支持物理交互的无人机飞控系统安全测试方法

习 宁[1],周晓琳[1*],孙 聪[1],李乔杨[2],马建峰[1],郭鑫玉[1]
(1. 西安电子科技大学网络与信息安全学院,陕西西安 710071;2. 中国航空工业集团西安航空计算技术研究所,陕西西安 710065)

摘 要: 作为信息物理系统(Cyber-Physical Systems,CPS)的典型设备之一,无人机使用方便、对作业环境要求低、灵活性强,已广泛应用于农业、工业、军事等领域.其中,飞行控制系统是无人机核心基础服务,保障无人机遥测感知、通信覆盖、测绘救灾等应用的有效执行.但多变的物理环境、复杂的功能结构使无人机飞行控制系统在开发过程中容易引入各类软件安全问题,导致无人机发生劫持、坠毁、失控等严重问题.如何检测无人机飞控软件系统的安全问题变得非常重要.现有的大多数无人机异常检测技术依靠数字世界构造输入,难以及时发现无人机逻辑安全的问题,本文提出一种支持物理交互的无人机飞控软件安全检测方法,将静态与动态分析方法相结合,用模糊测试方法对无人机飞行控制软件的安全性进行测试,结果表明该方法能够以97%的高覆盖率对无人机飞控任务进行安全检测,并根据测试结果进行无人机特征数据提取,基于该特征数据采用机器学习的方法训练出双重异常检测模型,在多组数据集上与现有检测方法进行对比,本文方法达到发现无人机异常状况97.5%的准确率,有效检测出无人机飞控软件系统中的已知安全问题.

关键词: 无人机飞控软件安全;模糊测试;异常检测;安全检测;机器学习
基金项目: 国家自然科学基金(No.92267204,No.62232013,No.U24A20243)
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112(2025)03-0765-17
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240890

Security Testing Method for Unmanned Aerial Vehicle Flight Control System Supporting Physical Interaction

XI Ning[1], ZHOU Xiao-lin[1*], SUN Cong[1], LI Qiao-yang[2], MA Jian-feng[1], GUO Xin-yu[1]
(1. *School of Cyber Engineering, Xidian University, Xi'an, Shaanxi* 710071, *China*;
2. *AVIC Xi'an Aeronautics Computing Technique Research Institute, Xi'an, Shaanxi* 710065, *China*)

Abstract: As one of the typical equipment of cyber-physical systems (CPS), UAVs are easy to use, have low requirements for the working environment and strong flexibility, and have been widely used in agriculture, industry, military and other fields. Among them, the flight control system is the core basic service of UAV, which ensures the effective implementation of UAV telemetry perception, communication coverage, surveying, mapping and disaster relief applications. However, the changeable physical environment and complex functional structure make it easy to introduce various software security problems in the development process of the UAV flight control system, resulting in serious problems such as hijacking, crashing, and loss of control of the UAV. How to detect the security of the UAV flight control software system has become very important. Most of the existing UAV anomaly detection technologies rely on the input of digital world construction, and it is difficult to find the problem of UAV logic security in time, so this paper proposes a security detection method for UAV flight control software that supports physical interaction, combines static and dynamic analysis methods, and combines fuzzing testing methods to test the security of UAV flight control software, the results show that the method can detect the safety of UAV flight control tasks with a high coverage rate of 97%, and extract UAV feature data according to the test resultsBased on the feature data, the machine learning method is used to train a double anomaly detection model, and by comparing with the existing detection methods on multiple datasets, the proposed method finds the abnormal condition of the UAV with an accuracy rate of 97.5%, and effectively detects the known safety problems in the UAV flight control software system.

收稿日期:2024-10-07;修回日期:2024-12-14;责任编辑:覃怀银
*通讯作者:周晓琳

Key words: unmanned aerial vehicle flight control; fuzzing; software security; anomaly detection; security testing
Foundation Item(s): National Natural Science Foundation of China (No.92267204, No.62232013, No.U24A20243)

1 引言

作为信息物理系统(Cyber-Physical Systems,CPS)的典型设备之一,无人机使用方便、对作业环境要求低、灵活性强,广泛应用于农业、工业、军事等领域. 其中,飞行控制系统是无人机核心基础服务,保障无人机遥测感知、通信覆盖、测绘救灾[1]等应用的有效执行. 但是,多变的物理环境、复杂的功能结构使无人机飞行控制系统在开发过程中容易引入各类软件漏洞,包括内存泄漏[2]、浮点数异常、软件功能逻辑漏洞[3]等,导致无人机发生劫持、坠毁、失控等安全问题. 因此,如何及时发现无人机飞控代码中的安全漏洞是保障无人机飞行安全的核心与关键.

在软件漏洞安全检测领域中,内存泄漏、浮点数溢出作为典型的软件漏洞,易被攻击者利用,发动各种安全攻击. 无人机软件逻辑漏洞是指软件系统对外部环境的激励做出错误响应,与无人机的物理环境、用户输入、运动学模型紧密相关,例如无人机在降落模式下,已降落后未自动关闭电机,此时油门过高会导致无人机侧翻. 目前无人机软件漏洞检测主要采用静态测试和动态测试,模型检测和信息流分析是静态测试中最主要的2种方法,模糊测试是动态测试中最常用的方法.

模型检测[4,5]是一种自动化程度较高的形式化验证方法,给定已知系统的有限状态模型和逻辑属性,检查该系统是否满足或违反逻辑属性. 传统模型的检测方法主要应用于检测软件漏洞,并不能直接应用于检测无人机飞控软件系统,基于模型检测思想的现场模型检测方法成功检查了许多实际系统,例如文件系统. 近些年,该技术也应用分布式计算,在Guo教授团队[6]的研究中为框架提供支持并行分布式模型检查的分布式引擎. Tanakorn团队[7]的研究引入了一种新的方法来提高分布式系统的可靠性. 他们通过使用语义感知检测的模型来获取目标系统的简单语义信息,并将这些信息纳入系统状态空间的缩减策略中. MQTTactic[8]利用静态代码分析、形式化建模和自动化模型检查来识别安全问题,定义了MQTT代理的状态机模型,从源代码提取信息,并转换成模型,最后用模型检查器验证安全性. 基于对分布式系统的研究,有研究人员将其应用于无人机系统,提出了更适合无人机系统的现场模型检测方法. Shaikh等人[9]提出一种概率模型检测方法,通过构建模型来分析无人机面临的安全威胁,并用模型检测技术识别系统中的安全漏洞和风险,但这种方法主要分析已知异常. Taylor团队[10]提出的Avis现场模型检测器旨在检测无人机软件系统对传感器故障的逻辑错误响应. 这一检测器通过在无人机模式转换期间注入传感器故障发现潜在的不安全条件. 然而,尽管Avis检测器在识别无人机系统中对传感器故障处理不当的逻辑错误方面表现良好,但除此之外的模型检测器并未考虑到无人机系统的复杂性. 在实际使用中,可能存在适配性不足以及无法用形式化方式表达无人机物理逻辑正确性的问题. 此外,Avis检测无人机漏洞仅关注传感器漏洞,无法检测其他安全问题,并且是一种完全的黑盒检测方法,无法提供具体漏洞可能出现的代码位置信息.

信息流分析[11]的基本思想是审查程序中变量之间的信息传递是否符合安全标签的设定. Ferraiuolo等人[12]基于程序依赖图和非干扰性原则提出一种信息流静态分析算法,即程序中的低安全级别操作不能被高安全级别数据影响. 在安全的信息流中,对于程序中的任何语句s,其静态后向影响只能包含安全级别低于s的语句. Kirchner等人[13]提出一套协作静态分析器Frame-C,通过对程序的中间语言进行抽象语法树构建,并通过自动定理证明器验证属性,使用程序切片技术将程序拆分成小部分进行分析. LLift[14]是一个将大型语言模型(Large Language Models,LLMs)与静态分析相结合的框架,旨在增强对实际软件漏洞的检测能力. LLift利用LLMs对代码的理解能力来指导路径分析,减少探索路径并提高对复杂漏洞的分析精度,但其无法对逻辑漏洞进行检测. 最新的研究将信息流检测技术创新地应用于无人机飞控系统. 研究人员基于无干扰理论对飞控系统源代码进行静态分析,手动为变量添加安全标签. 之后基于自合成理论对信息流流向进行检测,以此检测无人机飞控系统信息流的机密性和完整性. 静态分析技术能够实现较高的代码覆盖率,但在应用于无人机飞控代码检测时,更关注的是信息流的保密性和完整性,而不是传统软件漏洞或逻辑漏洞的检测.

模糊检测方法对目标知识要求较低,易于扩展到大型应用中,已成为最受欢迎的漏洞检测方法. Alhawi等人[15]提出利用模糊技术和BMC技术来检测隐藏在软件状态空间深处的安全漏洞的方法. OSS-Fuzz[16]将传统模糊测试技术与可扩展的分布式执行相结合,使通用开源软件更加安全和稳定. 但以上方法对软件系统中潜在的逻辑漏洞无法进行有效数学公式表示,无法检测出逻辑漏洞. 针对逻辑漏洞,RVFuzzer[17]建立了一种控制不稳定性检测机制,通过观察机器人车辆的操

作行为检测控制系统缺陷,控制不稳定性检测机制的结果被用作反馈指导输入生成,但仍然无法实现高覆盖率,并错过一些包含不正确配置的搜索空间. LGDFuzzer[18]采用带学习模型的模糊算法来评估配置,加速搜索过程. 然而,该工作中的评估只考虑了一个时间戳的预测偏差,该时间戳的搜索结果很可能受到瞬态干扰,影响搜索的准确性. PGFuzz[19]通过带有时间约束的时间逻辑公式表达期望的策略,使用这些策略作为模糊测试的指导指标,生成模糊输入,使当前状态和策略违反之间的距离最小化. 但是,这种静态分析方法不够健壮,无法处理需要触发动态执行的策略违反. ICSEARCHER[20]应用遗传算法搜索“高概率”的错误配置,并通过基于深度学习的预测器验证配置. 虽然可以检测到逻辑漏洞,但依旧无法达到高覆盖率. EM-Fuzz[21]结合实时内存检查和模糊测试来检测漏洞,使用内存插桩技术来收集代码覆盖率,引导模糊测试. 但EM-Fuzz使用的测试用例生成策略可能在初期有效,随着测试深入,无法适应代码的复杂性,导致生成的测试用例无法有效触发新的代码路径. Firm-Fuzz[22]利用静态分析辅助生成测试用例,通过在模拟仿真环境中植入监控装置捕捉程序异常. 但其依赖于特定的策略生成用于测试的输入请求. 如果这个策略没有覆盖到系统所有可能的输入路径,那么它无法生成能够触发某些漏洞的测试请求. ChatAFL[23]利用LLMs对协议规范的理解能力引导种子变异并提升代码覆盖率,但LLMs无法完全理解所有类型的协议或特定协议的深层细节,导致路径覆盖率不高,有些漏洞无法被检测. 因此,模糊测试仍然存在效率低、代码覆盖率低等缺点.

针对上述现有大多数静态测试方法仅依靠数字世界构造输入,难以及时发现无人机逻辑错误,动态测试方法可以发现逻辑错误,但测试覆盖率低的问题,本文提出了一种支持物理交互的无人机飞控软件安全测试方法. 针对检测代码覆盖率低的问题,在代码层面通过静态分析得到代码可执行路径. 针对静态测试无法检测逻辑安全的问题,本方法基于其代码的物理语义,将无人机代码运行转换为仿真环境下的飞行任务并执行,在仿真环境下对无人机进行安全性判断. 针对无人机物理环境及飞行状态复杂,人工制定飞行安全策略不完备这一问题,本文提出结合基于机器学习的异常检测方法,对无人机飞行安全的状态进行更加充分准确的分析. 此外,多组对比实验也验证了本文所提方法在无人机飞控系统异常检测问题上的有效性.

2　基础知识

本文所提方法主要基于软件安全检测方法和异常检测方法,下面就相关概念和基本知识予以介绍.

2.1　软件安全检测方法

软件安全检测方法包括静态测试和动态测试.

2.1.1　静态测试

静态测试[24]是通过对程序代码本身、二进制文件以及相关元数据的分析来确定程序的行为和特征,而不需要实际执行程序. 这种方法可以用于发现程序中的漏洞和错误,提供一些有用的信息优化程序的性能和可靠性. 静态分析的局限性是受到混淆技术的影响,无法分析程序在运行时实际触发的行为检测. 符号执行方法是典型的静态检测方法.

符号执行[25]是一种基于静态分析的漏洞检测方法,通过对程序的符号变量进行代数求解,自动生成程序的各种路径,并在路径中插入各种输入值进行测试,找到程序的所有漏洞.

符号执行的实现通常包括以下步骤:

(1)将程序中的每个变量都用一个符号来代替,生成符号执行树;

(2)从程序的入口开始,按照程序的控制流图(Control Flow Graph,CFG)生成符号执行路径;

(3)对符号执行路径中的每个约束条件,如if语句和while循环等,使用符号变量代替,生成符号约束条件;

(4)对符号约束条件进行求解,得到所有满足条件的输入值;

(5)将每个满足条件的输入值插入到符号执行路径中,执行程序并分析程序的执行结果,找到程序中的漏洞.

符号执行的优点是能够对程序进行全面分析,找到程序中所有可能的漏洞,包括难以发现的漏洞. 缺点是符号执行的计算复杂度很高,需要耗费大量的时间和计算资源.

2.1.2　动态测试

基于动态分析的软件安全检测方法主要是通过执行程序,观察程序在运行过程中的行为和状态,发现潜在的漏洞. 模糊测试[26]是基于动态检测的软件安全检测方法. 它通过生成大量的随机、不合法或异常输入尝试触发目标程序中的漏洞或异常行为,进一步验证程序的稳定性和安全性.

AFL(American Fuzzy Lop)[27]是一种基于模糊测试的软件安全工具,旨在发现程序中的漏洞和缺陷. 通过智能的测试策略和自动化的测试过程,AFL能够生成多样化的测试用例并评估程序的稳定性和安全性. 因广泛适用性、容易集成性以及强大的社区支持使其成为各种编程语言和软件项目的通用漏洞测试工具,有助于提高软件的安全性和可靠性. AFL提供2种主要的

插桩方法，分别是GCC插桩（GNU Compiler Collection）和LLVM插桩（Low Level Virtual Machine）. GCC插桩是AFL的传统插桩方法之一，它通过修改程序源代码并使用GCC编译器实现插桩. 这种方法简单直接，但可能会对程序的性能产生一定影响，并且不够灵活，无法应用于所有编程语言和项目.

LLVM插桩是AFL的另一种插桩方法，它使用了LLVM编译器框架来实现插桩. 在LLVM插桩中，AFL会通过LLVM的中间表示（Intermediate Representation，IR）来修改程序，并在IR级别插入额外的代码收集测试覆盖率信息. 然后，使用LLVM编译器将修改后的IR代码编译成二进制可执行文件. 相比GCC插桩，LLVM插桩则使用LLVM编译器框架来实现插桩，具有更高的灵活性和性能. 它能够适应不同编程语言和项目的需求，并且能够进行更精细的代码分析. 本文在之后的研究中采取LLVM插桩.

2.2 异常检测方法

异常检测又称为离群点检测，对于监控CPS的运行状态至关重要，一旦检测到偏离正常模式的异常活动，它能够及时发出警报. 这些异常活动通常表现为数据分布的偏移，被称为入侵、异常值或故障. 传统的异常检测，人们会通过手动设置固定阈值来识别异常，但随着数据规模和复杂性的增加，这种方法变得耗时且不切实际. 为了解决这个问题，已经为单变量时间序列开发了一些异常检测方法[28]，其中异常是根据单个特定指标识别的. 但由于CPS的复杂性，单个指标无法全面代表其整体状态. 同时，由于异常的多样性和缺乏足够的标记数据，传统的异常检测方法无法应对现代CPS产生的大规模数据，近年来，许多研究人员开始采用机器学习方法来处理大规模多变量数据. 在这些方法中，深度学习备受青睐，根据标签的可用性采用有监督、半监督和无监督的异常检测方案.

有监督异常检测方法需要正常和异常的标签，神经网络通过多层结构学习数据特征，并用带标签数据调整参数进行分类. 但因为依赖异常标签且难以覆盖所有异常样本，所以应用范围有限.

无监督异常检测方法直接检测数据中的异常值，通常使用聚类、密度估计、统计学方法等来确定数据中的异常值. ECOD是一种无监督异常检测方法，通过评估每个特征的累积分布函数值找出异常点. 它的优势在于不需要标签，适用于未标记数据. 但其半监督的学习过程限制了对已有异常规则的利用，会导致部分与正常相近的异常模式未被发现. Yuen等人[29]提出的基于回归的异常值检测（Regression based Outlier Detection，ROD）算法用于多元数据的离群点检测，通过分析三维子空间提高检测精度，它擅长处理高维数据，但计算成本较高，且对非线性数据的检测能力有限. Liu等人[30]提出的孤立森林是一种无监督的异常检测方法，通过随机特征分割识别异常，它适用于连续和高维数据，但对噪声敏感，可能会误判低密度的正常样本为异常.

半监督异常检测方法仅需要正常训练数据，使用基于有监督的学习方法来识别正常数据，然后使用无监督的方法来检测剩余数据中的异常值. 在实际的大量飞行任务中，发生异常的过程在整体过程中占比很少，因此，应用半监督的方法在进行半监督检测时，有效数据集的获取是重点，标签的质量影响数据集的有效性，进而影响模型的效果，而本文通过对测试用例进行模糊测试，得到可执行路径覆盖率高的飞控任务，从这类任务中获得的数据集更有效.

3 无人机飞控软件安全测试方法

针对现有漏洞检测方法中存在的物理逻辑正确性难以表示、人工定义安全策略困难、自动化程度低等问题，本文设计了一种支持物理交互的无人机飞控软件安全漏洞智能检测框架及方法，如图1所示.

在映射表构建模块中，本研究设计了一种代码约束语句——物理控制映射方法，支持对飞控系统Ardupilot进行分析，可以采用同样的方法对飞控系统PX4进行分析. 本研究首先使用静态分析技术提取控制语句中的除运算符以外的符号，为了能更准确地进行映射，本研究将这些符号区分为变量和函数，依据其特性进行不同的映射策略，得到变量—物理语义映射表、函数—物理语义映射表，通过与飞控任务执行方法的结合得到代码约束语句——物理控制方法映射表，并对其生成的飞控指令进行动态检测，判断其是否可真实执行.

在飞控任务构建测试模块中，本研究首先通过对飞行控制系统源代码进行静态分析得到程序执行图，基于程序执行图提取分支条件以及执行路径，之后基于映射表对分支条件进行转换得到物理控制方法，在条件分析转换部分，本研究设计了基于编译原理的代码约束语句转换方法. 基于本研究已构造的映射表，经过词法分析、语法分析、语义分析对代码约束语句进行分解后转换，最终得到较为准确的物理控制方法. 再将物理控制方法与执行路径结合，同时对触发该路径的条件分支进行模糊测试，对条件进行物理控制转换，得到可触发执行路径的无人机物理控制方法. 之后，根据物理控制方法特性，将其插入基础飞行任务之中，得到最终可触发路径的无人机飞行任务，在仿真环境下运行并观测. 本文以飞控系统Ardupilot中降落模式的一段代码为例，对无人机飞控软件系统安全测试方法过程进行说明.

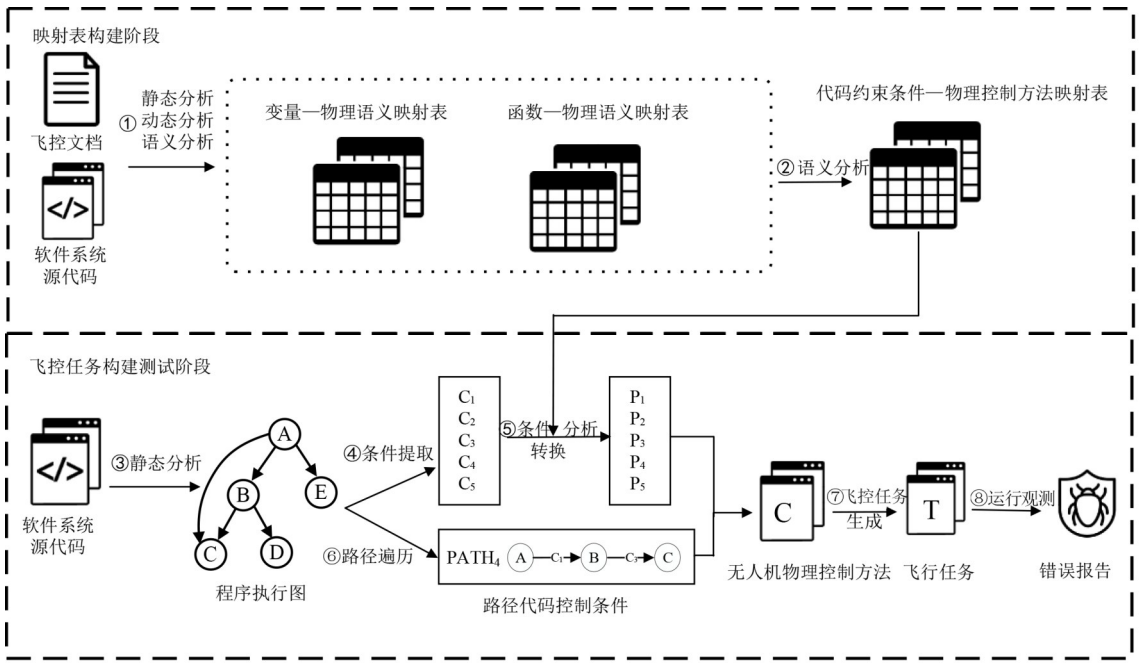


图1　无人机飞控软件系统安全测试方法流程图

3.1　构建映射表

无人机软件系统的代码中许多变量和函数与无人机的配置参数、物理状态、用户控制有关. 在对待检测程序检测之前,为了模拟真实物理环境下无人机的测试用例,本研究对无人机软件系统源代码的代码变量及函数进行分析后得到其对应的配置参数和物理状态,同时构建与其关联的飞控指令,用于构建真实物理环境下的飞控任务. 为了在真实物理环境下实现无人机飞控代码程序控制,本研究对无人机飞控系统代码与飞控文档进行静态分析,基于动态分析其可执行性,构建代码约束条件——物理控制方法映射关系表. 映射表构建流程如图2所示.

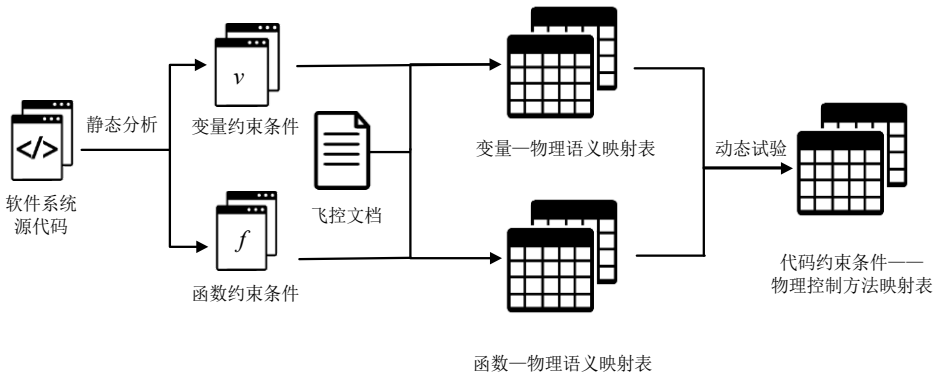


图2　映射表构建流程

如图3所示,对于待检测程序,为了在仿真环境下触发每一条可能的执行路径,需要对它的约束条件进行分析,将代码与无人机真实物理控制方法相对应. 找到对应的物理控制方法. 将代码中复杂的约束条件分为2类来获取其物理语义,一类为变量约束条件,一类为函数约束条件.

(1)对于变量约束条件,分为2类:

(a)无人机高级配置参数:飞控软件系统在编译运行之前,存在默认的飞控文档,运行时会读取飞控文档中的配置参数值. 一些变量对应的配置参数是只读的,但是无人机高级配置中的参数在特定条件下是可写的. 因此,对于代码约束条件,通过对飞控文档的人工分析,提取出代码约束条件、物理语义、参数配置、物理状态等关键信息去构造变量——物理语义映射表,如表1所示. 若想对无人机进行操作,此类约束条件需要对配置参数进行值的设定,因此采取配置的方式触发.

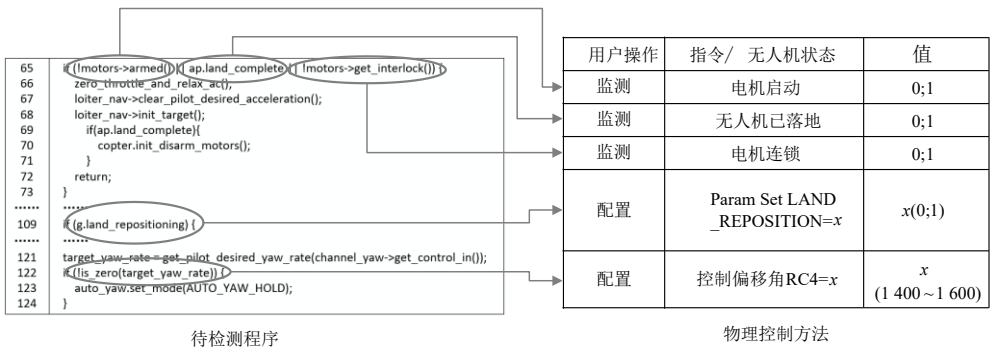


图3 无人机飞控系统约束条件转换示例

例如对于代码中g.land_repositioning这一变量,它对应的配置参数为LAND_REPOSITION,这一配置可设为0或者1.

(b)无人机当前物理状态:一些约束条件需当无人机处于某种物理状态时满足. 当无人机处于某种物理状态时,传感器会将其状态反馈于软件系统,触发与之相关的变量发生改变. 如表1所示,该表格中代码变量的值来源于飞控系统Ardupilot官网对参数列表取值范围的整理. 对于代码中ap.land_complete这一变量,它对应无人机已降落的物理状态,当无人机处于已降落的物理状态时,该参数值为1,反之为0. 对于此类变量约束条件的物理控制方法,本研究会采取配置加监测结合的方式,对于所示例子,首先调用用户命令使其降落,再设置监测点,从监测点开始记录数据,即为触发该约束条件之后的无人机状态.

表1 变量—物理语义映射示例表

代码约束条件	物理语义		参数配置	物理状态	监测/配置	值
g.land_ repositioning	降落重定位	LAND_REPOSITION		—	配置	0;1
ap.land _complete	无人机完全降落	—		无人机已降落	配置+监测	0;1

(2)对于函数约束条件,因为其函数调用的复杂性,笔者从3个方面对其进行语义分析,分别是对象、函数名、参数.

若能从对象名的中文释义直接获得代表对象含义的语义信息(例如表2中的motor,通过其含义可知motor为电机)则直接添加其语义信息;反之,该对象名中无法提取语义信息,则需查找该对象定义及注释,获取该对象语义. 函数名是函数约束条件核心,与对象名的语义获得方式相同,根据函数名中文释义及约束条件的注释信息构建其对应的物理语义(例如对于表2中的is_zero即可识别为“是否为0”). 一些函数调用需要传递参数,若函数约束条件中包含参数,需要对参数进行信息提取,根据函数名中文释义及约束条件的注释信息构建其对应的物理语义(例如表2中的target_yaw_rate即为目标偏航角). 在对函数对象、函数名、函数调用进行物理语义分析后,将其组合得到函数约束条件最终的物理语义. 函数—物理语义映射示例如表2所示,值的范围来自飞控系统Ardupilot官网对参数列表取值范围的整理.

表2 函数—物理语义映射示例表

对象	函数名	参数	物理语义	监测/配置	值
motor	armed	—	电机启动	配置	0;1
motor	get_interlock	—	电机安全锁打开	配置	0;1
—	is_zero	target_yaw_rate	偏移角是否为0	配置	0~360

对约束条件中的变量及函数构建物理语义映射表,无法直接使用他们的物理语义对仿真环境下的无人机进行控制. 需要将物理语义进一步与无人机飞行控制指令结合,得到与代码约束条件对应的物理控制方法,本研究以控制无人机的Python库(DroneKit)命令为例进行物理控制方法的生成. Python库中包含对无人机进行控制的一些指令,根据表1和表2中的物理语义及配置/监测项,查询DroneKit库文档以找到实现所需控制功能的相应方法或属性,DroneKit库文档提供了详示例代码. 例如ap.land_complete对应的物理语义为无人机完全降落,根据该物理语义在文档中找到能实现该物理语义对应的属性location.global_relative_frame.alt,接着根据

检测项对飞控任务中对应的无人机高度变量vehicle.Location.global_relative_fram.alt进行监测,当其为0时,则该条件为真,反之为假. 另一类约束条件需要对无人机进行操作,例如g.land_repositioning,采用参数配置指令"vehicle.parameters['LAND_REPOSITION']=x"来改变参数LAND_REPOSITION,进一步改变g.land_repositionin. 当LAND_REPOSITION赋值为1时,g.land_repositioning为真,反之为假,如表3所示.

表3　代码约束条件—物理控制方法映射示例表

代码约束条件	物理语义	飞控指令	x值
motors->armed()	电机启动	vehicle.armed	0;1
ap.land_complete	无人机已降落	vehicle.location.global_relative_frame.alt=x	0~max_height
motors->get_interlock()	电机安全锁打开	vehicle.disarmed=x	0;1
g.land_repositioning	高级配置LAND_REPOSITION	vehicle.parameters['LAND_REPOSITION']=x	0;1
is_zero(target_yaw_rate)	偏移角是否为0	condition_yaw(x)	0~360

在最终的代码约束条件——物理控制方法映射表中,包含代码约束条件、物理语义、飞行控制指令、指令可选择值. 研究对无人机飞控系统代码的所有条件中的变量及函数进行提取分析,得到代码约束条件——物理控制方法映射表如表3所示,为之后构建可以触发代码中约束条件的飞行任务提供便利.

3.2　构建飞控任务

在构建映射表的基础上,本研究对无人机飞控软件系统程序进行测试,首先由分析引擎对软件系统源代码进行约束条件分析及执行路径分析,构造仿真环境下可运行的飞控程序,对无人机运行状态进行观察,判断是否发生安全问题.

3.2.1　构建程序执行图

为了深入代码层面对无人机软件系统进行验证,需要先对飞控系统代码进行代码编译分析. 一段代码的运行可以看成系统的状态变化,并发系统因为行为不确定性可以有多个可能的后续状态,产生一棵状态树. 它又可以看成是所有可能的系统初始状态经历各种可能变化的集合,也就是把一棵状态树看成是有限或无限条路径的集合,一条路径所代表的是系统的一次可能的运行情况. 飞控代码进行编译并生成控制流图,进一步重构出对应的程序执行图来构建路径集合,通过在仿真环境下触发每条路径,确保无人机在各类运行情况中都能安全运行.

本研究对源代码中分支的关键词if进行递归搜索分析,得到并列的分支关系及嵌套的分支关系,在分析过程中,研究对代码进行分类,将其分为分支条件下的代码及分支约束条件代码,根据分支关系及代码块,本研究构造待验证程序的控制流图 $\boldsymbol{G}_{\mathrm{f}}$:

$$\boldsymbol{G}_{\mathrm{f}} = < V_{\mathrm{f}}, \boldsymbol{E}_{\mathrm{f}} > \tag{1}$$

其中,V_{f}为节点集合;$\overrightarrow{\boldsymbol{E}_{\mathrm{f}}}$是一组表示节点代码执行方向的有向边.为准确描述程序的执行流程,为之后触发执行过程做准备,本研究根据执行顺序和条件关系及先前提取的分支条件构造程序执行图,在控制流图的基础上,为有向边添加分支条件代码,进一步将其重构为带条件的程序执行图 $\boldsymbol{G}_{\mathrm{c}}$,具体定义如下:

定义3.1节对于程序代码C构建程序执行图 $\boldsymbol{G}_{\mathrm{c}}$:

$$\boldsymbol{G}_{\mathrm{c}} = < V_{\mathrm{c}}, \boldsymbol{E}_{\mathrm{c}}, \Phi, I_{\mathrm{c}} > \tag{2}$$

其中,V_{c}为程序段节点集合;$\boldsymbol{E}_{\mathrm{c}}$为一组表示节点代码执行方向的有向边集合;$\Phi = \{\phi_i | 1 \leqslant i \leqslant N\}$($N$为约束条件总数)为代码中的约束条件集合;$I_{\mathrm{c}}$为$\boldsymbol{E}_{\mathrm{c}}$与$\Phi$单射且满射的映射关系,$\boldsymbol{G}_{\mathrm{f}}$到$\boldsymbol{G}_{\mathrm{c}}$的转换由算法1实现.

算法1　$\boldsymbol{G}_{\mathrm{f}}$到$\boldsymbol{G}_{\mathrm{c}}$转换算法

输入: 控制流图 $\boldsymbol{G}_{\mathrm{f}}$
输出: 程序执行图 $\boldsymbol{G}_{\mathrm{c}}$
1. $V_{\mathrm{c}} = V_{\mathrm{f}}$
2. $\boldsymbol{E}_{\mathrm{c}} = \boldsymbol{E}_{\mathrm{f}}$
3. For $\boldsymbol{e}_i \in \boldsymbol{E}_{\mathrm{c}}$ do:
4. $v_{\mathrm{start}} = \boldsymbol{e}_i$.startnode//设置路径开始节点
5. $v_{\mathrm{end}} = \boldsymbol{e}_i$.endnode//设置路径结束节点
6. ϕ_i= IRanalysis($v_{\mathrm{start}}, v_{\mathrm{end}}$)//获取分支条件
7. I_{c}.add($\boldsymbol{e}_i, \phi_i$)//添加分支条件

为了方便后续对约束条件进行物理控制方法转换过程的叙述,在此,令 ϕ_0=(g.throttle_behavior&THR_BEHAVE_HIGH_THROTTLE_CANCELS_LAND)!=0.

3.2.2　约束条件物理语义分析

飞控系统存在多条执行路径,为遍历触发其路径,需分别满足每条路径的约束条件. 不同以往形式化逻辑公式求解中直接对变量赋值来满足约束条件. 仿真环境下运行无人机来触发条件,需要考虑到无人机的用户指令、配置参数及物理状态. 在单纯程序分析基础上需要对无人机约束条件进行更深入的分析,基于编译方法和对无人机整体静态分析及映射表,设计对约束条件的词法分析、语法分析及语义分析. 如图4所示,以约束条件ϕ_0为例,首先对其进行词法分析,将其内部词汇分类为标识符、运算符等. 之后在语法分析的基础上,

得到每个变量/函数应该满足的条件，最后根据构建的代码约束条件——物理控制方法映射表得到可满足各条件的物理控制方法. 其中，若一些变量是配置文档的宏定义，那么在词法分析时，就对其进行替换，例如THR_BEHAVE_HIGH_THROTTLE_CANCELS_LAND值替换为(1<<1).

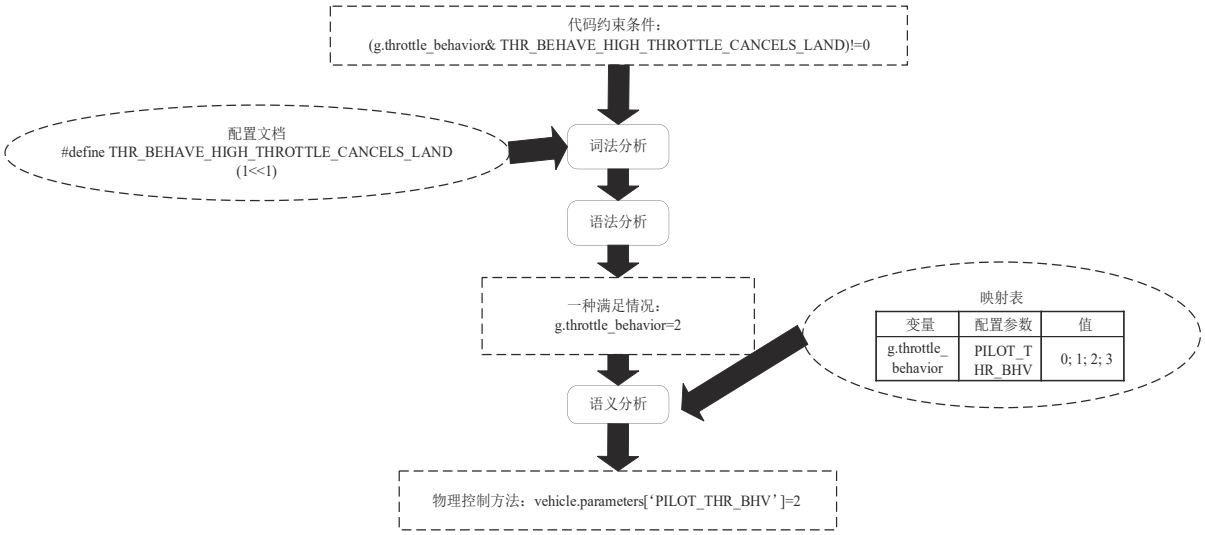


图4 单个条件分析方法

具体分析方法如下：

(1)词法分析. 约束条件中经常会包含复杂的运算，主要有逻辑运算、算术运算、位运算、关系运算. 为之后对条件进行深入分析，条件中的每个单词划定标识，以便之后的语法分析. 基于编译原理教程中单词种别标识规则，种别标识符及含义如表4所示. 对于条件控制语句进行词法分析后，提取出每一个词及为它划定标识符.

定义1 对于单个传输条件ϕ_i词法分析后单词标识集合$\boldsymbol{\phi}_i^w=\{\boldsymbol{\phi}_{i,k}^w|1\leqslant k\leqslant N\}$，其中，$\boldsymbol{\phi}_{i,k}^w$为单个单词标识二元组：

$$\boldsymbol{\phi}_{i,k}^w=<\mathrm{id},\mathrm{value}> \tag{3}$$

其中，id为该单词的种别标识符，value为该单词具体的值.

例如对于ϕ_0，其词法分析之后的结果如下：

表4 标识符规则示例表

分组	标识符	含义
1	SLP	(
2	SRP	)
3	INC	++
4	NE	!=
5	CONST	常量
6	FUNC	函数名
7	SLP	变量
…	…	…

$$\boldsymbol{\phi}_0^w=\{\boldsymbol{\phi}_{0,0}^w=<\mathrm{SLP},->,\boldsymbol{\phi}_{0,1}^w=<\mathrm{IDN},\mathrm{g.throttle\_behavior}>,$$
$$\boldsymbol{\phi}_{0,2}^w=<\mathrm{BIT},\&>,\boldsymbol{\phi}_{0,3}^w=<\mathrm{IDN},(1\ll 1)>,\boldsymbol{\phi}_{0,4}^w=<\mathrm{SRP},->,$$
$$\boldsymbol{\phi}_{0,5}^w=<\mathrm{REL},!=>,\boldsymbol{\phi}_{0,6}^w=<\mathrm{CONST},0>\}$$

(2)语法分析. 基于左依赖最左推导(Left to right Leftmost derivation，LL)文法分析，按照参考编译器文法规则，本研究可以对条件分析得到每个变量应赋予的值.

定义2 对于代码约束条件ϕ_i，定义$\boldsymbol{\Phi}_i^S=\{\boldsymbol{\phi}_i^{(1)},\boldsymbol{\phi}_i^{(2)},\cdots,\boldsymbol{\phi}_i^{(j)},\cdots,\boldsymbol{\phi}_i^{(n)}\}$，其中$\boldsymbol{\phi}_i^{(j)},1\leqslant j\leqslant n$，表示$\phi_i=\mathrm{true}$时$\phi_i$中变量或者函数的一组赋值. $\boldsymbol{\phi}_i^{(j)}$采用合取范式表示如下：

$$\boldsymbol{\phi}_i^{(j)}=\boldsymbol{\phi}_{i,x}^{w(j)}\wedge\boldsymbol{\phi}_{i,y}^{w(j)}\wedge\cdots\wedge\boldsymbol{\phi}_{i,z}^{w(j)} \tag{4}$$

其中，$\boldsymbol{\phi}_{i,x}^{w(j)}$为第$j$种满足约束条件情况下单词$\boldsymbol{\phi}_{i,x}^w$.value的赋值，$\boldsymbol{\phi}_{i,x}^w$.id$\in\{\mathrm{IDN},\mathrm{FUNC}\}$，例如，对于输入$\phi_0$，依据映射表中g.throttle_behavior对应的取值范围推导分析可知，可以满足$\phi_0==1$的取值为

$$\boldsymbol{\phi}_0^{(1)}=\left\{\boldsymbol{\phi}_{0,1}^{w(1)}\right\}=\{\mathrm{g.throttle\_behavior}=2\}.$$

(3)语义分析. 当明确代码中的变量及函数如何赋值后，若要将其映射到无人机真实的物理输入，需要根据先前构建的代码约束条件——物理控制方法映射表，对赋值语句进行物理语义分析及转换.

定义3 对于代码约束条件ϕ_i，定义$\boldsymbol{\Theta}_i=\{\boldsymbol{\theta}_i^{(1)},\boldsymbol{\theta}_i^{(2)},\cdots,\boldsymbol{\theta}_i^{(n)}\}$，其中，$\boldsymbol{\theta}_i^{(j)}$表示$\phi_i=\mathrm{true}$时，与$\boldsymbol{\phi}_i^{(j)}$对应的一组物理控制方法. $\boldsymbol{\theta}_i^{(j)}$采用合取范式表示如下：

$$\theta_i^{(j)}=\theta_{i,x}^{(j)}\wedge\theta_{i,y}^{(j)}\wedge\cdots\wedge\theta_{i,z}^{(j)} \tag{5}$$

其中,$\theta_{i,x}^{(j)}$为赋值语句$\phi_{i,x}^{w(j)}$对应的一条物理控制方法语句. 对于赋值语句$\phi_{0,1}^{w(1)}$=(g.throttle_behavior = 2),通过在表3代码约束条件——物理控制方法映射表查找throttle_behavior这个变量,可以得到在仿真环境下需发送参数配置指令$\theta_{0,1}^{(1)}$=(vehicle.parameters[PILOT_THR_BHV]=2)触发.

3.2.3　模糊测试得到飞行控制方法集合

在3.2.2节中已经通过词法分析和语法分析将代码约束条件转换为可触发飞控代码的语句,即触发语句,并通过语义分析得到能在仿真环境下运行的物理控制方法,若要尽可能多地得到能触发飞控软件系统异常的任务,需要对程序执行图中每条路径上的可触发语句进行模糊测试. 首先对程序执行图进行遍历,得到初始节点到终止节点的所有路径,构建路径集合. 其本质为查找2个节点之间的所有路径. 生成的可执行路径集合为$\boldsymbol{P}=\{\boldsymbol{p}_k|1\leqslant k\leqslant N\}$,其中,$\boldsymbol{p}_k=\{<v_i,\phi_i>|1\leqslant i\leqslant N\}$,由第$k$条执行路径上的所有节点及代码约束条件组成. v_i为该路径上执行经过的节点,ϕ_i为经过到下一个节点边上的约束条件(最后一个节点的约束条件记为NULL). 单边ϕ_i存在一种或多种可能的触发语句,则单条路径$\boldsymbol{p}_k$对应一种或多种可能的触发语句集合$c_k=\{c_k^i|0\leqslant i\leqslant N\}$,即对单个路径查找其所有可能执行触发语句,并通过语义分析得到触发语句取值范围,即单个路径上有一种或多种触发语句组合,将每一个语句组合写进一个触发文件,以便后续模糊测试使用,得到路径对应的触发语句及其取值范围集合构成待检测程序的整体检测集合.

对路径上的触发语句进行模糊测试采用AFL(American Fuzzy Lop)工具. 这一工具的优势在于能够自动地对飞控系统中每个路径上的飞控方法进行模糊测试. 其利用遗传算法,以覆盖率为导向,不断生成测试用例,通过动态插桩技术监控程序的行为. 对触发文件进行模糊测试的流程如图5所示.

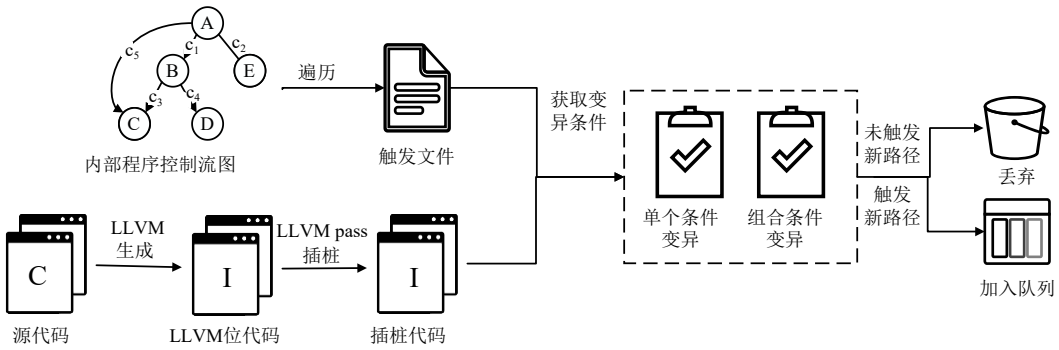


图5　模糊测试流程

首先,使用LLVM中的C++编译器Clang将飞控源代码编译成LLVM位代码,afl-llvm-pass.so.cc实现了LLVM-mode下的一个插桩LLVM Pass,可以用其对生成的LLVM位代码进行插桩. 插桩是为了在代码执行时收集执行路径的覆盖信息,以便模糊测试工具后续使用. 接着使用AFL对插桩后的LLVM位代码进行编译. AFL提供了afl-clang-fast编译器,它会自动添加模糊测试所需的代码. 研究将3.2.3节得到的待检测程序集合划分为与无人机地面站输入相关的命令及与具体飞行任务相关的命令,地面站输入的命令包括飞行指令(如起飞、降落、悬停)、飞行姿态(如俯仰、偏航、滚转)、飞行速度和高度等. 具体飞行任务涉及飞行路径、风速、气压、温度等参数,以及遇到的环境条件(如障碍物、天气情况)等,这样进行划分以便于第5节在仿真环境下进行测试. 将上面得到的触发文件作为初始测试集加入输入队列,AFL首先会对文件中的每一个触发语句进行突变,将触发语句划分为含参数的语句及不含参数的语句,对于含参数的语句进行合理数值内的合法变异及合理数值外的异常变异,变异策略如下.

(1)命令参数值的合理范围边界测试:在命令的参数值中针对边界情况进行测试,比如极大值、极小值以及边缘情况. 例如,在飞行速度参数中,将速度设置为最大值、最小值,或者尝试非常接近这些边界值的情况.

(2)命令参数值的合理范围随机变异测试:在参数合理范围内修改控制指令中的参数值,例如目标位置、飞行高度、飞行速度等,以探索不同的控制路径.

(3)命令参数值的异常测试:在命令的参数值中引入异常情况,比如超出正常范围的值、不合理的值或者错误的值. 例如,在高度参数中,尝试将高度设置为负数或者超出无人机可达范围的值.

对于不含参数的指定指令对其进行非法变异,策略如下.

(1)插入异常字符或数据:向控制指令中插入异常

字符、特殊符号或非法数据,以测试飞控系统对异常输入的处理能力.

(2)引入无效指令:向输入中引入一些无效或未知的指令,以测试系统对于未知指令或非法指令的处理方式.

在对单个指令进行变异后,对指令组合进行变异,策略如下.

命令组合方式的变异:修改输入命令的组合方式,比如改变命令的顺序、重复发送相同命令、发送不完整的命令序列等. 这有助于测试系统对不同命令组合的处理是否正确,以及系统在处理异常命令序列时的稳定性.

若文件经过变异触发了新路径,则将其保留添加到队列中,若没有触发新路径,则不进行记录. 上述过程会一直循环进行,本文将触发了新路径的全部文件进行记录,触发异常的文件作为异常文件,未触发异常的文件作为正常文件,接着将全部文件映射到无人机真实的物理输入,需要根据先前构建的代码约束条件——物理控制方法映射表,对触发语句进行物理语义分析及转换,即语义分析,最终得到每条路径的飞行控制方法集合,即飞行任务.

4 基于机器学习的无人机飞控软件异常检测方法

研究提出的支持物理交互的无人机飞控软件安全测试方法有效对无人机飞控代码进行物理世界下飞控任务的转换,但是无人机飞控运行输出状态复杂,传统安全测试基于数学公式判定难以鉴别,采用人工分析开销大. 为此,本文提出了一种基于机器学习的无人机飞控软件异常检测方法.

图6展示了基于机器学习的无人机飞控软件异常检测方法的总体架构设计. 该架构主要由2个阶段组成,一是训练异常检测模型,二是使用异常检测模型对无人机状态进行检测,发现异常飞行任务. 该方法使用无人机正常飞行任务飞行数据作为正样本,基于统计学方法训练异常检测模型. 在检测阶段,对于单条路径

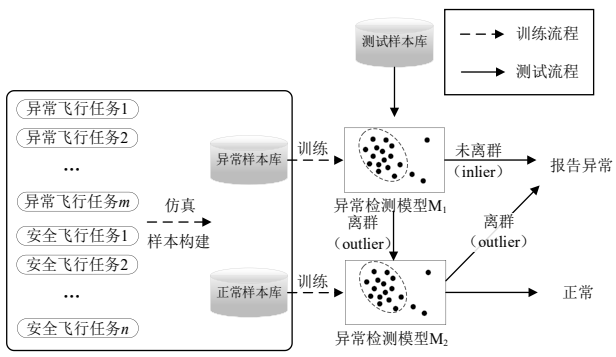


图6 基于机器学习的无人机飞控软件异常检测方法架构

而言,本研究生成多个飞行任务并执行,提取其飞行数据,数据进一步分析得到异常检测需要的特征向量,之后输入异常检测模型,判断是否为正常的飞行任务. 若存在异常,那么记录该路径在异常飞行任务中存在安全问题.

4.1 数据集收集

若要训练可以有效监测异常的模型,需要选择合适的无人机飞行状态特征. 本研究根据其特性将故障情况分为2类,一类是严重事故情况,一类是规则违反情况.

当无人机发生严重事故,例如迅速坠落或者撞毁时,它的速度和加速度会有较大幅度的变动,因此选择瞬时速度及加速度作为其特征. 同时无人机出现异常往往不是瞬间发生,其正常状态到异常状态具有一定连续性,所以本研究不仅记录瞬时数据,还对一段时间内无人机状态变化进行观察,选取平均速度作为特征.

第2类异常状态不十分明显,经过多次实验发现,此类异常过程中除了可能有速度加速度的变化,还有无人机姿态与模式的相关性变化. 例如在引导模式下起飞后不应该存在偏移角的变化. 因此选取当前模式、模式转换、位置、航向、高度作为特征.

综上所述,本方案从8个维度来检测无人机飞行状态是否发生异常.

在特征选择完成后,需要收集数据样本进行模型训练. 由于无人机飞行数据量庞大,不可能将所有飞行任务数据用于训练异常检测模型. 本文的第3节中通过模糊测试获取的飞行任务对任务路径具有高覆盖率,涵盖了飞控代码的关键部分. 因此,在有限的数据选择中,这些任务是最佳选择. 本研究使用无监督异常检测方法对无人机进行实时监测.

(1)首先选取30个正常飞行任务. 这些飞行任务同时满足以下条件:

A. 不发生坠毁、飞走、碰撞等严重故障;

B. 始终与地面控制系统保持联系,每隔3 s向地面控制系统发送心跳包;

C. 不违反某种模式下的安全状况(例如在引导模式下起飞后不会出现偏移角不规则变化).

(2)本研究选取30个异常任务,选择其故障时间段进行记录,这些飞行任务满足以下一种或多种条件:

A. 发生坠毁、飞走、碰撞等严重故障;

B. 与地面控制系统失去联系;

C. 违反某种模式下的安全状况.

本研究构造一定量的安全飞行任务和不安全飞行任务,将其在仿真环境下执行,并选择需要的数据进行日志保存(例如高度、瞬时速度、位置、模式、航向等),得到训练数据集. 用于后续双重异常检测模型的训练.

4.2 运行状态验证

传统的软件检测方法,其定义较为正常的模型,即基于正常行为数据训练的模型,进行自动化检测. 因为控制系统的不确定性及无人机运行环境的恶劣性、无人机物理输入的多样性,难以直接使用运动学模型定义一个通用的计算公式来判断无人机的飞行状态是否正常. 使传统的验证方法难以直接用于无人机判定. 基于机器学习的异常检测算法方法具有及时、灵活等特点,目前已广泛用于嵌入式设备的实时异常感知,在实际的大量飞行任务中,发生异常的过程在整体过程中占比很少,因此应用半监督的异常检测方法学习无人机系统的正常模式.

无人机飞行过程中,可能发生坠落碰撞等事件,可通过无人机飞行状态检测,漏洞也可能导致无人机突然的失控,可通过对无人机系统运行状态进行检测. 所以对于无人机的安全检测,主要针对无人机飞行状态和软件系统运行状况进行监测.

研究认为无人机各个检测维度之间不是独立的,而是具有关联性. 采用传统的异常检测方法单纯看一个点是否偏离均值点是盲目的,更合理的方法是去估计一个点的尾端概率,就是分布在极端位置的可能性,当数据尾部概率值非常低时,表明其中的数据值被认为是异常的. 若要计算尾端概率,可以通过估计多维数据的累积分布函数(Cumulative Distribution Function, CDF)来计算. 对于所有实数,累积分布函数定义如下:

$$\hat{F}_d^l(\boldsymbol{x}) = \frac{1}{n} H_d\left(\boldsymbol{X}_i \leqslant \boldsymbol{x}\right) \tag{6}$$

右端CDF计算公式如下:

$$\hat{F}_d^r(\boldsymbol{x}) = \frac{1}{n} H_d\left(-\boldsymbol{X}_i \leqslant -\boldsymbol{x}\right) \tag{7}$$

其中,H为d维度下,$\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n$中所有满足$(\boldsymbol{X}_i \leqslant \boldsymbol{x})$的出现次数.

偏度向量$\boldsymbol{b}_i$计算公式如下:

$$\boldsymbol{b}_i = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \overline{\boldsymbol{x}_i}\right)^3}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \overline{\boldsymbol{x}_i}\right)^2}^3} \tag{8}$$

左端CDF、右端CDF与偏度向量共同构成异常检测模型. 对于待检测数据$\boldsymbol{X}_i$,d维度的copula函数观测值由以下公式计算:

$$\hat{U}_{d,i} = \hat{F}_d^l\left(\boldsymbol{x}_i\right) \tag{9}$$

$$\hat{V}_{d,i} = \hat{F}_d^r\left(\boldsymbol{x}_i\right) \tag{10}$$

$$\hat{W}_{d,i} = \begin{cases} \hat{U}_{d,i}, \boldsymbol{b}_d < 0 \\ \hat{V}_{d,i}, \boldsymbol{b}_d \geqslant 0 \end{cases} \tag{11}$$

其中,d维度的$\boldsymbol{x}_i$左端经验观测值为$\hat{U}_{d,i}$,d维度的$\boldsymbol{x}_i$右端经验观测值为$\hat{V}_{d,i}$,偏度校正的经验观测值为$\hat{W}_{d,i}$.

之后计算$\boldsymbol{X}_i$在所有维度下的尾端分布概率负对数:

$$p_l = -\sum_{j=1}^{d}\log\left(\hat{U}_{j,i}\right) \tag{12}$$

$$p_r = -\sum_{j=1}^{d}\log\left(\hat{V}_{j,i}\right) \tag{13}$$

$$p_s = -\sum_{j=1}^{d}\log\left(\hat{W}_{j,i}\right) \tag{14}$$

最后的$\boldsymbol{X}_i$尾端分布概率取左尾经验、右尾经验、偏度校正经验生成的概率负对数的最大值Score,尾部概率越小,其负对数越大,如果某个点具有较小的左尾部概率、较小的右尾部概率或较小的偏度校正尾部概率,则认为该点是异常值.

4.3 基准模型

单一的异常检测模型难以较准确地检测出无人机的各类异常问题,为对已有的无人机安全及异常规则进行较好利用,本研究提出一种基于机器学习的无人机飞控软件异常检测方法,即分别利用正常训练样本和异常训练样本训练出2个半监督模型,其中异常训练样本训练出的半监督模型用于识别已知的无人机异常问题;正常训练样本训练出的半监督模型用于对初步判定正常的无人机状态进行进一步筛选. 通过双层异常检测模型的筛选,实现对已有安全规则的利用和更精确的异常检测效果.

训练出2个异常检测模型M_1与M_2,其中M_1通过异常样本进行训练,M_1的未离群点(inlier)即为被已知的无人机异常类型所命中,而M_1的离群点(outlier)即为未被已知的无人机异常类型所命中. M_2通过正常样本进行训练,可以为对上述未命中的初步正常样本进行进一步筛选,其中M_2的离群点即为与正常模式偏离的无人机状态点,极有可能出现无人机故障等异常情况. 因此,在检测阶段,待测试样本首先经过M_1进行初步筛选,可以筛选出可能存在安全问题的飞行任务. 本研究又基于不安全飞行的训练数据训练了一个异常检测模型M_2,对第1步筛选出的飞行任务进行2轮异常检测,筛选出表现更为正常的飞行任务.

5 实验结果与分析

本研究中采用了Gazebo进行无人机仿真,利用其高度真实的物理模拟功能,能够对无人机的动力学、传感器、行为和环境情况进行准确模拟. 同时,通过软件在环仿真(Software In The Loop,SITL)的方法,实现了对无人机系统的全面仿真,包括其软件系统的运行和交互,提供了一种有效的研究手段.

5.1 任务覆盖率对比

首先对第3节中的方法进行测试. 代码单条路径为程序执行图中开始节点到叶子节点的路径. 路径覆盖率,可以展现漏洞检测工具对代码可执行空间进行的检测覆盖率. 根据3.2.1节中构建程序执行图可知,采用条件覆盖,本研究从模糊测试已获得的飞控任务中提取出与Ardupilot3.6.0中的常用的降落模式(mode land)、定高模式(mode althold)、引导模式(mode guided)、绕圈模式(mode circle)、返航模式(mode RTL)、游荡模式(mode loiter)6种模式相关的任务,比较该任务下路径覆盖率. 在3.2.3节中将待检测程序集合划分为与无人机地面站输入相关的命令及与具体飞行任务相关的命令,在仿真环境下对不同指令进行执行,本研究方法对每一模式下相应的任务路径覆盖数目以及6种模式下实际运行路径覆盖数目占总路径比率进行记录,如表5和图7所示.

表5 路径覆盖表

模式	land	althold	guided	RTL	circle	loiter	总和	覆盖率/%
本研究	60	3	64	24	10	16	177	97.7

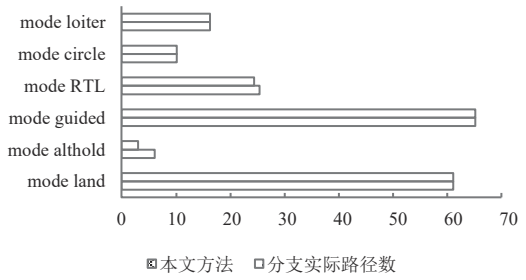


图7 路径覆盖率统计图

本研究对已检测代码的路径覆盖率为97.7%,可以看到几乎每种模式下,本研究都可以几乎完全覆盖所有任务路径,这得益于本研究使用深度遍历对源代码路径进行了全部静态分析得到其可执行路径. 同时,该数据也可以反映本研究所构建的代码约束语句——物理控制方法映射表与语义分析方法的有效性,有效地对源代码中的分支条件进行转换,以满足可执行路径真正在仿真环境下的触发.

不过,仍然有个别路径未在仿真环境下成功触发,例如自稳模式下,函数takeoff.running()无法触发,导致2条路径无法在仿真环境下执行,无人机飞控系统3.6.0中,并不可以进行在自稳模式下执行定高的自动起飞任务,这条路径及接口可能是为之后代码的更新做准备. 本研究在实验时,并未对gazebo环境进行修改,所以在RTL模式中,无人机返航时,因为始终拥有地形数据,导致无法触发! wp_nav->set_wp_destination(rtl_path.return_target). 本研究阅读了该段代码,发现其逻辑为当无人机无法加载地形时,会重新启动RTL,并再次试图加载地形,这个行为是较为安全合理的.

综上所述,整体而言,以上实验数据说明本研究支持物理交互的无人机安全测试方法是可以高覆盖率地生成无人机飞行任务,本研究设计的代码约束语句——物理控制方法映射表与语义分析方法可以有效地对无人机飞控软件系统源代码中的分支条件语句进行转换.

为了证明提出方法的有效性,本研究与现有较新的研究成果进行了对比,对每一模式下不同方法的路径覆盖数目以及6种模式下实际运行路径覆盖数目占总路径比率进行对比,对比结果如表6和图8所示,因为本章方法主要在于将代码约束条件转换为物理控制方法,并提高覆盖率. 并且对于漏洞检测工具而言,覆盖率也是重要的检测指标之一. 这里本研究主要对路径覆盖率做一个对比.

表6 与其他方法对比

模式	land	althod	guided	RTL	circle	Loiter	总和	覆盖率/%
本研究	60	3	64	24	10	16	177	97
Avis	12	2	15	19	7	16	71	39
ICSEARCHER	28	3	21	15	7	16	90	49

本研究主要侧重对源代码的可执行路径代码进行物理语义转换,将其路径上的约束条件转换为物理控制方法,所以在条件分支覆盖上显示了更好的效果,基本可以完全覆盖.

Avis是一种基于立即模型检测的仿真测试方法. 它侧重在不同的运行模式之前注入故障,因此对于单个模式的检测较为宏观,每个模式下更多的是触发默认运行路径以及故障保护路径. 对于分支较多的降落模式覆盖率较低,对于分支较少的源代码更容易触发.

ICSEARCHER主要采用的是遗传算法(Genetic Algorithm,GA)和机器学习预测器来检测无人机飞行控制系统中的配置错误,但它依赖于搜索策略来探索参数空间,这些策略可能无法覆盖所有可能的飞行控制逻辑和条件,例如存在一个配置参数集,其中多个参数以非线性或复杂的相互作用影响飞行稳定性. 如果ICSEARCHER的搜索策略没有考虑到这种复杂的交

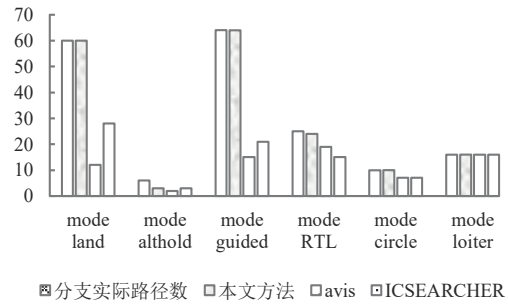


图8　与其他方法对比图

互,可能无法识别出这种组合导致的潜在不稳定配置,所以部分条件无法触发.

经过对比实验,可以看出本研究提出的方法生成的飞行任务可以较好地对无人机源代码在运行时进行测试覆盖.

5.2　异常检测模型评价指标及实验结果

为了评估模型的性能,本研究使用了精确率(Precision)、召回率(Recall)以及 F_1-score 等评价指标. 在介绍这些指标之前先引入如下 4 个基本参数:True Positive(TP)、False Negative(FN)、False Positive(FP)、True Negative(TN).

基于这 4 个参数,评价指标即可通过计算获得:

Precision 为模型预测为良性的应用程序中,预测正确的比例. 其计算公式为

$$\text{Precision}=\frac{\text{TP}}{\text{TP}+\text{FP}} \tag{15}$$

Recall 为模型预测为良性的应用数量除以真实值为良性的应用程序数量. 其计算公式为

$$\text{Recall}=\frac{\text{TP}}{\text{TP}+\text{FN}} \tag{16}$$

F_1-score 综合了 Precision 和 Recall 的产出结果,是一个较为综合的指标. 其计算公式为

$$F_1\text{-score}=\frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}} \tag{17}$$

为评价本研究所提出的异常检测算法的准确性,本研究基于 Ardupilot 在 gazebo 环境下共进行了 110 次飞行,其中包含 80 次未触发异常的飞行、20 次触发异常的飞行和 10 次利用安全规则产生的飞行过程. 记录上述飞行过程中无人机状态变化数据,共得到 5 000 个无人机状态数据点位,选取其中的 3 000 条正常数据作为训练集,1 000 条正常数据和 1 000 条异常数据作为测试集,10 次安全规则的飞行过程用于对异常检测模型进行补充. 为对比本研究提出的异常检测算法,选择了现有的先进异常检测算法作为基线方案,其中包括:

(1)IsolationForest. IsolationForest 也被称为孤立森林算法,是一种经典的异常检测算法,由南京大学的周志华团队在 2008 年提出.

(2) ECOD (Empirical-Cumulative-distribution-based Outlier Detection). ECOD 设计了一种无监督离群点检测方法,采用经验累积分布函数.

(3)ROD(Rolation-based Outlier Detection). 一种基于旋转的离群点检测方法.

在本研究的自制数据集下,各种方法的检测效果如表 7 所示.

表 7　在自制数据集下的异常检测效果　　单位:%

异常检测方法	精确度	召回率	F_1-score
孤立森林	93.3	92.7	93.0
ECOD	97.0	91.7	94.3
ROD	93.3	92.7	93.0
神经网络(有监督分类)	74.5	73.4	73.9
本研究	97.5	92.3	94.8

在现有较新研究成果的对比下,本研究所提出的基于双层异常检测的异常检测算法实现了最好的效果. 其中孤立森林与基于旋转的异常值检测(ROD)方法表现稍差,因为无法学习到无人机状态的不同变量之间的关联性. ECOD 算法也利用了经验累积分布函数,但其半监督的学习过程限制了对已有异常规则的利用,会导致部分与正常相近的异常模式未被发现,本研究所提出基于双层异常检测的异常检测算法,通过对已有异常规则的过滤,进一步提升了算法的召回率. 同时,与有监督分类相比,较少的异常规则也难以对各种异常情况进行覆盖,因此有监督的效果较差. 本研究所提出的方法在多种现有方法中得到了最高的 F_1-score,实现了最佳的综合性能.

为进一步分析方法中双层异常检测模块的有效性,本研究进行了消融实验以验证,实验结果如表 8 所示,其中 M_1、M_2 代表仅使用本研究所提出方案中的单层模型,M_1+M_2 代表本研究所提出的方法.

表 8　在自制数据集下的消融实验　　单位:%

方法	精确度	召回率	F_1-score
M_1	99.3	23.6	38.0
M_2	97.4	91.3	94.3
M_1+M_2	97.5	92.3	94.8

异常规则难以完全覆盖所有的异常情况,M_1 模型的召回率很低,仅能对相似类型和场景的异常进行识别,而例如坠落和飞走等异常规则在不同场景下的表现形式也会有较大差异(例如跌落高度、方向的不同),同时,异常规则也无法覆盖所有的无人机状态情况. 当只使用 M_2 模型时,此时异常检测的原理即为半监督的异常检测模型,其检测性能与无监督的异常检测方法相似. 最终 M_1+M_2 相较于 M_2 实现了更高的召回率,M_1

模型的叠加增强了对已有异常规则的利用能力，使模型可以发现部分与正常相近的已知异常模式，例如翻滚角变化、偏移角变化等不甚明显的异常.

5.3 基于机器学习的无人机飞控软件异常检测方法实验结果

(1)对漏洞 APM# 17970的检测

本研究对RTL模式进行检测，异常检测模型检测一个飞行任务存在问题，查看日志发现RTL着陆时与最初起飞位置存在偏移，并且发生了无人机侧翻. 飞行任务为：起飞任务—降落—重定位—RC1信道值为1 400—进入RTL模式. 如图9所示：该飞行任务对应的理论测试路径为<descent_run—!copter.failsafe.radio—!g.land_repositioning>但是要触发该测试路径，飞行任务还包括一些前置任务，通过本研究对执行路径跟踪，路径为<takeoff—mode_land_gps_run—ap.land_complete—g.land_repositioning—!is_zero(target_roll)—mode_rtl_init—mode_rtl_run—climb_start—climb_return_run—return_start—loiterathome_start—loiterathome_run—descent_run(检测)—!copter.failsafe.radio—g.land_repositioning—!is_zero(target_roll)—land_run>.

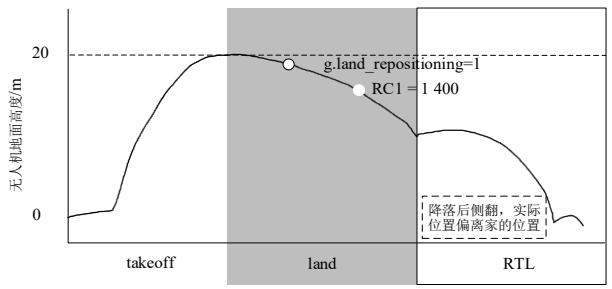


图9 APM#17970 漏洞检测飞行过程模拟图

发现异常情况后，结合运行过程与飞行日志发现，实际飞行的路径与本研究理论测试路径不同. 为避免遗漏未重新初始化导致的漏洞，对于存在默认值的变量，若初始化值满足检测路径，会分为2条路径，一是进行重新初始化，一是不进行重新初始化. g.land_repositioning默认为0，因此，该次实际飞行任务中，本研究并未在测试descent_run前对变量g.land_repositioning进行重新初始化. 在本次实验中，降落模式中对g.land_repositioning进行了更改，之后RTL模式下g.land_repositioning依然是1，在这种情况下，roll持续不为0导致无人机侧翻，而如果按照理论测试路径，g.land_repositioning为0的情况下，执行精确着陆，即使roll的值不为0，软件系统也不会控制无人机实际roll值改变. 另一条对g.land_repositioning重新初始化的飞行任务执行后，飞行数据显示正常.

综上所述，本研究的无人机飞控软件系统安全检测方法可以对漏洞APM#17970成功检测，即RTL模式缺少对g.land_repositioning的重新初始化，导致在RTL模式前进行重定位后，切换到RTL模式后，若误触roll角或pitch角控制且并未及时矫正，则可能导致无人机无法回到家的位置，也有可能导致侧翻.

(2)对漏洞 APM#17709的检测

本研究对loiter模式进行检测，异常检测模型检测一个飞行任务存在问题，查看发现在模式切换后，出现一段较大的减速过程. 飞行任务为：起飞—RTL模式—loiter模式，模拟情况如图10所示.

该飞行任务对应的理论测试路径为<mode_loiter_init-mode_loiter_run—!copter.failsafe.radio-Loiter_Flying>，但是要触发该测试路径，飞行任务还包括一些前置任务，通过本研究对执行路径跟踪，路径为<takeoff—mode_rtl_init-mode_rtl_run—climb_start—climb_return_run—mode_loiter_init—mode_loiter_run—!copter.failsafe.radio—Loiter_Flying>.

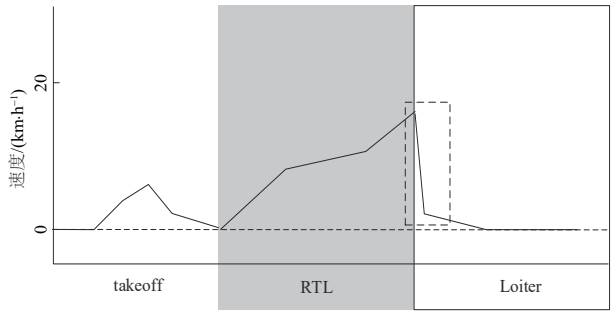


图10 APM#17709漏洞检测飞行过程模拟图

发现异常情况后，结合日志查看，发现是在切换到Loiter模式的一瞬间发生了较大减速，虽然没有造成直接的危害，但是这样急速的减速显然存在潜在危害. 因为减速发生在Loiter模式下，所以首先对Loiter模式源代码进行查看，根据此次实验中实际执行路径，对路径执行过程中源代码进行查看，在Loiter_Flying分支下存在对速度进行调节的函数.

根据飞控官方漏洞报告，此项漏洞报告错误位置在此对速度进行调节的函数之中.

(3)对比实验

为评价本研究所提出的安全检测方法的有效性，本研究对第3节分析得到6个模式的飞行任务进行仿真环境下执行，并使用异常检测检测是否存在异常.

为对比本研究提出的异常检测方法，选择了现有的先进异常检测方法作为基线方案，其中包括：

(a)Avis. 一种针对无人机软件系统的现场模型检测方法，通过实时监测无人机的行为来验证其是否符合预定义的安全模型. 该算法动态收集飞行过程中的

关键参数和事件数据,并实时检测不符合的情况,确保无人机的安全性.

(b)ICSEARCHER. 一个基于遗传算法和机器学习预测器的自动化测试系统,旨在优化测试用例的选择和生成. 它通过评估测试用例的适应度,优先选择最有效的用例,以提高发现软件缺陷的能力.

本研究采用以上2种方法对降落、定高、游荡、返航、引导、绕圈模式进行检测,测试其对13个已知漏洞的检测能力. 检测结果如表9所示,√表示可以成功检测到,×表示未成功检测到.

由于Avis侧重于对传感器故障逻辑漏洞进行检测,它所执行的飞行任务为简单的模式切换,仅进行少量的用户控制,任务改变重点在于故障注入位置及时间不同,所以仅识别了测试漏洞中的两个传感器故障逻辑漏洞.

ICSEARCHER基于遗传算法得出策略进行空间搜索,即使ICSEARCHER的测试用例与可能的影响因素已经具有极高的相关性,但是它依赖于搜索策略来探索参数空间,这些策略可能无法覆盖所有可能的飞行控制逻辑和条件,忽略一些可能的用户输入和参数,所以对于测试漏洞中的5个漏洞未检测出.

本研究的方法生成的飞行任务可使待检测模式的该飞行任务对应的可执行路径做到完全覆盖,但是有些漏洞需要其前置任务处于特定条件下才会触发,当本研究的前置任务未模糊到该模式下时,此漏洞可能不会被发现. 本研究的方法对于测试漏洞检测,仅对APM-13441和APM-17970 2个漏洞未识别,如表9所示.

表9　不同方法对已知漏洞检测结果

漏洞编号	Avis	ICSEARCHER	本研究
APM-351	×	√	√
APM-2147	×	√	√
APM-8390	×	√	√
APM-10671	×	×	√
APM-13441	×	√	×
APM-16021	√	√	√
APM-16027	√	×	√
APM-17709	×	√	√
APM-17951	×	√	√
APM-17954	×	×	√
APM-17970	×	×	×
APM-18700	×	√	√
APM-20939	×	×	√

6　结论

多变的物理环境、复杂的功能结构使得无人机飞行控制系统在开发过程中容易引入各类软件安全问题,使得无人机可能发生任务中断、直接损毁等状况. 因此,如何检测无人机飞控软件系统的安全问题成为热点研究方向. 现有的安全测试方法主要依靠数字世界输入,难以准确表征物理世界行为,导致安全测试结果准确率低、测试路径覆盖率低. 本研究提出了一种支持物理交互的无人机飞控软件安全测试方法. 将静态与动态分析方法相结合,并结合物理语义对无人机飞行控制软件的安全性进行测试,保障了测试方法的准确率和覆盖率. 同时,本文对无人机飞行数据进行合理的特征提取训练双重异常检测模型,从而对无人机飞控软件运行输出进行检测. 本文方法可有效检测出无人机飞控软件系统中的安全问题.

参考文献

[1] 王祥科, 刘志宏, 丛一睿, 等. 小型固定翼无人机集群综述和未来发展[J]. 航空学报, 2020, 41(4): 15-40.
WANG X K, LIU Z H, CONG Y R, et al. Miniature fixed-wing UAV swarms: Review and outlook[J]. Acta Aeronautica et Astronautica Sinica, 2020, 41(4): 15-40. (in Chinese)

[2] 高玉伟, 韩庆, 裴扬. 某型无人机的易损性评估和减缩设计[J]. 航空计算技术, 2007, 37(3): 44-47.
GAO Y W, HAN Q, PEI Y. Vulnerability assessment and vulnerability reduction design of an UAV[J]. Aeronautical Computing Technique, 2007, 37(3): 44-47. (in Chinese)

[3] ZHANG B B, CHEN X. Research on fault injection in UAV dynamic test[J]. Industrial Control Computer, 2005, 18(5): 9-10.

[4] JHALA R, MAJUMDAR R. Software model checking[J]. ACM Computing Surveys, 2009, 41(4): 1-54.

[5] HUMPHREY L R. Model checking for verification in UAV cooperative control applications[J]. Lecture Notes in Control and Information Sciences, 2013, 444: 69-117.

[6] GUO H Y, WU M, ZHOU L D, et al. Practical software model checking via dynamic interface reduction[C]//Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. New York: ACM, 2011: 265-278.

[7] LEESATAPORNWONGSA T, HAO M, JOSHI P, et al. SAMC: Semantic-aware model checking for fast discovery of deep bugs in cloud systems[C]//Operating Systems Design and Implementation. Emeryvile: USENIX Association, 2014: 1-16.

[8] YUAN B, SONG Z X, JIA Y, et al. MQTTactic: Security analysis and verification for logic flaws in MQTT implementations[C]//2024 IEEE Symposium on Security and

Privacy (SP). Piscataway: IEEE, 2024: 2385-2403.

[9] SHAIKH E, MOHAMMAD N, MUHAMMAD S. Model checking based unmanned aerial vehicle (UAV) security analysis[C]//2020 International Conference on Communications, Signal Processing, and their Applications (ICCSPA). Piscataway: IEEE, 2021: 1-6.

[10] TAYLOR M, CHEN H C, QIN F, et al. Avis: In situ model checking for unmanned aerial vehicles[C]//2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Piscataway: IEEE, 2021: 471-483.

[11] MOORE S, CHONG S. Static analysis for efficient hybrid information-flow control[C]//2011 IEEE 24th Computer Security Foundations Symposium. Piscataway: IEEE, 2011: 146-160.

[12] FERRAIUOLO A, XU R, ZHANG D F, et al. Verification of a practical hardware security architecture through static information flow analysis[J]. ACM SIGARCH Computer Architecture News, 2017, 45(1): 555-568.

[13] KIRCHNER F, KOSMATOV N, PREVOSTO V, et al. Frama-C: A software analysis perspective[J]. Formal Aspects of Computing, 2015, 27(3): 573-609.

[14] LI H N, HAO Y, ZHAI Y Z, et al. Enhancing static analysis for practical bug detection: An LLM-integrated approach[J]. Proceedings of the ACM on Programming Languages, 2024, 8(OOPSLA1): 474-499.

[15] ALHAWI O M, MUSTAFA M A, CORDIRO L C. Finding security vulnerabilities in unmanned aerial vehicles using software verification[C]//2019 International Workshop on Secure Internet of Things (SIOT). Piscataway: IEEE, 2019: 1-9.

[16] SEREBRYANY K. OSS-Fuzz-Google's continuous fuzzing service for open source software[C]//26th USENIX Security Symposium. Emeryvile: USENIX Association, 2017: 1-28.

[17] KIM T, KIM C H, RHEE J, et al. Finding input validation bugs in robotic vehicles through testing[C]//28th USENIX Security Symposium (USENIX Security 19). Emeryvile: USENIX Association, 2019: 425-442.

[18] HAN R D, YANG C, MA S Q, et al. Control parameters considered harmful: Detecting range specification bugs in drone configuration modules via learning-guided search[C]//2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE). Piscataway: IEEE, 2022: 462-473.

[19] KIM H,OZMEN M O, BIANCHI A, et al. PGFuzz: Policy-guided fuzzing for robotic vehicles[C]//Proceedings 2021 Network and Distributed System Security Symposium. San Diego: NDSS, 2021: 1-17.

[20] HAN R, MA S, LI J, et al. Range specification bug detection in flight control system through fuzzing[J]. IEEE Transactions on Software Engineering, 2024, 50(3): 461-473.

[21] GAO J, XU Y W, JIANG Y, et al. EM-fuzz: Augmented firmware fuzzing via memory checking[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(11): 3420-3432.

[22] SRIVASTAVA P, PENG H, LI J H, et al. FirmFuzz: Automated IoT firmware introspection and analysis[C]//Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things. New York: ACM, 2019: 15-21.

[23] MENG R J, MIRCHEV M, BÖHME M, et al. Large language model guided protocol fuzzing[C]//Proceedings 2024 Network and Distributed System Security Symposium. Singapore: Internet Society, 2024: 1-17.

[24] PARK J, LEE H, RYU S. A survey of parametric static analysis[J]. ACM Computing Surveys, 2021, 54(7): 1-37.

[25] BALDONI R, COPPA E, D´ELIA D C, et al. A survey of symbolic execution techniques[J]. ACM Computing Surveys, 2018, 51(3): 1-39.

[26] YUN J, RUSTAMOV F, KIM J, et al. Fuzzing of embedded systems: A survey[J]. ACM Computing Surveys, 2022, 55(7): 1-33.

[27] GUTMANN P. Fuzzing code with AFL[C]//Login Usenix Mag. Chicago: Computer Science, 2016: 11-14.

[28] GRAABÆK S G, ANCKER E V, FUGL A R, et al. An experimental comparison of anomaly detection methods for collaborative robot manipulators[J]. IEEE Access, 2023, 11: 65834-65848.

[29] YUEN K V, ORTIZ G A. Outlier detection and robust regression for correlated data[J]. Computer Methods in Applied Mechanics and Engineering, 2017, 313: 632-646.

[30] LIU F T, TING K M, ZHOU Z H. Isolation forest[C]// 2008 Eighth IEEE International Conference on Data Mining. Piscataway: IEEE, 2008: 413-422.

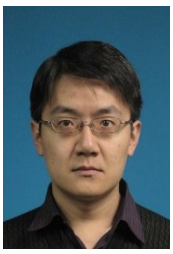
作者简介

习　宁　男,1986年10月出生,陕西省渭南人. 西安电子科技大学网络与信息安全学院教授,博士生导师. 主要研究方向为无人系统安全、工业互联网安全、信息流安全.

E-mail: nxi@xidian.edu.cn

周晓琳　女,2001年2月出生,黑龙江省齐齐哈尔人. 西安电子科技大学网络与信息安全学院硕士研究生. 主要研究方向为工业互联网安全,软件验证.

E-mail: zhouxiaolin0223@163.com



孙　聪　男,1982年7月出生,陕西省兴平人. 西安电子科技大学网络与信息安全学院教授,博士生导师. 主要研究方向为软件安全、程序分析、无人系统安全.

E-mail: suncong@xidian.edu.cn

李乔杨　男,1985年6月出生,陕西省商洛人. 西安航空计算技术研究所服务保障中心中层助理. 主要研究方向为嵌入式计算机设计.

E-mail: 2828884@qq.com

马建峰　男,1963年10月出生,陕西省西安人. 西安电子科技大学网络与信息安全学院教授,博士生导师. 主要研究方向为应用密码学、无线网络安全、数据安全、移动智能系统安全. 中国电子学会会员编号:E190004733F.

E-mail: jfma@mail.xidian.edu.cn

郭鑫玉　女,1998年6月出生,山西省晋中人. 主要研究方向为无人系统安全、软件验证.

E-mail: xinyu.g@foxmail.com