Code Repository Vulnerability Focusing on RepoJacking

Minjae Kim
Data Intelligence Lab,
Information Security Institute
ESTsecurity
Seoul, Republic of Korea
mjkim45@estsoft.com

Shinho Lee
Data Intelligence Lab,
Information Security Institute
ESTsecurity
Seoul, Republic of Korea
lee1029ng@estsecurity.com

Taehyeong Kwon
Data Intelligence Lab,
Information Security Laboratory
ESTsecurity
Seoul, Republic of Korea
brightkwon@estsecurity.com

Wookhyun Jung
Data Intelligence Lab,
Information Security Institute
ESTsecurity
Seoul, Republic of Korea
pplan5872@estsecurity.com

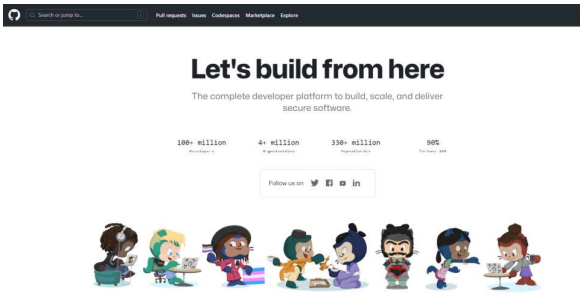
Eui Tak Kim
Business Intelligence Center
ESTsecurity
Seoul, Republic of Korea
etkim@estsecurity.com

***Abstract*—A supply chain attack is a type of cyberattack executed by infiltrating a target system during software distribution, allowing malicious code disguised as something benign to be distributed without raising suspicion. Any supply chain attack that targets open sources may lead to critical consequences because open sources are made available for use by anyone. Meanwhile, GitHub, a widely used open-source management tool, supports the function of code repositories. For that reason, GitHub has been widely used by individuals and renowned businesses for open-source management and distribution. RepoJacking is a type of supply chain attack wherein GitHub's repositories are hijacked and exploited by malicious actors. This study aims to assess the practical feasibility of RepoJacking and verify the vulnerability of a set of repositories publicly released on GitHub against RepoJacking. In addition, methods for protecting against RepoJacking, along with a tool designed to assess the integrity of repositories against RepoJacking, are proposed.**

Keywords—supply chain, repository, hijacking, vulnerability

I. INTRODUCTION

Efficient code development requires reliable code repositories. GitHub [1], a well-known code repository, is utilized for managing or releasing code developed by individuals or businesses, similar to other code repositories. More specifically, GitHub allows individuals and businesses to distribute code or packaged code for different projects either in a public or private manner. According to GitHub [2], as of 2023, more than 330 million code repositories are registered on the platform, as shown in Figure 1.



[Figure 1] Statistics on GitHub: account/organization/repository

Among the numerous code repositories available on GitHub, many are recognized for their popularity and have been made accessible to the general public as open-source projects. Developers often refer to open-source projects to find the necessary code, aiming to facilitate the development process and reduce the required time. Consequently, it is possible for a certain project to remain intricately linked to the development outcomes of numerous other projects. If this is the case, any repository hijacking attack conducted against open-source projects may pose a serious threat in many ways, for example, by infecting them with malicious code.

Repository hijacking, also known as RepoJacking, is a cyberattack in which malicious actors aim to steal others' information while masquerading as legitimate owners themselves [3]. As such, through RepoJacking, attackers steal others' repositories and pretend as if they are the legitimate owners of them. This allows them to gain the trust of users attempting to access the repositories, ultimately facilitating the distribution of the specific malicious code they intend to spread. As a supply chain attack, RepoJacking empowers attackers to conduct a range of malicious activities without the need to infiltrate multiple individual computing systems. Therefore, RepoJacking may pose a substantial threat not only to the open-source ecosystem but also to the overall supply chain system.

The present study aims to verify how vulnerable these repositories are to RepoJacking from a practical perspective and describe measures to address the identified issues. The objectives of this paper are as follows.

- To verify the vulnerabilities of a proof of concept (PoC) designed for RepoJacking and actual GitHub repositories, which may lead to supply chain attacks
- To propose countermeasures against RepoJacking and effective tools to assess the risks

Chapter 2 provides a description of a tool designed to assess the integrity of repositories against RepoJacking, along with previous studies on RepoJacking. Chapter 3 identifies and verifies the vulnerabilities of repository systems against RepoJacking. Finally, in Chapter 4, countermeasures to address vulnerability concerns associated with RepoJacking,

as well as tools that can be used to evaluate the risks, are proposed.

II. Background

This chapter is composed as follows. Section A provides a brief description of Git and GitHub. Section B then introduces GHTorrent, an archival system designed for tracking and storing GitHub activities. Subsequentially, Section C provides an overview of RepoJacking attacks. In Section D, previous studies on RepoJacking attacks and relative issues are reviewed.

A. Git & GitHub

Git[4] is a snapshot stream-based distributed version control system (DVCS) used to track changes to files while facilitating coordination of work on these files by multiple users. This system is mainly used for source code management and currently available for use as open-source software. GitHub is a web service that provides support for hosting Git repositories. This service is compatible with the graphic user interface (GUI) and thus more user-friendly compared to Git. Additionally, as it is built on Git, GitHub supports source code, as well as a variety of other formats, such as graphs and markdowns. This enables users not only to manage source codes but also to access various services, such as Wiki. Thanks to these benefits, many users choose to use GitHub for storing and distributing code.

B. GHTorrent

GHTorrent [5] is a dataset that collects a series of events occurring on GitHub using an API provided by GitHub. This system collects GitHub commits, along with repository information, and releases them on the web. The GHTorrent project was launched in 2013. In GHTorrent, data is provided in the form of an SQL dump, as presented in Figure 2. GHTorrent's data include repository addresses, details of commits, and hash values. Thus, GHTorrent provides access to the GitHub repository data needed for addressing the risk of RepoJacking.

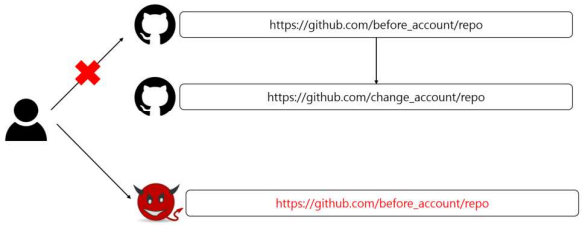
```
-- MySQL dump 10.13  Distrib 5.5.31, for debian-linux-gnu (x86_64)
--
-- Host: dutihr    Database: ghtorrent
-- ------------------------------------------------------
-- Server version       5.5.30-1.1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
```

[Figure 2] GHTorrent dump data head

C. RepoJacking

RepoJacking is an abbreviated form of the term "repository hijacking." As its definition says, the term refers to the malicious act of attackers hijacking repositories belonging to other individuals. RepoJacking is classified as a supply chain attack because it targets repositories, which are publicly released or available on the web to be utilized in various individual projects. RepoJacking is commonly executed by taking advantage of GitHub's repository redirection policy. GitHub has a policy that when users attempt to access their repository using their old account name after changing it, they will be redirected to a repository under the new account name. By exploiting this point, RepoJacking creates an account that has the same name as the old account name of a user and then generates a repository with the same name as the user's, thereby leading the user to unknowingly access the attacker's repository. The user will then mistakenly believe that the repository is the one under the user's previous account name, not the one belonging to the attacker. From then on, the user will execute any code available in the repository without any doubt. Figure 3 illustrates how RepoJacking attacks user repositories.



[Figure 3] RepoJacking attack

Using RepoJacking, malicious actors can execute the following three types of attacks.

1. The attacker collects a repository address available in the installation script of a project and creates a repository with the same name, inducing the user to be redirected to the attacker's repository and execute the code that the attacker wants to spread.

2. The attacker identifies a repository to which the user will be redirected for a reason, for example, due to a change to the username, and then creates a repository using the same name as the user's previous username before being changed to redirect the user to the attacker's repository and induce the user to unknowingly execute the code that the attacker wants to spread.

3. The attacker recognizes that, despite a change in the username, which results in a corresponding change in the repository link, some projects' repository releases have not updated the link address. The attacker then creates a repository using the user's previous username, leading the user to unknowingly access the attacker's link and execute the code that the attacker intends to spread.

The feasibility of these RepoJacking attacks is verified in Chapter 3.

D. Previous studies

The concept of "RepoJacking" first emerged in a blog hosted by Security Innovation in 2020 [6]. The blog [6] demonstrated that RepoJacking could be actually executed in three scenarios exploiting GitHub's repository redirection policy. Afterward, actual case studies on the vulnerabilities of repositories against RepoJacking were reported by multiple

research groups, such as the Aqua Nautilus team and the Checkmarx team [7, 8]. The Aqua Nautilus team introduced various vulnerability issues that might arise in the face of RepoJacking while describing the execution process of a PoC using actual repositories [7]. Meanwhile, the Checkmarx team presented protection measures taken by GitHub against RepoJacking. The team also described possible methods for bypassing such protection measures in order to execute RepoJacking [8]. A relevant previous study defined RepoJacking as a type of supply chain attack [9]. In another study [10], actual RepoJacking cases were introduced, and possible protection measures that could be taken against such a supply chain attack by open-source management teams were proposed.

III. Experiment

In this chapter, the actual process of RepoJacking, as describe earlier, is executed in practice. Afterward, a list of repositories is collected, and the number of repositories that are vulnerable to RepoJacking is estimated.

A. *RepoJacking exploit*

To practically verify the vulnerability of repositories against RepoJacking, two GitHub accounts were created as follows.

- Sktestme (attacker): An account that executes RepoJacking
- Sktestme (victim): An account that is targeted by RepoJacking

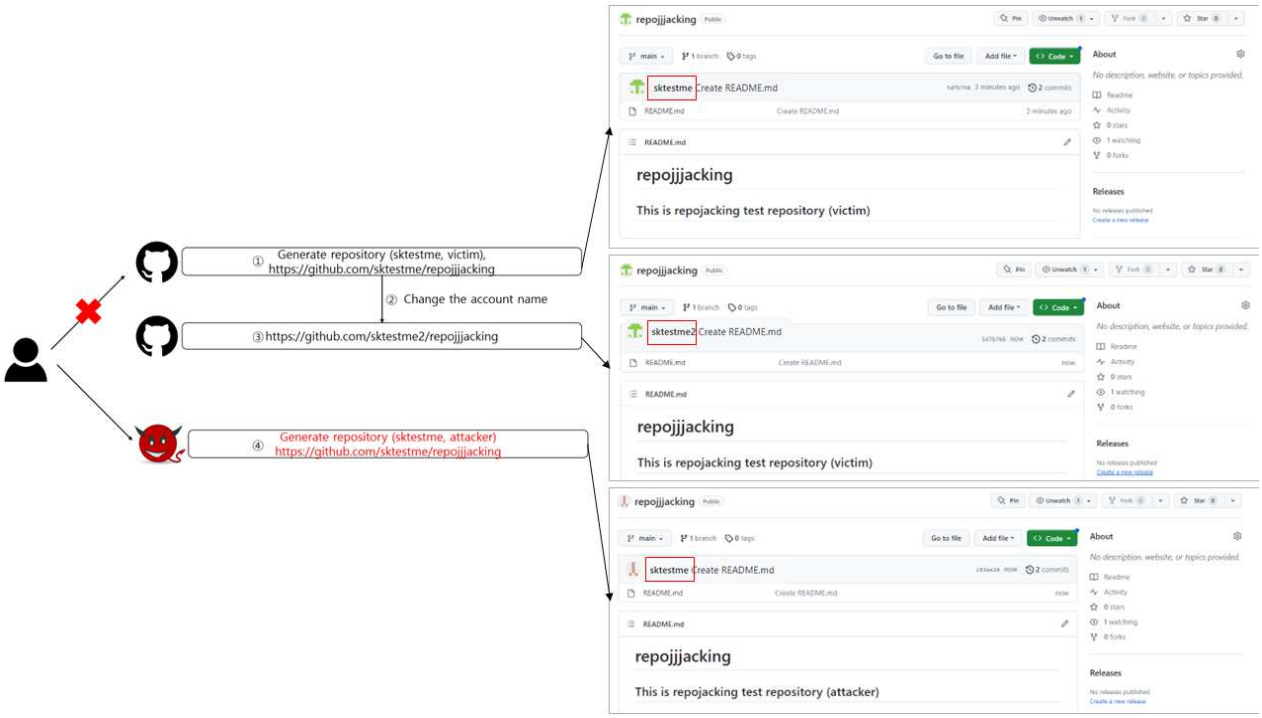
The vulnerability assessment process was performed according to the following procedure. The results confirmed the possibility that the tested repository could be exploited by RepoJacking. Figure 4 presents a schematic diagram of the developed PoC, illustrating how RepoJacking works, i.e., when attempting to access the repojjjacking repository belonging to sktestme (victim), the user is redirected to the repository owned by sktestme (attacker).

Detailed verification steps

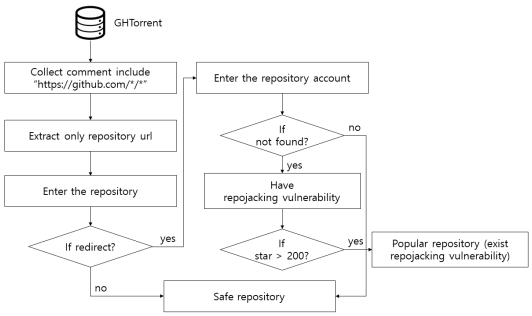
1. Create a repojjjacking repository using the account of sktestme(victim).
2. Change the account name of sktestme(victim) to sktestme2. Check to see if the user is redirected to the repository under the new account sktestme2 (https://github.com/sktestme2/repojjjacking) when attempting to access the link "https://github.com/sktestme/repojjjacking."
3. Create a new account named sktestme (attacker) and a corresponding repojjjacking repository.
4. Check to see if the user is redirected to the repository under the account of sktestme (attacker) instead of the one under sktestme2 when attempting to access the link "https://github.com/sktestme/repojjjacking."

B. *Mass RepoJacking Exploit*

SQL data for the period of 2013-2015 were collected from GHTorrent to identify repositories vulnerable to RepoJacking. Subsequently, the collected GHTorrent data were processed in a suitable form and assessed according to the same procedure as shown in Figure 5.



[Figure 4] RepoJacking PoC

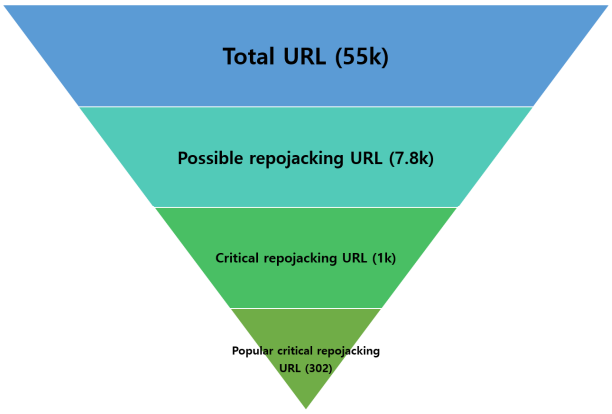


[Figure 5] RepoJacking verification flow

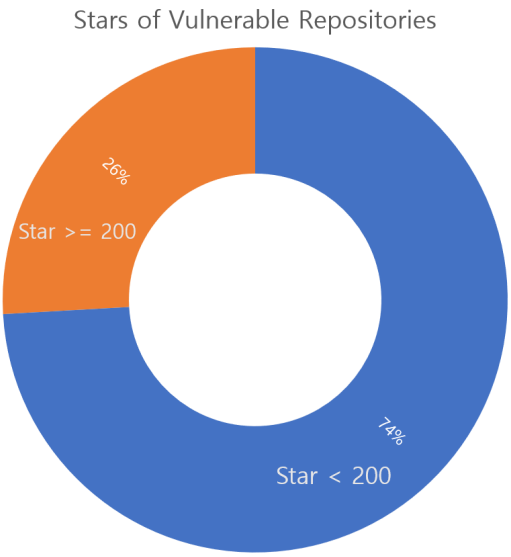
As a result, a total of 55,000 repositories were collected. The collected repositories were tested for their vulnerability to RepoJacking based on the following two criteria.

- *Vulnerable to RepoJacking*: Redirected to a new account or repository when attempting to access the target repository
- *Critical to RepoJacking*: Redirected to a new repository, also leading to the deletion of the previous account when attempting to access the target repository, which means that the target can be immediately attacked

The verification results demonstrated that 7,737 of the 55,000 repositories were *Vulnerable to RepoJacking*. The repositories recognized as being *Vulnerable to RepoJacking* were subjected to further verification to determine whether they were *Critical to RepoJacking*. Operators attempted to access their account address and checked the response. If the message "404 Not Found" was returned, the corresponding repository's previous account name was considered to have been deleted. The results confirmed that 1,039 repositories were *Critical to RepoJacking*. Next, the number of stars, a popularity indicator for GitHub's repositories, was analyzed for the 1,039 repositories identified as *Critical to RepoJacking*. The top 26% of them were found to have a rating of 200 stars or more, as shown in Figure 7. These repositories were classified as popular ones.



[Figure 6] Experimental results



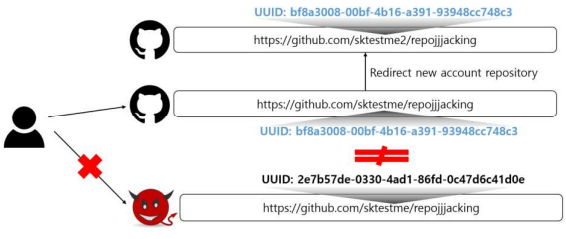
[Figure 7] Number of stars of the repositories vulnerable to RepoJacking

Out of the repositories that were *Critical to RepoJacking*, 302 were identified as popular repositories with a rating of 200 stars or more. The exploitation of these popular repositories for RepoJacking could be a major concern for various projects, developers, and users. Possible countermeasures against RepoJacking, which can develop into a significant supply chain attack, will be proposed in Chapter 4.

IV. Countermeasures

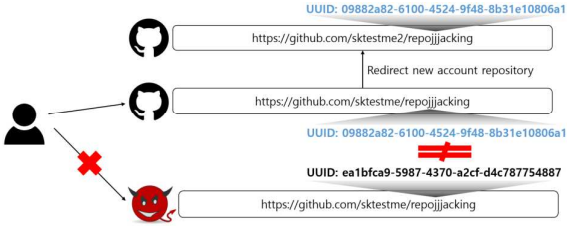
In an attempt to address vulnerability issues associated with RepoJacking, as discussed above, GitHub revised its policy to discard any popular repositories for which the number of copies is 100 or more when their account name is changed. Consequently, the corresponding repository names can no longer be used [10]. However, this protection measure can be bypassed, as demonstrated by a case study by the Checkmarx team [7]. Furthermore, it is not applicable in situations where the number of copies is fewer than 100. Thus, this approach does not constitute a fundamental solution. The present study proposes two methods for protecting against RepoJacking, as detailed below.

- Assign unique eigenvalues, such as UUID, to each account based on their creation time and other indexes. This approach ensures that individual accounts are separately managed, even if they have the same name.



[Figure 8] Account UUID

- Assign unique eigenvalues, such as UUID, to each repository based on their creation time and other indexes. This approach ensures that individual repositories are separately managed, even if they have the same name.



[Figure 9] Repository UUID

Assigning a unique UUID to each account limits the scope of integrity test subjects solely to accounts. This ensures that even if repositories share the same name, only integrality tests on accounts are necessary to prevent the user from being redirected to the attacker's repository. Meanwhile, assigning a unique UUID to each repository expands the scope of integrity test subjects to the entire group of repositories because only repositories possess unique UUIDs regardless of the associated account. Consequently, only integrity tests on repositories are needed, regardless of the account name, to prevent the user from being redirected to the attacker's repository. However, this approach is expected to be more costly compared to assigning UUIDs to account names.

The proposed approaches are expected to ensure that users will be properly redirected to the legitimate repository, rather than the attacker's repository, through integrity tests on both accounts and repositories, even if the same account or repository names are used.

This paper also proposes a simple tool designed to assess the integrity of repositories against RepoJacking. The functioning of the tool is outlined as follows.

1. Access GitHub's repository given as input.
2. Upon accessing the repository, if its address differs from the one provided in the input (i.e., redirected to another repository), the corresponding repository is categorized as *Vulnerable to RepoJacking*.
3. If the target repository is identified as *Vulnerable to RepoJacking*, access the corresponding account address. If the response is "404 Not Found," it is categorized as *Critical to RepoJacking*.
4. If the target repository does meet the conditions specified in 2 and 3, it is categorized as "*Safe*."

This verification tool is available for use in [12].

V. Conclusions

The present study verifies the integrity of actual repositories against RepoJacking, a malicious attack executed by exploiting Github's redirection policy. This malicious action is known to evolve into a supply chain attack. This study also proposes two methods for protecting against RepoJacking, along with a simple tool designed to assess the integrity of repositories against RepoJacking. RepoJacking can potentially escalate into a supply chain attack, leading to the execution of the code intended by the attacker across multiple PCs or servers. Thus, it could pose a significant threat to cybersecurity. The proactive protection measures against RepoJacking proposed in this paper are expected to mitigate potential threats associated with RepoJacking.

Acknowledgment

This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy (MOTIE) of the Republic of Korea (No. 20212020800120).

References

[1] "GitHub." GitHub. Accessed July 6, 2023. https://github.com/.

[2] "About." GitHub. Accessed July 4, 2023. https://github.com/about.

[3] "TTA 정보통신용어사전." TTA, Accessed July 31, 2023. https://terms.tta.or.kr/dictionary/dictionaryView.do?subject=%ED%95%98%EC%9D%B4%EC%9E%AC%ED%82%B9

[4] "Git." Git, Accessed July 6, 2023. https://git-scm.com/

[5] Gousios, Georgios. "The GHTorrent dataset and tool suite." . In Proceedings of the 10th Working Conference on Mining Software Repositories (pp. 233–236). IEEE Press, 2013.

[6] "RepoJacking: Exploiting the Dependency Supply Chain." Security Innovation. Accessed July 21, 2023. https://blog.securityinnovation.com/repo-jacking-exploiting-the-dependency-supply-chain

[7] "GitHub Dataset Research Reveals Millions Potentially Vulnerable to RepoJacking." Aquasec. Accessed July 6, 2023. https://blog.aquasec.com/github-dataset-research-reveals-millions-potentially-vulnerable-to-repojacking

[8] "GitHub RepoJacking Weakness Exploited in the Wild by Attackers." Checkmarx. Accessed July 6, 2023. https://checkmarx.com/blog/github-repojacking-weakness-exploited-in-the-wild-by-attackers/

[9] Cordey, S. (2023). Software Supply Chain Attacks. An Illustrated Typological Review (J. Bund, B. Scharte, S. Soesanto, & T. Grossman, Eds.). doi:10.3929/ethz-b-000584947

[10] "GitHub repojacking attack: 10 lessons for software teams." Reversing Labs. Accessed July 21, 2023. https://www.reversinglabs.com/blog/github-repojacking-10-lessons-for-software-teams

[11] "GitHub patches bug that could allow access to another user's repo." Portswigger. Accessed July 6, 2023. https://portswigger.net/daily-swig/github-patches-bug-that-could-allow-access-to-another-users-repo.

[12] "repojacking_check." GitHub. Accessed July 13, 2023. https://github.com/p3ngdump/repojacking_check