Demo: Policy-based Discovery and Patching of Logic Bugs in Robotic Vehicles

Hyungsub Kim, Muslum Ozgur Ozmen, Antonio Bianchi, Z. Berkay Celik, and Dongyan Xu
Purdue University
{kim2956, mozmen, antoniob, zcelik, dxu}@purdue.edu

Video 1: https://youtu.be/nhmKE03-bnk **Video 2**: https://youtu.be/gI2BPWIkYTs

This demo is based on PGFuzz [1], a policy-guided fuzzer that discovers logic bugs in a robotic vehicle (RV), and PGPatch [2], a policy-guided logic bug patching for the RV. Logic bugs cause deviations in the RVs' behavior from the developer's expectations but do not cause the program to stop execution. Discovering and patching logic bugs is challenging as there is no (i) clear definition of an RV's "correct behavior" and (ii) methodology to find and fix them.

Discovering Logic Bugs. We address these challenges with PGFuzz. It (1) takes, as input, a policy (expressed with temporal logic) that describes the RV's expected behaviors, (2) finds inputs related to the policy via static and dynamic analysis, and (3) mutates the identified inputs based on distance metrics measuring "how close" the RV's state is to a violation.

For instance, ArduPilot documentation states that the following two conditions must hold to deploy a parachute. $\mathtt{C_1}$: the vehicle must not be in the FLIP or ACRO flight modes, and $\mathtt{C_2}$: the barometer must show that the vehicle is not climbing. Based on these requirements, we express a policy in temporal logic: $\square\{(\mathtt{Parachute} = \mathtt{on})\} \rightarrow \{(\mathtt{Mode_t} \neq \mathtt{FLIP/ACRO}) \wedge (\mathtt{ALT_t} \leq \mathtt{ALT_{t-1}})\}$ where t and ALT represent time and altitude, $\square$ denotes "always", and $\wedge$ represents "and".

PGFuzz first discovers that *flight mode*, *throttle*, and *parachute* inputs are related to the parachute policy via a combination of static and dynamic analysis. PGFuzz then mutates the inputs based on propositional and global distances to violate the parachute policy. Particularly, when the global distance is a negative value, the RV violates the policy. As shown in Figure 1, the global distance becomes negative when the propositional distances are positive. Hence, PGFuzz aims to maximize the propositional distances $(\mathtt{P_1} - \mathtt{P_3})$. As a result, PGFuzz discovers that ArduPilot improperly checks the two requirements ($\mathtt{C_1}$ and $\mathtt{C_2}$). The policy violation causes the RV to deploy the parachute when its flight mode is ACRO/FLIP, making the RV crash on the ground (Figure 2a).

We ran PGFuzz for two days for each RV control program (ArduPilot, PX4, and Paparazzi) to evaluate them against 56 policies. PGFuzz discovered 156 previously unknown bugs.

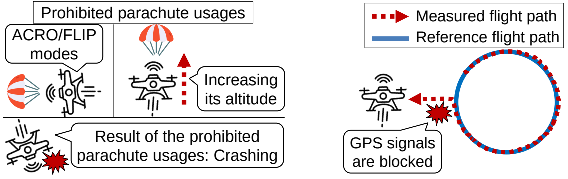
Fixing Logic bugs. The root cause of the parachute policy violation is that the RV does not check the preconditions specified in the policy. This bug can be fixed by inserting an "if statement" using the policy as a guide (Listing 1). PGPatch (1) takes, as input, an existing policy that was used to discover the logic bug through PGFuzz and bug-triggering inputs that PGFuzz finds, (2) maps the policy's terms to variables/functions in source code via static and dynamic analysis, (3) infers a patch location from the policy and the logic bug's symptom, (4) synthesizes a patch code from the policy, and (5) verifies the correctness of the patch on an RV simulator. We collected bugs from three popular RV software and found that PGPatch correctly fixes 258 out of 297 bugs (86.9%).

Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2022
24 April 2022, San Diego, CA, USA
ISBN 1-891562-75-4
https://dx.doi.org/10.14722/autosec.2022.23029
www.ndss-symposium.org

$$P_1 = \begin{cases} 1 & \text{if } \mathrm{Chute}_t = \text{on} \\ -1 & \text{if } \mathrm{Chute}_t \neq \text{on} \end{cases} \qquad P_2 = \begin{cases} 1 & \text{if } \mathrm{Mode}_t = \text{FLIP/ACRO} \\ -1 & \text{if } \mathrm{Mode}_t \neq \text{FLIP/ACRO} \end{cases}$$

$$P_3 = \frac{ALT_t - ALT_{t-1}}{ALT_t} \qquad G = -1 \times [\min\{P_1, \max(P_2, P_3)\}]$$

Fig. 1: The propositions ($\mathtt{P_1}$-$\mathtt{P_3}$) and the global distance ($\mathtt{G}$).



(a) Parachute operations. (b) GPS fail-safe.

Fig. 2: Illustration of two logic bugs discovered by PGFuzz.

```
1 void Copter::parachute_manual_release() {
2     if (control_mode != Mode::Number::ACRO
3       && control_mode != Mode::Number::FLIP)
4       parachute_release();
```

Listing 1: A patch (at lines 2 and 3) for the parachute bug.

Demonstration Plan. We provide videos to demonstrate (1) two logic bugs discovered by PGFuzz and (2) the effect of fixing those bugs through PGPatch. The first bug is the parachute policy example detailed before. The second shows that the RV fails to trigger a GPS fail-safe under changed environmental conditions, which causes the RV to float unpredictably (Figure 2b). Our project website includes (1) details of the discovered 156 logic bugs and (2) previews of the demo videos: **https://github.com/purseclab/PGFUZZ**.

Acknowledgment

This work was supported in part by ONR under Grants N00014-20-1-2128 and N00014-17-1-2045. Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of the ONR.

References

[1] H. Kim, M. O. Ozmen, A. Bianchi, Z. B. Celik, and D. Xu, "PGFUZZ: Policy-Guided Fuzzing for Robotic Vehicles," in *Network and Distributed System Security Symposium (NDSS)*, 2021.

[2] H. Kim, M. O. Ozmen, Z. B. Celik, A. Bianchi, and D. Xu, "PGPATCH: Policy-Guided Logic Bug Patching for Robotic Vehicles," in *IEEE Symposium on Security and Privacy (S&P)*, 2022.