揭开刚性智能合同的组成和代码重用

孙凯然
南洋理工大学
新加坡、新加坡
sunk0013@e.ntu.edu.sg

徐正子*
南洋理工大学
新加坡、新加坡
zhengzi.xu@ntu.edu.sg

刘成为
南洋理工大学
新加坡、新加坡
chengwei001@e.ntu.edu.sg

李凯旋
华东师范大学
上海，中国
kaixuanli@stu.ecnu.edu.cn

刘洋
南洋理工大学
新加坡、新加坡
yangliu@ntu.edu.sg

摘要

随着稳固性智能合同的发展越来越受欢迎，对第三方软件包等外部来源的依赖也在增加，以降低开发成本。然而，尽管使用外部源为开发带来了限制性和e伍ciency，但由于缺乏针对标准化方法和源的包管理器，它们也可能使确保下游应用程序安全性的过程复杂化。虽然以前的研究只关注代码克隆，而没有考虑到它是如何引入外部组件的，但智能契约的组成及其特征仍然令人困惑。

为了弥补这些差距，我们对超过35万个可靠性智能契约进行了一项实证研究，以揭示它们的组成部分，进行代码重用分析，并识别流行的开发模式。我们的研究表明，一个典型的智能合同包括大约10个分包合同，其中超过80%来自外部来源，这反映了对第三方软件包的重大依赖。对于自开发的分包合同，大约50%的分包合同的唯一功能少于10%，这表明在函数级别上的代码重用也很常见。对于外部分包合同，尽管大约35%的分包合同是为标准或协议提供模板的接口，但也发现了分包合同类型在使用方面的不一致性。最后，我们提取了61种经常重用的开发模式，为安全和e伍cient智能合同开发提供了有价值的见解。

中国化学会概念

. 一般的和参考的→实证研究；。软件及其工程设计的→软件库和存储库。

关键字

智能合约组合、代码重用、开发模式

*徐正子是通讯作者

ESEC/FSE '23,12月3—9,2023年，旧金山，美国

ACMISBN979-8-4007-0327-0月23日/12。
https://doi.org/10.1145/3611643.3616270

ACM参考格式：
孙冠、徐正子、刘成伟、李凯旋、刘杨。2023.揭开刚性智能合同中的组合和代码重用的神秘面纱。在第31届ACM欧洲软件工程联合会议和软件工程基础研讨会（ESEC/FSE '23）的会议论文集上，2023年12月3—9，旧金山，加利福尼亚州，美国。ACM，纽约，纽约，美国，12页。
https://doi.org/10.1145/3611643.3616270

1介绍

智能合同是建立在区块链技术之上的一种特殊类型的程序，用于自动化、验证和强制执行双方之间的协议谈判。近年来，智能合约的发展获得了巨大的吸引力，因为它有潜力彻底改变金融服务，供应链管理和物联网（IoT）[23]。为了减轻智能合约的发展，可靠性[1]已被提出并广泛应用于主流区块链平台的主要语言。例如，以太坊[18]，作为最流行的开源区块链平台之一，通过使用稳固的智能合约，已经拥抱了去中心化应用程序（DApps）的多样化生态系统。

然而，随着DApps的复杂性的增长，智能合约的发展(i。e.，智能合同越来越依赖第三方建立的分包合同（如第2节所述的合同级代码块），以避免重新发明车轮和减少开发速度。虽然第三方分包合同的重用也带来了一个新的威胁，就像传统Web2生态系统中的供应链攻击一样，但来自第三方合同的漏洞和法律可能会进一步崩溃下游应用程序，特别是在缺乏成熟的依赖管理解决方案的早期智能合同开发的情况下。据2022年2月[14]报道，多链[11]提供的分包合同中有两个关键漏洞涉及7962个用户地址，总共利用了超过300万美元。在这种情况下，及时披露隐藏在被重用的第三方分包合同中的潜在安全风险对于保证Web3的安全至关重要。

然而，多样化和灵活的第三方分包合同重用使得精确管理在智能合同开发过程中引入的第三方分包合同变得非常重要。尽管最常用的第三方分包合同以NPM包的格式发布，而且它们可以由NPM介绍

在依赖关系方面，仍然有各种方法和实践来重用现有的第三方分包合同(i。e.，简单地通过来自Github [2] Repos、IPFS网关[3]、群集网关[4]等的url声明依赖关系)。尽管很方便，但这种多样性和可限制性也使下游应用程序的安全保证复杂化，因为url可能是不可信任的。此外，尽管NPM提供了依赖项的版本管理，但跟踪导入的分包合同的版本仍然具有挑战性，特别是在链上跟踪之后。这使得它可以获得标准化的解决方案，以识别、管理和修复存在于依赖项中的已知漏洞，如软件组合分析工具[48]对Web2应用程序所做的那样。除了外部进口外，许多开发商还倾向于直接将现有的分包合同克隆到他们的合同中。据Pierro等人[41]透露，79.1%的智能合同包含重复的代码。这样的代码克隆进一步挑战了对被重用的第三方分包合同的管理。

为了揭示智能合约发展中的奥秘，人们进行了各种探索性研究。一些研究者研究了现有智能契约的特征，如不同类型智能契约的设计模式[25]和源代码复杂度[40]，而一些研究者[34,35]研究了智能契约在不同粒度下的智能契约的代码重用。然而，大多数现有的工作只关注于单一粒度的智能合同，如分包合同或函数，而忽略了在开发过程中是如何引入这些元素的。这可能会限制对智能合约的理解，并导致对智能合约的不完整或不准确的结论。在这类组合中缺乏探索，也损害了对整个生态系统中智能契约重用的进一步管理和治理的衍生指导。为了弥补这些差距，我们从以太坊区块链上的契约分解的角度进行了实证研究来研究分包合同的重用情况。

本研究的总体框架如图1所示。我们首先在2021年1月和2023年1月期间从以太扫描[7]收集了超过35万份智能合约，即以太坊领先的区块链浏览器和分析平台。此外，为了增强我们对用于智能合同开发的现有第三方包的理解，并进一步扩展我们的数据集，以便后续的代码重用分析，我们从收集的合同的导入报表中汇编了来自NPM [9]和GitHub [2] Repos的353个第三方包的列表。为了确保后续代码重用分析的健壮性，我们首先验证了这些契约，并使用ANTLR-4 [6]将它们解析为JSON文件。然后，我们通过进行合同级代码克隆检测来检查并过滤掉了没有或轻微修改的合同。经过这一过程，296236份不同的合同供进一步研究。我们的实证研究建立了对合同组成和代码重用实践的全面观点，并考虑了引入外部组件的法规。具体地说，我们首先根据合同的起源以及如何引入合同，将合同分解为各种分包合同。这允许我们分离自开发的和外部的分包合同，以进行深入的代码重用分析。对于自开发的子合同，我们研究函数级代码重用，总结了通常重用的函数和需要自定义的函数。对于外部分包合同，我们研究了经常重用的分包合同和克隆的使用情况

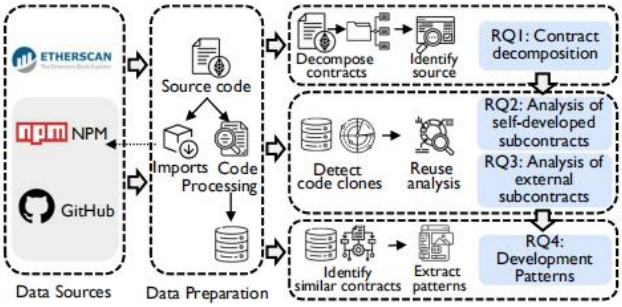

图1：本研究的框架

跨第三方包。最后，基于这种理解，我们提取并分析了经常重用的开发模式。

在整个实证研究过程中，我们得出了以下一些结论。例如，①超过80%的分包合同来自外部来源。最大的已确定的外部来源是NPM，占总外部分包合同的72%以上。②尽管稳固性允许通过在导入语句中指定包名所需的分包合同来包含NPM依赖关系，但超过56%的克隆分包合同来自NPM包，这表明依赖关系管理存在混乱。③函数的代码重用是常见的，因为大约50%的自我-已开发的分包合同的独特功能少于10%。随着自行开发的分包合同中功能数量的增加，独特功能的百分比趋于减少。这突出显示了开发人员在开发智能合约时依赖于预先存在的代码块的趋势。④虽然大约35%的外部分包合同是接口，通常用作协议模板，但我们发现分包合同类型的使用不一致，这给合同管理带来了挑战。⑤我们确定了61种流行的开发模式，其中68.75%是用于令牌创建的。虽然这些模式可以大大降低开发成本，但如果在这些模式中发现漏洞或安全漏洞，将存在风险。因此，以这种模式进行管理以持续监测和评估其安全性是很重要的。

总之，本研究的主要贡献如下。

- 我们提供了对在智能契约中引入分包合同的常见方法的全面分析，并通过进一步的代码重用分析，以确定最频繁重用的代码块及其目的。
- 我们对353个已确定的第三方智能合同开发软件包进行了全面的研究，分析了它们的使用频率和功能特性。
- 我们总结并评估了61个常用的智能契约开发模式，这些模式有可能用于低代码开发。
- 基于我们的研究结果和信息，我们提供了一个深入的讨论了在管理智能合同的外部组件和开发模式方面所面临的挑战。

2背景

在本节中，我们将提供本研究中使用的术语的解释和解释。

✦ **智能合同和分包合同。**在本研究中，智能契约是指共享相同地址的代码块，而分包合同是指合同级别的代码块。根据可靠性文件[12]，根据它们的使用方式确定了四种分包合同：

•接口：分包一个标准或协议
方法实现。
•摘要：用一些基本结构
方法实现。
•合同：完整的和可执行的分包合同。•库：提供分包合同，以便为其提供可重用的代码块
没有存储空间的普通操作。

为了进一步澄清这一点，在我们的研究中，分包合同是指如上所述的接口、抽象合同、合同和库。这用于区分模块化的合同级代码块（分包合同）和最终部署在区块链（合同）上的复合代码。本质上，这些分包合同可以组合成一个准备部署的完整合同。

✦ **以太坊[18]和以太扫描[7]。**以太坊是一个开源的区块链平台，通过智能合约支持去中心化的应用程序。在我们的研究中，我们选择了两个网络：

•以太坊主网：themainblockchainnetworkforEthereum
区块链涉及真实的交易。
•一个以太坊区块链的测试网络
用于智能合同的测试和开发。

以太扫描是以太坊区块链的区块链资源管理器，用于提供有关块的详细信息，包括交易、地址和可用的智能合约的源代码。✦ **EIPs.**EIPs（以太坊改进建议）[8]是以太坊网络的正式建议，有助于以一种良好组织的格式记录可能实现的标准化协议。由于EIPs经过审计并期望安全重用，我们将EIPs作为评估智能合同、第三方包和开发模式安全性的基本方法。✦ **克隆类型。**四种克隆类型对软件系统[42]中的代码克隆进行分类。在我们的研究中，我们关注的是2型克隆，它们是语法上相同的代码，具有不同的标识符或文字，并使用1型克隆，仅在空白或注释中使用不同的精确代码副本进行验证。我们排除了类型3和类型4，它们集中于主要的变化和语义相似性。虽然智能合约相对简单和标准化，但3型和4型克隆可能会引入大量的错误相似性，从而影响结果的总体准确性。

3实证研究

.13个研究问题

我们的研究问题的结构如图2所示。为了探索考虑依赖关系引入的契约组成和常见开发实践，我们首先将合同分解为分包合同，确定它们的起源和引入方法。然后，我们分别对来自不同来源（即自我开发或外部）的分包合同进行深入的代码重用分析，并提供对其个体特征的见解。最后，为了更好地理解合同的组成和代码重用，我们提取了常用的开发模式。

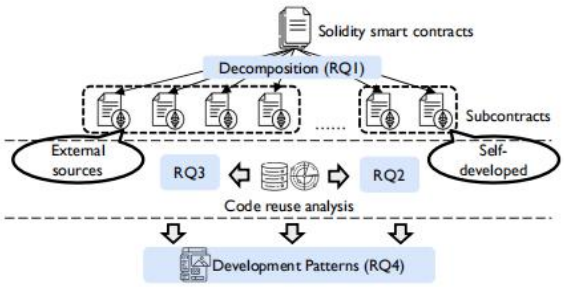

图2：研究问题之间的关系

3.1.1合同克隆分解（RQ1）：将分包合同引入智能合同及其来源的常用方法是什么？确定合同的组成部分，并量化引入分包合同的各种方法的使用情况。勘探结果是后续分析的基础。

3.1.2自开发分包合同分析（RQ2）：自开发分包合同的特点是什么？了解自开发的分包合同中代码重用的特征和程度可以突出定制的共同领域，从而为定制的开发实践和工具提供信息。

3.1.3外部分包的分析（RQ3）：外部分包的特点是什么？分析顶级重用的外部分包合同和第三方包的特点和常见应用，以帮助更好的合同开发指导和第三方包管理。

3.1.4开发Pa计erns（RQ4）：在智能合同开发中常用的开发pa计erns是什么？提取和分析智能合同开发中经常重用的开发模式，这些模式反映了常见的使用和安全实践，从而告知最佳实践，并可能暗示更安全的智能合同开发框架的设计。

3.2数据集准备

图3概述了我们的数据准备过程。我们从以太扫描[18]、NPM [9]和GitHub Repos [2]收集智能合同源代码。为了将收集到的代码规范化并转换为后续分析的合适格式，我们在分析之前还进行了一系列的数据预处理。

3.2.1合同收集。在我们的实证研究中，我们首先从以太扫描[7]收集了以太坊主网上221309个部署的智能合约和谷歌测试网[15]上131207个合约的源代码。为了扩展我们的数据集和更好地跟踪分包合同起源，我们随后收集了用于智能合同开发的第三方包的源代码，其中的列表是通过对来自主数据集的导入语句执行基于语法的提取来编译的。最后，我们成功地确定了用于智能合同开发的337个NPM软件包和33个GitHub存储库的使用情况。

3.2.2合同预处理。在收集智能合约的源代码后，依次采取以下三个步骤为后续实验的合约。

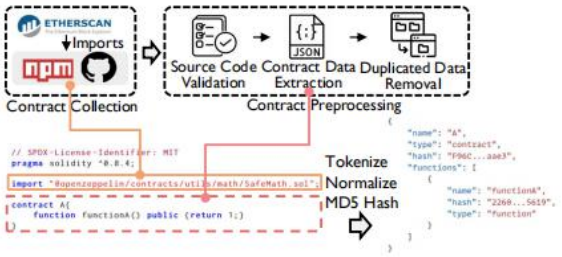

图3：数据准备工作概述

源代码验证。由于我们的主要合同收集来自以太扫描[7]，这是一个用户可以上传源代码的开放平台，因此我们进行了两阶段的验证，以确保数据集的完整性和完整性。首先，我们确认合同存储在一个文件中，其中包含所有必需的元详细信息（即地址、联系人名称和创建日期）。其次，我们通过检查每个分包合同最外层的花括号内的内容来验证每个合同是非空的。验证后，大约3%的合同被注销，其中以太坊主网约6241份，高利测试网约2240份。

合同数据提取。为了确保合同数据的可访问表示，以便进一步分析，我们使用ANTLR4 [6]开发了一个稳健智能合同解析器。解析器首先将智能契约转换为令牌流，为每个令牌提取两种类型的信息：其原始文本和令牌类型标识器。然后，我们提取代码块，如分包合同和函数
在第2类标准化后计算每个块的MD5散列（即，删除空格和注释，并重命名变量），以方便进一步分析。在此之后，每个合同将被保存为一个JSON文件，其中包含所有提取和计算的信息，如图3所示。一个完整的输出文件的例子可以在我们的网站[17]上找到。**重复的合同删除。**本研究的目的是确定在智能合同开发中的常见做法。但是，重复合同的多个副本只做微小更改或没有更改可能会扭曲结果。为了解决这个问题，我们根据2型规范化后的合同级别MD5散列对合同进行了分组。这确保了同一组中的契约只能有微小的差异，比如变量名。结果，7617组被确定有一个以上的合同。然后，我们对这些小组进行了人工检查，并确定有90.17%是由相同的团队创建的，详见第3.3节。因此，为了避免类似合同的过度代表，并增强我们的数据集的多样性，我们决定只保留每个合同组的最早版本。

最后，我们为以太坊主网保留了189,229个不同的合同项目，为Goerli测试网保留了107,034个项目。处理后的数据被保存到数据库中，以用于以后的代码重用分析。

.33.合同克隆体分解（RQ1）

3.3.1实验设置。为了深入了解合同的组成和开发中的常见实践，我们通过评估合同级代码克隆和量化将分包合同引入合同的各种方法来开始我们的研究。这为后续的分析奠定了基础。

最初，对于合同级克隆分析，我们检查了在数据预处理过程中识别的重复合同组，详见第3.2.2节。一个由三名智能合同审计师组成的团队对366组随机抽取的样本进行了人工检查。样本量由科克伦方程[29]和总体校正[36]共同确定。在……期间
在进行检查时，每个审计员都独立地评估了重复背后的原因。然后进行集体讨论，直到在最初意见不同的情况下达成协议。

在获得了对合同克隆的全面了解后，我们将重点转向了合同的组成。认识到以太扫描[7]在已发布的合同源代码中包含了外部导入的分包合同，我们将引入外部分包合同的方法集成到我们的分析中。在基于预先计算的MD5散列识别分包合同克隆后，我们根据它们的引入方法对它们进行了分类，量化了这些方法的流行程度，并追踪了外部引入的分包合同的起源。

3.3.2结果和讨论。我们的合同和分包合同的信息在以下小节中详细介绍。**合同克隆的分析。**以太坊主网和高尔利测试网的合同克隆率分别为14.50%和18.42%。其中，以太坊主网共有7,617个合同组在数据预处理阶段被确定为重复的合同组(i。e.，每个组内的合同具有相同的MD5哈列)。通过对样本组进行人工检查，我们发现约89.16%的样本组是由版本迭代引起的，而3。31%与发电机的使用有关。其余的组似乎也遵循了预先确定的模板，然而，由于缺乏明确的声明或可用的在线文档，这背后的确切原因仍然不清楚。另一方面，90.17%的团队是由同一团队或个人部署的合同组成的。这是通过调查作者在源代码中的记录和在以太扫描[7]上的合同创建者的细节来验证的。由于检查的结果，我们可以一致认为，确定的群体对合同开发的多样性没有贡献，因此，被排除在我们的分析之外是可以接受的，以避免类似合同的过度代表。

为了更深入地了解契约克隆的模式和特征，我们分析了群体规模的分布。尽管确定的组数量很多，但只有2个组包含超过1000份合同，只有40个组超过100份合同。这表明，高容量的复制在数据集中并不常见。事实上，这种广泛的复制大多与令牌生成器的使用有关。例如，拥有超过1,000个合同的两个组都是令牌生成器的产品。最大的组为2,849是由[20][20]发布的ERC1155令牌创建者生成的，而第二大的组为1,105是由于使用了[30]发布的ERC20令牌生成器。除此之外，我们注意到存在一种合同，它是按需创建和执行的，只有微小的变化。这种合同类型的一个例子是锁定支付，它是一个令牌锁，只在每次部署中更改委托人的地址，近藤等人讨论了这一问题。[35].在光谱的另一端，我们观察到由于版本迭代的组通常有少于5个契约。

表1：在使用各种方法中引入的分包合同的平均百分比

	自行开发的	导入	克隆的
以太坊主网	16.43%	22.62%	60.95%
Goerli测试网	17.68%	39.82%	42.51%

此外，我们还量化了在检查过程中这些合同组的目的。超过80%的类群组是针对具有各种功能的令牌的。具体来说，66.10%的组是ERC20标记，22.38%的组是ERC721标记。

发现1：在7617组重复合约中，超过80%的组与代牌有关，ERC20代牌占组的66.10%。大规模的合同复制通常来自于使用发电机。此外，90.17%是由同一团队创建的合同组成的。

分包克隆分析。因为集市合同平均包括约10个分包合同。这些分包合同可以根据它们对合同的引入方式分为三类：①导入分包合同。由导入语句引入的分包合同，引用外部源，如NPM包或URL。**②克隆分包合同。**从另一个外部源复制的分包合同，包括早期部署的智能合同和第三方包，而没有引用该源的导入声明。**③自行开发的分包合同。**一种既不从外部来源导入也不克隆的分包合同，包括早期部署的智能合同和第三方软件包。在这三类合同中，①和②被认为是外部分包合同。在我们的案例中，我们首先通过检查每个合同的导入报表来约束导入的分包合同。接下来，我们进行了一个分包合同级的代码克隆检测，以识别克隆的分包合同。最后，我们将那些未导入或克隆的合同分类为自我开发的分包合同。

为了更好地理解合同的组成，我们随后确定了上述三种类型的分包合同在两个以太坊主网的智能合同中的平均分布
测试，如表1所示。结果表明，克隆是在智能合同开发中引入分包合同的最普遍的方法，特别是对于以太坊主网上的合同，其中60.95%的分包合同是克隆的。此外，只有不到20%的分包合同被归类为自我开发，这表明当前智能合同的开发严重依赖于外部解决方案。尽管使用外部解决方案有助于降低开发成本和提高代码质量，但它也可能带来潜在的安全风险，并使长期维护代码变得更加困难。

此外，我们还设法追踪了一些通过不同的方法引入的外部分包合同的起源。对于导入的分包合同，我们关注两种类型的导入语句，以介绍来自外部来源的分包合同：使用导入语句导入NPM包的名称；②导入URL，如Github [2]、IPFS网关[3]和Swarm网关[4]。在我们的数据集中，我们只发现使用了引用GitHub Repos的url，它占0.13%

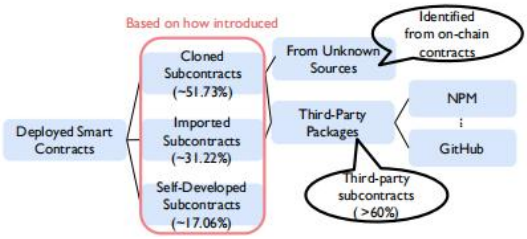

图4：已部署的智能合约的分解图

这表明在智能合同开发中，这两种导入报表的使用不平衡。事实上，大约51%的GitHub Repos已经发布了NPM软件包。这表明，NPM在智能合同开发中充当了大多数第三方包的包管理器。但是，NPM并没有为管理稳健性库或依赖项进行优化。例如，①缺乏版本控制和部署管理。NPM并不能帮助分包合同的版本控制和各种环境中的智能合同的部署。在合同在线发布后，重要的分包合同的版本可能会丢失，因为NPM生成的记录这些信息的日志文件不会一起发布。②缺乏安全性。NPM没有为管理可靠性库提供必要的安全措施，而这对于智能合同对于确保链上信息的可靠性至关重要，如DApps的完整性和有价值资产的保护。③缺乏兼容性。NPM最初是为管理库而设计的，因此，它并不直接与以太坊兼容。要在稳固智能合同中使用NPM包，需要额外的步骤，例如在导入语句中的NPM包中指定导入分包合同的路径。

而对于克隆的分包合同，我们发现超过56%的克隆分包合同来自以太坊主网和Goerli Testnet中的第三方软件包。这表明了一种常见的做法，即通过克隆而不是使用导入语句来重用来自第三方包的分包合同。在开发者看来，一些可能的原因可能是：①缺乏意识。
开发人员可能对可用的软件包管理工具缺乏认识或熟悉程度。②缺乏信任。开发人员可能对现有的智能合同开发包管理工具缺乏信任，因此更喜欢直接从他们可靠的来源复制和粘贴。②用户不友好。开发人员可以在第三方包中识别分包合同的路径，这是在di伍cult中的导入声明所需要的。③需要定制。开发人员可能希望对分包合同进行修改，但使用导入语句是不允许进行修改的。

发现2：我们在图4中总结了智能合约的组成。智能合约通常包括三种类型
分包合同：克隆（51.73%）、进口（31.22%）和自开发（17.06%）。然而，超过56%的克隆subcontractscanbeimportedfromthird-partypackages.此外，我们发现NPM包含了大多数用于智能合同开发的第三方软件包，尽管它的使用并没有得到优化。

表2：顶级重复使用的外部分包合同

秩	已知来源						未知源					
	以太坊主网			Goerli测试网			以太坊主网			Goerli测试网		
	名称	类	%	名称	类	%	名称	类	%	名称	类	%
1	IERC20	界面	39.66%	上下文	摘要	25.75%	IUniswapV2工厂	界面	25.79%	ERC721创造者	合同	5.30%
2	上下文	摘要	34.44%	国际会计准则165	界面	22.36%	安全数学	图书馆	22.20%	VRF协调器V2接口	界面	1.26%
3	国际会计准则165	界面	28.47%	IERC721接收器	界面	16.70%	上下文	摘要	21.76%	ERC721创造者	合同	1.03%
4	IERC721接收器	界面	25.04%	电子邮件165	摘要	15.12%	IUniswapV2路由器02	界面	12.58%	一厂	界面	0.97%
5	电子邮件165	摘要	21.90%	地址	图书馆	14.11%	自己的能力	合同	12.39%	安全数学	图书馆	0.64%
6	IERC721元数据	界面	20.51%	IERC721元数据	界面	12.63%	ERC721创造者	合同	10.51%	上下文	摘要	0.58%
7	乐队的弦乐器部	图书馆	20.39%	自己的能力	摘要	11.28%	IUniswapV2路由器02	界面	8.70%	ERC1155创造者	合同	0.48%
8	国际会计准则721	界面	17.32%	IERC20	界面	10.55%	自己的能力	合同	7.19%	IERC20	界面	0.40%
9	地址	图书馆	17.08%	国际会计准则721	界面	10.30%	安全保障	图书馆	4.28%	上下文混合在	摘要	0.36%
10	自己的能力	摘要	16.27%	乐队的弦乐器部	图书馆	9.19%	安全信息	图书馆	4.20%	价格转换器	图书馆	0.35%

(1)表中的%表示分包合同占总合同的比例。

.43自开发分包合同分析（RQ2）

3.4.1实验设置。为了评估自开发的分包合同中的代码重用，我们实现了一个两步分析。首先，我们在分包合同中进行了功能级克隆检测，以确定合同开发中代码重用的程度。其次，我们量化了每个分包合同中独特函数的百分比，为理解开发人员与合同复杂性相关的代码重用行为提供了见解。

此外，为了理解重用函数的常用用法，我们通过识别函数名（例如，mint和转移）中的关键字、可识别的函数的起源以及函数类型(i。e.，函数、修改器、事件和构造函数)。我们还对最频繁重用的函数进行了案例研究，以更好地理解函数级代码重用的趋势。

3.4.2结果和讨论。函数重用的结果
自开发的分包合同如图5和图6所示。**功能克隆分析。**结果显示了在以太坊主网和Goerli Testnet之间的自开发分包合同的一致函数重用模式，如图5所示。大约50%的已标识的自开发分包合同包含的唯一函数少于10%，这表明在函数级别上重用代码也是智能合同开发中的一种常见做法。
尽管如此，在对自己开发的分包合同中的函数使用频率进行编目时，我们发现以太坊主网和Goerli测试net分别只有27.10%和28.05%的函数被重用。这表明通常重用的函数往往集中在一小组函数上。

此外，在探索自开发分包函数中唯一函数的数量与总函数计数之间的相关性时，我们发现函数较少的自开发分包函数具有较高的唯一函数比例，如图6所示。其中，功能少于10个的自开发分包合同的独特函数比最高（40.53%）。72个自开发的分包合同被确定具有100%唯一的功能，其中56.94%只有一个功能。然而，由于在自我开发的分包合同中需要更多的功能，独特功能的百分比就会下降。这表明，开发人员倾向于在开发过程中重用来自外部源代码的更多代码块，而不是在所需的功能变得更加复杂时从头开始编写所有内容。而这样的

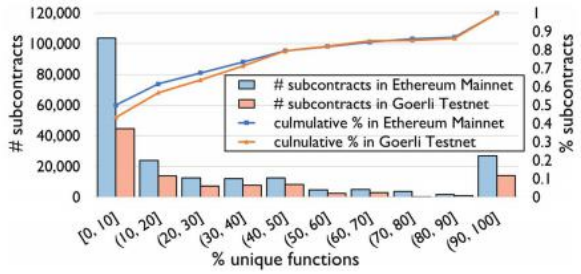

图5：以太坊主网和Goerli测试网在自开发的分包合同中的独特功能

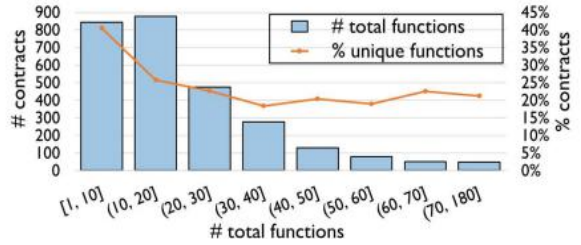

图6：在智能合约中自开发的分包合同中，唯一功能相对于总功能的平均百分比

代码重用通常是为了减轻潜在的漏洞，外部功能的安全可能无法得到保证，因为只有大约8%来自成熟的包。此外，在分包合同中集成这些功能可能具有挑战性，特别是当需要考虑函数之间的交互时。

查找3：代码重用是函数级别上的一种常见做法，特别是当所需的功能很复杂时。此外，我们注意到跨分包合同一致重用一小功能的集合，在开发过程中引用了常见的功能需求。

功能使用分析。大多数独特函数和重用函数都是标准函数。很大一部分独特的函数是构造函数，在以太坊主网中占21.56%，在Goerli测试网占13.64%。构造函数是一种特殊类型的函数，它在创建合同时被调用来初始化合同并设置初始状态。这意味着

合同初始化过程通常需要特殊的定制，以满足单个智能合同的特殊需求。

另一方面，重用的函数通常表示通常需要的功能。我们发现，这些函数中超过20%是获取器和设置器，它们被广泛用于访问和修改合同的当前状态。与铸造和转移相关的功能也占重用和独特功能的重要部分，共占10%以上。

事实上，智能合约中独特功能和重用功能之间的平衡似乎是定制操作和广泛使用的标准功能的混合。随着智能合同开发的不断发展，理解这些独特和重用函数的相互作用和适当应用将对优化合同e伍ciency和效率至关重要。

查找4：唯一函数和重用函数主要是标准函数，构造函数构成唯一函数的显著部分，表明在合同初始化过程中频繁的定制需求。

.53外部分包合同分析（RQ3）

3.5.1实验设置。为了理解占总分包合同比例为80%的外部分包合同的意义和含义，我们从三个方面进行了实验。**①对分包合同类型的分析。**为了理解外部分包合同的常见用法，我们量化了四种类型的分包合同的使用情况（即合同、抽象、接口和库）。此外，我们还检查了分包合同类型在使用中的不一致性。**②对分包合同使用频率的分析。**为了深入了解外部分包合同的使用趋势，我们根据分包合同在数据集中的频率对其进行了排序，然后使用重用最多的合同示例进行了案例研究。**③对第三方软件包的概述。**为了理解现有第三方包的使用情况，我们分析了包的使用频率，并通过对所有已识别的第三方包进行分包包级克隆检测，发现了跨包的潜在代码克隆。

3.5.2结果和讨论。我们将在下面的三个小节中分别介绍这三个阶段的信息。

分包合同类型分析。这四种分包合同的使用频率详见表3。在四种分包合同中，接口在以太坊主网（36.59%）和GoerliTetestnet（34.85%）中所占比例最多。这是合理的，因为接口通常被用作成熟的标准（如EIPs）和协议(如。Uniswap [19]).因此，接口的使用通常表明可以重复使用已建立的标准和协议，以降低开发成本，同时确保合同之间的一致性和互操作性。

在检查合同类型占总分包合同的外部分包合同时，我们观察到分包合同类型的使用不一致。我们在本文中提供的一个案例研究是关于安全数学方法的，一个用于公共安全数学操作的实用程序库。在智能契约开发的上下文中，库是指一个可重用的代码块多个契约在不存储的情况下执行公共任务，而契约是执行自身功能的自包含逻辑单元

表3：各种类型的外部分包合同的百分比

	接口	合同	摘要	图书馆
以太坊主网	36.59%	34.86%	16.11%	12.45%
Goerli测试网	34.85%	34.28%	16.99%	13.88%

与存储。因此，安全数学应该在这里被定义为库。然而，在超过26,000起案件中，有10%的人被错误地认定为合同。我们手动检查了作为合同定义的分包合同，并注意到SafemMath是代码注释中的一个库的声明。类似地，IERC20应该是提供EIP20 [13]定义的接口，但在某些合同中被定义为没有函数实现的合同。事实上，超过93%的已鉴定的案件涉及到合同的类型。

这些不一致可能源于开发人员对分包合同类型使用的混淆，这可能导致兼容性问题、维护中的di伍culties和增加的漏洞风险。契约类型的灵活使用可能会导致这种混淆，因为它可以模拟接口、抽象和库的功能。事实上，超过93%的已鉴定的案件涉及到合同的类型。尽管各种类型的分包合同有望帮助开发人员组织更多的代码e伍ciently，同时提高代码块的可维护性和可读性，但分包合同类型使用的不一致性给智能合同管理带来了更多的挑战。

发现5：虽然35%的外部分包接口表明良好的实践遵循完善的标准和协议，分包合同的使用合同的类型，占总分包合同的35%，揭示了使用分包合同类型的不一致，这给智能合同管理带来了挑战。

分包合同使用频率的分析。顶级重用的外部分包合同的结果如表2所示。在这里，通过与已知来源的分包合同，我们是指从我们收集的第三方软件包中采购的导入分包合同和克隆分包合同。而具有未知来源的分包合同是克隆的，但其来源尚未确定的分包合同。由于页面的限制，我们在本文中只提供前10个被重用的分包合同，其余的将在我们的网站[17]上找到。

结果表明，与未知来源的分包合同相比，第三方分包合同的使用更频繁，特别是在Goerli Testnet中，未知来源最常用的分包合同只出现在5.3%的合同中，低于第10个最常用的第三方分包合同。此外，在第三方分包合同中IERC20和IERC721的频繁使用突出了智能合同开发中最流行的应用之一是创建包括可替代（ERC20）和不可替代（ERC721）的令牌。

此外，如表3所示，虽然外部分包合同类型的分包合同占35%左右，但大多数是不经常重用的分包合同。在前1000个被重用的外部分包合同中，具有抽象和库类型的分包合同占总数的50%，而接口又占32%。这在智能契约开发中是一个很好的实践，因为与接口、库、

表4：顶级可重用的第三方软件包

	导入		克隆		合计	
以太坊主网	@openzeppelin/contracts @openzeppelin/contracts-upgradeable erc721a 安全帽 @uniswap/v2-外围	29.59% 4.25% 2.83% 1.15% 0.55%	@openzeppelin/contracts @aragon/应用程序-Vault 泳池合同erc721a	43.90% 3.52% 1.52% 0.97% 0.89%	@openzeppelin/contracts @openzeppelin/contracts-upgradeable erc721a @aragon/应用程序-Vault 平底锅交换自由	73.46% 4.50% 3.72% 3.52% 1.52%
Goerli测试网	@openzeppelin/contracts @openzeppelin/contracts-upgradeable 安全帽 erc721a @uniswap/v3核	47.97% 10.40% 5.78% 3.54% 1.05%	@openzeppelin/contracts @lair-sdk/contracts synthetix @thirdweb-dev/contracts @connext/nxtp合同	0.10% 0.04% 0.04% 0.04% 0.03%	@openzeppelin/contracts @openzeppelin/contracts-upgradeable 安全帽 erc721a @uniswap/v3核	48.07% 10.42% 5.78% 3.54% 1.05%

和抽象应该是被重用的代码块。我们希望将来能够提取或创建更多这样的可重用代码块，以用于安全的智能合约开发。此外，还观察到一些分包合同经常同时出现。例如，我们注意到IERC721和IERC165总是成对出现的。这是因为IERC721是一个标准的不可替代令牌开发的接口，它继承了IERC165，以确保接口之间的兼容性。这表明了发现智能合约的开发模式的潜力。

发现6：来自知名第三方的分包合同比来自未知来源的分包合同更频繁。最频繁被重用的分包合同是设计良好的具有接口、库和抽象类型，在前1000个被重用分包合同中占82%。

第三方软件包的概述。表4总结了第三方软件包的使用频率。结果突出了齐柏林[10]的广泛使用，这是安全智能合同开发包。在以太坊主网上，超过73%的合同引入了来自欧柏林飞艇包的分包合同。事实上，表2中这两个网络中重用的前10个第三方分包合同都来自OpenZeppelin软件包。除此之外，我们还确定了erc721a的频繁使用，这是一个探索ERC721代币分批铸造nft中气体优化的软件包。其他经常重用的第三方包要么用于测试(i。e.，硬帽[16])或令牌交换（即，统一交换[19]）。此外，由于第三方分包合同可以导入或克隆到智能合同中，我们也量化了通过不同方法引入的第三方包的使用，从那里我们注意到以太坊主网的合同倾向于克隆子包
而不是使用导入语句。除此之外，还强调了其他一些软件包的使用。开发/合同是网络应用，而煎饼交换库是联合交换的扩展，专注于安全和气体e伍ciency。

在检查第三方包的过程中，我们总结了以下三种类型的包：用于安全智能合同开发的①包。；②包或在特定协议下使用。用于实用程序和测试的③包。g.Hardhat [16]).在我们确定的第三方包中，25%的包遵循EIP，该标准表明以太坊[8]具有潜在的新特性或流程。其余的包要么在特定协议下使用。31%)或测试和效用（9.69%）。

此外，为了更好地了解第三方包中克隆的程度，我们还对所有已识别的第三方分包合同进行了分包级克隆检测，通过检测发现超过10个分包合同出现在19个库中，其中38.85%是接口。除此之外，大约90%的被重复使用的分包合同都来自开放齐柏林飞艇。事实上，我们收集的软件包中大约有36%是开放齐柏林飞艇的扩展。这也表明了欧柏林飞艇在智能合约开发中的巨大影响力。在检查其余10%与Open齐柏林无关的重用分包合同时，我们发现了另一组专注于去中心化加密货币交换的第三方包，Open齐柏林没有覆盖它。

由于智能合约开发严重依赖于Open齐柏林飞艇包，如果Open齐柏林飞艇的代码或系统受到损害，就会增加单点故障的风险。事实上，我们发现了一个关于open齐柏林艇的现有依赖管理问题。在检查齐柏林飞艇包时，我们注意到包的名称从齐柏林飞艇可靠性迭代到齐柏林飞艇/合同。这两个包名在当前的开发中仍然在使用，尽管它们链接到同一个包。此外，虽然@齐柏林/合同是当前的o伍cial名称，并已被广泛使用，但由NPM生成的package-lock.json文件中出现的名称仍然是开柏林固体的。如果基于package-lock.json文件进行漏洞检测，这可能会导致漏洞检测失败。除此之外，欧柏林柏林的普及表明，需要完善的分包合同来建立安全可靠的智能合同。

发现7：开放齐柏林[10]在智能合同开发方面对第三方分包合同有很大的影响。大约90%的进口第三方分包合同来自Open齐柏林飞艇，我们的约36%的识别第三方包是Open齐柏林飞艇的扩展。如果报告了与open齐柏林飞艇相关的漏洞，这可能存在风险。

.63开发模式（RQ4）

3.6.1实验设置。受智能契约开发中常见实践的启发，我们进一步研究了使用频繁项目集（FP）挖掘[31]的数据集中开发模式的存在，这是数据挖掘中一种成熟的技术。具体地说，每个合同都由其分包合同的预先计算的MD5哈希值表示，作为FP挖掘输入，以识别经常出现在一起的分包合同集。

为了验证所获得的分包合同集，我们与三名智能合同审计师进行了人工检查。每一位审计师

表5：已提取的开发模式

秩	模式详细信息	协议	上下文	问题	解决方案	频繁性	一天范围
1	IERC721元数据，ERC165，IERC721接收器，IERC721，字符串，地址，上下文	EIP721，电子邮件165	令牌创建	要拥有和交易的标准代币	使用ERC721接口与元数据和接收器接口，丰富的元数据和安全接收功能	63,316 (13.72%)	698天
2	IUniswapV2路由器02，IUniswapV2工厂，IERC20，上下文，可拥有	EIP20，乌尼斯瓦普	代币交换	需要控制所有权	使用UniswapV2的接口与可拥有的合同的权限	62,855 (21.34%)	699天
3	IERC721可举，IERC721元数据，ERC165，IERC721接收器，IERC721，字符串，地址，上下文	EIP721，电子邮件165	令牌创建	需要枚举所有已拥有的令牌	使用ERC721可枚举扩展来允许令牌枚举	53,679 (8.84%)	698天
4	IUniswapV2对，IUniswapV2路由器02，IUniswapV2工厂，IERC20，上下文，可拥有	EIP20，乌尼斯瓦普	代币交换	需要与一对具有所有权控制的特定令牌进行交互	使用UniswapV2的对和工厂接口与可拥有的合同的权限	14,869 (7.86%)	694天
5	地址，安全数学，IERC20，上下文，可拥有的	EIP20	令牌创建	需要安全的数学操作和所有权控制	使用安全数学库进行安全算术运算，使用拥有合同获得权限	14,012 (7.40%)	699天
6	存储插槽、代理、地址	1967年	代理合同	需要升级合同以存储当前状态	使用代理模式和存储槽一起在升级期间保持合同的状态	13,873 (7.33%)	604天
7	ERC721创建者，存储槽，代理，地址	电子邮件721	代理合同	需要升级ERC721令牌以存储当前状态	使用ERC721创建者来创建令牌和具有存储槽的代理模式以进行升级	12,532 (6.62%)	388天
8	IERC721元数据，ERC165，IERC721接收器，IERC721，字符串，地址，上下文	EIP721，电子邮件165	令牌创建	防止重新入神攻击	使用再入保护合同，以防止再入攻击	8,579 (4.53%)	684天
9	默克尔Proof，IERC721元数据，ERC165，IERC721接收器，IERC721，地址，上下文	EIP721，电子邮件165	令牌创建	需要创建和验证证明	使用商品证明合同来创建和验证证明	4,806 (2.54%)	645天

独立地评估这些分包合同集是否可以被视为模式。分歧通过小组讨论重新讨论，审计人员可以在讨论后改变他们的决定。然后，我们整理结果，按照大多数人获胜的规则确定开发模式。我们进一步验证了具有频率和日期范围的模式。频率量化了一个模式的出现，而日期范围表示从一个模式的第一次出现到最后一次出现的时间跨度。这些都为了解合同开发模式的流行程度和寿命提供了见解。

3.6.2结果和讨论。我们没有确定了61种开发模式，其中6种出现了超过1万次，另外一种
13个在我们的数据集中出现了超过1000次。表5列出了前9个开发模式，而已识别模式的完整列表可以在我们的网站[17]上找到。

在已识别的模式中，超过63%与标准令牌创建相关，特别是，74%涉及ERC721令牌。例如，第三种模式是创建可枚举的令牌，第8种模式是防止再入性攻击[37]，这是智能契约最常见的攻击之一，第9种模式是创建和验证证明。其他模式与代理合约和代币掉期有关，分别占20.31%和10.94%。模式的出现表明，令牌、令牌交换和代理契约的创建已经是高度标准化的，这反映了开发过程中的常见功能需求和体系结构范式。

每个模式的合同地址示例可以在我们的网站[17]上找到，以帮助理解模式在现实世界开发中的应用。例如，第二种模式是通过统一wapV2协议[19]促进ERC20令牌交换。为了使用这种模式，通常需要一个额外的定制分包合同来确定令牌的详细信息。清单1显示了一个示例，其中定义了令牌名称、符号和其他相关信息。事实上，这可以通过低代码开发很容易地处理。由于这个定制的分包合同通常遵循类似的模式，因此用户提取的信息可以集成到一个模板中，并与预先设计的模式合并在一起。

此外，我们还对Goerli Testnet的智能合同进行了同样的分析。除了在以太坊主网中已经确定的开发模式外，Goerli Testnet还包含了来自在线教程的开发模式。例如，[21]是一个由在线课程提供的简单存储模板，它在我们的数据集中出现了超过800次。虽然使用模式可以提高代码质量并减轻潜在的漏洞，但确保模式的安全性却至关重要。

```
语用固度^0 .0；.8
合同无路径是上下文，IERC20，可拥有的{
    对uint256使用安全数学；
    字符串私有常量_name =“无路径”；
    字符串私有常量_符号=“无路径”；
    ……
    构造函数（）{
        _rOwned [_msgSender ()] = _rTotal ；
        IUniswapV2Router02 _uniswapV2Router =
            IUniswapV2Router02 (0
            x7a250d5630B4cF539739dF2C5dAcb4c659F2488D) ；
        uniswapV2Router = _uniswapV2Router ；
        uniswapV2Pair = IUniswapV2Factory (_uniswapV2Router .
            工厂（）)。创建图对（地址（此），
            _uniswapV2Router .WETH ()) ；
        _isExcludedFromFee[所有者（）] = true；
        _isExcludedFromFee[地址（这是）]=true；
        _isExcludedFromFee [_developmentAddress ] = true；
        _isExcludedFromFee [_marketingAddress ] = true；
        发射传输（地址(0)，_msgSender（），_tTotal）；
    } //主体
}
```

清单1：表5中Rank 2模式的一个应用程序示例

在我们的检查中，我们确定的模式通常是安全的，因为大多数分包合同来自经审计的第三方软件包，如开柏林[10]和统一wap[19]。此外，大多数已标识的模式都遵循eip，这有助于增强它们的安全性、互操作性和对社区标准的遵从性。然而，如果不正确处理，对许多模式的定制要求可能会带来风险。

发现8：我们确定了61种开发模式。具体来说，令牌创建、代理契约和令牌交换占标识模式的68.75%、20.31%和10.34%，这表明这些领域的高度标准化。

4的影响和教训

.14结果汇总

与现有的工作忽略了重用外部分包合同的常见做法不同，在本研究中，我们研究了智能契约中的代码重用，并考虑了依赖关系管理。第一

我们在统计上确定了重用外部分包合同的高比率（82.94%），与之前的工作[33,35]一致。然而，在研究了他们的引入方法后，我们注意到31.22%的分包合同是通过导入而不是简单的代码克隆引入的。接下来，我们将分包合同分为自我开发的。06%)和外部的，用于进一步的代码重用分析。例如，我们观察到，当自开发的分包合同中的函数数量增加时，对函数重用的依赖就会增加。此外，超过35%的外部分包合同是接口式的，这表明了遵循既定标准和协议的趋势。然而，分包合同类型有时会被滥用，这给代码管理带来了挑战。此外，Open齐柏林航空公司吸收了约90%的进口外部分包合同，突出了其在合同开发中的突出作用。最后，我们确定了61种令牌创建（68.75%）、代理合同（20.31%）和令牌交换（10.34%）的开发模式，这些具有高度标准化的领域。

.24个经验教训

在本节中，我们将总结在智能合同开发中发现的挑战和提出的可能的解决方案。**为合同开发人员。**①尽管如第3.3节中所讨论的那样，有各种方法可以引入外部分包合同，但我们建议避免使用URL来导入外部分包合同，除非URL已经为安全开发进行了彻底的验证和审计。这是为了避免由于恶意url或损坏的分包合同而在合同中引入不可靠的代码块。②如第3.3节所述，超过80%的分包合同是在智能合同的外部引入的。然而，这种用法的文档经常会被遗漏，特别是在链上之后。这给外部分包合同的管理带来了挑战，更重要的是，给识别和减轻外部分包合同的漏洞带来了挑战。因此，我们建议包括关于每个分包合同的基本信息，如在已发布的源代码中可用的来源和版本。这种做法将有助于对下游依赖关系的代码维护和管理。.5③第3节中指出的分包合同类型的不一致使用可能会使代码组织和管理更加复杂，这表明需要在合同开发中加强标准化。
第三方审计师。①考虑到较高的代码重用率(82。在第3.3节中强调，通过建立一个标准化的基于知识的漏洞数据库，该数据库可以由每个审计员访问和贡献，特别是对于经常重用的组件。一旦数据库建立起来，就可以通过代码克隆检测来识别外部组件的相关信息，从而避免了对相同的组件进行重复审计。②鉴于其严重依赖外部分包合同，特别是第3.5节中提到的来自成熟第三方的分包合同，必须定期审计软件包，以确保其安全性和可靠性。此外，第三方软件包在合同开发中并没有作为一个整体来使用，相反，开发人员只导入所需的分包合同。因此，将已识别的漏洞链接到特定的分包合同或函数，而不是软件包是很重要的。③虽然有已发布的特殊智能合同包应用程序，但我们没有看到任何经常使用的试镜如第3.6节所述的开发模式。然而，持续监控和管理这些模式以确保安全性、可维护性和可靠性。
为社区。虽然大多数第三方包都是在NPM下管理的，详见第3.3节，但是有一个专门的智能合同开发包管理器可以更好地在开发期间和部署后管理智能合同的依赖关系。作为可靠性智能合同的专门包管理器，最好拥有：①自动化安全分析。包管理器应不断访问第三方包的安全性和使用情况。②包版本控制。包管理器应该有一个健壮的版本控制系统来跟踪智能合同包中的更新和更改。③易于集成。包管理器应该很容易与流行的开发工具和环境集成，以使智能合同开发更顺畅。④包部署。即使在部署后，软件包管理器也应该不断地监控和管理所使用的第三方分包合同。此外，考虑到在第3节中确定的open齐柏林飞艇软件包的主导地位。鼓励和支持替代和安全包的开发，为智能合同开发人员提供更多选择是有益的。

.34未来的研究方向

我们的发现为未来的研究提出了几个方向。首先，智能契约中普遍存在的代码重用需要研究跨链传播的漏洞和对策的e伍ciency。我们可以开发基于克隆的漏洞检测器，以自动补丁风险，并使用二进制代码克隆检测来改进漏洞识别。其次，将我们的分析扩展到包括3型克隆，可以为相同功能的不同实现提供更深入的见解。可以进一步研究各种实现的使用场景。此外，我们的分析可以通过包括多个区块链网络(e。g.，BNB Chain [24]和多边形[22])。这将增强我们对合同特征和跨链交互的理解，丰富我们对更广泛的智能合同生态系统的把握。最后，根据第3.6节中提供的开发模式，可以进一步研究它们对合同稳健性和风险的影响以及模式的演变。此外，研究功能级模式可以深入了解分包版本，并提高低代码开发词典性。

.44对有效性的威胁

4.4.1内部有效性。2型克隆的选择可能会导致克隆检测中的错误相似性。在这项研究中，我们选择关注2型克隆，因为它们更容易在检测的严谨性和丰富的洞察力之间取得平衡。例如，我们在第3.3节中的重复合同分析可以使用类型-
2个克隆，但与1型克隆相比无法实现。然而，我们也承认，由于变量名的重命名，识别与2型克隆的错误相似性的风险。为了缓解这种情况，我们以1型克隆检测为基线重新进行了实验，结果证明了我们的初始信息仍然成立。

4.4.2外部效度。一个潜在的威胁与数据集的代表性有关。收集所有用户的源代码

由于经常缺乏公开发布和大量现有的区块链网络，已部署的合同具有挑战性。为了缓解这种情况，我们通过从两个不同的以太坊网络（即主网和Goerli Testnet）收集合同，并识别第三方软件包来扩展我们的数据集。

4.4.3构造有效性。第3.6节中确定的模式可能会被我们的主观解释所暗示。我们通过一个严格的检查过程来缓解这种情况，包括三个智能合同审计师和交叉验证的小组讨论。然后，我们使用频率和日期范围来重新验证结果，以衡量已识别模式的流行率和寿命。通过这些验证，我们确保了这些开发模式是普遍存在的，并可以在实际应用程序中应用。在第3.3节中的重复合同分析过程中，也进行了类似的检查。利用中的信息进一步验证了该结果来自源代码和以太扫描[7]的创建者。这两种人工检查的记录都可以在我们的网站[17]上找到。

5相关工作

关于智能契约的实证研究。过去几年来，人们对智能契约中的代码重用进行了一些实证研究，重点关注了克隆的特征[33,35]和对[28,41]的影响。近藤等人。[35]报告称，79.2%的分包合同是以太坊中的1型和2型克隆，并使用了一种名为Deckard [32]的树状克隆检测器。而Khan [33]进一步将近藤的研究扩展到功能和3型克隆的粒度。结果表明，在功能水平上，总无性系比例为30.13%，其中约90%属于1型无性系。陈等人。[28]研究了智能契约中代码重用的影响，并确定了重用外部分包契约的常见修订方法。皮埃罗等人。[41]对智能合约开发中的克隆进行了分类，并从智能合约开发人员的角度对克隆的发展趋势提出了一些可能的解释。虽然之前的研究得出了智能合同中克隆率高的结论，但他们忽略了许多外部分包合同是直接导入的，但与源代码一起发布的。相比之下，我们在分析过程中考虑了介绍的方法。这使我们能够更好地理解使用外部组件的现有法规，并发现超过20%的克隆是导入的。 此外，也有一些研究从各个方面讨论了当前智能合同的发展。邹等人。[51]总结了智能合同在开发中面临的主要挑战，并讨论了提高当前智能合同开发质量的可能方向。陈等人。[27]研究了已部署的智能合同中与维护相关的问题，并讨论了如何从开发人员的角度来维护基于智能合同的项目。一些研究已经讨论了智能合同开发中的问题，如安全[26,43,45]和维护[44]。然而，这些研究只关注于一个单一的粒度(i。e.，因此，智能合约的组成就未被探索。我们的研究通过提供对智能契约组合的详细理解，并对多个粒度进行代码重用(i。e.合同、分包合同和功能)。对于每个在粒度级别上，我们设法发现了代码重用和功能需求中的一些常见实践。

智能契约开发中的发展模式。近年来，已经进行了一些与现有发展模式相关的研究。Marino和Juels [38]提出了改变和撤销智能合同的设计模式。巴托莱蒂和波米亚努[25]手动检查了来自以神扫描的811个经过验证的智能合同的源代码，通过它他们总结了智能合同的目的(e。g.财务、钱包等）以及现有的设计模式，包括令牌、授权和甲骨文。在Worley的作品[47]中也讨论了类似的设计模式。Oliva等人。[40]调查了合同的活动水平，并总结了智能合同的主要用途。沃勒等人。[46]详细阐述了几种常见的安全模式，并描述了针对典型攻击场景的解决方案。郑等人。[49]总结了智能合同开发的整个生命周期，包括合同的创建、部署、执行和完成。此外，他们还讨论了当前开发周期中的挑战，以及包括以太坊在内的智能合约平台之间的差异。Zheng [50]的另一项工作描述了区块链和共识算法的架构，并介绍了基于区块链的应用程序的开发。Mohanta等。[39]总结了智能合同的7个用例，包括物联网、供应链管理和医疗保健系统。尽管有这些贡献，但以前的研究未能提供具体的开发模式，而是侧重于各种功能需求的一般步骤。在我们的研究中，我们将开发模式作为一套有可能用于低代码开发的分包合同集来提取。这也反映了在智能合同开发中具有高度标准化的领域。

6结论

本文从以太坊区块链上的契约分解的角度讨论了稳固智能契约中的代码重用。为了理解合同的组成，我们首先量化了引入分包合同的三种常见方法，通过这些方法，我们还编制了一个在智能合同开发中常用的第三方软件包的列表。然后，我们对自开发的和外部分包合同进行了单独的代码重用分析，以分析最频繁重用的代码块及其功能属性。为了深入了解合同开发中的常见实践，我们还提取了61个经常重用的开发模式。最后，我们讨论了当前在管理智能合同方面所面临的挑战，并提出了可能的解决方案。

7数据可用性

支持本研究内容的详细分析结果可在[17]中获得。

致谢

该研究由新加坡国家研究基金会和网络安全局根据其国家网络安全研发计划（NCRP25-P04-TAICeN）提供支持。本材料中所表达的任何意见、意见、结论或建议均为作者的意见，不涉及新加坡国家研究基金会、新加坡和新加坡网络安全局的观点。

参考文献

[1] 2022.https://github.com/ethereum/solidity.
[2] 2022.https://github.com/.
[3] 2022.https://docs.ipfs.tech/concepts/ipfs-gateway/.
[4] 2022.https://gateway.ethswarm.org/.
[5] 2022.https://remix-ide.readthedocs.io/en/latest/import.html.
[6] 2022.ANTLR v4.https://github.原始日期：2010-02日04T01:36:28Z.
[7] 2022.以太坊（ETH）区块链资源管理器。http://etherscan.io/.
[8] 2022.以太坊改进方案（EIPs）。https://github.com/ethereum/原始日期：2015年10月26日T13：57：23Z。
[9] 2022.npm.https://www.npmjs.com/
[10] 2022.欧柏林飞艇合同。https://github.com/OpenZeppelin/开放齐柏林-合同原始日期：2016年08月01日T20：54：54Z。
[11] 2023.多链智能合约：多链智能合约。https://github.多链智能合约。（2023年1月23日访问）。
[12] 2023.合同一坚固性0.8.17文件。https://docs.soliditylang.org/en/v0.8.17/contracts.html.（于2023年1月29日通过）。
[13] 2023.EIPs/eip-20.md在大师·以太坊/EIPs上运行。https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.（2023年1月31日访问）。
[14] 2023.解释道：多链黑客攻击（2022年1月）。https://halborn.com/explained-the-multichain-hack-january2022/.（2023年1月23日访问）。
[15] 2023.goerli.etherscan.io.https://goerli.etherscan.io/.（2023年1月23日访问）。[16] 2023.硬帽|以太坊开发环境的专业人士由Nomic基金会https://hardhat.org/.（2023年1月27日访问）。
[17] 2023.家https://sites.google.com/view/solidity-contract-analysis/home.（于2023年2月2日通过）。
[18] 2023.家| ethereum.org。https://ethereum.org/en/.（2023年1月20日访问）。[19] 2023.家庭| Uniswap协议。https://uniswap.org/.（2023年1月27日访问）。[20] 2023.人工模型/创造者-核心-坚固性。https://github.com/manifoldxyz/creator-coresolidity.（于2023年11月6日通过）。
[21] 2023.补丁字母c/储存-工厂-fcc。https://github.com/PatrickAlphaC/存储工厂-fcc。（于2023年1月2日通过）。
[22] 2023.多边形技术。https://polygon.technology/.（访问时间06/25/2023）.
[23] 2023.智能合同-维基百科。https://en.wikipedia.org/wiki/Smart_contract .（2023年1月24日访问）。
[24] 2023.www.bnbchain.org.https://www.bnbchain.org/.（2023年6月25日访问）。[25]·马西莫·巴托莱蒂和利维奥·波姆皮亚努。2017.智能技术的实证分析合同：平台、应用程序和设计模式。计算机科学课堂讲稿（03 2017）。https://doi.org/10.1007/9783- 319702780_31
[26]莱希布伦特，内维尔格雷奇，西伊斯拉古瓦多斯，伯恩哈德肖尔茨，和扬尼斯斯马拉格-达基斯。2020.伦理器：针对复合漏洞的智能合同安全分析器。在第41届ACM编程语言设计与实现签署计划会议的记录。454一469.
陈家立[27]，新霞、罗伟、格伦迪、杨晓虎。2021.对部署后的以太坊智能合同开发的维护相关问题：问题、技术和未来的挑战。实证软件工程，26,6（2021年），1一44。
陈扁平、廖培勇、张义新、黄元、郑子斌。2021.理解智能合约中的代码重用。470一479.https://doi.org/10.1109/SANER50967.2021.00050
[29] W.G.科克伦的变体1934.在一个正态系统中，二次型的分布，并应用于协方差分析。剑桥哲学学会的数学学报30,2（1934），178一191。https://doi.org/10.1017/ S0305004100016595
[30]令牌生成器。2023.令牌生成器|创建ERC20或BEP20令牌|智能-合同工具。https://www.smartcontracts.tools/token-generator/.（于2023年3月2日通过）。
韩加味、洪成、董欣、燕西峰。2007.频繁模式采矿：当前的状态和未来的发展方向。数据挖掘和知识发现15,1（2007），55一86。
[32]凌凌、加珊、苏振东、葛龙都。2007.Deckard：可伸缩的和精确的基于树的代码克隆检测。在第29届国际软件工程会议（ICSE '07）上。IEEE，96一105。https://doi.org/10.1109/ICSE.2007.30
[33] Faizan·汗，伊斯特万·大卫，丹尼尔·瓦罗和谢恩·麦金托什。2022.编码在以太坊平台上的智能契约中的克隆：一个扩展的复制研究。IEEE软件工程学报（2022年），1一13。https://doi.org/10.1109/TSE28会议名称： IEEE软件交易.2022.32074工程
[34] Shafaq·纳希德·汗，菲扎·鲁基尔，吉迪拉-盖根，埃尔哈德·本克赫利法，和AnoudBani-哈尼。2021.区块链智能合约：应用程序、挑战和未来趋势。点对点网络和应用程序14,5（2021），2901一2925。
近藤[35]，古斯塔沃。奥利瓦，振明（杰克），艾哈迈德。哈桑和Mizuno。2020.智能合约中的代码克隆：一个关于来自以太坊区块链平台的已验证契约的案例研究。经验软件工程25,6。2020），4617一4675.https://doi.org/10.1007/s10664-020098525
[36]保罗利维。2014.有限的人口修正。https://doi.org/10.1002/9781118445112.stat05700
刘超超、韩汉、赵曹、钟陈、陈青岛、比尔·罗斯科。2018.保留：在智能合同中找到可入的漏洞。在第40届软件工程国际会议的会议记录：同伴程序（哥德堡，瑞典）（ICSE '18）。计算机机械协会，纽约，纽约，美国，65一68。https://doi.org/10.1145/3183440.3183495
[38] Bill·马里诺和Ari Juels。2016.设置更改和撤销智能功能的标准合同，卷。9718.151一166.https://doi.org/10.1007/9783- 319-420196_10
[39]熊猫，熊猫，和堕落的耶拿。2018.区块链技术中的智能契约和用例概述。2018年第九届计算、通信和网络技术国际会议（ICCCNT）。1一4.https://doi.org/10.1109/ICCCNT.2018.8494045
[40]古斯塔沃。奥利瓦，艾哈迈德E。哈桑和姜明（杰克）。2020.一个前ploratorystudyofsmartcontractsintheEthereumblockchainplatform.实证软件工程25,3（2020年5月），1864一1904。https://doi.org/10.1007/s10664019-097965公司：施普林格经销商：施普林格机构：施普林格标签：施普林格编号：3出版社：施普林格美国。
[41]朱塞佩·安东尼奥·皮埃罗和罗伯托·托内利。2021.源代码分析在企业智能合同中的复制。701一707.https://doi.org/10.1109/SANER50967.2021.00089
[42]挑战了罗伊和詹姆斯·科迪。2007.软件克隆检测技术综述研究计算机学院TR2007-541(1月1日。2007).
[43]，辛志民，张齐，和阿里德汉达尼亚。2020.区块链智能合同的形式化：解决漏洞的方法和挑战。压缩。安全。88（2020）.
[44]，安娜·维卡，安德里亚·迪·索尔博，科拉多·维萨吉奥，和杰拉尔多·坎福拉。2021.关于区块链和智能合同开发的系统文献综述：技术、工具和公开的挑战。系统与软件杂志174（2021），110891。
[45]万志远、新霞、罗大卫、陈家立、罗霞浦、杨晓虎。2021.智能合同安全：一个从业者的观点。2021年，IEEE/ACM第43届软件工程国际会议（ICSE）。IEEE，1410一1422。
[46]马克西米利安·沃rer和Uwe Zdun。2018.智能合约：安全模式以太坊生态系统和固度。2018年区块链面向软件工程国际研讨会（IWBOSE）。IEEE，2一8。
[47]卡尔R。沃利和安东尼·斯凯勒姆。2018.机会、挑战和智能合同设计模式的未来扩展。在商业信息系统中。
[48]吴家慧、徐正子、魏唐、张玉、吴月明、刘承月、孙冠然、赵立达、刘杨等。2023.OSSFP：精确、可扩展的C/C++第三方库检测。2023年，IEEE/ACM第45届软件工程国际会议（ICSE）。270一282.https://doi.org/10.1109/ICSE48619.2023.00034
郑子斌、谢少安、戴洪宁、陈伟力、陈湘平、吉安翁和穆罕默德·伊姆兰。2020.关于智能合约的概述：挑战、进步和平台。未来一代计算机系统105（2020），475一491。
郑子斌、谢少安、戴洪宁、陈湘平、王怀民。2018.区块链面临的挑战和机遇：一项调查。《国际网络和网格服务杂志》14（10 2018），352。https://doi.org/10.1504/IJWGS 095647.2018.
邹[51]，罗大卫，乐宣巴赫，新霞，杨峰、陈振宇、徐鲍文。2019.聪明的合同开发：挑战和机遇。IEEE软件工程学报47,10（2019），2084一2106。

2023年02月02；2023年07月27日