CI 安全初步 - 从某开始

PLCT Lab 每周技术分享

熊家辉
浙江工商大学
2024 年 6 月 5 日

目录

CI 安全初步
2024-06-05
└目录

第 1 节

CI 的安全强化

第 1 节
CI 的安全强化

第 1 节

CI 的安全强化

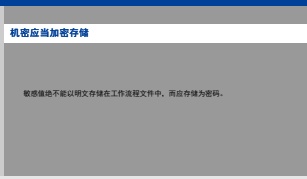
第 1.1 小节

使用机密

第 1 节
CI 的安全强化
第 1.1 小节
使用机密

机密应当加密存储

敏感值绝不能以明文存储在工作流程文件中，而应存储为密码。



敏感值绝不能以明文存储在工作流程文件中，而应存储为密码。机密可在组织、存储库或环境级进行配置，使你可在 GitHub 中存储敏感信息。

机密使用 Libsodium 密封箱，以使其在到达 GitHub 前被加密。当使用 UI 或通过 REST API 提交机密时会发生这种情况。此客户端加密有助于最大程度地减少与 GitHub基础架构中的意外日志记录相关的风险（例如，异常日志和请求日志等）。密钥在上传后，GitHub 可对其进行解密，以便它能够被注入工作流程运行时。

为了帮助防止意外泄露，GitHub 使用一种机制尝试对运行日志中显示的任何密码进行编校。此编辑会寻找作业使用的任何已配置机密的精确匹配项，以及值的常见编码，如 Base64。但是，由于密码值可以通过多种方式转换，因此不能保证此编校。此外，运行器只能对当前作业使用的机密进行编辑。因此，你应该采取某些积极主动的步骤和良好的做法，以帮助确保密码得到编校，并限制与密码相关的其他风险：

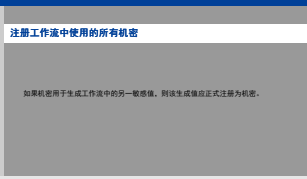
切勿将结构化数据用作机密

不要使用 JSON、XML 或 YAML（或类似）的 Blob 来封装密码值.

结构化数据可能导致日志中的密码编校失败，因为编校很大程度上取决于查找特定密码值的完全匹配项。例如，不要使用 JSON、XML 或 YAML（或类似）的 Blob 来封装密码值，否则会显著降低密码被正确编校的可能性。而应为每个敏感值创建单独的密码。

注册工作流中使用的所有机密

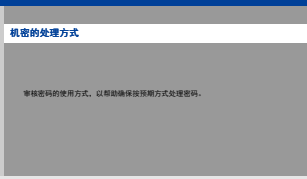
如果机密用于生成工作流中的另一敏感值，则该生成值应正式注册为机密。



如果机密用于生成工作流中的另一敏感值，则该生成值应正式注册为机密，以便在它出现在日志中时对其进行编辑。例如，如果使用私钥生成签名的 JWT 来访问 Web API，请确保将该 JWT 注册为密码，否则，如果它进入日志输出，则不会得到编校。注册密码也适用于任何类型的转换编码。如果以某种方式（如 Base64 或 URL 编码）转换您的密码，请确保将新值也注册为密码。

机密的处理方式

审核密码的使用方式，以帮助确保按预期方式处理密码。



审核密码的使用方式，以帮助确保按预期方式处理密码。您可以通过检查执行工作流程的仓库的源代码并检查工作流程中使用的任何操作来进行审核。例如，确认它们未发送到非预期主机，或明确打印到日志输出。在测试有效/无效输入后查看工作流程的运行日志，并确认密码已正确编校或未显示。调用的命令或工具向 STDOUT 和 STDERR 发送错误的方式并不总是很明显，并且机密随后可能会出现在错误日志中。因此，在测试有效和无效的输入后，最好是手动查看工作流程日志。有关如何清理可能无意中包含敏感数据的工作流日志的信息，请参阅“使用工作流运行日志”。

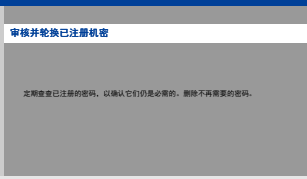
使用最小范围的凭据

确保工作流程中使用的凭据具有所需的最小权限。

确保工作流程中使用的凭据具有所需的最小权限，并请注意，任何对仓库具有写入权限的用户都可访问仓库中配置的所有密码。 Actions 可以从 githubtoken 上下文访问 GITHUBTOKEN 来使用它。有关详细信息，请参阅“上下文”。因此，应确保向 GITHUBTOKEN 授予所需的最低权限。将 GITHUBTOKEN 的默认权限设置为只读取存储库内容是良好的安全做法。然后可以根据需要增加工作流程文件中个别任务的权限。有关详细信息，请参阅“自动令牌身份验证”。

审核并轮换已注册机密

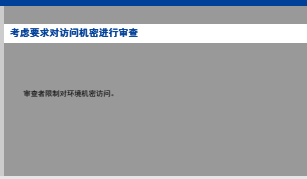
定期查查已注册的密码，以确认它们仍是必需的。删除不再需要的密码。



定期查查已注册的密码，以确认它们仍是必需的。删除不再需要的密码。定期轮换密码，以减小泄露的密码有效的时间窗。

考虑要求对访问机密进行审查

审查者限制对环境机密访问。



您可以使用所需的审查者来保护环境机密。在审查者批准之前，工作流程作业无法访问环境机密。有关在环境中存储机密或需要审查环境的详细信息，请参阅“在 GitHub Actions 中使用机密”和“使用环境进行部署”。对存储库具有写入访问权限的任何用户都有权读取存储库中配置的所有机密。因此，应确保工作流中使用的凭据具有所需的最低权限。

第 1 节

CI 的安全强化

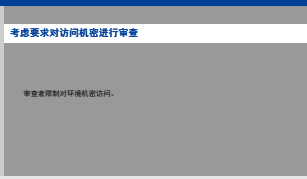
第 1.2 小节

了解脚本注入的风险

第 1 节
CI 的安全强化
第 1.2 小节
了解脚本注入的风险

考虑要求对访问机密进行审查

审查者限制对环境机密访问。



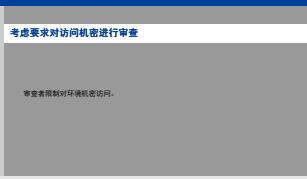
创建工作流、自定义操作和复合操作操作时，应始终考虑代码是否可能执行来自攻击者的不受信任的输入。当攻击者将恶意命令和脚本添加到上下文时可能发生这种情况。当您的工作流程运行时，这些字符串可能会被解释为代码，然后在运行器上执行。

攻击者可以将其自己的恶意内容添加到 github 上下文中，这应被视为潜在的不受信任的输入。这些上下文通常以 body、defaultbranch、email、headref、label、message、name、pagename、ref 和 title 结尾。例如：github.event.issue.title 或 github.event.pullrequest.body。

您应该确保这些值不会直接流入工作流程、操作、API 调用，或任何可能被解释为可执行代码的其它地方。通过采用您将用于任何其他特权应用程序代码的相同防御编程姿态，，您可以帮助安全保护 GitHub Actions 的使用。若要了解攻击者可能采取的某些步骤，请参阅“GitHub Actions 的安全强化”。

考虑要求对访问机密进行审查

审查者限制对环境机密访问。



此外，还有其他不太明显的潜在不信任输入来源，如分支名称和电子邮件地址，这些输入在允许的内容方面可能相当灵活。例如将是一个有效的分支名称，并将成为目标存储库可能的攻击途径。
以下部分解释了如何帮助降低脚本注入的风险。
此示例易受脚本注入的影响，因为 run 命令在运行器的临时 shell 脚本中执行。在 shell 脚本运行之前，内的表达式被评估后替换为结果值，这使它易受 shell 命令注入的攻击。
使用操作而不是内联脚本、使用中间环境变量

谢 谢

谢 谢