Implementation of Real-time Security Attack Detection and Classification Algorithms for Embedded Devices

KOGAWARA Toru, YAMAMOTO Taisei, ZENG Zhen and HIROBE Naoki

With the spread of IoT, embedded devices are also connected to the Internet, which is in danger of security attacks. If a security attack stopped the control of embedded devices, substantial damage is expected. Therefore, a measure is demanded that does not lose control even if the device is attacked. If the device can immediately classify the type of attack when it experiences a security attack, it can automatically respond to the attack. As a result, it is possible not to lose control and to minimize any damage. That is why we developed and implemented a lightweight attack detection and classification algorithm and confirmed that it was possible to be run in real time on embedded devices.

1. Introduction

Following the penetration of IoT into diverse fields, not only personal computers or servers but also various embedded devices have come to be connected to the Internet. As a result, embedded devices are now exposed to security attack risks in much the same way as personal computers or servers have been.

The above is particularly true with embedded devices, such as automotive vehicle control units and controllers for factory machines. These embedded devices that perform physical actions; therefore, security attacks on them may lead to serious personal and material damage. For instance, successful car hacking incidents have been reported in which vehicle operations, including the steering wheel and engine braking, were remotely controlled via a cell-phone line[1].

Moreover, many examples of malware, such as viruses, targeting control equipment in facilities, such as factories and power plants, have been reported that caused massive losses, such as broken machinery and power outages[2].

The standard reference consulted when planning countermeasures against security attacks is the National Institute of Standards and Technology's (NIST) Cybersecurity Framework[3], which consists mainly of five phases of identification, protection, detection, response, and recovery for security measures to be worked on. Fig. 1 shows the outline of the Cybersecurity Framework.

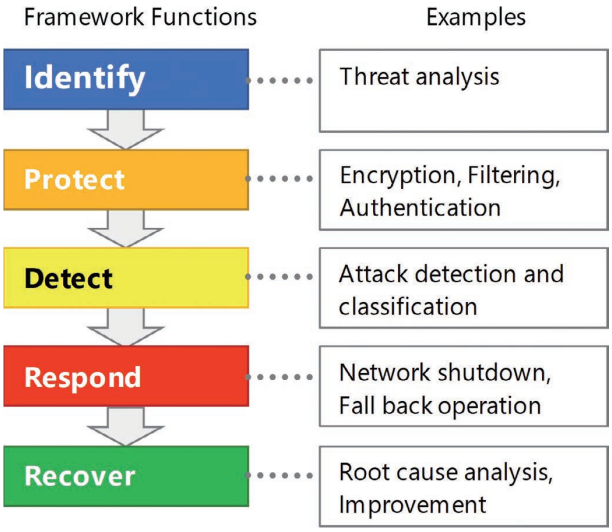


Fig. 1 Outline of the NIST Cybersecurity Framework

The basic principle of a security measure is to analyze security risks to a target system in the identification phase and protect the target system from security attacks in the protection phase. Security attacks are, however, in constant evolution. Attacks able to break through the protection will emerge sooner or later.

What then becomes necessary are the detection of security attacks, including ones that broke through the protection and an appropriate response to the detected attack for damage minimization. This paper presents a report on the development and implementation of a detection algorithm designed to run in real time on embedded devices, even on low-resource embedded devices, and minimizes damage through responses.

Contact : *KOGAWARA Toru* toru.kogawara@omron.com

2. Challenge

A proper response to a detected attack requires determination of the nature of the attack. This requirement aims to select appropriate countermeasures for the types of attacks encountered. Attack detection and classification are supposed to occur simultaneously. Hence, these are lumped together and called “detection and classification” throughout the rest of this paper.

In the case of control equipment for machines that perform physical actions, any problem that may occur to it must be addressed without delay to prevent damage to nearby personnel and the machinery itself. The same holds for responses to security attacks. Hence, the detection and classification function needs to be executable in real time.

Accordingly, the challenge herein is to develop a method of executing the attack detection-and-classification function on an embedded device in real-time.

3. Conventional attack detection technology

Security measures at the embedded device level are just beginning to emerge. Particular note should be taken that no products exist that can execute the attack detection-and-classification function in real time. Meanwhile, some IT security software products for personal computers, servers, and other computing equipment can execute the attack detection-and-classification function in real time.

In IT attack detection and classification technologies, very minute details are identified, including, for example. variants and version names of detected viruses. Some technologies use both machine learning and AI for detecting unknown attack methods, and hence need massive amounts of data along with massive amounts of computation for data retrieval.

Accordingly, IT attack detection and classification technologies require extremely fast CPUs and large memory/storage capacities and hence cannot be ported directly into embedded devices. AppGuard[4], available from Blue Planet Works, may serve here as an illustrative example of such technologies. AppGuard is software that runs on the Windows series of operating systems. One of its selling points is that it runs very light. Nevertheless, its program body (protection engine) has a size of 1 MB and requires free disk space of 500 MB and a CPU clock speed of 1.8 GHz or more. Though considerably light for an application for Windows, AppGuard is well beyond the specs of embedded devices. For example, the specs of the in-vehicle microcomputer used for the development presented herein are a 1 MB flash ROM capacity (instead of PC free disk space), a memory capacity of 128 KB, and a CPU clock speed of approximately 100 MHz. This platform is far from an environment that allows direct import of the technology of AppGuard.

4. Development of the attack detection-and-classification algorithm

This chapter explains the attack detection and classification technology we developed and previously reported[5]. This technology is intended for embedded devices used as control units and is designed to perform attack detection and classification on embedded devices in real time.

A security attack on an embedded device comes from an external source. In other words, external inputs include anomalous ones. If anomalous inputs are defined as “anomalies,” and if more than one anomaly is detected, a device is found to be under attack. Our basic idea of attack detection is based on this sequence of events. The combination of anomalies that constitute an attack depends on the type of attack. Therefore, the detection of a specific combination of anomalies means that of a specific attack. Such detection of specific attacks for each type of attack of interest leads to the simultaneous execution of attack detection and classification. Fig. 2 shows the conceptual diagram of the attack detection-and-classification algorithm.

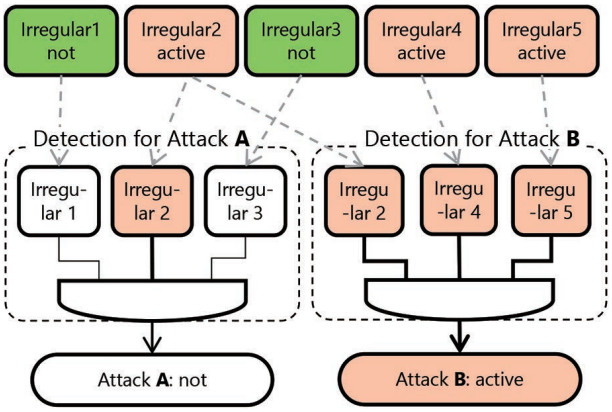


Fig. 2 Conceptual diagram of the attack detection-and-classification algorithm

This algorithm only judges simple combinations of occurrences and non-occurrences of anomalies and seems to meet the two requirements of high-speed processing capability and low required data capacity, which are imposed by the hardware limitations of embedded devices. Table 1 shows the six types of attacks to be classified. These types are in line with STRIDE[6], one of the standard threat analysis models, and correspond to the six threats, the initials of which are S, T, R, I, D, and E, respectively.

Table 1 Types of attacks to be classified

Type of attack	Example
Spoofing	Sending in fake control commands disguised as an authentic device
Tampering	Tampering firmware and configuration data
Repudiation	Erasing or overwriting logs to erase the traces of intrusion
Information disclosure	Acquiring confidential information, such as special communication commands or configuration data
Denial of service	Sending a huge amount of communication messages to a system to force the system out of service
Elevation of privilege	Illicitly obtaining a privileged access right

Moreover, when coming under a complex attack consisting of multiple attacks, the embedded device must be able to determine the priority order for dealing with the attacks. Therefore, we modeled the progress of an attack as an irreversible state transition process and proposed an approach that deals with attacks in the descending order of their progress.[7] Fig. 3 shows a state-transition diagram that models the progress of an attack (called an attack state-transition diagram).

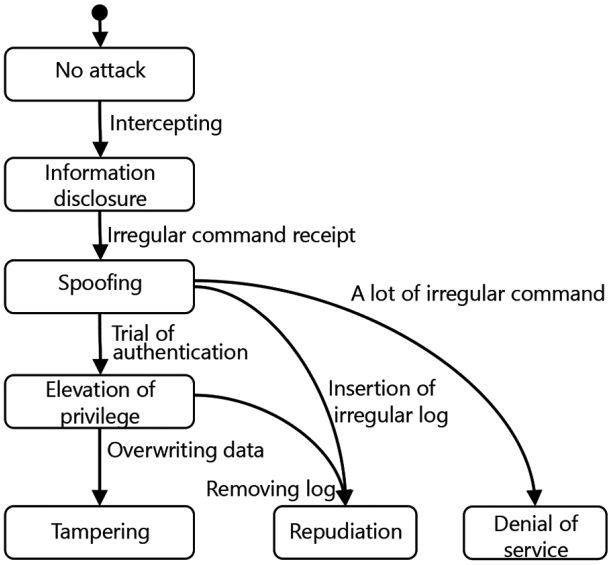


Fig. 3 Typical attack state-transition diagram

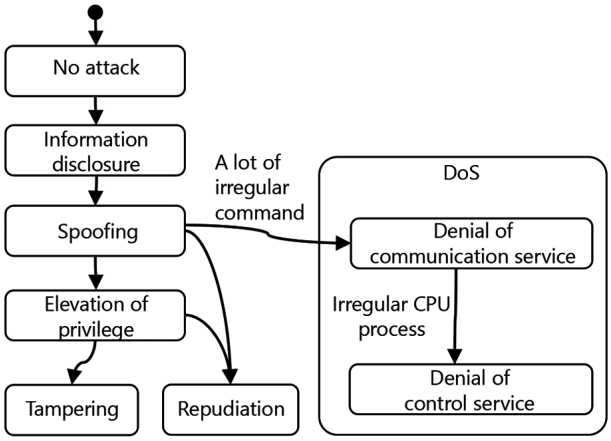
5. Finer classification of attacks

5.1 Challenge to the classification granularity for attacks

The technology presented in Chapter 4 classifies attacks into six types. The six-type classification is, however, still insufficient to achieve the purpose of making appropriate actions for the types of attacks encountered. The action to be taken for, for example. a DoS attack may differ depending on whether the service is unavailable simply because of an overloaded communication bus or the device is down because of the CPU's failure to meet the required processing time while processing a huge amount of messages. Hence, attacks should be more finely distinguished to be reliably detected and classified.

5.2 Finer classification through the integration of internal state anomalies

The attack detection-and-classification algorithm explained in Chapter 4 assumes that the anomalies to be detected are external inputs. Meanwhile, however, as shown by the example given in Section 5.1, an algorithm needs to make judgments on external inputs as well as the internal state of the embedded device to ensure a proper response. If a device is in an anomalous internal state or behaving anomalously, such a state or behavior should also be defined as an "anomaly." Moreover, attacks detected at the occurrence of anomalies due to the internal state of the device should be finely classified as new types of attacks. We added this finer classification to the attack state-transition diagram in Fig. 3 to implement an attack detection function for making integrated judgments based on both external input anomalies and internal state anomalies. Fig. 4 shows a typical attack state-transition diagram with a finer classification of attacks.



* The same transition conditions as in Fig.3 have been omitted.

Fig. 4 Typical attack state-transition diagram with a finer classification of attacks through the integration of internal state anomalies

6. Implementation of the algorithm

6.1 Model intended for in-vehicle ECUs

We developed a model that assumes that the device in which the algorithm is to be implemented is an electronic control unit for in-vehicle installation (hereafter "in-vehicle ECU"). Being a lower resource among embedded devices and, as explained in Chapter 1, exposed to security attack risks, in-vehicle ECUs are suitable for the technology presented herein to be applied to. Table 2 shows the hardware specifications. This hardware includes an in-vehicle one-chip microcontroller as its CPU and has a multiple number of channels compliant with CAN, one of

the standard in-vehicle network protocols. Additionally, the hardware performs transmission and reception of specified messages as its intended function for ECUs.

Table 2 Hardware specifications for the model intended for in-vehicle ECUs

Item	Specification
CPU clock speed	96 MHz
Flash ROM	1 MB
RAM	128 KB
Communication protocol	CAN
Transfer rate	500 kbps
Control period	1 ms

6.2 Algorithm implementation method

The attack classification algorithm developed this time makes separate judgments on attacks based on their types and hence can perform high-speed processing by bit operations. As explained in Chapter 4, in connection with attack classification, attacks are modeled as irreversible state transitions. Hence, the algorithm judges that a certain attack A has occurred when the following conditions are both met: all transition conditions leading up to A are met (IN Conditions), and none of the conditions for the transition from A to the next are met (OUT Conditions). Fig. 5 shows the conceptual diagram for this judgment.

All the IN Conditions hold as ANDed conditions and hence can be expressed as a bit string. The data set with a value of 1 for the bit corresponding to the anomaly constituting an IN Condition and a value of 0 for the other bits in a bit string is called an IN Condition bitmask. An OUT Condition can similarly be expressed using a bitmask.

If for a detected anomaly, a bit string is available with the value of the bit for the detected anomaly being 1 and that of other bits being 0, the algorithm can tell whether an IN or an OUT Condition has occurred by performing a single AND operation with each of their respective bitmasks. In other words, two bit operations suffice to determine that a single attack has been detected.

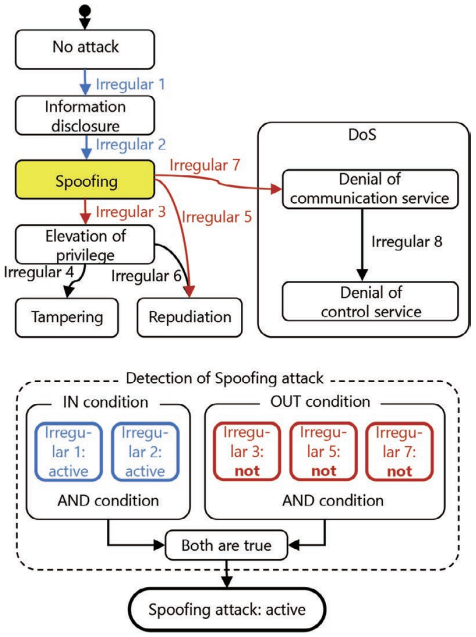


Fig. 5 Judgment criteria based on the attack state-transition diagram

Then, for the algorithm thus implemented, the CPU computational load and the data capacity need to be estimated.

Fig. 6 shows the bitmasks necessary for the IN and OUT Conditions of each attack, assuming that n types of attacks are to be detected and, for their detection, m types of anomalies are to be detected.

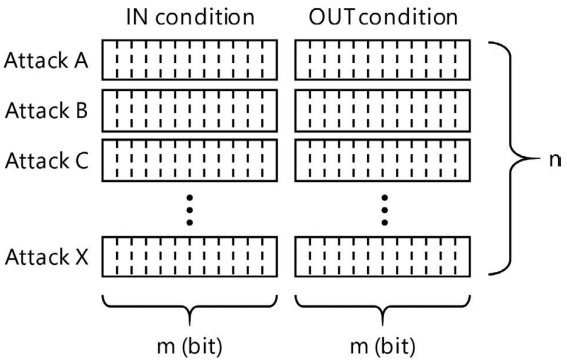


Fig. 6 Bitmasks required for attack detections

Let us here estimate the CPU computational load. Assuming that the CPU is a 32-bit CPU commonly used for in-vehicle ECUs, 32 bits' worth of data can be processed per operation; hence, the number of times of operation is $2 \times [m/32] \times n$ times.

Similarly, let us estimate the required data capacity. Data are stored in bytes (1 byte = 8 bits); hence, the required data capacity is $2 \times [\mathrm{m}/8] \times n$ bytes.

6.3 Specifications for the attack detection-and-classification function

For the model intended for in-vehicle ECUs, we designed and

developed a security measure. Using this security measure, we identified the attacks to be classified and the anomalies to be detected for attack detection. As a result, the number of types of attacks to be classified increased to 11. Table 3 and Fig. 7 show the specifications for the designed attack detection-and-classification function.

Table 3 Specifications for the attack detection-and-classification function

Item	Specification
Types of detectable and classifiable attacks	11
Types of anomalies used for attack detection and classification	45

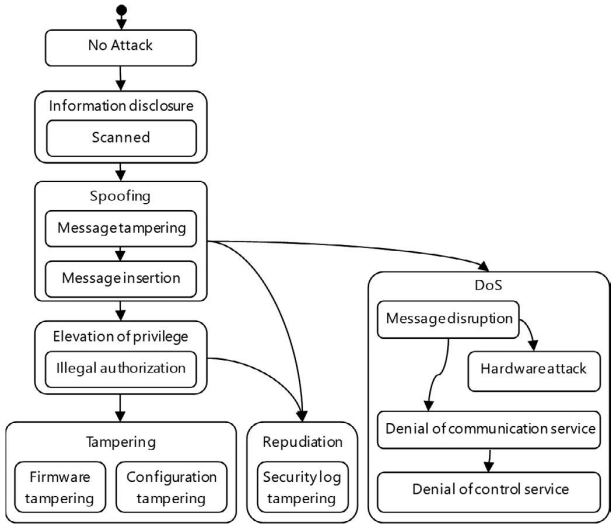


Fig. 7 Designed attack state-transition diagram

Let us apply these functional specifications, together with the hardware specifications in Section 6.1, to the estimation results in Section 6.2 to estimate the actual processing time and data capacity. The parameters n and m in Section 6.2, in other words, the number of types of attacks and the number of types of anomalies, are $n = 11$ and $m = 45$, respectively.

The estimated CPU computational load is 44 bit operations. Assuming that a bit operation completes in one CPU clock period, the processing time required for the operations is approximately $44/(96 \times 10^6) \approx 0.5$ μs, which is approximately 1/2,000 of a control period of 1 ms. Meanwhile, the estimated data capacity is 132 bytes, which is approximately 1/8,000 of a flash ROM capacity of 1 MB.

From the above, our algorithm is expected to be implementable in embedded devices, such as in-vehicle ECUs, with no problem with the computational load and the data capacity.

7. Evaluation of the implemented model

7.1 Requirements

The requirements for the attack detection-and-classification function are defined as follows:

- Proper function
  Ability to perform correct detection and classification for all the defined attack types
- Performance sufficient to run in real time without disrupting the inherent function of the embedded device
  Sufficiently low memory usage (less than 10% of the respective capacities of the flash ROM and the RAM)
  Even under the heaviest load, the total time of all functions does not exceed the control period (1 ms).

7.2 Functional evaluation

Table 4 shows the attack method we set for each of the defined attack types.

Table 4 Test cases for the attack detection-and-classification function

No.	Type of attack	Description of attack
1	Scanning	Injecting brute force attack messages from the attack device into the communication path to identify a CAN ID
2	Message tampering	Rewriting and forwarding data through an intermediate device inserted in the communication path
3	Unauthorized message insertion	Injecting illegal messages with the same IDs as those of legal messages from the attack device into the communication path
4	Illegal authentication	Using an illegal encryption key to start an authentication sequence from the attack device
5	Message blocking	Selectively blocking messages with certain IDs using an intermediate device inserted in the communication path
6	Hardware attack	Causing an hardware error by sending illegal signals from the attack device only during the transmission of a message with a certain ID
7	Communication denial of service	Injecting a huge amount of messages not meant to be received from the attack device into the communication path to saturate the communication
8	Control denial of service	Injecting a huge amount of messages meant to be received from the attack device into the communication path to saturate the control process
9	Security log tampering	Tampering the security log on the nonvolatile memory and start the model
10	Firmware tampering	Tampering the firmware on the nonvolatile memory and start the model
11	Configuration data tampering	Tampering the configuration data on the nonvolatile memory and start the model

Fig. 8 shows the schematic of the system configuration as tested.

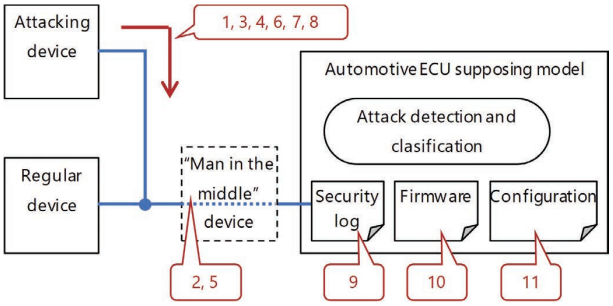


Fig. 8 System configuration schematic

The points of attack corresponding to the test case numbers in Table 4 are indicated by the numbers in the balloons in the figure. Table 5 shows the results of running the test cases in Table 4 with the system configuration in Fig. 8.

Table 5 Test results for the attack detection-and-classification function

No.	Elapsed time (ms)	Result of attack detection and classification	Judgment
1	1	Scanning	OK
2	1	Message tampering	OK
3	1	Unauthorized message insertion	OK
4	1	Illegal authentication	OK
5	200	Message blocking	OK
6	200	Message blocking	OK
	226	Hardware attack	
7	1	Scanning	OK
	38	Message blocking	
	1419	Communication denial of service	
8	1	Unauthorized message insertion	OK
	27	Message blocking	
	376	Control denial of service	
9	—	Security log tampering	OK
10	—	Firmware tampering	OK
11	—	Configuration data tampering	OK

For each of Test Cases Nos. 1 to 11, the type of attack detected and classified and the elapsed time from the start of attack until its detection and classification are shown in the above table. Although some types of attacks were first detected as another type of attack, they were eventually detected and classified as the expected attack, hence resulting in no problem. The elapsed times for Cases Nos. 9 to 11 are shown as N/A because the attacks in these cases were detected during the initialization process at the start of the model intended for ECUs. The above results confirm that the implemented attack detection-and-classification function correctly worked as supposed to.

7.3 Performance evaluation

This section first shows the confirmed results on the memory usage. The flash ROM capacity used by the attack detection-and-classification function is the total of the data capacity required for the bitmasks shown in Section 6.2 and that required for the algorithm code. The RAM capacity to be used is the amount of space required to save the execution state and is statically reserved. Table 6 shows the results of memory usage measurement.

Table 6 Results of memory usage measurement by the attack detection-and-classification function

Item	Memory usage	
	Requirement	Performance
Flash ROM	Below 100 KB	6.2 KB
RAM	Below 12.8 KB	8.4 KB

Both the flash ROM usage and the RAM usage are below 10 percent of the respective total capacities of the flash ROM and the RAM shown in Table 2. Thus, the requirements are met. The confirmed results on the processing time are as follows:

We measured the processing time by observing the digital signal waveforms using the software we modified so that a digital signal output would occur at each switchover of the internal processing. The modification affected the processing time only to a negligible degree. Fig. 9 shows the results of the processing time measurement. Even when the maximum processing time was reached and hence the communication load was 100 percent, the requirement of 1 ms or less was met with a safe margin. These results confirm that the implemented attack detection-and-classification function has performance sufficient to run in real time without disrupting the inherent function of the embedded device.

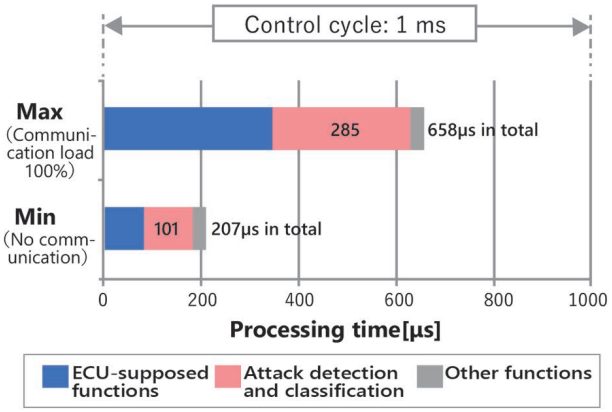


Fig. 9 Processing time in the model intended for in-vehicle ECUs

From the above, we believe that the algorithm implemented this time is a practically feasible technical solution to detect and classify attacks without compromising the inherent function of even a low-resource embedded device.

8. Conclusions

As a solution to the challenge of performing attack detection and classification on embedded devices in real time, we developed a high-speed, lightweight attack detection-and-classification algorithm able to run on embedded devices. Additionally, we improved the attack detection-and-classification algorithm for a finer classification of attacks and the proper selection of appropriate actions.

Moreover, we implemented the developed algorithm in actual embedded-device hardware to run functional and performance tests. The test results have shown that the algorithm has performance sufficient to run in real time while achieving a finer classification of attacks.

With this technology implemented in an embedded device, all the necessary steps up to emergency response actions would be taken immediately after the encounter with a security attack. Moreover, this technology allows the fine classification of attacks down to an appropriate degree of granularity and hence enables automatic selection and execution of an emergency response action matching the type of attack encountered.

When, in the future, we can set appropriate emergency response actions for the types of attacks, we will be able to minimize damage from security attacks on embedded devices.

The development presented here was intended for implementation in in-vehicle ECUs and hence was customized to the needs of in-vehicle ECUs with the focus mainly on the anomaly detection method and scheduling. From now on, we intend to deploy this technology to embedded devices in areas/fields other than in-vehicle ECUs.

References

1) C. Miller and C. Valasek. “Remote Exploitation of an Unaltered Passenger Vehicle,” ilmatics.com, http://ilmatics.com/RemoteCar Hacking.pdf, (accessed Nov. 27, 2019).
2) A. Shingo, “Recent Threats to Control Systems and Activities of JPCERT/CC” (in Japanese), 5th SICE Multi-symp. Control Syst., http://mscs2018.sice-ctrl.jp/program/_tutorials/mscs2018tutorial_ abe.pdf, (accessed Nov. 19, 2019).
3) National Institute of Standards and Technology, “Framework for Improving Critical Infrastructure Cybersecurity,” Cybersecurity Framework, https://www.nist.gov/cyberframework/framework, (accessed Nov. 27, 2019).
4) Blue Planet-Works, “Product Profile: AppGuard” (in Japanese), AppGuard, https://www.blueplanet-works.com/solution/appguard.html, (accessed Nov. 20, 2019).
5) H. Naoki, “Introduction to Attack Detection and Classification Technologies for General-Purpose Embedded Devices” (in Japanese), in *Proc. 4th IoT Security Forum*, Tokyo, Japan, Jul. 30, 2019.
6) Microsoft. “The STRIDE Threat Model,” https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878 (v=cs.20), (accessed Jan. 8, 2020).
7) T. Kogawara, “Proposal for an Attack State Transition Approach to Attack Detection and Classification Technologies for Embedded Devices.” (in Japanese), in *Proc. SCIS2020*, Kochi, Japan, Jan. 20, 2020, no. 2E2−4.

About the Authors

KOGAWARA Toru
Technology Research Center
Technology and Intellectual Property H.Q.
Speciality: Software Science

YAMAMOTO Taisei
Technology Research Center
Technology and Intellectual Property H.Q.
Speciality:Electrical, Electronical and Information engineering

ZENG Zhen
Technology Research Center
Technology and Intellectual Property H.Q.
Speciality: Software Engineering

HIROBE Naoki
Technology Research Center
Technology and Intellectual Property H.Q.
Speciality: Software Engineering