Car Hacking and Defense Competition on In-Vehicle Network

Hyunjae Kang, Byung Il Kwak, Young Hun Lee, Haneol Lee, Hwejae Lee and Huy Kang Kim
School of Cybersecurity, Korea University
{trifle19, kwacka12, dlddudfkr, roytravel, hwejae94, cenda}@korea.ac.kr

***Abstract*—Cybersecurity competitions can promote the importance of security and discover talented researchers. We hosted the Car Hacking: Attack & Defense Challenge from September 14, 2020 to November 27, 2020, and many security companies and researchers participated. To the best of our knowledge, it is the first competition to contest both attack and detection techniques on an in-vehicle network, specifically Controller Area Network (CAN). The participants developed various injection attacks and high-performance detection algorithms based on the real vehicle environment. Rule-based and ensemble tree-based models dominated the final round. Also, time interval and data byte patterns worked as major features to detect attacks.**

I. Introduction

The development of connected vehicles inevitably increases security threats to the vehicle. Modern vehicles contain three main components, in-vehicle systems, external communication interfaces such as Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2I), and internet-based applications, to provide more innovative driving experiences like autonomous driving [3]. Unlike traditional vehicles, an in-vehicle network such as Controller Area Network (CAN) [14] is no longer isolated from an external attacker's threats. As one of the famous hacking cases, Miller and Valasek demonstrated that an attacker could control the 2014 Jeep Cherokee by obtaining access to the CAN network over a compromised radio [13].

In cybersecurity area, competitions to test hacking and defense skills are being held worldwide. Such competitions can promote the importance of security, discover professionals, and educate students in an efficient and attractive way [4]. However, there are only a few competitions held in car security area. Pwn2Own, a cybersecurity conference with hacking contests on popularly used software, included automotive category in 2019 and 2020 [5]. They used a Tesla Model 3 as a target and encouraged researchers to exploit the modem, infotainment, gateway, CAN bus of the vehicle. Also, SINCON 2020 conference opened a Capture The Flag (CTF) style competition on car security [8]. Each team could remotely access a virtual simulated vehicle environment or physical test bench and solved questions related to the CAN bus. Despite the positive effects of competitions, holding a car security competition has challenges due to difficulties of vehicle testbed set-up and lack of researchers in the related field.

Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2021
25 February 2021, Virtual
ISBN 1-891562-68-1
https://dx.doi.org/10.14722/autosec.2021.23035
www.ndss-symposium.org

The purpose of this paper is to share our experience of holding a car security competition. We hosted the Car Hacking: Attack & Defense Challenge from September 14, 2020 to November 27, 2020, in South Korea. The competition problem was to develop attacks and detection algorithms for CAN, a widely used standard of in-vehicle communication. The participants earned scores when they detect other team's attacks and inject critical or stealth attacks. The most significant difference from the events of Pwn2Own and SINCON is that we held attack and detection competitions simultaneously.

The contributions of our competition are listed below:

- We used a commercial car in the competition: Hyundai Avante CN7, a model released in 2020. The participants could inject attacks and see the following effects of a real vehicle.
- The competition aimed to challenge participant's attacks and detection algorithms concurrently. On the day of the main contest, we captured the CAN traffic while the red team injects attack messages into the car and transmitted the traffic to the rest of the teams' (i.e., blue teams) detection systems. It is the first attempt to contest both attack and detection skills in the same car security contest to the best of our knowledge.
- We ran a testbed with real vehicles before the finals, so even individual participants who do not have equipment could use the environment to inject and analyze CAN traffic. We believe this opportunity helped increasing interest in car security to researchers and students.
- We could test and compare attack and detect methods of many research teams, including companies and universities in car security field. They showed statistical rule-based approach is effective to detect attacks, and reducing frequency of attack messages helps to avoid detection systems.

II. Background

A. In-vehicle network

Most vehicles have standard protocols, i.e., LIN, CAN, MOST, and FlexRay, to communicate with in-vehicle nodes. In those protocols, the CAN protocol has high-integrity data communications for real-time applications, reliability, and excellent error detection. For such reasons, most common vehicles apply CAN to an in-vehicle network to transmit sensor data between the nodes: Electronic Control Unit (ECU), microcontrollers, and sensors [14]. It has characteristics such as multi-master bus access, bus topology, and message priority. An accessed CAN bus node can transmit CAN message without any configuration

to communicate with other nodes. Accessed nodes broadcast CAN message because of its bus topology where all nodes share one line. When some nodes send messages into the CAN bus, messages identifier works as a priority of transmission.

B. CAN message injection attacks

The CAN bus' distinctive characteristics could be vulnerable due to no consideration of security. Many injection attacks, such as flooding, fuzzing, spoofing, replay, and bus-off attacks, exploits CAN bus vulnerabilities [7]. The CAN bus has no authentication for source and destination addresses, so the injected data can be processed in normal ECUs without verification. The flooding attack causes CAN message delaying of normal ECUs by sending bunch of messages that has the highest priority (CAN ID 0x000). The fuzzing attack injects random CAN IDs and data. The spoofing attack controls certain vehicle functions by setting specific CAN ID and data field. The replay attack is to inject normal CAN bus traffic, which was collected during driving. Lastly, the bus-off attack is to increase error counts via intentional message collision. When the target node's error count is large enough, the node is set to bus-off status [1].

C. Intrusion detection techniques

Various intrusion detection methodologies have been studied to increase the security of the CAN bus network. These researches could be categorized based on detection layers, i.e., physical layer, data link layer, and application layer. In the physical layer, voltage differences between a normal ECU and an abnormal ECU were used to detect intrusions [2], [9]. In the data link layer, the CAN traffic sequences were used to the deep learning algorithm to detect the in-vehicle intrusion [15]. Moreover, the status features, i.e., the survival analysis of CAN message's frequency [7] and CAN bus's self-similarity [10], were utilized for the anomaly detection algorithms. In the application layer, in-vehicle sensors' correlation was used to check the abnormality with an On-Board Diagnostic (OBD) data [6]. Various deep learning algorithms based on increased computing power have recently been studied to detect in-vehicle network anomalies and intrusions [11].

III. Competition Problems

This competition targeted the CAN bus of Avante vehicle. The CAN bus of the Avante is divided into several sections (e.g., D-CAN, C-CAN, B-CAN) according to its role. To focus on a specific goal, we tapped a DB9 cable directly to the C-CAN high/low wires, like Fig. 2a. C-CAN contains ECUs related to a cluster, motor-driven power steering, airbag control, rear camera, vehicle dynamic control, and several Advanced Driver Assistance Systems (ADAS).

We assumed the scenario that an attacker is already acquired access to the CAN bus. It can happen in practice by hacking the infotainment system or adding a physical node to the CAN bus. It is important to develop an intrusion detection system for the CAN layer because the attacker will eventually inject CAN messages to control the car in many cases. The participants were required to suggest a high-accuracy detection algorithm along with developing attacks (i.e., injecting CAN messages) by analyzing the CAN data. The scope was limited to C-CAN, considering the short period of the competition.

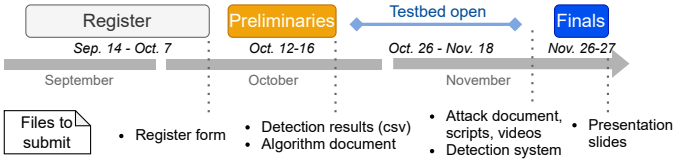


Fig. 1. Car Hacking: Attack & Defense Challenge 2020 timeline

(a) Tapping C-CAN (b) Finals site

Fig. 2. Vehicle settings for the competition. (a) C-CAN wires were tapped to connect a CAN interface device. (b) Cars were lifted up in the finals.

A. Preliminary round

We gave CAN traffic files, including normal and attack messages, to the participants. Each team had to develop an attack detection method and submit the class ('Normal' or 'Attack') of each message. The teams, in order of high detection accuracy, could advance to the finals. For dataset description, see Section IV-A.

B. Final round

The teams were requested to develop five kinds of injection attacks. On the finals, the teams took turns to inject attack messages into the vehicle. Each team had one attack session for 10 minutes to inject their attacks into the vehicle; other teams should submit detection results. Each team's attacks were not shared with other teams; thus, all teams should prepare their detection algorithm to work on unknown attacks. There were some restrictions on the attacks. Teams could conduct only five injection attacks, and the attack effects should be visible in naked eyes to confirm the validity of attacks. Also, unlike preliminary round, attacks injecting random CAN messages such as fuzzing attacks were not allowed in order to prevent mixing no-effect messages.

IV. Running Competition

Overall competition schedule and submission files are briefly described in Fig. 1. Note that the competition was held during the period of the COVID-19 pandemic, so both preliminary and final rounds were held as virtual events. For preliminary, we opened a leaderboard website so the participants could see their scores. For finals, we provided a live broadcast along with the leaderboard.

A. Preliminary round dataset

In the preliminary round, we provided two datasets depending on the purpose: the training set and submission set. Table I shows the number of messages for normal and attack classes. Both datasets contained two status of the target vehicle.

TABLE I. SUMMARY OF PRELIMINARY ROUND DATASET

Purpose	Car Status	Normal	Flooding	Spoofing	Replay	Fuzzing
Training	Driving	1,724,630 (92.0%)	77,373 (4.1%)	3,879 (0.2%)	23,775 (1.3%)	45,474 (2.4%)
	Stationary	1,648,113 (91.7%)	76,807 (4.3%)	3,877 (0.2%)	23,818 (1.3%)	44,405 (2.5%)
Submission	Driving	1,799,046 (89.9%)	96,559 (4.8%)	22,489 (1.1%)	37,869 (1.9%)	44,770 (2.2%)
	Stationary	1,559,164 (89.0%)	95,120 (5.4%)	20,094 (1.1%)	25,012 (1.4%)	51,923 (3.0%)

TABLE II. EXAMPLES OF TRAINING DATA

Timestamp	Arbitration ID	DLC	Data	Class	SubClass
1597759747.427094	507	4	08 00 00 01	Normal	Normal
1597759747.427486	000	8	00 00 00 00 00 00 00 00	Attack	Flooding
1597759747.427742	356	8	00 00 00 80 1D 00 00 00	Normal	Normal

Driving. The data extracted when driving, including general driving motions (e.g., accelerate, braking, turning the steering wheel).
Stationary. The data extracted when the car engine is on but not moving (park gear), including subsidiary motions (e.g., pressing buttons of the center console).

In each state, there were four types of attacks:
Flooding. Flooding attack aims to consume the CAN bus bandwidth by sending a massive number of messages. We transmitted messages having the lowest value of CAN ID (0x000) to take advantage of the priority mechanism.
Spoofing. An attacker injects CAN messages to control the desired function after reversing the vehicle traffic. Several spoofing attacks were performed in the preliminaries; 1) factory mode activated warning, 2) printing higher RPM gauge, 3) engine off warning, 4) blind spot collision warning, 5) sudden turn-on of the rear camera screen. We included 1), 2) to the training set, and 3), 4), 5) to the test set; different CAN IDs were used in the test set to increase detection difficulty.
Replay. Replay attack is to extract normal traffic at a specific time and replay (inject) it into the CAN bus. It is relatively hard to make changes to a specific function, but much easier than spoofing attack because it does not require hard reversing. When we replayed the traffic, some warnings flashed on the cluster, and odd reactions appeared when driving the vehicle.
Fuzzing. If an attacker does not have sufficient information on the values of CAN IDs and the data field, he may inject random messages. When we transmitted randomly generated data of several arbitrary IDs, random warnings flashed on the cluster, and occasionally it caused sudden stop or flinching of the steering wheel.

As shown in Table II, the training dataset was CSV format files consisting of 6 fields: Timestamp, Arbitration ID (i.e., CAN ID), DLC, Data, Class, and SubClass. Timestamp indicates the Unix time when the CAN message was logged. CAN ID is a hex value of the identifier. DLC is an integer value of the data length code. Data is hex values of the data field, each byte separated by a space. Class and subclass are fields to indicate normal or attack, and type of attack of the CAN message.

In the submission dataset, we deleted Class and SubClass fields and added Number field indicating the row number in front of the Timestamp field. Participants submitted the predicted Class (Normal of Attack) of each CAN messages. Note that predicting SubClass (i.e., attack type classification) was not required in the competition to consider scalability, because participants may carry out any type of unknown attacks in the finals.

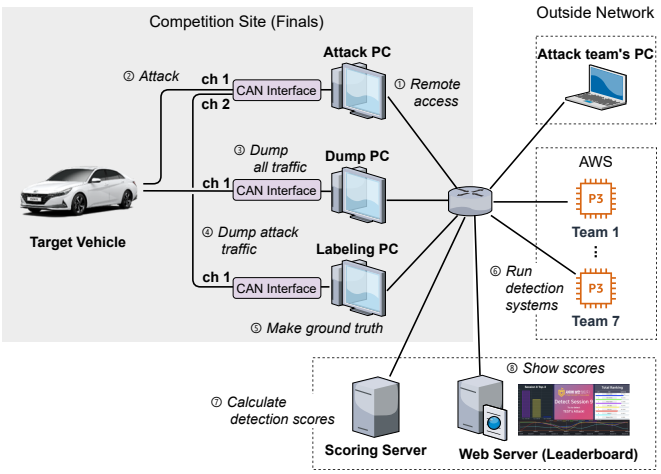


Fig. 3. Overview of final round systems and process

The preliminary round datasets are available for research purposes on our website.[1]

B. Systems and process for final round

In the finals, all teams had one attack session (10 minutes) in turn to inject five types of prepared attacks. In each attack session, all teams' attack detection systems, including the attack team's system, submitted the predicted Class of each CAN messages collected.

The overall systems and process is shown in Fig. 3. To hold attack and detection competition simultaneously, we considered the following conditions when building the finals environment:

- All C-CAN traffic of the vehicle and only attack traffic should be captured separately.
- Creating ground truth should be automated.
- Running each team's detection system and evaluation should be automated.

1) Performing attacks: For safety, the target vehicle was lifted up and kept stationary status. Participants may change the vehicle's settings except for drive or rear gear (e.g., park or neutral gear, turning the steering handle, opening/closing doors). Staffs changed the settings at the site if requested since the participants remotely accessed the attack PC because of the pandemic. Each team's pre-submitted attack scripts were downloaded on the PC, so the participants could run the scripts to send attack messages to the vehicle. The PC was connected to the vehicle through a CAN to USB interface product of Kvaser's. The interface supported a dual CAN channel, and the first channel was cabled with the vehicle's C-CAN.

2) Collecting CAN data: The CAN message frame does not contain sender information, so it is hard to distinguish attack messages only from the dumped traffic of the vehicle's CAN. We revised attack scripts after submission to make them re-send duplicate messages to the second channel to solve this problem. The second channel was connected to another CAN interface of other PC ("Labeling PC" in Fig. 3). Since it is a separate bus network from the vehicle's network, we could collect attack messages only. Also, another PC ("Dump PC") collected all traffic from the vehicle.

[1] https://ocslab.hksecurity.net/Datasets/carchallenge2020

TABLE III. COMPETITION PERFORMANCE EVALUATION INDICATORS

Category	Evaluation Indicators		Proportion		Remarks
Attack Score	Visibility of attack impact		Y/N		Qualitative
	Number of attack kinds		6%	30%	Quantitative
	Different attacks from preliminary round		6%		Quantitative
	Severity of attacks		6%		Qualitative
	Detection difficulty		12%		Quantitative
Detection Score	Attack detection accuracy (F1-score)	Host's session	10%	50%	Quantitative
		Participant's session	40%		
Presentation	Understanding of CAN data and validity of idea-making process		4%	20%	Qualitative
	Features, advantages, and differentiation of detection algorithm		8%		
	Features, advantages, and differentiation of attack techniques		8%		

3) Making ground truth: After each attack session was finished, the dump PC sends dumped traffic to the labeling PC. The labeling program flagged certain messages as "Attack" if the message could be found in attack traffic, dumped from the other channel. Timestamp, arbitration ID, DLC, and data fields were compared to determine the attack messages.

4) Running detection systems and scoring: Each team installed their attack detection system on an individual cloud server. We provided Amazon EC2 P3.2xlarge instance[2], and the teams were able to set up their system for two days before the finals. The system should be automated and the participants were inaccessible to the server on the day of the final. After each attack session was finished, we transmitted unlabeled CAN traffic and ran all detection servers in parallel. The servers submitted detection results and run times to a scoring server. The submission time limit was 20 minutes, twice the time of an attack session. The scoring server calculated detection scores by using the ground truth from the labeling PC, and finally, it uploads each team's detection score on a leaderboard. Considering the time required for data transmitting and processing, participants could check their scores in at most 25 minutes.

C. Scoring

We conducted quantitative and qualitative evaluations together for a comprehensive assessment. All evaluation indicators and distribution is on Table III.

1) Attack score: When making the indicators, we deliberated on how to tempt teams to make stealth and critical attacks; 'detection difficulty' and 'severity of attacks' are the related indicators. Since there are temporal constraints to hold a competition, we limited the attack ranges to the cases where we can visually identify the attack impact.

Visibility of attack impact. The impact of every attack should be visible; therefore, the staff could recognize that the team is injecting valid attack messages. If the staff cannot confirm the impact, the corresponding attempt is not counted as an attack.

Number of attack kinds. The teams could prepare at most 5 different attacks. The 'kind' of attack is distinguished as if an attack affects different functions of the vehicle, regardless of attack types such as flooding, spoofing, replay, and fuzzing. For example, if a first attack changes an RPM gauge value and a second attack turns a steering wheel, both attacks are counted as different.

Different attacks from preliminary round. This indicator is included to encourage participants to make different kinds of attacks from preliminary round datasets. If a team reuses the attack that is already performed in the preliminary round, this score will be deducted.

Severity of attacks. If an attack arises malfunctioning of essential driving functions such as controlling a brake or direction of wheels, the team gets a higher score.

Detection difficulty. If a team performs attacks that are difficult to detect from other teams' detection systems, the team gets points. From an attacker's point of view, it is important to carry out silent attacks that an IDS may miss. Note that this score is calculated on a session basis, unlike the above indicators, which are scored on each attack. If n is the total number of teams and F_i is the F1-score of team i, the red team gets an inverse proportion of all teams' F1-score average:

$$1 - \frac{\sum_{i=1}^{n} F_i}{n}$$

2) Detection score: We used F1-score indicator to assess an accuracy of detection algorithms. F1-score is a harmonic mean of precision and recall, where precision (recall) is the number of correctly identified attack messages divided by the number of all messages predicted as attack (divided by the number of all real attack messages). It can measure detection performance objectively even when normal and attack data distribution is unbalanced:

$$\text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

For the preliminaries, only the detection score of the submission dataset was used. For the finals, the detection scores of all attack sessions were aggregated. There was a host's session; we injected attacks similar to the preliminaries to assess teams' basic detection performances. Also, each team submitted results for all team's sessions, including their own, and the weighted sum of F1-scores were the final score.

3) Presentation: Each team made a presentation for 25 minutes (15 min presentation, 10 min Q&A) on the 2nd day of finals, explaining the detection algorithm and prepared attacks. The evaluation committee composed of 6 experts estimated qualitative factors such as background knowledge level of CAN or in-vehicle IDS, the effectiveness of the proposed detection algorithm, and efforts to develop attack techniques.

D. Running testbed

A testbed, a real vehicle environment, was provided before the finals to test their skills and collect real data. We operated it from October 26, 2020 to November 18, 2020. Two cars were placed so that two teams could visit at the same time. Kvaser CAN interface, the same device used in finals, was equipped, and sample codes of reading and transmitting CAN messages were provided. For safety reasons, participants were only allowed to inject or collect data when the car was stationary.

[2]The instance's specifications are one NVIDIA Tesla V100 GPU, 16GB GPU memory, 8 virtual CPUs, and 61GB memory.

TABLE IV. COMPETITION RESULTS: DETECTION SCORES, ATTACK SCORES, AND DETECTION ALGORITHMS

Team	Fin. avg. F1-score	Fin. avg. detection time (s)	Fin. attack score	Detection algorithm
A	**0.869**	**33**	71.0	Rule-based
B	0.864	155	57.2	XGBoost, Light GBM, rule-based
C	0.816	355	**79.6**	Random Forest, white & black list
D	0.812	257	66.7	Light GBM
E	0.675	295	63.4	Rule-based, Random Forest
F	0.445	147	38.8	Random Forest, Light GBM
G	0	114	51.7	DBSCAN

Instead, they could request us to dump and share data while driving normal roadway with videos filmed driver actions and front views. Occasionally, wrong alarms were often turned on in the cluster when unusual CAN messages were injected, and they did not disappear by turning the car off and on. Every day after the testbed closed and anytime users requested, our staff drove the car for a moment to refresh sensors.

V. COMPETITION RESULTS

A. Summary

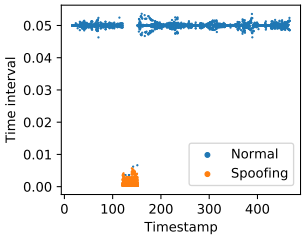
We had 25 registrations on the preliminary round, and 19 teams submitted results. Top 7 teams entered finals; two security companies, one university (professor and graduate students), one research institute, and three individual teams composed of undergraduate students or office workers. Table IV shows the final teams' overall scores and detection algorithms. We listed the teams in order of finals detection score. As shown in Table IV, Participant's detection scores decreased significantly compared to the preliminary round; -0.13 points based on the highest score. We surmised that there were two main reasons; 1) any information of attacks prepared by participants were not shared with other teams–unlike training sets provided in preliminaries, 2) the requests of the team varied vehicle settings during the attack session, so it increased diversity of normal traffic.

1) Detection algorithms: Interestingly, all teams except Team G used rule-based or tree-based ensemble classifiers (e.g., Random Forest, XGBoost, Light GBM) in finals. While many teams also used tree-based models in the preliminary round, five teams out of 19 submitted teams used deep learning models, such as BERT, CNN, autoencoder, and LSTM. However, even the teams that used deep learning models changed their algorithms to rule-based or tree-based models in finals. Final round teams preferred such models because of fast inference speed to submit results within the time limit (20 min). Also, the models did not lack detection performance compared to deep learning models in our competition environment.

2) Features: There were two major features that the final teams focused on–time interval and data field patterns.

Time interval. Most of the CAN IDs have regular transmitting cycle in normal status. When attack messages targeted on certain CAN ID are injected, time interval values becomes much shorter than the normal cycle, as shown in Fig. 4a.

Data field patterns. Several CAN IDs have constant, counter, or CRC (Cyclic Redundancy Check) bytes in the data field [12]. When an attack message is injected between normal messages, it breaks the pattern of such data bytes. For example, Fig. 4b shows abnormal counter bytes when an attack message



(a) Time interval example

Timestamp	ID	Data							
120.78377	490	00	00	08	21	00	00	3C	7C
120.83318	490	00	00	08	21	00	10	3C	C8
120.88338	490	00	00	08	21	00	20	3C	09
120.90583	490	03	00	08	21	00	00	3C	7C
120.93318	490	00	00	08	21	00	30	3C	BD
120.98348	490	00	00	08	21	00	40	3C	96
120.03387	490	00	00	08	21	00	50	3C	22

(b) Counter example

Fig. 4. Feature values observed in CAN ID 0x490. (a) shows a scatter plot of time interval values. Normal messages have regular intervals around 0.05 sec, while spoofing messages have intervals smaller than 0.005 sec. (b) shows sample data bytes of counter pattern. The red row is a spoofing message. It shows a wrong counter value while the data byte of normal messages increases 0x10 each time.

is injected. Rule-based models used certain condition of CAN ID, data byte, and type of pattern (e.g., constant, counter) while developing the rules. Tree-based models used data byte difference from last same CAN ID message as a feature, so they could train the pattern of change in data bytes.

3) Proposed attacks: To achieve high attack scores in the competition, severity of attacks and detection difficulty are major indicators.

Severity of attacks. The attacks were divided into two main categories: controlling powertrain and false alert. Attacks that could make changes on powertrain modules, such as steering wheel and brake, got high severity scores. These attacks may cause critical accidents by controlling the car's movement. False alerts are the attacks that spoof signs on the dashboard. Various scenarios were proposed, e.g., hiding door open sign, showing wrong gear sign, turning on the ADAS signs or warnings. The purpose of these attacks was to confuse the driver and interfere with driving. They got relatively low severity scores than powertrain related attacks.

Detection difficulty. To avoid detection, the process of reducing the frequency of attack messages and making them similar to normal messages was important. Regardless of the severity, the low frequency decreased time interval violation and the probability of detected by other teams' algorithms. Also, avoiding unused IDs and data bytes also helped.

B. Team A's detection algorithm (detection score rank 1)

Team A used only rule-based algorithm. Their algorithm mainly worked as following steps; 1) detect unused CAN ID and DLC, 2) check each data byte whether specific byte has wrong constant, counter, or CRC value, 3) check if time interval of packets is outside of normal range. These features were commonly used by other teams, though their strength was thorough analysis of each CAN ID and data bytes; they tried to discover each byte's pattern type (e.g., constant, counter, CRC, or reflecting sensor input) for all CAN IDs. Interestingly, it was the fastest algorithm among all teams; it took only 33 seconds in average to submit detection results of 10 min traffic, while other teams took 1-6 minutes generally.

C. Team C's attacks (attack score rank 1)

Team C endeavored to find attacks that could succeed with the minimum number of messages. They repeated fuzzing on

CAN ID and data fields to find messages that effects functions such as steering wheel, brake, autohold, and cruise control. Also, they tried ID elimination test–checking the car's reaction after eliminating certain ID from the collected traffic and replaying it. These process made them available to find the exact message that could control the car with small amount of messages. They injected only 2,000 messages while other team's injected at least 6,000 to at most 160,000 messages. As result, they got remarkable score on the detection difficulty.

VI. Discussion: Limitations and Potential Improvements

As we held data-driven hacking and defense style competition in the car security field for the first time, we would like to mention the limitations and factors that could be improved in the future.

1) Limited vehicle status: Using a commercial vehicle enables developing attacks and detection algorithms that work in practice; however, it inevitably causes cost and safety problems. The non-driving status was only allowed at the finals and the testbed because of safety issues such as driver injury due to unexpected movement while testing attacks. Therefore, some attacks, like changing the vehicle's speed by malfunctioning ADAS systems, were hard to develop in this competition. A proven testing place with a chassis dynamometer or a track should be reserved to provide a driving status. Otherwise, we may consider building a simulator similar to the real car environment. Also, the contest hall for the finals was an indoor place, so there were difficulties in resetting the wrong alarms on the car by driving. We removed the remaining error codes via the OBD-II scanner and disconnected the battery for more than 10 min to solve this problem; this process was repeated between all attack sessions.

2) Accessibility to a real vehicle: A real car is a costly device, especially for individuals. Although few teams owned or rented the same vehicle model, the other teams had to test their skills within a limited period at the testbed (at most two days per team). For a future competition, it is recommended to increase the testbed providing period more sufficiently by considering the schedule from the planning stage.

3) Extending the scope of challenge: As we mentioned in chapter III, only C-CAN was the competition's scope. We may consider extending the scope by adding other sections: P-CAN (connected to engine/transmission/powertrain, fuel injection pump, vehicle dynamic control modules, etc.) and B-CAN (connected to a seat position control module, etc.). Also, considering other important layers of a vehicle, such as Bluetooth/USB modules, infotainment, and gateway devices, would be challenging but valuable competition items.

VII. Conclusion

The Car Hacking: Attack & Defense Challenge 2020 is the first car security competition that held both attack track and detection track concurrently. Several security companies, researchers, and students participated and tested their detection system performance in the real-car environment. Surprisingly, the top-ranked detection algorithm was a rule-based model. The model scored 0.87 F1-score on binary classification per message ('Normal' or 'Attack'). It used precise rules of time intervals and data byte patterns of each CAN ID. The team with the highest attack score earned both great scores on severity and detection difficulty. They tried to find out the exact CAN message that could control the car with a small amount of injection.

There is a lot of room to expand the competition, such as targeting multiple vehicle models or preparing more various vehicle status. Continuing to hold competitions will help discover talented researchers and activate automotive security research.

Acknowledgment

This work was supported by Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00866, Challenges for next generation security R&D).

References

[1] 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*, 2016, pp. 1044–1055.

[2] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "Voltageids: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2114–2129, 2018.

[3] J. den Hartog, N. Zannone *et al.*, "Security and privacy for innovative automotive applications: A survey," *Computer Communications*, vol. 132, pp. 17–41, 2018.

[4] E. Gavas, N. Memon, and D. Britton, "Winning cybersecurity one challenge at a time," *IEEE Security & Privacy*, vol. 10, no. 4, pp. 75–79, 2012.

[5] B. Gorenc, "Pwn2Own returns to vancouver for 2020," https://www.zerodayinitiative.com/blog/2020/1/8/pwn2own-returns-to-vancouver-for-2020, accessed on: Jan. 11, 2021.

[6] F. Guo, Z. Wang, S. Du, H. Li, H. Zhu, Q. Pei, Z. Cao, and J. Zhao, "Detecting vehicle anomaly in the edge via sensor consistency and frequency characteristic," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5618–5628, 2019.

[7] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular communications*, vol. 14, pp. 52–63, 2018.

[8] Infosec In the City, "Overview of SINCON car security kampung," https://www.infosec-city.com/post/sin20-ctf-car-security, accessed on: Jan. 11, 2021.

[9] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 787–800.

[10] B. I. Kwak, M. L. Han, and H. K. Kim, "Cosine similarity based anomaly detection methodology for the can bus," *Expert Systems with Applications*, vol. 166, p. 114066, 2021.

[11] Y. Lin, C. Chen, F. Xiao, O. Avatefipour, K. Alsubhi, and A. Yunianta, "An evolutionary deep learning anomaly detection framework for in-vehicle networks-can bus," *IEEE Transactions on Industry Applications*, 2020.

[12] M. Markovitz and A. Wool, "Field classification, modeling and anomaly detection in unknown can bus networks," *Vehicular Communications*, vol. 9, pp. 43–52, 2017.

[13] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.

[14] Robert Bosch GmbH, "CAN specification 2.0," http://esd.cs.ucr.edu/webres/can20.pdf, 1991, accessed on: Jan. 11, 2021.

[15] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020.