**摘** **要**

随着物联网技术的快速发展，JavaScript凭借其灵活易用、交互性强和跨平台优势，在嵌入式系统开发中占据了重要地位，为了满足嵌入式设备的开发需求，面向嵌入式平台的JavaScript引擎不断涌现。为了提升嵌入式系统的稳定性与性能，确保所采用JavaScript引擎的可靠性显得尤为关键。然而嵌入式设备低内存、弱算力等资源受限的特性与JavaScript动态语言特性的结合，导致传统测试方法难以有效挖掘引擎的潜在缺陷。同时，由于其广泛应用于物联网终端、工业控制系统等关键领域，若引擎中潜在的缺陷未被及时检测修复，不仅会造成资源浪费，甚至引发嵌入式系统和应用的安全隐患。

为了检测嵌入式JavaScript引擎中的缺陷，本文提出了一种基于参数类型推断的测试用例变异方法，并与差分测试结合实现了模糊测试框架。具体研究内容如下：

1. 对编译器进行模糊测试需要大量符合语法语义的测试用例，为满足这一需求，本文选取GitHub等主流开源代码平台中维护质量较高的JavaScript项目，并从中提取函数构建初始测试样本。
2. 本文设计了一种基于参数类型推断的测试用例变异方法，基本过程为从函数定义中识别并抽取其参数列表，根据函数体内该参数的行为模式，分析该参数可能的数据类型，并基于各类型设计丰富的变异策略与自调用表达式，最终生成了大量语法语义正确、高覆盖率的测试用例。
3. 本文设计并实现了EJSFuzz测试系统，并对多个主流的JavaScript嵌入式引擎进行了模糊测试，对发现的缺陷进行案例分析。通过与Fuzzilli，AFL等模糊测试工具进行对比实验，验证了EJSFuzz在代码覆盖率、漏洞发现率等方面的优势。

**关键词：**嵌入式JavaScript引擎 模糊测试 用例变异

作为软件生态的基础设施，编译器的正确性与可靠性不言而喻。编译器负责将人类可读的编程语句转换为机器码，这一转换过程严格遵循预定义的语言标准。语言规范明确定义了合法语法结构、语义约束条件以及对应的机器操作指令。各类编程语言均具备其独特的语法标准，并且这些标准会随着语言特性的演进而不断发展。语言规范的复杂性使得程序员在阅读和理解时面临很大的挑战，同时也给编译器测试带来了巨大的困难。如果编译器有问题，可能导致语义变化、性能退化等严重问题。对于大多数的开发人员来说，这类问题很难被定位和发现。因此，早期对编译器的测试十分重要。目前编译器的测试方法有很多，包括兼容性测试、自举测试等。

在现代软件安全领域，模糊测试由于其高覆盖率和自动化能力，已成为一种被广泛采用的漏洞检测技术。其核心机制是通过自动化生成或特殊变异策略构造海量合法/非法的输入样本，将这些样本动态注入目标程序，触发程序出现诸如崩溃、内存资源泄露或逻辑紊乱等异常现象，从而暴露潜在漏洞。该技术可被划分为基于生成模型的方法以及通过修改现有输入实现的变异型方法。模糊测试适用于多种测试模式，不论是具备源代码访问权限的白盒测试、仅有部分信息的灰盒测试，还是完全未知结构的黑盒测试。自1988年威斯康星大学的Barton Miller教授提出模糊测试以来，模糊测试已被广泛应用到各个研究领域，包括编译器，网络协议，Web漏洞挖掘等。此外，它也常与其他测试方法相融合，例如结合差分测试用于揭示不同编译器版本间的行为差异，或借助动态污点分析与符号执行提升模糊测试在路径探索方面的能力。

* 1. 在自动化测试领域，模糊测试凭借其高效的缺陷检测能力已被信息安全领域的研究人员广泛应用，逐渐演变为漏洞挖掘过程中的关键分析手段。随着AFL++、LibFuzzer等工具的相继出现，这些工具以其良好的用户交互设计和简便的操作流程，极大地推动了模糊测试技术的普及。下面介绍一些最新的模糊测试技术相关的安全研究。本文研究内容

本文提出了一种基于参数类型推断的测试用例变异方法，通过分析原始测试用例得到代码段的抽象语法树，提取语义信息，确定可进行变异的参数，根据数据类型执行变异，提升覆盖率与测试效率。具体研究内容如下：

1. JavaScript引擎语法解析策略。研究引擎在处理JavaScript代码时如何生成并优化抽象语法树，确保代码的语法和语义信息正确，从而执行高效准确的变异。
2. 测试用例变异策略，通过对JavaScript代码段的静态语法树分析，推断其可变异参数和数据类型，基于数据类型对该参数执行不同的变异策略，从而发现引擎更深层次的缺陷与漏洞。
3. 设计并实现EJSFuzz系统，通过对上述方法的研究，本文实现了EJSFuzz系统。对该系统的模块结构，算法流程做详细介绍，将其应用到各大主流嵌入式JavaScript引擎做模糊测试，评估该方案的可行性与实用性。
   1. 本文组织结构
4. **绪论。**本章首先概述了JavaScript语言及其应用背景，随后回顾了编译器测试领域的相关研究，并对当前JavaScript引擎测试的现状进行了简要分析。最后，阐明了本文的研究问题和研究目标。
5. **理论与实验论证**。本章主要阐述了本文测试对象JavaScript引擎的基本架构，系统中使用的模糊测试技术、测试用例变异技术。
6. **基于类型推断的嵌入式JS引擎的模糊测试方法**。 本章重点阐述了本文所提出的差分模糊测试方法的整体框架，详细描述了参数类型推断模块和变异算法的具体实现。
7. **系统设计与实验评估**。 本章主要介绍基于本文提出的模糊测试方法开发的嵌入式JavaScript引擎差分模糊测试系统EJSFuzz，介绍实验的设置与环境，然后对测试结果进行深入分析，并进一步评估实验的有效性。最后，探讨缺陷产生的根本原因并提出修复方案，同时与其他模糊测试工具进行对比实验。
8. **结论与展望**。 本章通过对实验数据的深入分析，揭示了差分模糊测试系统在推动嵌入式JavaScript引擎测试工作中的重要作用。此外，本文还指出了该系统在应用过程中的一些局限性，对未来的测试工作研究展望。

如图2-3所示，模糊测试的基本流程分为四个阶段：测试用例生成、执行测试用例、异常监视以及漏洞确认。

测试用例生成。为全面评估被测软件的稳定性与安全性，必须持续生成多样化的测试用例，而用例生成的策略对测试质量具有决定性作用。目前常见的方法大致可分为两类：生成型策略与变异型策略。生成型方法通常基于预设的语法规则或结构模板，通过一定的随机机制构造出新的输入数据，代表性工具如 jsfunfuzz。相较而言，变异型技术依赖于已有的测试样本，即“种子用例”，通过对其进行改写、重组或注入代码片段以形成新的测试样本。例如，LangFuzz通过拼接方式整合多个代码片段生成测试输入。生成式方法需要深入的领域知识以编写适合的规则，而突变式方法则更依赖随机性，可能难以触发特定漏洞。  
确定性变异：

1. bitflip（位翻转）：将1变为0，0变为1；即在二进制层面对数据位进行取反处理。
2. arithmetic（算术运算）：整数加/减算术运算
3. Interest（特殊值替换）：将AFL内置的边界测试值替换到原始测试用例中。

随机性变异：

由于 JavaScript 语言动态类型的特性，其变量在定义阶段不要求预先指定具体的数据类型，变量的类型只有在引擎执行时才能被确定。动态类型的设计为JavaScript带来了独特的灵活性与开发效率，也不可避免地引入了若干限制与问题，如代码的可读性低，类型错误只能在运行时暴露等。对于含有参数的函数，如果传入错误的参数会导致触发类型错误而提前中止测试流程，导致代码覆盖率降低，因此参数类型推断是执行精准变异不可或缺的一步。

JavaScript中共有8种基本的数据类型，其中值类型有6种：字符串（String）、数字(Number)、布尔(Boolean)、空（Null）、未定义（Undefined）、Symbol。引用数据类型有3种：对象(Object)、数组(Array)、函数(Function)[26]。在类型推断模块中，通过静态文本分析推断参数可能的数据类型，以图3-2为例讲解。

函数体中有一个返回语句ReturnStatement，返回了一个二元表达式BinaryExpression。这个二元表达式的左操作数为字面量“Hello”，右操作数为变量name。Function Properties为函数属性，它表明了该函数非生成器、非表达式、非异步。

escodegen 是一款 JavaScript 代码生成库，它用于将抽象语法树结构重新构建为具备可执行特性的源代码，通常与解析器esprima配合使用，形成完整的代码解析-修改-生成工作流。

、

测试结果处理模块由俩个子模块构成。分别负责对测试结果进行裁剪以及对无效或冗余信息进行筛除。在执行差分模糊测试的过程中，不同JavaScript引擎在持续接收输入用例的情况下，往往会生成数量庞大的输出数据。由于缺陷信息可能混杂于这些测试结果之中，仍需借助人工手段对其进行比对与判别。

1. 代码覆盖率：在评估模糊测试性能时，代码覆盖率常被用作核心评估参数之一。该指标反映了测试用例在执行期间能够触达的程序路径相对于所有可能路径的比例。依据检测粒度的不同，覆盖率通常可细分为以下几种类型：语句覆盖率，关注所有语句是否被运行；条件分支覆盖率，旨在验证每个判断条件的所有分支是否均被执行；路径覆盖率，强调测试能否遍历程序中所有的执行路线。例如，若某程序包含 1000 行代码，而模糊测试仅触发其中 600 行的执行，则代码覆盖率为 60%。高覆盖率通常意味着测试更充分，能够暴露更多潜在漏洞。

实验结果如表4.3所示，由于对照组采用随机策略，导致其类型准确率普遍偏低，范围为9.94%~21.28%。而在实验组中，各数据类型的准确率明显高于对照组。

基于上述参数类型变异实验，我们进一步将此策略应用于测试用例变异模块，聚焦于五类参数类型的修改操作。代码4-3与代码4-4中分别呈现了变异实施前后，测试用例结构和内容的具体差异。

在对JavaScript引擎进行模糊测试时，测试用例必须符合 JavaScript 语言的语法和语义要求，才能通过引擎的语法检查。因此生成的JavaScript代码的语法正确性很大程度上决定了模糊测试的效果。JShint是一款JavaScript代码静态语法检测工具，我们可以通过这款工具静态检查变异后的测试用例，统计其中语法正确的测试用例，依据公式（4-1），计算语法正确的测试用例在总变异测试用例里的比例，以此衡量变异测试用例的质量。

代码覆盖率能反映测试用例的利用率，通常情况下，代码覆盖率的提高意味着测试用例执行了更多的代码路径，也意味着该测试用例发现缺陷方面的能力更强。通过对比不同工具所生成的测试用例的覆盖率，可以评估各测试方法在生成测试用例质量方面的优劣。

本文通过对现有的模糊测试技术进行研究，提出了一种基于类型推断的测试用例变异方法，并在此方法的基础上实现了EJSFuzz系统。

本文的研究内容如下：

1. 本文研究了当前具有代表性的软件模糊测试技术，并进一步对JavaScript引擎模糊测试方向已有的研究成果进行了归纳分析。测试用例生成模型很大程度上决定了模糊测试方法的有效性，当前主流的方法分为生成算法与变异算法，生成式算法很难生成涵盖语言的丰富特性，而传统的变异算法基于字节变异，无法兼顾用例语法语义的正确性。
2. 本文提出了基于类型推断的测试用例变异方法。JavaScript语言动态类型的特点使其无法在运行前进行静态类型检查。而类型推断模块通过分析参数在函数内部的操作模式，确定其数据类型，本文还根据数据类型的不同设计了丰富的变异策略，提升了测试用例的代码覆盖率。

本文基于上述的测试用例变异方法实现了EJSFuzz系统，对系统内部各个模块做了简要介绍