Author(s): Turtiainen, Hannu; Costin, Andrei; Khandker, Syed; Hämäläinen, Timo

Title: GDL90fuzz : Fuzzing "GDL-90 Data Interface Specification" Within Aviation Software and Avionics Devices : A Cybersecurity Pentesting Perspective

Year: 2022

Version: Published version

Please cite the original version:

Turtiainen, H., Costin, A., Khandker, S., & Hämäläinen, T. (2022). GDL90fuzz : Fuzzing "GDL-90 Data Interface Specification" Within Aviation Software and Avionics Devices : A Cybersecurity Pentesting Perspective. IEEE Access, 10, 21554-21562. https://doi.org/10.1109/ACCESS.2022.3150840

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3150840, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.DOI

GDL90fuzz: Fuzzing "GDL-90 Data Interface Specification" Within Aviation Software and Avionics Devices — A Cybersecurity Pentesting Perspective

HANNU TURTIAINEN, ANDREI COSTIN, SYED KHANDKER, AND TIMO HÄMÄLÄINEN
University of Jyväskylä
Jyväskylä, Finland

Corresponding author: Hannu Turtiainen (e-mail: hannu.ht.turtiainen@jyu.fi).

This work was supported in part by the Finnish Grid and Cloud Infrastructure (FGCI) persistent identifier urn:nbn:fi:research-infras-2016072533, in part by the decision of the research dean on research funding within the Faculty of Information Technology of the University of Jyväskylä (07.04.2021), and in part by the Finnish Cultural Foundation, grant decision no.00211119.

ABSTRACT As the core part of next-generation air transportation systems, the Automatic Dependent Surveillance-Broadcast (ADS-B) is becoming very popular. However, many (if not most) ADS-B devices and implementations support and rely on Garmin's GDL-90 protocol for data exchange and encapsulation. In this paper, we research GDL-90 protocol fuzzing options and demonstrate practical Denial-of-Service (DoS) attacks on popular Electronic Flight Bag (EFB) software operating on mobile devices. For this purpose, we specifically configured our own avionics pentesting platform. and targeted the popular Garmin's GDL-90 protocol as the industry-leading devices operate on it. We captured legitimate traffic from ADS-B avionics devices. We ran our samples through a state-of-the-art fuzzing platform (AFL), and fed the AFL's output to the EFB apps and GDL-90 decoding software via the network in the same manner as legitimate GDL-90 traffic is sent from ADS-B and other avionics devices. The result shows a worrying anc critical lack of security in many EFB applications where the security is directly related to aircraft's safety navigation. Out of 16 tested configurations, our avionics pentesting platform managed to crash or otherwise impact 9 (or 56%) of those. The observed problems manifested as crashes, hangs, and abnormal behaviours of the EFB apps and GDL-90 decoders during the fuzzing test. Attacks on core sub-system availability (such as DoS) pose high risks to safety-critical and mission-critical systems such as avionics and aerospace. Our work aims at developing and proposing a systematic pentesting methodology for such devices, protocols, and software, and discovering and reporting as early as possible such vulnerabilities.

INDEX TERMS GDL-90, ADS-B, attacks, cybersecurity, pentesting, resiliency, DoS, aviation, avionics, airtraffic.

I. INTRODUCTION

In the United States aviation, the Federal Aviation Administration (FAA) is pushing a shift from Secondary Surveillance Radar (SSR) interrogations to the more modern Automatic Dependent Surveillance-Broadcast (ADS-B) technology in air traffic control. As of January 2020, aircraft operating under the continental United States are required to use ADS-B [1]. European aviation is following suit as the gradual shift to mandatory ADS-B broadcasting has started in June of 2020 [2]. ADS-B offers many benefits over the SSR, such as enhanced situational awareness to all the aircraft and ATCs in the vicinity in a fully automatic manner, increasing system efficiency by eliminating interrogation processes, and cost-effective implementation. Moreover, FAA and its stakeholders are actively experimenting with ADS-B for commercial space transportation applications [3]. Due to its efficiency, lightweight, and cost-efficient features, it is gaining popularity among all sorts of users. Using portable ADS-B transceiver (e.g., SkyEcho2, Sentry, echoUAT) mobile cockpit solution is very trendy nowadays, especially in the general aviation sector. Such portable ADS-B devices provide service through EFB application hosted on a mobile

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3150840, IEEE Access

tablet or smartphone. Garmin's GDL-90 is one of the de-facto standards leading the avionics industry, and is one of the main and most used protocol to exchange data between ADS-B devices (e.g., SkyEcho2, Sentry, echoUAT) and EFB applications. GDL-90 is also used in many Integrated Flight Deck (IFD) and Electronic Flight Instrumentation Systems (EFIS) (such as Garmin's G1000 and Avidyne's IFD440/540, EX5000) as well as in many mobile cockpit devices and EFB applications (such as AvPlan, Naviator, Airmate). Any potential vulnerability in GDL-90 poses elevated cybersecurity risks to the avionics systems as well as safety risks to the passengers and crew lives. Researchers have reported several types of security threats regarding ADS-B, such as ghost aircraft, aircraft disappearance, denial of service [4], [5]. However, protocol fuzzing in mobile cockpit systems has not been thoroughly investigated yet, which has motivated us to conduct this study. Our present work is important as it systematically approaches discovering potential bugs and cybersecurity vulnerabilities within GDL-90 implementations. Our main contributions with this work are:

1) To the best of our knowledge, we are the first to propose, implement and execute a systematic fuzzing platform and experiments aiming specifically at GDL-90 protocol (while our method is easily extensible to more avionics and aerospace data-link protocols).
2) We are the first to discover and report safety-critical Denial of Service (DoS) vulnerabilities found in a handful of most popular aviation apps and mobile EFBs resulting from fuzzing the GDL-90 inputs.

The rest of this article is organized as follows. We have discussed different fuzzing aspects in Section II. Then, in Section III we introduce our attacking strategy. We present our results in Section IV. We discuss related works in Section V. Finally, with Section VI, we discuss possible workarounds, future work and conclude this paper.

II. BACKGROUND

In this section we briefly present background technologies and techniques that are involved in our experiments.

A. FUZZING

Fuzzing (or fuzz testing) is an automated software testing method for finding implementation and input sanitization bugs by using intentionally malformed or randomized inputs. It was originally developed by Professor Barton Miller and his team of students in the University of Wisconsin Madison in 1989 [6]. With fuzzing, a generator of sort is used to create random and semi-random (known to be dangerous) data usually sampled from real inputs and it is input to the software, which behaviour is observed. Fuzzing relies on the premise that bugs exists in every program and therefore, consistent and systematic approach will eventually discover them [7]. Fuzzing is a blind testing technique with it's caveats, such as the possibility of missed program paths due to the random nature of the input mutations [8]. In our experiments, we target the GDL-90 protocol, which means that we are employing protocol fuzzing by forging packets with real protocol like format, but with some parts malformed (more on the topic at Section III-D).

As the core fuzzing toolset, in our work we have employed the American Fuzzy Lop (AFL) which is a security-oriented greybox fuzzer originally developed by Michal Zalewski [9]. It is a proven, easy-to-use, stable, and effective fuzzer that utilizes performance optimizations to decrease unnecessary runtime [10]. AFL uses an instrumentation-guided genetic algorithm to fuzz the software-in-test with a brute-force approach. In essence, AFL takes the sample test cases supplied by the user one by one, it trims them and mutates the trimmed versions with traditional fuzzing strategies. The behaviour of the software-in-test is recorded and interesting test cases are recorded for further use and for runtime modifications of the fuzzer [8]. AFL is currently maintained by Google Open Source and it is licensed with Apache License 2.0 [8], [11].

B. GDL-90 PROTOCOL

Garmin DataLink 90 (GDL-90) format is supported by many aviation hardware and software (see Tables 3). Its features is described in RTCA DO-267A as asynchronous High-Level Data Link Control (HDLC)-based messaging structure with some modifications to better suit avionics data interfaces [12], [13]. The basic GDL-90 message format is presented in Figure 1.

Flag Byte	Message ID	Message Data	CRC	Flag Byte

FIGURE 1: GDL-90 message format.

The message starts with a Flag Byte (0x7E) followed by a one-byte Message ID, which specifies the type of message being transmitted. The message type sets the message data content and length. All the message definitions have been listed in Table 1.

TABLE 1: GDL-90 Message IDs.

Message ID	Message Name
0	Heartbeat
2	Initialization
7	Uplink Data
9	Height Above Terrain
10	Ownship Report
10	Ownship Geometric Altitude
20	Traffic Report
30	Basic Report
31	Long Report

A two-byte frame check sequence (16-bit CRC, LSB first) is calculated for the data and appended to the message,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3150840, IEEE Access

and the message ends with another Flag Byte. If a Flag Byte (0x7E) or a Control-Escape Character (CEC, 0x7D) is present in the original message, the message byte is XOR'd with 0x20, and a CEC is prefixed to it. Thus the integrity of the message is preserved. The receiving end checks the incoming traffic for the Flag Bytes and captures the data between them. The captured data is inspected for CECs. If CEC is found, the CEC is discarded, and the byte after it is XOR'd again to return its old form properly. CRC for the message data part of the message is calculated and verified. If deemed valid, the message is ready for use. In operation, GDL-90 devices transmit a heartbeat message once every second followed by an ownship report. In between these "pulses", other messages such as traffic reports can be transmitted. In our experiments, we focused on three message types

- Heartbeat
- Traffic report
- Ownship report

A heartbeat message is used for the devices to indicate that they are operational and to submit information about their status. Two status bytes in the message tell information about the transmitter in a boolean fashion. This information includes battery low, GPS fix, maintenance requirement, etc., flags. A timestamp is also present in the message after the status bytes.

Traffic reports are in the output in each second for each proximate target. GDL-90 expects at least 32 simultaneous targets to be handled, but more can be processed depending on the uplink configurations and the interface baud rate. Traffic report data uses 27 bytes to represent every needed attribute. Table 2 shows the fields of the traffic report data in order.

An ownship report message follows the traffic report format, and it is always in output even without a proper GPS fix. It broadcasts the transmitter information to the network.

C. GDL90 PROTOCOL EXTENSIONS

Some vendors have their own interpretation of the protocol outside of the Garmin standard. For example, Uavionix's SkyEcho2 mainly uses the standard messaging types, but it outputs its ownship message with a message type code 101. On the other hand, ForeFlight's Sentry extends the protocol and does not communicate with the standard message types. Sentry transmits messages with IDs 37 and 38, which are longer than the standard heartbeat, ownship, and traffic messages. Most likely containing multiple message types in a single packet. The ForeFlight EFB supports both devices. It broadcasts messages to the network. When the app is accepting traffic it sends "i-want-to-play-ffm-udp", and sends "i-cannot-play-ffm-udp" when it goes to sleep. It also identifies itself to the network by broadcasting "App: ForeFlight, GDL90: port :4000" messages. For our experiments, we did not delve deeper into the ForeFlight protocol as it was not necessary. We were able to capture, modify, resend, and receive Sentry packets just like with the other devices. Thus the integration with AFL was quite straightforward. Figure 2 shows Skyecho decoded heartbeat packet in Wireshark.

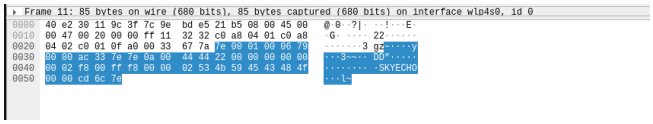

FIGURE 2: Heartbeat messages of "SkyEcho2" specific GDL-90 extension as captured and decoded in Wireshark software.

Figure 3 depicts the system diagram of Garmin G1000 – a real-world EFIS / IFD / avionics system. It is important to note that GDL-90 inputs go to the GIA 63/63W avionics unit that is also directly controlling the auto-pilot systems such as Bendix/King KAP-140 [14]. Therefore, any GDL-90 vulnerabilities present within the avionics units could potentially have a direct effect on the auto-pilot systems. Therefore it is important to discover such GDL-90 (and other data-link protocols) vulnerabilities as fast and as efficient as possible, for example using our approach and results.

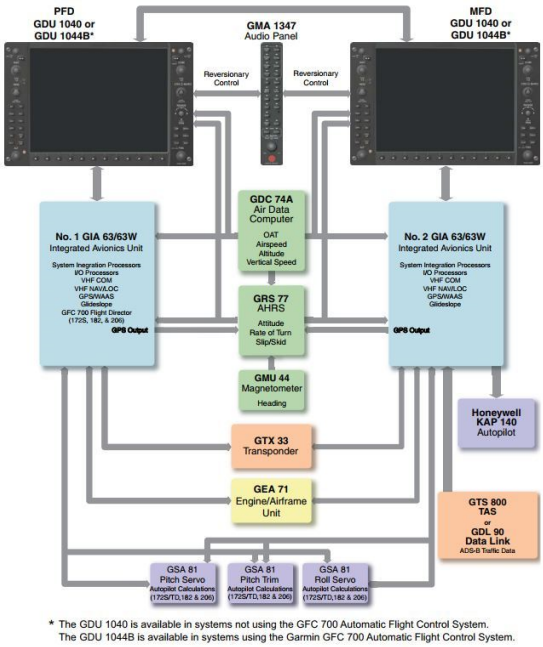

FIGURE 3: System diagram of "Garmin G1000" EFIS/IFD [15] – the GDL-90 inputs going into "No.2 GIA 63/63W" that in turn controls the auto-pilot "Honeywell KAP 140" [14].

III. FUZZING ATTACKS ON GDL-90

A. DIAGRAMS OF OUR APPROACH

In Figure 4 we present the high-level diagram [1] of where GDL-90 outputs and inputs are connected in real-world sys-

[1]This setup is part of a larger pentesting platform for aviation/avionics and maritime technologies [5].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3150840, IEEE Access

TABLE 2: GDL-90 Traffic/Ownship Report fields.

Field	Description	Length (bits)
Traffic Alert Status	0: No alert, 1: Traffic alert for this target, 2-15: Reserved	4
Target Identity (type)	0: ADS-B with ICAO address 1: ADS-B with Self-assigned address 2: TIS-B with ICAO address 3: TIS-B with track file ID. 4: Surface Vehicle 5: Ground Station Beacon 6-15: Reserved	4
Participant Address	Unique address	24
Latitude and Longitude	Encoded range -180 to 180 degrees, resolution of approximately 2.14577×10^{-5}	24×2
Altitude	Pressure altitude (referenced to 29.92 inches Hg), encoded using 25-foot resolution, offset by 1,000 feet. 0xFFF is invalid/unavailable	12
Miscellaneous Indicators	Bits 0 and 1: Additional information for the Track/Heading field, Bit 2: Report derived from ADS-B or extrapolated from previous report, Bit 4: Air/Ground State	4
Integrity and Accuracy	Integrity and accuracy of the traffic reported (in nautical mile thresholds)	4×2
Horizontal Velocity	Velocity in knots, above 4094 knots, the value will hold at 0xFFE	12
Vertical Velocity	Velocity in 64 feet per minute, +/- 32,578 feet per minute	12
Track/Heading	Weighted heading value, resolution 360/256 (approx. 1.4 degrees)	8
Emitter Category	Type of the vehicle in use represented by an integer. Values 0-21 are in use and the rest are reserved	8
Call Sign	8 ASCII characters (0-9, A-Z), space is only used for padding in the end	64
Emergency/Priority Code	0: no emergency 1: general emergency 2: medical emergency 3: minimum fuel 4: no communication 5: unlawful interference 6: downed aircraft 7-15: reserved	4
Reserved	Reserved for future use	4

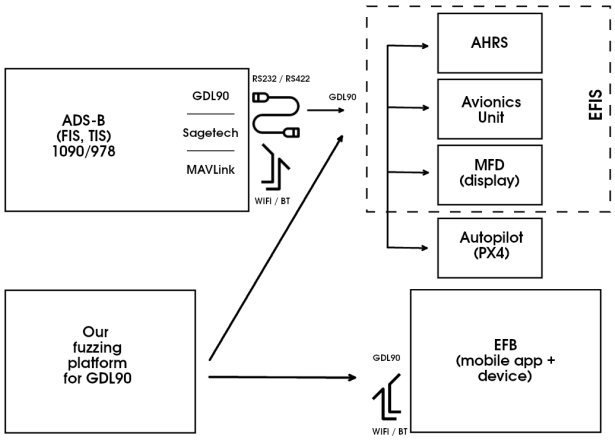

FIGURE 4: Overview of the GDL-90 test-bench and the positioning of our fuzzing platform (for GDL-90 and similar avionics data-link protocols).

tems and where our platform can be connected during the execution of GDL-90 fuzzing. It is important to note that discovering or triggering such protocol implementation vulnerabilities does not necessarily require physical or adjacent proximity. In our another research, we have demonstrated that carefully crafted wireless ADS-B communications can be used to achieve the same goals, i.e., crash EFB/ADS-B apps or ADS-B avionics devices which can be due to GDL-90 or ADS-B vulnerabilities, or a handful of other reasons [5]. This is possible because many ADS-B devices with ADS-B IN function provide processed data using GDL-90 protocol encoding.

B. ADVANTAGES OF OUR APPROACH

Using the GDL-90 fuzzing approach that we developed and proposed in this paper has the following main advantages:

1) Does not require aviation-spectrum wireless transmission (e.g., ADS-B) – avoids any radio interference, lowers the costs as SDR devices are not required (i.e., works directly at GDL-90 receiving point).
2) Is not limited to the capacity of radio channels – thus can perform fuzzing/testing at considerable higher speeds (e.g., WiFi/ethernet has higher throughput than ADS-B RF link).
3) Works closer to the source of the possible GDL-90 implementation problems – thus avoids the extra layer(s) introduced by higher protocols processing chains (such as ADS-B) which could be sources of possible bottlenecks, false negatives/positives, and air-transmission regulatory challenges.

C. OVERALL HARDWARE-SOFTWARE SETUP

Our attacks were made simple by the fact that the common GDL-90 enabled WiFi ADS-B devices (such as SkyEcho2, echoUAT, Sentry) use connectionless UDP packets to send data. Therefore, we can easily capture, manipulate, and resend the packets to the applications without issues. First, we observed the packets transmitted in the WiFi networks created by the Sentry and SkyEcho2 with a network packet inspection tool called Wireshark [16]. We applied GDL-90 dissector [17] lua-script to Wireshark to identify and analyze the packets. We transmitted ADS-B traffic messages via HackRFOne to the receivers as well. We copied the required messages from the packet captures and saved them as samples for the fuzzer. Depending on the device and its configuration, we either left the different message types as separate samples or left them as one in the case of the Sentry. In addition to the samples we gathered from real device

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3150840, IEEE Access

networks, we also utilized Eric Dey's GDL-90 code [18] to simulate Stratus [19] and SkyRadar [20] ADS-B receivers and created samples for those. In total, we tried four different samples with the applications. Some applications only worked with one sample-specific sample set. The simulated SkyRadar sample set was deemed the best generalizing of the four, and it was the most widely used in our tests.

We were inspired by Eric Dey's GDL-90 code [18] and made our own GDL-90 sender script for fuzzing purposes. We chose AFL as our fuzzer of choice since we were adamant that the input coverage with AFL would be sufficient. We set up our environment as a Docker container with AFL and our sender/fuzzing script. With our sender script, the target IP address and the target port must be set at the beginning. When the parameters are set, we can start fuzzing. As we are using UDP packets over WiFi, the applications in the mobile phone end are not aware that the device at the other end is not legitimate; therefore, the testing is realistic. However, as we have no feedback from the mobile device through the network to the fuzzer, we cannot have AFL recording the exact input that made an app crash. We can only observe the applications. Running the fuzzer over the network with a packet sending delay made the fuzzing quite slow for AFL standards. However, the applications that were affected the most did crash within the first 60 minutes of the test. For the initial test, the target and the attacking PC were both connected to the same home network via a WiFi access point running OpenWRT 17.01.0 [21] or with ethernet to the router.

Overall, our test setup works on the one-click-test principle. After the Docker container is built, a test can be started by running a script with four arguments; IP-address of the attacked device, UDP-port (4000 or 43211 in our tests), sample folder (one of our four offerings), and output folder (arbitrary, useful for resuming long fuzzing sessions). Logs are saved to the specified output folder. With the inclusion of Docker, the setup is easy as every component is installed automatically. Figure 5 shows a status display during the test.

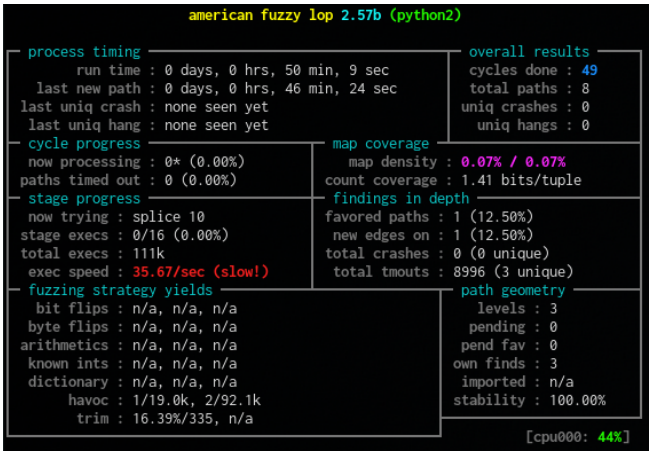

FIGURE 5: Example of an AFL run status.

D. AFL SETUP

We used AFL's Python implementation (python-afl v.0.7.3) and the latest AFL as of date (afl-fuzz v.2.57b) in our tests. As our test setup is quite slow, we specified "quick and dirty"-mode (-d option), which skips deterministic steps and usually yields faster results. This limits the depth that we could achieve with the tests, but we discovered that this mode was perfectly adequate for many applications to falter. With the non-deterministic mode on and with the sample variety being low, our most prolonged 1-hour fuzzing sessions reached at least 50 cycles. A cycle in AFL means that the fuzzer went through all the interesting test cases [22]. Therefore, we would argue that the tests were quite thorough within the limitations of the samples we acquired. We observed that the crashes occurred at several stages of the fuzzing cycles. Even if the test applications did not crash, the usability of the data it presented was greatly hindered due to malformed input data (details in the result Section IV).

E. GDL-90 FUZZING TARGETS

In Table 3, we present a comprehensive list of the targeted software. Most of our efforts were put to targeting mobile EFB apps, but some open-source tools were also tested. For Eric Dey's GDL-90 code [18], we targeted the decoding script only.

TABLE 3: List of software exposed to fuzzing attack (Software Under Test).

Name	Price	Version (Android / iOS)	Installs (Android / iOS)
OzRunways	Free	v.4.5.6 / v.10.1.9	50k+ / –
iFlightPlanner	Free	– / v.4.5.6	– / –
Airmate	Free	v.1.6.1 / v.2.3	50k+ / –
AvPlan	$76.16/year	v.1.3.28 / v.8.1.2	5k+ / –
Levil Aviation	Free	– / v.1.3	– / –
FLYQ EFB	Free	– / v.5.0	– / –
EasyVFR4	Free	v.4.0.870 / v.4.0.899	100+ / –
Horizon	Free	v.3.0 / v.3.0	10k+ / –
ForeFlight	Free	– / v.13.4	– / –
Pilots Atlas	$89.99/year	– / v.5.13.0	– / –
Xavion	Free	– / v.2.81	– / –
SkyDemon	$162/year	v.3.15.0 / v.3.15.3	100k+ / –
Stratus Insight	$99.99/year	– / v.5.17.3	– / –
Naviator	$34.99/year	v.4.2.2 / –	100k+ / –
Traffic (by Control-J) traffic	Free	v.0.0.2.4 / –	10+ / –
Eric Dey's GDL-90 gdl90etdey	Free	github repo commit d7e5936	–

F. HIGH-LEVEL GDL-90 ATTACK DESCRIPTION

For example, a possible cybersecurity attack involving vulnerable GDL-90 implementations could look as follows:

1) At research time: An exploitable GDL-90 vulnerability is first discovered (e.g., using our implementation-independent GDL-90 protocols fuzzing approach).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3150840, IEEE Access

2) At design/manufacturing time: An adversary designs and puts to market an ADS-B-capable and GDL-90-compatible "backdoored" device that contains the GDL-90 exploitation payloads and attack vectors. The "backdoor" could be implemented at the hardware or at the firmware level in such a way to avoid the detection at (re-)certification time (similar to the Volkswagen's emission ECU manipulation scandal [23]).
3) At usage time: The "backdoored" ADS-B-capable device sends or activates the GDL-90 exploitation payload. Such exploitation payloads could be activated conditionally: at certain altitudes, within certain geo-fence areas, upon receiving a "secret knock" ADS-B message, etc.
4) At usage time: Alternatively, the discovered GDL-90 vulnerability can be reconstructed back to a specially-crafted triggering ADS-B payload/message. Therefore it may be even possible to trigger the GDL-90 vulnerability without "backdoored" hardware, just by simply sending a specially-crafted ADS-B payload/message.
5) Ultimately, backdoors have been shown to be implanted even in military-grade chips [24], therefore it is more than reasonable to believe backdoor implanting is also feasible for ADS-B devices destined for avionics/EFIS/IFD/EFB setups within commercial/general aviation and amateurs airplanes.

IV. RESULTS

The fuzzing result are presented in Table 4. Out of 15 tested mobile EFB applications under test, 6 apps crashed (4 iOS-only, 2 iOS+Android) and 2 apps became unresponsive (1 iOS-only, 1 Android-only). In addition to mobile EFB apps, Eric Dey's open-source GDL-90 [18] decoder experienced several dozen of unique crashes during a day long fuzzing session on a normal PC (Linux). We focused only on fuzzing Eric Dey's GDL-90 decoder leaving its network component out of the equation. The unique errors and crashes that we recorded were related to different inputs generating Python assertion statement failures that in turn were due to the faulty lengths of the messages (i.e., exactly the aim of fuzzing tests in general to find such issues). These results allow us to assume that Eric Dey's open-source GDL-90 [18] could pose stability, availability, and DoS-resiliency issues if deployed or operated "as-is" in real-world systems and devices.

In one of our recent work [5], we tested almost the same set of mobile apps and devices for DoS attacks via the ADS-B layer, and found 6 of the mobile apps from Table 4 to be impacted by ADS-B IN DoS, possibly affecting over 200,000 mobile application installs worldwide. In [5] altogether we have tested 68 different ADS-B configurations (mobile and non-mobile) for ADS-B IN DoS attack. We managed to crash 25% of them mostly within 2 minutes, while overall the DoS attack impacted 51.47% of tested configurations. In comparison, the fuzzing results presented in this paper have similarly worrying results in terms of aviation safety and lack of resiliency to cybersecurity attacks such as DoS. Attacks on core sub-system availability (such as DoS) pose high risks to safety-critical and mission-critical systems such as avionics and aerospace.

TABLE 4: Details of mobile applications (apps) considered for "Attacked software".

App Name	GDL-90: Android	GDL-90: iOS	Comparison with our ADS-B-level DoS attacks [5]
OzRunways	CRA (once)	CRA	CRA
Stratus Insight	NA-P	CRA	CRA
iFlightPlanner	NA-P	CRA	DNW
AirMate	DNW	CRA	CRA
AvPlan	CRA / UNR	CRA	CRA
Levil Aviation	NA-P	CRA	DNT
FlyQ EFB	NA-P	UNR	DNC
EasyVFR4	DNC	DNC	DNC
Horizon	DNC	DNC	DNT
ForeFlight	NA-P	DNC	CRA
Pilots Atlas	NA-P	DNC	DNC
Xavion	NA-P	DNC	DNT
Traffic (by Control-J)	DNC	NA-P	DNT
Naviator	UNR	NA-D	DNW
SkyDemon	DNC	DNC	DNW

Android = Samsung Galaxy A21s, Android v 11
iOS = Apple iPhone SE, iOS v 13.3
Acronyms: CRA=Crash; UNR=Unresponsive/Hang; DNC=Did Not Crash; DNT=Did Not Test; DNW=Did Not Work (e.g., did not connect to hardware, did not receive data); NA-G=Not Available for this Geography/Country/Region; NA-P=Not Available for this Platform; NA-D=Not Available for this Device.

A. VISUAL OBSERVATIONS

All of the mobile application crashes were observed by visually inspecting the device/software under test. The crashes happened either themselves or while trying to operate software (e.g., any touch input, move the map, zoom in/out) while the test was running. For each tested configuration that was impacted, the crashes were observed and confirmed at least three times (unless noted otherwise) before registering the result.

Although FlyQ did not crash, it went unresponsive and had to be closed by the user. OzRunways on Android did crash, but the result was not consistently repeated. Naviator on Android phone did not crash during the test. However, it consistently closed the GDL-90 ADS-B input on an error state in each of our attempts and only recovered after a restart. Otherwise, the application remained functional. Most of the test applications showed some abnormal behaviour, e.g., irrationally flinching map screen, fluctuating GPS data (due to GPS positioning taken from the GDL-90 messages), alerts (due to plane proximity or altitude readings), and other non-standard or device operator alerting behaviour. Therefore, DNCs in Table 4 should not be interpreted as conclusive evidence of the stability of the application [25]. The application that did not crash (DNC) during our present tests may crash on some other sample data or testing methods.

V. RELATED WORK

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3150840, IEEE Access

A. SOFTWARE FUZZING

Reliable and efficient aerial communication is at the heart of aerospace safety. Any defects in this safety-critical technology may cost human life and property. However, modern protocols and the accompanied software is not always up to the task. Several studies have shown numerous viable attacks on these protocols and software [4], [26]. Developers, researchers, and hackers are using many tools to find out the security vulnerabilities of this kind of mission-critical system. Here we discuss a few of them.

The success stories and the open-source nature of AFL have encouraged researchers to customize this fuzzer for different tasks [10]. Numerous studies have added many functionalities to the AFL (e.g., pathfinding, sample creation, and coverage) to improve its performance and effectiveness [27]–[35]. The support for AFL has been added for Commercial Off-The-Shelf (COTS) binaries [36]–[38]. AFL has also received modifications for its parallel run capabilities [39].

B. ATTACKS AND FUZZING ON AVIONICS DATA-LINK

Micro Air Vehicle Communication (MAVLink) Protocol is a bidirectional communication protocol which used in drones and ground control stations. It offers a variety of message types that can be transmitted reliably in an efficient package [40]. However, Domin et al. reported a crash of MAVLink capable software in their protocol fuzzing tests in 2016 [41]. They were able to crash a virtual drone with a random payload by incrementally increasing the payload bytes from 1 to 255, thus increasing the length of the whole message. An open-source MAVLink fuzzing software is available [42].

PX4 is a widely avaible and extremely popular flight controller that also supports MAVLink protocol as well as data from ADS-B IN capable devices (such as Aerobits AERO, uAvionix pingRX). Alias Robotics [43] perform a general cybersecurity overview of PX4 from threat modelling and static analysis perspectives, as well as introduce in this context the Robot Vulnerability Database (RVD). Subsequently, Jang et al. [44] performed a thorough static analysis for various PX4 firmware codebases.

Other communication protocols are also used for drones in particular. Rudo and Zeng [45] showed fuzzing results on File Transfer Protocol/Session Initiation Protocol (FTP/SIP) and Session Description Protocol (SDP) embedded within consumer-grade drones. They raised concerns about the state of security with commercial drone software. They demonstrated Global Positioning System (GPS) navigation and other subsystem failures (e.g., video feed and motor issues). Internet-of-Things (IoT) and embedded devices have been shown to be quite vulnerable by multiple studies [46], [47].

In regards to the drone security issues, Kim et al. published their Robotic Vehicle (RV) fuzzing tool called RV-Fuzzer [48]. This tool intended to highlight missing or faulty validation checks for control inputs. These bugs and missing features may cause physical disruptions, such as mission failures or crashes, on RV's, such as drones, if exploited. The authors constructed the RVFuzzer to employ three distinct strategies for searching input validation bugs, e.g., control parameter mutation, one-dimensional mutation, and multi-dimensional mutation. Throughout their evaluation, they discovered 89 input validation bugs from two control programs. Since the attacks do not require any code injection or other invasive procedures, they cannot be detected by security solutions [48]. Hence more specific code improvements and internal security audits for source code during development are required.

C. ATTACKS ON AVIONICS SYSTEMS AND PROTOCOLS

There has been adamant scrutiny towards ADS-B communication security from the research community over the years. In 2004, Korzel et al. [49] demonstrated issues in the protocol with it's data integrity due to erroneous input and data dropouts. Concerns over the authenticity, security, confidentiality, and integrity of the protocol have been periodically raised since [50]–[52].

Attacks against the ADS-B protocol have been demonstrated frequently by several researchers. Costin and Francillon [4] showcased the first practical ADS-B message injection and spoofing attacks. Schäfer et al. [26] exposed several attacks such as ghost aircraft attacks and virtual trajectory modification with budget devices. Sjödin and Gruneau [53] used HackRF SDR to demonstrate data injection and flooding attacks on Sentry ADS-B transceiver. They concluded that the device does not validate the messages from the ADS-B protocol. McCallie et al. [54] classified attacks and explored the consequences of the them resulting in worrying results.

Portable ADS-B transceivers (e.g., SkyEcho2, Sentry, echoUAT), which are operated with iPads and other tablets are favored by many general aviation pilots due to the ease of setup, ease of use, and affordable pricing. As these devices are not per-se part of the onboard avionics, Lundberg et al. [55], [56] pointed out that these devices do not, nor do they need to, meet the standards of traditional avionics (e.g., RTCA, ARINC, EUROCAE). The authors also found vulnerabilities on all of their test samples and recommended further product development steps to the device and software designers.

VI. CONCLUSION

In this paper, we have have studied the impact of GDL-90 protocol fuzzing on a range of popular mobile EFBs and some standard PC software. Our results show a worrying lack of security in many EFB applications where the security is directly related to aircraft's safety navigation. Out of 16 tested configurations, our avionics pentesting platform managed to crash or otherwise impact 9 (or 56%) of those. The observed problems manifested as crashes, hangs, and abnormal behaviours of the EFB apps and GDL-90 decoders during the fuzzing test. The consistency of our test results on a heterogeneous and representative set of different EFBs (Android, iOS, Linux platforms) support the reliability of our approach and results. DoS attacks can be devastating for mission-critical

systems such as avionics and aerospace, where availability and reliability of the system is crucial. However, we hope that our results and presented methodology can motivate the standardization and regulatory bodies, as well as industry and air-traffic organizations, to improve the requirements and implementation checks of avionics devices and apps in connection to resiliency to cybersecurity attacks, and in particular resiliency to DoS attacks. To ensure adequate safety in such type of mission critical system, multidimensional security measures need to be taken. For avionics devices and the related software/firmware, defence against cyberattacks should be considered a continuous process where research and development need to be run along with the operation.

ACKNOWLEDGMENT

The authors acknowledge the grants of computer capacity from the Finnish Grid and Cloud Infrastructure (persistent identifier urn:nbn:fi:research-infras-2016072533).

Major parts of this research supported by cascade funding from the Engage consortium's Knowledge Transfer Network (KTN) project "Engage - 204 - Proof-of-concept: practical, flexible, affordable pentesting platform for ATM/avionics cybersecurity" (SESAR Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No 783287). All and any results, views, and opinions presented herein are only those of the authors and do not reflect the official position of the European Union (and its organizations and projects, including Horizon 2020 program and Engage KTN).

Part of this research was supported by a grant from the Decision of the Research Dean on research funding within the Faculty (07.04.2021) of the Faculty of Information Technology of University of Jyväskylä (The authors thank Dr. Andrei Costin for facilitating and managing the grant).

Hannu Turtiainen also thanks the Finnish Cultural Foundation / Suomen Kulttuurirahasto (https://skr.fi/en) for supporting his Ph.D. dissertation work and research (under grant decision no.00211119) and the Faculty of Information Technology of the University of Jyvaskyla (JYU), in particular, Prof. Timo Hämäläinen, for partly supporting and supervising his Ph.D. work at JYU in 2021–2022.

REFERENCES

[1] "No Kidding: ADS-B Deadline of Jan. 1, 2020, is Firm," https://www.faa.gov/news/updates/?newsId=90008, accessed: 2021-06-11.
[2] EASA, "Easa seasonal technical commission," https://www.easa.europa.eu/sites/default/files/dfu/EASA_STC_NEWS_JUNE_2018.pdf, 2018, accessed: 2021-03-02.
[3] N. Demidovich, "Federal Aviation Administration Incremental Flight Testing of Automatic Dependent Surveillance-Broadcast (ADS-B) Prototype for Commercial Space Transportation Applications," 2015.
[4] A. Costin and A. Francillon, "Ghost in the Air (Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices," Black Hat USA, 2012.
[5] S. Khandker, H. Turtiainen, and A. Costin, "Practical denial-of-service and combined high-level attacks on real-world ADS-B, ATC, ATM hardware and software," 2021, (Preprint).
[6] B. Miller, "Fuzz testing of application reliability." [Online]. Available: http://pages.cs.wisc.edu/~bart/fuzz/
[7] OWASP, "Fuzzing." [Online]. Available: https://owasp.org/www-community/Fuzzing
[8] G. O. Source, "Github.com: Afl." [Online]. Available: https://github.com/google/AFL
[9] M. Zalewski, "American Fuzzy Lop." [Online]. Available: https://lcamtuf.coredump.cx/afl/
[10] ——, "American Fuzzy Lop - The bug-o-rama trophy case." [Online]. Available: https://lcamtuf.coredump.cx/afl/#bugs
[11] A. S. Foundation, "Apache license, version 2.0." [Online]. Available: https://www.apache.org/licenses/LICENSE-2.0
[12] RTCA DO-267: Minimum Aviation System Performance Standards (MASPS) for Flight Information Services-Broadcast (FIS-B) Data Link, RTCA, 2014.
[13] GDL 90 Data Interface Specification, Garmin, 2007.
[14] Bendix/King, "KAP 140 Autopilot System," 2021. [Online]. Available: https://www.bendixking.com/content/dam/bendixking/en/documents/document-lists/downloads-and-manuals/006-18034-0000-KAP-140-Pilots-Guide.pdf
[15] Garmin, "G1000 System," 2021. [Online]. Available: https://buy.garmin.com/en-US/US/p/6420
[16] "Wireshark homepage." [Online]. Available: https://www.wireshark.org/
[17] B. Kyser, "Github.com: gdl90dissector." [Online]. Available: https://github.com/BrantKyser/gdl90Dissector
[18] E. Dey, "Github.com: gdl90." [Online]. Available: https://github.com/etdey/gdl90
[19] "Stratus ADS-B Receiver," Stratus. [Online]. Available: https://stratusbyappareo.com/products/stratus-ads-b-receivers/
[20] "SkyRadar ADS-B Receiver," SkyRadar Radenna LLC. [Online]. Available: https://www.skyradar.net/skyscope-receiver/receiveroverview.html
[21] "OpenWrt Project," OpenWrt. [Online]. Available: https://openwrt.org/
[22] "AFL User Guide," Google. [Online]. Available: https://afl-1.readthedocs.io/en/latest/user_guide.html
[23] M. Contag, G. Li, A. Pawlowski, F. Domke, K. Levchenko, T. Holz, and S. Savage, "How they did it: An analysis of emission defeat devices in modern automobiles," in IEEE Symposium on Security and Privacy (SP). IEEE, 2017.
[24] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2012.
[25] M. Muench, J. Stijohann, F. Kargl, A. Francillon, and D. Balzarotti, "What You Corrupt Is Not What You Crash: Challenges in Fuzzing Embedded Devices," in NDSS, 2018.
[26] M. Schäfer, V. Lenders, and I. Martinovic, "Experimental analysis of attacks on next generation air traffic communication," in Applied Cryptography and Network Security, M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds. Springer Berlin Heidelberg, 2013.
[27] C. Lemieux and K. Sen, "Fairfuzz: A targeted mutation strategy for increasing greybox fuzz testing coverage," in 33rd ACM/IEEE International Conference on Automated Software Engineering, 2018.
[28] N. Nichols, M. Raugas, R. Jasper, and N. Hilliard, "Faster fuzzing: Reinitialization with deep neural models," arXiv preprint arXiv:1711.02807, 2017.
[29] K. Patil and A. Kanade, "Greybox fuzzing as a contextual bandits problem," arXiv preprint arXiv:1806.03806, 2018.
[30] R. K. Prakash, I. Vasudevan, I. Indhuja, T. Thangarasan, and C. Krishnan, "Hardiness sensing for susceptibility using american fuzzy lop," in ITM Web of Conferences, vol. 37. EDP Sciences, 2021.
[31] M. Rajpal, W. Blum, and R. Singh, "Not all bytes are equal: Neural byte sieve for fuzzing," arXiv preprint arXiv:1711.04596, 2017.
[32] L. Sun, X. Li, H. Qu, and X. Zhang, "AFLTurbo: Speed up Path Discovery for Greybox Fuzzing," in IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), 2020.
[33] J. Wang, B. Chen, L. Wei, and Y. Liu, "Superion: Grammar-aware greybox fuzzing," in IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019.
[34] X. Yuan, L. Pan, and S. Luo, "Binary fuzz testing method based on lstm," in IOP Conference Series: Materials Science and Engineering, vol. 612. IOP Publishing, 2019.
[35] G. Zhang and X. Zhou, "AFL extended with test case prioritization techniques," International Journal of Modeling and Optimization, vol. 8, 2018.

[36] Y. Chen, D. Mu, J. Xu, Z. Sun, W. Shen, X. Xing, L. Lu, and B. Mao, "Ptrix: Efficient hardware-assisted fuzzing for cots binary," in ACM Asia Conference on Computer and Communications Security, 2019.
[37] S. Dinesh, N. Burow, D. Xu, and M. Payer, "Retrowrite: Statically instrumenting cots binaries for fuzzing and sanitization," in IEEE Symposium on Security and Privacy (SP). IEEE, 2020.
[38] Y. Zheng, A. Davanian, H. Yin, C. Song, H. Zhu, and L. Sun, "FIRM-AFL: high-throughput greybox fuzzing of iot firmware via augmented process emulation," in 28th {USENIX} Security Symposium, 2019.
[39] J. Ye, B. Zhang, R. Li, C. Feng, and C. Tang, "Program state sensitive parallel fuzzing for real world software," IEEE Access, 2019.
[40] MAVLink.io, "Mavlink developer guide," 2021. [Online]. Available: https://mavlink.io/en/
[41] K. Domin, I. Symeonidis, and E. Marin, "Security analysis of the drone communication protocol: Fuzzing the MAVLink protocol," ORBIlu, 2016.
[42] Auterion, "Github.com: MAVLink Fuzz Testing," 2019. [Online]. Available: https://github.com/Auterion/mavlink-fuzz-testing
[43] Alias Robotics, "The Cybersecurity Status of PX4." [Online]. Available: https://aliasrobotics.com/files/cybersecurity_status_PX4.pdf
[44] J.-h. Jang, Y.-s. Kang, and J.-h. Lee, "Static Analysis and Improvement Opportunities for Open Source of UAV Flight Control Software," Journal of the Korean Society for Aeronautical & Space Sciences, vol. 49, 2021.
[45] D. Rudo and D. Zeng, "Consumer UAV Cybersecurity Vulnerability Assessment Using Fuzzing Tests," arXiv preprint arXiv:2008.03621, 2020.
[46] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," in 23rd USENIX Security Symposium, 2014.
[47] A. Costin, A. Zarras, and A. Francillon, "Automated dynamic firmware analysis at scale: a case study on embedded web interfaces," in 11th ACM on Asia Conference on Computer and Communications Security, 2016.
[48] T. Kim, C. H. Kim, J. Rhee, F. Fei, Z. Tu, G. Walkup, X. Zhang, X. Deng, and D. Xu, "RVFuzzer: Finding input validation bugs in robotic vehicles through control-guided testing," in 28th {USENIX} Security Symposium, 2019.
[49] J. Krozel, D. Andrisani, M. Ayoubi, T. Hoshizaki, and C. Schwalm, "Aircraft ADS-B data integrity check," in AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum, 2004.
[50] K. Samuelson, E. Valovage, and D. Hall, "Enhanced ADS-B Research," in IEEE Aerospace Conference. IEEE, 2006.
[51] R. G. Wood, "A security risk analysis of the data communications network proposed in the NextGen air traffic control system," Ph.D. dissertation, Oklahoma State University, 2009. [Online]. Available: https://search.proquest.com/dissertations-theses/security-risk-analysis-data-communications/docview/305083310/se-2?accountid=11774
[52] L. Purton, H. Abbass, and S. Alam, "Identification of ADS-B System Vulnerabilities and Threats," 33rd Australasian Transport Research Forum (ATRF), 2010.
[53] A. Sjödin and M. Gruneau, "The ADS-B protocol and its' weaknesses: Exploring potential attack vectors," 2020.
[54] D. McCallie, J. Butts, and R. Mills, "Security analysis of the ADS-B implementation in the next generation air transportation system," International Journal of Critical Infrastructure Protection, vol. 4, 2011.
[55] D. Lundberg, B. Farinholt, E. Sullivan, R. Mast, S. Checkoway, S. Savage, A. C. Snoeren, and K. Levchenko, "On the security of mobile cockpit information systems," in ACM SIGSAC Conference on Computer and Communications Security, 2014.
[56] D. A. Lundberg, "Security of ADS-B Receivers," Ph.D. dissertation, UC San Diego, 2014.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3150840, IEEE Access

HANNU TURTIAINEN received his M.Sc. in Cybersecurity in 2020 and he is currently working towards his Ph.D. in Software and Communication Technology at the University of Jyväskylä, Finland. His research topic is Machine Learning and Artificial Intelligence in the Cybersecurity and Digital Privacy field. He holds a B.Sc. in Electronics Engineering from the University of Applied Sciences, Jyväskylä Finland. Hannu Turtiainen is also working in the IoT field as a Cybersecurity and Software Engineer in Binare.io, a deep-tech cybersecurity spin-off from the University of Jyväskylä.

DR. ANDREI COSTIN is currently a Senior Lecturer/Assistant Professor in Cybersecurity at University of Jyväskylä (Central Finland), with a particular focus on IoT/firmware cybersecurity and Digital Privacy. He received his PhD in 2015 from EURECOM/Telecom ParisTech under co-supervision of Prof. Francilon and Prof. Balzarotti. Dr. Costin has been publishing and presenting at more than 45 top international cybersecurity venues, both academic (Usenix Security, ACM ASIACCS, etc.) and industrial (BalckHat, CCC, HackInTheBox, etc.). He is the author of the first practical ADS-B attacks (BlackHat 2012) and has literally established the large-scale automated firmware analysis research areas (Usenix Security 2014) - these two works are considered seminal in their respective areas, being also most cited at the same time. Dr. Costin is also the CEO/co-founder of Binare.io, a deep-tech cybersecurity spin-off from University of Jyväskylä, focused on innovation and tech-transfer related to IoT cybersecurity.

SYED KHANDKER received his M.Sc. degrees in Web Intelligence and Service Engineering from the University of Jyväskylä, Finland, in 2016. Since his childhood, he has been a radio enthusiast and holds an amateur radio operator license. He is currently pursuing his doctoral degree from the Faculty of Information Technology, University of Jyväskylä, Finland. His research area spans the field of RF Fingerprint positioning, Automatic Dependent Surveillance-Broadcast, Automatic Identification System, Wireless Communications, and Artificial Intelligence. He has authored 4 Journal and conference publications.

PROF. TIMO HÄMÄLÄINEN has over 25 years of research and teaching experience related to computer networks. He has lead tens of external funded network management related projects. He has launched and leads Master Programs in the University of Jyväskylä (currently SW and Comm. Eng,) and teaches network management related courses. He has more than 200 internationally peer reviewed publications and he has supervised 36 Ph.D theses. His current research interests include wireless/wired network resource management (IoT, SDN, NFV) and network security.