模糊测试技术白皮书

主编单位 云起无垠
数说安全

2024 年 8 月

版 权 声 明

本书编委会

总顾问： 夏　虹

主　编： 谭晓生

副主编： 陈广勇　崔宝江　陈建军　胡晓旭　张明明　牛伟纳　郑　立　沈凯文

成　员： （按姓氏拼音为序）

陈宇轩　陈安莹　曹思玮　丁　皓　甘成昱　韩敬维　侯安然　季可航

李　唯　刘　坤　李　晶　李静文　李学龙　马思远　谭　杰　王海清

王　勇　王　雪　王书辉　夏　营　徐士朋　张新怡　周　鹏

作者单位： 清华大学　北京邮电大学　电子科技大学　北京中关村实验室　公安部第三研究所　中国信息通信研究院稳定性保障实验室　长安通信科技有限责任公司　数说安全　北京云起无垠科技有限公司

引言

在信息技术迅猛发展的数字时代，软件已成为一切的基石。海量代码正在快速改变各行业的生产范式，加速全球信息技术的进步。然而，随着开源组件和第三方代码的广泛应用，前所未有的安全风险也随之而来，使得软件安全问题成为企业乃至国家的关注焦点。《新思科技应用安全测试服务分析》报告[36]数据显示，97%的被测目标被发现存在某种形式的漏洞，其中 36%为严重及高风险漏洞，导致软件安全“灰犀牛事件”[1]频发。面对日益复杂的网络环境和多样化的攻击手段，传统基于规则的安全检测方法逐渐暴露出其局限性，难以及时识别新出现的安全漏洞。这种局限性突显了采用更先进、更灵活的安全技术的重要性，例如运用人工智能和机器学习技术提升安全检测能力，以更有效地识别和应对未知的安全威胁，从而确保软件和系统的安全。

在此背景下，模糊测试[11]（Fuzz Testing 或 Fuzzing）作为验证软件健壮性和安全性的关键技术，受到了广泛关注与应用。模糊测试通过模拟攻击者的行为，生成大量意外或随机的输入，挑战系统的稳定性和安全性，旨在发现潜在的安全漏洞和弱点。相比传统的基于规则的测试方法，模糊测试不依赖于已知的漏洞模式，而是通过随机变异的输入数据，全面覆盖系统的各类输入情况，从而大幅提高漏洞检测的广度和深度。因此，模糊测试在发现未知漏洞、提升测试覆盖率以及减少误报率方面具有无可比拟的优势。

[1] “灰犀牛事件”（Grey Rhino Event）是一个比喻性的术语，用来描述那些虽然明显可见但通常被忽视，直到它们成为严重问题的大型风险。

近年来，随着大语言模型和安全智能体技术的引入，模糊测试在应对复杂软件系统和未知威胁方面展现出更大的潜力。大语言模型通过对程序代码的深度理解和分析，能够生成更具针对性的测试用例，极大提升模糊测试的覆盖率和有效性。而且，安全智能体技术能够自动化地进行漏洞检测、分析和修复，大幅提高整体测试效率和安全防护水平。

本白皮书共分为八个部分：

- 模糊测试概述：介绍模糊测试的基本概念和历史发展。
- 模糊测试技术解析：深入探讨模糊测试的基本原理、关键技术和不同分类。
- 应用领域分析：详细介绍模糊测试在不同应用领域中的实际应用和效果。
- 需求与创新案例：展示行业内模糊测试的最佳创新案例，涵盖多个行业的具体需求和成功案例。
- 技术实施指南：详细讲解模糊测试的技术实施，包括工具选择、流程方法以及自动化和智能化的最新进展。
- 挑战与未来趋势：分析当前模糊测试面临的挑战和未来发展趋势。
- 政策和标准介绍：介绍国内外模糊测试相关的政策和标准。
- 结论与展望：总结模糊测试的整体价值，并对未来的发展方向进行展望。

目录

图目录

表目录

一、模糊测试概述

模糊测试[11]是一种前沿的自动化测试技术，专注于挖掘软件系统中的漏洞和缺陷。它的核心理念是通过向程序输入大量随机、异常或意外的数据（即“模糊”数据），从而触发系统异常行为或崩溃。模糊测试不依赖已知的漏洞模式或规则，而是利用半随机或基于人工智能的策略生成输入，深入探索系统潜在的漏洞。模糊测试能够发现内存泄漏、缓冲区溢出、代码注入等安全漏洞，以及程序中的其他错误或异常情况。这种测试方法目前已广泛应用于软件开发的各个阶段，包括产品测试、安全评估和漏洞挖掘等领域。本章将通过比喻生动地介绍模糊测试的基本思路，回顾其历史与发展，分析其与传统软件安全测试方法的区别，并展示其在现代软件开发和安全评估中的优越性、重要性和广阔前景。

1.1 什么是模糊测试技术

模糊测试就像一位探险家在未知迷宫中的探索。在这个比喻中，复杂的应用系统好比是一个充满未知和挑战的迷宫，而模糊测试器则是那位勇敢的探险家，努力探索迷宫中隐藏的秘密通道和出口。探险家依靠智慧和勇气揭开迷宫的秘密；同样，模糊测试器通过生成大量非预期或随机的输入，挑战应用系统的极限，揭露隐藏于代码深处的漏洞

和未知威胁。其具体工作步骤和过程如下：

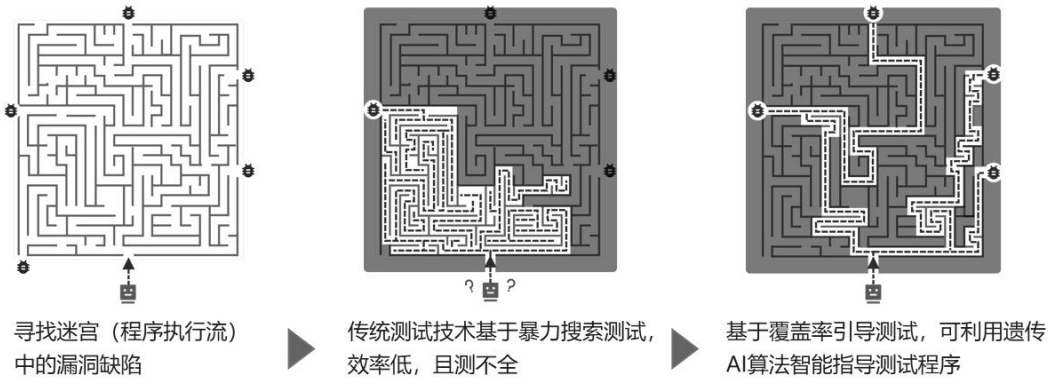

图 1 模糊测试迷宫探索示意图

- **进入迷宫（开始测试）：**模糊测试工具就像勇敢的探险家，踏进了一个充满未知路径和转折的迷宫。这个迷宫象征着一个复杂的软件系统，它包含了众多待探索和解密的执行路径和状态。在这一步，探险家（模糊测试工具）准备好一切必要的装备和信息，开始对迷宫（软件系统）进行探索。

- **迷宫探索（生成测试数据并测试）：**在这场探险里，探险家不按照预定路径寻找出口，而是随机或启发式引导地选择不同路径前行。相似地，模糊测试通过生成各种随机或异常数据，并将这些数据输入软件系统中进行测试，以观察其反应。探险家的每一步选择都可能带来新的发现，而模糊测试的每一组数据输入同样可能揭示出系统的未知行为。

- **寻找隐藏的通道和出口（发现新路径和软件问题）：**探险家可能在迷宫的每个转角发现一个隐藏的通道或未知的出口。在模糊测试中，生成的异常数据可能触发软件

系统中隐藏的漏洞，或导致程序行为异常，从而揭露潜在的安全问题。这一步骤至关重要，能够帮助测试者有效识别出系统中的薄弱环节和潜在风险。

- **记录和分析（分析测试结果）：**探险家在发现新路径或遭遇陷阱时，会仔细记录并分析。模糊测试过程同样需要记录软件对于异常输入的反应，并进行深入的分析，确定安全漏洞或程序错误的存在是真实的。详细的记录和分析有助于理解系统的行为模式，从而为后续的改进和修复提供依据。

- **不断迭代（持续测试）：**探险家会通过不断的探索，尝试新的路径，直到彻底探索迷宫的每一个角落。模糊测试亦然，通过持续生成新的测试数据，测试者能够全面检测并增强软件系统的安全性，确保更多安全漏洞被发现并修复。持续的迭代测试有助于提高系统的稳定性和安全性，使其在面对各种异常输入时仍能保持正常运行。

通过这个比喻，模糊测试的复杂性和重要性得到了生动的诠释。探险家的勇敢和智慧，正如模糊测试工具在揭示软件系统中潜在问题时所展现的精确和效率。

1.2 模糊测试的历史与发展沿革

模糊测试技术自从问世以来，经历了三个显著的技术发展阶段，反映了该领域技术进步的轨迹。从最初的随机模糊测试，到现在应用最广泛的反馈式模糊测试，模糊测试技术始终是科研界与产业界关注的焦点。时至今日，随着 GPT 大模型带来的技术变革，

模糊测试技术已经迈入智能化时代。

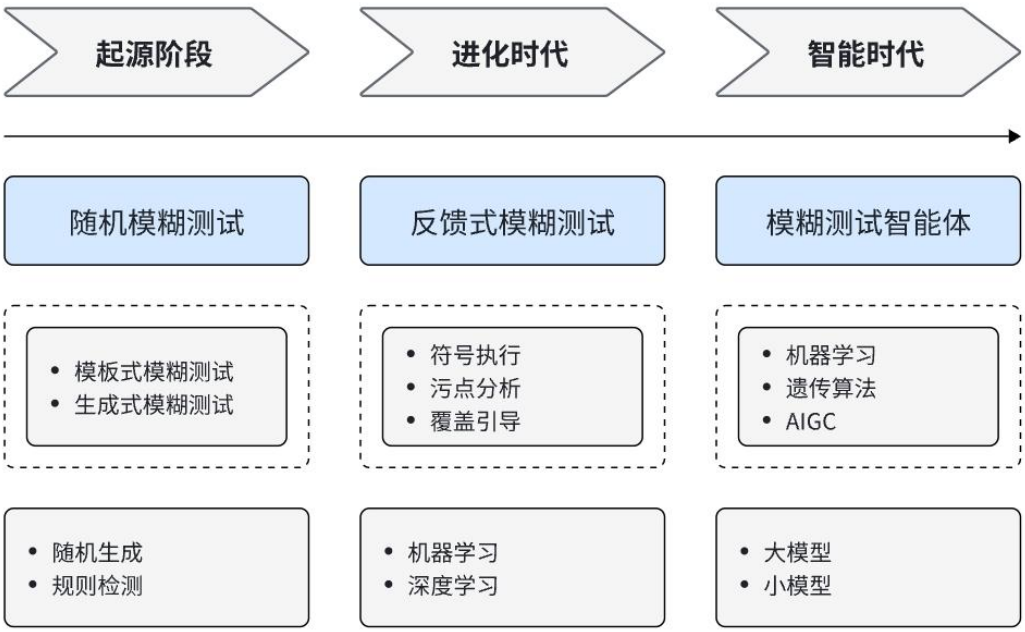

图 2 模糊测试技术发展过程中的三个阶段

1.2.1 起源时代：随机模糊测试

在“模糊测试”这一词语被提出之前，最初采用的概念被称为“猴子测试”[12]。1983 年，Steve Capps 开发了一款名为“Monkey”的应用程序，它通过生成随机的鼠标点击和键盘输入来测试 MacWrite 和 MacPaint 应用程序。测试人员认为这就像一只看不见的猴子在无规律地操作计算机，因此得名“猴子测试”。这种基于随机输入和操作的自动化测试方法成为了模糊测试技术的早期形式。

“模糊测试”这一术语由 Barton Miller 教授在 1988 年提出[47]。一次课程实验

中，Miller 教授尝试通过拨号连接远程登录到一个 Unix 系统时，遭遇了大量干扰噪声，导致依赖外部数据输入的应用程序崩溃。这一经历促使 Miller 教授在威斯康星大学指导学生开展了一个名为“操作系统实用程序的可靠性”的项目。该项目组成员开发一个命令行工具，通过向 Unix 程序发送随机数据来测试其可靠性，并监控程序是否会出现异常或崩溃。这一简单的测试方法揭示了当时 Unix 系统中超过 25%的程序存在崩溃问题，展示了模糊测试在发现软件缺陷方面的强大能力。此后，模糊测试作为一个概念被广泛认知。

早期模糊测试主要采用黑盒测试[7]的方法，测试人员不需要了解目标程序的内部结构和实现机制。他们通过观察程序对输入和输出的处理来评估其性能和稳定性。这个阶段的模糊测试相对简单，缺乏自动化工具和框架的支持，主要依赖生成半随机数据来检测目标程序如何处理异常输入。测试人员使用模板式模糊测试或生成式模糊测试方法，创建各种随机字节序列、字符串或数字作为输入，以此发现程序的异常行为和安全漏洞。

1.2.2 进化时代：反馈式模糊测试

黑盒模糊测试方法因不深入应用程序的内部逻辑和代码结构，测试覆盖度和效率较低。2013 年，AFL（American Fuzzy Lop）[8]框架的问世有效解决了这一问题，标志着反馈式模糊测试时代的到来。

AFL 框架使用覆盖引导的模糊测试方法，其核心在于通过计算和跟踪程序覆盖率来指导测试。在运行时，它会收集覆盖信息，并基于这些数据和遗传变异算法生成新测试用例，以提高测试效率和准确性。这种策略确保测试能够更全面地探索未覆盖的代码区域，从而增强发现软件隐藏错误和未知漏洞的能力。

反馈式模糊测试[2]的引入为安全专家提供了许多全新研究方向。从此，在大型安全产业界会议（如 BlackHat[10]、Defcon[9]等）和网络安全顶尖学术会议（如 USENIX Security[32]、CCS[49]、S&P[49]、NDSS[48]等）中，关于覆盖引导式模糊测试与符号执行、污点分析相结合的讨论日益增多。随着模糊测试效率和覆盖度的显著提升，许多商业化的模糊测试产品也相继推出，模糊测试已逐渐成为软件检测体系中不可或缺的一环。

随着机器学习和深度学习技术的发展，反馈式模糊测试的能力得到了进一步增强。基于机器学习的模糊测试方法能够利用历史测试数据训练模型，以预测哪些输入更可能触发新的代码路径或漏洞。这种智能化的测试生成策略，不仅提高了测试的覆盖度和效率，还能够更有效地发现复杂的安全漏洞。

深度学习技术的应用则进一步推动模糊测试向智能化和自动化发展。通过使用神经网络模型，测试系统可以自动学习和识别复杂的输入模式，高效生成测试用例。同时，深度学习模型还可以帮助分析和分类测试结果，快速定位和修复漏洞。

[2] 反馈式模糊测试（Feedback-based Fuzz Testing）是一种模糊测试技术，通过在测试过程中动态地收集和分析目标系统的反馈信息，来指导和优化后续的测试输入。

1.2.3 智能时代：模糊测试智能体

随着生成式人工智能技术的飞速发展，大语言模型与模糊测试的融合为漏洞检测与修复领域带来了革命性进展。2023 年底，云起无垠首次提出模糊测试智能体[37]架构，标志着模糊测试正式步入“智能体”时代。

在反馈式模糊测试时代，许多开源模糊测试项目和商业化工具已经展示了强大的漏洞挖掘能力。然而，模糊测试工具难以实现自动化漏洞检测，自动化漏洞修复难度大，使其难以大规模普及。在智能体时代，这些瓶颈得以突破。模糊测试智能体方案由代码大模型、模糊测试引擎、静态分析引擎和知识引擎组成，通过智能化手段实现自动化漏洞检测和修复，全面提升检测效率和覆盖率。

其中，代码大模型具备强大的语义理解能力，能够深入分析程序代码的结构和逻辑，并生成测试驱动代码和修复代码。模糊测试引擎结合多种漏洞检测方法，实时检测代码中的异常行为，并结合大模型生成的测试用例进行深入测试。静态分析引擎主要负责语法分析，提供精确的代码分析结果，辅助生成高质量测试驱动代码和修复补丁。知识引擎包含种子库、漏洞库和修复策略库，不仅能够提供高质量的初始测试样例，还可对漏洞的成因分析和归类提供数据支撑。

总体而言，云起无垠提出的模糊测试智能体方案，支持多种编程语言和应用场景，

适用于不同开发环境，具备高效漏洞检测、自动化漏洞修复和大规模测试覆盖的能力，可以显著降低人力成本。该方案不仅解决了模糊测试技术使用门槛高、漏洞修复困难的痛点，还进一步提升了模糊测试的检测效率与覆盖度，实现了真正意义上的规模化漏洞自动化挖掘与修复。

1.3 模糊测试与传统测试技术的差异与优势

软件安全测试是确保软件系统安全性的关键过程，涉及对系统中潜在安全漏洞和弱点的持续检测与评估。企业需定期利用专业的安全测试工具，如静态分析、动态分析、模糊测试、渗透测试和安全漏洞扫描等，来识别并修复安全漏洞，从而有效预防潜在的安全风险。正如图 3 所示，模糊测试在软件测试的各个阶段以及发布与运营阶段中都起着重要作用，不仅能发现已知漏洞，还能挖掘未知漏洞。

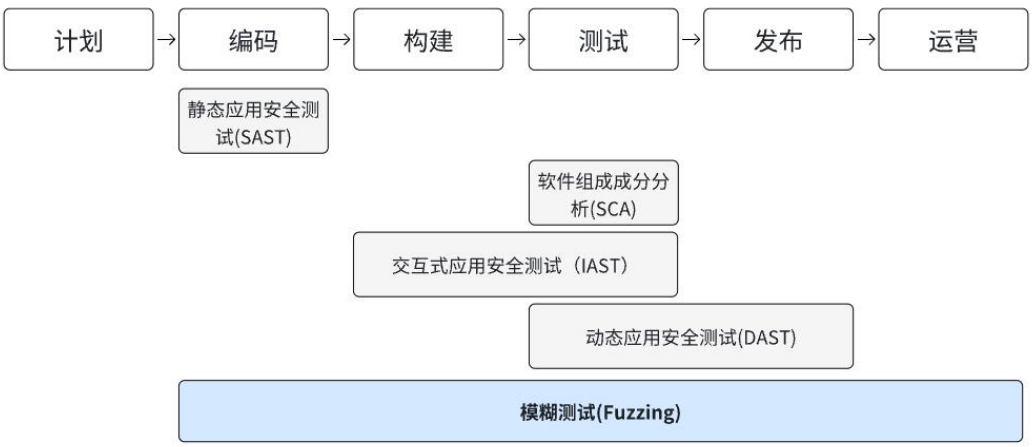

图 3 不同阶段软件安全测试工具的生命周期

具体而言，在软件安全测试领域，模糊测试与传统的基于规则的测试技术有着显著的差异，并展现出独特的优势。以下将详细阐述两者之间的差异以及模糊测试的优势。

1.3.1 模糊测试与传统测试技术的差异

本白皮书从测试方法、测试依赖性、测试覆盖范围、检测能力和测试成本与资源五个方面，对比模糊测试与传统测试技术的差异。

- **测试方法**

- 传统测试技术：主要包括静态分析与动态分析，其中静态分析不需要运行程序，通过检查代码的静态结构（如语法和逻辑）来寻找潜在漏洞，这种方法存在严重的误报干扰，依赖专家经验进行人工降噪，耗时费力。动态测试方法主要以黑盒测试为主，旨在在不查看内部代码的情况下测试软件的外部功能。然而，这类方法依赖已知的攻击模式和漏洞库，主要集中识别已知的安全问题，存在较多的漏报与局限性。

- 模糊测试技术：通过生成大量随机或意外的输入数据，动态地测试软件系统。模糊测试的核心在于模拟攻击者的行为，生成不可预测的输入数据，挑战系统的稳定性和安全性，以发现潜在的安全漏洞和弱点。与静态分析不同，模糊测试在软件运行时进行，能够检测到实际运行时的缺陷和异常行为。

- **测试依赖性**

- 传统测试技术：依赖已知的漏洞模式和预定义的规则，测试重点主要集中在已知的安全问题和预定义的攻击路径。它们往往基于历史数据和已知的攻击模式进行检测。这种方法的有效性高度依赖于漏洞库的完备性和规则的更新速度。

- 模糊测试技术：不依赖已知的漏洞模式，通过半随机变异的输入数据来全面覆盖系统的各类输入情况。模糊测试能够生成各种非预期输入，测试系统在异常情况下的表现，从而发现更多的已知、未知漏洞。其优点在于能够探索未被预见的输入情况，识别出潜在的安全风险。

- **测试覆盖范围**

- 传统测试技术：测试范围有限，主要集中在已知的漏洞和预定义的测试场景。由于依赖于规则和模式，这些方法容易忽略未知的威胁和意外的输入情况。测试的深度和广度受限于规则和模式的设计质量。

- 模糊测试技术：测试范围广泛，通过半随机生成的输入数据，可以覆盖更多的测试场景和潜在漏洞。模糊测试的输入数据不受预定义规则的限制，可以探索系统的各种边界情况和异常行为。这使得模糊测试能够在更广泛的输入空间内发现更多的潜在漏洞。

- **检测能力**

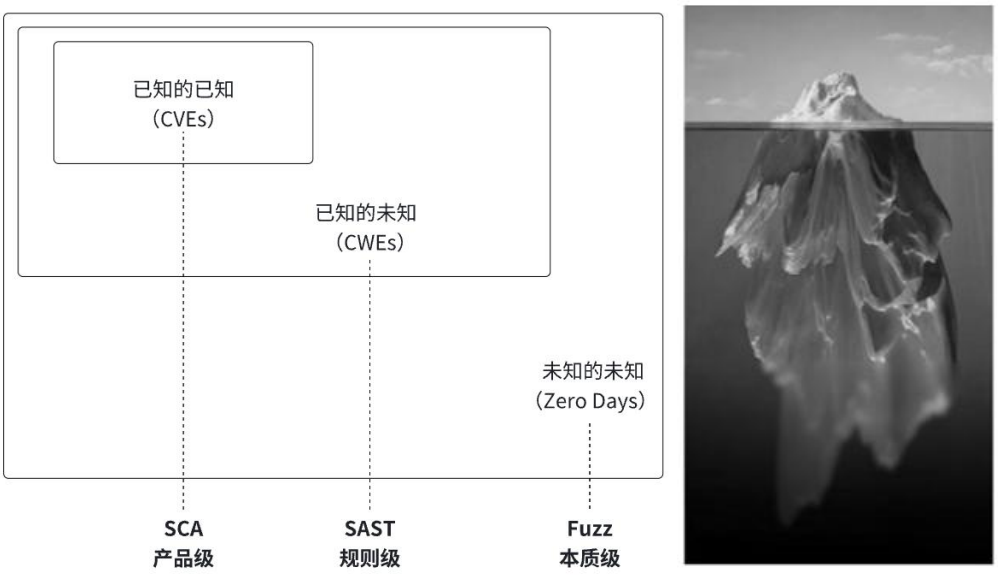

图 4 不同技术漏洞检测能力

- 传统测试技术：传统安全检测技术，如 SAST 和 SCA 在已知漏洞的检测方面表现良好，但对未知漏洞的检测能力较弱。由于依赖于预定义规则和模式，传统测试方法难以识别未被记录的新型攻击手法和漏洞。对于复杂的、未被预见的输入模式，传统方法的有效性大打折扣。

- 模糊测试技术：在未知漏洞的发现方面具有显著优势，能够发现传统测试技术难以检测到的潜在安全问题。模糊测试通过随机和变异输入，能够触发系统中隐藏的漏洞和错误。其动态性使得它能够在实际运行环境中揭示出潜在的安全缺陷。

➢ **测试成本与资源**

- 传统测试技术：需要投入大量的人力和时间进行规则定义、漏洞库维护以及测试执行。这些方法的效果很大程度上依赖于测试人员的专业知识和经验，导致测试成本较高，效率较低。
- 模糊测试技术：能够自动生成测试用例并自动执行测试，大大降低了人力成本和时间投入。特别是在引入大语言模型和安全智能体技术后，模糊测试的自动化水平和智能化程度显著提升，进一步降低了测试成本，提高了测试效率。

1.3.2 模糊测试技术的核心优势

相较于传统的软件检测技术而言，模糊测试凭借其独特的优势，在不断变化的安全挑战中得以广泛应用，已成为现代软件安全测试中不可或缺的一部分。其核心优势包括：

➢ **高覆盖率和测试效率**

现代模糊测试工具（如 AFL）采用覆盖率引导模糊测试方案，通过计算和跟踪程序覆盖率来指导测试过程，能全面探索未测试的代码区域，显著提升测试覆盖率和效率。此外，模糊测试能够覆盖更多的测试场景和边界条件，可以在更广泛的输入空间内发现更多潜在漏洞，从而提供更全面的安全保障。

- **动态行为检测能力**

模糊测试在运行时生成并输入随机数据，实时观察软件的动态行为。这使得模糊测试能发现传统静态应用测试工具（SAST）无法检测到的深层问题，并验证缺陷的可利用性。在实际运行环境中测试，模糊测试能够更准确地评估软件的安全性和稳定性，确保软件在各种使用场景下的可靠性。

- **低误报率**

模糊测试通过生成随机或特制的输入数据并执行目标程序，观察程序的异常行为（如崩溃、内存泄漏等），有效减少误报。与依赖规则和模式的传统测试方法不同，模糊测试基于程序的实际运行行为发现问题，因此检测结果更真实可信。低误报率不仅提高了检测的准确性，还减少了开发团队在筛查和验证漏洞上的负担，节省大量时间和资源。

- **发现未知漏洞**

模糊测试不依赖已知漏洞模式和特征，能发现传统测试方法无法检测到的未知漏洞，特别是 0day 漏洞。通过不断变换输入数据，模糊测试揭示程序在各种非预期输入下的表现，挖掘出潜在的安全漏洞。这说明模糊测试在面对新型和未知威胁时表现出色，

可以显著增强软件安全性。

- **适应复杂系统**

模糊测试在应对复杂软件系统和多样化输入方面表现优异，能够适应各种不同的应用场景和测试需求，包括但不限于传统软件、嵌入式系统及现代分布式应用。这种灵活性和适应性使模糊测试成为应对复杂系统安全挑战的理想工具。

- **自动化与智能化**

随着生成式人工智能技术的飞速发展，模糊测试与大语言模型的结合进一步提升了测试的智能化水平。模糊测试智能体方案通过结合语义理解和语法分析能力，能够深度感知被测程序结构，实现自动化测试驱动生成和自主调度，全面覆盖潜在漏洞。此外，智能体还能进行成因分析和自动化漏洞修复，显著提升了漏洞修复的效率。自动化与智能化的结合不仅提高了测试的精确性和效率，还使得模糊测试能够持续进化，适应不断变化的安全威胁。

二、模糊测试技术解析

模糊测试技术作为提升软件安全性和稳定性的关键手段，已成为现代软件开发和测试流程中不可或缺的一部分。本章将深入探讨模糊测试的基本原理、分类方法、核心技术及其前沿研究热点。首先，介绍模糊测试的技术原理，阐述其工作机制和基础理论。其次，分析模糊测试的不同分类，明确各自特点和适用场景。此外，本章节也将重点讲解模糊测试的关键技术，包括预处理、测试用例生成、调度优化、测试去重和优先级策略以及缺陷检测技术，这些技术对于提高模糊测试效率和覆盖率至关重要。最后，探讨模糊测试的前沿研究，包括大模型驱动的智能测试生成、种子变异、优化技术以及代码修复生成技术，以全面了解当前技术现状和未来研究方向。

2.1 模糊测试的技术原理

模糊测试技术的原理是向软件系统输入大量随机或半随机生成的数据，以触发潜在的错误、漏洞或异常行为。主要架构由五个核心模块组成（如图 5 所示），每个模块承担着不同的功能和责任，共同确保模糊测试的有效性和高效性：

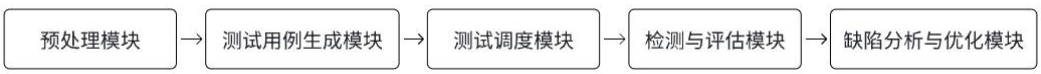

图 5 模糊测试技术架构五大核心模块

- **预处理模块**

负责对待测试的软件系统进行初步分析和准备工作，以确保系统能够接受和处理后续的测试输入。该模块通常会对目标程序进行静态分析，识别程序的入口点、关键函数和潜在的攻击面，为后续的模糊测试奠定基础。预处理模块还可能执行一些初步的输入验证和格式转换工作，以确保输入数据的格式符合目标程序的预期。

- **测试用例生成模块**

这是模糊测试的核心组件之一，主要职责是根据程序监控模块反馈的信息生成大量的测试用例。通过分析程序执行过程中的路径信息、崩溃数据等，该模块能够智能化地调整和优化测试用例的生成，以发现更多潜在的漏洞和问题。例如，现代模糊测试工具如 AFL 利用覆盖引导模糊测试方案，通过实时计算和跟踪程序覆盖率来指导测试用例生成，确保测试全面覆盖未被测试的代码区域。

- **测试调度模块**

在测试用例生成完毕后，负责调用目标程序并将测试用例输入执行。该模块是模糊测试的执行者，确保每个测试用例都能准确送达并执行。测试调度模块还需处理测试执行过程中的各种异常情况，如程序崩溃、超时等，确保测试过程的连续性和完整性。此外，该模块还可能记录测试用例的执行时间、资源消耗等信息，为后续分析提供数据支持。

- **检测与评估模块**

在测试用例执行的同时，负责实时监控目标程序的动态行为，包括种子的执行路径信息、程序状态变化等。该模块将收集到的信息反馈给测试用例生成模块，以指导后续测试用例生成，确保测试过程具有动态适应性。如果在执行过程中发生程序崩溃，该模块会记录崩溃现场的详细信息，包括中断信息、输入信息、内存状态以及寄存器和堆栈信息，并将这些信息传递给崩溃分析模块。通过这种实时监控和反馈机制，检测与评估模块能够有效评估程序的运行状态和安全性。

- **缺陷分析与优化模块**

负责接收程序监控模块传递的程序崩溃信息，并基于崩溃现场的内存地址、寄存器、堆栈信息等进行深入分析。该模块通过分析崩溃原因，判断崩溃的可利用性，以确定是否存在潜在的安全风险。例如，如果崩溃是由于缓冲区溢出或内存泄漏等典型的安全漏洞引起的，崩溃分析模块将评估其对系统安全的影响，并提供修复建议。崩溃分析模块还能生成详细的漏洞报告，帮助开发团队理解问题根源，制定有效的修复方案。

在实际的模糊测试工作中，整个流程可以被细分为图 6 中的关键步骤，以确保测试的全面性和效率：

图 6 模糊测试工作流程

步骤一：识别目标系统。首先，明确被测对象并详细了解其类型。这包括确认目标是客户端还是服务端程序，识别其为二进制可执行文件还是源代码，以及查阅是否存在已知的漏洞记录。获取这些信息对于制定精确的测试计划至关重要，因为它们决定了测试的方向和深度。

步骤二：识别输入。大多数安全漏洞都是由于应用程序未能有效校验或处理用户输入而产生的。成功进行模糊测试的关键在于识别所有可能的输入向量。这些向量可能包括应用程序接受的头部信息、文件名、环境变量、注册表键等。全面的输入识别有助于发现潜在的安全隐患，确保测试的覆盖充分。

步骤三：生成模糊数据。在识别输入向量之后，应根据测试对象的具体特性，制定出模糊测试数据的生成策略。这些测试数据可以是基于现有数据的变种，也可以是程序运行时生成的临时数据。生成多样化的测试用例有助于揭示应用程序在不同输入条件下的行为，从而发现隐藏的漏洞。

步骤四：使用模糊数据执行测试。执行测试时，操作可能包括向应用程序发送数据包、打开文件或启动进程等。在模糊测试中，自动化这一过程至关重要。通过自动化工具，可以大规模地生成和执行测试用例，从而提高测试效率和覆盖率。如果缺少自动化，测试过程将无法有效进行，容易遗漏潜在的安全问题。

步骤五：监控系统行为。在模糊测试过程中，实时监控系统行为是必不可少的。这有助于及时发现任何故障或异常。若缺少有效的异常监控，程序的异常崩溃可能会被测试人员忽略，导致无法确定引发异常的具体测试用例。通常，监控过程是自动化的，以应对大量测试用例所带来的高负荷。常见的异常监控方法包括基于调试和插桩技术：

- 基于调试的异常监控机制：这种方法通过在调试模式下运行目标软件，并使用操作系统提供的调试 API 开发专用的异常监测模块。尽管实现较为复杂，但其能够高效地检测到异常情况。
- 基于插桩的异常监控机制：插桩技术[18]通过向程序内添加额外代码（探针）来收集运行时信息，例如方法调用、参数值和返回值。这些信息有助于捕获程序执行的动态上下文。插桩可以是静态的（在源代码、中间码或二进制代码中插入）或动态的（在二进制执行文件中插入）。

步骤六：记录缺陷，确定可利用性。在模糊测试中发现潜在故障后，接下来的步骤是验证这些故障是否能够稳定复现。通常通过重放检测实现，即利用数据包重放工具来重现捕获的网络数据包。如果能够成功复现故障，则需要进一步评估该缺陷的可利用性，即判断其是否能够被恶意利用。

通过上述一系列步骤，模糊测试不仅能够揭示潜在的安全问题，还能评估这些问题

对系统安全的具体影响。这样发现的漏洞可以转化为明确且可实施的安全增强措施，从而提高系统的整体安全性。

2.2 模糊测试的分类

如图 7 所示，根据测试者对被测软件内部结构的了解程度和访问权限的差异，模糊测试可分为黑盒、白盒和灰盒模糊测试。

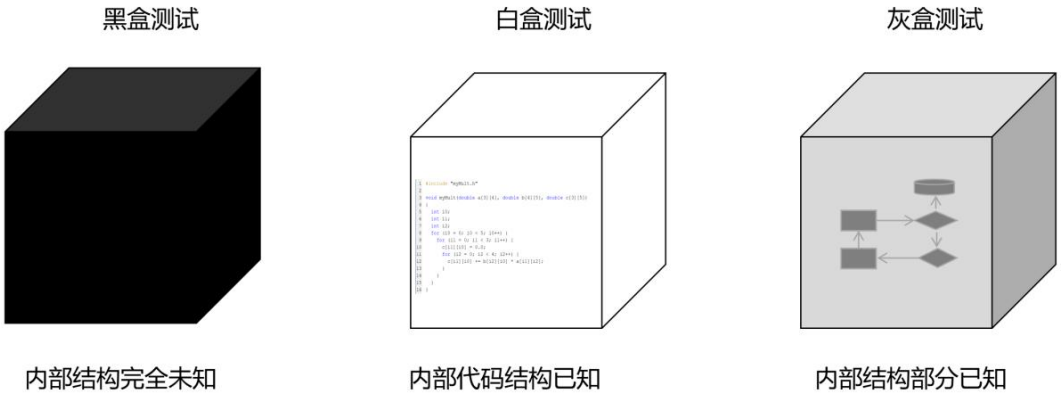

图 7 模糊测试的分类

- **黑盒模糊测试**

黑盒模糊测试是最基础和传统的模糊测试形式。其核心特点是测试者对软件的内部工作原理几乎一无所知。测试完全基于对软件外在行为的观察，无需访问源代码或了解其内部结构。测试者通过向软件输入异常或随机数据，观察其反应，从而识别潜在的漏洞和缺陷。黑盒测试的优势在于其简单性和直接性，能够迅速对软件进行广泛的测试覆盖。然而，由于对内部逻辑的了解有限，这种方法可能无法探测到更深层次的复杂安全

问题。因此，黑盒测试适用于初步安全评估和快速检测明显的漏洞。

- **白盒模糊测试**

白盒模糊测试与黑盒测试相反，测试者可以完全访问软件的内部逻辑和源代码。这种测试方法依赖于详细的应用程序分析技术，如静态代码分析和动态执行跟踪，能够深入探测软件的内部行为。白盒测试能够系统地识别由特定代码路径触发的漏洞，提供更高的测试覆盖率和深入的安全评估。然而，这种方法对测试者的专业知识要求较高，同时也需要较大的资源投入。白盒模糊测试特别适合于需要深入了解和评估软件安全性的场景，如关键基础设施的软件评估和高安全性要求的应用程序测试。

- **灰盒模糊测试**

灰盒模糊测试结合了黑盒和白盒测试的优势，提供了一种平衡的方法。测试者对软件的内部信息有一定了解，但无需完全访问源代码。灰盒测试通常依赖于对部分代码结构的知识，如通过程序插桩收集运行时的反馈信息，以指导测试用例的生成和调整。这种方法既保持了黑盒测试的灵活性和快速性，又通过有限的内部视角提升了测试的效率和有效性。灰盒模糊测试适用于需要在有限资源和时间内进行有效安全评估的场景，是在实际应用中常用的一种方法。

模糊测试作为一种重要的安全测试方法，通过不同的方式（黑盒、白盒、灰盒）为软件安全性提供了不同层次的保护。每种测试方法都有其独特的优势和适用场景。在实

际应用中，选择适当的模糊测试方法，结合具体的需求和资源，能够最大化测试效率和安全评估的覆盖范围。通过综合运用这些方法，能够更好地识别和修复软件中的潜在安全漏洞，提升软件的整体安全性和可靠性。

2.3 模糊测试的关键技术

模糊测试是一种有效的自动化测试技术，其通过生成大量随机或意外的输入数据，来发现软件系统中的潜在漏洞和弱点。为了实现这一目标，模糊测试依赖于多种关键技术，这些技术不仅提高了测试的效率和覆盖率，还增强了对复杂系统和未知漏洞的检测能力。以下是模糊测试的主要关键技术：

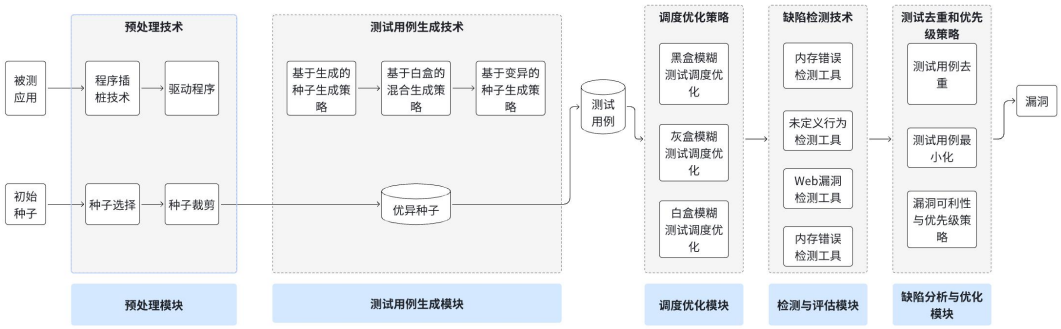

图 8 模糊测试不同阶段关键技术

2.3.1 预处理技术

在开始模糊测试流程之前，许多模糊测试工具需要执行一系列预处理步骤。这些步骤主要涉及对目标程序的插桩，目的是去除潜在的冗余配置，优化种子集，并创建能有效触发应用程序的测试案例。此外，预处理阶段还包括为接下来的输入生成（即输入生

成阶段）做好准备，建立必要的模型。本文将详细探讨以下四种核心的预处理技术：

- **程序插桩技术**

程序插桩技术可以分为静态插桩和动态插桩两种类型。静态插桩在程序运行前的预处理阶段进行，通常作用于源代码或中间代码，并在编译时完成。这种方式的优势在于，处理是在运行前完成的，因此运行时的开销相对较低。如果程序依赖库文件，这些库也需要进行插桩，通常是通过使用相同技术重新编译实现的。除了源代码，也有针对二进制代码的静态插桩工具。相比之下，动态插桩虽然在运行时开销更大，但它可以在程序运行时方便地对动态链接库进行插桩。

当前，有多种著名的动态插桩工具，如 DynInst[13]、DynamoRIO[42]、Pin[51]、Valgrind[52]和 QEMU[53]。模糊测试工具可能支持一种或多种插桩技术。例如，AFL 既可以在源代码级别通过修改编译器实现静态插桩，也可以利用 QEMU 在二进制级别进行动态插桩。在采用动态插桩时，AFL 提供两种选项：一是默认情况下只对目标程序的可执行代码进行插桩；二是通过启用 AFL_INST_LIBS 选项，对目标程序及其所有外部库代码进行插桩。后者虽然能增加代码的覆盖范围，但也可能扩大 AFL 对外部库函数的测试范围。通过这些方式，可以收集包括程序的抽象语法树（AST）[14]、程序调用图（CG）[15]和控制流图（CFG）[16]在内的各种执行信息。插桩技术能够详细跟踪程序的执行路径和覆盖范围，从而极大地提升测试的效果和效率。

- **种子选择技术**

种子选择技术在提升模糊测试效率中扮演着至关重要的角色。在每轮测试开始时，从种子池中选择合适的种子进行变异是一个关键步骤。通过有效的种子选择策略，可以挑选出高质量的种子作为初始测试用例，从而全面探索程序的各个部分。高效的种子选择策略不仅能显著提高代码覆盖率，还能增加发现潜在漏洞的机会。

为了解决这一问题，研究人员开发了多种方法和工具。一个常用的策略是寻找能最大化特定覆盖度量（如节点覆盖率）的最小化种子集合，这个过程被称为计算 minset。例如，如果一个种子集合 C 包括种子文件 s1 和 s2，它们覆盖了不同的程序地址。如果第三个种子文件 s3 能够覆盖 s1 和 s2 的所有地址，并且执行效率相当，那么仅使用 s3 进行测试将是更理想的选择，因为它能在更短的时间内测试更多的程序逻辑。这一理论得到了米勒研究的支持，该研究表明，代码覆盖率每提高 1%，漏洞发现率就会增加 0.92%。

此外，这种策略还可以作为测试过程中的一个反馈更新机制，特别适合那些可以持续添加新种子的模糊测试工具。在实际应用中，模糊测试工具采用了多种覆盖度量标准。例如，AFL 根据分支覆盖率来定义 minset，并使用对数计数器来区分不同的分支。而 Honggfuzz[17]则通过执行指令数、分支数和独立基本块数来衡量覆盖率，这使得测试器可以将执行时间较长的路径考虑进 minset，有助于发现服务拒绝或性能相关的问题。

- **种子修剪技术**

在模糊测试过程中，种子修剪技术被广泛应用，以优化内存使用效率并提高测试吞吐量。种子修剪指的是在保持种子覆盖率不变的前提下，通过减少种子大小来提升测试效率。这个过程通常在主模糊测试循环之前或作为更新过程的一部分进行。

AFL 是一个著名的模糊测试工具，它应用种子修剪技术，通过迭代使用与代码覆盖率相关的工具来修剪种子，确保修剪后的种子保持相同的覆盖范围。研究者 Rebert 等人的研究进一步证实了种子修剪的有效性。他们发现，使用最小尺寸算法选出的小种子，比随机选取的种子更有效地降低了唯一错误的发生率。

特别是在针对 Linux 系统调用处理程序的模糊测试中，MoonShine 工具对 syzkaller[18]进行了扩展，在保持静态分析识别出的调用依赖性的同时，减少种子文件的大小。通过这种优化，模糊测试工具可以更高效地利用内存并提高测试吞吐量，从而更快地发现潜在漏洞。

- **驱动程序技术**

当直接对目标应用进行模糊测试遇到障碍时，开发专用的驱动程序（harness[54]）成为一种行之有效的策略。这种方法通常在模糊测试的初期阶段手动执行，并且只需进行一次。

例如，在对库文件进行模糊测试时，需要设计专用的驱动程序来调用库中的函数。

类似地，为了测试内核，内核模糊测试工具可能会通过模糊测试特定的用户级应用程序，间接实现对内核的测试。通过这些专用驱动程序，可以更精准地控制测试范围和细节，确保测试的覆盖率和深度。

通过采用上述预处理技术，可以大幅提升模糊测试的覆盖范围和效率，从而更有效地发现程序中的潜在漏洞。

2.3.2 测试用例生成技术

生成测试用例是模糊测试过程中一个关键环节，它直接决定了测试的有效性以及是否能够成功发现缺陷。

- **基于模版的生成策略**

通过根据程序输入的预期格式生成各种合法和非法的输入数据来发现程序中的漏洞和错误。在这种测试中，测试数据是通过分析目标软件的输入格式并创建符合该格式的输入数据生成的。这些数据通常通过一些规则、模式或生成器生成，以确保覆盖各种边界情况和输入组合。基于生成的模糊测试的核心思想是生成具有高度多样性的输入，以尽可能地探索程序可能存在的漏洞。

- **基于变异的生成策略**

通过修改现有的有效输入数据来生成大量的测试用例，以发现程序中的漏洞和错误。在这种测试中，测试数据通常来自于真实的应用场景或以前的测试用例，然后通过随机或有目的地修改它们来创建新的输入。这些修改可以包括添加、删除、替换或重新排序数据的部分，以及对数据进行简单的变换或扰动。基于变异的模糊测试的关键思想是通过对现有数据进行变异来尽可能地覆盖程序的不同执行路径，以发现潜在的漏洞和异常行为。

- **基于白盒的生成策略**

基于白盒的模糊测试是一种依赖对程序内部结构深入了解的软件测试策略，旨在生成针对性更强、覆盖更广的测试用例。在白盒模糊测试领域，有三种先进技术特别值得关注：

- 动态符号执行[19]（Concolic Testing）：动态符号执行结合了传统符号执行与具体执行的优势。该技术通过符号值运行程序，并为每个条件分支建立路径公式，然后利用 SMT（Satisfiability Modulo Theories Solver）求解器检查路径公式的满足情况，从而生成具体的输入值。动态符号执行的优点在于降低了符号约束的复杂度并提高了路径覆盖率。然而，由于需要对程序指令进行细致分析，其执行速度较慢。为提高效率，常用策略包括缩小分析范围和结合灰盒模糊测试技术。

- 引导模糊测试[20]（Guided Fuzzing）：引导模糊测试结合了程序分析技术（无论是静态还是动态）与模糊测试，以增强测试用例的生成效能。该方法首先对程序进行分析，捕获关键信息，然后基于这些信息生成更有针对性的测试用例，从而提高测试的效率和效果。

- 待测程序变异与校验绕过技术：待测程序变异与校验绕过模糊测试的主要挑战是如何绕过程序内置的校验和验证机制，这些机制通常在输入数据处理前执行，导致一些测试用例因不满足特定条件而被提前排除，限制了模糊测试的漏洞探索范围。以下三种技术通过智能变异有效绕过内置检查机制，提升测试用例生成效率和漏洞发现能力。TaintScope[21]的“校验和感知模糊测试”利用污点分析技术，识别并修改校验和计算指令，以绕过验证，使修改后的测试用例被程序接受。若程序崩溃，TaintScope 生成匹配正确校验和的测试用例，为未修改程序生成有效数据。Caballero 等提出的“拼接动态符号执行”技术，生成能绕过校验和验证的测试用例，提升模糊测试的有效性和覆盖范围。TFuzz 在灰盒模糊测试中标识非关键检查（NCC）分支，即那些可以修改而不影响程序主逻辑的条件分支。当发现新路径停滞时，TFuzz 修改一个 NCC 并重新进行模糊测试。如修改后的程序版本发现崩溃，TFuzz 尝试通过符号执行在原程序上复现崩溃，以验证其有效性。

2.3.3 调度优化策略

在模糊测试中，调度优化策略负责选择下一轮迭代的配置。其目标是基于现有信息，如发现更多独特漏洞或更全面地覆盖输入集合。调度算法需要在探索（收集更多信息以指导未来决策）和利用（使用当前最有效的配置进行测试）之间取得平衡，这一过程也被称为模糊配置调度问题[22]（Fuzz Configuration Scheduling, FCS）。

- **黑盒模糊测试的调度优化策略**

黑盒模糊测试的调度优化策略[22]通过分析测试结果（如发现的崩溃和错误数量及测试耗时）来提高测试效率和效果。首先，记录和分析每次测试中发现的崩溃和错误数量，以优化输入生成策略。其次，通过分析测试耗时，识别出耗时较短但漏洞发现效果显著的输入配置，优先选择这些配置进行测试。为了增加发现漏洞的可能性，调度算法需要确保输入多样性。通过变异和交叉操作，生成更多不同类型的输入，覆盖更广泛的输入空间。反馈机制可以根据每轮测试结果动态调整输入生成策略，例如增加表现突出的输入类型的生成频率。通过这些优化策略，黑盒模糊测试可以更高效地发现程序中的崩溃和漏洞，提高软件质量和安全性。

- **灰盒模糊测试的调度优化策略**

灰盒模糊测试的调度优化策略[24]通过利用代码覆盖率等信息来优化测试。AFL 框架通过遗传进化算法（EA）领先于该领域。EA 维护配置种群及其适应度值，通过变异和重组产生更优配置。其核心包括确定配置适应度、配置选择方法和应用方式。AFL 优先使用涉及控制流边缘且快速小巧的“favorite”配置。Böhme 等研究者改进了 AFL，创造了 AFLFast。AFLFast[23]引入两个新标准：偏好选择遍历次数少的控制流边缘和执行次数少的路径，并通过优先级改进配置选择机制。AFLFast 采用功率调度策略，显著提升了模糊测试效率。总之，灰盒模糊测试通过结合代码覆盖率和进化算法，显著提升了测试效果和效率。智能选择高潜力配置和改进的调度策略，使其更高效地发现软件漏洞和错误，提高软件质量和安全性。

- **白盒模糊测试的调度优化策略**

白盒模糊测试的调度优化策略[25]利用程序的内部信息，如源代码、数据流和控制流信息，来指导模糊测试过程。其优化包括基于符号执行技术生成路径约束条件，以探索更多未覆盖路径；根据路径重要性和复杂度进行优先级排序，优先测试复杂路径以提高漏洞发现率；结合静态和动态分析信息，优化种子调度和输入生成；以及采用分布式调度策略，将测试任务分配到多个节点并行执行，显著提升测试效率和覆盖率。通过这些方法，白盒模糊测试能更高效地发现软件漏洞和错误，提高软件质量和安全性。

2.3.4 缺陷检测技术

在模糊测试过程中，缺陷检测技术是确保软件安全性的重要手段。以下是几种常见的缺陷检测技术及其应用。

- **内存错误检测工具**

内存错误检测工具主要针对两类安全问题：空间错误和时间错误。空间错误包括缓冲区溢出或下溢，发生在指针解引用越出其目标对象边界时。时间错误则发生在对象销毁后，指针仍尝试访问该内存位置，如使用已释放的内存。典型的内存错误检测工具包括：

- 地址检查器[38]（ASan）：ASan 是一个在编译时插入检测代码的工具，能够快速检测空间和时间错误。虽然使用 ASan 会导致运行时间增加约 73%，但它通过阴影内存技术在内存被解引用前检查其有效性，从而有效识别不安全的内存访问。此外，ASan 的增强版 MEDS 通过创建“红区”（不可访问的内存区域）进一步提升了检测效果，使程序更容易因内存错误而崩溃，从而更快发现问题。
- SoftBound/CETS[55]：该工具在编译时追踪指针的边界和生命周期信息，理论上能够检测所有空间和时间错误。虽然它比 ASan 更全面，但也带来了约 116%的运行时间开销。

- CaVer[56]、TypeSan[57]与 HexType[58]：这些工具专注于检测 C++中的不当类型转换。例如，它们可以识别将基类对象错误地转换为派生类对象的情况。这样的类型转换错误在 C++编程中相当常见且危险，这些工具的使用能够提高程序的安全性。

➢ **未定义行为检测工具**

在 C 语言等编程语言中，未定义行为的具体表现可能因编译器而异，导致程序在不同编译器上表现不一致，进而引发漏洞。为了发现和解决这些问题，以下是几种常见的未定义行为检测工具及其特点：

- 内存检查器[38]（MSan）：MSan 专门用于检测 C 和 C++程序中因使用未初始化内存而导致的未定义行为。它通过阴影内存技术追踪每个位的初始化状态，确保在使用前每个位都已初始化。然而，这种方法会导致大约 150%的性能开销。

- 未定义行为检查器[39]（UBSan）：UBSan 通过在编译时修改程序代码，检测各种未定义行为。这些行为包括使用不对齐指针、零除法、解引用空指针和整数溢出等。通过这种方式，UBSan 可以有效地发现潜在的漏洞并防止它们在运行时引发问题。

- 线程检查器[40]（TSan）：TSan 用于检测数据竞争问题。数据竞争通常发生在两个线程并发访问同一内存位置且至少一个线程执行写操作时，可能导致数据损坏并且难以复现。TSan 通过在编译时修改程序代码，在精确性与性能之间取得平衡，从而有效地检测和防止数据竞争。

- **Web 漏洞检测工具**

Web 漏洞检测工具用于识别和修复 Web 应用中的安全漏洞，以下是几种常见的工具及其特点：

- SSRFuzz[64]：这是一个专门用于检测服务端请求伪造（SSRF）漏洞的工具。SSRFuzz 通过检测 PHP 程序中的服务端请求行为，如网络请求或者文件访问，以系统化识别 SSRF 漏洞相关的 Sink 函数。通过将动态污点分析与模糊测试结合，SSRF 能够快速搜索 Web 应用程序中触发 SSRF sink 的潜在参数列表，并生成多样化的输入，以提升检测 SSRF 漏洞的效果。

- KameleonFuzz[29]：这是一个专门用于检测跨站脚本（XSS）攻击的工具。KameleonFuzz 通过解析真实 Web 浏览器中的测试用例，提取 DOM 树，并将其与已知的 XSS 攻击模式进行比较，以识别成功的 XSS 攻击。通过与模糊测试结合，KameleonFuzz 能够生成多样化的输入，模拟攻击者可能尝试的各种恶意输入，进一步提升检测 XSS 漏洞的效果。

- μ4SQLi[30]：这个工具采用类似技术来检测 SQL 注入攻击。由于从 Web 应用响应中难以可靠地检测 SQL 注入，μ4SQLi 通过数据库代理拦截 Web 应用与数据库之间的通信，检查输入是否触发了有害行为，从而有效识别 SQL 注入攻击。结合模糊测试，μ4SQLi 能够自动生成并测试大量不同的输入，发现那些可能触发 SQL 注入漏

洞的特定输入模式，提高检测的全面性和精度。

- IAST[31]（Interactive Application Security Testing）：IAST 是一种交互式应用安全测试工具，能够实时监测和检测 Web 应用中的安全漏洞。它通过在应用运行时注入检测代码，实时分析应用行为并识别潜在漏洞，如 XSS、SQL 注入和其他常见 Web 攻击。IAST 工具结合了静态和动态分析的优点，提供更全面的漏洞检测能力。通过与模糊测试结合，IAST 工具可以在检测过程中不断生成和测试新的输入数据，发现更多潜在的安全漏洞，增强检测的深度和广度。

➢ **语义差异检测工具**

语义差异检测工具在模糊测试中扮演着发现语义错误的关键角色，通过比较行为相似但不完全相同的程序，这些工具能够揭露潜在的错误。差分测试通过比较多个程序或同一程序在不同输入下的行为，来识别潜在的语义错误，尤其适用于发现复杂的、仅在特定条件下出现的错误。多种模糊测试工具采用差分测试来识别程序间的行为差异，这些差异通常标示着错误。黑盒差分模糊测试不依赖于程序的源代码，而是通过观察输入输出关系来识别错误，具有较强的适应性和广泛的应用前景。结合模糊测试，语义差异检测工具能够自动生成和测试大量输入，模拟多种运行场景，揭示程序在不同条件下的行为差异。模糊测试通过生成随机或特定模式的输入，帮助检测工具发现平常难以捕捉的语义错误，尤其是在应对复杂应用程序时，这种随机性和多样性为漏洞检测提供了强

有力的支持。这种结合不仅提高了测试覆盖率，还使检测结果更加可靠，有助于开发人员及时发现并修复潜在的语义错误，从而提高程序的健壮性和安全性。通过不断优化和改进，语义差异检测工具在保障软件质量和安全性方面发挥着越来越重要的作用。

- HDiff[62]：HDiff 是一个高效的差分计算和应用框架，结合了自然语言分析和差异测试技术，旨在自动化发现 HTTP 软件实现中的语义差异问题。它通过从 RFC 文档中自动提取规范约束，并对不同 HTTP 软件实现进行差异测试。

- WAFManis[63]：该工具可以自动化检测 Web 应用防火墙（WAF）中的协议层通用绕过漏洞。WAF 是当前 Web 系统最主流的安全防护方案。传统 WAF 绕过工具如 SQLMap 利用 WAF 规则集的遗漏，特定 Payload 绕过 WAF，不具备通用性。WAFManis 利用 WAF 与后端 Web 应用框架的协议层面漏洞歧义绕过 WAF，有效协议层 WAF 通用绕过漏洞。结合混合模糊测试，WAFManis 能够利用开源 Web 框架的代码覆盖率，自动生成并测试大量不同的输入，并对 WAF 系统进行黑盒测试，以快速发现那些可能造成 WAF 歧义漏洞的样例。

缺陷检测技术在软件开发过程中起着至关重要的作用。通过有效利用内存错误检测工具、未定义行为检测工具、Web 漏洞检测工具和语义差异检测工具，开发人员可以及时发现并修复潜在的安全漏洞和错误，从而提高软件的安全性和可靠性。随着技术的不断进步，这些工具将继续发展，为软件质量保障提供更强有力的支持。

2.3.5 测试去重和优先级策略

在模糊测试过程中，测试去重和优先级设置是提高测试效率和有效性的重要策略。测试去重能够减少冗余测试用例、节约资源，而优先级设置则有助于集中精力解决最严重的漏洞问题。

- **测试用例去重算法**

在模糊测试中，去重是剔除重复测试用例的重要步骤，旨在形成一个能揭示独特错误的测试用例集合。这不仅节约磁盘空间和资源，还能帮助用户更轻松地识别和分析不同的错误类型。当前，常用的去重技术包括基于栈回溯哈希去重、基于覆盖率去重和基于语义感知去重。基于栈回溯哈希去重通过捕获崩溃时的调用栈回溯和分配栈哈希实现去重，有些实现会生成一个主要哈希和一个次级哈希以提高效率。基于覆盖率去重的工具如 AFL，通过源代码插桩记录测试用例发现的新路径，将这些路径视为独特。基于语义感知去重的 RETracer[59]利用逆向数据流分析技术，从崩溃事件中提取语义信息进行分群，识别触发错误的指令和最高级别函数。

- **测试用例最小化算法**

测试用例最小化是模糊测试中的关键步骤，旨在简化测试用例，生成能触发相同错误的更小、更精简的版本。不同的模糊测试工具实施了各自的最小化算法。例如，

BFF[26]使用专为模糊测试设计的算法，尽量减少与原始种子文件的差异。AFL 通过将字节置零和缩减长度实现简化。Lithium 是一种通用测试用例最小化工具，通过指数级减小尺寸，移除测试用例中的连续行或字节块。此外，还有适用于多种格式的 delta 调试工具和针对特定格式的 CReduce，用于 C/C++文件。尽管专用技术在处理文件类型上有限制，但由于能理解目标语法，通常比通用技术更高效。

- **优先级设置与漏洞可利用性分析**

模糊测试中的优先级排序主要基于漏洞的严重性及其独特性，特别是在内存漏洞的情况下。漏洞的可利用性决定了攻击者能否制定出有效的利用代码，防御方通常优先修补可被利用的漏洞。微软的!exploitable 系统结合启发式算法与污点分析评估崩溃可利用性，引领了这一趋势。随后，GDB[27]的可利用性插件和苹果的 CrashWrangler[28]等基于规则的启发式系统也相继出现，尽管这些系统的准确性尚未经过系统性验证。

2.4 模糊测试产品的前沿研究热点

尽管模糊测试技术在软件安全领域发挥着关键作用，但其应用仍面临使用门槛高、随机性影响检测效率与覆盖度、以及漏洞修复自动化等挑战。为提升检测能力，当前的研究热点包括智能变异策略优化、测试工具的易用性改进、测试覆盖度评估和自动化漏洞修复等。国内以云起无垠为代表的公司通过引入大模型技术，打造出更加易用、高效

的模糊测试智能体产品，显著降低了使用门槛，提高了测试效率和覆盖度。通过这些努力，模糊测试技术将不断发展，为软件安全保障提供更强有力的支持。

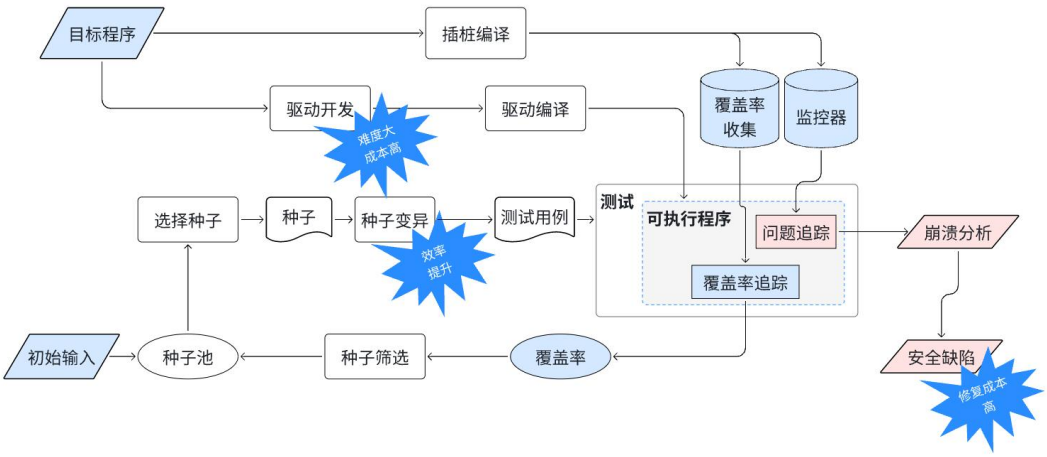

图 9 传统模糊测试典型工作流程与能力瓶颈

2.4.1 基于大模型的测试驱动智能生成技术

高质量的测试驱动对于保障模糊测试的高效、稳定运行至关重要。传统模糊测试方案针对测试驱动的生成方法主要有两类：

- **测试人员手动编写：** 这不仅需要测试人员了解被测目标代码的结构信息，例如函数调用关系、参数类型及参数间的逻辑关系，还需要熟悉测试驱动的编写规范。整个过程对测试人员的技术素养要求高，并需要投入大量时间。例如，为 100 个函数入口编写测试驱动代码可能需要 8 小时的投入。

- **通过静态分析代码结构自动生成测试驱动代码：** 尽管通过静态分析可以生成测

试驱动代码，但由于静态分析只能分析代码结构与语法信息，缺乏对代码语义的理解，导致自动生成的函数入口参数类型有限制或驱动代码质量不高，甚至可能引起缺陷检出的误报，增加人工复核的成本。

有别于传统方案，云起无垠采用“静态分析+大语言模型”的方式，从“输入—生成—反馈”三个阶段，构建了一套完整的测试驱动智能生成框架。

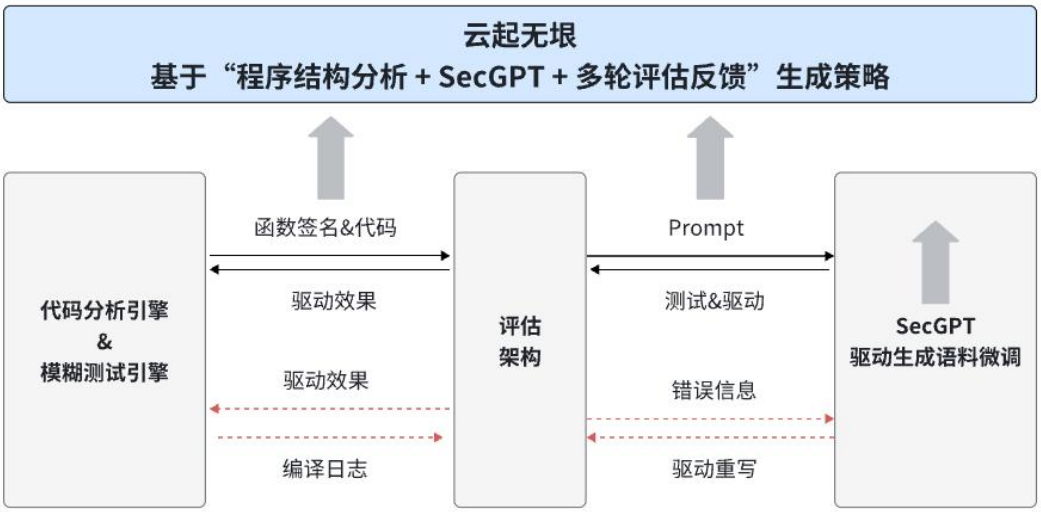

图 10 云起无垠测试驱动智能生成策略

- **输入阶段：** 秉承“输入质量越高输出质量越高”的原则，通过静态分析提取测试函数的详细签名信息，并结合提示词工程构造完整的上下文。详细的签名信息和上下文确保了输入数据的高质量，为后续生成阶段提供了坚实基础。

- **驱动生成阶段：** 云起无垠基于自研的安全大语言模型 SecGPT[41]，提升了模型驱动生成的能力。SecGPT 利用其强大的语义理解和生成能力，能够自动生成高质量的测试驱动代码，克服了传统静态分析方法的局限性。SecGPT 不仅能够识别复杂的函数

调用关系和参数类型，还能够理解代码的语义，生成更准确、更有效的测试驱动。

- **反馈阶段：** 在反馈阶段，模型会进行驱动的正确性验证，并结合验证日志进行持续优化。通过反复的迭代测试和优化，确保生成的测试驱动代码不仅正确，还能高效地覆盖目标代码的各种行为路径。

最终，通过这种“静态分析+大语言模型”的创新方式，云起无垠实现了高质量测试驱动的自动生成，极大降低了测试人员的使用成本，提高了测试效率。这一智能生成框架，不仅简化了测试驱动的生成过程，还显著提升了模糊测试的覆盖率和检测效率，为软件安全保障提供了更强有力的支持。

云起无垠的这一创新框架是模糊测试技术的重大突破，展示了其在软件安全测试领域的领先地位和技术实力。通过持续优化和改进，该框架将进一步推动模糊测试技术的发展，为更广泛的应用场景提供支持。

2.4.2 基于大模型的种子变异和优化技术

模糊测试利用变异策略，随机生成大量非预期输入种子文件，并将这些种子作为目标程序的输入，以观察程序是否出现异常或崩溃，从而检测漏洞。这相当于解决一个复杂的搜索问题，从预先构造的种子输入开始。测试种子的质量直接影响模糊测试对目标

空间探索的效率。然而，现有模糊测试工具在测试效率和代码覆盖率方面仍存在巨大改进空间。

为了解决模糊测试效率低、代码覆盖率不高的问题，云起无垠提出了一个基于大语言模型的创新框架，旨在提升模糊测试的有效性和效率。该框架通过提升种子输入质量和改进种子变异机制，显著提高了对目标程序中潜在错误和崩溃的检测效率，并探索了更多代码和行为路径。

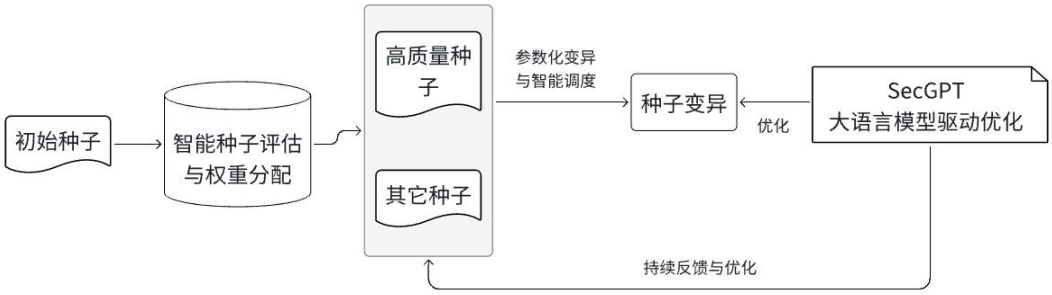

图 11 基于大模型的种子变异和优化技术

- **智能种子评估与权重分配：** 该框架首先评估模糊测试引擎生成种子的质量，并将其与目标执行状态的相关性进行智能分析。基于这种分析，模糊测试种子池被划分为不同权重的队列，从而确保高质量的种子能够优先被使用和变异。这种智能评估和权重分配极大地优化了种子的利用率。

- **参数化变异与智能调度：** 框架结合当前测试路径的参数信息，智能调度种子遗传变异算法。这种调度机制使得种子变异更加有针对性和高效，能够快速探索程序中的

复杂行为路径，显著提高了代码覆盖率和漏洞检测的深度。

- **大语言模型驱动优化：** 云起无垠基于自研的安全大语言模型 SecGPT，全面提升了模型驱动生成的能力。SecGPT 不仅能够生成高质量的种子输入，还能够在变异过程中不断优化种子，提高检测效率。通过持续的反馈和迭代，SecGPT 不断学习和改进，使模糊测试变得更加智能和高效。

- **持续反馈与优化：**在反馈阶段，模型会对生成的测试驱动进行正确性验证，并结合验证日志进行持续优化。经过反复迭代测试，该框架能够在同等时间内覆盖更多程序行为，触发更多潜在漏洞，从而实现更高效、更全面的安全测试。

通过与大模型技术的创新融合，云起无垠提出的这一技术框架不仅优化了测试种子的质量和变异策略，还显著提高了模糊测试的效率和代码覆盖率，从而实现了更深层的智能化。

2.4.3 基于大模型的修复代码生成技术

在软件开发和维护过程中，针对模糊测试检出的安全缺陷，测试人员往往需要进行详细的验证与修复。通常，这一过程不仅需要依赖模糊测试工具提供的详细缺陷信息，例如缺陷类型、触发缺陷的测试用例、函数调用栈与源代码等，还需要测试人员自行进行缺陷分析，手动编写修复补丁，并进行修复后的功能完整性和安全性测试。这个过程

对测试人员的技术要求极高，并且需要投入大量时间和精力，增加了项目的复杂性和成本。

云起无垠针对这些挑战，提出了一套基于大语言模型 SecGPT 的革命性解决方案，构建了一个集“缺陷分析—修复—验证”于一体的自动化框架（如图 12 所示）。这个框架充分利用了模糊测试对缺陷现场信息的全面捕捉能力，以及 SecGPT 对代码语义的深刻理解，极大地优化了缺陷修复的整个流程。

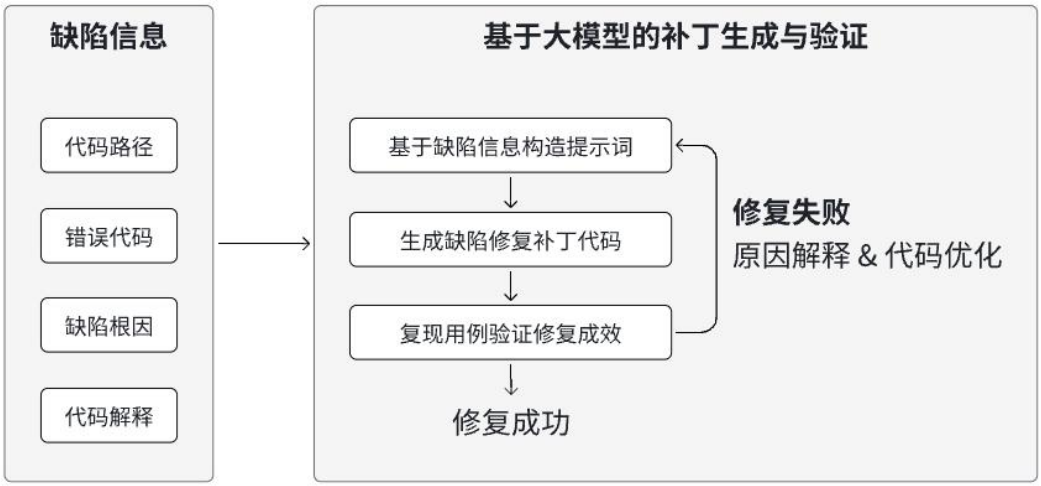

图 12 自动化漏洞修复、验证框架

- **全面的缺陷分析**

该框架利用模糊测试工具捕捉的详细缺陷现场信息，结合 SecGPT 对代码语义的强大理解能力，进行深入的成因分析。SecGPT 不仅能够识别缺陷的类型，还能提供详尽的代码解释，帮助测试人员快速定位问题根源，显著降低了缺陷分析的复杂性和成本。

- **智能修复建议**

基于对缺陷调用栈数据的智能分析，SecGPT 能够对每一个函数进行详细解释，并提供精准的代码级修复建议。这些修复建议不仅考虑了函数的当前状态，还考虑了代码整体的逻辑和安全性要求，确保修复方案的有效性和高质量。

- **自动化验证**

对于修复后的代码，框架提供了一套完整的自动化验证流程。测试人员可以利用系统提供的正确用例和触发缺陷的用例，自动完成功能完整性校验和安全性校验。通过这一自动化过程，修复后的代码在投入使用前得到了充分验证，确保其稳定性和安全性。

云起无垠提出的这一创新框架，利用大语言模型的语义理解能力，为用户提供详细的代码解释和修复建议，提高了补丁的准确性和有效性。该框架不仅降低了对高技术测试人员的依赖，减少了人工复核和修复的成本，还进一步提升了团队的整体效率，实现了智能化的漏洞验证与修复。

三、模糊测试的检测对象

如下图所示，我们根据测试目标的工作层将模糊测试按测试对象分类为应用程序、Web API、数据库、协议、操作系统和固件模糊测试。在提供模糊测试解决方案的厂商中，国外公司如 Code Intelligence 和 ForAllSecure，以及国内厂商如云起无垠[46]、安般科技[44]和水木羽林[45]，都为这些测试对象提供了不同程度的支持。值得注意的是，云起无垠的产品全面覆盖了上述所有测试对象，展示了其在模糊测试领域的强大实力和全面能力。本章节将专注于探讨不同测试对象的模糊测试技术细节。

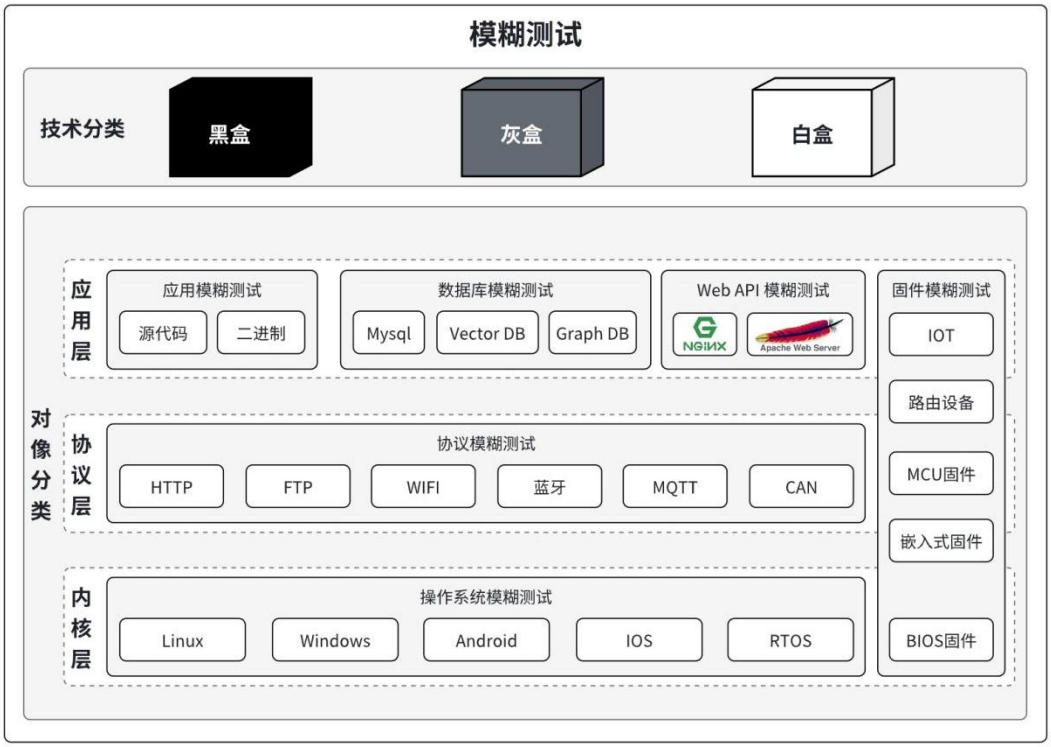

图 13 模糊测试分类

3.1 应用程序模糊测试

在数字化时代，应用程序的快速发展和互联网的广泛普及使其成为黑客攻击的主要目标，造成巨大的财产损失。因此，设计一套具有低误报率、低漏报率、高准确性和高自动化的应用程序漏洞挖掘机制尤为重要，这不仅对缓解应用系统面临的网络攻击风险具有重要的理论意义，也具备显著的实用价值。

应用程序具体是指为用户执行特定任务而设计的软件，能够在桌面计算机、移动设备和网络环境中运行，涵盖了文档处理、数据管理、通讯和娱乐等多种功能。在技术上，应用程序可以分为桌面应用程序、移动应用程序、Web 应用程序和企业应用程序等多种类型。

应用程序模糊测试[2]作为一种通用的自动化测试技术，通过向应用程序输入大量随机或异常数据来发现潜在的安全漏洞和缺陷，在安全测试中发挥着至关重要的作用。根据测试者对源码的获取程度， 应用程序模糊测试可分为源代码模糊测试和二进制模糊测试两大类：

3.1.1 源代码模糊测试

源代码模糊测试是一种旨在深度探索应用程序潜在漏洞的测试方法，适用于测试人

员能够直接访问程序源代码的测试场景。通过此方法，测试人员不仅能评估应用对异常输入的响应，还可以利用对代码逻辑的深层理解来识别和定位那些仅通过表面测试难以发现的复杂安全漏洞。

特别是在对安全要求极高的应用程序开发中，源代码模糊测试能在软件发布前有效提升对未知威胁的防护能力。执行全面的源代码分析与测试能够显著提高应用程序的安全性，减少漏洞的数量和严重程度。

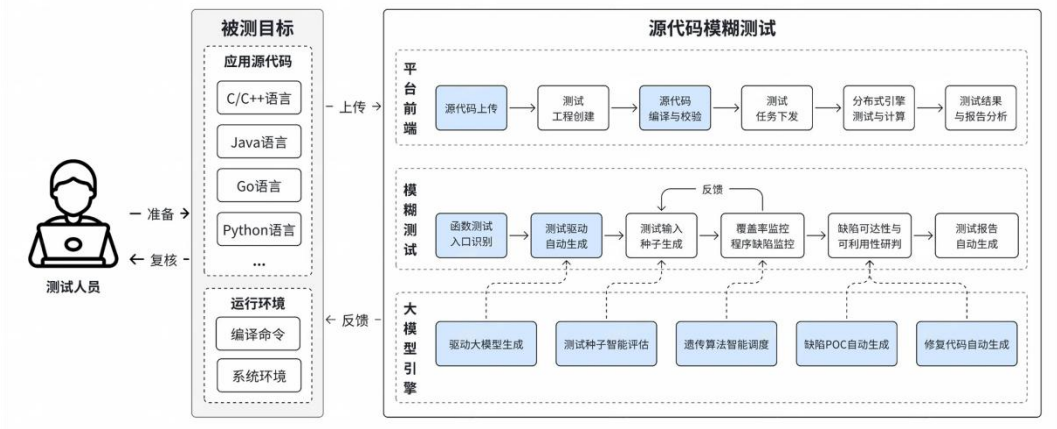

图 14 源代码模糊测试流程图

在执行源代码模糊测试时，测试人员首先需要将被测源代码上传到模糊测试系统进行自动编译，转换为可执行文件或可分析格式。编译完成后，系统自动生成测试任务，并将其下发到模糊测试引擎以便进一步测试。

在模糊测试阶段，系统会识别需要测试的入口点，确定测试范围和目标。根据这些入口点，系统生成测试驱动程序并自动生成各种测试输入，确保尽可能多的代码路径被

覆盖。在测试执行过程中，系统会实时监控测试覆盖率和程序状态，动态调整样例的生成策略。针对检出的异常情况，系统会对缺陷的可达性进行分析，并生成 PoC 对其可利用性进行研判。同时，为更好地帮助用户理解和修复漏洞，系统还将为用户提供漏洞详情、复现脚本以及修复代码。 最后，系统会生成完整的测试报告，包含测试结果、漏洞详情和覆盖率数据等信息。

值得一提的是，对于业内较为领先的模糊测试方案而言，大语言模型的融入是必不可少的。通过大模型的理解、规划与生成能力，可有效提升源代码模糊测试的检测效率与自动化程度，实现更加智能化的漏洞挖掘与修复。

3.1.2 二进制模糊测试

二进制程序的模糊测试[1]是一种专门针对编译后的二进制代码进行的测试方法。由于这种测试方案不依赖于源代码，因此应用范围更加广泛，即使是无法获得源代码的闭源软件也适用于此种测试方法。在现实中，二进制模糊测试被广泛应用于软件安全测试领域，尤其是在需要确保软件稳定性和安全性的关键系统中，如嵌入式设备、工业控制系统和网络设备等。

二进制程序的模糊测试是一种专门针对编译后的二进制代码进行的测试方法，旨在

发现程序中潜在的安全漏洞和稳定性问题。与传统的基于源代码的模糊测试不同，二进制模糊测试不依赖于源代码，因此应用范围更加广泛，尤其适用于那些无法获得源代码的闭源软件。随着现代软件系统的复杂性不断增加，二进制模糊测试变得愈加重要。许多现代软件都是由多个模块和库组成，这些模块和库可能来自不同的开发团队，甚至是第三方供应商。由于无法获得所有模块和库的源代码，二进制模糊测试成为唯一可行的全面测试方法，这种特性使得二进制模糊测试在当今软件安全测试领域中占据了重要地位。

在实际应用中，二进制模糊测试被广泛用于需要确保软件稳定性和安全性的关键系统中。例如，嵌入式设备、工业控制系统和网络设备等，这些系统的稳定性和安全性直接关系到人们的日常生活和工业生产的正常运行。通过对二进制进行模糊测试，可以有效地检测出应用中的漏洞，从而预防潜在的安全威胁。

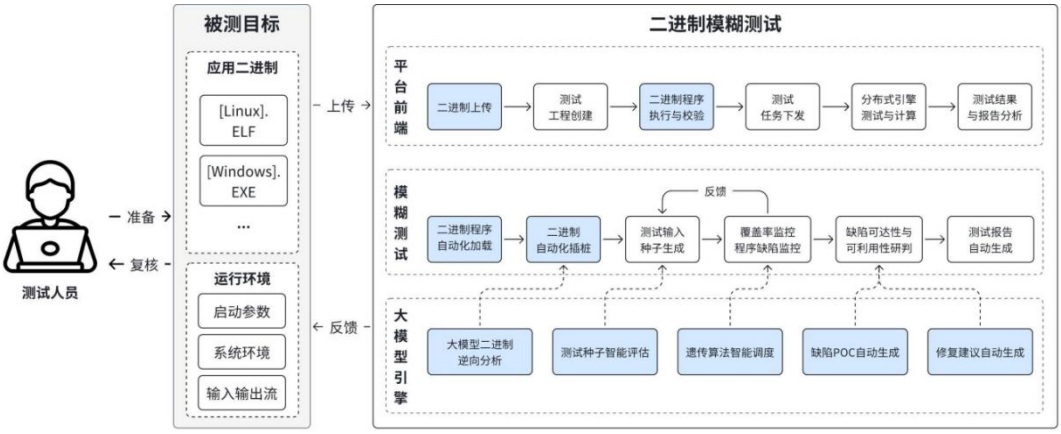

图 15 二进制程序的模糊测试流程图

在执行二进制模糊测试时，测试人员只需将待测二进制文件上传到系统前端，创建测试任务。随后，系统会加载二进制程序，并对其进行自动插桩，以便在执行过程中收集覆盖率和状态信息。在此环节，较为先进的二进制模糊测试方案，还会采用基于大模型的二进制逆向分析，用以获取更多的程序信息。在测试任务开始后，系统自动生成各种测试输入，确保尽可能多的代码路径被覆盖。同时，系统实时监控测试的覆盖率和程序状态，检测异常情况，如崩溃或内存泄漏。针对检出的缺陷，系统会自动验证其可达性与可利用性，最终生成完整的测试报告与修复方案，帮助用户快速排查安全威胁。

总的来说，二进制模糊测试作为一种不依赖源代码的测试方法，为应用安全测试提供了一种强有力的测试方法。它能够广泛应用于各种闭源软件和关键系统中，有效提高软件的安全性和稳定性，预防潜在的安全威胁。在日益复杂的现代软件环境中，二进制模糊测试的重要性不容忽视。

3.2 Web API 模糊测试

在云计算和微服务架构的迅猛发展的时代背景下，REST API[(4)]因其简洁性和易用性，显著提高了软件应用的可扩展性，成为简化软件开发的主流 Web 应用和云服务访问方式。 许多 Web 服务供应商，包括 Google、Twitter 和 Amazon，通过 REST API 提供其服务的访问。然而，API 相关的安全问题日益增多。例如，2020 年，美国的开

放银行应用 Dave 发生了 700 万用户信息泄露事件，泄露的信息随后在黑市出售。2021 年，Facebook 的一个 API 被误用，导致约 5 亿用户信息被泄露。这凸显了确保 API 安全和可靠性的重要性。

传统的手工测试效率低下，且难以实现回归测试。API 模糊测试[3]是一种测试应用程序 API 安全性和稳定性的技术，它通过向 API 发送大量随机或不正确的输入来发现潜在漏洞，例如输入验证问题、缓冲区溢出、注入攻击等。API 模糊测试的核心原理是自动或半自动地将随机或错误的数据注入函数参数中，监控软件表现，以发现潜在问题。API 模糊测试的目的是识别可能被攻击者利用的漏洞或弱点。通过注入意外或格式错误的数据，模糊测试可以引发意外行为或暴露 API 处理输入的不足，从而发现潜在的安全漏洞。

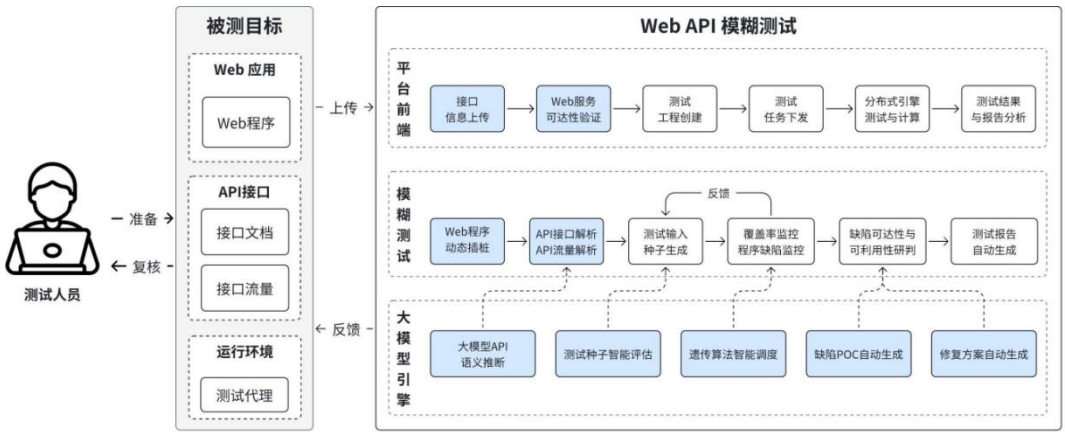

图 16 Web API 模糊测试流程图

在进行 Web API 模糊测试时，测试人员需要准备可运行的待测 Web 应用环境以

及 API 接口信息。然后上传接口信息，让被测程序与模糊测试系统连接，创建测试任务。

在模糊测试阶段，系统首先会对 Web 应用服务进行自动化动态插桩，以便在执行过程中收集覆盖率和状态信息。同时，系统会解析用户上传的 API 接口和流量，以了解应用的运行逻辑和数据流。在这一环节，传统的分析方式大多基于语法规则，而先进的二进制模糊测试方案会引入大模型能力，以补足其对语义分析的能力，从而实现更深层次的 API 解析。随后，系统根据解析结果自动生成各种测试输入，确保尽可能多的 API 路径和数据流被覆盖。在测试执行过程中，系统实时监控测试的覆盖率，确保所有 API 路径和数据流都被充分测试。

模糊测试完成后，系统记录测试过程中发现的所有缺陷现象，并生成详细的缺陷日志以及全面的测试报告和修复方案。

总体而言，Web API 模糊测试在现代软件开发和云服务中的安全保障中起着至关重要的作用。通过自动化 Web API 模糊测试，不仅提高了检测效率，还有效提高了整体的 API 测试深度，避免了人工测试不足带来的风险。未来，随着大模型能力的深入融合，Web API 模糊测试方法将进一步优化，为软件系统的安全性和稳定性提供更有力的支持。

3.3 数据库模糊测试

随着数据在各行各业中成为关键资产，数据库的重要性日益增加。大数据、云计算和物联网技术的发展进一步扩展了数据库的应用范围。然而，存储在数据库中的大量敏感信息也使其成为攻击者的主要目标，导致恶意攻击事件频发。特别是随着数据库系统向多节点分布式方向发展，各类数据库解决方案日益复杂，传统的数据库测试方法已经不足以应对现代数据库系统的检测需求。因此，亟需有效的数据库质量与安全测试方法，以保护敏感信息并维护系统的稳定运行。

数据库模糊测试作为一种关键的安全测试方法，在保护数据库安全和稳定性方面发挥着重要作用，该测试方法采用的核心策略是生成异常的、半随机的非预期输入数据，并将这些数据注入数据库的查询语句、存储过程或相关应用程序中，从而模拟潜在攻击者可能采取的行动。通过分析系统对这些异常输入的处理反应与行为，能够识别出数据库可能面临的安全威胁和漏洞。

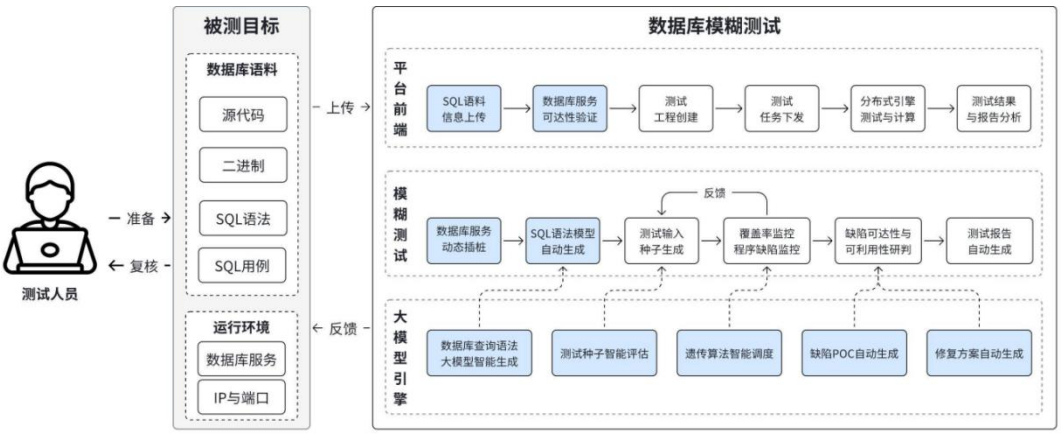

图 17 数据库模糊测试流程图

如图 17 所示，数据库模糊测试的工作流程如下：

首先，测试人员需要准备数据库语料和运行环境。其中，数据库语料包含源代码、二进制文件、SQL 语法和 SQL 用例，而运行环境包括数据库服务以及相关的 IP 和端口信息。

在平台前端，测试人员将 SQL 语料信息上传，并进行数据库服务的可达性验证，确保可以进行测试。然后，创建具体的测试项目并分配测试任务。分布式测试引擎会并行执行测试与计算，生成测试结果和报告分析。

在模糊测试阶段，首先对数据库服务进行动态插桩，插入测试代码。然后自动生成 SQL 语法模型，并创建测试输入种子。在测试过程中，监控覆盖率和程序缺陷，分析缺陷的可达性与利用性，最后自动生成测试报告。大模型引擎通过智能生成数据库查询语法，对测试输入数据进行智能评估，并利用遗传算法优化测试过程。此外，大模型引擎还能自动生成漏洞的 PoC 验证和修复方案，显著提高了测试的智能化水平。

整体来看，先进的数据库模糊测试方案通过高度自动化、分布式测试引擎和智能化的大模型引擎，全面覆盖了数据库测试的各个方面，提高了测试效率和准确性，并生成了详细的测试报告，有助于快速定位和修复漏洞。

3.4 协议模糊测试

协议模糊测试[5]是一种用于测试网络协议安全性和鲁棒性的方法。它通过生成随机或半随机的输入数据，向目标协议的实现发送大量无效、异常或边缘情况的输入，以检测潜在的安全漏洞和错误。

早期网络协议的模糊测试工具主要采用黑盒测试方法，通过随机生成数据并发送到目标服务器进行测试。然而，由于缺乏有效的系统反馈，这种方法在提升代码覆盖率和发现漏洞的效率上存在局限性。为应对这些挑战，研究人员开发了包括基于模型的协议模糊测试技术在内的多种改进方法。这些改进技术的核心在于准确定义程序的输入数据格式和测试用例生成方法。通过结合格式规范等先验知识构建数据模型，该方法能够有效地指导测试用例的生成和执行。

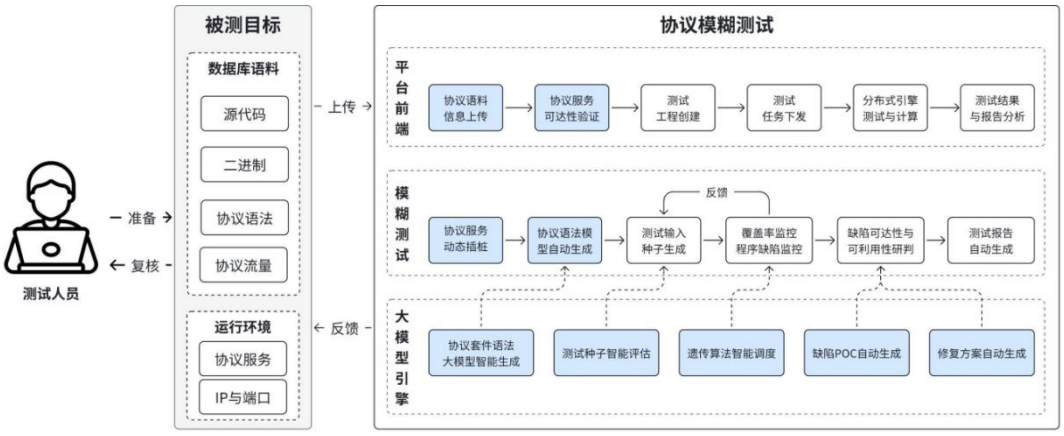

图 18 协议模糊测试流程图

如图 18 所示，协议模糊测试工作流程如下：

首先，测试人员将协议语料信息上传到平台前端。平台验证协议服务的可达性后，

创建测试工程并下发测试任务。模糊测试开始后，首先对协议服务进行动态插桩，并自动生成协议语法模型和测试输入种子。在测试过程中，系统会监控覆盖率和程序缺陷，并对缺陷的可达性与利用性进行研判，自动生成测试报告。其中，大模型引擎能够智能生成协议事件语法，对测试种子进行智能评估，并利用遗传算法优化测试过程。此外，大模型引擎还能自动生成漏洞的 PoC 验证和修复方案，大大提高了测试的智能化水平。

近年来，随着灰盒模糊测试工具 AFL 的崛起，研究人员开始探索将基于程序反馈的灰盒测试方法应用于协议模糊测试中。这种新方法结合了黑盒测试和灰盒测试的优点，不仅能模拟各种输入，还能捕捉到协议的实际反馈，从而显著提高了协议模糊测试的效率。协议模糊测试作为一种在安全测试中广泛使用的方法，适用于多个领域，包括但不限于以下几个方面：

- **网络服务及设备：**协议模糊测试被广泛用于测试网络产品的网络协议解析模块。它通过发送构造的模糊协议数据包至目标应用，以检测在解析协议数据过程中可能存在的缺陷和漏洞。这种测试方式既适用于客户端也适用于服务器模式，其中模糊测试工具能扮演客户端或服务器的角色，用于评估网络服务程序或客户端的安全性。此外，协议模糊测试还可以用于评估部署在网络中的中间设备，如防火墙、路由器和安全网关，检查这些设备在重组和解析数据时的安全漏洞。

- **车联网：**随着汽车与外部通信接口的增加，车辆内部的网络通信变得更加复杂，

协议模糊测试在车联网领域的应用也非常重要。攻击者可以通过与汽车内部的 CAN 总线网络[6]进行通信来对汽车产生影响，而协议模糊测试可以用于测试 CAN 总线的安全性。此外，协议模糊测试还可应用于测试车辆的入侵检测算法，并通过发送大量伪造报文进行拒绝服务攻击的测试。

- **物联网：**随着物联网设备的普及，对提升物联网系统的安全性的需求日益迫切。协议模糊测试可用于测试物联网设备的安全性，例如智能家居设备、传感器和智能城市系统等。通过构造模糊的协议数据并发送给物联网设备，可以发现其中存在的潜在安全漏洞和缺陷，以便及时采取相应的风险应对措施。

- **工业控制：**工业控制系统的安全性极为关键，尤其是面对病毒和木马等网络攻击的风险。协议模糊测试是一种有效的手段，用于评估工控设备的安全性，这包括检测基础通信协议、通用服务协议以及专门的工业控制协议[7]。通过生成并发送含有模糊数据的协议包至工控设备，协议模糊测试能够检测并揭示潜在的安全漏洞，从而使得相关人员能够及时采取防护措施，确保工控系统的安全稳定运行。

3.5 操作系统模糊测试

操作系统作为现代信息技术架构的基础，扮演着至关重要的角色。它不仅在传统的桌面和服务器领域扮演着重要角色，而且在工业控制、航空航天以及智能设备等新兴领

域的重要性日益增强。然而，作为一个广泛应用并且对安全要求极高的基础软件，操作系统存在着重大的安全隐患。

特别是操作系统的核心部分——系统内核，由于其庞大的代码量、快速的更新速度和可能使用的不安全编程语言，成为漏洞频发的高风险区域。截至 2021 年 9 月，Linux 内核的代码行数已达到惊人的 2800 万行，并以每年 30%的速度持续增长。每天都有大量的内核补丁被开发并合并到内核源码中。根据全球公开的漏洞数据库，平均每年都有上百个 Linux 内核的高危漏洞被报告。自 2016 年以来，谷歌已经修复了超过 2000 个深度嵌入 Linux 内核中的漏洞。在 CVE 历史漏洞统计中，排名前 5 的操作系统占据了主导地位，而前 10 名中有 9 个是操作系统。模糊测试作为一种针对操作系统代码和系统功能的检测技术，能够有效地应对操作系统存在的安全威胁问题。

操作系统模糊测试是一种效果显著的漏洞挖掘方法，能够帮助识别操作系统内核、驱动程序、系统服务等多个方面的漏洞。其应用原理是向系统注入大量异常和非预期数据，测试系统处理这些数据时的容错能力和稳定性。

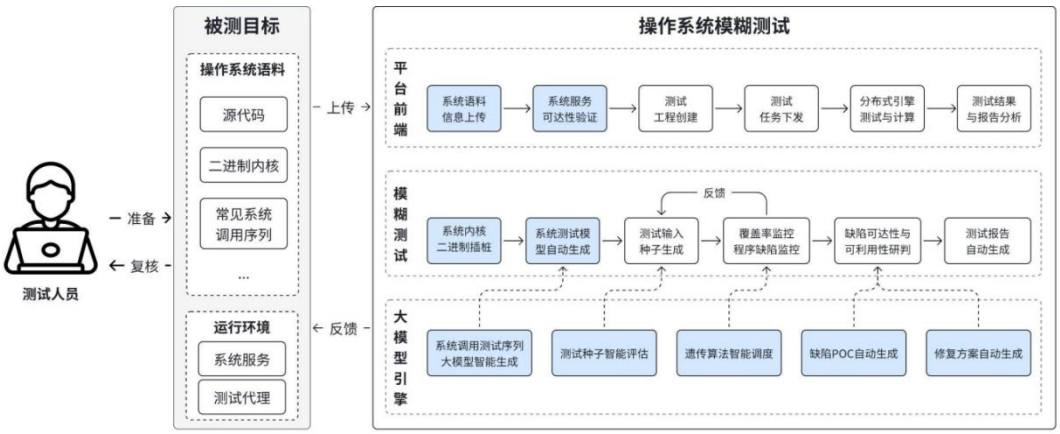

图 19 操作系统模糊测试流程图

通常情况下，如图 19 所示，操作系统模糊测试工作流程如下：

首先，测试人员准备操作系统语料和运行环境。操作系统语料包括如源代码、二进制内核和常见系统调用序列，运行环境包括系统服务和测试代理。在平台前端，测试人员将操作系统语料信息上传，并进行系统服务的可达性验证，确保测试环境准备就绪。随后，创建具体的测试工程并分配测试任务，分布式引擎负责并行执行测试和计算，生成测试结果与报告分析。

在模糊测试阶段，首先对系统内核进行二进制插桩，插入测试代码，然后自动生成系统测试模型，并创建测试输入种子。在测试过程中，系统会监控覆盖率和程序缺陷，分析缺陷的可达性与利用性，最终自动生成测试报告。其中，大模型引擎通过智能生成系统调用测试序列，对测试输入数据进行智能评估，并利用遗传算法优化测试过程。此外，大模型引擎还能自动生成漏洞的 PoC 验证和修复方案，显著提高了测试的智能化水平。整体流程高度自动化和智能化，极大地提高了测试效率和准确性。

3.6 固件模糊测试

与传统软件安全分析不同，由于不统一的开发标准以及封闭的市场环境，嵌入式设备固件的安全问题检测难度更大。具体而言，固件安全分析面临着两大关键挑战，即运行环境架构的多样性以及固件获取的困难性。

通常情况下，嵌入式设备的固件在专为其设计的嵌入式系统中运行。为了对运行中的嵌入式设备进行固件分析，安全研究人员会尝试采用诸如逆向工程、动态调试等不同方法来收集信息。但这些方法往往具有局限性，而且鉴于运行环境架构的多样性，分析人员通常仅能获取到有限的信息。此外，出于安全方面的考虑，许多制造商不但不公开其固件源码，还禁用调试模式，这导致许多基于固件分析的方法在没有物理设备的情况下难以实施。

为了应对嵌入式设备固件安全问题检测所面临的挑战，固件模糊测试方案，通过随机数据生成大量测试用例来发现安全问题，被广泛应用于固件安全检测领域。

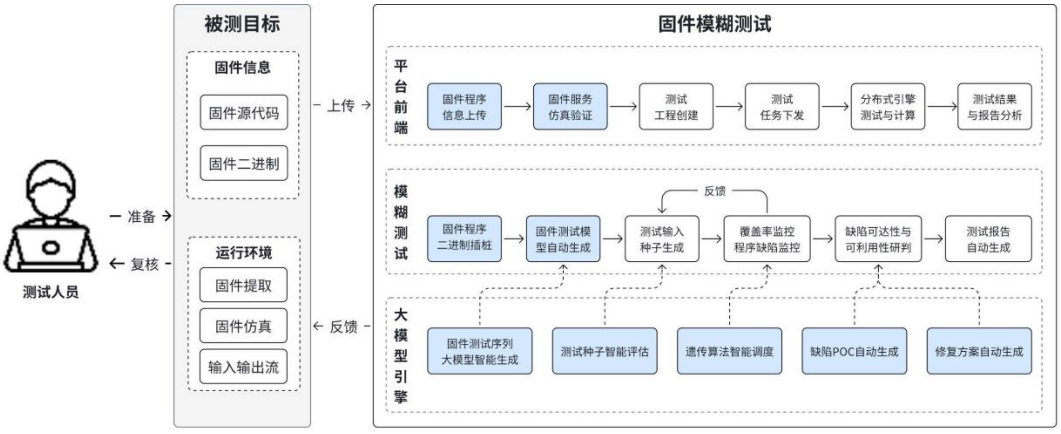

图 20 固件模糊测试流程图

如图 20 所示，固件模糊测试的工作流程涵盖了从准备测试目标到生成详细测试报告的各个环节。首先，测试人员需要进行测试准备，调试被测目标和运行环境。固件信息包括固件的源代码和二进制文件；运行环境包括从设备中提取固件，在仿真环境中运

行固件，以及模拟固件的输入输出行为。

在模糊测试阶段，首先对固件程序的二进制文件进行插桩，插入测试代码。然后自动生成固件测试模型，并创建测试输入种子。在测试过程中，监控覆盖率和程序缺陷，分析缺陷的可达性与利用性。最后，自动生成测试报告，提供详细的测试结果。大模型引擎通过智能生成固件测试序列，对测试输入数据进行智能评估，利用遗传算法优化测试过程。此外，大模型引擎还能自动生成漏洞的 PoC 验证和修复方案，显著提高了测试的智能化水平。

四、模糊测试的需求与应用案例

模糊测试技术凭借其自动化和高覆盖率的特性，在金融、智能网联汽车、工业互联网、信创、信息和通信技术行业（ICT）和人工智能等多个关键领域中得到了广泛的应用和实际应用。

4.1 金融领域

4.1.1 金融领域的模糊测试需求

金融行业在促进经济增长和维持社会稳定方面发挥着至关重要的作用。随着数字化转型的步伐加快，金融服务已广泛迁移至线上平台，包括网上银行、移动支付、在线投资及保险服务等多个方面。这一转型极大地提升了金融服务的便利性和效率，同时为个人和企业提供了前所未有的财务管理能力。然而，这一进程也伴随着新的安全挑战和风险。

首先，金融机构存储大量敏感的个人和企业信息，如账户详情、交易记录和个人身份信息。非法访问这些信息可能导致严重的隐私泄露和财务损失。其次，金融系统的连续性和稳定性对社会经济至关重要；任何服务中断都可能导致客户不满、信任危机，甚至市场动荡。此外，随着技术进步，金融行业的攻击面不断扩大，包括移动应用、在线

服务平台、云计算资源和物联网设备等。新兴技术的引入增加了系统复杂性，为恶意攻击者开辟了新的入侵途径。对金融机构而言，一旦信息系统中的漏洞被恶意利用，可能造成难以挽回财产损失、品牌信誉以及监管方的巨额处罚。例如：2016 年 2 月，孟加拉国央行遭到黑客攻击，导致 8100 万美元被窃取。2017 年，韩国多家银行遭受黑客组织的分布式拒绝服务（DDoS）攻击威胁。同年，攻击者利用 Equifax 网站应用程序的安全漏洞，盗取了大约 1.43 亿美国消费者的个人信息，包括社会安全号码、出生日期、地址等敏感信息。

面对金融领域的复杂安全挑战，模糊测试已成为一项关键的安全测试技术。该技术使金融客户能够在软件的开发和维护早期阶段发现并修复安全漏洞，从而显著降低潜在安全风险。这对于保障金融系统的稳定性和安全性，以及满足严格的合规要求，具有至关重要的作用。

在支付系统和移动支付应用领域，模糊测试能有效评估支付系统的弹性，尤其是对于处理信用卡信息和移动支付交易的应用程序。它帮助识别处理支付请求时可能遇到的各种异常或边缘情况，确保支付系统能够安全、稳定地处理交易，防止数据泄露或未授权的交易发生。

对于在线银行和交易平台，通过对其网页及后端服务进行模糊测试，金融机构能有

效发现可能导致系统崩溃或数据泄露的漏洞。这涉及测试各种输入字段、请求参数以及 API 的异常处理能力，以确保系统能正确处理非法输入，而不会暴露敏感信息或允许未授权访问。

随着金融服务越来越多地依赖于云计算和外部 API，模糊测试也常用于评估这些组件的安全性与稳定性。通过对云服务配置和 API 端点进行模糊测试，可以识别配置错误、认证绕过漏洞和数据处理错误，进而防止可能的数据泄露或服务中断。

此外，金融机构的信息系统中运行着大量第三方组件，这些组件的安全性至关重要。在实际应用中，模糊测试可用于评估金融机构供应链中第三方软件和服务的安全性，确保这些外部组件不会成为安全漏洞的来源。通过定期对这些第三方服务进行模糊测试，金融机构可以要求供应商修复发现的漏洞，从而提升供应链的整体安全性。

4.1.2 金融领域的应用案例

某银行客户高度重视软件开发的安全性，积极采纳“安全左移”的理念，力求在软件开发的早期阶段就能够识别并修复潜在的安全漏洞。为了实现这一目标，该银行实施了 DevSecOps 体系，这一举措显著提升了软件开发的安全性和效率，同时也加快了软件的交付速度。然而，尽管 DevSecOps 的实施为银行带来了积极的变化，但在面对

0day 漏洞等新型未知威胁以及 API 安全风险时，该银行发现传统的应用程序安全测试（AST）工具已不足以应对这些挑战。这些未知威胁的存在可能随时对银行业务系统造成严重影响，而 API 作为现代软件架构的核心，其安全性对整个系统的安全性至关重要。鉴于此，该银行迫切需要引入更加高效和先进的安全检测工具，以进一步提升银行业务系统的安全性和稳定性。

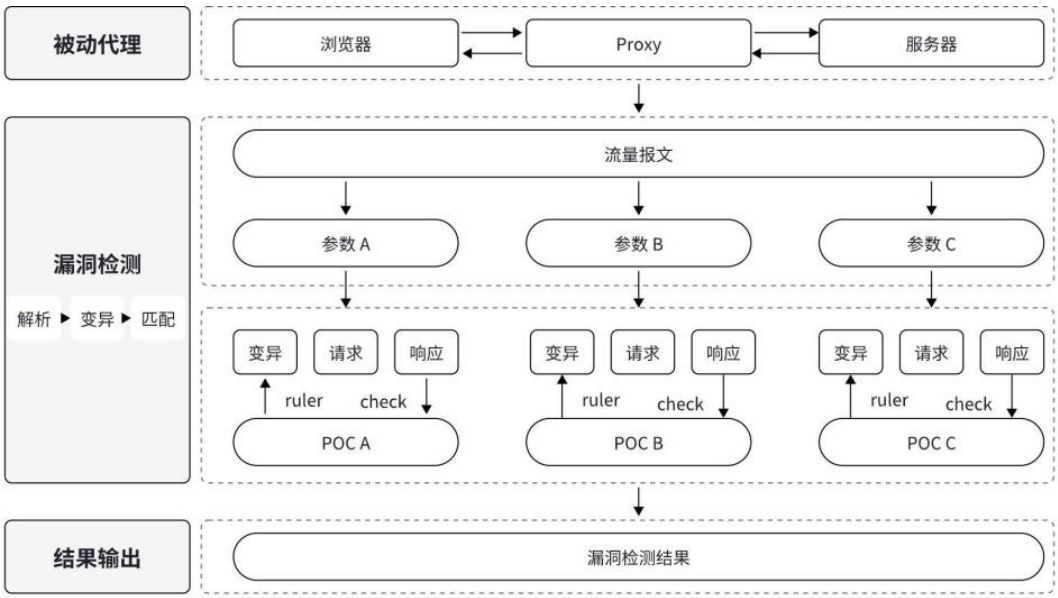

图 21 某银行模糊测试方案部署图

为了有效应对日益复杂的网络安全威胁，该银行客户将模糊测试技术融入软件开发的几个关键阶段：

- 需求阶段：银行基于风险评估的结果，制定了详尽的模糊测试计划。该计划明确了测试的目标、范围、所需资源以及具体的时间节点，确保测试活动能够有序进行。

- 测试阶段：银行在测试环境中部署了自动化的模糊测试工具，对整个系统进行了全面的模糊测试。特别是针对 API 接口，进行了专项模糊测试，确保每个输入点都得到严格的安全审查。通过这些举措，在软件开发过程中，任何潜在的安全漏洞都能被及时发现并修复。

- 部署阶段：银行将模糊测试无缝集成到了持续集成/持续部署（CI/CD）的工作流程中。这样，每当代码更新时，模糊测试都会自动触发，从而确保新代码的安全性得到持续保障。

通过引入模糊测试技术，该银行客户显著降低了安全测试的人力成本，并显著提升了代码的安全质量。模糊测试的持续应用不断识别和修复漏洞，增强了代码的健壮性，进而提高了软件的整体质量。这不仅为银行带来了更强的安全保障，也为客户带来了更优质的用户体验。

具体来说，模糊测试在以下几个方面发挥了关键作用：

- 提高安全检测效率：自动化模糊测试工具能够在短时间内生成大量异常输入，覆盖传统测试难以触及的边缘情况，迅速发现潜在漏洞。

- 增强 API 安全性：通过专项模糊测试，银行能够确保 API 接口在各种异常输

入下的稳定性和安全性，防止数据泄露和未授权访问。

- 持续安全保障：模糊测试与 CI/CD 流程的集成确保了每次代码更新都经过严格的安全测试，降低了新代码引入安全漏洞的风险。

通过这些措施，该银行不仅提升了系统的整体安全性，还增强了客户的信任和满意度，为金融业务的持续发展提供了坚实的基础。模糊测试的成功应用为金融行业树立了典范，展示了先进安全检测技术在现代软件开发中的重要性和有效性。

4.2 智能网联汽车领域

4.2.1 智能网联汽车领域的模糊测试需求

作为人类社会的重要交通工具，汽车的安全性对每个驾驶者和乘客都至关重要。随着软件定义汽车时代的到来，软件正成为汽车的“灵魂”和主要价值点，未来汽车的价值构成中软件占比将达到 60%。2023 年，中国汽车电子软件市场规模已接近 300 亿元。这些智能汽车融合了先进的自动驾驶技术、车联网（V2X）通信功能以及多种智能传感器和系统，为人们提供更加安全、高效和舒适的驾驶体验。然而，这些技术的整合显著增加了汽车系统的复杂性，并引入了前所未有的安全挑战。

近年来，汽车网络安全攻击事件层出不穷。例如，2013 年研究人员通过 OBD-II

（第二代车载自动诊断系统）成功入侵车载 CAN（控制器局域网）总线，从而控制了福特翼虎和丰田普锐斯的方向盘、刹车、油门和仪表盘等汽车组件。2015 年，研究人员证明可以通过软件漏洞无线侵入 Jeep 切诺基系统，导致该公司召回了一大批汽车。现代智能汽车巨头之一的特斯拉也曾被科恩实验室通过远程方式入侵，实现了解锁车辆、打开天窗及转向灯、调节座椅等多项汽车功能的控制。2016 年，特斯拉汽车再次遭受远程攻击，研究人员通过恶意 Wi-Fi 网络成功获取了车辆的访问权限。这一事件引发了业界对智能汽车安全的广泛关注，并促使制造商采取措施提升安全性。2019 年，一些汽车的信息娱乐系统被发现存在漏洞，黑客可以通过 USB 接口入侵车辆系统。

在智能汽车时代，车联网通信（V2X）协议的复杂性为恶意攻击者提供了新的入侵途径，使得安全保障更加复杂和紧迫在这一背景下，模糊测试作为一种先进的软件测试技术，成为保障智能汽车安全的关键手段。模糊测试通过自动化地发送异常或非预期数据包，可以有效揭示由于协议设计缺陷、实现错误或异常处理机制不足而引发的安全漏洞。它不仅能够模拟多种攻击场景，如拒绝服务攻击和缓冲区溢出，还能发现传统安全测试可能忽略的隐患，从而全面提升汽车系统的安全性。

传统的功能测试，如协议一致性测试等，通常无法评估系统在非预期输入下的行为表现，难以有效保障智能网联汽车的安全目标和设计符合性。在这种背景下，模糊测试

技术因其卓越的漏洞发现能力而显得尤为重要。模糊测试通过向系统输入异常或无效数据，能够模拟各种攻击场景，揭示系统在应对非预期输入时的脆弱性。这种测试方法不仅可以发现传统测试方法无法覆盖的漏洞，还能评估系统的稳健性和抗攻击能力，从而全面提升智能网联汽车的安全性。

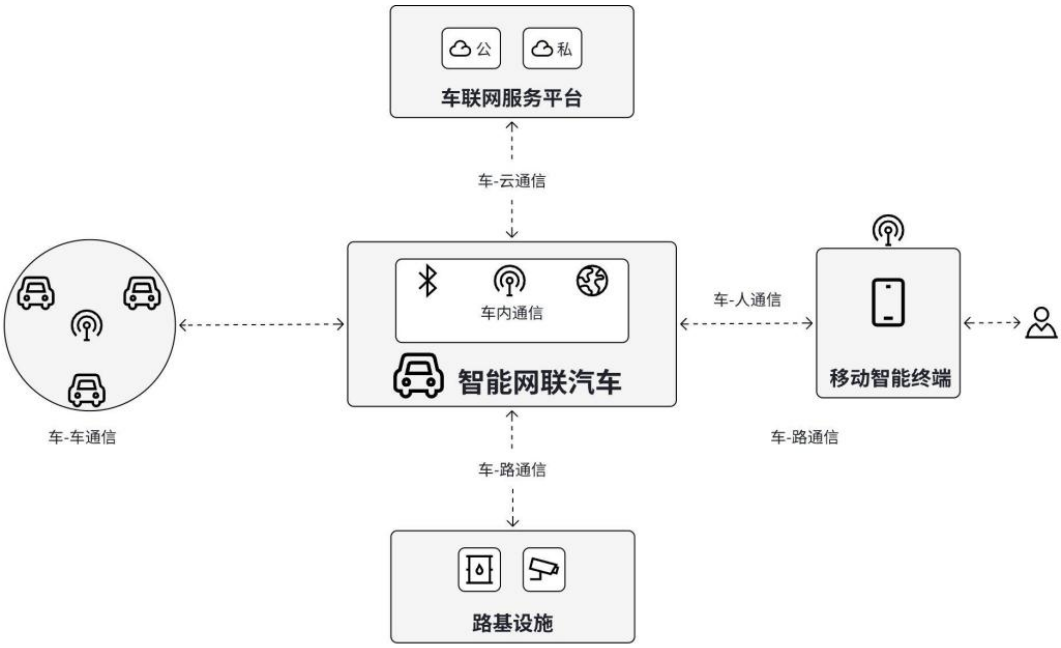

图 22 智能网联汽车通信场景示意

模糊测试能够自动生成并发送异常数据包，模拟攻击者可能采取的多种手段。例如，通过协议模糊测试，可以对 WIFI、蓝牙、NFC、CAN、V2X、充电桩协议等进行全面检测，暴露协议设计和实现中的安全漏洞。此外，在车载系统和应用的测试中，模糊测试同样扮演着重要角色。尤其是在自动驾驶系统中，任何微小的缺陷都可能导致严重后果。通过对摄像头和激光雷达等传感器输入数据进行模糊测试，可以有效检验系统在处理异常情况时的能力，确保系统在面对未知或异常数据时仍能安全、准确地作出决策。

随着智能网联汽车的发展，模糊测试将继续发挥其关键作用，帮助汽车制造商提升系统安全性，确保用户能够安心享受智能网联汽车带来的便利和安全。通过长期进行安全测试和改进，构建一个更安全可靠的车联网生态系统，模糊测试无疑是实现这一目标的重要保障。

4.2.2 智能网联汽车领域的应用案例

随着智能网联汽车复杂性的增加，以及 UN R-155 和 ISO/SAE 21434 等国际法规要求，网络安全测试已成为汽车系统开发和验证过程中不可或缺的一部分。模糊测试作为 ISO/SAE 21434 标准中明确推荐的测试方法之一，能够验证汽车系统的稳健性和网络弹性，并在早期阶段识别潜在弱点。该标准由国际标准化组织（ISO）和美国汽车工程师学会（SAE）联合制定，旨在为汽车行业提供系统化的网络安全管理框架。

ISO/SAE 21434 标准特别强调对软件模块进行安全测试的必要性，并规定产品开发过程中必须进行“威胁代理风险评估”（TARA），以识别、评估、确定优先级和控制网络安全风险。TARA 涉及多种测试方法和流程，包括功能测试、网络安全规范审查、漏洞挖掘和渗透测试等。虽然渗透测试在检测安全漏洞方面非常有效，但由于其高昂的成本和耗时特性，定期对每个模块进行渗透测试并不现实。因此，汽车制造商一直在积极探索利用强大的模糊测试技术来提高测试效率并降低成本。

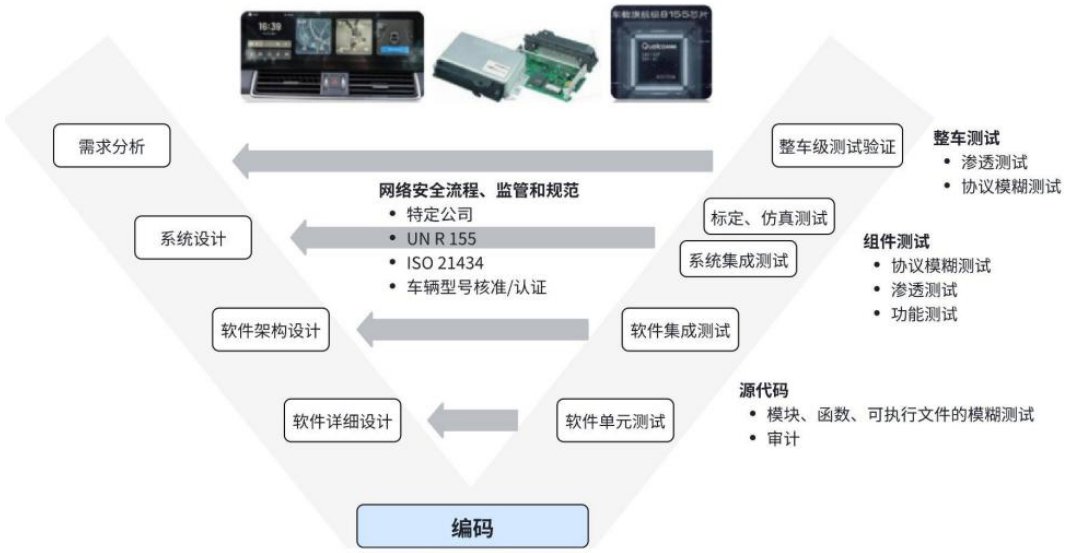

图 23 模糊测试在汽车软件不同测试阶段的应用

其中，智能网联汽车的协议安全性和稳定性成为关注焦点。智能网联汽车协议的特点包括数据格式不统一、传输速率不一致、通信协议复杂以及通信内容不确定等。这些特点对智能网联汽车的安全性能提出了更高要求，同时也给协议安全性测试带来了更大挑战。模糊测试技术因其高效发现协议漏洞和缺陷的特性，在智能网联汽车领域备受关注。

安全测试人员可以通过协议模糊测试对 WIFI、蓝牙、NFC、CAN、V2X、充电桩协议等进行自动化测试。协议模糊测试通过模拟通信节点与被测件建立通信信道，发送对应协议的模糊报文并观察系统响应，从而发现潜在的漏洞和异常行为。车载通信协议通常运行在汽车行业专有的嵌入式设备上，测试时很难提取被测程序或进行插桩，因此基于协议字段规则变异的黑盒模糊测试特别适用于智能网联汽车协议安全测试。测试人员只需获取协议知识并设置合适的监控组件，就能通过分析测试的输入输出获取反馈结果。

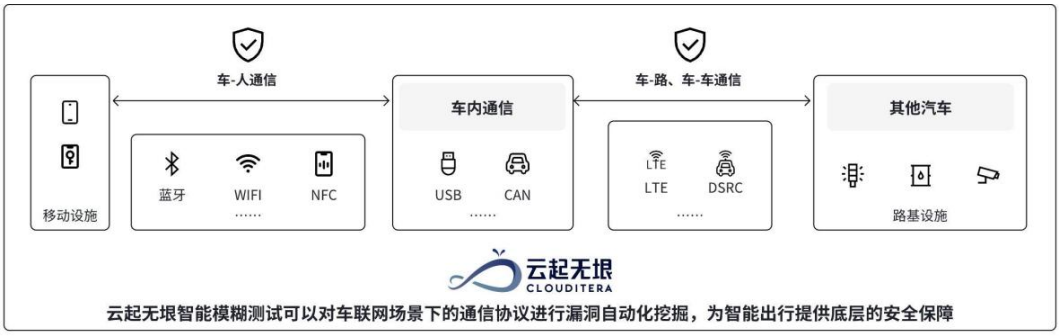

图 24 汽车行业协议模糊测试案例

总体而言，在汽车行业，模糊测试是一种重要的技术手段，能够帮助发现潜在的安全漏洞并评估系统的防御能力。通过向系统中输入异常或无效数据，模糊测试可以模拟攻击场景，发现系统可能存在的漏洞。在智能网联汽车领域，模糊测试可以发现和修复各种安全漏洞，如缓冲区溢出和输入验证错误等。通过模拟各种攻击情景，评估系统安全性，并提供有效的修复措施。

除了发现漏洞，模糊测试还可以评估系统对各种攻击的防御能力。通过模拟真实攻击场景，包括网络攻击和物理层攻击，模糊测试能够测试系统的抗攻击能力，确保智能网联汽车在面对各种安全威胁时保持安全可靠。随着智能网联汽车的发展，模糊测试将成为车联网安全领域不可或缺的工具。开源网安将继续深入研究和应用模糊测试技术，以保护智能网联汽车的安全性，并为其发展贡献力量。通过长期进行安全测试和改进，确保人们能够安心享受智能网联汽车带来的便利，构建一个更安全可靠的车联网生态系统。

4.3 工业互联网领域

4.3.1 工业互联网领域的模糊测试需求

在当今的数字化时代，工业互联网作为连接物理工业系统与计算机网络的桥梁，已成为现代工业发展的核心。它不仅仅是物联网在工业领域的延伸，更是一种全新的产业变革。工业互联网利用先进的信息技术和工业自动化技术，通过传感器、软件和大数据分析等手段，实现了机器设备的智能化管理和运维，极大提高了生产效率和经济效益。这一进步不仅促进了生产流程的优化，也为产品质量控制、能源管理和维护预测提供了前所未有的可能性。

然而，随着工业互联网的快速发展和广泛应用，其安全挑战也日益凸显。一方面，工业互联网的系统通常包含大量的传感器和设备，这些设备的多样性和分布式特性使得整个系统的安全管理变得复杂。另一方面，工业控制系统与信息网络的紧密集成，虽然带来了效率的大幅提升，但也增加了系统面临网络攻击的风险。例如，一旦控制系统遭受到恶意软件攻击，不仅可能导致生产线停滞，严重时甚至会威胁到工人的生命安全及环境安全。因此，确保工业互联网的安全，已经成为了一个亟需解决的问题。

表 1 工业互联网攻击典型事件

网络攻击事件	时间	简述
Code Red	2001	利用 HTTP 漏洞攻击微软服务器，感染 36 万服务器
Blaster	2003	利用 DCOM RPC 漏洞攻击微软旗下操作系统感染百万用广
Stuxnet	2010	利用工业控制系统漏洞攻击伊朗核电站，导致离心机损坏
Duqu	2011	Duqu 木马潜伏于工业控制系统中，收集工业控制系统的信息
Flame	2011	Flame 病毒肆虐中东国家，窃取工业控制信息，并预留后门
Steel Mill	2014	德国钢厂受到 APT 攻击，导致工业生产被迫停运
Power Grid	2015	乌克兰电网被恶意攻击导致全国 140 万用户停电
NotPetya	2017	勒索软件攻击乌克兰
TeamViewer	2021	攻击者渗透美国弗罗里达州的饮用水设施，致使全市供水受限
Colonial	2021	美国原油管道遭遇勒索软件攻击，导致全球原油供应价格飙涨
ZeroX	2021	沙特阿拉伯国家石油公司被黑客攻击损失大量工业设施数据
SolarWinds	2021	微软公司遭到 SolarWinds 黑客攻击，导致 14 家系统被攻破

为应对工业互联网场景中多样且复杂的安全检测需求，模糊测试扮演着重要的角色。以下是模糊测试在几个细分场景中的具体作用：

工业控制系统是工业互联网中最关键的组成部分，包括监控和控制生产过程的各种系统，如 SCADA 系统、分布式控制系统（DCS）和可编程逻辑控制器（PLC）。模糊测试在这里的适用场景包括测试系统的通信协议和软件接口，通过发送畸形或非预期的数据包来检测系统对异常输入的处理能力。这对于防止黑客利用通信协议的漏洞进行攻击至关重要。例如，在某电力公司的 SCADA 系统中，模糊测试揭示了多个协议处理漏洞，从而防止了潜在的网络攻击对电力传输和分配系统造成破坏。

工业互联网场景中应用了大量的物联网设备，用于数据收集和监控。模糊测试可以用于探测这些设备的固件和应用程序中可能导致设备崩溃或被远程控制的漏洞。这不仅限于通信协议，还包括设备的 Web 界面和 API 等潜在的攻击面。例如，在某能源管理系统中，模糊测试发现了物联网传感器固件中的多个安全漏洞，及时修复后大幅提升了整个系统的安全性。

工业数据处理软件（如数据分析、资产管理和维护规划系统）也是模糊测试的重要应用领域。通过测试这些软件系统的输入接口和数据处理逻辑，模糊测试可以发现可能导致数据泄漏或不当处理敏感信息的安全漏洞。例如，在某石油公司的数据分析平台上，模糊测试发现了几处输入验证不足的问题，从而避免了数据泄漏的风险。

随着新技术和应用（如边缘计算和人工智能）在工业环境中的不断涌现，模糊测试的应用场景也在不断扩展，包括测试边缘计算节点的安全性或人工智能模型的鲁棒性。例如，在某制造企业的边缘计算节点上，模糊测试帮助识别了多种潜在的攻击路径，增强了边缘计算的安全防护能力。

现如今，随着新技术和应用，如边缘计算和人工智能在工业环境中的不断涌现，模糊测试的应用场景也在不断扩展，包括测试边缘计算节点的安全性或人工智能模型的鲁棒性。

因此，模糊测试在工业互联网领域的应用极为重要，不仅有助于发现系统中的潜在

安全问题，提升系统的安全性和稳定性，还能降低系统遭受攻击的风险，确保工业互联网系统的安全运行。加强模糊测试技术的研究和实践应用，对于深化工业互联网安全防护工作，推动产业的稳定发展具有重大意义。

4.3.2 工业互联网领域的应用案例

在电力领域，工业互联网的应用尤为广泛，涉及电网监控、能量管理、智能配电等多个方面。某大型电力公司在这一领域通过实施模糊测试技术，取得了显著成果，提高了系统的安全性和稳定性。

该电力公司依靠复杂的电网监控和智能配电系统来管理和控制全国范围内的电力传输和分配。这些系统包括监控和数据采集（SCADA）系统、分布式控制系统（DCS）、可编程逻辑控制器（PLC）、变电站自动化设备、智能电表和远程终端单元（RTU）。随着系统的日益复杂，网络攻击和安全漏洞成为该公司面临的重大挑战。为了预防潜在的网络攻击和确保系统的安全运行，该公司决定引入模糊测试技术，对其电网监控和智能配电系统进行全面检查。

图 25 工业控制系统模糊测试框架

上图展示了模糊测试在工业控制系统中的应用，包括通信协议、外部接口、设备固件和管理软件的全面测试。通过这一框架，该公司确保了其电力系统的全面安全性和稳定性。该公司首先利用模糊测试工具对电网监控系统和智能配电系统中的通信协议进行了详细测试。通过发送畸形数据包和异常输入，模拟各种可能的攻击手段，检查系统在接收到异常数据时的响应情况。同时，对所有外部接口进行了严格的模糊测试，特别是 SCADA 系统、PLC 的通信接口以及智能电表和 RTU，确保它们在面对异常输入时不会崩溃或被攻击者控制。模糊测试还对各类通信协议（如 IEC 61850、DNP3 等）进行了广泛测试，确保这些协议在处理异常数据包时不会导致系统崩溃或功能失效。

此外，模糊测试还对系统中的设备固件和管理软件进行了测试，特别是检查设备固件的更新机制和远程管理功能，确保其免受攻击。通过对输入数据和控制命令进行详细测试，模糊测试发现了一些数据处理和控制逻辑中的漏洞，例如未能正确验证数据合法

性的问题，这些问题可能导致数据泄漏和操作失控。

模糊测试揭示了电网监控和智能配电系统中多个协议处理和设备管理方面的关键漏洞。通过及时修复这些漏洞，该公司显著提升了系统的安全性，防止了可能的网络攻击对电力传输和分配系统造成破坏。模糊测试还帮助公司优化了系统处理逻辑和设备管理，增强了系统的稳健性和可靠性，确保电力供应的连续性和稳定性。通过定期实施模糊测试，该公司能够持续保持高水平的安全防护，有效降低运维过程中因安全漏洞导致的风险和损失。

4.4 军队军工领域

4.4.1 军队军工领域的模糊测试需求

军队和军工在国家安全和防御体系中占据着核心位置。近几十年来，这一行业的信息化水平显著提升，网络信息、人工智能、卫星导航等技术被广泛的应用。这些技术的融合不仅提高了装备的性能，也极大地增强了作战效率和精确度。然而，日趋复杂的信息系统也为国防事业带来了更多安全挑战。

该领域的信息系统和网络架构极其复杂，包含大量的敏感数据和关键技术，如武器设计、制造流程、战术策略和通信协议等。这些信息若落入敌对势力手中，可能会直接威胁到国家安全。因此，保护这些数据免受网络间谍和信息战的威胁是军队和军工企业

面临的重要安全挑战。此外，军事系统和基础设施，包括武器控制系统、监视和侦察系统、通信网络等，已越来越多地与互联网连接或采用数字化技术，这不仅增加了它们面临的网络攻击面，也使得这些关键系统更易受到黑客攻击、恶意软件侵害和拒绝服务攻击等网络威胁的影响。一旦这些系统被破坏或操控，可能会造成无法挽回的损失和后果。

在此类强对抗场景下，模糊测试被证明是一种极其有效的安全测试方法，对于发现和修复潜在的安全漏洞来说至关重要，特别是在防止关键的国防系统和武器装备受到未知的 0day 漏洞攻击方面具有重要的战略意义。模糊测试通过 AI 算法生成大量的随机、畸形或无效输入，并将这些测试数据送入目标程序中，以此来寻找可能存在的未知漏洞和安全问题。在军队军工行业，这种测试不仅适用于软件应用，也同样适用于嵌入式系统、通信设备和其他关键硬件，其典型应用场景包括但不限于：

- **通信系统：**军事通信系统的稳定与可靠性对于保障信息传递和指挥控制至关重要。模糊测试通过挑战这些系统的稳健性——检测面对异常或恶意构造数据时的响应能力，确保通信设备（如无线电、卫星通讯设备）和协议能够有效抵御篡改和拦截。
- **武器控制系统：**这些高度复杂的系统控制着从简单的单兵武器到大型武器平台（如战舰、飞机、导弹系统）的运作。模糊测试帮助识别软件和固件中的漏洞，防止可能导致系统崩溃、误操作或被远程操控的安全问题。

- **物联网（IoT）与无人系统：**随着无人机、自动化监控装备及其它智能设备在军事领域的广泛应用，确保这些设备软件和固件的安全变得尤为关键。模糊测试能够发掘这些系统可能存在的设计和实现上的缺陷，降低被攻击的风险。

- **供应链安全：**军事软件供应链涵盖众多组件和软件供应商，其安全直接影响军队和军工企业的整体安全。模糊测试对供应链中的软件和组件进行安全评估，帮助识别和修复潜在的安全漏洞，确保供应链中的每一环都不会成为安全威胁的潜在源头。

综上所述，国防事业面临的网络安全挑战复杂多样，通过引入模糊测试技术，能够有效发现并修复可能被敌对势力利用的安全漏洞，从而增强整体的安全防护能力。通过对关键系统和设备进行全面而细致的模糊测试，可显著降低安全漏洞的风险，避免潜在的安全威胁，为国防事业筑牢安全屏障。

4.4.2 军队军工领域的应用案例

美国政府问责办公室（GAO）在 2018 年指出，由于美国的武器系统高度依赖软件并且有着极高的网络化程度，面对日益复杂的网络威胁，美国国防部（DoD）在保护国防武器系统方面临着诸多的挑战。那时，用于检测军事武器系统未知漏洞的方法，过度依赖专家小组和传统检测工具，不仅难以实现规模化且自动化程度偏低，无法有效地验证武器系统的安全性。这将造成美国数万亿美元的军事化设施暴露在严重的安全隐患

之中。

为解决迫在眉睫的安全风险，美国国防创新部（DIU）投入 4500 万美元采购商业化模糊测试产品，用于对国防部武器系统进行安全测试。该模糊测试方案针对基于 Linux 的系统进行了优化，且深度结合了符号执行技术，实现了对程序执行路径的全面探索，显著增加了未知漏洞的挖掘能力。该方案的一大特点在于自动化程度高，无需介入复杂的人工操作就能自动化的执行软件漏洞挖掘任务，大幅降低了发现和消除漏洞的成本和时间，减少了对专家小组的依赖。此外，该方案还可直接向用户展示漏洞的触发方式，无需专家人工验证，这为漏洞的理解与修复工作提供了关键的支持。

据悉，美国国防部深度参与了该模糊测试原型平台的开发过程。目前，包括美国空军第 96 网络空间测试小组、空军第 90 网络空间作战中队、海军海上系统司令部(NAVSEA)和美国陆军指挥、控制、通信、计算机、网络、情报、监视和侦察中心(C5ISR)在内的多个机构已大规模使用该模糊测试平台。部分机构也将模糊测试能力整合至敏捷软件开发流程，旨在更加标准化地对武器系统的安全性与稳定性进行深度检测。通过引入模糊测试技术方案，美国国防部成功建立了规模化的未知漏洞自动化挖掘体系，有效降低了重要武器系统中潜在安全威胁。

4.5 信创领域

4.5.1 信创领域的模糊测试需求

在国家政策的推动下，中国信创软件行业的背景是建立一个更加自主可控的软件生态系统，提升国内软件产业的整体技术水平和国际竞争力。这一举措的核心在于减少对外国技术的依赖，特别是在操作系统、数据库、中间件等关键技术领域，从而增强国家的信息安全和经济安全。

随着信创产业的发展，国内对基础软件和应用软件的信创替代和增量采购需求激增。以应用软件市场为例，预计到 2025 年，中国信创应用软件市场的规模将达到 1.5 万亿元，2021-2025 年间的复合增长率约为 35.1%。

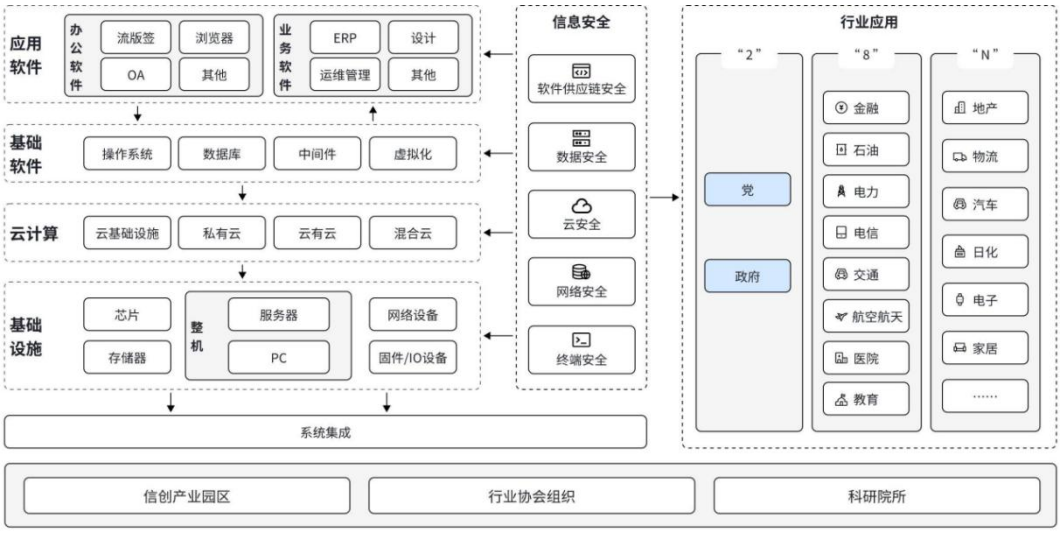

图 26 信创产业链

然而，信创软件的发展面临多项挑战。首先，信创软件开发大量依赖开源软件，但对开源软件的安全漏洞缺乏足够的了解。开源软件安全漏洞的依赖关系、影响范围等信

息不明确，将给信创软件带来极大的安全风险。其次，受到开源社区影响以及国际关系制约，信创软件的底层架构和依赖的第三方组件，可能会面临断供的风险，若未遵循开源许可协议，也会引发知识产权纠纷。此外，为了实现全面自主可控，面对激增自研代码及产品，缺少与之匹配的安全漏洞检测能力与手段，仅依靠人工检测，检测效率无法覆盖代码的增长速度。

为应对这些挑战，使用软件成分分析工具是目前行业中应用较为广泛的解决方案。通过利用软件成分分析提供的软件物料清单分析能力、开源许可协议风险管理能力，以及开源软件已知漏洞检测能力，能够部分解决信创软件的安全漏洞问题。但为了有效的保障信创软件的可信性和安全性，仍有如下几个方面需要改进：

- **缺少信创软件专用漏洞库：**信创软件是近些年的新兴产物，且与通用软件的技术架构与运行环境存在差异，CVE、NVD、CNVD、CNNVD 等通用漏洞库缺少对信创软件的漏洞积累，现有漏洞也不能完全适用于信创软件，需要积累专属于信创软件的漏洞库，为信创软件提供更加准确的漏洞的技术支撑和数据支持。
- **降低信创软件已知漏洞误报率：**基于漏洞库提供的已知漏洞特征，可以快速识别已知漏洞，但另一方面也不可避免的造成了大量的误报信息。我们需要优化漏洞检测机制，补充漏洞定位及复现能力，从而降低信创软件漏洞误报率，提高漏洞处置效率。
- **提升信创软件未知漏洞挖掘能力：**通过已知漏洞特征比对，可以发现已知漏洞，

但对未知漏洞无能为力。并且已知漏洞特征的积累，也是来源与对未知漏洞的发现与分析。所以对未知漏洞的挖掘才是信创软件漏洞检测的底层能力，也是我们当前最为缺失的能力，亟须提升。

面对上述改进需求，模糊测试是最行之有效的技术手段。在漏洞库建设方面，谷歌公司以模糊测试为核心构建开源漏洞库（Open Source Vulnerabilities，简称 OSV）的实践，被验证是一种可行且高效的漏洞库建设方案。模糊测试基于内存级别的细粒度检测能力与动态执行验证能力，使模糊测使的检测缺陷误报率无限趋近于零，且对被发现的软件缺陷，可定位、可复现、可验证。进而，为信创软件检测提供有效的漏洞验证能力，降低误报率，提高漏洞处置效率。其次，模糊测试可通过对软件的运行态进行安全检测，利用 AI 遗传算法辅助测试用例智能变异，基于反馈学习机制，自动生成海量（亿万级）测试用例，触发传统检测手段容易忽略的异常分支，可有效提升信创软件未知漏洞挖掘能力。

随着中国信创软件产业的高速发展，引入模糊测试技术成为解决信创软件安全难题的关键方法。在建立漏洞数据库、降低已知漏洞的误报率、以及增强发现未知漏洞的能力方面，模糊测试技术显示了其显著的效果。这些技术的进步对于增强信创软件的安全性和可靠性提供了重要支持，同时也积极地促进了中国软件产业在全球竞争中的地位提升。

4.5.2 信创领域的应用案例

某数据库厂商作为企业级数据库供应的领军企业，其推出的数据库产品在数据管理和存储方面发挥着核心作用，对企业的日常运营和战略决策具有举足轻重的影响。近年来，随着企业对数据依赖度的增加，该数据库产品的应用范围日益扩大。然而，数据量的急剧增长和数据价值的显著提升也使得该数据库成为了网络攻击者的主要攻击目标。攻击者可能利用 SQL 注入、系统漏洞、弱密码等手段非法侵入数据库系统，从而引发敏感信息的泄露。鉴于此，加强数据库产品的安全性，确保数据的安全性和可靠性显得尤为关键。

为了应对数据库安全性方面的挑战，该数据库厂商采纳了模糊测试技术作为其产品上线前的一项关键安全检测措施。模糊测试，作为一种高效的漏洞检测方法，通过生成大量接近实际应用场景的、语法和语义上正确的 SQL 语句，模拟了潜在攻击者可能采取的行为。这种测试方法的优势在于其能够追踪代码覆盖率，智能地选择那些可能触发数据库管理系统（DBMS）新代码路径的输入进行变异，有效地探索了 DBMS 的状态空间，从而发现那些静态分析或传统测试方法难以发现的漏洞。

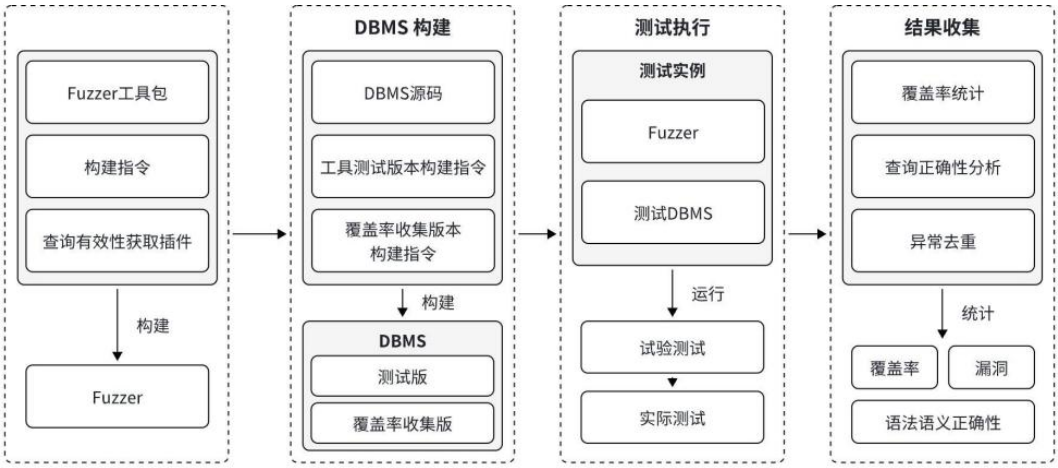

图 27 数据库模糊测试方案图

模糊测试技术的应用带来了多方面的优势：

- **提高安全性：**通过对数据库系统进行全面的测试，模糊测试能及时发现和帮助修复潜在的安全漏洞，减少被攻击者利用的风险。
- **增强稳定性：**在产品上线前通过严格的模糊测试，可以确保数据库系统在各种异常情况下的稳定性和可靠性。
- **优化开发流程：**模糊测试能够自动化地发现漏洞，减少了对人工检测的依赖，提高了漏洞检测的效率，优化了开发和测试流程。
- **减少误报率：**通过智能分析和追踪代码覆盖率，模糊测试能有效降低漏洞检测的误报率，使安全团队能够集中精力处理真正的威胁。
- **支持合规性：**在数据安全法规日益严格的背景下，模糊测试帮助企业符合相关安全标准和法规要求，降低法律风险。

该数据库厂商通过实施模糊测试，显著提高了数据库产品的安全性和稳定性。这种方法不仅有助于及时发现和修复潜在的安全漏洞，还为企业构建了一个更为坚固的数据管理环境，确保了客户数据的安全性和可靠性。因此，模糊测试技术的发展和应用对于提升数据库系统的整体安全性能具有极其重要的价值，是数据库安全策略中不可或缺的一环。

综上所述，模糊测试技术在数据库安全性方面发挥了关键作用，帮助企业应对日益复杂的网络威胁，确保数据库系统的安全性和稳定性。随着模糊测试技术的不断发展和完善，未来将会有更多的企业受益于这一技术，进一步提升其数据库产品的竞争力和安全性。

4.6 检验检测领域

4.6.1 检验检测领域的模糊测试需求

检验检测机构在确保软件和系统安全性方面扮演着至关重要的角色。随着信息技术的快速发展，软件和系统的复杂性日益增加，安全漏洞也随之增多。这使得检验检测机构对高效、精准的检测工具的需求显得尤为迫切。在众多检测技术中，模糊测试因其全面性和高效性，成为应对这些挑战的重要手段。

近年来，检验检测机构面临的检测任务越来越繁重，每年需要检测的软件和系统数

量庞大，传统的人工检测方式已难以应对这种工作量。人工检测不仅耗时耗力，而且容易受到主观因素的影响，导致检测效率低下和准确性不足。在这种情况下，模糊测试的自动化特性尤为重要。模糊测试能够自动生成大量接近实际应用场景的输入数据，并模拟潜在攻击者的行为，从而大幅提升检测效率，减轻检测人员的负担。

然而，检验检测机构对模糊测试的需求不仅仅停留在高效自动化层面。低误报率也是一个关键要求。传统的漏洞检测方法往往会产生大量误报信息，迫使检测人员花费大量时间和精力去筛选和验证真正的漏洞。这不仅浪费了宝贵的资源，还可能导致实际漏洞被忽视。模糊测试通过智能化的漏洞检测机制，能够有效降低误报率，提供更精准的漏洞定位和分析，从而提高漏洞处理效率。

此外，检验检测机构需要面对的是各种不同平台和操作系统的检测需求，包括 Windows、Linux、macOS 以及 Web 应用、移动应用和嵌入式系统等。模糊测试工具的多平台支持能力，确保了不同系统和应用的安全性检测可以一体化进行，避免了因平台差异导致的检测盲区。

在实施模糊测试技术时，检验检测机构还需关注测试流程的优化和专用漏洞库的建立。通过结合传统测试方法和模糊测试技术，可以形成完整的安全测试体系，确保检测的全面性和深度。专用漏洞库的建立则能够积累针对特定类型软件和系统的漏洞信息，提高检测的精准度和效率。

总的来说，模糊测试技术对于检验检测机构提升其检测能力和效率具有重要意义。通过引入和优化模糊测试技术，检验检测机构可以应对日益复杂的检测任务，确保软件和系统的安全性，为客户提供更加可靠的安全保障服务。模糊测试的高自动化和低误报率特性，使其成为检验检测机构在面对繁重检测任务时不可或缺的工具。

4.6.2 检验检测领域的应用案例

某专业安全测评中心以检验检测为核心，打造了完整的公司生态系统，支持各大国企、央企以及相关部委完成信息化建设和数字化转型工作。该中心具备各类国家级认可检测资质，每年为数百家企事业单位提供服务。为了强化产业生态的发展基础，该安全检测机构投入大量资源建立了专门的检验检测实验室，专注于软件源代码的安全分析和健壮性评估，旨在通过专业服务帮助客户提升系统效能。

然而，在代码审计领域，传统静态分析工具的高误报率需要大量人力进行人工确认，直接影响了检测效率和质量。面对繁重的检测任务，有限的人力资源成为业务发展的巨大挑战。为了解决这一问题，该检测机构开始转向模糊测试技术。模糊测试作为一种自动化检测工具，有望减少误报，从而显著提升检测效果并扩大检测范围。该中心的策略是通过模糊测试的高效性和精确性，有效解决代码审计中的关键问题，确保客户系统的安全性和稳健性。通过这种先进的技术手段，该安全测评中心不仅能够提高服务质量，

还能在激烈的市场竞争中保持领先地位。

图 28 检验检测机构模糊测试架构图

该检测机构自采用模糊测试技术以来，已在多个方面实现了显著的效益提升。首先，在节省人工成本方面，模糊测试作为一种自动化测试技术，能够在极短时间内生成并执行大量测试用例。这不仅大幅度减少了手工测试和静态分析所需的人力资源，也显著降低了测试的时间和成本。其次，模糊测试依托于覆盖引导和遗传变异等先进算法，能够有效覆盖更多的程序路径和分支，提高了测试的全面性和准确性，使得检测机构能够快速识别出潜在的漏洞。最后，模糊测试通过向被测系统输入意料之外的数据，突破了传统测试手段的局限，触发了更多异常分支。这不仅有助于检测已知漏洞，更增加了发现 0day 漏洞的可能性，从而为检测机构提供了更深入和全面的安全测试手段。

4.7 信息和通信技术领域

4.7.1 信息和通信技术领域的模糊测试需求

信息和通信技术（ICT）行业是一个快速发展且广泛应用的领域，涵盖计算机硬件、软件、网络、通信设备和服务等多个方面。随着技术的不断进步，ICT 系统的复杂性和安全性要求也在不断增加，因此模糊测试技术在该行业的应用显得尤为重要。

ICT 领域中的产品涉及多种平台和操作系统，如 Windows、Linux、MacOS、iOS 和 Android 等，模糊测试工具需要具备多平台支持能力，以确保在不同环境下的安全性检测。同时，传统的手工漏洞检测方法已经难以满足现代 ICT 系统的需求。模糊测试能够自动生成大量测试用例，模拟各种异常输入，从而高效发现潜在的安全漏洞，显著提高检测效率。

此外，模糊测试工具还需具备降低误报率的能力。传统检测方法往往会产生大量误报，浪费宝贵的资源和时间。通过智能化算法，模糊测试可以减少误报率，提高漏洞检测的准确性。实时监控和详细报告功能也是必不可少的，帮助开发者和测试人员及时了解测试进展和结果，快速响应和修复漏洞。

4.7.2 信息和通信技术领域的应用案例

在信息和通信技术（ICT）领域，模糊测试技术得到了广泛应用，它对提高系统、软件以及硬件的安全性和稳定性发挥着重要作用。微软、Google、Cisco、Oracle 和华为等公司的成功案例，充分展示了模糊测试技术在确保 ICT 产品和服务安全性与可靠性方面所具有的巨大潜力。

微软在其安全开发生命周期（SDL）中广泛采用模糊测试技术，以确保操作系统和应用程序的安全性。通过模糊测试，微软成功发现并修复了 Windows 操作系统中的多个关键漏洞，显著增强了系统的安全性和稳定性。Google 开发的 ClusterFuzz 是一种基于模糊测试的自动化漏洞检测系统，广泛应用于 Chrome 浏览器和其他 Google 产品的测试。通过大规模分布式模糊测试，自动生成并执行数百万个测试用例，发现了大量高危漏洞。

Cisco 运用模糊测试技术对网络设备进行安全测试，成功发现了路由器和交换机中多种协议实现的漏洞，有效防止了可能发生的网络攻击。Oracle 在数据库产品的开发与维护过程中，借助模糊测试发现并修复了多个数据库引擎中的安全漏洞，尤其是在 SQL 语句处理和存储过程调用方面，有力地确保了数据库产品的安全性和可靠性。

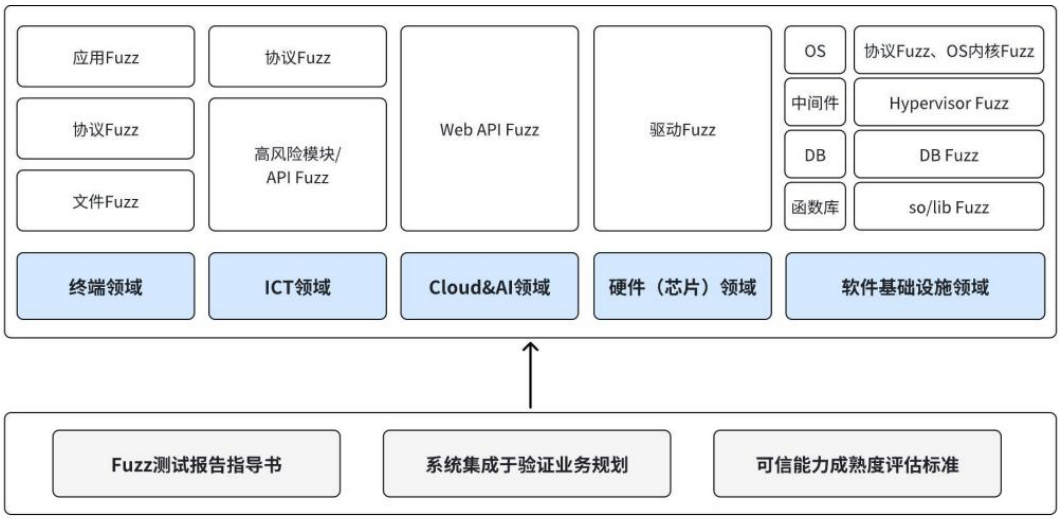

图 29 华为模糊测试应用实践图

在我国，华为作为 ICT 领域的代表厂商，在通信设备和网络解决方案的开发过程中也大范围应用了模糊测试技术。在华为的应用实践里，模糊测试技术一方面被用于对路由器和交换机进行全面的安全评估，确保这些设备在处理大量网络流量时的稳定性与安全性；另一方面，它还帮助华为识别并修复了移动通信基站中的潜在漏洞，进而提升了整个通信网络的安全性和可靠性。

此外，在芯片测试领域，华为同样广泛采用模糊测试技术，极大地提升了固件、驱动程序、输入/输出接口和通信协议的稳定性与安全性。其中，固件作为芯片功能的核心组件，华为借助模糊测试生成并注入异常输入，以此检测固件在处理这些输入时的稳定性，从而显著提升了固件的可靠性。在驱动程序测试方面，华为通过模拟不规范输入数据，成功发现并修复了诸如资源泄漏和竞态条件等潜在缺陷，进而提升了驱动程序的稳定性和兼容性。针对输入/输出接口，华为利用模糊测试工具模拟异常数据和边界条

件，对接口的鲁棒性进行评估，并解决了多个接口在处理异常数据时的稳定性问题。对于通信协议，华为通过模糊测试发现状态机错误和消息解析问题，有力地推动了协议的改进和优化，增强了其在实际应用中的可靠性和安全性。通过系统性地应用模糊测试，华为在发现和修复缺陷方面取得了重大进展，使模糊测试成为芯片验证流程中不可或缺的工具，为产品质量的持续改进奠定了坚实基础。

这些成功案例充分展示了模糊测试技术在 ICT 行业中的有效应用。模糊测试技术通过自动化生成和执行大量测试用例，能够快速识别和修复潜在漏洞，大大减少了手工测试的工作量，同时降低了误报率，为 ICT 行业的安全测试提供了强有力的支持。

4.8 人工智能领域

4.8.1 人工智能领域的模糊测试需求

近年来，人工智能技术的飞速发展不仅极大推动了社会进步和产业转型，更深刻改变了人们的生活和工作方式。AI 技术的广泛应用覆盖了医疗、教育、金融服务、制造业以及自动驾驶技术等众多领域，显著提升了工作效率、经济效益，并降低了生产成本。这些革新不但催生了创新产品和服务，还为人类社会带来了深远的变革。

然而，随着 AI 技术的持续深入发展，其所面临的安全性和可靠性挑战也日渐凸显。AI 系统的复杂性逐步增加，保证其安全运行成了一个重大挑战。尤其是在自动驾驶、

医疗诊断、金融服务和智能制造等关键领域，任何算法缺陷或数据处理错误都可能造成严重后果。为了应对这些挑战，安全专家开始利用模糊测试技术，测试 AI 系统的鲁棒性和异常情况下的处理能力，以保障其安全可靠。

模糊测试，作为一种先进的测试方法，在 AI 行业扮演着至关重要的角色。它主要通过对 AI 模型输入的随机化和异常化处理，探测模型处理异常数据时的行为和反应。这种方法尤其适用于评估 AI 系统在遇到未知或恶意构造的输入时的安全性和稳定性。例如，在自动驾驶领域，通过模糊测试可以识别出处理传感器数据算法在遇到异常输入时可能的错误决策或系统崩溃，及时修复潜在安全隐患。

此外，随着对抗性攻击在 AI 领域的兴起，模糊测试也被用于评估 AI 模型对这类攻击的脆弱性。对抗性攻击旨在通过精心设计的输入欺骗 AI 模型，使其做出错误判断或行为。通过模糊测试，研究人员和开发人员能够发现并加固 AI 模型的弱点，增强其对抗攻击的能力。

模糊测试在提高 AI 系统面对异常、未知或恶意输入时的安全性和鲁棒性方面发挥了显著作用。随着 AI 技术在不同领域的广泛部署，确保 AI 系统的安全、可靠运行显得尤为重要。模糊测试不仅揭露了潜在的安全漏洞和弱点，还为开发者提供了修复指导和改进方向，有效减少了安全事故的风险，保障了用户数据的安全，增强了公众对 AI 技术的信任。

4.8.2 人工智能领域的应用案例

在人工智能（AI）领域，模糊测试技术已被广泛应用于确保系统和模型的安全性和可靠性。以下是一些成功的模糊测试案例，展示了模糊测试在 AI 技术中的重要作用。

- **OpenAI 的 GPT-3 和 GPT-4**

OpenAI 在开发其大规模语言模型 GPT-3 和 GPT-4 时，采用了模糊测试技术来确保模型在处理异常输入时的稳定性和安全性。模糊测试生成大量异常和边界输入，测试模型在不同情况下的反应。例如，通过输入随机生成的非结构化文本或恶意构造的句子，研究人员能够发现模型在这些情况下的潜在问题，并进行针对性的调整和改进。这提高了 GPT 模型在实际应用中的鲁棒性，增强了其处理各种输入时的可靠性。

- **特斯拉的自动驾驶系统**

特斯拉在其自动驾驶技术的开发过程中，广泛应用了模糊测试技术。通过模糊测试，特斯拉生成各种异常传感器输入（如模糊图像、失真雷达信号等），测试其自动驾驶系统在不同情况下的表现。模糊测试帮助特斯拉发现并修复了多个在特定环境条件下可能导致系统失效的漏洞，确保了自动驾驶系统在各种复杂和突发情况下的稳定运行。例如，模糊测试发现了一个在特定光照条件下导致摄像头失效的问题，修复后极大提升了系统的安全性。

- **谷歌的 TensorFlow**

谷歌在其机器学习平台 TensorFlow 的开发过程中，使用模糊测试技术确保其框架的安全性和稳定性。模糊测试生成大量不同类型的输入数据，测试 TensorFlow 在处理这些数据时的表现。通过模糊测试，谷歌发现并修复了多个潜在的安全漏洞和性能问题，确保了 TensorFlow 在各种应用场景中的可靠性。这不仅提高了开发者对 TensorFlow 的信任，也增强了其在工业界的应用。

- **IBM Watson 的医疗诊断系统**

IBM Watson 在其医疗诊断系统的开发过程中，使用模糊测试技术对 AI 模型进行全面测试。通过对医学图像和病历数据进行随机化和异常化处理，模糊测试帮助 IBM 发现了模型在处理某些模糊或异常图像时的误诊问题。通过修复这些问题，IBM Watson 的医疗诊断系统显著提高了诊断的准确性和可靠性，为医疗行业提供了更加安全和有效的 AI 工具。

模糊测试在 AI 领域的成功应用，显著提高了系统和模型的安全性和稳定性。这些案例展示了模糊测试技术在发现和修复潜在漏洞、增强系统鲁棒性方面的重要作用。通过自动化生成和执行大量测试用例，模糊测试能够快速识别和修复 AI 系统中的潜在问题，减少了手工测试的工作量和误报率，为 AI 技术的安全可靠应用提供了有力保障。

4.9 区块链领域

4.9.1 区块链智能合约的模糊测试需求

区块链不可篡改、可追溯、去中心化等特点使其成为了新一代的信任传递工具，在各行各业落地了大量应用。在金融领域，区块链技术通过去中心化的交易平台，降低了交易成本，提高了资本流动的效率。在供应链管理中，区块链确保了商品从生产到消费的每一步都可追溯，增强了供应链的透明度和信任度。在版权保护领域，区块链为创作者提供了一种新的版权认证和收益分配机制，保护了他们的知识产权。在身份验证方面，区块链技术的应用使得个人身份信息更加安全，减少了身份盗用的风险。然而，随着区块链技术的深入应用，其安全性和稳定性的问题也日益凸显。

以太坊是首个支持智能合约的区块链平台，在以太坊中智能合约以合约帐户的形式存在，用于管理存储在区块链 平台中的电子加密货币。截至 2022 年 3 月,由以太坊中智能合约管理的数字资产价值就超过了 3000 亿美元。随着智能合约的广泛应用，智能合约的安全问题也逐渐暴露出来，若智能合约存在漏洞，则攻击者可以利用这些漏洞盗走智能合约管理的电子加密货币或者使其无法取出，使合约的拥有者和参与者蒙受难以估量的经济损失。

智能合约的安全性问题尤其关键，因为它们直接处理资产和交易，任何一个小小的漏洞都可能导致巨大的经济损失。例如以太坊基于智能合约的著名众筹项目 Dao，在 2016 年被不法者利用智能合约的重入漏洞盗取了约 6000 万美元的资金。2020 年 4 月 19 日，去中心化借贷协议 Lendf.Me 遭遇黑客攻击，合约内价值 2500 万美元的资产被洗劫一空，直接原因在于产品本身的可重入问题和特殊的 ERC-777 类型代币 imBTC 组合之后引入的新的安全风险。2024 年 6 月，去中心化交易所 Velocore 遭遇安全漏洞，导致约 680 万美元的 ETH 损失。

智能合约的安全性问题，不仅源于其代码的复杂性，还与其运行环境的不确定性有关。区块链网络的去中心化特性，使得智能合约一旦部署，就难以进行中心化的干预和修正。此外，智能合约的透明性虽然有助于提高信任度，但同时也使得潜在的攻击者能够更容易地发现和利用合约中的漏洞。

为了应对这些挑战，模糊测试技术的应用变得尤为重要。模糊测试通过自动化地生成大量随机或异常的数据输入，模拟各种可能的攻击场景，以此来测试智能合约的安全性和稳定性。模糊测试的优势在于其能够覆盖广泛的测试场景，包括那些开发者可能未曾考虑到的极端情况。通过模糊测试，开发者可以更全面地了解智能合约在面对未知或恶意输入时的行为，从而提前采取措施，提升合约的安全性。

随着区块链技术的不断发展，智能合约的应用场景也在不断扩展。从简单的货币交易到复杂的金融衍生品，从供应链管理到物联网设备控制，智能合约的安全性和稳定性对于整个区块链生态系统的健康运行至关重要。因此，模糊测试将在保障智能合约安全方面发挥越来越重要的作用，从而帮助构建更加安全、可靠的区块链生态系统。

4.9.2 区块链领域的应用案例

在区块链行业，模糊测试技术已经成为确保智能合约安全性的关键工具。以下是一些区块链行业中成功的模糊测试案例。

- **以太坊**

以太坊在其智能合约平台的开发过程中，采用了模糊测试技术来增强合约的安全性和鲁棒性。模糊测试通过生成各种异常和随机的交易数据，模拟攻击者可能使用的输入，以测试智能合约在面对这些输入时的反应。通过这种方式，以太坊团队发现了多个潜在的合约漏洞，及时进行了修复，从而保护了用户的资产安全，增强了开发者对以太坊平台的信任。

- **微软 Azure 区块链服务**

微软在其 Azure 区块链服务中，使用模糊测试技术来验证智能合约的健壮性。模

糊测试自动生成了数以千计的边界条件和异常值，用以测试智能合约在极端情况下的表现。这种方法帮助微软识别并修复了智能合约中的逻辑错误和安全缺陷，提高了服务的整体质量，同时也为区块链技术在企业级应用中的推广打下了坚实的基础。

- **Chainlink 预言机**

Chainlink 在开发其去中心化预言机网络时，利用模糊测试技术来验证智能合约与外部数据源交互的安全性。模糊测试生成了各种异常的数据输入，以测试智能合约在接收到不合规或恶意数据时的响应。这一过程帮助 Chainlink 团队识别并解决了多个安全问题，保障了预言机服务的稳定性和准确性，为区块链项目提供了更加可靠的数据支持。

- **Tezos 区块链**

Tezos 在其区块链平台的开发中，采用了模糊测试技术来确保智能合约的稳健性。模糊测试通过生成大量异常的交易和状态转换，测试智能合约在面对这些情况时的响应。Tezos 团队利用这一技术发现了多个潜在的安全问题，并迅速进行了修复，确保了智能合约在各种复杂环境下的稳定运行。

- **Polkadot 的跨链技术**

Polkadot 在其跨链技术的开发中，实施了模糊测试来确保智能合约的交互安全。模糊测试通过生成各种异常的链间通信数据，测试智能合约在处理这些数据时的稳定性。Polkadot 团队通过这一技术发现了智能合约在跨链交互中的潜在风险，并及时进行了优化，提高了整个网络的安全性和互操作性。

- **Nervos Network 的 Layer 2 扩展**

Nervos Network 在构建其 Layer 2 扩展解决方案时，运用了模糊测试技术来验证智能合约的安全性。模糊测试生成了各种复杂的交易序列和状态变化，以测试智能合约在处理这些变化时的响应。Nervos Network 团队利用这一技术发现了智能合约中的潜在问题，并进行了修复，确保了 Layer 2 解决方案的安全性和效率，为用户带来了更加快速和安全的交易体验。

这些案例展示了模糊测试技术在区块链行业的成功应用，通过自动化生成大量异常的数据、交易和状态转换，加强了对智能合约的全面审查。它不仅加速了问题识别的过程，而且提高了整体的测试效率和精准度，为构建安全的区块链生态系统提供了有力支撑。

五、模糊测试的技术实施

模糊测试作为一种先进的软件测试技术，其成功实施需要系统性的方法和多样化的技术手段。从选择和使用适当的工具，到具体测试场景的应用，每一个步骤都对测试的有效性和全面性有着重要影响。本章将深入探讨模糊测试的技术实施过程，涵盖开源与商业工具的优势分析以及模糊测试的典型应用场景介绍。

5.1 模糊测试工具的选择与使用

模糊测试工具的选择与使用是模糊测试过程中的关键环节。不同的工具在功能、适用场景、操作复杂性等方面各有特点。选择合适的工具可以显著提高测试效率和效果。

5.1.1 开源模糊测试工具

当前市场上有许多开源模糊测试工具，这些工具专注于不同测试对象的安全性检测。常用的开源工具包括 AFL、AFL++、OSS-Fuzz、Honggfuzz、LibFuzzer、PeachFuzzer、Boofuzz 以及 Clusterfuzz 等。这些工具在模糊测试领域扮演着至关重要的角色，提供了多样化的选择，以适应不同的应用场景和需求。

- American Fuzzy Lop[8]（AFL）：由 Google 的安全研究员 Michal Zalewski 开发并交由开源社区维护，是一款具有重大影响力的模糊测试工具。尽管 AFL 已停止

更新并被集成进了 AFL++，但其设计和功能对开源软件测试工具产生了深远影响。AFL 主要针对源代码和非源码二进制文件进行黑盒和灰盒测试，支持 C、C++和 Objective-C 等多种编程语言。AFL 的关键优势包括利用代码覆盖率信息指导测试用例生成，探索未被覆盖的代码路径；引入词典支持功能，增强对结构化语法的测试灵活性；通过最小化处理和分类存储优化测试用例管理；并行测试支持，提升整体测试效率。

- AFL++[32]：继承并发展自 AFL，目前由开源社区积极维护和推进。AFL++是一款全面的模糊测试工具，适用于源代码、程序和固件二进制文件、图形用户界面（GUI）、网络服务以及 Android 库等的黑盒和灰盒测试，支持 C、C++和 Objective-C 等多种编程语言。AFL++通过引入核心改进如可自定义变异 API 和并行测试支持，显著提升了测试的适应性、效率和速度。可自定义变异 API 允许用户灵活地对复杂结构化输入进行变异，极大提高了处理不同类型输入的能力。并行测试及引入的持久模式则可以将测试速度提高 2 到 20 倍，这主要得益于其能够在单个进程内多次执行模糊测试，避免了每次测试都需启动新进程的低效率。

- OSS-Fuzz[33]：由 Google 开发、开源社区维护，是专门针对黑盒和灰盒测试而设计的模糊测试工具。它支持多种编程语言，包括 C/C++、Rust、Go、Python、Java/JVM 和 JavaScript，使其适用于各种开源项目。OSS-Fuzz 通过集成多个开源引擎和工具，实现了模糊测试的大规模自动化，并与持续集成系统无缝集成，简化了模糊

测试流程。

- Honggfuzz[17]：由 Google 的安全研究员开发并由开源社区维护，是一款多功能模糊测试工具，支持黑盒、白盒和灰盒测试。Honggfuzz 主要用于测试源代码和二进制程序，涵盖了 C、C++、Go 和 Rust 等多种编程语言。Honggfuzz 通过支持多进程和多线程提高了模糊测试的效率。

- LibFuzzer[34]：由 LLVM 开发团队研发并维护的一款模糊测试工具，专门用于对 C 和 C++源代码执行黑盒和灰盒测试。LibFuzzer 与 LLVM 编译器框架的紧密集成使其能够直接对源代码进行模糊测试，实现较低的性能消耗和较高的代码覆盖率，非常适合在持续集成（CI）流程中使用。

- PeachFuzzer[60]：由 PeachTech LLC 开发并于 2020 年被 GitLab 收购，是一款集成了黑盒和灰盒模糊测试的工具。它提供广泛的测试功能，涵盖文件格式、网络协议和 API，并支持 C、C++、C#、Java 和 Python 等多种编程语言。PeachFuzzer 能够生成基于文件的输入、模糊网络协议、执行网络请求和处理状态感知协议，为各种测试场景提供灵活支持。

- Boofuzz[35]：由 ErrataSecurity 开发和维护，是一款专为网络协议和应用程序设计的黑盒和灰盒测试工具，重点关注物联网（IoT）协议。Boofuzz 提供自动测试用例生成、灵活的配置和扩展选项以及网络协议的突变引擎，能够连续检查应用程序对输

入测试用例的异常响应，并记录测试结果。

- ClusterFuzz[61]：由 Google 开发并维护的模糊测试工具，专为源代码和二进制级应用程序设计，支持 C 和 C++编程语言。ClusterFuzz 具备全自动的错误分类功能和简洁的错误分析报告系统，同时支持计划内的回归测试，为大规模分布式执行环境提供了配套的报告工具，极大简化了语料样本集的管理。

尽管这些开源模糊测试引擎在网络安全领域得到了广泛应用，但它们的应用场景相对局限，且使用门槛较高，检测效果因人而异。使用开源模糊测试工具通常需要测试人员手动进行详细配置。例如，AFL 要求用户对测试目标进行编译，并使用特定的编译器参数，这对初学者来说可能非常困难。Peach Fuzz 的配置文件格式复杂，需要用户对其语法有深入了解。若想让开源模糊测试工具发挥出最大的能力，用户需要对软件编译、操作系统内核、二进制文件结构有深刻的技术理解。其次，应用开源模糊测试引擎时需要大量人工介入，用户需要借助额外的工具或编写自定义脚本来深入分析测试结果，这增加了使用难度。因此，为了实现更加智能化的模糊测试，商业化模糊测试产品成为了绝大多数企业的首选。

5.1.2 商业模糊测试产品

在国际市场上，Code Intelligence（CI FUZZ）[43]和 ForAllSecure[42]（Mayhem for Code）是代表性厂商。在中国，云起无垠的无垠模糊测试智能体[46]、安般科技的智能模糊测试平台[44]和水木羽林的 WINGFUZZ[45]是主要的产品。这些产品以其先进技术和高效性能，为软件安全领域提供了强有力的支持。此外，新秀公司"华清未央"在模糊测试技术方面也展现出深厚的积累，并开发出了自己的模糊测试产品。尽管其商业化进程仍在完善之中，但该产品已经具备了赋能特定行业的潜力。随着技术的不断进步和市场的逐步成熟，华清未央有望在软件安全领域发挥更大的作用。

- Code Intelligence（CI FUZZ）：CI Fuzz 旨在通过模糊测试自动发现软件漏洞，帮助开发和安全团队。其优势在于无缝集成到持续集成/持续部署（CI/CD）工作流程中，自动化模糊测试并识别潜在的安全漏洞和软件缺陷。这种方法使开发人员能够在软件开发生命周期的早期发现并修复漏洞，从而提高软件的整体质量和安全性。CI Fuzz 适用于单元测试、集成测试和系统测试。在单元测试中，CI Fuzz 帮助开发人员为特定代码单元生成大量测试输入，以评估这些单元的稳健性和异常处理能力。通过自动探索代码路径，CI Fuzz 揭示了标准测试用例中可能被忽视的边缘情况和隐藏错误，显著增强了单元测试的能力。

- ForAllSecure（Mayhem for Code）：Mayhem for Code 是一款自动化安全测试工具，利用先进的模糊测试和符号执行技术，自动探测软件中的潜在漏洞。通过动态生成大量测试用例并送入应用程序，Mayhem for Code 不依赖预定义的测试用例，而是探索软件的执行路径，寻找可能导致崩溃或异常行为的输入。此外，它还采用符号执行技术深入分析程序执行路径，精确识别逻辑错误，揭示模糊测试难以发现的复杂漏洞。

- 无垠模糊测试智能体：无垠模糊测试智能体集成了模糊测试技术、遗传变异算法、神经网络和大模型技术。该智能体融合了源代码、二进制、协议、操作系统、数据库、WebAPI 和固件嵌入式的模糊测试引擎，支持多种编程语言，包括 C/C++、Java、Python、Go、Rust 和 JavaScript。无垠模糊测试智能体在软件开发、部署、测试和运维的各个阶段提供全面的安全检测和漏洞修复服务。大模型技术显著增强了系统的测试驱动生成能力，大幅提升了自动化测试的效率和深度，使安全检测更加彻底。通过生成式人工智能，无垠模糊测试智能体不仅能够精准识别漏洞，还能提供详细的代码修复建议和方案。这种高效、可靠的安全检测解决方案，使企业能够更好地保障软件安全，提升整体开发质量和运维效率。

- 安般智能模糊测试平台：安般科技的智能模糊测试平台包含四大模块：源代码模糊测试、API 模糊测试、嵌入式模糊测试和协议模糊测试。该平台整合了云端模糊测

试引擎，能够无缝集成到企业的开发环境中，应用于软件开发生命周期的测试阶段，帮助在系统上线前识别软件缺陷，全面提升开发效率，保障系统安全性。

- 翼卫 WINGFUZZ 智能模糊测试平台：WINGFUZZ 是水木羽林推出的智能软件质量与安全检测平台，整合了针对数据库、操作系统、协议和 Web 应用的多种模糊测试引擎。WINGFUZZ 可以对各种层次与类型的软件进行深度的自动化漏洞挖掘，全面检测高危漏洞，有效提升软件健壮性和安全性，保障关键产品交付和业务安全运行。

这些商业产品弥补了开源工具的的诸多局限性，展现出了明显的优势：

- 广泛的适用场景：商业产品的引擎能够覆盖复杂的企业业务系统和多种应用场景，而开源引擎往往仅适用于特定测试对象。选择商业产品，企业能够全面应对各种挑战，规避自建系统可能带来的局限性。

- 自动化流程闭环：商用产品无需测试人员手动寻找测试入口和编写测试驱动，解决了开源模糊测试工具高度依赖人工和测试人员经验的问题。自动化与智能化手段显著降低了使用门槛，解放了人力资源，让企业更专注于核心业务。

- 高效的分布式并行测试：模糊测试对硬件和操作系统的消耗极大，测试运行时间长。商用产品通过多进程、多线程和分布式并行测试，在不同机器间灵活调度任务，极大提升了测试效率，确保系统高可用性。这是开源自建难以企及的高效解决方案。

- 精细化的缺陷管理：商业产品提供精准的缺陷定位和生命周期管理，从发现到

修复建立了完整的闭环。这不仅大幅提升了软件安全性，还节约了大量人力物力，让企业无需再为漏洞担忧。

- 卓越的易用性：这些产品通常配备交互式 WebUI 界面，用户可以轻松查看和管理项目、任务、测试用例及测试结果，体验远超开源工具。选择商业产品，即选择了更高效的工作流程和更愉悦的用户体验。

综上所述，商业模糊测试产品凭借其卓越的技术优势和高效的性能，远超开源工具，相比基于开源引擎自建系统时大量时间和资源投入和最终效果预期的不确定性，通过采购成熟的商业产品来快速获取可靠、易用的模糊测试检测能力，可能是更为便捷和有效的方式。

5.1.3 产品差异性分析

尽管商业模糊测试产品采用的核心技术存在一定的趋同性，但在产品设计方面，它们展现出各自独特的差异。以下是对国内外典型模糊测试产品进行的差异化对比分析：

表 2 模糊测试产品差异性分析

工具		CI FUZZ	Mayhem for Code/API	无垠模糊测试智能体	安般智能模糊测试平台	翼卫 WINGFuzz 智能模糊测试平台
厂商		Code Intelligence	ForAllSecure	云起无垠	安般科技	水木羽林
国家		德国	美国	中国	中国	中国
测试类型	白盒	√	√	√	√	√
	黑盒	√	√	√	√	√
	灰盒	√	√	√	√	√
测试对象	源代码	√	×	√	√	√
	二进制	×	√	√	×	×
	协议	×	×	√	√	√
	操作系统	×	×	√	×	√
	数据库	×	×	√	×	√
	Web API	√	√	√	√	√
	固件嵌入式	√	×	√	√	×
语言支持	C/C++	√	√	√	√	√
	C#	√	×	×	×	√
	Java	√	√	√	√	√
	Go	√	√	√	×	√
	JavaScript	√	×	√	×	×
	Python	√	√	√	×	×
	Rust	×	√	√	×	×
架构支持	Linux	√	√	√	√	√
	X86	√	√	√	×	√
	X64	√	√	√	×	√

	ARM	×	√	√	×	√
	MIPS	×	√	√	×	√
	windows	√	×	√	×	×
部署架构	本地	√	√	√	√	√
	分布式	√	×	√	√	×
全自动化测试		×	×	√	×	×
并行测试		×	×	√	√	√
种子同步		×	√	√	×	×
未知漏洞挖掘能力		√	√	√	√	√
缺陷定位		√	√	√	√	√
缺陷复现	本地	√	√	√	√	√
	在线	×	×	√	×	×
AIGC 融入	缺陷分析	×	×	√	×	×
	生成测试驱动	×	√	√	×	×
	生成测试用例	×	√	√	×	×
	智能遗传算法调度与种子评估	×	×	√	×	√
分析报告		√	√	√	√	√
缺陷检测类型		数百个安全问题和代码漏洞	支持 34 种 CWE 和 50 余种其它缺陷	支持对应用程序常见 100+种漏洞检测	支持检测 67 种缺陷和漏洞	支持 50 多种 CWE 类型
集成引擎	自研	√	√	√	√	√
	开源	√	√	√	√	√
覆盖率计算		代码级别	块、函数和源代码级别	函数、分支、代码行	语句、分支、函数、MC/DC	分支路径覆盖
误报率		极低	极低	极低	极低	极低

备注：√代表支持，×代表不支持。

如上表所示，各工具在测试类型、支持平台、编程语言、测试能力和 AIGC 融入能力增强等方面各有优劣，根据具体需求和应用场景，选择合适的模糊测试工具可以更好地保障软件的安全性和可靠性。其中，云起无垠的无垠模糊测试智能体在与国内外产品的对比中展现了明显的优势：

- **自动化和智能化测试流程：**无垠模糊测试智能体真正实现了全自动化测试流程。从工程编译到生成测试驱动、生成测试用例、缺陷分类以及系统内快速复现和修复缺陷，全过程无需人工干预。相较于传统模糊测试工具对人工和经验的高度依赖，这种全自动化显著提升了测试效率和准确性，彻底解放了人力资源。
- **大模型赋能生成高质量测试用例：**无垠模糊测试智能体不仅结合了程序分析、符号执行、覆盖反馈信息引导和遗传变异算法等多种策略，还利用 AIGC 赋能，探索全新的自动化生成方法。通过 LLM 框架，智能地生成测试驱动和测试用例，并运用智能遗传算法调度和种子评估，从而生成全面、灵活且高质量的测试用例。这种强大的生成能力，不仅满足了复杂软件系统的安全测试需求，还极大地降低了使用门槛和技术难度，提升了测试效率和覆盖率。
- **漏洞自动化修复：**无垠模糊测试智能体率先将生成式人工智能技术融入模糊测试领域，通过深入分析缺陷相关代码，全面理解代码上下文和逻辑关系，快速生成修复建议。相比其他产品，无垠模糊测试智能体能够更高效地帮助开发者定位和解决问题，

从而显著提高了修复速度和质量。

- **分布式并行测试：**无垠模糊测试智能体的分布式并行测试能力首屈一指。通过自研的分布式并行任务调度框架，支持高并发、高扩展和智能负载均衡，实现跨集群的资源调度。这种大规模分布式并行模糊测试能力，极大地提升了测试效率和覆盖率，使企业能够快速应对各种复杂测试需求。

- **全面覆盖测试场景：**无垠模糊测试智能体在多种测试对象上的表现尤为出色。它支持源代码、二进制、协议、Web API、固件嵌入式、操作系统和数据库等多种对象的安全检测，能够适应各种应用场景。无论是何种目标，无垠模糊测试智能体都能精准检测，全面保障软件安全。

- **信创环境下适配国产操作系统：**作为自主可控的国产商用产品，无垠模糊测试智能体完美适配中标麒麟、银河麒麟等国产 Linux 系统，满足国产自主化产品的安全需求。在国产自主化浪潮下，无垠模糊测试智能体提供了更全面的支持和更高的安全保障，是企业信创环境下的不二之选。

综上所述，无垠模糊测试智能体通过其先进的技术特点，为软件安全测试提供了全面而高效的解决方案。 它结合了全自动化和智能化测试流程、高质量测试用例生成、高效的漏洞检测与修复机制、强大的分布式并行测试能力、广泛的测试场景覆盖以及在信创环境下出色适配性，确保了软件的稳定性和安全性。

5.2 模糊测试场景应用

在当前的软件安全领域，模糊测试技术因其卓越的效能而备受重视，成为保障软件安全的重要手段之一。模糊测试在漏洞挖掘、网络入侵检测以及 DevSecOps 等多个关键场景中展现了强大的应用价值。以下将详细阐述模糊测试技术在这些领域中的具体应用及其重要性。

5.2.1 模糊测试在漏洞挖掘场景中的应用

漏洞挖掘平台是网络安全防御体系中的关键组件，通过合理组合利用各类技术和工具，通过自动化和智能化手段，对软件和系统进行深入的安全检测，以识别和修复潜在的安全漏洞，在攻防两端都具有重要意义，但现有漏洞挖掘平台面临诸多痛点问题。

- **隐蔽安全漏洞难以检测：**现代软件系统的复杂性使得安全漏洞更加隐蔽，传统的测试方法很难发现这些漏洞。
- **0day 漏洞威胁防范困难：**未知漏洞（0day 漏洞）的存在对网络安全构成了巨大威胁，但已有漏洞挖掘手段对 0day 漏洞检测效果不佳。
- **漏洞检测成本高昂：**传统的漏洞检测方法往往需要大量的人力和时间，这对于快速迭代的软件开发周期来说是一个挑战。

模糊测试作为一种高效的漏洞挖掘技术，在漏洞挖掘中发挥着重要作用。通过自动

化和智能化的手段，模糊测试能够对软件和系统进行深入的安全检测，识别和修复潜在的安全漏洞，对于网络安全防御体系具有重要意义。

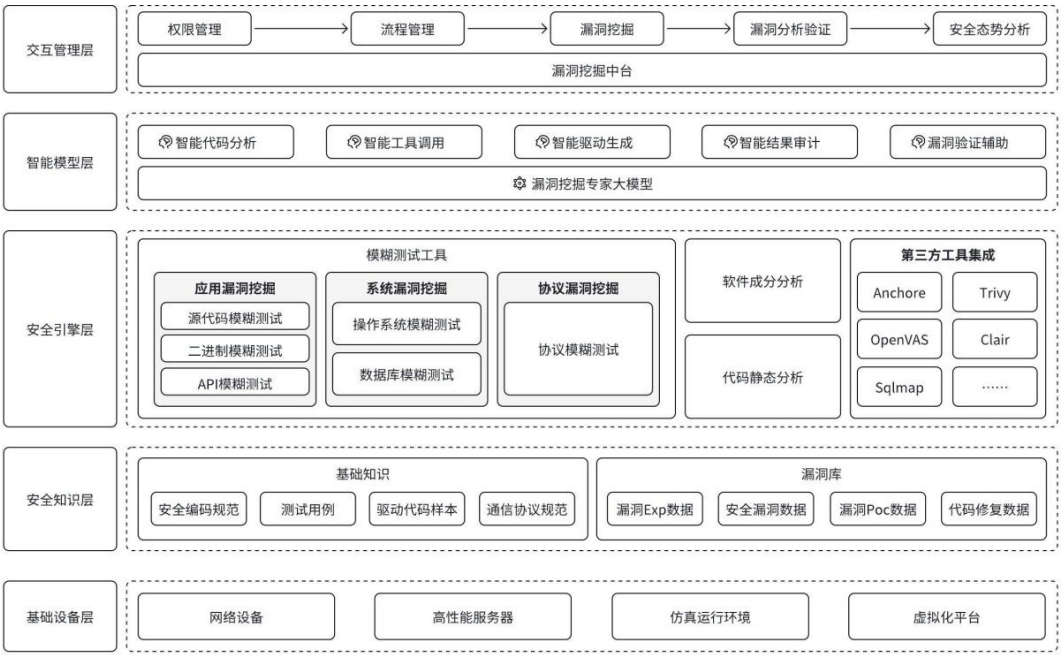

图 30 模糊测试在漏洞挖掘场景中的应用

模糊测试工具可以作为漏洞挖掘平台的核心组件，通过集成多种模糊测试引擎，支持针对于各种类型的软件和系统展开漏洞挖掘，覆盖操作系统、网络协议、数据库、Web 应用和二进制可执行文件等不同目标。通过自动化生成大量异常或随机的输入数据，减少了人工编写测试用例的需求，并覆盖广泛的输入场景，包括正常、边界和异常情况，充分遍历程序运行状态。同时模糊测试不依赖于对程序行为的先验知识，基于软件完整动态运行信息检测漏洞，能够有效地揭示深层次隐蔽漏洞和未知漏洞。

现阶段将 AIGC 等技术融入模糊测试大大增强了模糊测试能力。基于安全知识层的安全知识和漏洞库信息，可自动生成大量测试驱动，提高漏洞挖掘效率。通过向被测试

的应用程序注入大量测试样例，更全面深入地挖掘潜在安全威胁，还能通过对测试用例的分析学习，优化生成策略，增强智能化程度，提高语义解析能力，使安全检测更具针对性。而且，AIGC 融入后，可基于代码执行路径、分支覆盖和函数调用等信息，指导生成和选择能探索未覆盖代码路径的变异操作和输入，降低误报率。

将模糊测试应用于漏洞挖掘平台的主要优势如下：

- **高覆盖：**基于自动生成的海量测试用例进行全面的动态检测，遍历目标程序执行路径，可有效发现复杂和隐蔽的安全漏洞。
- **高精度：**模糊测试针对运行中的系统进行检测，具备完整的程序运行信息，并且基于目标系统实际运行状态定位安全漏洞，误报率极低。
- **高效率：**模糊测试具备高自动化程度，支持长时间持续自动化漏洞挖掘，减少了对专业安全人员的依赖，降低了安全测试的成本，具有很高的检测效率。
- **支持未知漏洞挖掘：**模糊测试基于目标系统实际运行状态定位安全漏洞，不依赖已有漏洞信息，具备 0day 漏洞检测能力，帮助信息系统防御未知风险。
- **高效漏洞验证分析：**基于静态/动态插桩技术，模糊测试可以获取目标系统代码上下文、调用栈、内存状态和崩溃用例等全面的漏洞相关信息，有效支持漏洞验证分析。

5.2.2 模糊测试在 DevSecOps 场景中的应用

DevSecOps 将安全集成到开发和运维流程中，通过自动化和协作来提高软件交付的速度和质量同时确保安全性，已有广泛的实践。在 DevSecOps 模型中，开发、安全和运维团队共同工作，从设计到部署的每个阶段都内嵌安全措施，不仅提升了软件的安全性，也加快了上市时间，降低了风险。

现有 DevSecOps 体系痛点问题包括：

- **敏捷开发流难以集成复杂安全测试：** 传统安全测试手段中，复杂的安全检测工作需要大量人工参与，自动化程度低且测试周期长，难以与敏捷开发流程无缝集成。
- **安全漏洞的验证和修复成本高：**以静态代码检测、黑盒动态测试等检测手段往往存在检测精度低、安全漏洞难以复现和验证的问题，导致安全漏洞的确认和修复成本较高。

模糊测试在 DevSecOps 中的应用主要体现在自动化和持续集成/持续交付（CI/CD）流程中，以确保在开发、测试和发布的各个阶段都进行安全检测。以下是基于所提供的图示详细解读模糊测试在 DevSecOps 中的应用：

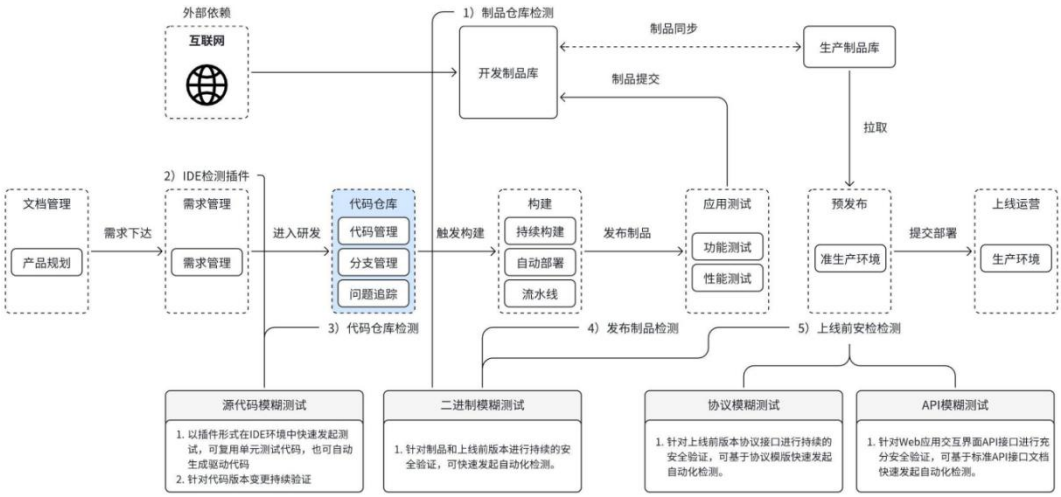

图 31 模糊测试在 DevSecOps 场景中的应用

- **需求管理和开发阶段**

在需求管理阶段，需求下达后进入开发流程，代码存储在代码仓库中。模糊测试在代码安全方面可以发挥重要作用。源代码模糊测试在开发环境的 IDE 中快速启动测试，可以在开发早期发现并修复漏洞。通过自动生成大量边界和异常输入数据，模糊测试能够动态地发现静态分析工具可能遗漏的安全问题，具有检测精度高、覆盖率高的优势。

- **代码检测和构建阶段**

在代码仓库中进行代码管理和分支管理代码时，模糊测试可以对代码库进行检测，确保代码质量和安全性。源代码模糊测试针对从代码仓库中提取的未经编译的代码，通过变异输入数据测试程序的执行逻辑和错误处理机制，从而发现潜在的缓冲区溢出、非法内存访问等漏洞。

- **应用测试阶段**

在构建完成后，进入应用测试阶段，进行功能测试和性能测试。协议模糊测试针对应用中的网络协议接口进行，自动生成不符合协议规范的数据包，测试应用程序对非标准输入的处理能力，发现协议解析器中的漏洞。API 模糊测试则针对 Web 应用及后端 API 接口，通过自动生成海量畸形输入，模拟各种攻击手段，如 SQL 注入、XSS、CSRF 等，充分检测和验证 Web 应用的安全性。

- **发布准备和预发布阶段**

在发布准备和预发布阶段，需要确保系统在准生产环境中的安全性。通过模糊测试对准生产环境进行全面的安全检测，遍历目标程序的执行路径，检测信息系统中的潜在缺陷，并支持快速验证和修复。

- **上线运行阶段**

在系统上线运行后，模糊测试依然发挥着重要作用。通过持续进行模糊测试，监控系统在生产环境中的安全状况，发现和修复潜在漏洞，防止安全事件的发生。

模糊测试在 DevSecOps 体系中的应用带来了以下效果和优势：

- 提高安全性：模糊测试可以应用在 DevSecOps 的多个环节，通过早期发现和修复漏洞，显著提高软件的安全性。
- 加速上市时间：模糊测试与 DevSecOps 深度嵌合，自动化的测试流程减少了手动测试的需要，加快了软件的开发周期。
- 降低成本：模糊测试能够助力实现安全左移，在软件生命周期的早期便发现潜

在安全风险，同时提供丰富详细的风险信息来协助快速修复，进而帮助企业降低漏洞修复的后期成本。

- 持续改进：通过对测试用例集的积累和复用，实现对同类安全问题的快速验证，促进了软件安全设计的持续改进。

5.2.3 模糊测试在入网检测场景中的应用

入网安全检测是企业或机构在自研或外购信息系统进入核心网络部署运行前的重要安全检查，其目的是综合利用各类安全检测手段，确保系统在接入核心网络前不携带任何潜在的安全威胁。这对于保护企业的核心资产、数据和知识产权至关重要。通过入网安全检测，可以识别和修复潜在的安全漏洞，从而降低数据泄露、服务中断和声誉损害的风险，也有助于满足合规性要求，为企业提供法律和技术上的安全保障。

现有入网安全检测痛点如下：

- **难以全面充分评估系统安全性：** 传统安全检测手段中 SAST 等静态检测方法缺少系统动态运行信息，DAST 等动态检测手段则受限于测试用例规模，难以实现对目标系统充分全面的安全评估。
- **难以防御未知安全风险：** 已有安全检测手段对未知漏洞检测能力有限，难以应对不断演变的安全威胁。

智能模糊测试作为一种先进的安全检测技术，通过自动生成大量异常、随机的输入

数据，对系统进行深入的动态检测，从而发现传统静态检测方法难以发现的安全问题，提供丰富全面的风险信息并支持自动复现验证，在入网安全检测场景中发挥着至关重要的作用。

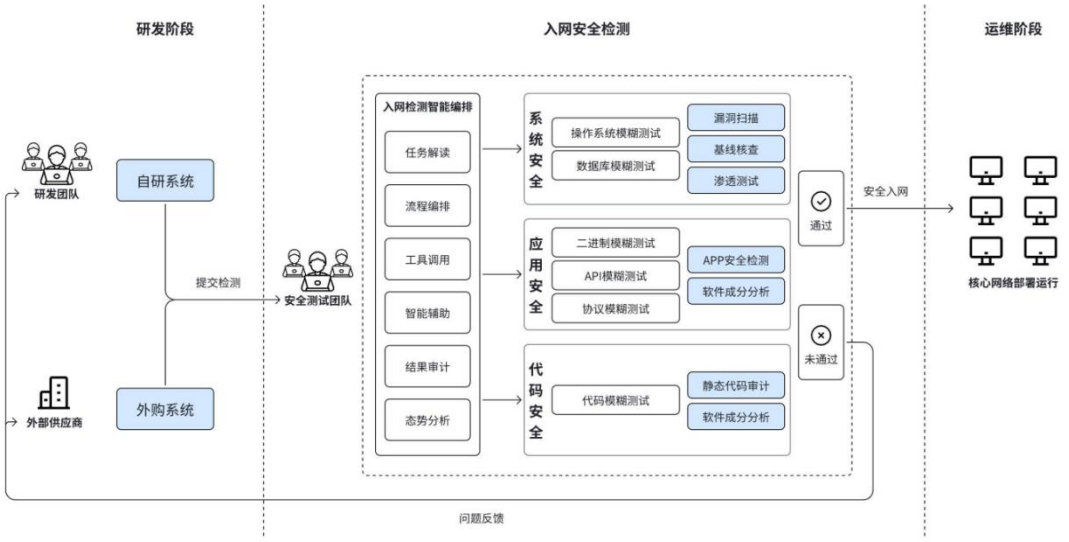

图 32 模糊测试在入网检测场景中的应用

通常，模糊测试在入网安全检测中的应用包含如下：

- **代码安全**

在代码安全检测中，代码模糊测试作为静态代码分析的有力补充，能够动态地发现那些静态分析工具可能遗漏的漏洞。静态分析可能因代码复杂性和缺少动态信息而具有较高误报率和漏报率，而代码模糊测试则通过实际执行代码来验证潜在的安全问题，检测精度高。通过自动化生成大量边界和异常输入，可以显著提高代码的测试覆盖率，从而发现边缘案例和隐蔽漏洞。模糊测试为开发人员提供丰富的漏洞信息和复现用例，支

持用户快速完成漏洞验证和修复。

- **应用安全**

- Web 安全检测：对于 Web 应用，API 模糊测试可以基于 API 接口文档自动发起全面测试，通过自动生成海量畸形输入模拟各种攻击手段，如 SQL 注入、XSS、CSRF 等，以充分检测和验证 Web 应用的安全性。
- 应用软件检测：二进制模糊测试不依赖于源代码，直接对编译后的程序二进制文件进行测试，通过自动生成变异的输入数据，包括无效的指令、异常的内存访问模式等，以测试程序的执行逻辑和错误处理机制，可以有效地发现缓冲区溢出、非法内存访问、输入验证不足等漏洞。
- 协议模糊测试通过自动生成不符合协议规范的数据包，或者故意构造异常、错误的协议消息，来测试应用程序对非标准输入的处理能力，可以揭示协议解析器中的漏洞，如协议栈的崩溃、内存泄漏、不正确的协议实现等问题。

- **系统安全**

- 操作系统安全：操作系统模糊测试可以模拟各种系统调用和文件操作，评估操作系统对异常输入的处理能力，识别潜在的崩溃点、内存泄漏、权限提升等风险。
- 数据库安全：数据库模糊测试专注于检测数据库管理系统（DBMS）的漏洞和异常处理能力，通过向数据库接口发送大量随机生成的、异常的、甚至是恶意的 SQL 语句，以评估数据库对于非预期输入的处理机制，揭示 SQL 注入漏洞、不恰当的错误信息泄露、数据泄露、权限绕过等安全问题。

模糊测试在入网安全检测中的优势如下：

- **提高安全性：**基于自动生成的海量测试用例进行全面的动态检测，遍历目标程序执行路径，充分检测信息系统潜在缺陷并支持快速验证修复，显著提高系统的安全性。
- **防御未知风险：**模糊测试基于目标系统实际运行状态定位安全缺陷，不依赖已有漏洞信息，具备 0day 漏洞检测能力，帮助信息系统防御未知风险。
- **提高效率：**模糊测试在被测目标解析、测试用例生成、缺陷定位等环节都具备高自动化程度，支持长时间持续自动化检测。它提供丰富全面的安全缺陷信息，并支持复现验证和修复指导，大幅提高检测和修复效率。

六、模糊测试的未来与发展趋势

模糊测试技术作为软件安全领域的重要工具，正在不断发展和进步。随着人工智能、大数据和云计算等新兴技术的融合，模糊测试的应用场景和技术能力也在逐步扩展和提升。未来，模糊测试将向着更加智能化、全栈化、持续集成化、并行分布式以及模糊测试即服务化的方向发展。这些趋势不仅将显著提高模糊测试的效率和覆盖率，还将降低测试的技术门槛，提升软件系统的整体安全性和质量。以下章节将详细探讨模糊测试在智能化能力、全栈漏洞检测能力、持续集成能力、并行分布式能力和模糊测试即服务能力等方面的发展趋势。

6.1 智能化模糊测试能力

随着人工智能技术的飞速发展，尤其是大模型技术的应用，模糊测试领域正迎来一场革命性的变革。模糊测试原本是一种通过自动生成大量随机或半随机的测试输入来发现软件缺陷的方法，现在正变得更加智能化、高效化和自适应化。未来的模糊测试工具将具备智能生成测试用例的能力，不仅能够根据目标应用程序的特性和行为模式自动调整测试用例，还能更有效地覆盖潜在的漏洞。

这种智能化的测试策略会依据目标系统的反馈信息动态调整，集中测试可能隐藏漏洞的代码路径或输入点，从而显著提升漏洞发现的效率与准确性。此外，这些工具还将

采用尖端的智能漏洞分析技术，通过深度分析漏洞报告、代码执行路径及输入数据，自动识别和分类漏洞，提供更精准的漏洞分析结果。

在该领域，云起无垠公司走在了行业前沿，率先提出融合模糊测试引擎、静态分析引擎与安全大模型的无垠模糊测试智能体方案。该方案不仅为测试用例生成和代码检测带来了显著优势，还极大地提升了测试工具的智能化水平。无垠模糊测试智能体通过深入分析和学习多轮模糊测试的执行结果，能够自动优化测试策略和生成的测试用例，不断提升性能和测试效果，展现了自我学习和优化的强大能力。

云起无垠的无垠模糊测试智能体不仅体现了它在模糊测试领域的领先地位，还展示了智能化安全工具在软件安全保障中的巨大潜力。通过结合先进的 AI 技术和深度学习能力，云起无垠将模糊测试带入了一个全新的智能时代，为客户提供了卓越的安全性和可靠性。该方案不仅是行业的标杆，更是未来模糊测试技术发展的风向标，引领着整个领域迈向更高的智能化和自动化水平。

6.2 全栈漏洞检测能力

未来的模糊测试技术将展现出前所未有的全栈漏洞检测能力，不仅能够高效识别内存型漏洞，还将显著增强对逻辑型漏洞的检测能力，全面涵盖 SQL 注入、命令执行、条件竞争等多种漏洞类型。这意味着模糊测试工具将具备覆盖更广泛应用程序和系统的

能力，为全方位的安全防护提供强有力支持。智能化的测试策略和自适应的测试用例生成技术将是这场变革的核心。

未来的模糊测试工具将不仅仅是简单的漏洞扫描器，而是能够深入挖掘应用程序复杂逻辑的智能检测系统。这些工具将具备动态调整测试策略的能力，依据目标系统的反馈信息，集中检测可能隐藏漏洞的代码路径或输入点，显著提升漏洞发现的效率与准确性。

目前，云起无垠已在这一领域展现了卓越的全栈覆盖能力，不仅能检测传统的内存型漏洞，还能够深入检测各种逻辑型漏洞，涵盖了从前端到后端、从应用层到系统层的全面安全检测。通过融合最先进的模糊测试引擎、静态分析引擎与安全大模型，云起无垠的无垠模糊测试智能体能够深入分析漏洞报告、代码执行路径及输入数据，自动识别和分类漏洞，提供精准的漏洞分析结果。

通过强化学习和优化机制，无垠模糊测试智能体还能够进一步利用每一次测试的结果进行自我学习，持续提升检测的准确性和效率。这种智能化、自适应的能力，使得云起无垠在应对现代复杂应用程序和系统的安全挑战中，具有显著的优势。

6.3 持续集成模糊测试能力

随着敏捷开发和持续交付模型的广泛采用，将模糊测试工具融入持续集成/持续交

付（CI/CD）流程将成为未来的主流趋势。通过这种深度融合，模糊测试不仅能够被嵌入到开发流程的早期阶段，还能实现与持续集成系统的无缝对接，使得模糊测试工具自动生成和执行测试用例，并与开发团队实时共享测试结果。

这种实践确保每一次代码更改都经过彻底的安全测试，能够尽早识别和解决潜在的安全漏洞。无论是新增功能的代码，还是修复已知问题的补丁，都会在第一时间接受模糊测试的检验，从而最大限度地降低风险。这种提前发现和修复漏洞的机制，不仅显著降低了后续开发阶段的修复负担和风险，还大大提升了代码的安全性和可靠性。

云起无垠在持续集成模糊测试领域展示了卓越的能力。通过创新的智能模糊测试智能体方案，云起无垠不仅能够在开发流程中无缝集成模糊测试，还能自动化地适应和优化测试策略，确保测试覆盖率和效果的最大化。模糊测试智能体利用先进的 AI 算法，根据每次测试结果进行学习和优化，逐步提高测试的精确性和效率。

此外，将模糊测试集成到持续集成流程中，云起无垠的方案显著加速了应用程序的交付速度。早期发现并修复漏洞，不仅减少了后期的修复工作量，还降低了因漏洞修复导致的发布延迟，使团队能够更迅速地发布安全且高质量的软件。云起无垠的解决方案通过自动化和智能化的测试流程，彻底改变了传统的开发和测试方式，为软件开发团队提供了一个高效、可靠的安全保障体系。

6.4 并行分布式模糊测试能力

随着模糊测试技术的持续进化，预计未来并行分布式模糊测试将实现前所未有的效率和成果。模糊测试的目标是通过提高测试效率、扩大测试覆盖范围，并增强系统安全性，以应对持续变化的风险和应用程序的复杂性。采用并行计算和分布式测试技术，模糊测试能够在多个计算资源上同时进行，大幅缩短测试周期。相比传统单一资源测试，分布式模糊测试能更迅速地覆盖广泛的测试用例，有效提速测试进程。利用并行分布式模糊测试，可以在更短时间内生成和测试更多的输入数据，从而显著提高软件缺陷和安全漏洞的发现率。

云起无垠在这一领域展示了全面的创新和技术实力。通过自主研发的并行分布式模糊测试能力，云起无垠不仅将模糊测试的效率提升到新的高度，还显著扩大了测试的覆盖范围。他们的解决方案能够在多个节点上同时执行模糊测试任务，充分利用闲置计算资源，如夜间或周末，在多台机器上分布式并行执行测试任务，极大提高资源的使用效率。这种高效的测试方法不仅加速了测试流程，而且为软件安全提供了更全面的保障，是适应快速迭代开发和应对日益增长的软件复杂性的关键策略。

云起无垠的并行分布式模糊测试技术不仅在效率上取得了突破，更在准确性和覆盖率上达到了前所未有的高度。通过智能调度和资源优化，云起无垠的系统能够动态分配测试任务，确保每一个计算资源都被高效利用。与此同时，他们的智能模糊测试工具能

够实时分析和反馈测试结果，自动调整测试策略，以最大化漏洞发现的概率。这种分布式测试技术的应用使得测试过程更加高效和全面，也显著提升了测试的可靠性。

云起无垠通过整合先进的人工智能和大数据分析技术，打造了一个高度智能化的模糊测试平台。这个平台不仅能够自动识别和修复软件中的安全漏洞，还能通过持续学习和优化，不断提升测试的效果和性能。

6.5 模糊测试即服务能力

随着集群化模糊测试技术的快速发展，模糊测试正逐渐演化成云原生场景中软件安全测试不可或缺的一环。未来的模糊测试不仅仅是一种测试手段，它将转变为一项服务——模糊测试即服务（Fuzzing-as-a-Service，FaaS）。这一变革预示着，不论企业规模大小，所有开发主体都将能够毫不费力地融入到模糊测试的强大生态中，从而大幅提升软件的安全性与鲁棒性。

云原生技术的推动使模糊测试的实施变得更加灵活、高效，不仅拓宽了安全测试的覆盖范围，还深化了测试的深度，使得潜在的安全风险能够被全方位地识别和修补。通过这种服务模式，开发团队可以随时随地利用强大的模糊测试功能，而无需投入大量资源进行本地部署和维护。

云起无垠的模糊测试即服务（FaaS）解决方案，不仅集成了最先进的模糊测试引

擎，还结合了强大的云计算资源和智能化技术。无论是小型初创公司还是大型企业，云起无垠的 FaaS 都能提供灵活、高效且经济的安全测试服务，帮助客户显著提升软件的安全性和稳定性。

云起无垠的 FaaS 平台通过自动化的测试用例生成、智能调度和实时反馈，极大地提高了漏洞发现的效率与准确性。其独特的智能模糊测试能力，能够动态调整测试策略，根据每次测试的结果进行自我优化，不断提升测试效果。这种智能化、自适应的服务，使得模糊测试不仅更加高效，还能够实时适应变化的威胁环境。

此外，云起无垠的 FaaS 平台还提供了强大的分析和报告功能，使得开发团队能够迅速了解和处理发现的漏洞。通过详细的漏洞分析报告，团队可以精准定位问题根源，采取有效的修复措施，确保软件的安全性。

云起无垠的 FaaS 能力不仅提升了测试的自动化程度，还通过云平台的强大计算能力，显著降低了测试的时间和成本。这项解决方案不仅是对传统模糊测试技术的革新，更是对整个软件安全测试领域的一次颠覆性提升。凭借其强大的技术实力和创新能力，云起无垠正引领着模糊测试技术的发展，为客户提供了无与伦比的安全保障和服务体验。

随着技术的不断演进和应用的进一步深化，FaaS 有望成为推进软件安全进步的关键动力。这种以服务形式存在的模糊测试，将大力促进软件安全测试领域的创新和发展，为应对日益复杂的安全挑战提供一种灵活而有效的解决方案。

七、模糊测试的政策与标准

国际和国内的多项政策与标准均对模糊测试给予了明确指导和认可，从侧面印证了其在保障软件安全中的关键作用。随着技术的发展，模糊测试已成为软件漏洞评估的重要技术，对于提升软件质量和安全性具有不可或缺的价值。

7.1 国外政策与标准

模糊测试已成为数据保护和网络安全领域的重要工具，受到了多个国际标准和指南的推荐和规范。2016 年，GDPR 开始对在欧盟境内处理个人数据的组织提出严格要求，包括在模糊测试过程中确保数据最小化和数据保护。2021 年，ISO/SAE 21434、OWASP Mobile Security Testing Guide、NIST IR8397 等标准和指南都强调了模糊测试在系统集成、应用程序安全和软件测试中的重要性。美国政府也在其国防授权法案中建议使用模糊测试来提高安全性。

- 2016 年，欧洲联盟通过了《通用数据保护条例》（GDPR），规定了数据保护和隐私权方面的一系列严格要求。对于在欧盟境内处理个人数据的组织，GDPR 要求在进行模糊测试时必须遵守数据最小化和数据保护的相关规定。

- 2021 年 8 月，ISO/SAE 发布了 ISO/SAE 21434 标准，规定了在道路车辆网络安全工程中系统集成应结合适当的方法进行验证和测试，其中包括基于需求的正向和

反向测试、接口测试、渗透测试、漏洞扫描以及模糊测试。

- 2021 年 7 月，开放全球应用程序安全项目（OWASP）发布了《OWASP Mobile Security Testing Guide》，该指南提供了模糊测试的方法、工具和实施建议，帮助开发人员和测试人员在应用程序安全测试中应用模糊测试技术。

- 2021 年 5 月，美国国家标准与技术研究院（NIST）在其 NIST IR8397 文档中提到软件测试的最低标准，详细说明并明确推荐使用模糊测试工具进行自动化的主动测试。

- 2020 年，NIST 发布了 NIST SP 800-115 和 NIST SP 800-53 指南，这些指南提供了模糊测试的概述、方法、工具和实施建议，以支持组织进行软件安全测试和评估。

- 2019 年，美国政府在其国防授权法案（HR 5515）中大篇幅建议国防高级研究计划局使用模糊测试来增强国防系统的安全性。

- 2008 年，开放全球应用程序安全项目（OWASP）在其《OWASP Testing Guide》中详细介绍了在 Web 应用安全测试过程中，如何使用模糊测试技术来发现安全漏洞。该指南提供了一个测试框架和针对 Web 应用的各种安全测试方法，其中包括模糊测试的实践和技巧。

7.2 国内政策与标准

国内多项政策和标准明确了模糊测试的必要性，推动了网络安全和模糊测试技术的发展。2023 年，国家发展改革委发布的《产业结构调整指导目录（2024 年本）》以及信通院发布的《模糊测试技术分级能力要求》进一步推动了这一进程。《产业结构调整指导目录》新增了“网络安全”这一重要行业大类，特别强调了源代码审计工具和模糊测试工具在产业优化中的关键作用。《模糊测试技术分级能力要求》规范了黑盒和灰盒模糊测试的平台能力及引擎能力，提供了全面的指导并将模糊测试平台能力分为基础级、增强级和先进级三个等级。早在 2020 年，工业和信息化部发布的《汽车电子系统网络安全指南》以及国家证监会发布的《证券期货业软件测试指南 软件安全测试》已强调了模糊测试在确保系统和软件安全中的重要性。这些政策和标准共同推动了模糊测试技术在国内的广泛应用和发展。

- 2023 年，国家发展改革委发布了《产业结构调整指导目录（2024 年本）》，该目录自 2024 年 2 月 1 日起正式实施。与之前的版本相比，本次目录在行业设置上进行了全面升级，新增了“网络安全”这一重要行业大类，并对有利于产业优化升级的领域进行了细化。其中，网络安全检测工具部分重点提到了源代码审计工具、模糊测试工具等关键工具，为产业结构注入了新的活力。

- 2024 年，中华人民共和国公安部发布了《信息安全技术 模糊测试系统安全技

术要求与测试评价方法》，该文件规定了模糊测试系统的安全技术要求和测试评价方法，适用于指导其设计、开发和测试。文件明确了相关术语和缩略语，概述了产品功能和部署方式，规定了安全技术要求，包括安全功能、自身安全、环境适应性和安全保障等方面，并给出了相应的测试评价方法。此外，文件还附录了安全技术要求等级划分表和参考文献。该标准为模糊测试系统的安全性提供了全面的规范和指导。

- 2023 年，信通院发布了《模糊测试技术分级能力要求》标准，规范了黑盒、灰盒模糊测试的平台能力及引擎能力。该标准包含 4 个能力域、28 个能力项和 246 个能力指标，提供了模糊测试平台能力建设的全方位指导。根据模糊测试平台在不同能力域的综合表现，分为基础级、增强级和先进级 3 个能力等级。
- 2020 年，工业和信息化部发布了《汽车电子系统网络安全指南》（GB/T 38628-2020），为了确保应用的安全技术能够满足系统的网络安全技术需求，组织宜通过独立的网络安全测试团队对其有效性进行验证，可采用的验证方法包括模糊测试。
- 2020 年，国家证监会发布了《证券期货业软件测试指南 软件安全测试》，规范了安全功能检查、代码安全测试、漏洞扫描、渗透测试和模糊测试五项安全测试技术的定义与测试内容。该指南明确提出针对证券交易所、期货交易所、证券公司和期货公司，模糊测试是保障软件安全的必选项。

八、结论与展望

在当今瞬息万变的软件安全测试领域，传统的静态和动态分析技术正面临前所未有的挑战。这些方法主要依赖于漏洞特征库和规则集，难以全面覆盖所有潜在漏洞，漏报率居高不下，且在检测零日漏洞方面效果有限。此外，现有检测方案的误报率高得令人沮丧，安全团队因此需要耗费大量时间和精力来排查误报，从而严重影响了测试效率和实际漏洞检测效果。面对日益复杂的安全威胁，软件安全测试技术亟需更加全面、精准的解决方案，以实现大规模的漏洞自动化挖掘、检测与修复。

模糊测试技术在应对这些挑战时展现出显著优势。通过自动生成大量不同的输入，模糊测试技术能够广泛覆盖软件的各种代码路径，大幅提高漏洞发现的覆盖面。不依赖于已知的漏洞特征，模糊测试通过变异输入来发现新漏洞，特别是对零日漏洞的检测，效果极为显著。随着人工智能与模糊测试技术的深度融合，智能模糊测试将开启高度自动化的新纪元。这种技术能够自动生成输入、执行测试、分析结果并进行漏洞修复，极大减少人工干预，并通过并行化和分布式测试提高测试效率，实现更全面、更高效的漏洞挖掘、检测与修复能力。云起无垠公司的产品在这一领域表现尤为出色，其模糊测试智能体通过结合大模型技术，展现出强大的漏洞检测与修复能力。

本白皮书全面而深入地探讨了模糊测试技术的核心理念、发展历程、应用领域以及

具体实施步骤，不仅详细解析了模糊测试技术的根本原理及其演进轨迹，还阐述了其在不同实际环境下的应用范例。特别是对开源和商业化应用两种模式下的模糊测试技术进行了深入剖析，帮助读者更准确地把握其本质特性，将其有效应用于实际环境中。

此外，本白皮书精心挑选了一些具有代表性的实践案例，通过这些生动的例子，读者可以更直观地体验模糊测试技术在日常生活中的实际效果和价值。这些案例涵盖军工、信创、金融、汽车、工业互联网和检验检测机构等多个领域，展示了模糊测试技术在解决现实问题中的强大潜力。

展望未来，本白皮书提出了模糊测试技术应对未来挑战的战略规划。这些见解包括加强模糊测试技术的研发和创新、推动其标准化和规范化、提高技术的可重复性和可移植性等方面的建议。相信这份白皮书可以帮助读者深入理解模糊测试技术的精髓，还可为模糊测试未来的发展和应用提供有价值的参考，为软件安全测试领域的未来带来无限可能。

九、参考资料

[1]焦龙龙, 罗森林, 刘望桐, 等. 基于遗传算法的二进制程序模糊测试方法[J]. 浙江大学学报: 工学版, 2018, 52(5): 1014-1019.

[2]张瀚方, 周安民, 贾鹏, 等. 面向二进制程序的导向性模糊测试方法[J]. 计算机应用, 2019, 39(5): 1389-1393.

[3]陈静, 魏强, 武泽慧, 等. REST API 自动化测试综述[J]. Application Research of Computers/Jisuanji Yingyong Yanjiu, 2024, 41[2].

[4]李富合, 高东林, 曹宁生. 基于 RESTful 的中间件服务化体系结构及关键技术研究[J]. 舰船电子工程, 2019, 39(7): 113-118.

[5]田韵嵩, 李中伟, 谭凯, 等. 汽车 CAN 总线入侵检测算法性能模糊测试方法研究[J]. Information Technology & Network Security/Xinxi Jishu yu Wangluo Anquan, 2022, 41(4).

[6]赖英旭, 杨凯翔, 刘静, 等. 基于模糊测试的工控网络协议漏洞挖掘方法[J]. 计算机集成制造系统, 2019, 25(第 9): 2265.

[7]万年红, 李翔. 软件黑盒测试的方法与实践[J]. 计算机工程, 2000, 26[12]: 91-93.

[8]Fioraldi A, Mantovani A, Maier D, et al. Dissecting American Fuzzy Lop: A FuzzBench Evaluation[J]. ACM transactions on software engineering and methodology, 2023, 32[2]: 1-26.

[9]Cowan C, Arnold S, Beattie S, et al. Defcon capture the flag: Defending vulnerable code from intense attack[C]//Proceedings DARPA information survivability conference and exposition. IEEE, 2003, 1: 120-129.

[10]Aumasson J P, Romailler Y. Automated testing of crypto software using differential fuzzing[J]. Black Hat USA, 2017, 7: 2017.

[11]任泽众, 郑晗, 张嘉元, 等. 模糊测试技术综述[J]. 2021.

[12]Mohammed M, Cai H, Meng N. 猴子测试和人体测试的实证比较（wip 论文）[C]//第 20 届 ACM SIGPLAN/SIGBED 嵌入式系统语言、编译器和工具国际会议论文集。2019：188-192。

[13]Ravipati G, Bernat AR, Rosenblum N 等。《Dyninst 的解构》[J]。威斯康星大学，技术报告，2007，32.

[14]刘文伟, 刘坚. 一个重建 GCC 抽象语法树的方法[J]. 计算机工程与应用, 2004, 40[18]: 125-128.

[15]戴渭, 陆余良, 朱凯龙. 基于动态能量调控的导向式灰盒模糊测试技术[J]. 浙江大学学报: 工学版, 2020, 54(8): 1534-1542.

[16]杜子德. 程序控制流图: 一种可观化的程序设计工具[J]. 计算机研究与发展, 1995, 32[12]: 15-20.

[17]Bamohabbat Chafjiri S, Legg P, Tsompanas M A, et al. Honggfuzz+: Fuzzing by Adaptation of Cryptographic Mutation[J]. Phil and Tsompanas, Michail-Antisthenis and Hong, Jun, Honggfuzz+: Fuzzing by Adaptation of Cryptographic Mutation, 2023.

[18]Lan Y, Jin D, Wang Z, et al. Thunderkaller: Profiling and Improving the Performance of Syzkaller[C]//2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2023: 1567-1578.

[19]Sen K. Concolic testing[C]//Proceedings of the 22nd IEEE/ACM international conference on Automated software engineering. 2007: 571-572.

[20]Wen C, Wang H, Li Y, et al. Memlock: Memory usage guided fuzzing[C]//Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 2020: 765-777.

[21]Wang T, Wei T, Gu G, et al. TaintScope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection[C]//2010 IEEE Symposium on Security and

Privacy. IEEE, 2010: 497-512.

[22]Woo M, Cha S K, Gottlieb S, et al. Scheduling black-box mutational fuzzing[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. 2013: 511-522.

[23]Böhme M, Pham V T, Roychoudhury A. Coverage-based greybox fuzzing as markov chain[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 1032-1043.

[24]杨克, 贺也平, 马恒太, 等. 面向递增累积型缺陷的灰盒模糊测试变异优化[J]. 软件学报, 2022, 34(5): 2286-2299.

[25]肖天, 江智昊, 唐鹏, 等. 基于深度强化学习的高性能导向性模糊测试方案[J]. 网络与信息安全学报, 2023, 9[2]: 132-142.

[26]Zhu F, Sammler M, Lepigre R 等。BFF：位域操作程序的基础和自动验证[J]。ACM 编程语言论文集，2022 年，6(OOPSLA2)：1613-1638。

[27]Stallman R、Pesch R、Shebs S. 使用 GDB 进行调试[J]。自由软件基金会，1988，675。

[28]Miller C、Caballero J、Johnson NM 等。使用 BitBlaze 进行崩溃分析[J]。BlackHat USA，2010 年。

[29]Duchene F, Rawat S, Richier J L, et al. KameleonFuzz: evolutionary fuzzing for black-box XSS detection[C]//Proceedings of the 4th ACM conference on Data and application security and privacy. 2014: 37-48.

[30]Appelt D, Nguyen C D, Briand L C, et al. Automated testing for SQL injection vulnerabilities: an input mutation approach[C]//Proceedings of the 2014 International Symposium on Software Testing and Analysis. 2014: 259-269.

[31]Pan Y. Interactive application security testing[C]//2019 International Conference on

Smart Grid and Electrical Automation (ICSGEA). IEEE, 2019: 558-561.

[32]Fioraldi A, Maier D, Eißfeldt H, et al. {AFL++}: Combining incremental steps of fuzzing research[C]//14th USENIX Workshop on Offensive Technologies (WOOT 20). 2020.

[33]Ding Z Y, Le Goues C. An empirical study of oss-fuzz bugs[C]//2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR). IEEE, 2021: 131-142.

[34]Serebryany K. Continuous fuzzing with libfuzzer and addresssanitizer[C]//2016 IEEE Cybersecurity Development (SecDev). IEEE, 2016: 157-157.

[35]Cousineau P, Lachine B. Enhancing Boofuzz Process Monitoring for Closed-Source SCADA System Fuzzing[C]//2023 IEEE International Systems Conference (SysCon). IEEE, 2023: 1-8.

[36]新思科技《2021 年软件漏洞快照：新思科技应用安全测试服务分析》报告：https://www.synopsys.com/software-integrity/resources/analyst-reports/software-vulnerability-trends.html?cmp=pr-sig&utm_medium=referral

[37]模糊测试智能体：www.clouditera.com

[38]Zambelli F, Chiara M, Ferrandi E, et al. Ascan: a novel method for the study of allele specific expression in single individuals[J]. Journal of Molecular Biology, 2021, 433[11): 166829.

[39]Zhao W, Li Z, Li R, et al. CapSan-UB: Bug Capabilities Detector Based on Undefined Behavior[C]//Proceedings of the 2023 13th International Conference on Communication and Network Security. 2023: 6-11.

[40]Tsan M F, Gao B. Heat shock proteins and immune system[J]. Journal of Leucocyte Biology, 2009, 85(6): 905-910.

[41]SecGPT：https://github.com/Clouditera/SecGPT

[42]Brumley D. The cyber grand challenge and the future of cyber-autonomy[J]. USENIX Login, 2018, 43[2]: 6-9.

[43]Klooster T, Turkmen F, Broenink G, et al. Continuous Fuzzing: A Study of the Effectiveness and Scalability of Fuzzing in CI/CD Pipelines[C]//2023 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT). IEEE, 2023: 25-32.

[44]https://www.anban.tech/details/fuzzing

[45]https://www.shuimuyulin.com/pages/product/wfuzz.html

[46]https://www.clouditera.com/

[47]Miller B P, Koski D, Lee C P, et al. Fuzz revisited: A re-examination of the reliability of UNIX utilities and services[R]. University of Wisconsin-Madison Department of Computer Sciences, 1995.

[48]王天佐, 王怀民, 刘波, 等. 僵尸网络中的关键问题[D]. , 2012.

[49]寇春静, 刘志娟, 张弛, 等. 中国大陆信息网络安全学术研究的影响力分析[J]. 信息安全研究, 2020, 6(9): 0.

[50]Bruening D, Amarasinghe S. Efficient, transparent, and comprehensive runtime code manipulation[J]. 2004.

[51]Luk C K, Cohn R, Muth R, et al. Pin: building customized program analysis tools with dynamic instrumentation[J]. Acm sigplan notices, 2005, 40(6): 190-200.

[52]Nethercote N, Seward J. Valgrind: a framework for heavyweight dynamic binary instrumentation[J]. ACM Sigplan notices, 2007, 42(6): 89-100.

[53]Bellard F. QEMU, a fast and portable dynamic translator[C]//USENIX annual technical conference, FREENIX Track. 2005, 41[46]: 10-5555.

[54]李瑛, 张盛兵, 高德远. Verilog Testbench 设计技巧和策略[D]. , 2003.

[55]Nagarakatte S, Zhao J, Martin M M K, et al. CETS: compiler enforced temporal safety for C[C]//Proceedings of the 2010 international symposium on Memory management. 2010: 31-40.

[56]Chovancova E, Pavelka A, Benes P, et al. CAVER 3.0: a tool for the analysis of transport pathways in dynamic protein structures[J]. 2012.

[57]Haller I, Jeon Y, Peng H, et al. TypeSan: Practical type confusion detection[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 517-528.

[58]Jeon Y, Biswas P, Carr S, et al. Hextype: Efficient detection of type confusion errors for c++[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 2373-2387.

[59]Cui W, Peinado M, Cha S K, et al. Retracer: Triaging crashes by reverse execution from partial memory dumps[C]//Proceedings of the 38th International Conference on Software Engineering. 2016: 820-831.

[60]Zhang D, Wang J, Zhang H. Peach improvement on profinet-DCP for industrial control system vulnerability detection[C]//2015 2nd International Conference on Electrical, Computer Engineering and Electronics. Atlantis Press, 2015: 1622-1627.

[61]Chen Y, Jiang Y, Ma F, et al. {EnFuzz}: Ensemble fuzzing with seed synchronization among diverse fuzzers[C]//28th USENIX Security Symposium (USENIX Security 19). 2019: 1967-1983.

[62]HDiff: A Semi-automatic Framework for Discovering Semantic Gap Attack in HTTP Implementations, DSN 2022.

[63]Break the Wall from Bottom: Automated Discovery of Protocol-Level Evasion Vulnerabilities in Web Application Firewalls, IEEE S&P 2024

[64]Wang E, Chen J, Xie W, et al. Where URLs Become Weapons: Automated Discovery of SSRF Vulnerabilities in Web Applications[C]//2024 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 2024: 216-216.

服务热线：010-58171400

商务沟通：136-6112-5537

媒体联络：marketing@clouditera.com

云起无垠网址：www.clouditera.com

公司地址：北京市海淀区盈都大厦 C 座 3 单元

关注公众号

关注公众号