Infotainment System Matters: Understanding the Impact and Implications of In-Vehicle Infotainment System Hacking with Automotive Grade Linux

Seonghoon Jeong
School of Cybersecurity, Korea University
Seoul, Republic of Korea
seonghoon@korea.ac.kr

Minsoo Ryu
School of Cybersecurity, Korea University
Seoul, Republic of Korea
onsoim@korea.ac.kr

Hyunjae Kang
School of Cybersecurity, Korea University
Seoul, Republic of Korea
trifle19@korea.ac.kr

Huy Kang Kim
School of Cybersecurity, Korea University
Seoul, Republic of Korea
cenda@korea.ac.kr

ABSTRACT

An in-vehicle infotainment (IVI) system is connected to heterogeneous networks such as Controller Area Network bus, Bluetooth, Wi-Fi, cellular, and other vehicle-to-everything communications. An IVI system has control of a connected vehicle and deals with privacy-sensitive information like current geolocation and destination, phonebook, SMS, and driver's voice. Several offensive studies have been conducted on IVI systems of commercialized vehicles to show the feasibility of car hacking. However, to date, there has been no comprehensive analysis of the impact and implications of IVI system exploitations. To understand security and privacy concerns, we provide our experience hosting an IVI system hacking competition, Cyber Security Challenge 2021 (CSC2021). We use a feature-flavored infotainment operating system, Automotive Grade Linux (AGL). The participants gathered and submitted 33 reproducible and verified proofs-of-concept exploit codes targeting 11 components of the AGL-based IVI testbed. The participants exploited four vulnerabilities to steal various data, manipulate the IVI system, and cause a denial of service. The data leakage includes privacy, personally identifiable information, and cabin voice. The participants proved lateral movement to electronic control units and smartphones. We conclude with lessons learned with three mitigation strategies to enhance the security of the IVI system.

CCS CONCEPTS

• **Security and privacy** → **Penetration testing**; *Mobile platform security*; *Privacy protections*.

KEYWORDS

car hacking, cybersecurity competition, vulnerability, exploit, Automotive Grade Linux, privacy leakage

CODASPY '23, April 24–26, 2023, Charlotte, NC, USA

ACM ISBN 979-8-4007-0067-5/23/04…$15.00
https://doi.org/10.1145/3577923.3583650

ACM Reference Format:
Seonghoon Jeong, Minsoo Ryu, Hyunjae Kang, and Huy Kang Kim. 2023. Infotainment System Matters: Understanding the Impact and Implications of In-Vehicle Infotainment System Hacking with Automotive Grade Linux. In *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy (CODASPY '23), April 24–26, 2023, Charlotte, NC, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3577923.3583650

1 INTRODUCTION

In-Vehicle Infotainment (IVI) systems play an important role in the way drivers interact with state-of-the-art vehicles to navigate, utilize convenient features, and adjust autonomous driving features. For example, the IVI system in Tesla Model 3 is an exemplary one that takes charge of nearly all features—gear selection, powertrain/chassis management, seat positioning, heating ventilation, and air conditioning (HVAC), music streaming, autonomous driving, over-the-air software update, and so on. With the evolution of connectivity and computerized automotive technology [38], IVI systems have become an essential component of modern vehicles [12, 17]. An IVI system is tightly coupled to a vehicle via in-vehicle networks (IVNs), such as a Controller Area Network (CAN) bus and automotive Ethernet, to communicate with electronic control units (ECUs). In addition, an IVI system is equipped with external interfaces such as Bluetooth and Wi-Fi [29].

Today's IVI systems are becoming more complex than ever in both hardware and software aspects. In particular, the complex electrical/electronic (E/E) architecture broadens attack surfaces and the scope of affected components (e.g., ECUs). Infotainment operating systems (OSs) and libraries therein used to develop infotainment Apps may inherit vulnerabilities from other domains. Consequently, cybersecurity issues are emerging with IVI systems. Some studies have proved that vehicle hacking could start from a vulnerable IVI system.

IVI system security is fundamental to keeping a safe and secure driving environment in the vehicle cybersecurity domain. More than one hundred studies referenced in survey papers [15, 18, 37] acknowledged an adversary who might exploit vulnerable IVI systems in order to inject CAN messages remotely to tamper a target vehicle. Indeed, the adversary can cause severe consequences, including a car crash. However, a systematic view of the impact and implications is lacking. We should recognize them because today's

IVI systems handle many contents and sensitive data. For example, an IVI system deals with a vehicle geolocation and driver's favorite destinations. An IVI system can access the phonebook and text messages if it is paired with a smartphone. An IVI system listens to cabin voice for speech recognition and hands-free call. Previous studies are often unconcerned with audio systems, seat positioning, and HVAC, even though they could harass a driver. Such contents and features are sufficient to attract attackers.

So far, only a few studies have considered the IVI system security due to challenges in the experiment environment. To be more specific, there are two hurdles regarding the challenge. First, obtaining a testbed is not always feasible. An IVI system runs on neither a PC nor a general-purpose hardware development kit since an IVI system is supposed to be a part of integrated E/E architecture. Utilizing an actual vehicle is a good solution to obtain an experiment testbed only if legal and financial concerns are adequately resolved. Second, findings distilled from a study are not always applicable across multiple types of vehicles because each car maker customizes the hardware and software as they wish. Consequently, neither a general framework for assessing IVI systems nor a reproduction of prior works exists. As a result, we are not fully aware of risks of IVI system hacking.

In this paper, we report an in-depth analysis of the impact and implications of IVI system hacking based on 33 verified proof-of-concept (PoC) exploit codes. We carefully conceive and design a cybersecurity competition for IVI system hacking to leverage collective intelligence. To tackle the challenge mentioned above and share the same experimental testbed between participants, we utilize an integrated IVI system powered by Automotive Grade Linux (AGL) [33], a feature-flavored infotainment OS maintained by Linux Foundation. Thanks to the high-quality infotainment Apps, virtual CAN bus, and background services of AGL, we could conduct the study without an actual vehicle. Our study aims at readers trying to understand the security aspects and possible consequences on IVI systems. To the best of our knowledge, this is the first study providing various implications of IVI hacking based on reproducible impacts.

Our contributions are summarized as follows:

1. IVI System Hacking Competition. We hosted a cybersecurity competition to discuss IVI system security. We deliver the massive study result performed on the AGL-based IVI system. We summarize the competition result with 33 verified exploit codes submitted from 102 participants (=26 teams) working in the field of cybersecurity.

2. Vulnerability Identification. We identified three new critical vulnerabilities in 10 IVI services of AGL release version 11.0.0 (Kooky Koi)–12.1.6 (Lucky Lamprey), which can be exploited remotely. We reported the vulnerabilities as CVE-2022-24595, CVE-2022-24596, and CVE-2022-24597. In addition, we confirm that the AGL release inherits one vulnerability from Linux Bluetooth library.

3. Impact and Implications. Our study shows the impact and implications including (1) lateral movement to driver's smartphone as well as ECUs via a CAN bus, (2) sensitive content leakage—cabin voice, credentials (Wi-Fi password or PIN code), and personal identifiable information, (3) harassment of passengers—providing a fake map/route, playing a loud noise—which may cause a car accident, and (4) crash of target IVI service.

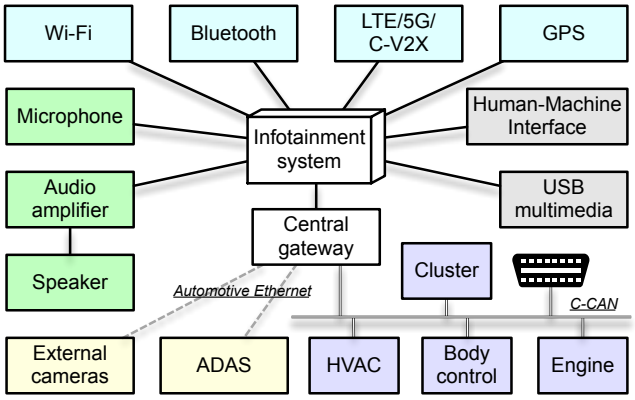


Figure 1: Overview of IVI system and E/E architecture

4. Root Cause Analysis. The root causes of the vulnerabilities imply the lack of security considerations for IVI systems. Moreover, we identified a coarse-grained permission (on two services *afbd-homescreen* and *bluetoothd*) maximizes the implications when a vulnerability is exploited.

The remainder of this paper is organized as follows: In §2, we introduce an IVI system and related work. In §3, we discuss our strategy to hold the cybersecurity competition. In §4, we evaluate the submissions (i.e., PoC codes). The implications are discussed in §5. In §6, we discuss the lessons learned and limitation of our work. Finally, §7 concludes our work.

2 BACKGROUND

2.1 In-Vehicle Infotainment System

An IVI system is a complex system that provides convenient features to drivers and passengers in cooperation with various in-vehicle components. Figure 1 shows the general design of the IVI system and E/E architecture, which consists of external interfaces (Wi-Fi, Bluetooth, C-V2X, and GPS), in-vehicle interfaces (HMI and USB port), many ECUs over in-vehicle networks (automotive Ethernet and CAN bus), and audio system. As the IVI system comes with a smartphone-grade CPU and user-convenient human-machine interface (HMI), such as a touch screen, more and more features are being implemented into the IVI system. Consequently, the IVI system becomes the centralized component. IVI systems today even can monitor, manage, and diagnose a vehicle by leveraging connection through in-vehicle networks. A driver's smartphone can be connected to an IVI system via Wi-Fi, Bluetooth, or USB port in order to provide hands-free call, screen mirroring, and music streaming.

Since an IVI system becomes complex and takes responsibility for mission-critical features, the importance of a stable infotainment OS is also increasing. Thus, many recent IVI systems leverage customized third-party OSs that empirically proved outside vehicles. In recent years, variations of existing OSs were introduced for IVI systems. For example, Android Automotive OS (AAOS) [11], QNX [3], and AGL are representative infotainment OSs based on Android OS, BlackBeery OS, and Linux, respectively. These infotainment OSs help car makers drastically reduce efforts in developing middleware for IVI systems. However, they could come with two

Table 1: Related works.

Reference	Attack vector	Impact discussed in the work	IVI system and infotainment OS
2015 [23]	Physical (USB), Remote (TCP/IP)	Overwrite malicious firmware, remote control of the IVI system and the vehicle	Uconnect, QNX
2016 [19]	Physical (USB)	Remote code execution on the OS	Undisclosed model from a vehicle OEM, Windows Embedded CE
2017 [16]	Physical (Android debug bridge tool, SD card)	Overwrite a malicious firmware and IVI App, trigger arbitrary telematics services (e.g., door open, GPS, engine start)	Undisclosed infotainment system in a real car, Android OS
2017 [26]	Remote (web browser renderer)	Injecting CAN messages, remote control of the vehicle physically	Tesla's infotainment system, Linux
2018 [10]	Physical (USB)	Overwrite malicious firmware	Subaru StarLink head units, QNX
2019 [5]	Physical (USB, OBD-II), Remote (web browser renderer, telematics)	Take control of infotainment system, inject CAN messages, reset ECUs	BMW NBT Head Unit, QNX
2019 [1]	Remote (web browser renderer)	Remote code execution on the firmware	Tesla's infotainment system, Linux
2019 [9]	Remote (TCP/IP)	Inject CAN messages	Undisclosed 3rd-party device, Android
2020 [31]	Physical (CAN bus, UART), Remote (Bluetooth)	Overwrite malicious firmware, Remote code execution,	Toyota/Lexus DCU (Display Control Unit) system, Linux
2021 [32]	Physical (Ethernet), Remote (web browser renderer)	Remote control of the IVI system and vehicle	Mercedes-Benz MBUX, MB.OS
2022 [25]	Remote (malicious Android apps)	Remote control of the IVI system, retrieving information of vehicle, tracking geolocation	IVI system unspecified, Android Automotive OS
This work	Remote (TCP/IP), Adjacent (Bluetooth)	Refer to Table 7	Raspberry Pi 4, AGL

concerns: (1) too complex to identify glitches within infotainment OS and (2) vulnerability inheritance. An insecure infotainment OS could jeopardize both an IVI system and entire vehicle. In the following section, we review several offensive studies proving that vehicles can be affected by the exploitations of IVI systems.

2.2 Related Work: Offensive Security Research against IVI Systems

Recent survey studies [15, 18, 37] and references therein imply that research has been growing quickly against cybersecurity issues on IVNs, especially on CAN buses, since 2015. Most references have assumed adversaries inject CAN messages via compromised IVI systems. On the other hand, only a few studies have discussed the severity of IVI hacking with empirical experiments. In this section, we survey the offensive security research on IVI systems to motivate the need for this study. The studies are summarized in Table 1.

Early work by Miller and Valasek [23] jumpstarted hundreds of studies in the vehicle cybersecurity area. This study proved the concept of remote car hacking via a Harman Uconnect IVI system with the 2014 Jeep Cherokee. With their experience obtained from a comprehensive survey on remote attack surfaces of a vehicle [22], they found the exposed Desktop-BUS (D-BUS) service on Transmission Control Protocol (TCP) port 6667 that requires no authentication method. The authors successfully accessed various infotainment services—including GPS, HVAC, radio, and displays. They also demonstrated a CAN message injection attack via modified firmware. Four years later, Miller [21] discussed the lessons learned from the study.

Another famous study by the Tencent Keen Security Lab [26] showed the remote hacking of a Tesla Model S. The study considered an exploit chain of two vulnerabilities: a remote code execution (RCE) vulnerability on QtWebkit and a local privilege escalation vulnerability on Linux Kernel 2.6.36. They successfully achieved a root privilege of the IVI system remotely. Even though the two vulnerabilities had already been registered in the Common Vulnerabilities and Exposures (CVE) database before their work, the authors could exploit the vulnerability chain because the IVI system inherited vulnerabilities from a personal computer. They could physically take control of the vehicle by injecting CAN messages via the compromised IVI system. In 2017, Tesla released a corresponding security update to mitigate vulnerabilities and protect the browser, kernel, and ECUs.

Despite the efforts of car makers, it is difficult to identify and resolve all vulnerabilities in an IVI system. In Pwn2Own 2019, the research team Fluoroacetate [1] demonstrated a remote code execution attack by exploitation of another vulnerability in the Tesla Model 3 web browser renderer. The vulnerability was resolved by Tesla right after the research team demonstrated it.

Tencent Keen Security Lab expanded their work onto BMW and Mercedes-Benz vehicles [5, 32]. Both works contain detailed information about how they analyzed attack surfaces, what the vulnerabilities were, how they exploited the vulnerabilities, and a post-attack procedure (i.e., installing a persistent backdoor). Two studies reported fourteen CVE vulnerabilities which have been resolved with the cooperation of the car makers.

Two studies demonstrated that internal USB ports could be used to attack IVI systems. The first study by Mazloom *et al.* [19] focused on a MirrorLink session between an IVI system and a smartphone. After reverse-engineering USB traffic, they crafted malformed MirrorLink packets in order to cause a heap memory corruption in an IVI OS. The second study authored by Gayou [10] found a vulnerability in the update mechanism of the Subaru StarLink head units from 2017 to 2019. They could overwrite malicious firmware using their USB memory stick and execute arbitrary code as the root user.

In 2017, Jo *et al.* [16] assessed vulnerabilities of an Android OS-based telematics system. They exploited a weak chain of firmware update procedure. More specifically, they manipulated the entire system by using a malicious firmware that they repackaged. Once the system is compromised, an adversary can remotely control it via cellular SMSs.

Some weaknesses or vulnerabilities were inherited due to insufficient security considerations. Costantino and Matteucci [9] targeted an Android debugging service through a TCP port to upload an

Table 2: IVI hacking competition schedule

Event	Important date
1. IVI hacking competition design	March–July, 2021
2. Call for competition [14]	August 2, 2021
3. Submission deadline (the qualification round)	September 15, 2021
4. Notification of acceptance for the final round	October 6, 2021
5. Submission deadline (the final round)	November 3, 2021
6. On-site competition [13]	November 24, 2021

attack script to an Android IVI system. A study [31] found Bluetooth vulnerabilities in IVI systems for Toyota and Lexus vehicles, which allows an attacker within the Bluetooth range to execute an arbitrary command. In 2022, Moiz and Alalfi [25] examined 14 Android mobile vulnerabilities on AAOS. They proved that the vulnerabilities could be utilized to disclose vehicle-related information and manipulate an IVI system through a malicious Android App.

2.3 Necessary and Scope

Without the pioneering studies in Table 1, real cars in the market could be exposed to attacks through vulnerable IVI systems. However, the studies mainly focused on unveiling vulnerabilities and providing PoCs. No further consequences have been discussed except the concern that IVI hacking could physically harm vehicles and passengers. We emphasize that the following questions remain unanswered:

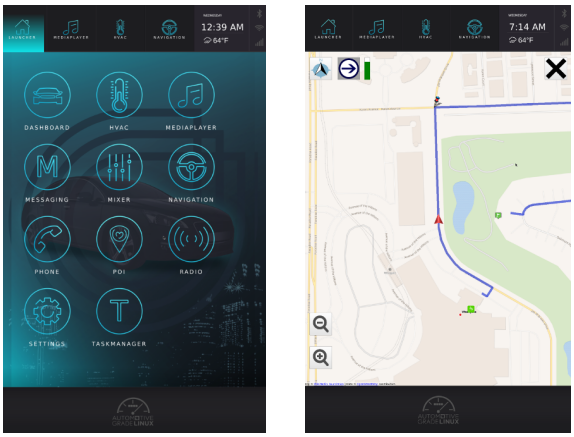
RQ1. Are the other IVI systems vulnerability-free?
RQ2. How many kinds of data should we protect against attacks?
RQ3. Is there any chance of privacy/PII leakage?
RQ4. Except for a car crash, how could adversaries harass drivers via a vulnerable IVI system?
RQ5. How is an IVI system designed to be resilient when they receive malformed requests?

There might be some plausible answers. However, we can get concrete answers only if we can assess a vulnerable IVI system in great detail. We believe there is a possibility that many vulnerable IVI systems could suffer from a zero-day exploit. Consequently, security considerations for minimizing the impact are necessary. To this end, researchers and car makers should understand the impact and implications of IVI system exploitation. Our goal is to discuss the impact and implications of IVI hacking in detail using an IVI system testbed and exploitations of vulnerabilities. For the rest of the paper, we answer the five questions mentioned above.

This paper aims to report our experience on our AGL-based experimental testbed. Thus we do not distill a general framework to assess many types of IVI systems or detect intrusions. We believe such a general framework is effective only when academia and industries cooperate to standardize an E/E architecture. Also, we do not consider the other IVI OSs to avoid broad but shallow outcomes.

3 IVI SYSTEM HACKING COMPETITION

Motivated by the previous works, we survey the impact and implications of IVI system hacking by conducting the IVI system hacking competition. In this section, we describe how we prepare the cybersecurity competition, named Cyber Security Challenge 2021 (CSC2021). The competition schedule is shown in Table 2. In step 1, we prepare the experiment testbed (§3.1, §3.2). In step 2, we announce the call for challenge along with the attack vector and evaluation metric of submission (§3.3). Each team has about one month for the qualification round (steps 2–3) and the final round (steps 4–5), respectively. For each round, they perform the penetration analysis and prepare their submissions. All teams share their findings on the day of the on-site competition (step 6).



(a) Homescreen App (b) Navigation App

Figure 2: Screenshots captured on AGL (v11.0.0 Kooky Koi)

3.1 Experiment Testbed

To conduct our study, we need an experimental testbed that realistically reflects a set of an IVI system and E/E architecture of a vehicle. It should be noted that we do not consider using a real car because some participants may not be able to obtain the exact car, which again places a research hurdle against participants. We prepared the testbed, a combination of an affordable embedded device and public infotainment OS for reproduction and sharing experiences. Among three popular infotainment OS (QNX, AAOS, and AGL), we carefully chose AGL for our testbed because of the following reasons: (1) A vanilla AGL release provides almost services that people expect in a real car, whereas the rest come with no/limited Apps, (2) AGL has never been considered in the literature, (3) The Linux Foundation provides pre-compiled binaries and source codes designed for Raspberry Pi 4 so that participants and we can easily compose the experimental testbed.

We collected source codes of the AGL release version 11.0.0 (Kooky Koi). We chose the version because it was the latest version when initiating the study. We compiled an AGL booting image by following the official AGL build instruction. We could obtain a fully functional IVI system testbed on our Raspberry Pi 4 and touch screen monitor. For the rest of our study, we leveraged the testbed setup to share the same experimental environment with the participants.

3.2 Automotive Grade Linux

3.2.1 Overview. AGL is an open-source infotainment OS maintained by a consortium of the Linux Foundation, car makers, and IT/hardware companies. AGL is a middleware providing infotainment App framework, in-vehicle bus manager, telematics support, and so on. In Figure 2, we present two screenshots of the Homescreen App and the Navigation App. The Homescreen App shows all the other Apps implemented that come with our AGL booting

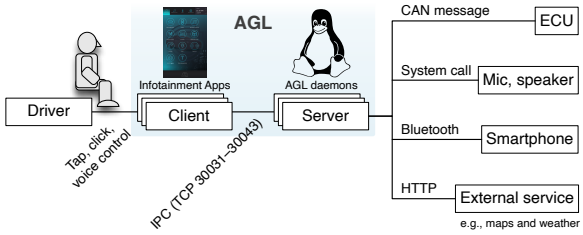


Figure 3: Communication diagram of AGL

Table 3: IVI services listening on TCP ports.

TCP port	Process name
30031	afbd-dashboard
30032	afbd-homescreen
30033	afbd-hvac
30034	afbd-launcher
30035	afbd-mediaplayer
30036	afbd-messaging
30037	afbd-mixer
30038	afbd-navigation
30039	afbd-phone
30041	afbd-radio
30042	afbd-settings
30043	afbd-taskmanager

image—for car management (dashboard and HVAC), convenience (navigation, media player, message, phone, POI, and radio), and infotainment service management (mixer, settings, and task manager). The Navigation App is presenting a map and a route to a certain destination.

We notice that an AGL release can be adopted as a complete IVI OS in a real car without further implementations. For instance, the Dashboard app periodically sends On-Board Diagnostics (OBD)-II queries to a vehicle via a CAN bus (or a virtual CAN bus if no physical CAN bus is available) to acquire the current velocity and engine speed from ECUs. The Navigation App is integrated with a Linux service *gpsd* to obtain the current geolocation of the vehicle; it also utilizes OpenStreetMap service [35] to provide maps and to route to a destination. The AGL release supports various Bluetooth profiles for hands-free calls, messaging, and music streaming to communicate with a Bluetooth-paired smartphone.

3.2.2 Security Risks. Figure 3 depicts the communication diagram of AGL that we identified. We classify components into four groups that consist of a driver, IVI Apps, background IVI services, and external components. We figure out TCP-based inter-process communications (IPCs) between IVI Apps and IVI services. The description of the IVI service and IVI App is as follows:

IVI service (server). a background Linux daemon listening on a TCP port. We thus consider an IVI service a server. An IVI service processes a driver's requests that come through a client. If necessary, it also cooperates with other components such as ECUs, a mic, a speaker, a smartphone, and external services. An IVI service then returns responses to an IVI App. We list the IVI services and corresponding TCP ports in Table 3.

IVI App (client). a foreground Linux process rendering a GUI (e.g., Figure 2) that allows a driver to use the IVI system through the touch screen monitor. We consider an IVI App a client because it always connects to one of the IVI services via a TCP/IP session to delegate a driver's request.

In the middle of preparation for the cybersecurity competition, we examined our AGL-based testbed in a heuristic manner. As a result, we found two security risks on our AGL-based IVI system testbed. Based on the security risks, we identified three vulnerabilities, which we discuss later in §4.1.1–§4.1.3.

Our first finding is on the IPC between the IVI Apps–IVI services. The servers listen on TCP ports 30031–30043 from *any hosts* (0.0.0.0/0) instead of *localhost* (127.0.0.0/8) or *a specific subnet* (e.g., 10.0.0.0/30). It means any node can communicate with the IVI services running on our IVI system. Moreover, neither encryption nor authentication is applied in the IPC. It implies a security risk that a remote adversary could monitor, reverse-engineer, and forge communications.

The second finding is HTTP sessions between IVI services–external services. An IVI service named *afbd-homescreen* connects to OpenWeatherMap [28] to present weather information of current location to the Homescreen App (see the upper right corner of Figure 2(a)). Also, the Navigation App connects to OpenStreepMap [35] to obtain map tiles and a route. Obviously, HTTP sessions via the Internet are a severe security risk, which may suffer from man-in-the-middle (MITM) attacks or data leakage [7].

3.3 Competition Design

3.3.1 Call for Competition. We state the research question—*"Given the AGL-based experimental testbed, figure out vulnerabilities and then prove maximum impacts by implementing exploit codes"* [14]. Anyone interested in cybersecurity can apply for the IVI hacking competition on a team basis. The competition consists of the qualification and final rounds. Each team needs to submit a vulnerability analysis report, a video demonstrating an exploitation procedure, and PoC exploit codes for each round. Note that we have not confirmed whether there are security risks or vulnerabilities in the statement.

We evaluate each submission based on the evaluation metric, the attack score. To quantify and compare the attack score of each team, all teams needed to comply with the following restriction—*"A team can submit up to five submissions. A submission must leverage the atomic vulnerability of a specific component (e.g., process, service). In other words, vulnerability chaining is not allowed, so the maximum severity derived from a single vulnerability can be considered."* We advance some teams for the final round, who prove high-impact hacking scenarios. Advanced teams have a chance to polish their exploits until the final round to achieve more severe consequences. It was also announced that the Republic of Korea government (MSIT) will fund a competition winner to extend their work onto real cars.

3.3.2 Attack Surface. Figure 4 shows the attack surface and adversary considered during the study. Each team is required to be the adversary who *remotely* attacks the IVI system via the designated attack surface. Based on the security risks discussed in §3.2.2, we mention the hint to participants—*"An adversary might inject an arbitrary command to the vulnerable IVI system via the attack surface. For example, the attacker might adjust the temperature of HVAC by sending a payload."* Even though the AGL release supports USB peripherals including an NFC reader, multimedia storage, and GPS signal receiver, we exclude the USB from the attack surface to focus on scalable remote attacks against connected vehicles.

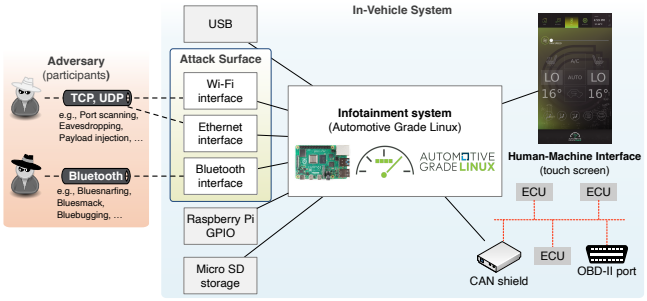


Figure 4: The attack surface and adversary with our experimental testbed.

3.3.3 Attack Score. To measure each participant submission, we use the "attack score" based on the MITRE Common Vulnerability Scoring System (CVSS) [8]. CVSS is a vulnerability metric used to rate the severity and risk of computer system security. CVSS consists of Base, Temporal, and Environmental groups. In this study, we adapt the Base metric group because it covers all environments and does not change over time. The Base metric group represents vulnerability's inherent characteristics in 0–10. Our strategy to measure the attack score is as follows:

Exploitability. We follow the original definitions of the Exploitability metrics: the attack vector, attack complexity, privileges required, and user interaction.

Scope. The CVSS specification document [8] defines that *"the Scope metric captures whether a vulnerability in one vulnerable component impacts resources in components beyond its security scope."* We customize the Scope metric to measure whether a CAN bus (i.e., a vehicle) is breached.

Impact. The Impact metrics are originally designed to capture potential effects of a successfully exploited vulnerability. However, we measure only the effects *confirmed* by us. The Impact metrics measure confidentiality, integrity, and availability breaches.

We list the scoring guideline in Table 4. It should be noted that different attack scores can be assigned for two submissions considering the same vulnerability in a specific component when the submissions achieve different outcomes.

4 EVALUATION

The IVI system hacking competition brought 26 teams of 102 researchers from universities, small/mid-size enterprises, and cybersecurity research institutes in the qualification round. Among them, we advanced seven teams (Teams A–G) to the final round who submitted high-impact and verifiable submissions for the qualification round. The seven teams had a chance to polish their submissions. The 35 submissions have been submitted for the final round, but we could verify 33 of them. On the day of the on-site competition, the participants (see Figure 5 and [13]) shared their findings in a conference room. For the rest of this section, we review the polished submissions listed in Table 5.

4.1 Vulnerability

The participants successfully identified and exploited four vulnerabilities even though we did not disclose our findings in §3.2.2 and

Table 4: Guideline to measure the attack score of submission (PoC exploit code). The AV, AC, PR and UI consider a vulnerability whereas the rest considers verified effects.

CVSS metric	Severity
Attack vector (AV)	• *N: Network.* The vulnerability is accessible through one or more hops. • *A: Adjacent network.* The vulnerability is accessible via a physical medium (e.g., Bluetooth and Wi-Fi) or an adjacent logical network (e.g. local area network). • *L: Local / P: Physical not available with respect to the designated attack vector.*
Attack complexity (AC)	• *L: Low.* The vulnerability is always exploitable regardless of any conditions. • *H: High.* Successful exploitation of the vulnerability depends on some conditions an adversary cannot handle.
Privileges required (PR)	• *N: None.* The vulnerability is successfully exploitable even without a privilege. • *L: Low.* One or more privileges need to be prepared before the exploitation of the vulnerability. • *H: High. not available because there is no administrative IVI feature in the AGL-based IVI system testbed.*
User interaction (UI)	• *N: None.* The vulnerability is exploitable without interaction from a user. • *R: Required.* Some actions from a user are required in order for the successful exploitation of the vulnerability.
Scope (S)	• *U: Unchanged.* The submission only affects the AGL-based IVI system testbed. • *C: Changed.* The submission affects the vehicle via the target IVI system. In particular, one or more CAN messages are sent.
Confidentiality (C)	• *N: None.* The submission does not breach confidentiality at all. • *L: Low.* The submission reads trivial information such as the current audio volume and name of the foreground IVI App. • *H: High.* The submission retrieves sensitive information such as a password and geolocation.
Integrity (I)	• *N: None.* The submission does not breach integrity at all. • *L: Low.* The submission makes the target execute an arbitrary command (e.g., play music) or manipulate information (e.g., modify the playlist). • *H: High.* The submission affects the vehicle through the target IVI system.
Availability (A)	• *N: None.* The submission does not breach availability at all. • *L: Low.* The submission temporarily interrupts the target while the submission is being executed. • *H: High.* The submission permanently disables the target even after the complete execution of the submission. Further actions (e.g., reboot) by a driver are required to get back to an attack-free state.

Figure 5: Top seven teams who advanced to the final stage of the CSC2021 competition.

no one had experienced AGL before the competition. Teams A–E figured out one vulnerability, whereas Teams F and G identified three and two vulnerabilities, respectively. One of the common root causes of the vulnerabilities is the lack of security design. We categorize them as four distinct vulnerabilities as follows.

4.1.1 Command Injection. The vulnerability exists within IVI services (servers) due to no authentication of clients. The clients must be legitimate IVI Apps running on the localhost or another in-vehicle component. However, the IVI services listed in Table 3 never examine the validity of the clients or the requests. As a result, the IVI services may handle a well-crafted request from a knowledgeable adversary.

Table 5: The 35 final round submissions from the Teams A–G, evaluation results and attack scores. CI: command injection, RE: resource exhaustion, IEC: insecure external communication, BB: bluebugging

ID	Target component	Vuln.	CVSS vector	Attack score
A-1	Homescreen server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:H	8.6
A-2	HVAC server	CI	AV:N/AC:L/PR:N/UI:N/S:C/C:N/I:N/A:N	0.0
A-3	Phone server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L	8.6
A-4	Messaging server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N	8.2
A-5	Settings server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:H	8.6
B-1	HVAC server	CI	AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:N	9.3
B-2	Audio mixer server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N	5.3
B-3	Messaging server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N	5.3
B-4	Phone server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N	8.2
B-5	Radio server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N	5.3
C-1	HVAC server	CI	AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L	9.9
C-2	Settings server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:H	9.4
C-3	Homescreen server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:H	9.4
C-4	Phone server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L	8.6
C-5	Messaging server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N	8.2
D-1	HVAC server	CI	–	0.0
D-2	Settings server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:H	8.6
D-3	Messaging server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N	8.2
D-4	Phone server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	7.5
D-5	Homescreen server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	7.5
E-1	HVAC server	CI	AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:H	10.0
E-2	Navigation service	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:H	9.4
E-3	Settings service	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:H	9.4
E-4	Phone service	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:H	9.4
E-5	Messaging service	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:H	9.4
F-1	Mediaplayer server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:H	8.6
F-2	Navigation App	IEC	AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:L/A:H	8.3
F-3	HVAC server	CI	AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:N	9.3
F-4	Phone server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L	8.6
F-5	Linux Bluetooth stack	BB [24]	AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:H	8.3
G-1	HVAC server	CI	AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:N	9.3
G-2	Phone server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N	8.2
G-3	Messaging server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N	8.2
G-4	Mediaplayer server	RE	AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	7.5
G-5	Settings server	CI	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L	7.3

```python
# pip install websocket-client
from websocket import create_connection
from json import dumps

conn = create_connection("ws://10.0.0.5:30032/api?token=@t")
payload = [2, "999999",
    "homescreen/showInformation", # an IVI service verb
    {"info": '<img height="215" src="https://example.com/banner.png"></img>'}] # an argument for the verb
conn.send(dumps(payload))
conn.recv()  # if an adversary is expecting some response from the target IVI service.
conn.close()
```

Listing 1: PoC exploit code of the command injection vulnerability.

Almost all submissions (32 of 35) tried to exploit the vulnerability because it allows an adversary to trigger any infotainment features implemented on AGL. Moreover, two security weaknesses make it feasible to exploit the vulnerability. First, the IVI services listen to any hosts, as mentioned in §3.2.2. Second, IPC communications between IVI Apps–IVI services are unencrypted, which helps the adversary reproduce an arbitrary payload. The participants identified that (1) each IVI service is an HTTP server expecting WebSocket [20] connections, and (2) the WebSocket payload consists of an unencrypted JSON [4] object. A JSON object consists

```python
import requests, time

# Create the 4 GB of dummy payload.
payload = bytes('A') * ((1024 ** 3) * 4)
while True:
    try:
        requests.post(url='http://10.0.0.5:30034/api/bluetooth-map/list_messages', headers={'Content-Type': 'application/json'}, data=payload)
    except requests.exceptions.ConnectionError:
        # Connection closed unexpectedly by an IVI server
        time.sleep(5) # Wait for 5 seconds.
    except ConnectionRefusedError:
        # TCP port closed (the IVI server terminated)
        break
```

Listing 2: PoC exploit code of the resource exhaustion vulnerability

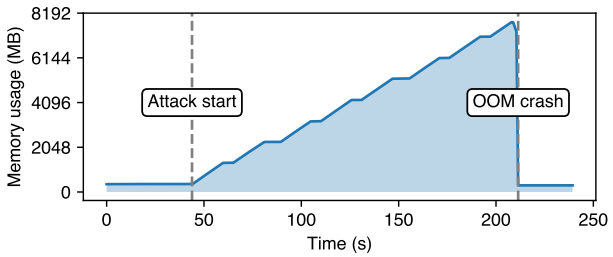


Figure 6: Memory usage benchmark. Our Raspberry Pi 4 was equipped with 8GB of RAM. An HTTP web server (IVI service) crashed when no free memory space had left.

of two sequential numbers, a verb, and an argument for the corresponding verb. For example, when a legitimate user manipulates the HVAC App to set 30 °C for the left seat temperature, the HVAC App transmits the payload *[2,70,"hvac/set,"{"LeftTemperature":30}]* to the IVI service afbd-hvac (port 30033). The service then answers the HVAC App whether the request succeeded. In a similar way, the Navigation app sends the payload *[2,70,"navigation/-subscribe,"{"value":"position"}]* to the afbd-navigation service (port 30038) to subscribe events that current vehicle location has changed.

An IVI server responds to forged requests generated by an adversary (i.e., the participants). Listing 1 is the PoC exploit code leveraging the vulnerability. It requires the IVI service afbd-homescreen to pop up the HTTP document (the *<img>* tag). The execution result is depicted in Figure 7(c). Table 6 lists the entire verbs, including `showInformation`, that can be triggered by the adversary.

4.1.2 Resource Exhaustion. The vulnerability exists within IVI services due to improper input validation and improper resource management. The two security weaknesses cause an IVI service to receive a malformed payload and cache it to the heap memory. The IVI service never frees the heap memory if a connection with an IVI App (or a remote adversary in the middle of an attack) is closed unexpectedly. An adversary can exploit the vulnerability to crash an IVI service due to the out-of-memory error.

Team G was the only team who identified the vulnerability. We provide Listing 2, a revision of submission G-4. The code is designed to transmit a huge payload to an IVI service afbd-launcher (port 30034). Figure 6 shows the memory usage benchmark of our experimental testbed while the PoC code is being executed. Whenever

Table 6: List of verbs implemented in the IVI services that can be triggered by remote adversaries.

Service category	TCP port and application	Verb
audiomixer	[30030] afbd-homescreen [30035] afbd-mixer	list_controls, volume, mute, subscribe, unsubscribe
Bluetooth-Manager	[30030] afbd-homescreen [30040] afbd-settings	subscribe, unsubscribe, managed_objects, adapter_state, default_adapter, connect, disconnect, pair, cancel_pairing, confirm_pairing, remove_device, avrcp_controls, set_pincode
bluetooth-map	[30034] afbd-messaging	compose, message, list_messages, subscribe, unsubscribe
bluetooth-pbap	[30037] afbd-phone	contacts, import, entry, history, search, status, subscribe, unsubscribe
HVAC	[30031] afbd-hvac	get_temp_left_zone, get_temp_right_zone, get_fanspeed, get, set, temp_left_zone_led, temp_right_zone_led
homescreen	[30030] afbd-homescreen [30032] afbd-launcher [30038] afbd-poi	ping, tap_shortcut, showWindow, hideWindow, replyShowWindow, on_screen_message, on_screen_reply, subscribe, unsubscribe, showNotification, showInformation, getRunnables
mediaplayer	[30033] afbd-mediaplayer	playlist, controls, subscribe, unsubscribe
navigation	[30036] afbd-navigation [30038] afbd-poi	subscribe, unsubscribe, broadcast_status, broadcast_position, broadcast_waypoints
network-manager	[30030] afbd-homescreen [30040] afbd-settings	subscribe, unsubscribe, state, offline, technologies, services, enable_technology, disable_technology, scan_services, move_service, remove_service, connect_service, disconnect_service, get_property, set_property, agent_response
radio	[30039] afbd-radio	frequency, band, rds, quality, alternative_frequency, band_supprorted, frequency_range, frequency_step, start, stop, scan_start, scan_stop, stereo_mode, subscribe, unsubscribe
signal-composer	[30029] afbd-dashboard	subscribe, unsubscribe, addObjects, list, get
telephony	[30037] afbd-phone	dial, last_dial, send_tones, hangup, answer, get_battery_level, get_network_registration, subscribe, unsubscribe
weather	[30030] afbd-homescreen	current_weather, api_key, subscribe, unsubscribe

the inbound payload size exceeds ≈1 GB, the IVI service closes a connection unexpectedly. Note that the memory is never freed, even when the adversary takes a five-second break. Eventually, the IVI service crashes when no free memory space has left.

4.1.3 Insecure External Communication. The vulnerability exists within the Navigation App and the IVI service afbd-homescreen due to insecure communication channel. Two vulnerable components establish HTTP sessions with OpenStreetMap and OpenWeatherMap, respectively. Navigation App tries to obtain map tiles from *http://c.tile.openstreetmap.org* whenever a driver manipulates the map on GUI. Regarding afbd-homescreen, we found a URL starting with scheme "http://" in a source code (c.f., line 35 [34]).

Team F was the only team who identified the vulnerability. Submission F-2 is designed to perform an ARP spoofing attack to intercept HTTP sessions between the AGL-based experimental testbed and the external services. It then exploits the vulnerability to replace map tiles and a route to a destination. We reproduce submission F-2 and present Figure 7(d) to show the alternation of map tiles.

4.1.4 Bluetooth Pairing. The vulnerability exists within the AGL Bluetooth library named BlueZ version 5.55. It allows physically proximate adversaries to pair their devices with the AGL-based IVI system without the user's acknowledgment. The vulnerability is inherited from the Linux kernel.

Only Team F figured out the vulnerability. They leveraged a security weakness that the Settings App indefinitely enables the Bluetooth discovery mode of AGL. Submission F-4 successfully reproduced Bluebugging ([6, 24, 27, 36]). The Bluebugging attack is commonly known as the attack targeting cell phones. However, Team F empirically proved that an IVI system could also be the target of Bluebugging.

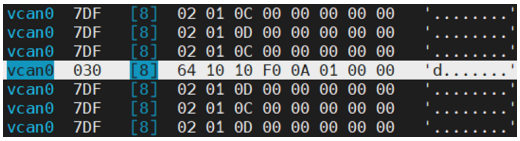
4.2 Impact

Based on the four vulnerabilities, the participants proved that not only a vulnerable IVI system but also a car, user, and user's smartphone can suffer from severe impacts. Due to the page limit, we summarize all impacts in Table 7 that we empirically confirmed during the competition. In addition, we visualize four exemplary impacts in Figure 7. We remark the impact of some submissions.

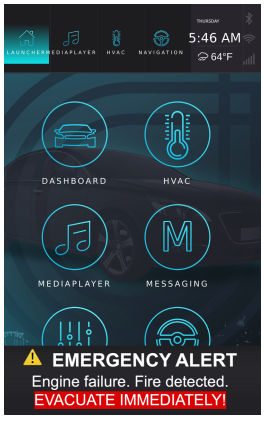

```
$ python3 hack.py
[+] Retrieve contacts
{'response': {'vcards': [{'fn': 'My Number', 'telephone': [{'CELL': '+82200000000'}]}, {'fn':
'John', 'telephone': [{'WORK': '1111111111'}, {'CELL': '010-1111-1111'}, {'UNKNOWN':
'+82111111111'}]}, {'fn': 'Jane', 'telephone': [{'WORK': '2222222222'}, {'CELL':
'010-2222-2222'}, {'UNKNOWN': '+82222222222'}]}, {'fn': 'Janet', 'telephone': [{'WORK':
'3333333333'}, {'CELL': '010-3333-3333'}, {'UNKNOWN': '+82333333333'}]}, {'fn': 'Hugo',
'telephone': [{'WORK': '4444444444'}, {'CELL': '010-4444-4444'}, {'UNKNOWN':
'+82444444444'}]}, {'fn': 'Office', 'telephone': [{'WORK': '5555555555'}, {'CELL':
'010-5555-5555'}, {'UNKNOWN': '+82555555555'}]}]}, 'jtype': 'afb-reply', 'request':
{'status': 'success', 'info': 'contacts'}}

[*] Retrieve call history
{'response': {'vcards': [{'fn': 'John', 'type': 'RECEIVED', 'timestamp': '20220419T174631',
'telephone': '+82111111111'}, {'fn': 'John', 'type': 'MISSED', 'timestamp':
'20220419T174528', 'telephone': '+82111111111'}, {'fn': 'John', 'type': 'MISSED',
'timestamp': '20220419T174447', 'telephone': '+82111111111'}, {'fn': 'Jane', 'type':
'DIALED', 'timestamp': '20220419T173512', 'telephone': '+82222222222'}, {'fn': 'Jane',
'type': 'DIALED', 'timestamp': '20220419T173508', 'telephone': '+82222222222'}]}, 'jtype':
'afb-reply', 'request': {'status': 'success', 'info': 'call history'}}
```

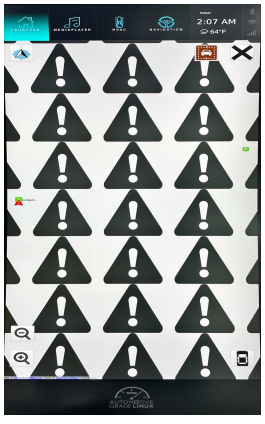

(a) Responses from an IVI service afbd-phone. The adversary is leaking call logs from a driver's smartphone.



(b) CAN message dump. The attacker is transmitting the CAN message with Arbitration ID 030h to ECUs.



(c) Compromised Homescreen App showing the fake alert.



(d) Compromised Navigation App rendering an invalid map.

Figure 7: Four exemplary impacts of IVI system hacking

Table 7: Various impacts of IVI system hacking identified by the participants and the authors

Target component	Breach of confidentiality	Breach of integrity	Breach of availability
HVAC server	• Get current HVAC settings (e.g., Left temperature)	• Manipulate HVAC settings • Send a CAN message to the CAN bus	• Override the driver's setup by brute-force requests • DoS in the in-vehicle CAN bus
Phone server	• Retrieve phonebook entries • Retrieve a list of recent calls • Receive an incoming call event	• Make a phone call to anyone • Answer the phone call • Hang up the phone call	• Repeat making a call and hanging-up the call by brute-force requests • DoS of Bluetooth Phone Book Access Profile (PBAP) service by using a huge payload
Messaging server	• Retrieve an incoming message list from a paired smartphone	• Send an SMS to anyone with an arbitrary content	• DoS of Bluetooth Message Access Profile (MAP) service by sending a large payload
Settings server	• Get a list of known Bluetooth devices and an list of Wi-Fi APs • Get Bluetooth PIN codes and Wi-Fi passwords • Retrieve nearby Bluetooth devices and Wi-Fi APs	• Make a new Bluetooth pair and a Wi-Fi connection • Remove Bluetooth-paired devices and known Wi-Fi APs • Turn off Wi-Fi, Bluetooth, and Ethernet interface	• Repeat turning off Wi-Fi and Bluetooth interface • Disable Ethernet interface permanently using a hidden API
Homescreen server	• *All impacts as in Settings server* • Get a current geolocation	• *All impacts as in Settings server* • Foreground another infotainment app • Pop-up a notification (An HTML document that may contain text, image, etc.)	• *All impacts as in Settings server*
Mediaplayer server	• Retrieve a playlist	• Manipulate a playlist	• DoS of media player due to incorrect metadata (e.g., invalid MP3 file path)
Audio mixer server	—	• Manipulate an IVI audio volume	—
Radio server	—	• Tune FM radio frequency	—
Navigation server	• Retrieve a current geolocation	• Set a destination • Start/stop navigation	• Repeat starting and stopping direction guide by brute-force requests
Navigation App	• Retrieve a current geolocation and a destination • Retrieve coordinates of areas of interest	• Provide fraudulent maps • Provide a fraudulent route to the destination	• Hide maps, Provide a route not directed to the destination
Linux Bluetooth stack	• Download an arbitrary file from the AGL filesystem • Eavesdropping passengers' voice using a mic	• Make a new Bluetooth pair between the attacker–the target IVI without user interaction • Upload a file to an arbitrary path in IVI filesystem	• DoS of the voice call feature in Phone App

4.2.1 MITM attack: Submission F-2. The submission obtained the vehicle's current geolocation (latitude and longitude), geolocation where a user tried to explore on the App, and geolocation of the destination. It is apparent sensitive data leakage. The submission altered the map and route. The route eventually guided to a destination that a user did not request. The cached map data was never refreshed until the Navigation App was terminated. Note that a user cannot re-execute the App except by rebooting the AGL-based IVI system, but the IVI system does not provide a reboot option on GUI. In such a case, a user must pull over the car and stop the engine once.

4.2.2 Bluebug: Submission F-5. The submission was designed for an adversary's PC to mimic a smartphone. The submission performed the Bluebug attack to trigger the following Bluetooth profiles: OBEX file transfer, OBEX phonebook access server, OBEX object push, and hands-free gateway. We found three kinds of impacts: (1) The submission eavesdropped on the in-vehicle sound by establishing a fake hands-free call. To verify this impact, we connected a USB microphone to our Raspberry Pi 4 and said something. (2) Using the OBEX file transfer AT command, the submission downloaded an arbitrary file from the filesystem. Also, it uploaded a file to the filesystem. It is possible because Bluetoothd, a Linux daemon, has access to the filesystem even though no file-sharing infotainment feature is implemented. (3) The submission caused the Phone App to ring permanently by transmitting one incoming call AT command. The Phone App could not work with another smartphone during it was ringing.

4.2.3 DoS attack: Submission G-4. The submission crashed the IVI service afbd-multimedia (port 30035). Once the target service crashed, it never rebounded again until reboot. It should be noted that the IVI service is not the Multimedia App. Even though the Multimedia App presented a GUI to a user, the user could not play the music because the IVI service did not respond.

4.2.4 Command injection attacks against HVAC service. Based on the experimental result, the HVAC service is the only component that allows lateral movement to the vehicle through CAN message injection. In particular, the HVAC service is designed to transmit a CAN message with Arbitration ID 030h so that an ECU physically works (see Figure 7(b)). Thus each team expected the submission could earn a high attack score. All teams leveraged the command injection vulnerability to compromise the IVI service afbd-hvac (submissions A-2, B-1, C-1, D-1, E-1, F-3, and G-1).

Team E expanded their submission to cause a denial of service (DoS) of a CAN bus. Submission E-1 is designed to transmit repetitive requests to the IVI service as quickly as possible. In this situation, due to the CAN principle, a CAN bus can only handle CAN messages with Arbitration ID (AID) 000h–02Fh. Consequently, ECUs utilizing CAN messages with AID 031h–7FFh will be out of order.

4.2.5 Command injection attacks against Phone and Messaging services. Some attacks proved another lateral movement to a driver's smartphone through the Phone and Messaging App. To verify impacts, we paired Android and iOS smartphones. At the paring time, the smartphones required permissions of sharing phonebook and messaging. Once the permissions had granted, adversaries could breach confidentiality, integrity, and availability as follows:
Confidentiality. Retrieve phonebook entries and call history stored in the smartphone (smartphone → IVI system → adversary). Retrieve text messages stored in the IVI system (IVI system → adversary). Listen to events of inbound phone calls and text messages.
Integrity. Make/answer/hang up a phone call. Send a text message.
Availability. Disable a phone call (repetitive hang-up). Notification bomb of Messaging App (an adversary makes a user's smartphone transmit hundreds of text messages to itself.)

While we verify the submission, our smartphones did not warn any abnormal situations. Except for notification sounds of inbound calls and text messages, the HMI did not present any indication. Thus a user cannot take a hint that his/her smartphone is compromised due to the vulnerable IVI system.

4.2.6 The other command injection attacks. The rest submissions leveraged the command injection vulnerability. In summary, an

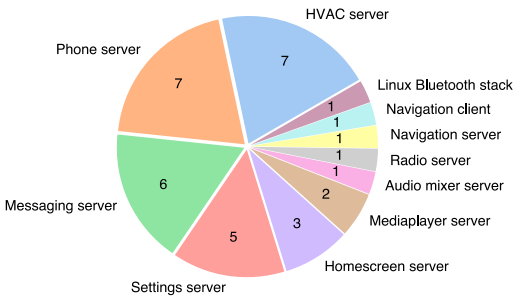


Figure 8: Number of times each target was considered by the submissions.

adversary remotely controlled entire IVI features implemented in IVI services. The confidentiality impact includes PII leakage—e.g., a playlist, point-of-interest, MAC address of the IVI system, name of currently paired smartphone—, IVI configuration leakage—e.g., a cabin temperature, audio volume, list of known/nearby Wi-Fi access points or Bluetooth devices—, and credentials—Wi-Fi passwords and Bluetooth PIN codes. In many submissions, the participants tried to breach availability using brute-force requests. For example, submission F-1 made the mediaplayer service keep playing music, although a driver tried to pause it. Submission B-2 maximized the audio volume constantly.

We report two interesting findings. Our first finding is the coarse-grained permission for the IVI service afbd-homescreen. As shown in Figure 2(a), the Homescreen App is designed to spawn another App and present information at the upper right corner. The afbd-homescreen should have only necessary verbs. However, afbd-homescreen included all the verbs implemented in afbd-setting.

The second finding is triggering a hidden feature that a legitimate user cannot trigger. An adversary can make the afbd-setting (and afbd-homescreen based on the first finding) to disable Ethernet, Wi-Fi, and Bluetooth interfaces. The problem is that the Settings App only provides toggle buttons for Wi-Fi and Bluetooth interfaces. Thus, the adversary can disable the Ethernet interface permanently.

4.3 Target Component

There are many features implemented in an IVI system. Assuming the early stage of IVI system hacking, an adversary cannot examine all features simultaneously. Therefore, an adversary should carefully choose primary target components that seem to have the potential to enlarge the impact. Our question is, which components are attractive to attackers? We foresee the attacker's perspective with participants' submissions to answer the question. Since the competition only accepted up to five submissions with limited time (about two months), each team would have carefully chosen target components to earn high attack scores.

Figure 8 shows how often the participants considered each component. HVAC, Phone, and Messaging services that allow lateral movement to ECUs and the driver's smartphone are often considered. Next, five teams considered the setting service. The five teams expected high attack scores with the setting service because it provides various features compared to the rest. Considering a human factor, we believe submissions targeting audio-related services (mediaplayer, audio mixer, radio) can annoy a driver by playing/stopping music, making a loud noise, and so on. However, only a few teams have considered these services.

Team F is the only team who utilized three vulnerabilities, including the MITM attack on the Navigation App and Bluebugging against the Linux Bluetooth stack. Therefore, all impacts on the Navigation App and Bluetooth stack were found by Team F. The other teams were unaware of such attacks until the day of the on-site competition.

5 IMPLICATIONS OF IVI SYSTEM HACKING

This section discusses real-world hacking scenarios and consequences that can happen through a vulnerable IVI system. We consider four implications by considering confirmed impacts.

5.1 Car Accident

The experimental result confirms that an attacker can exploit a vulnerable IVI system to cause car accidents in many ways intentionally. All teams attempted to exploit the HVAC service to breach integrity and availability by a DoS attack on the virtual CAN bus. When they flood CAN messages with the arbitration ID of 0x030, the other legitimate CAN messages with arbitration IDs 0x031–0x7ff cannot be transmitted due to the arbitration mechanism of the CAN principle [30].

An attacker may try to deceive a driver rather than a car. The following scenarios can be composed with the combination of participants' submissions: (1) An attack can provide a fake map and route to guide a driver off the road. (2) As demonstrated in Figure 7(c), an attacker could pop up a well-formatted HTML5 document as a notification, causing a driver to make dangerous decisions on a highway. (3) A driver could suffer a heart attack when an attacker maximizes the audio volume and plays a white noise of FM radio.

5.2 Privacy and PII Leakage

It should be noted that securing an IVI system is not just about securing a cyber-physical system (i.e., a vehicle). We confirmed the privacy leakage from the driver's smartphone via a vulnerable IVI system. Even though Android OS and Apple iOS are armed with state-of-the-art security technologies to protect users' privacy, an attacker can obtain phonebook entries, recent call logs, and text messages from Bluetooth-paired smartphones. Moreover, the attacker can make a phone call or send a message to any recipient. A driver should grant permission to share such information on their smartphone to conduct the attack. However, we believe many drivers will grant permission for their convenience since the literature has not yet mentioned the risks of such privacy leakage.

An IVI system also creates and handles PII, such as current geolocation, frequently visited areas, Bluetooth/Wi-Fi MAC addresses, media player metadata, and so forth. Consequently, an attacker can retrieve such data from a vulnerable IVI system to spy on a driver.

Modern vehicles use a cabin microphone to support hands-free calls and voice assistants. The problem is that an adversary could utilize the microphone to eavesdrop on the cabin voice. Specifically, our study figured out the two leakage paths as follows:

Table 8: Three new vulnerabilities identified during the competition. The checkmark represents the affected context.

	Vulnerability	Confidentiality	Integrity	Availability	Privacy
§4.1.1	CVE-2022-24595	✔	✔	✔	✔
§4.1.2	CVE-2022-24596	–	–	✔	–
§4.1.3	CVE-2022-24597	✔	✔	✔	✔

IVI system → Driver's smartphone → Attacker's phone. The driver's smartphone should be paired with the IVI system before an adversary conducts the attack. The attacker requests the IVI system to dial the attacker's phone. Consequently, the driver's smartphone dials the attacker's phone on behalf of the IVI system. The attacker can prevent the driver's awareness by muting the speaker of the IVI system. Eventually, the attacker can listen to the cabin sound by answering the phone call.

IVI system → Attacker's PC. An attacker should be located near the target IVI system to conduct the attack to maintain a Bluetooth connection. First, the attacker establishes a new Bluetooth pair with the IVI system by exploiting a Linux Bluetooth vulnerability. At the moment, the attacker's PC is considered a phone. Then, using Bluetooth hands-free gateway profile, the attacker establishes a pseudo-phone call. The IVI system should start transmitting the cabin voice to the attacker's PC.

5.3 Financial Damage

Nowadays, cellular text messages are used for communication between people and authentication, such as two-factor authentication and mobile banking. Thus, an attacker could earn financial profits by hacking a vulnerable IVI system. For example, the attacker withdraws money from the driver's banking account by sending/receiving SMS messages. On the other hand, the attack can transmit a bulk of spam messages via the IVI system.

When a driver is under an advanced persistent threat, a prolonged and targeted cyberattack, the attacker could steal text messages with confirmation codes of some other critical systems.

6 DISCUSSION

6.1 Lessons Learned

During the cybersecurity competition, we composed an experiment testbed, assessed the security risks, analyzed the vulnerability, and reviewed the impact and implications. We found three new vulnerabilities applicable on the AGL release version 11.0.0 (Kooky Koi)–12.1.6 (Lucky Lamprey). We reported them into the CVE database as shown in Table 8. In addition, one inherited vulnerability was figured out. The experimental result confirmed that Bluggbugging can also be used to attack IVI systems as well as cellphones.

By gathering 102 researchers, we successfully discussed severe impacts on nine IVI services, one IVI App, and one Linux daemon. We also discussed potential hacking scenarios that have never been considered in the literature. The participants had a chance to consider what kinds of data are handled in an IVI system, what is the attack surface, and how a vulnerable IVI system can suffer from an attack. On the day of the on-site competition, each team shared what their strategy was to exploit the given IVI testbed. Team E, the winner of our cybersecurity competition, is currently pursuing security issues on IVI systems of real cars with the winner grant funded by the Republic of Korea government.

We sincerely argue the following mitigations to remedy the impacts of further attacks against various IVI systems in the wild.

6.1.1 Fine-grained access control. It should be noted that an infotainment OS should provide fine-grained access control. Furthermore, each IVI service should have just enough permissions. During the study, we confirmed that the coarse-grained permissions maximize the impact when a vulnerability is exploited. For example, the homescreen service allows an adversary to steal Wi-Fi passwords (see Table 7) even though Homescreen App does not take such information from a user. Readers may refer to Table 6 to see the course-grained permissions of the afbd-homescreen.

Even though there is no file-sharing feature implemented on IVI Apps, the Linux Bluetooth daemons *bluetoothd* and *bluez* have access to the filesystem, which allows an attacker to upload/download an arbitrary file. Proper access controls will remedy such impacts.

6.1.2 Providing testbed for penetration test. To comply with the recent cybersecurity regulation on vehicles [2], car makers can take advantage by providing testbeds that reflect their vehicles' IVI system and E/E architecture. They could leverage multiple points of view of cybersecurity researchers. We believe providing a testbed resolves glitches and vulnerabilities significantly before an IVI system is released on the market.

Including our study, all studies have been conducted in a heuristic manner due to the limited access to many IVI systems. Therefore, public testbeds will also be beneficial in academia to formalize general frameworks.

6.1.3 Firewall. A firewall is considered one of the essential security countermeasures. Unfortunately, that is not applied in the vehicle cybersecurity area. The studies [5, 9, 23, 32] and our study confirmed that TCP or UDP ports are exposed to external hosts. It is obvious that a little effort in setting up the firewall on IVI systems will prevent many attacks.

6.1.4 Privacy indicator. Recent OSs such as Windows, macOS, Android, and iOS have a privacy indicator showing if a camera or microphone is active. However, AGL does not have such a privacy indicator. So a user never had a chance to notice an anomaly, especially when an in-vehicle microphone is active or an IVI system is communicating with a smartphone. We believe the privacy indicator on HMI will mitigate long-term exposure to attacks even after an IVI system has been compromised.

6.2 Limitation

The limitation of our study is that we only assessed single infotainment OS. Also, we did not utilize a real car. Since vehicles have their configuration and hardware, our findings might not seamlessly be applied to real-world vehicles. Our study did not propose a specific framework or methodology. Further efforts are still required to expand assessments on other vehicles.

One limitation regarding the cybersecurity competition is that the Scope metric in Table 4 only considers a CAN bus. That is because we did not presume that an IVI system hacking could affect a driver's smartphone when we prepared for the event. If we had

kept that in mind, we would have designed the Scope metric to consider all kinds of lateral movements.

7 CONCLUSION

This work represents the first step towards the massive discussion about how and why IVI system security is essential. We have conducted an empirical study to present how a vehicle and driver could suffer from attacks owing to a vulnerable IVI system. We have conceived and designed a cybersecurity competition with the AGL-based IVI testbed. The 102 researchers and we have gathered together and discussed 35 submissions. Among them, 33 submissions have successfully verified on the testbed. The evaluation of the submissions demonstrates the severe impacts that could result in various consequences. The consequences include not only car accidents but also deceiving a driver, privacy/PII leakage, and financial damage. Moreover, we have found lateral movement scenarios toward ECUs (via CAN bus) and driver's smartphones (via Bluetooth). We have unveiled three new vulnerabilities and one inherited vulnerability. Without our findings, they might be implemented with commercial vehicles without awareness. Based on the experimental result, we have suggested three mitigation strategies to enhance the security of IVI systems. We hope this study motivates many other researchers and car makers to start a discussion about general frameworks to assess IVI systems and detect intrusions. As part of our future work, we plan to extend our research to assess AAOS, QNX, and AGL implemented in real cars so that we can systematize assets, attack vectors, and assessment strategies.

ACKNOWLEDGEMENT

The authors sincerely thank Misim Jung from Culture Makers Company and Saewoom Lee from KISA for their commitment to make the CSC2021 a success. We appreciate participation from team AutoCrypt (Team E, the winner) from AutoCrypt Company, team Yullgok605B (Team F) from Sejong Univ., and the other participants for their participation and findings. 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] 2019. Tesla car hacked at Pwn2Own contest. https://www.zdnet.com/article/tesla-car-hacked-at-pwn2own-contest/.
[2] 2021. UN Regulation No. 155 - Cyber security and cyber security management system. E/ECE/TRANS/505/Rev.3/Add.154.
[3] BlackBerry. 2022. BlackBerry QNX. https://blackberry.qnx.com.
[4] Tim Bray. 2014. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159. https://www.rfc-editor.org/info/rfc7159
[5] Zhiqiang Cai, Aohui Wang, and Wenkai Zhang. 2019. 0-days & Mitigations: Roadways to Exploit and Secure Connected BMW Cars. In *Black Hat USA*, Vol. 39. 1–37.
[6] Luca Carettoni, Claudio Merloni, and Stefano Zanero. 2007. Studying Bluetooth Malware Propagation: The BlueBag Project. *IEEE Security & Privacy* 5, 2 (2007), 17–25.
[7] Ankita R Chordiya, Subhrajit Majumder, and Ahmad Y Javaid. 2018. Man-in-the-Middle (MITM) Attack Based Hijacking of HTTP Traffic Using Open Source Tools. In *2018 IEEE International Conference on Electro/Information Technology (EIT)*. 0438–0443.
[8] Common Vulnerability Scoring System SIG. 2019. *Common Vulnerability Scoring System version 3.1: Specification Document Revision 1.* Technical Report. The Forum of Incident Response and Security Teams.
[9] Gianpiero Costantino and Ilaria Matteucci. 2019. CANDY CREAM - Hacking Infotainment Android Systems to Command Instrument Cluster via Can Data Frame. In *Proc. 2019 IEEE Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. 476–481.
[10] Scott Gayou. 2018. Jailbreaking Subaru StarLink. https://github.com/sgayou/subaru-starlink-research.
[11] Google. 2022. Android Automotive OS (AAOS). https://developers.google.com/cars/design/automotive-os.
[12] Jie Guo, Bin Song, Ying He, Fei Richard Yu, and Mehdi Sookhak. 2017. A Survey on Compressed Sensing in Vehicular Infotainment Systems. *IEEE Communications Surveys & Tutorials* 19, 4 (May 2017), 2662–2680.
[13] Institute for Information & Communications Technology Planning & Evaluation (IITP). 2021. 2021 Cyber Security Challenge: the final round (in Korean). https://youtu.be/E-ZTuWSg-JU.
[14] Institute for Information & Communications Technology Planning & Evaluation (IITP). 2021. Call for Challenge: 2021 Cyber Security Challenge (in Korean). https://youtu.be/HS2PfBpwjU4.
[15] Hyo Jin Jo and Wonsuk Choi. 2022. A Survey of Attacks on Controller Area Networks and Corresponding Countermeasures. *IEEE Transactions on Intelligent Transportation Systems* (Jul. 2022), 6123–6141.
[16] Hyo Jin Jo, Wonsuk Choi, Seoung Yeop Na, Samuel Woo, and Dong Hoon Lee. 2017. Vulnerabilities of Android OS-Based Telematics System. *Wireless Personal Communications* 92, 4 (2017), 1511–1530.
[17] S. M. Ahsan Kazmi, Tri Nguyen Dang, Ibrar Yaqoob, Anselme Ndikumana, Ejaz Ahmed, Rasheed Hussain, and Choong Seon Hong. 2019. Infotainment Enabled Smart Cars: A Joint Communication, Caching, and Computation Approach. *IEEE Transactions on Vehicular Technology* 68, 9 (Sep. 2019), 8408–8420.
[18] Siti-Farhana Lokman, Abu Talib Othman, and Muhammad-Husaini Abu-Bakar. 2019. Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review. *EURASIP Journal on Wireless Communications and Networking* 2019, 184 (Jul. 2019), 1–17.
[19] Sahar Mazloom, Mohammad Rezaeirad, Aaron Hunter, and Damon McCoy. 2016. A Security Analysis of an In-Vehicle Infotainment and App Platform. In *Proc. 10th USENIX Workshop on Offensive Technologies (WOOT 16)*.
[20] Alexey Melnikov and Ian Fette. 2011. The WebSocket Protocol. RFC 6455. https://www.rfc-editor.org/info/rfc6455
[21] Charlie Miller. 2019. Lessons learned from hacking a car. *IEEE Design & Test* 36, 6 (Dec 2019), 7–9.
[22] Charlie Miller and Chris Valasek. 2014. A survey of remote automotive attack surfaces. In *Black Hat USA*.
[23] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. In *Black Hat USA*. 1–91.
[24] Nateq Be-Nazir Ibn Minar and Mohammed Tarique. 2012. BLUETOOTH SECURITY THREATS AND SOLUTIONS : A SURVEY. *International Journal of Distributed and Parallel systems* 3, 1 (2012), 127–148.
[25] Abdul Moiz and Manar H. Alalfi. 2022. A Survey of Security Vulnerabilities in Android Automotive Apps. In *2022 IEEE/ACM 3rd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*. 17–24.
[26] Sen Nie, Ling Liu, and Yuefeng Du. 2017. Free-fall: Hacking Tesla from wireless to CAN bus. In *Black Hat USA*, Vol. 25. 1–16.
[27] National Institute of Standards and Technology. 2017. *Guide to Bluetooth Security.* Technical Report Special Publications 800-121 Revison 2. U.S. Department of Commerce, Washington, D.C.
[28] OpenWeatherMap Ltd. 2022. OpenWeatherMap. https://openweathermap.org
[29] Pranav Kumar Singh, Sunit Kumar Nandi, and Sukumar Nandi. 2019. A tutorial survey on vehicular communication state of the art, and future research directions. *Vehicular Communications* 18, 100164 (2019), 1–39.
[30] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. 2020. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications* 21, 100198 (2020), 1–13.
[31] Tencent Keen Security Lab. 2020. Experimental Security Assessment on Lexus Cars. https://keenlab.tencent.com/en/2020/03/30/Tencent-Keen-Security-Lab-Experimental-Security-Assessment-on-Lexus-Cars/.
[32] Tencent Keen Security Lab. 2021. *Mercedes-Benz MBUX Security Research Report.* Technical Report.
[33] The Linux Foundation. 2016. Automotive Grade Linux. https://www.automotivelinux.org.
[34] The Linux Foundation. 2020. Git Repository: agl-service-weather/afm-weather-binding.c. https://git.automotivelinux.org/apps/agl-service-weather/tree/binding/afm-weather-binding.c?id=3b723a2f50dc467607a6e8b6d9c41e2d18cef17e
[35] The OpenStreetMap Foundation. 2022. OpenStreetMap. https://www.openstreetmap.org
[36] Trifinite. 2004. BlueBug. https://trifinite.org/stuff/bluebug/
[37] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. 2020. A Survey of Intrusion Detection for In-Vehicle Networks. *IEEE Transactions on Intelligent Transportation Systems* 21, 3 (Mar. 2020), 919–933.
[38] Wenchao Xu, Haibo Zhou, Nan Cheng, Feng Lyu, Weisen Shi, Jiayin Chen, and Xuemin Shen. 2018. Internet of vehicles in big data era. *IEEE/CAA Journal of Automatica Sinica* 5, 1 (Jan. 2018), 19–35.