsic nature of CAN protocols (e.g., bus-off attack exploiting the CAN error handling mechanism [11])	Lack of protection to prevent attacks exploiting the protocol features	Perform penetration testing on the CAN bus implementation	Perform best security practice for the emerging attacks (e.g., up-to-date IDS)

TABLE 20: Code 20: Communication protocols: In-vehicle Ethernet.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C20-1	Eavesdropping: attacker monitors the Ethernet communication channel with particular device attached on the IVN or a compromised ECU	Lack of encryption protection on the data transmitted via Ethernet	Perform eavesdropping attacks on the Ethernet communication	Enable encryption on critical data transmitted on Ethernet
C20-2	DoS or Jamming attacks: attacker spreads a large amount of garbage messages to jam the Ethernet communication channel	Lack of protections against DoS attacks on Ethernet	Perform DoS attacks on the bus to test the robustness	Implementation to increase the robustness against DoS attack (e.g., IDS and other detection machenisms)
C20-3	Spoofing / MITM / Replay attack: attacker captures and replay the messages to control particular ECUs on Ethernet	Lack of protection to prevent possible injection attacks	Perform the injection and replay attacks on Ethernet	Enable encryption on critical data transmitted on Ethernet
C20-4	Lack of network segment isolation, and thus the attacker can compromise other ECUs from one ECU (e.g., via unauthorized *ssh* or *Telnet* communication)	Lack of design to isolate ECUs on the Ethernet implementation	Penetration testing on the access control enabled on Ethernet implementation	Enable strict isolation mechanism for different domains (e.g., buses connected by gateway ECU or zone controller)
C20-5	Then time-sensitive design (TSN) does not meet demands and thus can be exploited by attacker [14]	Lack of protection to ensure the real-time performance of Ethernet (e.g., for ADAS)	Testing the real-time performance of Ethernet	Implementations to ensure the real-time performance of real-time performance
C20-6	Attacks exploiting the intrinsic nature of Ethernet protocols (e.g., ARP spoofing and MAC spoofing attcks)	Lack of protection to prevent attacks exploiting the Ethernet features	Perform penetration testing on the Ethernet implementation	Perform best security practice for the classic and emerging attacks on Ethernet (e.g., MAC spoofing, ARP spoofing)

TABLE 21: Code 21: Communication channel/interface: Wireless channels - Wi-Fi, Bluetooth, and Cellular.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C21-1	Best security practice on Wi-Fi service is not followed, and thus introducing risks	Failed to follow best security practice	Perform security testing according to the best security practice	Follow the best security practice on Wi-Fi, for example: 1). enable WPA3 encryption; 2). enable strong password; 3). follow the NIST standard [17])
C21-2	Attacker controls the vehicular traffic by triggering the vehicle to connect to malicious Wi-Fi AP (e.g., faking the 4S shop service Wi-Fi that the vehicle will automatically connnect to)	Unsafe implementation on the Wi-Fi network setting	Perform penetration testing on the Wi-Fi implementation on the system	1). Enable mutual authentication when connecting to service Wi-Fi; 2). restrict in-vehicle resources that can be accessed by Wi-Fi traffic; 3). critical data should be transmitted after encryption
C21-3	Attacker sends malicious messages via the Wi-Fi interface to the vehicle	Lack of implementation to filter the malicious messages on Wi-Fi	Penetration and fuzz testing on the Wi-Fi interface	Firewall mechanism should be properly set (e.g., *iptable*) to filter the malicious messages on Wi-Fi interface
C21-4	Best security practice on classic Bluetooth service is not followed, and thus introducing risks	Failed to follow best security practice	Perform security testing according to the best security practice	Follow the best security practice on Bluetooth (e.g., follow the NIST standard [16])
C21-5	Best security practice on Cellular network is not followed, and thus introducing risks	Failed to follow best security practice	Perform security testing according to the best security practice	Follow the best security practice on Cellular (e.g., follow the NIST standard [12])

TABLE 22: Code 22: Communication channel/interface: Physical - Charging port.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C22-1	Attacker can monitor the in-vehicle traffic via the charging port to steal critical data	Lack of data protection on the charging port	Monitor the data transmission on the charging port	1). Restrict the data that can be accessed by passively listening on the charging port; 2). Enable encryption on critical data transmitted from the charging port
C22-2	Attacker can inject malicious data via charging port to actively request in-vehicle data or modify the parameters of the BMS	Lack of protection for injection attacks on charging port	Penetration testing on the charging port to identify possible attacks	Authentication is required for the critical requests from the charging port (e.g., request data, launch the charging)
C22-3	Security flaws exist in the authentication process in communication on charging port	Unsafe authentication is implemented	Penetration testing on the authentication process enabled on charging port	Best security practice on the authentication process on charging port (e.g., strong mutual authentication)

TABLE 23: Code 23: Communication channel/interface: Physical - USB and SD card.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C23-1	Unused or verbose external physical interfaces introduce risks (i.e., USB or SD card slot)	Verbose physical interface introduces risks	Penetration testing on the physical interface	Remove unused physical interfaces to reduce the possibilities of being attacked via these interfaces
C23-2	Attacker compromises the system via a malicious device connecting to the physical interfaces	Lack of authentication on the data transmitting from USB or SD card	Perform injection attacks via the physical interface	The system should authenticate critical data coming from the external physical interfaces (e.g., updating data from USB)
C23-3	Vehicle services enabled via the USB port (e.g., Apple Carplay and Android Auto) has security flaws and thus introduces risks	Failed to follow best security practice on the service implementation	Penetration testing on the services enabled on USB port	1). Follow best security practice in development stage to prevent risks; 2). Frequently update the service version to ensure the security

TABLE 24: Code 24: Communication channel/interface: Physical - OBD port.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C24-1	Attacker can monitor the in-vehicle traffic by a device attached on OBD port to steal critical data	Lack of data protection on the OBD port	Penetration testing by monitoring the data transmission on the OBD port	Restrict the data that can be accessed by passively listening on the OBD port
C24-2	Attacker can inject malicious data via OBD port to actively request in-vehicle data or modify the parameters of the ECUs	Lack of authentication on the requests from the OBD port	Penetration testing by sending requests from the OBD port to access in-vehicle data or modify vehicular parameter	Authentication is required for the critical requests launched by the OBD device (e.g, read private data and change ECU parameter)
C24-3	Security flaws exist in the authentication process in OBD communication	Unsafe implementation on the authentication process on OBD communication	Penetration testing on the authentication process on OBD	Best security practice on the authentication process in OBD communication
C24-4	The third-party OBD device introduces additional risks to the vehicle system	Lack of protection to prevent risks introduced by third-party OBD device	Penetration testing on the OBD from the perspective of third-party device	The manufactures should restrict the resources that can be accessed by an untrusted third-party OBD device, for example: 1). critical data should not be accessed from the OBD port without authentication; 2). vehicle parameters cannot be changed without authentication; 3). control commands cannot be executed without authentication.

TABLE 25: Code 25: Vehicular function/services: OTA update.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C25-1	Attacker can illegally obtain the update image for further analysis	Lack of protection on the access control of the update image	Penetration testing on the access control of the update image (e.g., try to download the image from particular URL)	Update image should be transmitted in secure communication protocols (e.g., HTTPS), and itself should be encrypted
C25-2	Attacker can hijack the communication between the vehicle and server (e.g., fake the vehicle to illegally download firmware, or fake the server to distribute crafted firmware)	Lack of proper authentication to ensure the secure transmission of the update image	Testing by trying to tampering with the communication channels for OTA	Secure mutual authentication should be implemented during OTA process
C25-3	DoS attacks can happen during OTA process	Lack of mechanism to prevent DoS attacks	Testing by Launching DoS attack on the OTA process	Deploy mitigations to prevent DoS attacks (e.g., add validation on the timestamp and version to prevent rollback attacks)
C25-4	The vehicle installs incomplete or crafted firmware, due to the unsafe integrity and authenticity validation	Unsafe implementation on the authentication process of the downloaded firmware	Penetration testing on the authentication process of the ECU	The vehicle should perform extensive examination on the integrity and authenticity of the downloaded firmware (e.g., using strong hash function checksum and digital signatures)

TABLE 26: Code 26: Vehicular function/services: Diagnostic.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C26-1	Attacker can read private data from the ECUs via launching the diagnostic services (e.g., [1])	Lack of authentication on the diagnostic request for critical in-vehicle data	Testing by sending diagnostic requests to access in-vehicle data	Manufacturers should strictly restrict the range of data that can be accessed by diagnostic services, and essential data (e.g., private data, secrect keys) cannot be accessed without proper authentication
C26-2	Attacker can perform replay attack by sending crafted diagnostic messages to target ECUs	Lack of authentication on the diagnostic messages	Testing by sending diagnostic requests to perform replay attacks	ECUs should authenticate the received diagnostic messages, especially those for safety-critical controls (e.g, unlock car doors, folding rearview mirror)
C26-3	Attacker can manipulate the parameters within the ECUs with crafted diagnostic messages	Lack of authentication on the diagnostic messages	Testing by sending diagnostic requests to modify vehicular parameters	Manufacturers should strictly restrict the power of diagnostic messages to modify the ECU parameters. For example, critical parameters cannot be modified without authentication
C26-4	Attacker can compromise the remote diagnostic services if the service is not securely implemented	Lack of mechanisms to protect the remote diagnostic functions	Penetration testing on the remote diagnostic functions enabled by the vehicle (e.g., on Telematics)	Enable strong security implementations on remote diagnostic services (e.g., implementing TLS or SSL encryption for DoIP diagnostic)

TABLE 27: Code 27: Vehicular function/services: Remote monitor and control.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C27-1	Unsafe implementation on the communication channel or protocols for remote functions	Failed to follow best security practice on the communication channel	Penetration testing on the protocol implementation	Best security practice for specific implementation (e.g., BLE protocol)
C27-2	Attacker can perform MITM attack to steal critical information transmitted on the channel, or replay the captured message to launch attack afterwards	Lack of implementation to prevent replay attacks on the remote functions	Testing by launching replay and spoofing attacks to affect the remote functions	1). Ensure data are transmitted via a secure channel (e.g., two-way TLS). 2). Critical information should be encrypted in transmission. 3). Timestamp check should be implemented to prevent replay attack.
C27-3	The implementation of the digital key has security flaws that can be exploited by the attacker	Unsafe implementation on the secret keys involved in authentication	Penetration testing on the security of the secret key implementation	1). Digital keys should be securely stored (e.g., encrypt the key itself, or use Hardware-backed Keystore [7]); 2) digital keys should be strong enough to prevent it from being brute-forced; 3). digital keys should be frequently updated by the backend server
C27-4	The encryption / decryption algorithm involved has security flaws that can be exploited by the attacker	Unsafe encryption is used	Penetration testing on the authentication process	1). The authentication codes should be protection by encryption or obfuscation and cannot be easily accessed; 2). strong authentication algorithms should be used (e.g., AES and RSA); 3). use secure random number generators (e.g., use hardware-based true random number generator to generate unpredictable random numbers)

TABLE 28: Code 28: Others.

Number	Attack Description (AD)	Root Cause (RC)	Security Testing Approach (STA)	Mitigation (MG)
C28-1	Unsafe human action can introduce risks to the vehicular system (e.g., misled by the malicious page to install a malware in IVI)	Lack of warnings of the risky actions possibly done by the car owner	Testing the possible risky actions that can be performed by the car owner	1). Restrict the user privilege; 2). Well inform the user of the possible risks of dangerous actions.
C28-2	Insider attacks can happen and thus bringing loss to the manufacturer company	Failed to follow the best security practice for management	Checking the management system of the automotive company	Strictly restrict the resources that can be accessed by the employees, and follow the best security practice of developing process (e.g., ISO 21434 [2])
C28-3	Some safe actions when the vehicle is parked can be the dangerous action when the vehicle is running (e.g., playing video on IVI leading to distractive driving; diagnostic controls)	Lack of comprehensive consideration on the status of the vehicle	Testing by performing dangerous actions while the vehicle is moving (e.g., sending diagnostic control messages)	The vehicle should be *state-aware* to prevent dangerous actions when it is running