# PGPATCH: Policy-Guided Logic Bug Patching for Robotic Vehicles

摘要：自动程序修复（APR）方法旨在识别给定错误的补丁，并在最小的人为干预下应用它们。迄今为止，现有的APR方法侧重于修复软件缺陷，如内存安全缺陷。然而，我们对流行的机器人车辆（RV）控制软件的分析表明，它们的大多数错误不是内存错误，而是逻辑错误。这些错误虽然不会导致软件崩溃，但会导致RV达到不期望的物理状态（例如，撞击地面）。（APR用于自动修复程序漏洞，传统的APR针对内存错误，本文针对RV特定的逻辑错误进行ARP）

为了修复这些逻辑错误，我们引入了PGPATCH，这是一个用于RV控制程序的策略指导程序修复框架，它可以为给定的逻辑错误识别正确的补丁，并在没有人为干预的情况下应用它。PGPATCH将用于发现逻辑错误的现有或新逻辑公式作为输入。然后，它利用公式，使用专用的动态分析将之前已知的逻辑错误分类为补丁类型。接下来，它使用定制的算法，基于识别的补丁类型和违反的公式，生成源代码补丁作为输出。最后，它创建了可重复的测试来验证补丁的完整性，确保补丁正确且不会降低RV的性能。我们对来自三个流行RV控制软件的选定错误案例进行了PGPATCH评估，发现它正确修复了297个逻辑错误中的258个（86.9%）。我们还招募了18名经验丰富的RV开发人员和用户，并进行了一项用户研究，演示了使用PGPATCH如何使修复RV软件中的错误更快、更不容易出错。

## I. INTRODUCTION

及时修补安全漏洞对于防止攻击者破坏软件至关重要。然而，最近的工作表明，由于需要手动操作，按时创建补丁是一项挑战。由于这个原因，人们对自动程序修复（APR）越来越感兴趣，它可以在没有人为干预的情况下创建补丁。

有两种广泛的APR方法。**基于测试套件的方法**为导致输出偏离开发人员预期的错误生成补丁。他们的补丁定位和生成算法要求用户输入通过和失败的测试用例，这些用例定义了与特定输入（即测试套件）相对应的程序预期输出。**基于规范的APR方法**通过以自然语言文档或形式逻辑公式的形式定义功能需求的规范来修复错误。

在本文中，我们考虑了机器人车辆（RV）控制软件中的**逻辑错误（logic bugs）**，这些错误会导致RV的物理行为与开发人员的预期发生偏差，但不会导致程序停止执行。原因是我们对两个流行的RV控制程序（ArduPilot和PX4）的1257个现有缺陷的分析（从2014年到2021），表明其中98.2%是逻辑错误，显示了RV软件中逻辑错误的普遍性。RV中的逻辑错误主要源于错误实现和设计缺陷。例如，PX4的文件规定“检测到GPS丢失时必须触发故障安全模式”，然而在以下三种情况下，PX4无法触发故障保护模式：（1）COM\_POS\_FS\_DELAY配置参数为负值；（2）RV处于CIRCLE飞行模式；（3）RV经过GPS信号不可用的区域。此逻辑错误导致不安全状态，导致RV无法预测地浮动，可能会撞上障碍物。

及时修复RV中的错误至关重要，因为攻击者可以偷偷利用这些逻辑错误造成不期望的行为和物理损坏。然而，修补RV中的逻辑错误比修复传统软件中的错误更具挑战性，主要有两个原因。首先，RV的“正确行为”不仅取决于“网络空间”（即，RV如何根据给控制软件的输入进行行为），还取决于“物理空间”（例如，它们运行的物理环境）。例如，考虑用户提供RV输入以使RV向前移动。从传统软件的角度来看，RV的正确行为是向前移动，因为软件必须根据程序输入显示一致的行为。**然而，根据当前的物理环境，RV可能会表现出不同的正确行为**：（i）如果RV靠近障碍物，则向后移动；（ii）当RV失去GPS信号时，则降落在地面上；以及（iii）如果刮起强风，则保持在稳定的位置（RV会根据物理环境作出应对，不一定完全遵守命令）。其次，考虑到RV在任**何给定时间点接收和处理的数据量**，RV软件的“输入空间”和“输出空间”比传统软件大得多。该数据包括来自多个传感器的定期测量以及地面控制站（GCS）发出的命令。

不幸的是，现有的APR方法无法解决这些挑战。事实上，基于测试套件的方法无法修复RV控制程序中的逻辑错误，因为补丁的正确性取决于测试套件的完整性。如先前的工作所示，由于RV软件的输入和输出空间大，实现完整性具有挑战性。此外，基于规范的方法无法修复逻辑错误，因为它们主要关注内存错误，或者它们不会根据定义软件正确行为的规范创建代码级补丁。

为了解决这些限制，在本文中，我们介绍PGPATCH，一种用于RV控制程序的策略指导程序修复框架，它为给定的逻辑错误生成补丁，并在无需人工干预的情况下应用它们。PGPATCH由四个相互连接的组件组成：（1）预处理器；（2）补丁类型分析器；（3）补丁生成器；（4）补丁验证程序。首先，预处理器接收触发错误的输入和用PGPATCH策略语言（PPL）编写的公式，该语言定义RV的预期操作。此组件检查用户提供的公式的有效性。我们注意到，PGPATCH只需要一个触发错误的测试用例（即一个失败的测试用例），而基于测试套件的APR方法需要一整套测试用例，包括失败和通过的用例。其次，为了处理RV中不同类型的逻辑错误，补丁类型分析器会找到最合适的补丁类型来修复特定的逻辑错误。然后，补丁程序生成器组件为目标bug找到正确的修补程序位置并创建修补程序。PGPATCH首先通过PPL公式和模式匹配来识别补丁位置。再然后，它根据公式、识别的补丁类型和补丁位置创建补丁。最后，补丁验证程序将创建的补丁插入到目标源代码中，并编译补丁代码。然后测试补丁是否修复了错误，并确认补丁不会破坏RV的功能或降低其性能。

我们在ArduPilot、PX4和Paparazzi上评估了PGPATCH，这三种最常用的飞行控制软件用于许多商用RV。PGPATCH正确修复了297个逻辑错误中的258个（86.9%），平均需要12.5分钟才能创建一个补丁。此外，我们进行了一项用户研究，将RV软件开发人员和用户构建PPL公式所需的努力与创建手动代码级补丁的努力进行比较。我们的用户研究表明，与手动修补逻辑错误相比，构建PPL公式更容易，更不容易出错。

我们做出以下贡献：

* Behavior-aware Patch Generation. 我们介绍了PGPATCH，这是一个用于RV控制程序的政策指导的APR框架，它通过静态和动态分析的组合，利用现有或新的逻辑公式进行补丁定位和生成。PGPATCH还创建了可重复的测试来验证补丁的正确性。
* Evaluation with Real-world RV Software. 我们将PGPATCH应用于三个最流行的RV控制软件包。PGPATCH为之前已知的297个错误案例中的258个（86.9%）生成了正确的补丁。
* User Study. 我们招募了经验丰富的RV开发人员和用户，并进行了一项用户研究，该研究证明了PGPATCH与手动修补相比在修补漏洞方面的有效性。

## II. PRELIMINARIES

### Logic Bugs and Adversarial Exploitation in RVs

RV中的逻辑漏洞和对抗性利用。在本文中，我们使用术语“逻辑错误”来指代那些导致程序错误操作从而导致不希望的物理行为，而不导致程序崩溃或内存损坏的错误。从这个定义开始，我们将缓冲区溢出、空指针解引用和除零视为非逻辑错误。逻辑错误是由开发人员不正确地设计或实现软件组件引起的。开发人员的错误可能出现在各种各样的原因中，包括但不限于意外的环境条件(例如，强风和显著的传感器噪声)和复制粘贴有bug的代码片段

我们假设对手可以利用一个逻辑错误来暗中破坏RV的正常行为。特别是，RV的三种类型的输入(配置参数、用户命令和环境因素)可能会引发逻辑错误。攻击者可以(1)覆盖配置参数值，(2)重放或欺骗用户命令，(3)改变环境条件或等待合适的条件后进行攻击。对手可以利用RV通信协议和传感器中的已知漏洞进行输入操作

### PGPATCH Policy Language (PPL)

最近一项关于逻辑错误查找的工作(PGFuzz)创建了用于发现逻辑错误的线性时态逻辑(LTL)公式。特别是，用户通过官方文件识别rv的正确行为，并使用LTL模板正式表示它们。然而，定义LTL公式需要用户了解时态逻辑的语法和规则。

为了降低LTL公式的难度，我们在BNF表示法中引入了PPL，如Listing 1所示。在PPL公式中，“项”可以由RV的物理状态(S)、配置参数(P)、环境因子(E)、变量名(V)和函数名(F)组成。

