Exploring Containerization for Protecting Unmanned Robotic Vehicles

Chidera Okoro

cjokoro@shockers.wichita.edu

Monowar Hasan

monowar.hasan@wichita.edu

Robotic Vehicles (RVs)

Applications

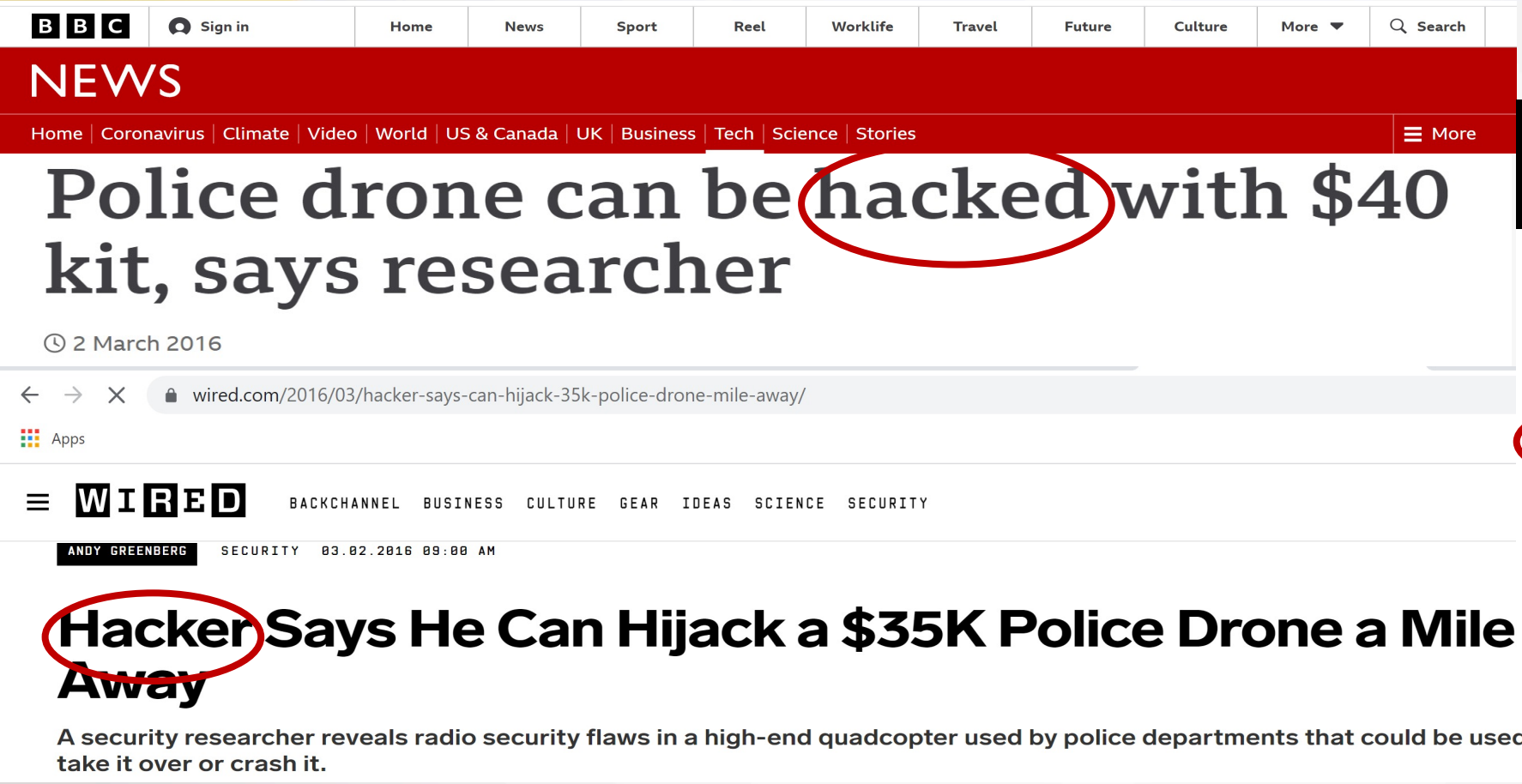
- Surveillance
- Communication
- Delivery
- Agriculture
- Manufacturing

RV Security?

- Traditional robotic vehicles:
  - Custom hardware
  - Closed-source software
  - Not interconnected
- Modern robotic vehicles:
  - Off-the-shelf (COTS) components
  - Open-source software
  - More interconnected → Internet

Larger attack surface!

RV Security?


BBC Sign in Home News Sport Reel Worklife Travel Future Culture More Search

NEWS

Home | Coronavirus | Climate | Video | World | US & Canada | UK | Business | Tech | Science | Stories | More

Police drone can be hacked with $40 kit, says researcher

2 March 2016

wired.com/2016/03/hacker-says-can-hijack-35k-police-drone-mile-away/

Apps

WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY

ANDY GREENBERG SECURITY 03.02.2016 09:00 AM

Hacker Says He Can Hijack a $35K Police Drone a Mile Away

A security researcher reveals radio security flaws in a high-end quadcopter used by police departments that could be used to take it over or crash it.


THE MANUFACTURER

Industrial robots are vulnerable to advanced hackers, report reveals

Posted on 7 Aug 2020 by James Devonshire

Vanilla Robotic Systems

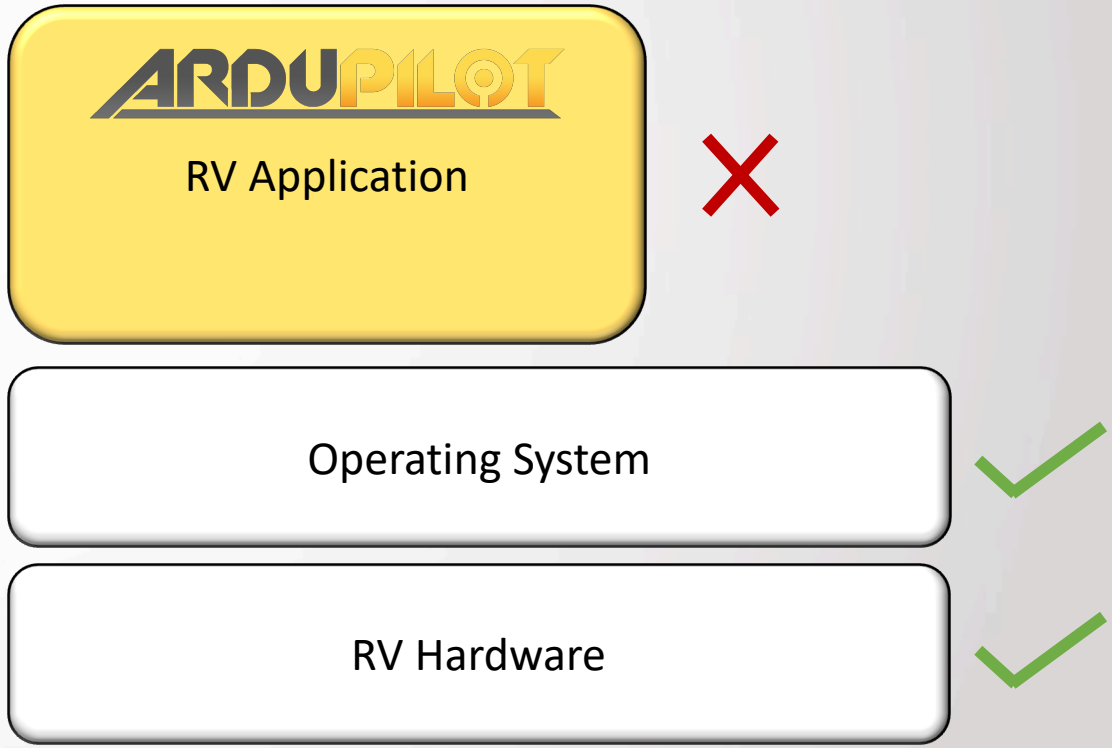
- RV malfunction
- No way to control what goes in and out of application
- System may crash
  - Fatalities among humans
  - Destruction to surrounding environment

Can we do better?

Adversary Model

Attack Model

- RV application can be compromised
  - No specific assumptions on how the attacker compromise
  - bugs, vulnerabilities, social engineering, remote update
- OS is trusted
- Hardware attacks are not considered



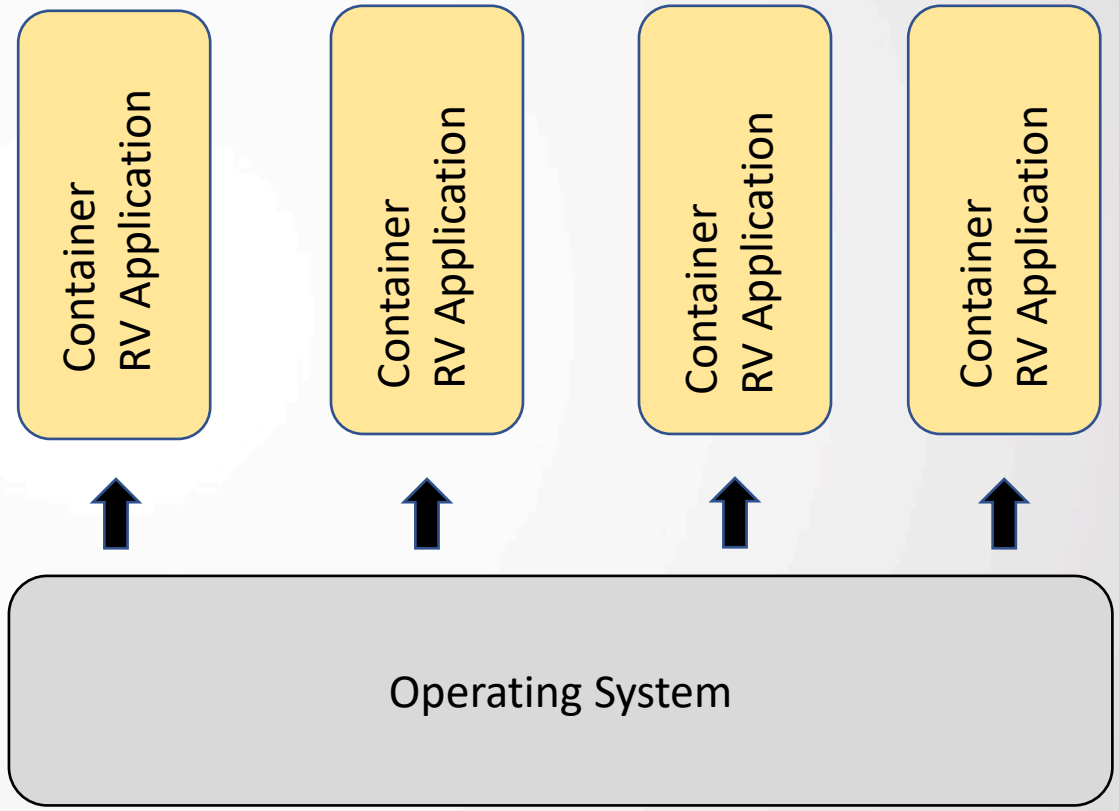
Our Approach

Sandboxing RV Applications

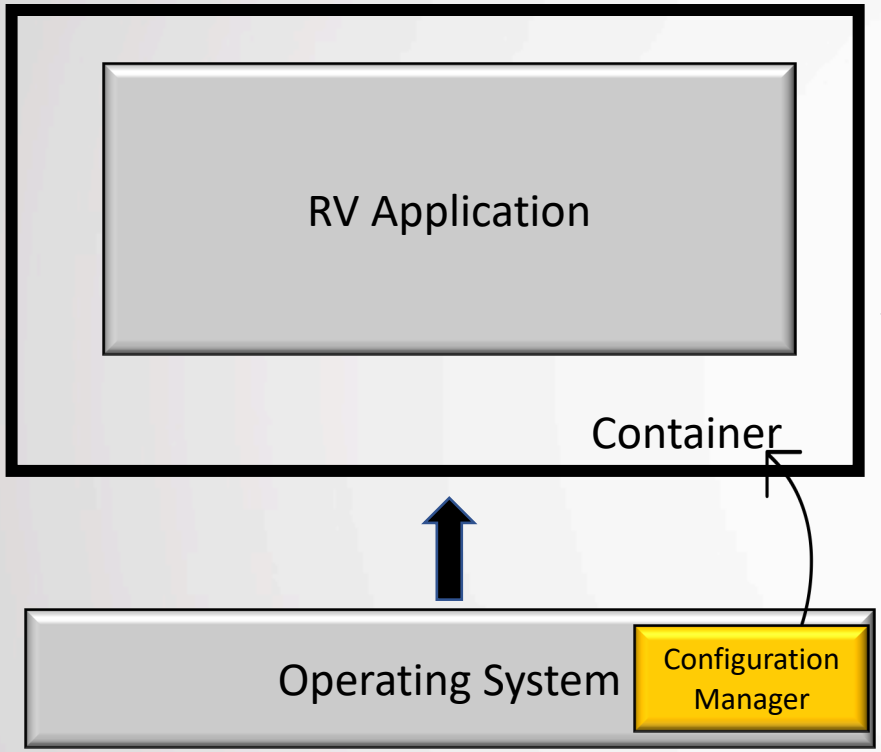
Containerization

- Isolates surrounding environment (sandboxing)
- Limits resources

- RV application runs inside the container
- Operating system communicates with containers



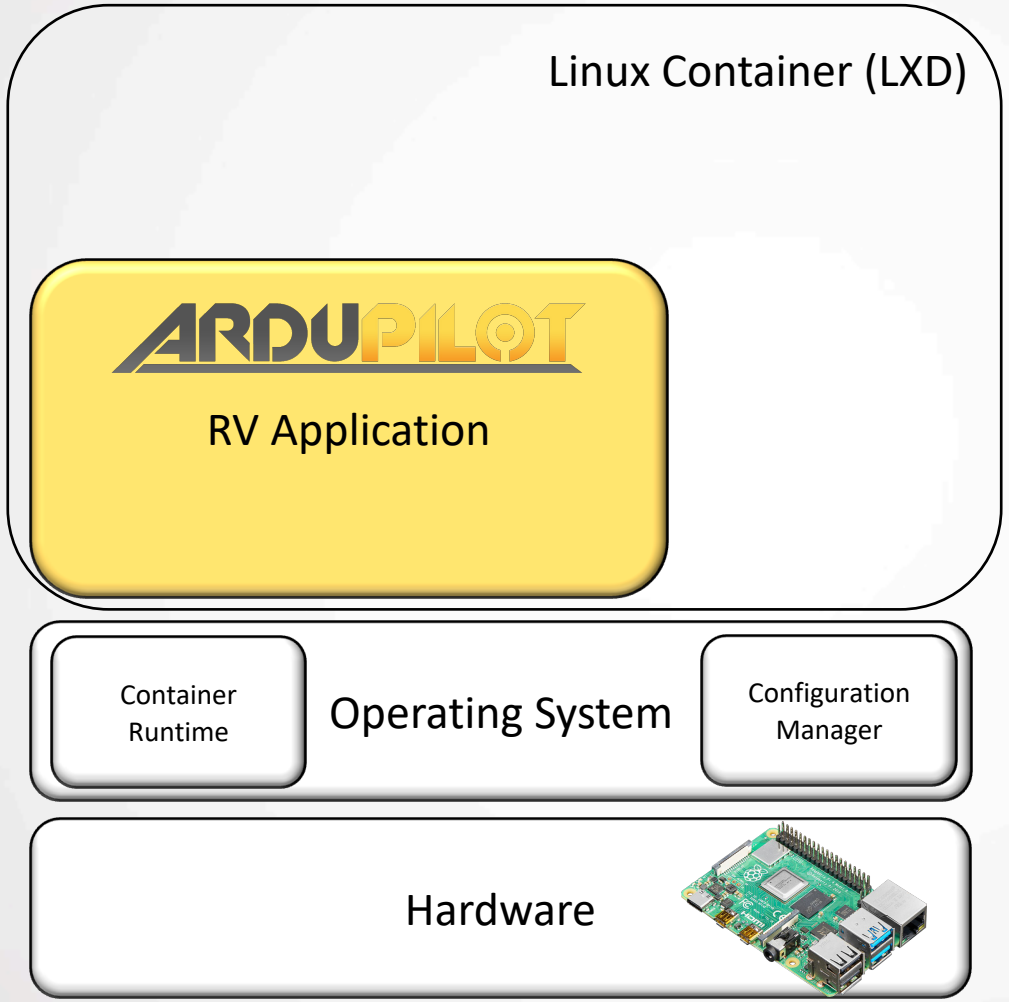
Containerization of RV Applications



malicious attack

- Monitors the behavior of the application
- Able to see what goes in and out of application
- We can enforce design time configurations

Containerization of RV Applications



Control groups
- responsible for managing resource usage

Namespaces
- provides isolation of system resources

Ethernet
- network interface that is bridged to OS network

Experiment Setup

- Hardware
  - Raspberry Pi 3 Model B
  - Navio2 extension board
- Software
  - Ardupilot Software Version 3
  - Linux Container – LXD 4.0

Implementation publicly available:
https://github.com/CPS2RL/rv_security_container.git

Experiments

Security Analysis and Overhead Analysis

Denial of Service

System call
- Implementing an attack that inserts a denied system call into a program

Memory Exhaustion
- Implementing an attack that overcomes the allowed threshold for memory storage

malicious attack

Performance Overhead

Performance in Execution Time
- Took over 10,000 data points to calculate the mean, standard deviation, 99th percentile, and ECDF

Performance in Memory Usage
- Took over 10,000 data points to calculate the mean, standard deviation, 99th percentile, and ECDF

Results

Security Analysis: System Call Access

Denial of Service

```
if(GPS.init(){
        brk();
        GPS.setSensor(1);
  }
  else{
      GPS.setSensor(0);
  }
```

```
brk(NULL)                                = -1 ENOSYS (Function not implemented)
mmap2(NULL, 1048576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = (
```

The brk() function is disabled from being accessed by the container.

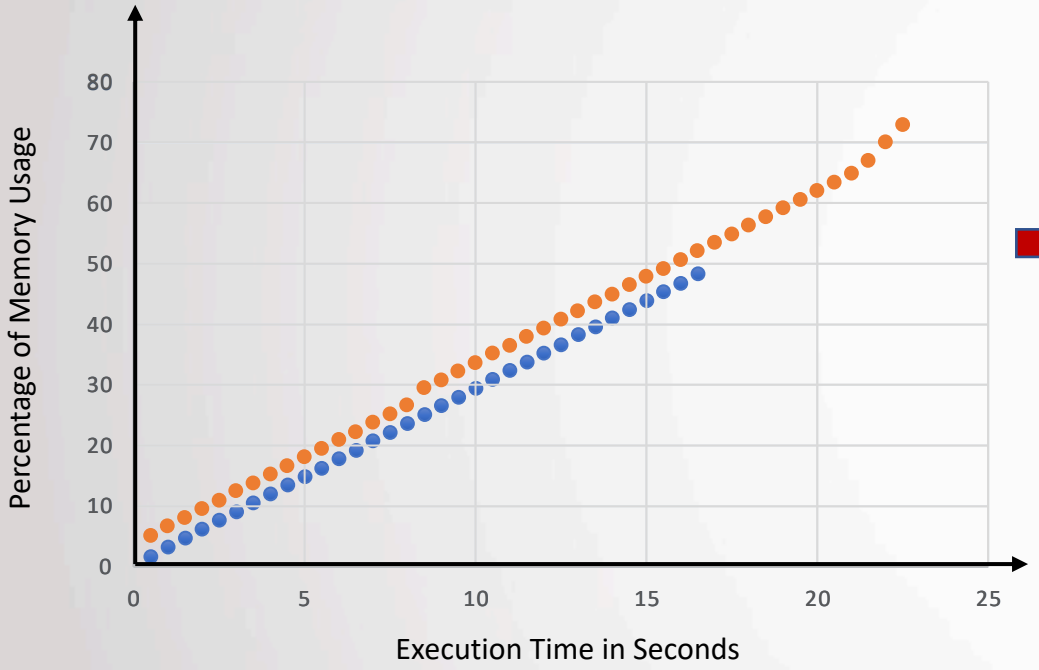
```
brk(NULL)                                = 0x12a7000
brk(0x12c8000)                           = 0x12c8000
```

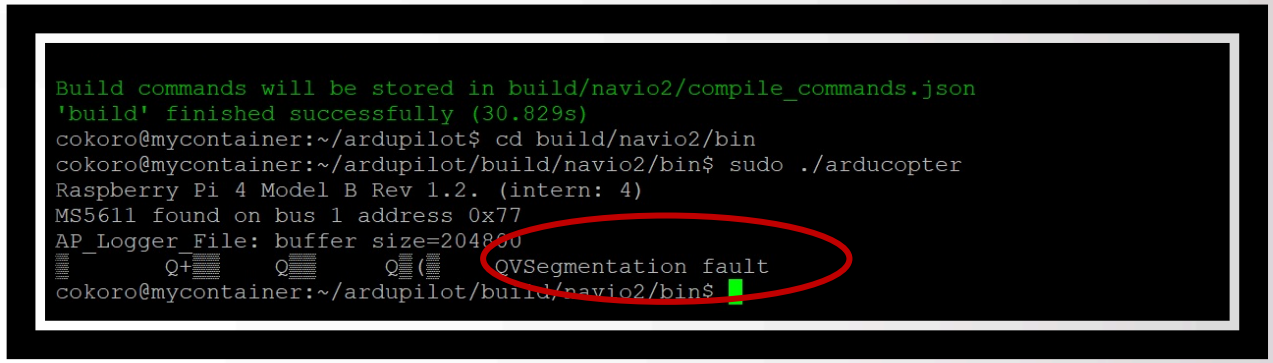
When an unauthorized system call is implemented into the program, it results in the change of the execution flow.

The brk() function is being called and used in our approach.

Security Analysis: Memory Exhaustion

Memory Exhaustion





Our Approach

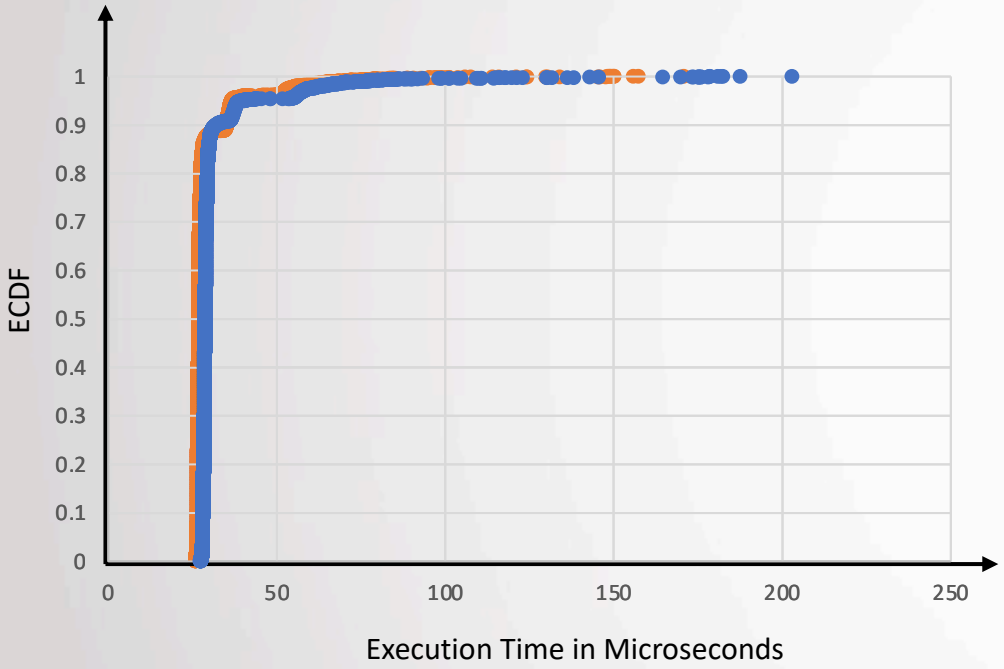
With the use of containerization, the configuration manager can detect the memory leak and prevent the software from a segmentation fault.

- Our Approach
- Vanilla

Overhead Analysis

Impact on Execution Time

Empirical Distribution Graph of Ardupilot Barometer Application



- Our Approach
- Vanilla

	Execution Time (microseconds)	
	Vanilla	Our Approach
Mean	**29.133**	**31.186**
STD. Deviation	**8.868**	**10.673**
99th Percentile	**68.206**	**76.023**

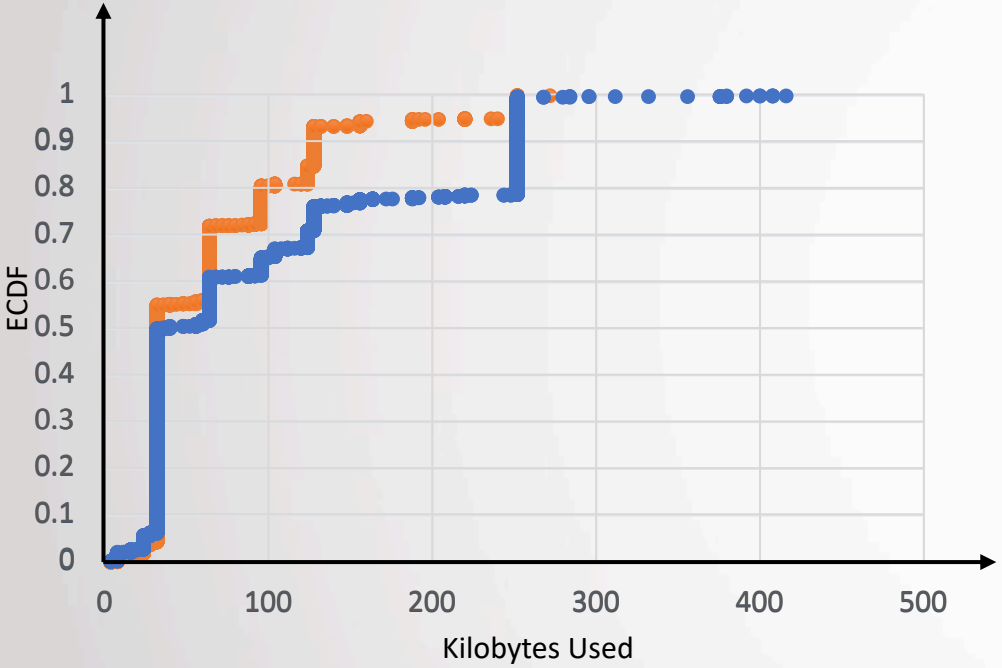
Summary:

The overhead of our approach was minimal with only a 1.07x increase in application execution time

Overhead Analysis

Impact on Memory Usage

Empirical Distribution Graph of Ardupilot Barometer Memory Usage



- Our Approach
- Vanilla

	Execution Time (kilobytes)	
	Vanilla	Our Approach
Mean	**67.302**	**98.522**
STD. Deviation	**55.906**	**90.538**
99th Percentile	**252**	**252**

Summary:

The overhead of our approach was minimal with only a 1.46x increase in application execution time

Conclusion

- Our research shows the feasibility of using containers for RV applications
- Containerization of RV Applications
  - Allows to monitor the behavior of the RV application without significant overheads
  - Can enforce design time configurations and prevent unauthorized access

- Our future research:
  - Will explore attacks on other signals (such as I/O, network, and timing anomalies).
  - Will also evaluate other autopilot software (e.g., Paparazzi and PX4).

Thank You!

Questions?