Formally Verified Software Update Management System in Automotive

Jaewan Seo
Korea University
bace@korea.ac.kr

Jiwon Kwak
Korea University
jwkwak4031@korea.ac.kr

Seungjoo Kim*
Korea University
skim71@korea.ac.kr

***Abstract*—Through wireless networks, the number of cyberattacks on automotive systems is increasing. To respond to cyberattacks on automotive systems, the United Nations Economic Commission for Europe (UNECE) has enacted the UN Regulation series. Among them, UN R156 specifies the requirements that are necessary for the design and implementation of a software update management system (SUMS). However, the requirements of UN R156 are too abstract to develop the overall systems of SUMS. Therefore, we conducted threat modeling to obtain more specific security requirements than those specified in the UN R156. Based on the threat modeling, we proposed a secure SUMS architecture that meets specific security requirements. Finally, we formally verified whether our SUMS architecture logically meets the security requirements by Event-B.**

I. Introduction

To provide convenience, original equipment manufacturers (OEMs) apply many wireless communication technologies to vehicles. Based on the technology, vehicles are providing various services such as over-the-air updates as well as entertainment [1,2]. Many OEMs actively provide over-the-air services to their clients. For example, Hyundai, a representative OEM in the automotive industry, provides various services to its clients in concert with Amazon [3]. However, as more software and hardware are built into or connected to vehicles, cases of vehicle hacking are also steadily being reported. In 2015, Miller and Valasek announced at Black Hat that vehicle hacking could endanger the life of the driver [4]. In 2017, researchers at Keen Security Lab announced that they could take control of the entire automotive system by forging Tesla firmware [5]. Recently, at the CanSecWest 2021 conference, a research team announced that hackers can arbitrarily manipulate confidential data in an automotive system using the wireless communication vulnerability of the vehicle [6]. In 2021, UNECE issued regulations to ensure the cybersecurity of vehicles. Among them, UN Regulation No. 156 (UN R156) [7] specifies security requirements for a SUMS. However, the security requirements in UN R156 are abstract, and thus it is difficult for OEMs to develop a SUMS. Therefore, in this study, we analyzed the SUMS architecture using a threat modeling technique [8]. Based on the results, we proposed security requirements that make SUMS secure and designed secure SUMS using those security requirements. Finally, we used Event-B, one of the formal methods, to prove that our design satisfies the security requirements. The remainder of this study is structured as follows. In Section II, we describe other research results that have topics that are similar to our research. In section III, we describe our research methodology. In Section IV, we describe the results of threat modeling, including a list of security requirements and a secure SUMS architecture. In Section V, we describe how our architecture mathematically meets the security requirements using Event-B. Finally, in Section VI, we summarize the results of our research.

* Corresponding author

Symposium on Vehicles Security and Privacy (VehicleSec) 2023
27 February 2023, San Diego, CA, USA
ISBN 1-891562-88-6
https://dx.doi.org/10.14722/vehiclesec.2023.23087
www.ndss-symposium.org

II. Related Works

In 2018, Lee analyzed the factors that caused risks in the software OTA update system. In addition, they proposed organizational and technical requirements to be required to handle the risks [9]. In 2020, Placho et al. proposed an automotive update process. To achieve this, they proposed organizational and technical requirements for securing the update process [10]. Hamad et al. analyzed threats related to vehicles. And they proposed security and privacy requirements to mitigate them [11]. Yu et al. proposed the security requirements for communication systems in vehicles such as Controller Area Networks(CAN) and Automotive Ethernet(AE) [12]. In 2021, Mukherjee et al. defined a threat model for automotive OTA systems and proposed security-enhanced OTA systems to mitigate the threats [13]. Ponsard et al. announced several cyberattacks which could be exploited on vehicles by hackers. And they analyzed the effectiveness of the existing update framework against cyberattacks from a security perspective and proposed the Upkit protocol, which can partially mitigate these cyberattacks [14]. Wu et al. identified threats to OTA technology and suggested countermeasures [15]. In 2022, Ghosal et al. proposed a threat model for systems providing OTA services when using cloud servers. They also suggested security requirements for securely updating their software through cloud servers [16]. Kirk et al. surveyed specific attack methods that can be used to exploit OTA systems [17].

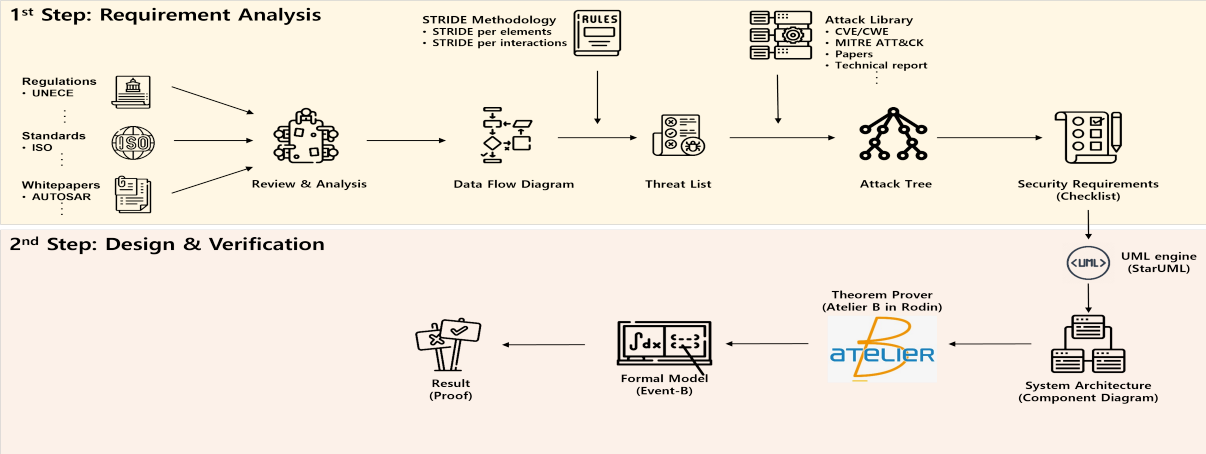

Fig. 1. Overview of our research

Among the nine studies, four proposed security requirements for systems which is related to software updates in the automotive industry. And only one study, conducted by Ghosal et al., verified their proposal by simulation. Therefore, we concluded that the aforementioned studies have two limitations. First, they did not meet the UN R156 regulation completely. When conducting their studies, they only analyzed parts of the vehicle that they were interested in. It prevented their solution from meeting UN R156 regulation completely because UN R156 regulation covered whole parts of software update systems in the vehicles. Second, they verified their solutions by a simulation-based testing method instead of formal methods. The simulation-based testing method can evaluate whether a solution can be valid depending on the number of test cases and scenarios. This prevents the method from guaranteeing the solution's validity because the number of test cases and scenarios to consider is limited. To overcome the aforementioned two shortcomings, we conducted our study by using two detailed techniques. First, we used a threat modeling technique to derive specific security requirements. During threat modeling, we checked whether our result completely meet UN R156 regulation. After threat modeling, we proposed security requirements and designed SUMS architecture that meets our requirements. Finally, we verified that our SUMS architecture accurately reflects the security requirements using formal methods. All results of threat modeling and formal specification for SUMS can be found at the following URL (https://github.com/HackProof/Secure-Software-Update-Management-System-SUMS).

III. Overview of our research

In this section, we describe the overall process of our research. Our research process is categorized into two parts. The first step is to derive security requirements for SUMS, and the second step is to design the SUMS based on the requirements. Figure 1 shows the process of our research.

A. Deriving security requirements for SUMS

Security vulnerabilities and threats can be easily found in poorly developed software, and these can lead to exploitation by attackers. Inappropriate or insufficient security requirements are one of the reasons that downgrades the security or quality of software [53]. This problem can be solved by taking security concerns into account from the early phases of the software development process. In the automotive industry, the ISO 21434 standard which covers the cybersecurity of vehicles recommends the usage of systematic methodology such as threat modeling when deriving requirements [54]. One of the methodologies for doing this is threat modeling. Threat modeling is a specific technique to identify not zero-day but known threats, attacks, and vulnerabilities and propose countermeasures that could make systems secure [55-58]. A typical case of threat modeling is Windows VISTA. Microsoft used threat modeling while developing Windows VISTA and SQL servers. As a result, a lot of vulnerabilities and threats are reduced [59-61]. Due to its effectiveness, threat modeling has been used in many fields, including the automotive industry. In ISO 21434, there are some recommendations to derive requirements by using threat modeling techniques such as EVITA, TVRA, PASTA, and STRIDE. As a result, many automotive manufacturers use these techniques. A typical example is the HEAVENS security model. This model identifies threats based on Microsoft's STRIDE and evaluates the risk level by calculating TL (Threat Level) and IL (Impact Level) for each threat. When identifying a threat to a vehicle, this HEAVENS Security Model not only considers the inside of the vehicle, but also considers surrounding factors such as OEM, vehicle, and driver. In addition, the GENIVI Alliance, which consists of various vehicle manufacturers/suppliers around the world, such as BMW, DAIMLER, HONDA, and Hyundai, is also using STRIDE to identify attack routes and evaluate the risk for vehicles. So we also used the STRIDE threat modeling technique to derive security requirements because it is widely used in the automotive industry. The specific threat modeling process is as follows:

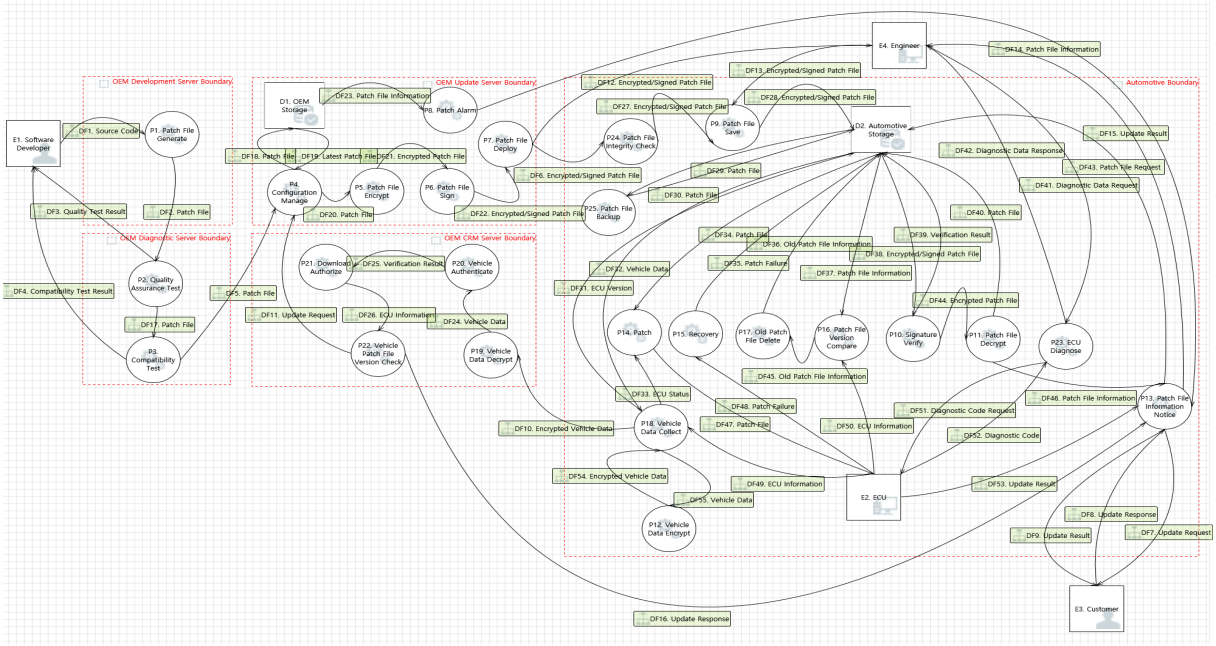

Fig. 2. Level 1 DFD model of SUMS architecture

- **Step 1. Draw data flow diagram (DFD):** To identify system behavior and vulnerability, we drew a DFD based on documents published by automotive organizations.
- **Step 2. Constructing an attack library:** To identify known vulnerabilities and attack techniques that can be used in a general SUMS, we collected them through a national vulnerability database, technical papers, and other sources.
- **Step 3. Analyzing threats in STRIDE:** To identify threats in general SUMS, we analyzed the DFD using the STRIDE method. Based on the analysis results, we derived the threats.
- **Step 4. Generating an attack tree:** We created an attack tree to obtain an attack scenario using the potential threats and vulnerabilities that were identified.
- **Step 5. Deriving security requirements:** Security requirements are derived by detailing items that can mitigate vulnerabilities associated with the lowest nodes of the attack tree.

B. Designing SUMS architecture based on requirements

A complex system, including automotive has numerous requirements that are challenging to fully implement in a single design. Furthermore, when designing the system via natural language, which means informal, misunderstanding can occur among developers. So ISO 26262 recommends using the (semi-) formal method while designing automotive [62]. So we designed our SUMS architecture in two stages. First, we design our architecture via UML. Then, corresponding natural language-based requirements are reflected on the UML component diagram. After completing the UML-based design, we refined the design using Event-b. In this stage, the design increases in detail and becomes closer to the implementation.

IV. Proposal for Secure SUMS Architecture

In this section, we describe the threat modeling process on SUMS and propose a secure SUMS architecture. To conduct a threat modeling process, first of all, we should model the target system. In the case of us, the target system is SUMS. However, there are few SUMS implementation models available to the public. This made it difficult to conduct threat modeling, because there were no specific targets for the analysis. Thus, we identified the specific components and functions of the SUMS architecture by analyzing documents of automotive organizations and project results before modeling the SUMS. Based on these documents, we conducted threat modeling.

A. Conducting threat modeling

1) Drawing Data Flow Diagram(DFD): DFD is an abstract model of the system to be analyzed in terms of data flow. It shows the flow of data through 5 components: entity, process, data store, data flow, and trust boundaries. When drawing a DFD, soundness, which is the property in which the DFD should be applied equally to the actual system, is important. Therefore, the DFD should be expressed in as much detail as possible such that there are no differences between the target source code and the drawn DFD model. The DFD model is divided into four levels. The first is the contextual model called the context diagram. This expresses the entire system based on roles such as the OEM, client, and auto repair shop. The second is the Level 0 DFD model, which decomposes its role based on the objective of its components, such as development, updates, file self-diagnostics, and customer management. The

third is the Level 1 DFD model, which decomposes its components based on their functionalities such as encryption, signing, and so on. The fourth is the Level 2 DFD model, which decomposes its functionalities based on further criteria [18- 20]. However, because Level 2 DFD is expressed in great detail, such as the functional level based on the source code, we were limited in drawing Level 2 DFD. Therefore, we created DFDs up to Level 1 by identifying essential modules based on documents. Figure 2 shows the DFD Level 1 model for the SUMS architecture.

2) Constructing attack library: After modeling the system, we collected all known vulnerabilities and security threats which is related to the SUMS architecture. These collected vulnerabilities and security threats are called attack library. This attack library includes common vulnerabilities and exposures (CVE), common weakness enumeration (CWE), papers, technical documents, and so on. In our case, the attack library includes a total of 70 pieces of vulnerabilities and security threats. Specifically, it is divided into 35 CVE cases, 6 CWE cases, 27 papers, 1 standard case, and 1 technical report related to the SUMS; their contents are briefly shown in Table I.

TABLE I
ATTACK LIBRARY FOR SUMS

No.	Category	Name	Ref.
AL-V-1	CVE	CVE-2010-2959	[21]
AL-V-2	CVE	CVE-2010-3874	
...	...	...	[22]-[47]
AL-P-28	Standard	ITU-T X.1373	[48]
AL-P-29	Technical Report	Automotive Industry Guidelines for Secure Over-the-Air Updates	[49]

3) Analyzing threats by STRIDE: Threat analysis is the process of identifying all potential threats that may exist in each DFD component. The STRIDE technique used in this study categorizes security threats into six types: spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege. All threats can occur for each component in DFD. Using STRIDE, we identified 1007 security threats. Table II lists all of the threats that can be derived from DFD by using the STRIDE technique.

TABLE II
THREAT FOR SUMS

Type	ID	Name	Threat	Attack Library	No.
Entity	E1	Software Developer	S	AL-V-19	T1
			S	AL-V-40	T2
			S	AL-P-1	T3
...	...	...	...	...	...
Data Flow	DF55	Vehicle Data	I	AL-P-4	T1004
			I	AL-P-13	T1005
			I	AL-P-18	T1006
			D	AL-P-19	T1007

4) Generating attack tree: Attack tree is a technique that creates an actual attack scenario for the system based on the threat derived from the previous step [50]. The root node represents the final goal of the attacker, and the leaf node is the specific attack method or attack point to reach the root node. The lower node can achieve the attack target of the upper node through AND and OR conditions. Under the AND condition, all lower nodes bounded by the AND condition must be achieved to attack the upper node. Under the OR condition, one of the lower nodes bounded by the OR condition must achieve the target of attacking the upper node. Examples of attack trees proposed in this paper are presented in Figure 3.

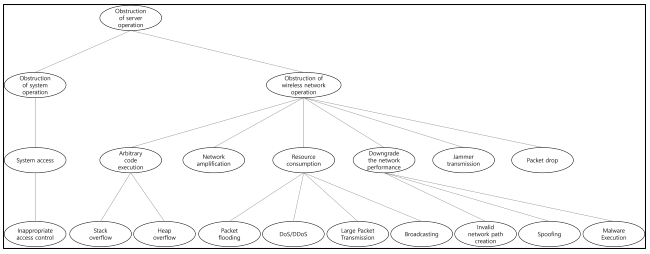

Fig. 3. Attack Tree for SUMS

5) Deriving security requirements: Security requirements are used to mitigate specific attacks on attack tree. It is important to note that when deriving the security requirements, we carefully determined that those requirements can mitigate the specific attacks included in attack tree. We eventually proposed 38 security requirements. Some of them are listed in Table III. As explained in Section I, security requirements in the UN R156 specify only abstract information. So, we proposed security requirements in detail. Security requirements of this paper identified all threats that could occur in SUMS through threat modeling. And attack scenarios that could occur through the identified threats, and derived security requirements that could mitigate the specific attack method in the attack scenarios. Therefore, applying the security requirements of this paper to the SUMS architecture can respond to all threats that may occur in SUMS.

TABLE III
SECURITY REQUIREMENT FOR SUMS

No (SR)	Security Requirement
1 (SR10)	When authentication of the OEM server is attempted, the authentication process must be terminated if the authentication fails more than a certain number of times (e.g., 5 times) to prevent attacks such as a full investigation.
2 (SR11)	Before performing an action on the OEM server, the OEM server must verify that each object is authorized to perform the action based on each object's identifier (e.g., ID/PW).
3 (SR12)	Before accessing the OEM server, the appropriate device and person must be identified based on each object's identifier (e.g., ID).

As explained in Section I, security requirements in the UN R156 specify only abstract information. So, we proposed security requirements in detail. Security requirements of this paper identified all threats that could occur in SUMS through threat modeling. And attack scenarios that could occur through the identified threats, and derived security requirements that could neutralize the attack scenarios. Therefore, the security

requirements of this paper can respond to all threats that may occur in SUMS. These security requirements are mapped to the requirements of UN R156 which are related to security as shown in Table IV.

TABLE IV
RELATIONSHIP BETWEEN OUR PROPOSALS AND UN R156

UN R156	Security requirement
The process they will use to ensure that software updates will be protected to reasonably prevent manipulation before the update process is initiated.	SR1, SR2, SR3, SR4, SR5, SR6, SR7, SR8, SR9, SR10, SR11, SR12, SR13, SR14, SR15, SR16, SR17, SR18, SR19, SR20, SR21, SR22, SR23, SR24, SR25, SR26, SR27, SR28, SR29, SR30, SR32, SR33, SR34, SR35, SR36, SR37, SR38
The update processes used are protected to reasonably prevent them being compromised, including development of the update delivery system.	SR1, SR2, SR3, SR4, SR5, SR6, SR7, SR8, SR9, SR10, SR11, SR12, SR13, SR14, SR15, SR16, SR17, SR18, SR19, SR20, SR21, SR22, SR23, SR24, SR25, SR26, SR27, SR28, SR29, SR30, SR32, SR33, SR34, SR35, SR36, SR37, SR38
...	...
The vehicle shall ensure that preconditions have to be met before the software update is executed.	SR1, SR2, SR3, SR4, SR5, SR6, SR7, SR8, SR9, SR10, SR11, SR12, SR13, SR14, SR15, SR16, SR17, SR18, SR19, SR20, SR21, SR22, SR23, SR24, SR25, SR26, SR27, SR28, SR29, SR30, SR31, SR32, SR33, SR34, SR35, SR36, SR37, SR38

B. Designing Secure SUMS Architecture

In this section, a secure SUMS architecture is proposed. Its design is based on the security requirements derived from threat modeling. The architecture for SUMS was designed using a deployment diagram and a component diagram among the UML diagrams. A deployment diagram was created to express the relationship between the hardware resources upon which the software under development was mounted. Based on the deployment diagram, we propose a relationship among four OEM servers, one vehicle, and one terminal in the repair shop, which are components of the SUMS architecture. This is similar to general SUMS architecture. This is because it only reflects the abstract requirements specified in the existing documents. After writing the deployment diagram, we constructed a component diagram of the SUMS. A component diagram is used to show the interactions between the software and its functions loaded into the system. Based on the derived deployment diagram and its security requirements, security functions are reflected on the component diagram of secure SUMS architecture. Based on the component diagram, the secure SUMS architecture operates as follows.

On the OEM side, the patch files are ready to be deployed, as shown in Figure 4.

- When the OEM developer finishes the development of the patch file, it is then uploaded to the development server. Currently, developers should authenticate their identities using the ID/password they registered earlier. If the authentication process is successful, the patch files are transmitted to the diagnostic server.
- In this server, various tests for patch files are conducted. The most representative example for testing on this server is whether patch files are developed in compliance with secure coding guidelines. If the patch files pass all of the tests, they are transmitted to the update server.

On the client side, patch files are installed as shown in Figure 5:

- The client can request patch files from the CRM server of the OEM directly or through engineers in the repair shop. Currently, the OEM CRM server authenticates the identity of the automotive system by checking the vehicle identification number (VIN) and its ECU version. If the authentication process succeeds, the server checks the version of files that are installed in automotive systems and up-to-date patch files stored in the OEM update server. This is possible because the CRM server stores information such as the client's personnel information, the VIN of their vehicle, patch histories, and so on.
- If the version of files installed in an automotive system is lower than the files in the update server of the OEM, the update server transmits the related files to the automotive system. Otherwise, they inform the client that the software embedded in automotive systems is up-to-date.

V. EVALUATION

We prove the security requirements derived from Section IV. There are many methods to prove a relationship between requirements and design. Among them, we choose a formal method to prove it.

A. Event-B

Event-B is one of the most frequently used formal methods. It expresses the state of a system using set theory and predicate logic [51]. The key features of this language are the use of refinement to model the system at different levels and the use of mathematical proof to verify consistency between models. The Event-B model consists of context and machine [52]. Context represents the static properties of the model. By contrast, the machine represents the dynamic properties of the model. By combining the context and machine, the state of the system can be completely expressed. An engineer can specify some constants and variables within a context or machine. The constants are restricted by axioms in this context. Variables are connected by invariants, and axioms and invariants are expressed by the set theory expression in the machine. In addition, Event-B specifies many events in a machine. An event comprises guards and actions. Guards represent the enabling condition of the event, and actions represent how the state is modified by the event.

B. Formal specification of security requirements

In this section, we showed that our security requirements, which are written in the informal form are converted to formal form by Event-B. Among the informal requirements, three security requirements in Table III are related to the login function for users. SUMS attempts to verify the user's identity

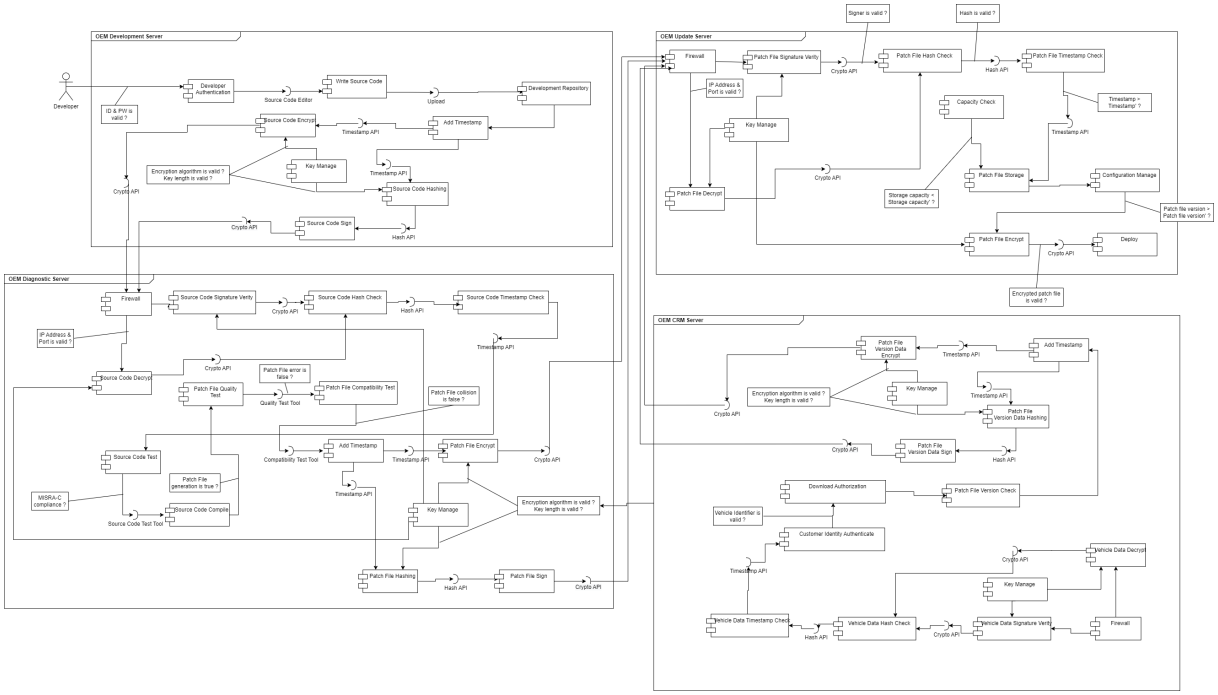

Fig. 4. Component diagram of OEM in secure SUMS architecture

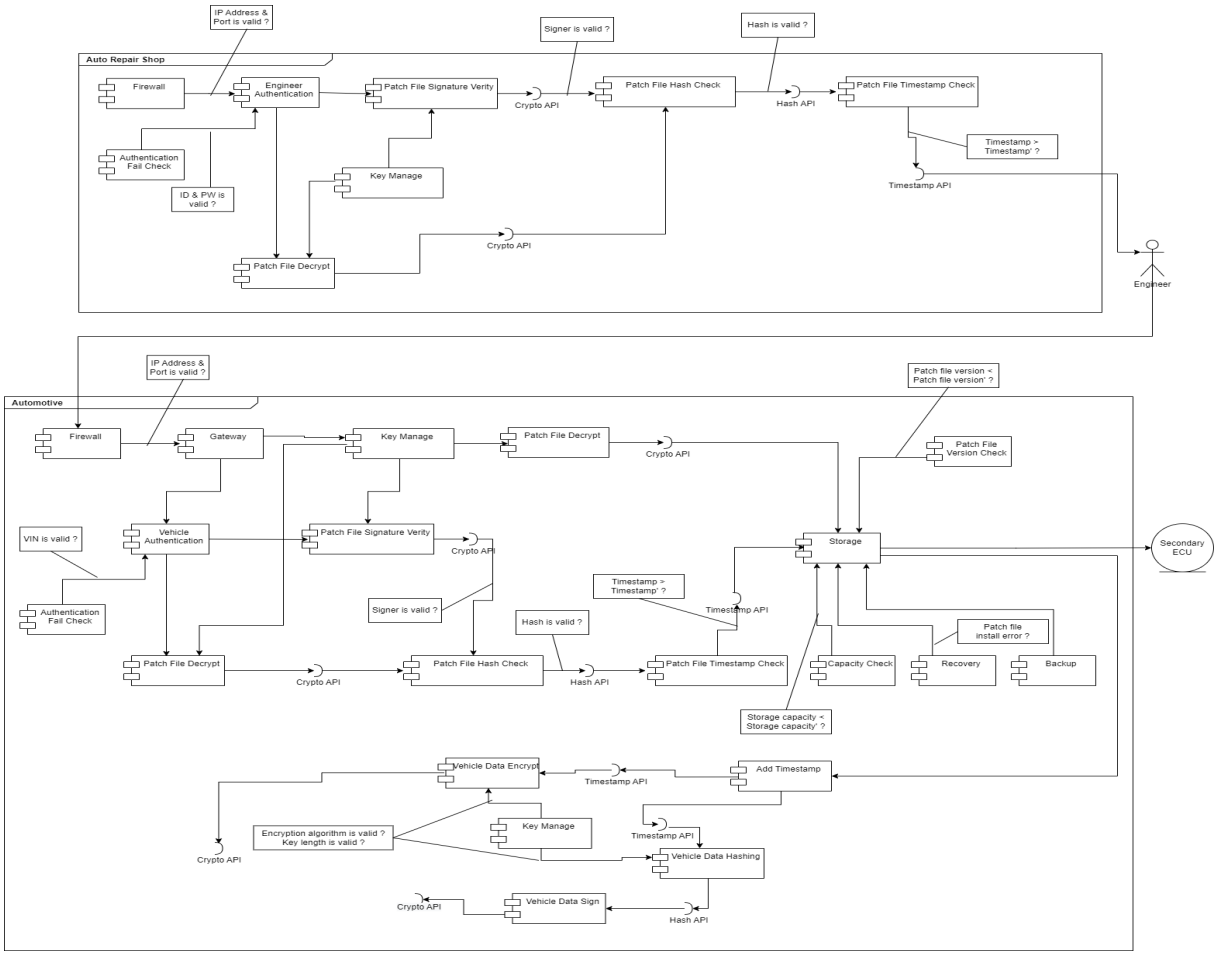

Fig. 5. Component diagram of automotive in secure SUMS architecture

via ID and password by using this function. In figure 6, the formal version of the log-in function is shown. The employee can access the OEM development server using the ID and password. After login Succeed, the employee is authorized according to their role. In the case of the development team leader, upload/fix/transfer/download/approve permission is granted. However, there is a restriction in the login trial. If the authentication process fails more than five times, the authentication of the ID is forbidden.

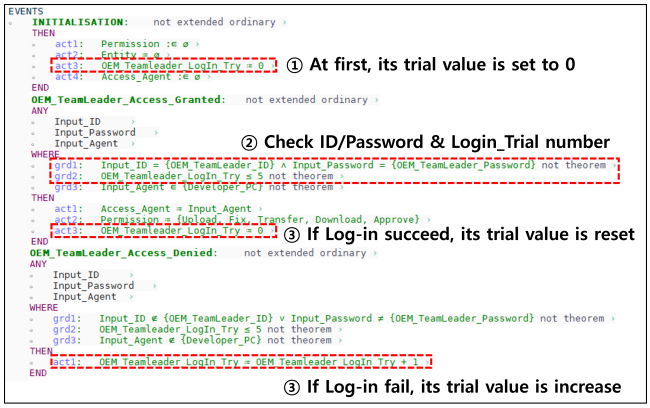

Fig. 6. Formal specification of IP and Port

C. Formal verification of security requirements

After the formal specifications, we conducted formal verification. Formal verification is conducted to verify that the requirements specified above are operating correctly and that there is no contradiction between the requirements in the secure SUMS architecture. As a result of formal verification, the proof obligation is discharged by the automatic verifier provided by Rodin and the external verifier named Atelier B. Figure 7 shows the proof of our formal model of the development server.

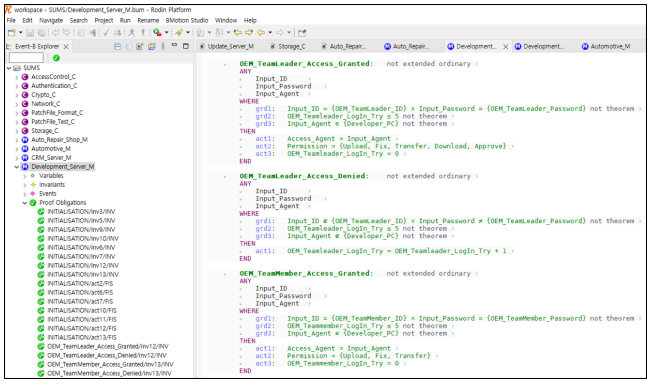

Fig. 7. Formal verification of SUMS

VI. Conclusion

Unlike in the past, vehicles are connected to many other devices to update their systems automatically through SUMS. However, it renders vehicles vulnerable because extra software and hardware, which are necessary for providing the functionality, become new attack surfaces. In this study, we conducted threat modeling for SUMS. Through the threat modeling process, we systematically identified all threats and attacks that may occur in SUMS as well as those specified in UN R156 regulations. To mitigate these threats and attacks, we proposed security requirements and designed a secure SUMS architecture. Finally, we formally verified that our secure SUMS architecture reflects the security requirements derived from threat modeling using Event-B.

Acknowledgement

This work was partly supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2018-0-00532, Development of High-Assurance(EAL6) Secure Microkernel, 50) and Korea Agency for Infrastructure Technology Advancement(KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant 23AMDP-C162334-03, Research and Development on Integrated Automotive Cybersecurity Assurance Technologies, 50)

References

[1] Z. Wang, J. J. Han and T. Miao *An Efficient and Dependable FOTA-Based Upgrade Mechanism for In-Vehicle Systems*, In Proceedings of the International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2019.

[2] *Software Updates*, [Online] Available: https://www.tesla.com/support/software-updates

[3] *Hyundai reduces ML model training time for autonomous driving models using Amazon SageMaker*, [Online] Available: https://aws.amazon.com/ko/blogs/machine-learning/hyundai-reduces-training-time-for-autonomous-driving-models-using-amazon-sagemaker/

[4] C. Miller and C. Valasek *Remote Exploitation of an Unaltered Passenger Vehicle*, Black Hat USA, 2015.

[5] S. Nie, L. Liu and Y. Du *Free-Fall: Hacking Tesla from Wireless to CAN Bus*, Black Hat USA, 2017.

[6] R. P. Weinmann and B. Schmotzle *T-BONE: Drone vs. Tesla*, CanSecWest Conference, 2021.

[7] *Uniform provisions concerning the approval of vehicles with regards to software update and software updates management system*, [Online] Available: https://unece.org/sites/default/files/2021-03/R156e.pdf

[8] A. Shostack *Threat modeling: Designing for security*, 1st ed John Wiley and Sons, 2014.

[9] C. W. Lee *A system theoretic approach to cybersecurity risks analysis of passenger autonomous vehicles*, Master, Degree-Granting Massachseets Inistitute of Technology, Cambridge, 2018.

[10] T. Placho, C. Schmittner, A. Bonitz and O. Wana *Management of automotive software updates*, Microprocessors and Microsystems, 2020.

[11] M. Hamad *A Multilayer Secure Framework for Vehicular Systems*, Von der Fakultet fer Elektrotechnik, 2020.

[12] J. Yu, S. Wagner and F. Luo *An STPA-based Approach for Systematic Security Analysis of In-vehicle Diagnostic and Software Update Systems*, Computer Science, 2020.

[13] A. Mukherjee, R. Gerdes and T. Chantem *Trusted Verification of Over-the-Air (OTA) Secure Software Updates on COTS Embedded Systems*, In Proceedings of the Third International Workshop on Automotive Vehicle Security, 2021.

[14] C. Ponsard and D. Darquennes *Towards Formal Security Verification of Over-the-Air Update Protocol: Requirements, Survey and UpKit Case Study*, In Proceedings of the 5th International Workshop on FORmal methods for Security Engineering, 2021.

[15] Z. Wu, T. Liu, X. Jia and S. Sun *Security design of OTA upgrade for intelligent connected vehicle*, In Proceedings of the 1st International Conference on Control and Intelligent Robotics, 2021.

[16] A. Ghosal, S. Halder, M. Conti *Secure Over-the-Air Software Update for Connected Vehicles*, Computer Networks, 2022.

[17] R. Kirk, H. N. Nguyen, J. Bryans, S. A. Shaikh and C. Wartnaby *A formal framework for security testing of automotive over-the-air update systems*, Journal of Logical and Algebraic Methods in Programming, 2022.

[18] *Data Flow Diagrams Examples*, [Online] Available: http://www.umsl.edu/ sauterv/analysis/dfd/dfd_intro.html

[19] *What is a Data Flow Diagram*, [Online] Available: https://www.lucidchart.com/pages/data-flow-diagram

[20] *Checking Threat Modeling Data Flow Diagrams for Implementation Conformance and Security*, [Online] Available: http://reports-archive.adm.cs.cmu.edu/anon/isri2006/CMU-ISRI-06-124.pdf

[21] *CVE*, [Online] Available: https://cve.mitre.org

[22] *CWE*, [Online] Available: https://cwe.mitre.org

[23] Y. Ashibani and Q. H. Mahmoud *Cyber physical systems security: Analysis, challenges and solutions*, Computers Security, 2017.

[24] C. Riggs, C. E. Rigaud, R. Beard, T. Douglas and K. Elish *A survey on connected vehicles vulnerabilities and countermeasures*, Journal of Traffic and Logistics Engineering, 2018.

[25] M. L. Manna, L. Treccozzi, P. Perazzo, S. Saponara and G. Dini *Performance Evaluation of Attribute-Based Encryption in Automotive Embedded Platform for Secure Software Over-The-Air Update*, Sensors, 2021.

[26] M. Zoppelt and R. T. Kolagari *UnCle SAM: Modeling Cloud Attacks with the Automotive Security Abstraction Model*, In Proceedings of the Tenth International Conference on Cloud Computing, GRIDs, and Virtualization, 2019.

[27] M. Kim, J. Park, E. Jeong, I. Oh, K. Yim and J. Park *OTA Vulnerability on User Equipment in Cloud Services*, In Proceedings of the International Conference on Information Technology Systems and Innovation, 2018.

[28] A. M. Nasser, D. Ma and S. Lauzon *Safety-Driven Cyber Security Engineering Approach Applied to OTA*, Computer Science, 2017.

[29] N. Weiss, E. Pozzobon and S. Renner *Extending Vehicle Attack Surface Through Smart Devices*, In Proceedings of the Eleventh International Conference on Emerging Security Information, Systems and Technologies, 2017.

[30] M. Levi, Y. Allouche and A. Kontorovich *Advanced Analytics for Connected Car Cybersecurity*, In Proceedings of the 87th International Conference on Emerging Security Information, Systems and Technologies, 2018.

[31] T. Alladi, V. Chamola, B. Sikdar and K. R. Choo *Consumer IoT: Security Vulnerability Case Studies and Solutions*, IEEE Consumer Electronics Magazine, 2020.

[32] P. Bajpai, R. Enbody, B. H. C. Cheng *Ransomware Targeting Automobiles*, In Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security, 2020.

[33] M. H. Eiza, Q. Ni *Driving with sharks: Rethinking connected vehicles with vehicle cybersecurity*, IEEE Vehicular Technology Magazine, 2017.

[34] P. Carsten, T. R. Andel, M. Yampolskiy, J. T. McDonald and S. Russ *A System to Recognize Intruders in Controller Area Network (CAN)*, In Proceedings of the 3rd International Symposium for ICS and SCADA Cyber Security Research, 2015.

[35] T. Hoppe, S. Kiltz and J. Dittmann *Security threats to automotive CAN networks-Practical examples and selected short-term countermeasures*, Reliability Engineering and System Safety, 2010.

[36] S. Nie, L. Liu, Y. Du and W. Zhang *Over-the-air: How we remotely compromised the gateway, BCM, and autopilot ECUs of Tesla cars*, Black Hat USA, 2018.

[37] B. M. Luettmann and A. C. Bender *Man in the middle attacks on auto updating software*, Bell Labs Technical Journal, 2007.

[38] J. N. Brewer and G. Dimitoglou *Evaluation of Attack Vectors and Risks in Automobiles and Road Infrastructure*, In Proceedings of the International Conference on Computational Science and Computational Intelligence, 2019.

[39] M. Dibaei, X. Zheng, K. Jiang, R. Abbas, S. Liu, Y. Zhang, Y. Xiang and S. Yu *Attacks and defences on intelligent connected vehicles: a survey*, Digital Communications and Networks, 2020.

[40] P. Carsten *In-Vehicle Networks: Attacks, Vulnerabilities, and Proposed Solutions*, In Proceedings of the 10th Annual Cyber and Information Security Research Conference, 2015.

[41] H. Wen, Q. Chen, Z. Lin *Plug-N-pwned: Comprehensive vulnerability analysis of OBD-II dongles as a new over-the-air attack surface in automotive IoT*, In Proceedings of the 29th USENIX Conference on Security Symposium, 2020.

[42] K. Cho *From Attack to Defense: Toward Secure In-vehicle Networks*, Doctor, Degree-Granting University of Michigan, 2018.

[43] S. Checkoway *Comprehensive Experimental Analyses of Automotive Attack Surfaces*, In Proceedings of the 20th USENIX Conference Security Symposium, 2011.

[44] Moukahal, Lama and M. Zulkernine *Security vulnerability metrics for connected vehicles*, In Proceedings of the 19th IEEE International Conference on Software Quality, Reliability and Security Companion, 2019.

[45] McCarthy, Charlie, K. Harnett and A. Carter *Characterization of potential security threats in modern automobiles: A composite modeling approach*, National Highway Traffic Safety Administration, 2014.

[46] Salfer, Martin and C. Eckert *Attack surface and vulnerability assessment of automotive Electronic Control Units*, In Proceedings of the 12th International Joint Conference on e-Business and Telecommunications, 2015.

[47] Thing, V. LL and J. Wut *Autonomous Vehicle Security: A Taxonomy of Attacks and Defences*, In Proceedings of the IEEE International conference on internet of things (ithings) and IEEE green computing and communications (greencom) and IEEE cyber, physical and social computing (cpscom) and IEEE smart data (smartdata), 2016.

[48] ITU-T *X.1373: Secure software update capability for intelligent transportation system communication devices*, [Online] Available: https://www.itu.int/rec/T-REC-X.1373-201703-I/en

[49] FASTR *Automotive Industry Guidelines for Secure Over-the-Air Updates*, [Online] Available: http://telematicswire.net/wp-content/uploads/2017/11/FASTR_AutomotiveIndustry_Guidelines_Secure Over-the-Air_Updates_final.pdf

[50] V. Saini, Q. Duan and V. Paruchuri *Threat modeling using attack trees*, Journal of Computing Sciences in Colleges, 2008.

[51] T. S. Hoang *An Introduction to the Event-B Modelling Method*, Industrial Deployment of System Engineering Methods, Springer-Verlag, 2013.

[52] C. Metayer, J. R. Abrial and L. Voisin *RODIN Deliverable 3.2 Event-B Language*, [Online] Available: http://rodin.cs.ncl.ac.uk/deliverables /D7.pdf

[53] Ansari, M. T. Jamal, D. Pandey and M. Alenezi *STORE: security threat oriented requirements engineering methodology*, Journal of King Saud University-Computer and Information Sciences, 2022.

[54] *ISO 21434*, [Online] Available: https://www.iso.org/standard /70918.html

[55] *Microsoft Threat Modling*, [Online] Available: https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

[56] Uzunov, V. Anton and E. B. Fernandez *An extensible pattern-based library and taxonomy of security threats for distributed systems*, Computer Standards and Interfaces, 2014.

[57] Baquero, O. Abraham, A. J. Kornecki and J. Zalewski *Threat modeling for aviation computer security*, crosstalk, 2015.

[58] Dahbul, C. Lim and J. Purnama *Enhancing honeypot deception capability through network service fingerprinting*, Journal of Physics: Conference Series, 2017.

[59] *The SDL Progress Report*, [Online] Available: https://www.microsoft.com/en-us/download/details.aspx?id=14107

[60] *Microsoft Security Development Lifecycle (SDL) – Process Guidance*, [Online] Available: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/cc307891(v=msdn.10)

[61] *The Microsoft® Security Development Lifecycle (SDL)*, [Online] Available: https://slideplayer.com/slide/13903228/

[62] *ISO 26262-6:2018 Road vehicles - Functional safety - Part 6: Product development at the software level*, [Online] Available: https://www.iso.org/standard/68389.html