Helix++：高效保障软件安全的平台

Jack W. Davidson[1,✉]、**Jason D. Hiser**[1]和 **Anh Nguyen-Tuong**[1]

[1]弗吉尼亚大学计算机科学系

arXiv:2304.04846v1 [cs.CR] 10 Apr 2023

开源项目 Helix++ 通过自动应用尖端网络安全技术，使软件实现多样化并加固，从而提升计算平台的安全态势。Helix++ 的一项显著特点是无需源代码或构建产物；它直接处理二进制形式的软件——即使是经过剥离的可执行文件和库也不例外。这一特点至关重要，因为从源代码重新构建应用程序既耗时，又常常令人沮丧。多样化打破软件单一化，使攻击更难实施，因为成功攻击所需的信息会以不可预测的方式发生变化。多样化还迫使攻击者针对每个目标定制攻击，而不是编写能在所有配置相似的目标上可靠奏效的漏洞利用程序。加固则直接针对关键攻击类别。多样化与加固相结合，既能提供纵深防御，也能形成移动目标防御，从而保障国家网络基础设施的安全。

网络安全 | 软件单一化 | 软件多样性 | 容器安全 对应：*jwd@virginia.edu*

简介

现代软件中的网络安全仍然是一个必须解决的关键问题，这关系到我们个人医疗数据和设备、关键基础设施以及银行系统的安全。包括代码审查技术、安全扫描器和模糊测试器在内的诸多进展，都有助于提高已部署软件的质量。（1–10）然而，随着新的软件缺陷不可避免地出现，这些技术仍然要求对软件进行修补并重新部署。这个过程可能需要数周、数月甚至数年！诸如兼容性问题之类的问题会使软件的重新部署变得复杂，因为这些问题可能阻止或延迟更新。有时，安全补丁会被“回移植”到软件的早期版本中，以提供与现有系统更加兼容的安全增强型软件。此外，即使安全更新已经准备好通过更新管理器进行部署，用户也可能因为这一过程需要停机（可能需要重启机器来安装更新）而将其延迟数月。你的Windows/Linux/macOS机器现在是否正在要求重启以安装更新？端点检测与防护技术也存在缺点。（11，12）它们通常只能检测已知威胁，而一旦攻击者知道如何规避该技术，这些技术就几乎没有用处。此外，它们极易产生误报。（13，14）

理想情况下，安全防护可以持续应用于已部署的软件。这种方法通过直接为已部署的软件添加安全功能，避免了重新部署的成本。

此外，由于软件安全防护应用于经系统管理员审核的软件，因此可以避免端点检测过程中出现误报或漏报。最后，还可以部署一组随机化软件变体，构建移动目标防御，从而挫败常见的“脚本小子”式攻击。这类网络攻击通过利用软件单一化实现自动化，并能轻易将已部署软件的详细信息嵌入攻击脚本中。

Docker 容器为实现这种安全性提供了一个很有意思的机会。软件系统的设计方式是基于现有的软件层构建，并可反复重建；当用户需要访问软件时，还能高效、快速地重新启动。

Helix++ 是我们提出的方案，旨在为 Docker 容器添加改造后的安全机制。Helix++将现代、稳健、透明且高效的二进制重写集成到软件的 Docker 构建流程中，从而创建可供最终用户透明使用的 Docker 镜像。我们扩展了镜像部署模型，使其能够从一组等效镜像中随机选择，以避免向每位用户部署相同的软件，打破软件单一化，并增加攻击者的工作量。

本文介绍了 Helix++ 的愿景，以及我们迄今为止在实现这些潜在优势、支持大规模分发方面取得的进展。我们针对弗吉尼亚大学 ACCORD 系统开展的初步案例研究表明，只需付出适度的分析时间和存储空间成本，就能保护关键基础设施。

Helix++

图 1 展示了 Helix 的整体++架构。其核心思路是：以一个包含可供部署软件（例如 Docker 注册表）的存储库为起点，对其中的软件进行安全加固和多样化处理，并构建一个加固注册表，其中包含经过加固和多样化处理的功能等效软件。

加固注册表中包含多个相同的软件副本，每个副本都使用不同的随机种子进行伪随机多样化变换。借助这一特性，每个用户的软件请求都能获得一个多样化的软件变体。理论上，每个请求都可以获得一份从未部署过的软件。不过在实际中，需求高峰期间，注册表可能不够大，或者变换器速度不够快，无法为每个请求都提供独一无二的软件。

我们的研 研究目标是了解请求频率 究

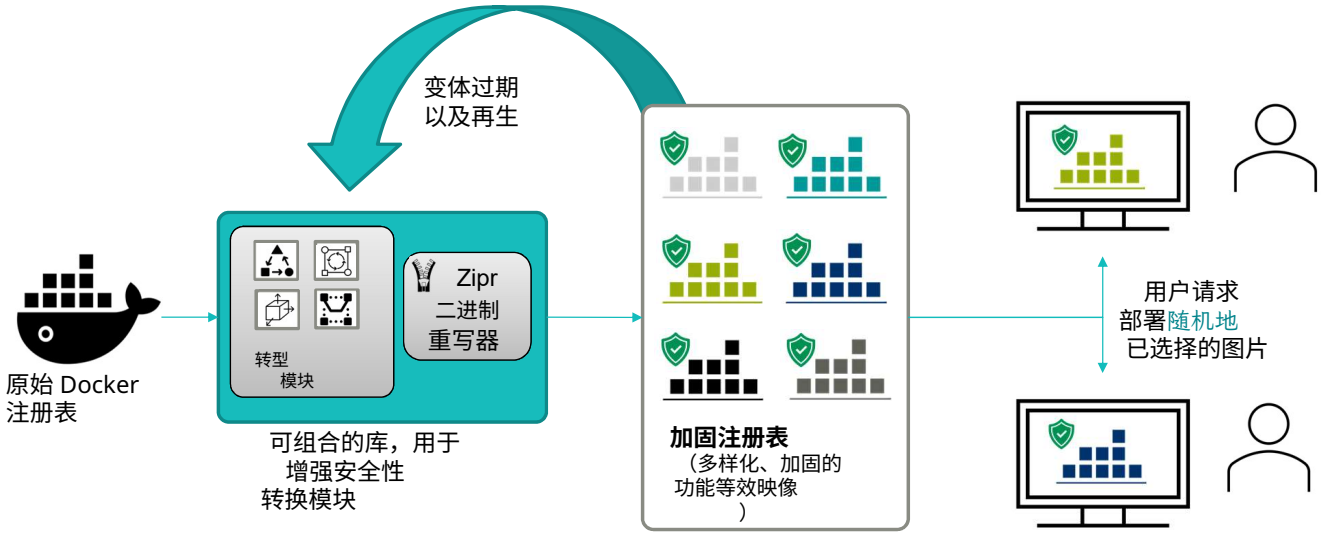


图 1。 Helix++ 架构。

分布、变体生成的计算成本，以及该系统的成本与收益。

二进制重写。 Helix++ 基于二进制重写。二进制重写允许在无法访问源代码的情况下修改可执行文件的行为。对于需要分析和修改二进制文件行为的安全研究人员来说，这项功能特别有用。通过二进制重写，用户可以替换或插入指令、更改函数参数并操作数据结构，从而修改可执行文件的代码。这项功能允许用户在软件应用程序中添加安全检查或修补漏洞。

许多二进制重写器都提供用于构建插件的 API。插件通常可以转换程序，并且可以组合多个插件来整合功能。例如，一个插件可以转换栈布局，另一个插件则可以重新排列全局变量的布局。两者结合后，栈和全局变量的位置都可以随机化。

Helix++ 利用了 Zipr 静态二进制重写器，更多介绍请参阅 Helix++开发现状章节。

加固注册表。一个汇集了各种安全加固程序的注册表，可以为用户和组织带来多项好处。首先，它提供了一个集中位置，方便用户获取各种安全加固程序，从而减少查找、下载和安装安全软件所需的时间和精力。对于需要确保系统和数据安全的组织而言，这种方式尤其有益，因为它们可以轻松地在网络中获取并部署安全软件。

拥有一个收录安全加固程序的注册表，另一个好处是有助于推广安全最佳实践和标准。通过让用户能够访问安全软件，它鼓励用户采用注重安全的实践，例如及时更新软件。此外，建立一个收录安全加固程序的注册表，还有助于促进安全专业人员之间的协作与知识共享，因为他们可以将自己的安全转换贡献给 Zipr，从而提升注册表的安全性。这一好处有助于提高整个软件生态系统的整体安全性，让所有依赖这些程序来保护系统和数据安全的用户都能从中受益。

注册表的一项关键功能是变体可以过期。如果计算资源充足，甚至可能在某个变体仅部署一次后就使其过期。

Helix++开发现状

Helix++正在由多个稳定的工业软件组件和 Zipr 静态二进制重写器构建而成。

使用 Zipr 进行二进制重写。Helix++ 的二进制重写基于 Zipr。（15–20）Zipr 的核心功能支持块级指令随机化（BILR），与 Zhan 等人的方法类似。（21）Zipr 通过执行深度二进制分析并构建中间表示来实现此功能。Zipr 的中间表示包含程序中的每条指令、程序中的静态数据对象（全局变量、开关表、ELF 链接表等）、异常处理信息以及程序的元数据（例如目标架构的位宽）。Zipr 的中间表示类似于低级编译器的后端中间表示。构建中间表示后，Zipr 可以调用基于 Zipr API 构建的 Zipr 插件。该 API 允许轻松组合插件，但当然，插件必须足够健壮，才能协同工作。例如，如果一个插件将间接分支转换为由 `if/then/else` 结构实现的直接分支，而后续插件使用控制流完整性技术对间接分支指令进行插桩，那么后续插件将找不到任何可供插桩的间接分支，因此这两个插件无法良好协同。不过，如果一个插件对栈操作进行插桩，另一个插件对全局数据操作进行插桩，那么两者无需特别处理即可协同工作。

Zipr 插件 API 支持对中间表示进行常见的底层修改——插入、修改或删除中间表示的任意部分。此外，该 API 允许用户选择性地在修改后的中间表示上重新运行用于构建初始中间表示的深度静态分析技术。这项功能在以下情况下很有用，

例如，插桩使用一个空闲寄存器进行性能优化，而后续优化需要知道哪些寄存器仍然空闲。

这些特性使 Zipr 成为数百万美元资金投入所产出的最稳定、高效且有效的静态二进制重写器之一。该项目源代码的首次提交可追溯至 2010 年。Zipr 可以转换经过剥离或未剥离的二进制程序，并生成在功能上等效的二进制程序。它对 x86-64 Linux ELF 二进制文件的支持最为稳健，同时也支持 ARM ELF 二进制文件、MIPS ELF 二进制文件以及 Windows PE/PE+ 文件。所有文件格式均支持 32 位和 64 位架构。

Zipr 已被证明是少数几种稳健的二进制重写器之一（22）。Zipr 的提交会定期使用 42 个真实世界的软件应用进行测试，这些应用由多种编译器（gcc、clang、ollvm、icx）编译，采用各种编译器标志（O0、O1、O2、O3、Ofast、OSize），分别以 PIE 和非 PIE 模式构建，并包含已剥离和未剥离两种形式；测试覆盖 Ubuntu 的 3 个不同版本（16.04 LTS、18.04 LTS 和 22.04 LTS）。所选程序由 C、C++ 和 Rust 应用组成。

Zipr 已用于多个项目，包括 DARPA 网络安全大挑战（CGC）。CGC 是一项自主夺旗竞赛，由网络推理系统参与角逐。Zipr 是 TechX 团队提交方案 Xandra 的一部分。Zipr 帮助其取得了所有参赛者中的最佳安全评分，并最终获得总排名第二。（23，24）

Zipr 也被用于 DARPA 的网络容错攻击恢复（CFAR）计划中（25）。该计划旨在生成多个不同版本的 Web 服务器，并行运行这些程序。如果各个版本的行为出现分歧，就可以假定正在发生网络攻击，并采取补救措施。Zipr 用于生成代码、栈、全局数据和堆布局各不相同的版本，从而可证明地防止针对若干常见内存错误类别的重要漏洞利用程序。由于 Zipr 的技术不依赖源语言，因此它是该计划中唯一能够处理 ADA Web Server 应用程序的解决方案（该应用程序显然是用 ADA 编写的）。

除了这些项目之外，Zipr 还被用于开展反脆弱性研究，并成为借助 Zafl 等工具进行高效纯二进制模糊测试，以及名为 HeXcite 的 Untracer 纯二进制版本的基础。(26–30)

弗吉尼亚大学 ACCORD 中的强化注册表。 作为在真实环境中使用强化注册表的实际示例，我们与弗吉尼亚大学（UVA）的 ACCORD 团队展开了合作。ACCORD 是：

> 一个基于 Web 的平台，允许弗吉尼亚州各地公立大学的研究人员在一个中央位置分析和存储敏感数据。
>
> ACCORD 适用于去标识化的个人身份信息、FERPA、去标识化的 HIPAA、商业机密以及其他类型的去标识化敏感数据。
>
> 得益于美国国家科学基金会（奖项编号：1919667）提供的资金，AC- CORD 可供弗吉尼亚州的研究人员免费使用。（31）

要向 ACCORD 安全存储区复制数据或从中复制数据，需要使用 Globus。(32) 要检查或计算这些数据，需要启动一个“会话”。用户通过 Web 前端启动这些会话。当前的会话包括 C/C++IDE、Python 解释器或 RStudio。(33,34) 每个会话都部署在 Kubernetes pod 中。（35）。每个 pod 包含两个 Docker 容器，其中一个供用户直接交互，另一个是 sidecar 容器，用于中介处理与用户控制容器之间的所有网络流量，以确保安全。当前的安全策略允许用户使用 `pip` 从一组“可信”仓库安装 Python 软件包。虽然这些仓库通常被认为是安全的，但人们无法确定 Python 仓库中的内容通常具有怎样的来源或安全性。因此，就安全性而言，Python 是管理员重点担忧的一个向量。

鉴于这一担忧，我们与 ACCORD 团队合作，为 Python 解释器应用了基本的 Zipr 保护措施。到目前为止，我们仅将工作限制在这一关键的软件组件上。我们创建了多个版本的 ACCORD Docker镜像。创建这些镜像的方式是扩展Github 工作流，添加新的作业，以修改之前步骤中创建的镜像。借助 Github 提供的大规模并行处理能力，这些工作流只增加了几分钟的处理时间。

我们进一步对ACCORD的 Web 界面进行了细微修改，使其在启动新会话时从功能等价的容器集合中随机选择一个。尽管据我们所知，这些机制已完全具备功能，但我们正在与ACCORD 团队合作，逐步将这些变更部署给 ACCORD 系统的最终用户。目前，我们需要与 ACCORD 管理员讨论这一流程的具体细节，但他们对为其系统增加安全性的总体计划感到兴奋。如果成功，加固容器将成为Helix++ 项目的最终目标。我们计划衡量诸如注册表增加的存储空间、转换映像所需的时间，以及我们的基础设施能够管理的替换率等特征。

结论

Helix++ 是一种系统，可使用二进制重写器为已部署的软件添加安全加固和多样化转换。系统会创建一个装有各种加固版本的 Docker 注册表。这些版本用于满足用户对软件的需求。过期策略可确保这些版本始终保持最新，从而避免软件单一化。此外，当软件有了新的加固措施时，可以使用二进制重写器自动重新应用这些措施。我们利用 Zipr 二进制重写器及其转换工具套件，在 UVA ACCORD 系统的基础上构建了一个系统，可随机选择版本供最终用户使用。虽然项目仍在进行中，但我们已取得了显著进展，

ACCORD 团队对有机会进一步保护其系统感到十分振奋。

致谢
本材料基于美国国家科学基金会第 2115130 号资助项目的研究成果。材料中表达的任何观点、发现、结论或建议均属于作者的观点，不一定反映美国国家科学基金会的看法。

参考文献

1. Jun Li、Bodong Zhao 和Chao Zhang。《模糊测试：综述》。网络安全，1(1)：1–13，2018年。 2.Marcel Böhme、Van-Thuan Pham、Manh-DungNguyen 和Abhik Roychoudhury。《定向灰盒模糊测试》。载于2017年ACMSIGSAC计算机与通信安全会议论文集，第2329–2344页，2017年。3. HongliangLiang、Xiaoxiao Pei、Xiaodong Jia、Wuwei Shen和JianZhang。《模糊测试：最新进展》。IEEE可靠性汇刊，67(3)：1199–1218，2018年。 4. Xiaogang Zhu 、 Sheng Wen 、 SeyitCamtepe 和 Yang Xiang。《模糊测试：路线图综述》。ACM计算调查（CSUR），54(11s)：1–36，2022年。5. Shane McIntosh、Yasutaka Kamei、BramAdams 和Ahmed EHassan。《代码审查覆盖率和代码审查参与度对软件质量的影响：对Qt、VTK和ITK项目的案例研究》。载于第11届软件仓库挖掘工作会议论文集，第192–201页，2014年。6. Shane McIntosh、Yasutaka Kamei、Bram Adams 和Ahmed EHassan。《现代代码审查实践对软件质量影响的实证研究》。实证软件工程，21：2146–2189，2016年。 7. Eduardo Witter dosSantos 和 IngridNunes。《调查同行代码审查在分布式软件开发中的有效性》。载于第XXXI届巴西软件工程研讨会论文集，第84–93页，2017年。 8. Richard Amankwah、Jinfu Chen、Patrick KwakuKudjo 和 Dave Towey。《商业Web漏洞扫描器与开源Web漏洞扫描器的实证比较》。软件：实践与经验，50(9)：1842–1857，2020年。 9.BalumeMburano 和 Weisheng Si。《基于OWASP基准测试评估Web漏洞扫描器》。载于2018年第26届国际系统工程会议（ICSEng），第1–6页。IEEE，2018年。 10. Gabriela Roldán-Molina、Mario Almache-Cueva、Carlos Silva-Rabadão、Iryna Yevseyeva 和 Vitor Basto-Fernandes。《网络安全风险分析工具的比较》。计算机科学 Procedia，121：568–575，2017年。 11. Asad Arfeen、Saad Ahmed、Muhammad Asim Khan和SyedFaraz Ali Jafri。《端点检测与响应：一种恶意软件识别解决方案》。载于2021年国际网络战与安全会议（ICCWS），第1–8页。IEEE，2021年。 12. Seung Jae Yoo。《关于改进端点安全技术的研究》。融合安全期刊，18(3)：19–25，2018年。 13. Cheng-Yuan Ho、Yuan-Cheng Lai、I-Wei Chen、Fu-Yu Wang和 Wei-Hsuan Tai。《使用入侵检测/防御系统对真实流量中的误报和漏报进行统计分析》。IEEE通信杂志，50(3)：146–154，2012年。 14. Liang Hu、TaihuiLi、Nannan Xie 和Jiejun Hu。《基于聚类的入侵检测误报消除》。载于2015年第12届国际模糊系统与知识发现会议（FSKD），第519–523页。IEEE，2015年。 15.William H Hawkins、Jason D Hiser、Michele Co、Anh Nguyen-Tuong和Jack WDavidson。《Zipr：用于安全的高效静态二进制重写》。载于2017年第47届IEEE/IFIP国际可靠系统与网络会议（DSN），第559–566页。IEEE，2017年。 16. William Hawkins、Jason D Hiser、Anh Nguyen-Tuong、JackW Davidson等。《保护二进制代码》。IEEE安全与隐私，15(06)：77–81，2017年。 17. Jason Hiser、Anh Nguyen-Tuong、William Hawkins、Matthew McGill、MicheleCo和 Jack Davidson。《Zipr++ 异常二进制重写》。载于2017年构建软件转换生态系统研讨会论文集，第9–15页，2017年。 18. Zephyr代码库；zipr源代码。https://git.zephyr-software.com/ opensrc/zipr，。访问日期：2023年3月21日。 19. Zephyr代码库；irdb食谱示例。https://git.zephyr-software.com/opensrc/irdb-cookbook-examples，。访问日期：2023年3月21日。 20. Zephyr代码库；irdb API。https://git.zephyr-software.com/opensrc/ irdb-sdk，。访问日期：2023年3月21日。 21. Xun Zhan、TaoZheng 和 Shixiang Gao。《使用基本块级随机化防御ROP攻击》。载于2014年IEEE第八届国际软件安全与可靠性会议（伴会），第107–112页。IEEE，2014年。 22. Eric Schulte、Vlad Folts 和MichaelBrown。《二进制提升器评估》。arXiv预印本 arXiv:2203.13231，2022年。 23. DARPA网站；网络安全大挑战。https://www.darpa.mil/program/cyber-grand-challenge。访问日期：2023年3月21日。 24. Anh Nguyen-Tuong、David Melski、Jack W Davidson、Michele Co、William Hawkins、Jason D Hiser、Derek Morris、Ducson Nguyen 和 Eric Rizzi。《Xandra：面向网络安全大挑战的自主网络作战系统》。IEEE安全与隐私，16(2)：42–51，2018年。 25. DARPA网站；网络容错攻击恢复。https://www.darpa.mil/ program/cyber-fault-tolerant-attack-recovery。访问日期：2023年3月21日。 26. Kevin Leach、Christopher S Timperley、Kevin Angstadt、Anh Nguyen-Tuong、Jason Hiser、Aaron Paulos、Partha Pal、Patrick Hurley、CarlThomas、Jack W Davidson 等。《Start：面向可信且具韧性的自动驾驶车辆的框架（实践经验报告）》。载于2022年IEEE第33届国际软件可靠性工程研讨会（ISSRE），第73–84页。IEEE，2022年。 27. Zephyr代码库；zafl源代码。https://git.zephyr-software.com/ opensrc/zafl。访问日期：2023年3月21日。

28. Stefan Nagy、Anh Nguyen-Tuong、Jason D Hiser、Jack W Davidson 和 Matthew Hicks。相同覆盖率，更少膨胀：通过保留覆盖率的覆盖引导跟踪加速仅针对二进制文件的模糊测试。载于 *2021* 年 *ACM SIGSAC* 计算机与通信安全会议论文集，第 351–365 页，2021 年。 29. Stefan Nagy、Anh Nguyen-Tuong、Jason D Hiser、Jack W Davidson 和 Matthew Hicks。突破二进制文件的限制：通过编译器级插桩改进仅针对二进制文件的模糊测试。载于第 *30* 届 *USENIX* 安全研讨会，2021 年。30. Stefan Nagy 和 Matthew Hicks。全速模糊测试：通过覆盖引导跟踪降低模糊测试开销。载于 *2019* 年 *IEEE* 安全与隐私研讨会（*SP*），第 787–802 页。IEEE，2019 年。31. 弗吉尼亚大学研究计算中心。https://www.rc.virginia.edu/userinfo/accord/overview/。访问日期：2023年3月21日。32. Globus 网站。https://www.globus.org/。访问日期：2023年3月21日。 33. Theia 网站。https://theia-ide.org/。访问日期：2023年3月21日。34. Posit 网站。https://posit.co/download/rstudio-desktop/。访问日期：2023年3月21日。 35. Kubernetes 网站。https://kubernetes.io/。访问日期：2023年3月21日。