



硕士研究生中期进展报告

|  |  |
| --- | --- |
| **学 号：** |  |
| **姓 名：** |  |
| **导 师：** |  |
| **论文题目：** |  |
| **学科专业：** |  |
| **学 院：** |  |
| **填写时间：** | 年 月 日 |

西安交通大学研究生院制

硕士研究生中期进展报告填写说明及管理规定

一、硕士学位论文中期考核要求在校硕士生必须在入学第四学期末（两年毕业试点学院的硕士生应在第三学期末）完成。

二、硕士研究生在完成了一定论文工作的基础上，填写完成《硕士研究生中期进展报告》。

三、《硕士研究生中期进展报告》完成以后，应组织公开的中期考核报告会。

四、中期考核由学院负责统一组织，考核专家组一般由5名副高以上（含副高）人员组成。考核专家主要依据硕士研究生的论文课题进展情况进行考核，同时可参阅其课程学习和选题报告情况。中期考核结束后，考核专家应在本表中填写考核评价结果、评语和论文修改意见。

五、《硕士研究生中期进展报告》必须采用A4纸双面打印，左侧装订成册，各栏空格不够时，请自行加页。本表可在研究生院主页<http://gs.xjtu.edu.cn/>下载。

六、《硕士研究生中期进展报告》由学院归档。

|  |  |  |
| --- | --- | --- |
| **论文题目** | 基于模糊测试路径探索的区块链共识算法研究与实现 | |
| **一、研究内容简介**（300～500字）  近些年随着各界对其研究与应用的不断深入，区块链技术已经开始独立于加密数字货币，发展成为一门新的研究领域。  区块链是一个去中心化的分布式网络，让所有的独立节点达成共识去做同一件事情，是“去中心化”这一概念首要解决的事情，区块链可以帮助所有人无门槛地构建自己的小经济系统。  区块链相关技术有很多方面，大致可分为4类：共识算法、隐私保护技术和相应密码学技术、智能合约相关技术、面向应用相关技术。其中，共识算法是区块链技术最核心，也是整个技术发展和学术界最热衷的领域。目前常见的共识算法有工作量证明（PoW，Proof-of-Work）、权益证明（PoS，Proof-of-Stake）、委托权益证明（DPoS，Delegated-Proof-of-Stake）等。  然而，传统PoW需要较高的处理器能力和能耗，环境成本高，且这些算力只是用于挖矿，并没有其他用途。因此，许多研究致力于将区块链与其他技术相结合，以利用被浪费的算力，同时还能激励其他技术的发展。  一些作者提出了一种新的范式，在这种范式中，挖掘新区块所需解决的数学问题本身可能具有一些兴趣，而不仅仅是支持区块链网络的有用性。在某些情况下，这种方法被称为有用的工作量证明[17]，从而认识到为挖掘一个区块所做的计算工作必须具有一些有趣或有用的应用。  与此同时，Fuzzing是一种将异常数据输入软件以发现潜在漏洞的技术。它主要用于发现漏洞，包括但不限于处理异常输入时的缓冲区溢出、内存泄漏和崩溃。现有的Fuzzing方法在面对复杂系统时，容易出现饱和效应（Saturation effects），指持续使用同一刺激材料导致刺激效果递减的心理现象，这使得在模糊测试中，测试人员难以高效发现新漏洞。  然而传统PoW存在的问题时是它的算力全部用在了暴力破解谜题中，其唯一目的是计算用于挖掘新区块的随机数，而在此之外没有其他有用的应用，因此本课题提出了一种有用的工作量证明，称为Proof-of-Fuzzing，旨在利用传统PoW挖矿的算力，并提供一个漏洞挖掘的综合平台，为有检测代码漏洞需求的人提供服务，同时也通过奖励机制激励矿工发现漏洞，生成被测代码的漏洞报告。 | | |
| **二、研究工作进展**（已完成的主要工作及已取得的成绩。3000～5000字）  1、将模糊测试过程转化为Proof-of-Fuzzing共识算法  1.1 Proof-of-Fuzzing共识算法的设计  考虑到要将一个新的共识算法引入区块链网络，必须遵守以下几个方面：   1. 这个问题必须是复杂的，需要一定的计算量，才能保证矿工进行了一些实际的工作，从而能够获得与区块挖掘相关的奖励。 2. 为了保证区块链的完整性，必须将前一个区块的哈希值作为问题的变量引入。 3. 挖矿方案必须有一个竞争性的组成部分，这样第一个解决问题的矿工（或者提供最佳解决方案的矿工）就是挖出区块并获得奖励的矿工。 4. 给定一个问题解决方案，必须很容易验证该解决方案的有效性并评估其质量。 5. 一旦某个矿工成功将一个新区块添加到当前区块链中，其他矿工正在挖掘的所有其他潜在区块都必须被丢弃。这保证了矿工不能“保存区块”用作未来的挖矿结果。   漏洞奖励计划（Vulnerability Rewards Program，VRP）是一种激励安全研究人员发现和报告软件漏洞的机制，通过这种机制，软件开发者可以及时修复漏洞，提高软件的安全性和稳定性，同时漏洞发现者也可以获得奖励。VRP是目前很多科技公司查找自家漏洞的主要方法之一，包括Google、Apple、Microsoft等公司。  本课题受漏洞奖励计划的启发，我们建议矿工解决的难题是在一定时间内，对发布者/供应商发布的被测程序进行持续性的模糊测试以达到覆盖率要求，并要求计算出的路径Hash值在一个区间内，这是一种计算昂贵的任务，因此满足条件1。矿工在计算路径Hash值时，需要包含前一个区块的Hash值，因此满足条件2。本共识算法要求计算出的路径Hash值在一个动态调整的区间中，第一个满足要求的矿工获得下一个区块的记账权，因此满足条件3。每次执行窗口结束后，各矿工提交自己的路径Hash和payload，由validator验证它们的合法性，根据payload可轻易算出Hash进行对比，检验该矿工提交的测试结果和签名是否准确以及确认该矿工执行的是指定目标程序，满足要求4。一旦确定获胜的矿工，该区块将被添加到区块链末端，那么最后一个区块的哈希值就发生了变化，成为待开采新区块的前一个区块，计算新Hash值中的pre\_Hash发生变化，矿工们将进入下一个执行窗口，满足要求5。  1.2 执行路径的获取  本课题选用AFL（American Fuzzing Loop）作为模糊测试工具，AFL是一种主流的覆盖引导（coverage-guided）灰盒模糊器（Fuzzer）,它维护一个测试用例队列，每次选择一个测试用例进行多轮变异（mutation），如位翻转、替换等，变异后的测试用例将作为被测程序的输入。之后Fuzzer便会向被测程序发送执行信号，等待执行完成后收集代码覆盖率和退出状态等信息。在本课题的设计中，每位矿工在执行给定时间的被测程序后都需要给出执行用例和它对应的执行路径，在AFL中并没有直接输出路径的功能，因此为实现这一目标，需要对AFL的源码进行修改。  AFL使用编译时插桩来跟踪被测程序的执行过程，它通过记录被测程序中基本块（Basic Block）之间的跳转来实现这一目的。AFL插桩代码使用的语言是AT&T风格的汇编语言，桩被插入到已编译程序中的特定位置，在以下几个地方插桩：函数入口点、条件跳转指令后、指令的标签后。桩指令完成的任务可以概括为图1。    图1 AFL插桩核心代码逻辑  AFL在插桩时为每一个基本块赋值随机数（即图1中的COMPILE\_TIME\_RANDOM），作为它的标识ID，两个ID对即表示控制流转换（我们称之为边）。当被测程序运行到这个基本块时，该块即被命中（hit），AFL将当前块和前一块异或后保存到共享内存（Fuzzer和被测程序间共享）中，最后将当前块右移一位（为了区分两个块之间不同方向的路径），完成对两个块之间的边的标记。因此获取执行路径就转换为获取这些被命中的块的ID。AFL的桩代码逻辑如下图所示。  （桩代码逻辑图，插入的存入文件的汇编代码）  1.2 执行窗口的实现  由于每个人每次执行代码的时间都是不一样的，为了公平判定矿工对该被测程序漏洞挖掘的贡献度以决定记账权的分配，本课题设计了一个执行窗口，该窗口以物理时间为界，统计每个矿工在窗口中执行的路径总数、发现的新路径和漏洞等。  考虑到执行窗口的部分可能会受到恶意矿工的攻击，使得该矿工并不遵守执行窗口的时间限制，从而提交更多的执行路径，这会破坏挖矿的公平性，为了确保矿工可信执行了挂起操作，选择将执行逻辑放入到Intel SGX（Intel Software guard extensions）中，确保定时挂起，SGX是Intel架构新的扩展，在原有架构的基础上增加了一组新的指令集和内存访问机制，这些扩展允许应用程序实现一个被称为enclave的容器，在应用程序的地址空间中划分出一块被保护的区域，为容器内的代码和数据提供机密性和完整性的保护，免受恶意软件的破坏。由因特尔颁发的证书，在里面的代码会被签名，这个签名所有人都可以验证，那么拿到这个签名就代表执行了这段不可更改的代码。如果绕过该代码就拿不到这个签名。  在每轮执行窗口结束后，矿工需要广播自己的结果，根据1.2节中获取的执行路径，对路径序列进行Hash计算，输入为payload, path, is\_crash, salt, prev\_block, transctions，采用sha256加密算法来进行Hash计算。计算得到的Hash值需要使用enclave私钥签名得到最终结果，然后进行广播。    1.3 获取记账权  共识算法主要解决两个问题，即在分布式记账的场景下，谁有权记账和如何避免记账者作弊的问题。发布者在发布待测代码后，矿工们即可开始漏洞检测工作，系统会在每个执行窗口结束后核算，以确定最终记账权的获得者。本课题设计的争夺记账权的条件有两个且优先级不同。条件一：如果某个矿工最先发现了新路径，那么该矿工将获得奖励和记账权；条件二：如果在某个执行窗口中，没有矿工发现新路径，那么则选择拥有最先满足哈希条件（即在给定的hash区间中）的矿工赋予记账权；条件一的优先级要高于条件二。如果在该轮执行窗口中，没有任何矿工符合这两个条件中的任意一个，则该轮置空，没有人获得奖励和记账权，系统需要动态调整条件二中满足哈希范围的难度使得下一轮不被置空。  对于上述条件二，由于路径hash值得分布是不均匀的，因此本课题引入了动态可变映射的hash区间，通过反馈的方式动态调整该hash区间，确保在固定时间块内有人一定可以获得记账权，即所探索的路径的hash落到该区间内，算法如下所示。    1.4奖励机制的设计  本课题的奖励机制包括三方面：基本奖励、漏洞奖励和路径贡献奖励。基本奖励即成功挖矿的奖励，与传统PoW类似，该奖励只发给在上一轮窗口中获得记账权的矿工，且随区块高度、难度、模糊测试难度等因素不断调整。基本奖励计算方法如下所示：  其中，D为模糊测试难度，它的计算方法如下所示：  其中，C为代码覆盖率因子，R为资源消耗因子，P为程序复杂度因子，分别为各因子对应的权重。  代码覆盖率因子计算方法如下所示：  其中，为行覆盖率，为分支覆盖率，为函数覆盖率。  资源消耗因子计算方法如下所示：  其中，为模糊测试执行时间，为模糊测试允许执行的最大时间，即执行窗口的时间范围，为模糊测试占用的内存，为允许的最大内存。  程序复杂度因子计算方法如下所示：  其中，为代码行数，为总代码行数，为函数数量，为总函数数量，为依赖库数量，为总依赖库数量。  代码覆盖率包括行覆盖率、分支覆盖率、函数覆盖率；资源消耗即模糊测试过程中的执行时间（如CPU时间）、内存占用等，奖励也会提高；程序复杂度通过目标程序的代码行数、函数数量、依赖关系等来判定。代码覆盖率越高、执行时间越长、内存占用越大、程序越复杂，模糊测试难度就越大，奖励也就会相应提高。  漏洞奖励是给发现并提交有效漏洞的矿工的额外奖励，奖励金额可以根据漏洞的严重程度进行评估。漏洞的严重程度通过通用漏洞评分系统（Common Vulnerability Scoring System，CVSS）来判定，CVSS是一个开放的行业标准，用于评估计算机系统安全漏洞的严重性，它的分值范围是0.0~10.0，数字越大代表漏洞的严重程度越高，漏洞奖励与成正比。根据不同矿工在本轮执行窗口中贡献的执行路径的占比来计算路径贡献度，路径奖励和成正比。最终奖励的计算为基本奖励、漏洞奖励和路径贡献奖励三者之和。  2、系统设计和实现  2.1 系统架构    本课题将实现一个基于Proof-of-Fuzzing的分布式漏洞挖掘系统，系统架构如图x所示。系统的使用者包括程序提供方和漏洞挖掘者（也就是矿工）。程序提供方即漏洞奖励计划中提供待测程序的公司（Google、Microsoft、Apple等），这些提供方发布待测程序后，矿工们可选择想测的程序开始Fuzzing，所有人都一起经过一个执行窗口后，Fuzzing被挂起，每位矿工向整个网络广播自己在该窗口中的payload和，其他矿工在接收后即进行验证和评估，对记账权和贡献度达成共识，产生本轮执行窗口的winner和漏洞排名，由winner向区块链中新增区块，程序提供方根据生成的排名分发漏洞奖励。  2.2 功能模块设计  整个系统分为6个模块：待测程序发布模块、模糊测试模块、消息发送接收模块、工作量和漏洞验证模块、区块生成和同步模块以及奖励发放模块。每个模块具体包含的功能如下所示：   1. 待测程序发布模块：由程序提供方发布需要漏洞挖掘的程序源码、覆盖率目标等； 2. 模糊测试模块：矿工获取任务，在执行窗口中进行模糊测试，生成运行的测试用例、执行路径、是否发现漏洞、覆盖率等信息，计算Hash后对它签名； 3. 消息发送和接收模块：矿工每次执行窗口结束后广播本轮结果（签名和payload），其他矿工校验和评估后将消息继续广播 4. 工作量和漏洞验证模块：矿工校验其他人发送的签名，是否是enclave签名，验证payload是否正确，漏洞是否真实有效，对记账权的归属达成共识，生成不同矿工在本次执行窗口中的路径占比贡献度； 5. 区块生成和同步模块：计算出的Hash第一个落在给定区间内的矿工可以获得记账权，由该winner生成新区块，其他人同步新区块，同时动态调整该区间； 6. 奖励发放模块：给挖矿成功的矿工发放奖励，根据计算出的模糊测试难度、发现的漏洞的严重程度、路径贡献度给成功挖掘漏洞的矿工发放奖励。 | | |
| **三、下一步的工作计划** | | |
| **四、指导教师评语**    签名： 日期： 年 月 日 | |
| **五、中期考核结果**（请在相应等级后的“（）”内打“√”，并给出评语和修改意见）  **考核结果：** 优秀（ ）; 合格（ ）; 不合格（ ）  **评语：**  **修改意见：**  中期考核专家小组签名：  组长  成员  时间： 年 月 日 | |