language=Python,keywordstyle=**blue**

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelor thesis

Privacy implications of exposing Git meta data

presented by

Arne Beer
born on the 21th of December 1992 in Hadamar
Matriculation number: 6489196
Department of Computer science

submitted on December 16, 2017

Supervisor: Dipl.-Inf. Christian Burkert
Primary Referee: Prof. Dr.-Ing. Hannes Federrath
Secondary Referee: Prof. Dr. Dominik Herrmann

Abstract

Even if you're not doing anything wrong, you are being watched and recorded.

Edward Snowden

Contents

Acronyms

API access point interface

FS file system

Chapter 1
Introduction

Git is a code version control system which is used by most programmers on a daily basis these days. According to the Eclipse Community Survey about 42.9% of professional software developers used git in 2014 with an upward tendency [1]. It is deployed in many if not most commercial and private projects and generally valued by its users. It allows quick jumps between different versions of a project's code base and to manage and merge code from different sources to one upstream.

Several million users send new commits to their Git repositories every day. On Github alone, the currently biggest open source platform, there exist about 25 million active repositories, a total of 67 million repositories and about 24 million users [2].

Some well known projects and organizations use Git, for example Linux[3], Google[4], Adobe[5] and Paypal[6]. Every repository contains the complete contribution history of every contributing user. Each commit contains the full directory structure, a link to a blob for every file, a timestamp, a commit message from the author and more additional metadata.

This raises the question how much information is hidden in the metadata of a Git repository and which attack vectors could be introduced by mining this information, regarding a contributer or the owner of the repository.

The newly gained knowledge could be utilized by employers to spy on their employees. It could be used by an unknown attacker who aims to obtain sensitive information about a company and its employees trough their open-source projects. It is even possible that a privat person wants to monitor another person that regularly contributes to open-source repositories.

As there have not been any papers published about this specific topic or at least no public paper and Git plays such a crucial role in todays information technology, I want to investigate and evaluate this potential threat.

[1] Ian Skerret. Exclipse Community Survey 2014 Results. `https://ianskerrett.wordpress.com/2014/06/23/eclipse-community-survey-2014-results/` Retrieved Oct. 25, 2017

[2] The State of the Octoverse 2017, Retrieved Oct. 25 2011, `https://octoverse.github.com/`

[3] `https://github.com/torvalds/linux`, Retrieved Nov. 24 2017

[4] `https://github.com/google`, Retrieved Nov. 24 2017

[5] `https://github.com/adobe`, Retrieved Nov. 24 2017

[6] `https://github.com/paypal`, Retrieved Nov. 24 2017

1.1 Motivation

1.2 Leading Questions and Goals

The ambition of this thesis is to find possible attack vectors for knowledge extracted from git metadata of a single or multiple git repositories and analyze the possible damage potential. For that purpose I will look at three realistic attacker models and try to get as much compromising and harmful knowledge for the objective of each specific model.

In the following three different attacker models with potential goals are listed. Some goals will probably be extended, changed, added or removed during the research process [**book:pro-git**].

Employer The employer tries to get as much information about its employees with the intention of spying on them:

- Direct comparison of productivity between employees
- Compliance of working hours
- Check if employees work on external projects during working hours
- Code quality between employees

Industrial Espionage The attacker tries to get as much information from the public open-source projects of a company:

- Company members
- History of all employees
- Overall status of a project
- A project's Code quality
- Internal team structures

Individual Somebody tries to get as much information about the personal life of a contributing individual:

- Sleeping rythm and daily routine
- Sick leave and holiday
- Interests
- Programming languages and skills
- Personal relationships between various programmers

Chapter 2
Data Aggregation

The biggest initial task for this thesis was the acquisition of data. The data should be as extensive as possible, feature a high conjunction between contributers over several repositories to verify a possible connection between those and have realistic meta data. Two different solutions approaches existed for these requirements.

The first approach was to design an algorithm for automatic local generation of repositories. The main problem with this solution is, that the visualization and data mining code might be highly optimized for this specific generation algorithm. Real world data is noisy and inconsistent. Thereby the developed solution might have worked on the generated data, but would have probably failed on real world data.

The second solution was to get real world data from somewhere. The most obvious choice was to mine data from open source projects. I chose Github for this purpose, as it hosts one of the biggest collection of open source projects and provides a great **api!** (**api!**) for querying Github's meta data. A problem with this approach is that we don't have access to all important meta data, as for example the full list of members for organizations or the internal team structure of organizations. Another problem are old email addresses, which are not related to any account anymore. Even though some ground truth is missing, I decided to use this approach as it was still the most viable and promising way to gain as much ground truth and real world noise as possible.

2.1 Structure of the Data

Before we get to the data aggregator, I want to explain the internal git storage data structure and mechanisms, which are important for the purpose of this thesis [**book:pro-git**].

Git, as most programmers know it, is a collection of tools to work with it's underlying file system (**fs!** (**fs!**)) on a high abstraction level. Every file which is added to a git repository is internally saved in the .git/objects under the respective SHA-1 of the file.

2.2 Problems

2.3 The Aggregator

List of Figures

List of Tables

Eidesstattliche Erklärung

„Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht."

Ort, Datum

Unterschrift