PGFUZZ: Policy-Guided Fuzzing for Robotic Vehicles

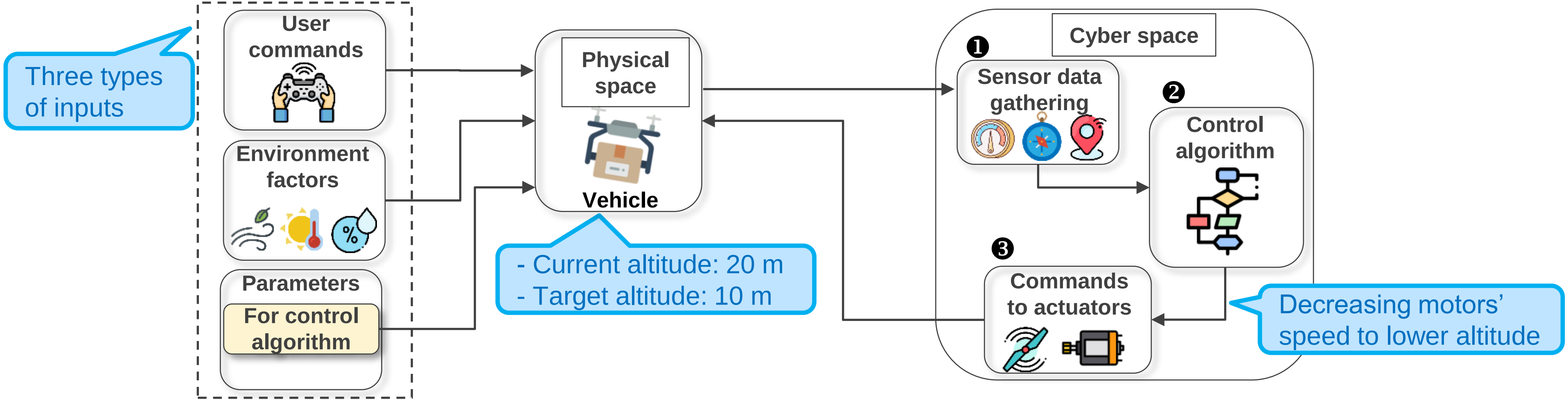
Hyungsub Kim, Muslum Ozgur Ozmen,
Antonio Bianchi, Z. Berkay Celik, and Dongyan Xu

Purdue University

NDSS 2021

Workflow of robotic vehicles (RV)

- Physical space
  - Attitude, altitude, speed, etc.
- Cyber space
  - Measuring the RV's current states
  - Adjusting actuators to reach target states



Fuzzers for robotic vehicles (RV)

- Rule:
  - *"Fail-safe mode must be triggered when the engine temperature is higher than 100 C° (212 F°)"*

```
// Developers forget to convert F° to C° scale
If (temperature >= 100) {
   Fail-Safe -> execute();
}
```

Fail-safe is triggered under 100 F° (37 C°).

Can traditional fuzzers (AFL, libFuzzer) discover such a design flaw? No

- Mutation: Code coverage
- Bug oracle: Memory access violation

Fuzzers for robotic vehicles (RV)

- Can fuzzers specialized for RVs discover the design flaw?
  - RVFUZZER, CPI, etc.

```
// Developers forget to convert F° to C° scale
If (temperature >= 100) {
   Fail-Safe -> execute();
}
```

Fail-safe is triggered under 100 F° (37 C°).

What about fuzzers for RVs? No

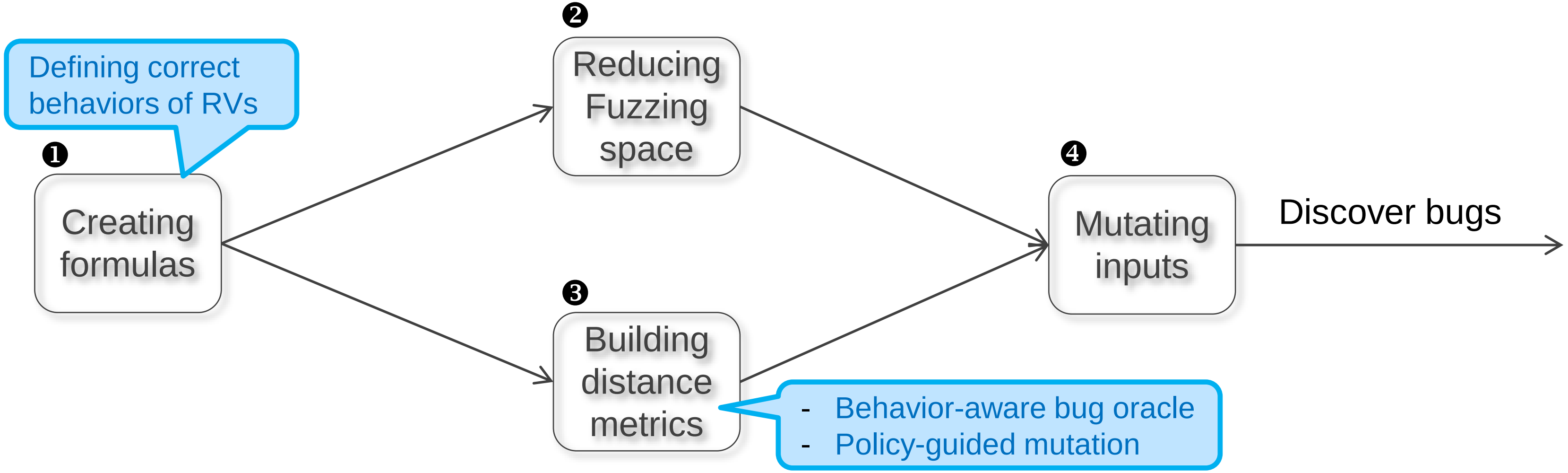
- Mutation & Bug oracle: unstable attitude

RVFUZZER: “Finding Input Validation Bugs in Robotic Vehicles through Control-Guided Testing,” in USENIX, 2019.
CPI: “Cyber-physical inconsistency vulnerability identification for safety checks in robotic vehicles,” in CCS 2020.

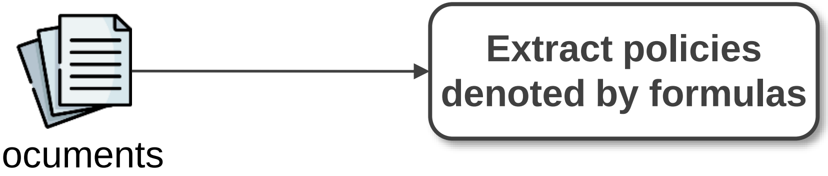
Overview of PGFUZZ

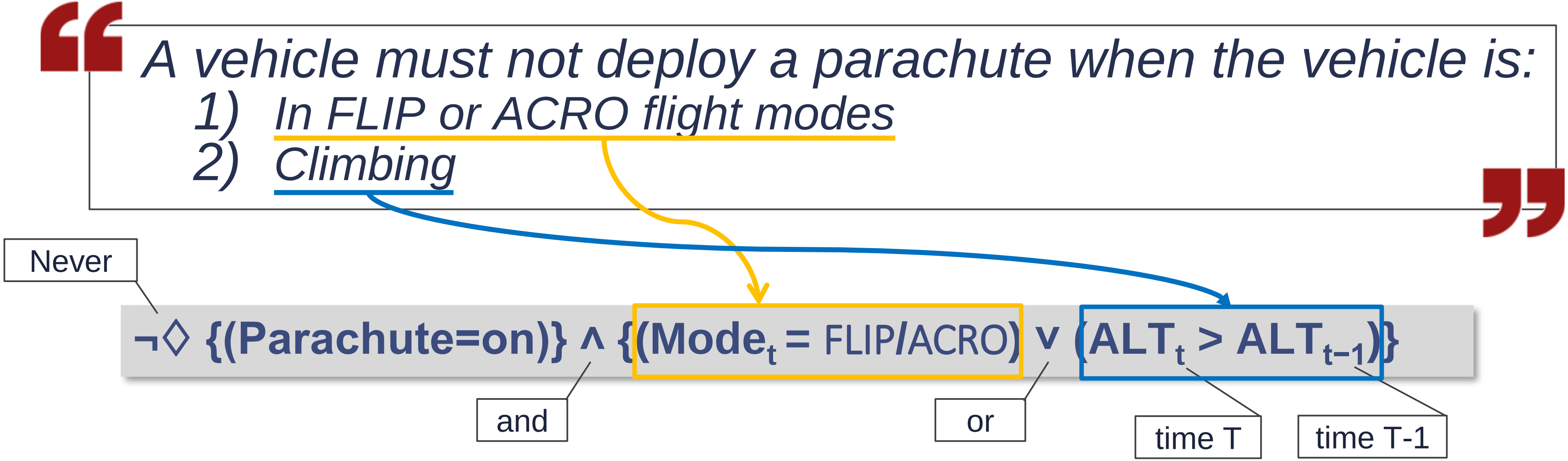
- Previous works do not
  - Know the RV’s correct behaviors
  - Consider entire input space

- PGFUZZ



Defining policies in formulas





> *A vehicle must not deploy a parachute when the vehicle is:*
> 1) *In FLIP or ACRO flight modes*
> 2) *Climbing*

$$\neg\Diamond \{(\mathbf{Parachute{=}on})\} \wedge \{(\mathbf{Mode_t} = \mathrm{FLIP/ACRO}) \vee (\mathbf{ALT_t > ALT_{t-1}})\}$$

The formula is created in the form of Metric temporal logic (MTL).

Finding inputs for mutation

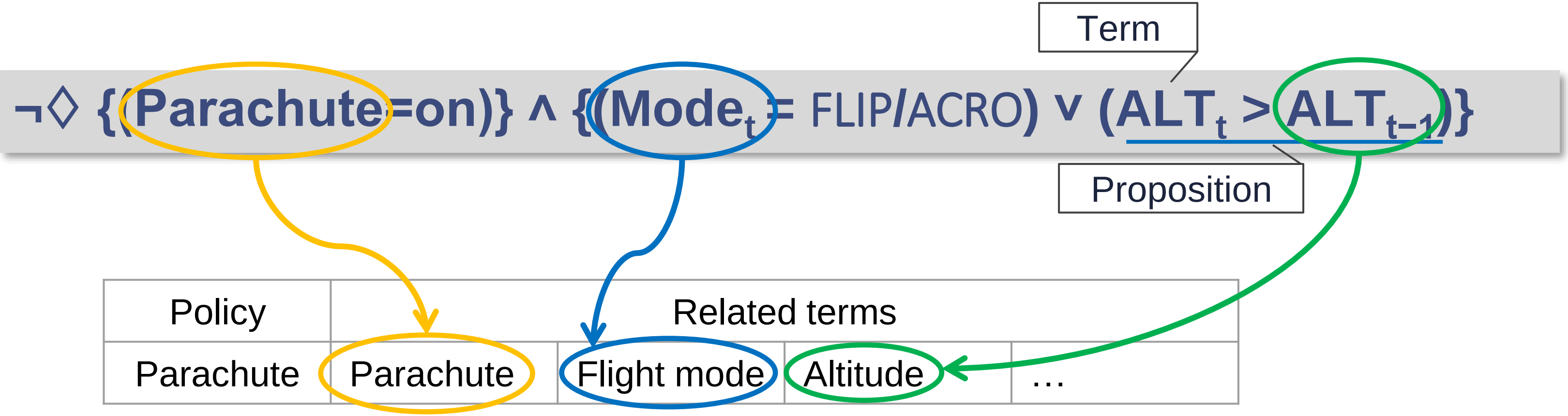
- Huge fuzzing space
  - 1,140 configuration parameters
  - 58 user commands
  - 168 environmental factors

- Only mutating inputs relevant to the policy

Finding inputs for mutation

- Policy consists of terms (physical states)
  - Only mutating inputs related to the terms

- Decompose the formula into terms (states)



Term

$$\neg\Diamond\ \{(\text{Parachute=on})\} \wedge \{(\text{Mode}_t = \text{FLIP/ACRO}) \vee (\text{ALT}_t > \text{ALT}_{t-1})\}$$

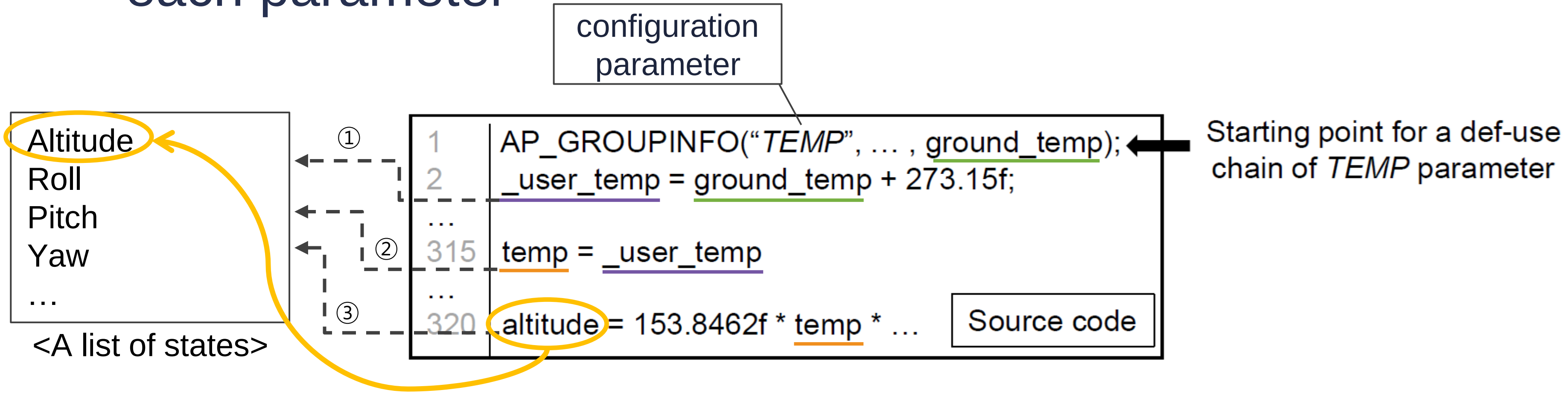
Proposition

Policy	Related terms			
Parachute	Parachute	Flight mode	Altitude	…

<Policy-term map>

Mapping parameters to each term

- Static analysis to identify which states are affected by each parameter

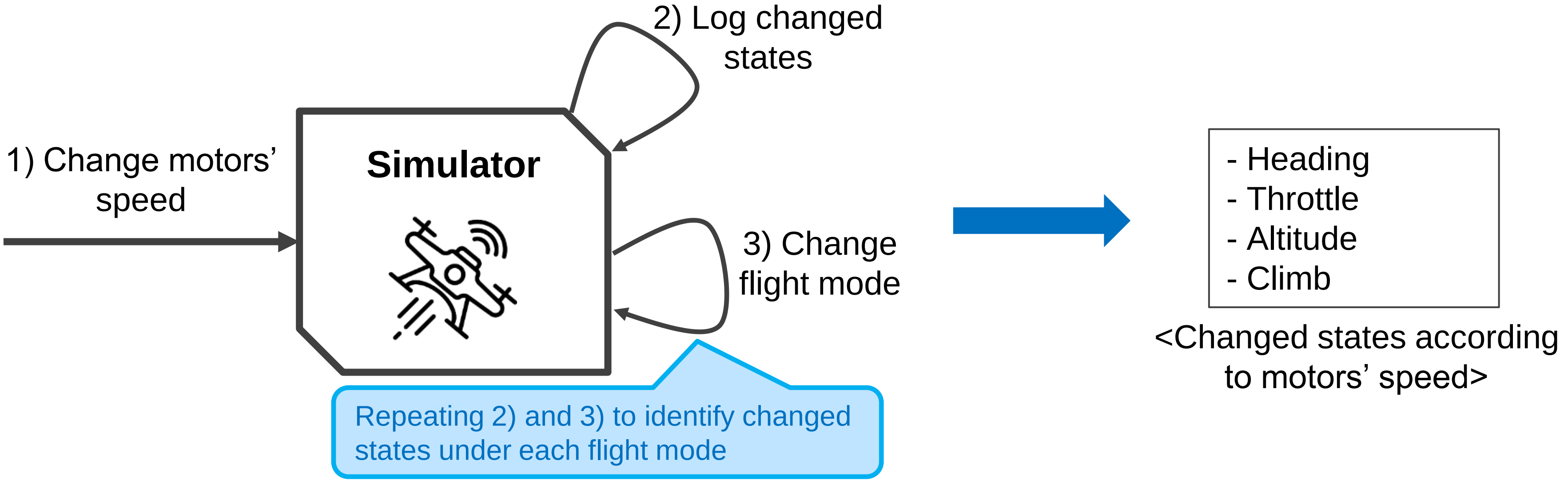


```
AP_GROUPINFO("TEMP", … , ground_temp);
_user_temp = ground_temp + 273.15f;
…
temp = _user_temp
…
altitude = 153.8462f * temp * …
```

Starting point for a def-use chain of *TEMP* parameter

Mapping other types of inputs to each term

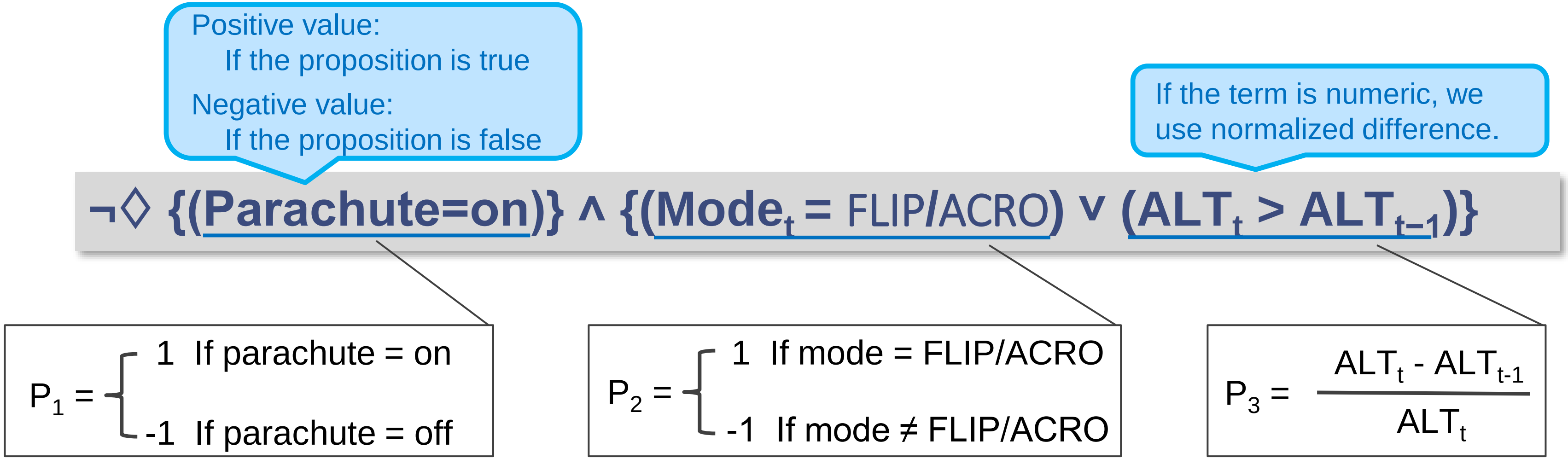
- How to map environmental factors and user commands to each term from source code? Use an RV simulator!



Repeating 2) and 3) to identify changed states under each flight mode

Two types of distances to mutate inputs

- Propositional distance
  - Goal: efficiently mutating inputs
  - Quantifies how close a proposition to the policy violation



Positive value:
If the proposition is true
Negative value:
If the proposition is false

If the term is numeric, we use normalized difference.

$$\neg\Diamond \{(\underline{\mathbf{Parachute=on}})\} \wedge \{(\underline{\mathbf{Mode}_t = \mathrm{FLIP/ACRO}}) \vee (\underline{\mathbf{ALT}_t > \mathbf{ALT}_{t-1}})\}$$

$$P_1 = \begin{cases} 1 & \text{If parachute = on} \\ -1 & \text{If parachute = off} \end{cases}$$

$$P_2 = \begin{cases} 1 & \text{If mode = FLIP/ACRO} \\ -1 & \text{If mode} \neq \text{FLIP/ACRO} \end{cases}$$

$$P_3 = \frac{ALT_t - ALT_{t-1}}{ALT_t}$$

Two types of distances to mutate inputs

- Global distance
  - Goal: detecting a policy violation

$$-1 \text{ X } [\text{Min}\{P_1, \text{Max}(P_2, P_3)\}] \begin{cases} \text{Positive value} & \text{if there is no policy violation} \\ \text{Negative value} & \text{if the RV violates the policy} \end{cases}$$

Working example (time T = 1)

$$P_1 = \begin{cases} 1 & \text{If parachute = on} \\ -1 & \text{If parachute = off} \end{cases}$$

$$P_3 = \frac{ALT_t - ALT_{t-1}}{ALT_t}$$

$$P_2 = \begin{cases} 1 & \text{If mode = FLIP/ACRO} \\ -1 & \text{If mode} \neq \text{FLIP/ACRO} \end{cases}$$

$-1 \text{ X } [\text{Min}\{P_1, \text{Max}(P_2, P_3)\}]$

Randomly select an input and assign a random value to the selected input

Time (T)	Parachute (on/off)	FLIP/ACRO mode (T/F)	Altitude (m)	P_1	P_2	P_3	Global distance	Next input for Time T+1
1	off	false	90	-1	-1	0	1	Motor speed = 1,800[1]
2								
3								
4								

🟩 : RV's current states at time T 🟨 : Calculated distances at time T

[1] (Motor speed > 1,500) → increasing RV's altitude
(Motor speed < 1,500) → decreasing RV's altitude

Working example (time T = 2)

$$P_1 = \begin{cases} 1 & \text{If parachute = on} \\ -1 & \text{If parachute = off} \end{cases}$$

$$P_3 = \frac{ALT_t - ALT_{t-1}}{ALT_t}$$

$$\boxed{-1 \text{ X } [\mathrm{Min}\{P_1, \mathrm{Max}(P_2, P_3)\}]}$$

$$P_2 = \begin{cases} 1 & \text{If mode = FLIP/ACRO} \\ -1 & \text{If mode} \neq \text{FLIP/ACRO} \end{cases}$$

1) We log (motor, 1,800) because the input increases P3.

2) PGFUZZ selects an input and assign a random value to the selected input

Time (T)	Parachute (on/off)	FLIP/ACRO mode (T/F)	Altitude (m)	P_1	P_2	P_3	Global distance	Next input for Time T+1
1	off	false	90	-1	-1	0	1	Motor speed = 1,800[1)]
2	off	false	100	-1	-1	0.1	1	Motor speed = 1,800[1)]
3								
4								

3) When the selected input increased a distance before, we reuse the input and value pair (motor, 1,800)

1) (Motor speed > 1,500) → increasing RV's altitude
(Motor speed < 1,500) → decreasing RV's altitude

Working example (time T = 3)

$$P_1 = \begin{cases} 1 & \text{If parachute = on} \\ -1 & \text{If parachute = off} \end{cases}$$

$$P_3 = \frac{ALT_t - ALT_{t-1}}{ALT_t}$$

$$P_2 = \begin{cases} 1 & \text{If mode = FLIP/ACRO} \\ -1 & \text{If mode} \neq \text{FLIP/ACRO} \end{cases}$$

$$\boxed{-1 \text{ X } [\mathrm{Min}\{P_1, \mathrm{Max}(P_2, P_3)\}]}$$

Time (T)	Parachute (on/off)	FLIP/ACRO mode (T/F)	Altitude (m)	P_1	P_2	P_3	Global distance	Next input for Time T+1
1	off	false	90	-1	-1	0	1	Motor speed = 1,800[1)]
2	off	false	100	-1	-1	0.1	1	Motor speed = 1,800
3	off	false	110	-1	-1	0.09	1	Parachute = on
4								

PGFUZZ selects an input

[1)] (Motor speed > 1,500) → increasing RV's altitude
(Motor speed < 1,500) → decreasing RV's altitude

Working example (time T = 4)

$$P_1 = \begin{cases} 1 & \text{If parachute = on} \\ -1 & \text{If parachute = off} \end{cases}$$

$$P_3 = \frac{ALT_t - ALT_{t-1}}{ALT_t}$$

$$P_2 = \begin{cases} 1 & \text{If mode = FLIP/ACRO} \\ -1 & \text{If mode} \neq \text{FLIP/ACRO} \end{cases}$$

$$\boxed{-1 \text{ X } [\text{Min}\{P_1, \text{Max}(P_2, P_3)\}]}$$

Time (T)	Parachute (on/off)	FLIP/ACRO mode (T/F)	Altitude (m)	P_1	P_2	P_3	Global distance	Next input for Time T+1
1	off	false	90	-1	-1	0	1	Motor speed = 1,800[1)]
2	off	false	100	-1	-1	0.1	1	Motor speed = 1,800
3	off	false	110	-1	-1	0.09	1	Parachute = on
4	on	false	112	1	-1	0.02	-0.02	Policy violation!

Vehicle must not increase its altitude

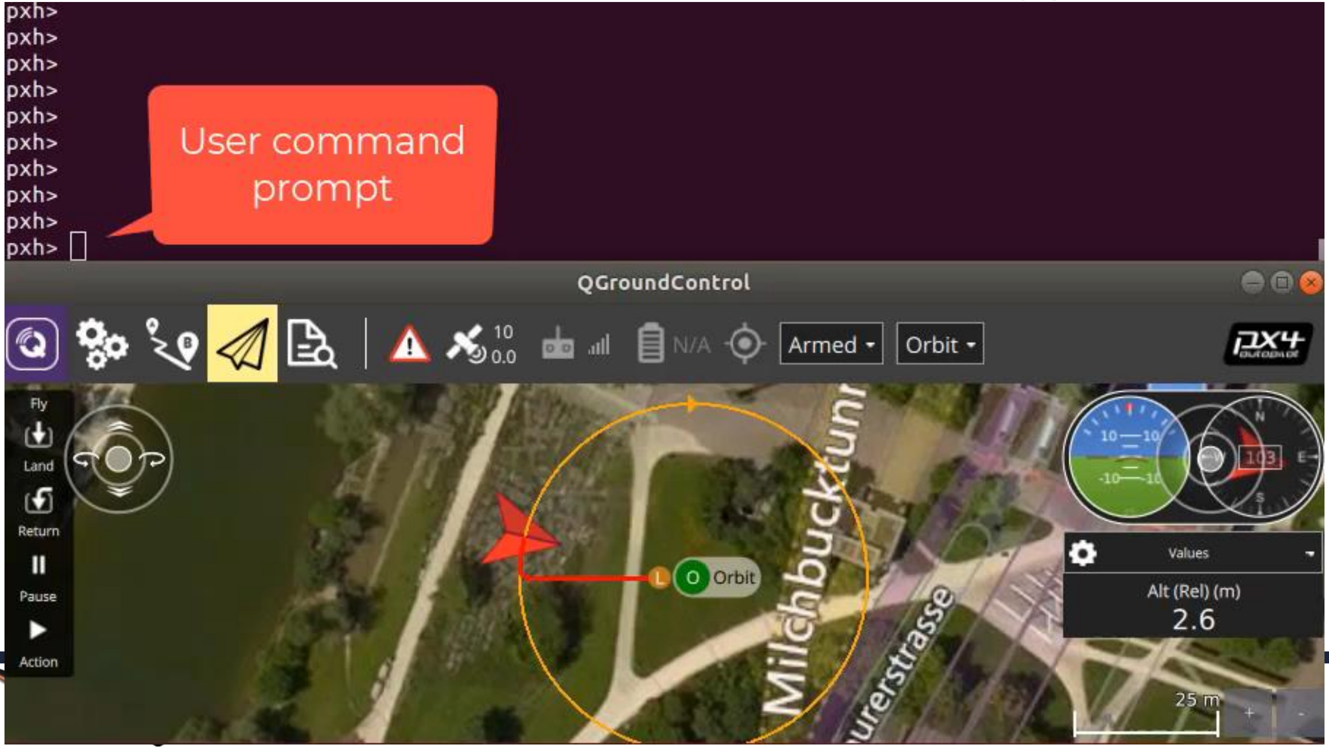
1) (Motor speed > 1,500) → increasing RV's altitude
(Motor speed < 1,500) → decreasing RV's altitude

Evaluation

- RV control software
  - ArduPilot, PX4, and Paparazzi
- 56 extracted policies
  - Fuzzing 48 hours per each control software
  - Violating 14 policies in the three-control software
- Found 156 bugs

Case study

- Policy
  - “If time exceeds COM_POS_FS_DELAY seconds after GPS loss is detected, the GPS fail-safe must be triggered”



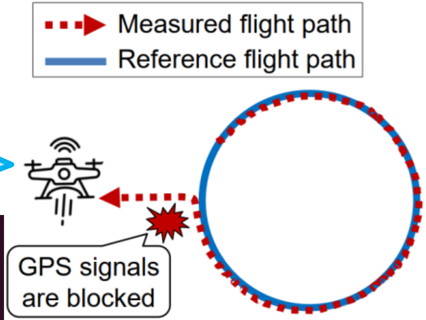
```
pxh>
pxh>
pxh>
pxh>
pxh>
pxh>
pxh>
pxh>
pxh>
pxh>
```

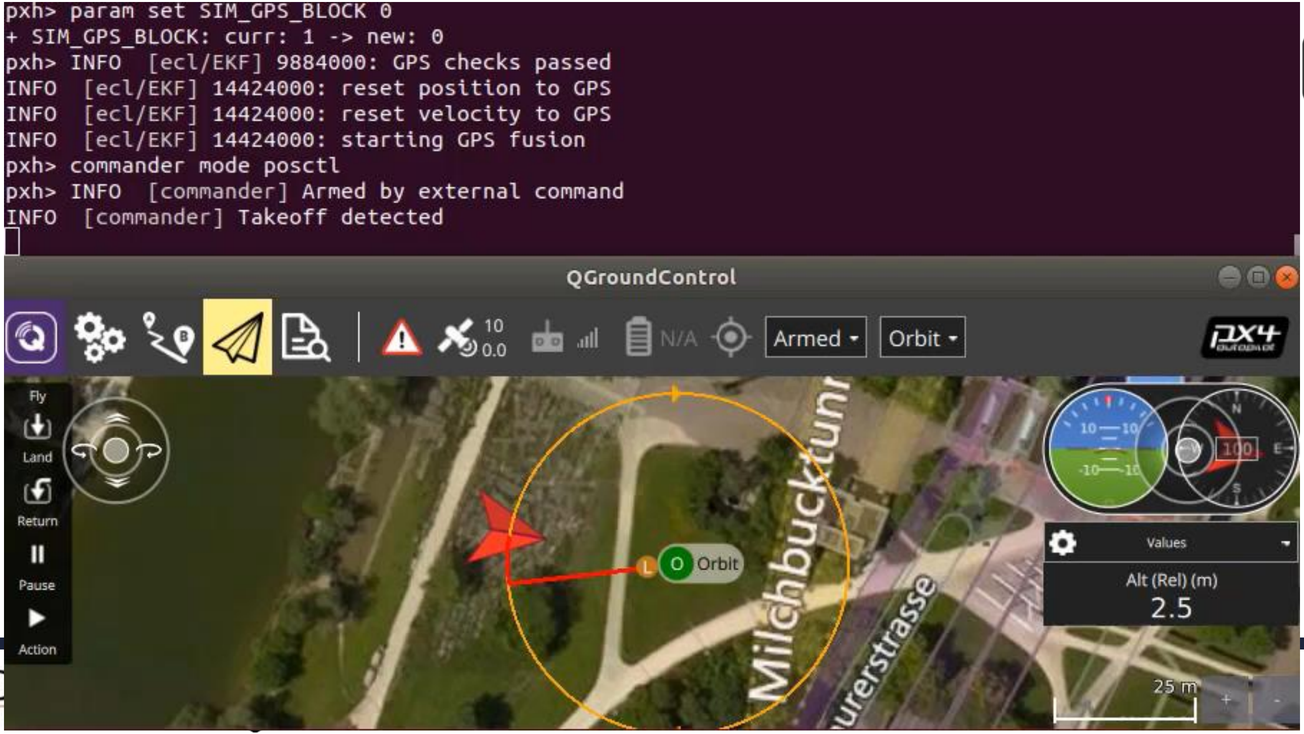
User command prompt

Case study

- Fail to trigger the GPS fail-safe under
  - COM_POS_FS_DELAY = -1

PX4 maintains the ORBIT flight mode under GPS signal loss.





```
pxh> param set SIM_GPS_BLOCK 0
+ SIM_GPS_BLOCK: curr: 1 -> new: 0
pxh> INFO  [ecl/EKF] 9884000: GPS checks passed
INFO  [ecl/EKF] 14424000: reset position to GPS
INFO  [ecl/EKF] 14424000: reset velocity to GPS
INFO  [ecl/EKF] 14424000: starting GPS fusion
pxh> commander mode posctl
pxh> INFO  [commander] Armed by external command
INFO  [commander] Takeoff detected
```

Alt (Rel) (m) 2.5

Conclusion

- Novel fuzzing approach to find logic bugs
  - Behavior-aware bug oracle
    - Leverage policies (MTL formulas)
  - Policy-guided mutation
    - Propositional and global distances
  - 156 previously unknown bugs
    - 128 out of 156 found bugs can only be discovered by PGFUZZ.
    - 106 bugs have been acknowledged
    - 9 bugs have been patched

Thank you! Questions?

kim2956@purdue.edu

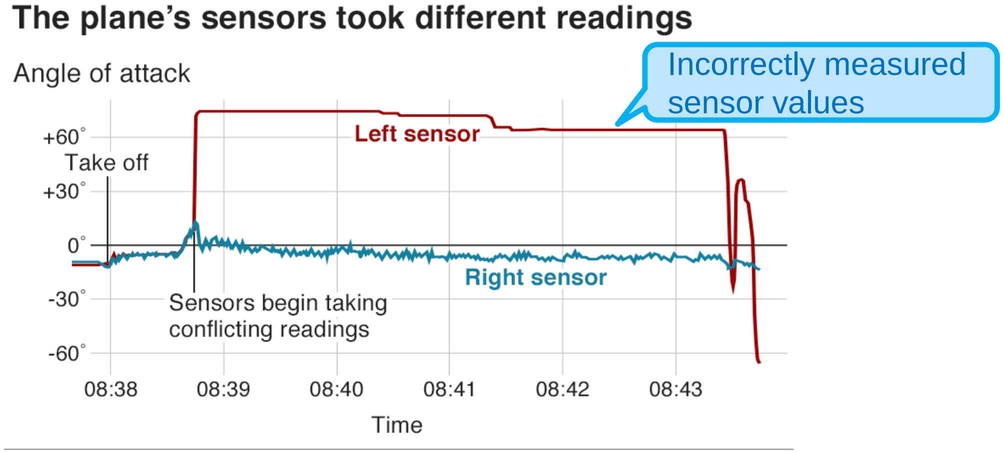
PURDUE UNIVERSITY CERIAS PurSecLab

Backup slides

Safety bug in real world

- Boeing-737 Max airplanes
  - Crashed due to a design flaw
  - Lowered its altitude based on only one broken sensor

How can we find such a critical bug in flight control software? Um… fuzzing?


The plane's sensors took different readings
Angle of attack
+60˚
+30˚
0˚
-30˚
-60˚
Take off
Left sensor
Right sensor
Sensors begin taking conflicting readings
08:38
08:39
08:40
08:41
08:42
08:43
Time
Incorrectly measured sensor values

Source: Ethiopian Aircraft Accident Investigation Bureau

https://www.dailymail.co.uk/news/article-7056177/US-investigators-believe-bird-strike-factor-Ethiopian-Airlines-Boeing-737-Max-8-crash.html

Threat model (1)

- Developers are benign
  - Incorrectly design or make buggy code

- Users are also benign
  - Unintentionally trigger the buggy code

Threat model (2)

- Attackers control three types of inputs
  - Further, they can wait until suitable conditions
- Attackers' goal
  - Stealthily triggering buggy code via sending inputs that looks innocent
- The followings are out of scope
  - Physical sensor attacks
  - Malicious code injections

Evaluation

- RV control software
  - ArduPilot, PX4, and Paparazzi
- 56 extracted policies
  - Fuzzing 48 hours per each control software
  - Violating 14 policies in the three-control software
- Found 156 bugs

Physical effect		
Unstable attitude	Software crash	Unexpected behavior
45	90	21
Total: 156		

For example, failing to trigger GPS fail-safe mode

Outline

- Defining RV’s correct behaviors as formulas
- Reducing fuzzing space
- Building distance metrics
- Evaluation