



# Table of contents

Overview .....	1
Introduction .....	1
Background and problem statement .....	2
Overview of the Agent Instrumentation Framework.....	3
Requirements, architecture, and configuration.....	3
Synchronous mode.....	4
Asynchronous mode.....	4
Examples of Agents.....	5
AgentCoreDump.....	5
AgentLogTailer.....	5
AgentProcessMonitor.....	5
AgentPID.....	6
AgentAddressSanitizer.....	6
AgentValgrind.....	6
An example <code>config.json</code> configuration file .....	6
Implementation and test results.....	7
Bluetooth fuzzing .....	8
Wireless fuzzing .....	9
Fuzzing MQTT.....	10
File format fuzzing .....	12
Conclusion.....	12

# Overview

In the past few years, cyber security has become more intertwined into each step of the automotive development process. In



By contrast, in this paper, we focus on fuzz testing of the connected car, specifically in-vehicle infotainment (IVI) systems and telematics units. We present the concept of the Agent Instrumentation Framework, which can be used to better instrument these systems to allow for more efficient and accurate fuzz testing. That is, for these types of systems, additional approaches can be taken to improve instrumentation. Since these systems are typically based on operating systems providing more functionality, such as Linux and Android,<sup>4</sup> using the appropriate tools, it is possible to collect information from the SUT to determine whether any exceptions were detected during fuzz testing. Additionally, more details about the detected exceptions can be fed back to the fuzz testing tool and stored in the log file. This additional information helps developers better understand and identify the root cause and eventually fix the problem.

In this paper:

- We introduce the Agent Instrumentation Framework and explain how it can be used to improve fuzz testing of IVIs and telematics units.
- We show how additional information can be collected on the target system and used to determine whether there are exceptions and help developers identify the underlying cause of any issues detected.
- We built a test bench based on this approach and fuzz tested several SUTs. We highlight some examples of our findings that would not have been detected without agent instrumentation.

## Background and problem statement

Many automotive organizations have made fuzz testing a mandatory step in their software development process or are moving toward doing so. In other industries, such as network and telecommunications or enterprise, fuzz testing has already been integrated into the software development process and proven to be an effective approach to identifying bugs and vulnerabilities quickly. These target systems are typically easy to instrument by monitoring just the same protocol that is being fuzzed. This approach is common when fuzzing IT solutions such as a web server or a specific communication library.

In many cases, monitoring the same protocol that is being fuzzed is effective. For example, monitoring HTTP requests and corresponding HTTP responses could help identify potential unknown vulnerabilities in a web server. Furthermore, the famous Heartbleed vulnerability (CVE-2014-0160), which was found by fuzzing the OpenSSL library, was identified by monitoring the responses to the heartbeat request message.<sup>5</sup> By contrast, automotive systems are often more complicated and interconnected with other systems and therefore not easy to instrument. As a result, without proper instrumentation, many unknown vulnerabilities and potential issues cannot be identified on these automotive systems.

As shown in previous research,<sup>6</sup> without proper instrumentation of deeply embedded ECUs using HIL systems, several potential issues go undetected. Similarly, without proper instrumentation of IVIs and telematics units, numerous potential issues go undetected.









You













Yo.

## File format fuzzing

A popular function of infotainment systems is the ability to play rich media content. Simpler systems play only audio file formats, but more expensive systems with larger displays can also display video and image content. The file format parsers on these devices are vulnerable to exploits embedded in the inputs they receive.

Defensics has several file format fuzzers, which can generate fuzzed versions based on the full specifications of various popular file formats. It can simply write these to a disk or use other logic to have these sent to software inputs and determine a verdict.

To test audio and video playback functionality, we created logic to send fuzzed files to various infotainment systems, automatically play these test cases, and assess the verdict using Agents.


## Conclusion

In this paper, we introduced the Agent Instrumentation Framework and explained how it can be used to improve the fuzz testing of IVIs and telematics units. We explained how to better instrument these target systems to allow for more efficient and accurate fuzz testing. One or more Agents deployed on the SUT are used to collect additional information to determine whether test cases on the SUT caused an exception. This information is also provided to the fuzz testing tool and stored in the log file, helping developers identify the underlying root cause of the issues detected and fix problems more efficiently. To show the effectiveness of the proposed framework, we built a test bench based on this approach and performed fuzz testing of several SUTs. We presented our findings and highlighted several examples where issues on the SUTs would not have been detected without agent instrumentation.

The Agent Instrumentation Framework is suitable for IVIs and telematics systems, which are typically based on operating systems providing more functionality, such as Linux and Android, making it possible to run agents on the SUT. We believe that the growth in connected cars and autonomous driving, which continues to drive large volumes of software development in the automotive industry, coupled with an increasing awareness of cyber security in the automotive development process, will lead automated fuzz testing to become a mandatory step for these types of systems. Our proposed framework can help support automated fuzz testing for SUTs running richer operating systems such as Linux and Android.

## References

1. D. K. Oka, "Security in the Automotive Software Development Lifecycle," in *SCIS*, Niigata, Japan, 2018.
2. S. Bayer, T. Enderle, D. K. Oka, and M. Wolf, "Security Crash Test—Practical Security Evaluations of Automotive Onboard IT Components," in *Automotive—Safety & Security 2015*, Stuttgart, Germany, 2015.
3. D. K. Oka, A. Yvard, S. Bayer, and T. Kreuzinger, "Enabling Cyber Security Testing of Automotive ECUs by Adding Monitoring Capabilities," in *esca Europe*, Munich, Germany, 2016; D. K. Oka, T. Fujikura, and R. Kurachi, "Shift Left: Fuzzing Earlier in the Automotive," in *esca Europe*, Brussels, Belgium, 2018.
4. Automotive Grade Linux, [Automotive Grade Linux](#), accessed May 6, 2018; Automotive Grade Linux, [Automotive Grade Linux Hits the Road Globally with Toyota: Amazon Alexa Joins AGL to Support Voice Recognition](#), accessed May 7, 2018; GENIVI, [GENIVI](#), accessed May 6, 2018; Open Automotive Alliance, [Introducing the Open Automotive Alliance](#), accessed May 6, 2018.
5. Synopsys, [Heartbleed Bug](#), accessed May 7, 2020.
6. Oka, Yvard, et al., "Enabling Cyber Security Testing"; Oka, Fujikura, and Kurachi, "Shift Left."



Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations optimize security and quality in DevSecOps and throughout the software development life cycle.

For more information, go to [www.synopsys.com/software](https://www.synopsys.com/software).

**Synopsys, Inc.**

185 Berry Street, Suite 6500  
San Francisco, CA 94107 USA

**Contact us:**

U.S. Sales: 800.873.8193

International Sales: +1 415.321.5237

Email: [sig-info@synopsys.com](mailto:sig-info@synopsys.com)