A Qualitative Analysis of Security Keys and Commit Signing on World of Code

Parker Collier
UTK
Knoxville, TN
pcollie4@vols.utk.edu

John Sadik
UTK
Knoxville, TN
JohnS@uk.edu

Anthony Roman
UTK
Knoxville, TN
aroman@vols.utk.edu

ABSTRACT

GitHub is the largest collaborative code development platform. Code contribution on this platform is not verified by default and requires explicit user action. We set out to investigate commit information on the World of Code project as well as on previous data we have acquired from the GitHub platform itself. We investigate the usage rates of users with keys and of general commit information on a large portion of GitHub. As well as a comparison between the data on World of Code and our own collected data

KEYWORDS

Github, Cryptography, Commits, Git, Signatures

1 INTRODUCTION

Git is the most prominent code versioning and development tool in today's coding scene. Its primary purpose is to allow for multiple developers to contribute to a code base simultaneously from any machine. Using Git, developers can produce code locally on their own devices and then add that code to a remote repository. Git also tracks the changes made to a repository by each individual contributor. This system of remote development and tracking is necessary for many production software pipelines. There are a variety of Git implementations, the most popular of these are GitHub, Bitbucket, and GitLab.

Of the three variants mentioned, GitHub is the most popular by far, especially for the purpose of open-source and personal development. GitHub serves as a remote database for over 300 million public repositories. Many of these repositories are widely used and distributed. While GitHub is primarily a code development and versioning tool, it has grown into a large-scale distribution platform that is used in many ways outside of this original purpose. Developer can distribute their code in a variety of ways. The releases tab allows for executables to be downloaded directly by any user who can see the repository. More complex features like GitHub Apps or GitHub Actions allow for the use of webhooks and build tools to automatically update and serve web applications such as websites or bot applications. Users can also clone any repository they have access to and run the code that way. Many users rely on GitHub code projects and often times these programs are distributed automatically, This makes the integrity and security of these repositories and developer accounts critical.

Any user can clone a public repository, make changes to their local version, and then attempt to merge their code back into the source repository. It is up to the discretion of the repository owners to verify the validity of this code and accept or reject the changes. When such changes are made, they are tied to commits, which track who did what for the current change. By default, there is no verification process for commits, meaning any user can claim to be any other user on GitHub. GitHub makes use of cryptographic keys to verify users accounts when pushing or pulling code. GitHub also optionally allows the use of cryptographic signatures for the purpose of user identification.

We set out to investigate the usage of these signatures. There is little other work investigating the usage of security keys on GitHub. This work is a continuation of an earlier work from the USER Lab at UTK that is collecting all user keys on GitHub in order to perform a quantitative analysis of these keys and their usage rates. Previously we collected keys from the GitHub API, this work is focused on comparing this collected data to information found on the World of Code project.

The World of Code project is a large-scale collection of git-based code forges. The project is open to researchers upon request and has collected information from the majority of public repositories on GitHub. This makes it ideal for additional investigation of real-world code examples on GitHub. We performed a few random scans of general data on World of Code as well as some comparisons between our previously collected data and data found on World of Code. While the focus of our previous work was on both types of GitHub keys, this one is looking only at keys used for the purpose of commit signing

2 BACKGROUND

Commit signatures can be made via two systems GPG and SSH.

- **GPG**. GPG or Gnu Privacy Guard is a cryptographic protocol primarily designed for creating digital signatures and keys. We have found it to be the most popular signing system in terms of usage and ownership on GitHub. GitHub's specific GPG implementation is PGP (Pretty Good Privacy).
- **SSH**. SSH or Secure Shell Hosting is a network tunneling tool most commonly used for remote connections to server machines. It is also the method employed by GitHub for user authentication when pushing or pulling code. GitHub allows for the use of these keys for both user authentication and commit signing. The same keys can be used for both purposes. We have found low usage rates of these keys in practice.

2.1 Previous Work

Our previous work scanned around 20 million user profiles for keys and information. We found signing key ownership to be around 1.2% of users. We also found that most users on the platform are not active, with a small portion being responsible for a large amount of the development activity on the platform. Our goal is to investigate

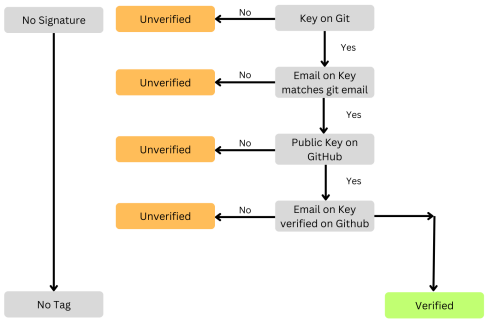

Figure 1: Flow chart of signature verification conditions

available data on World of Code to see if the real commit behavior aligns with our collected data.

2.2 Signature Verification

Signatures are created locally via Git and are then evaluated and marked with a verification status by GitHub. The tag given by GitHub is either verified or unverified, there are a variety of reasons why a signature might not be verified. The signature paired email must match the email set on Git, the public key of the signature must be uploaded to GitHub, and the email used in the key must be verified on the same account on GitHub. If any of these conditions are not met, the commit will be marked as unverified and will give the reason for the unverified status when clicked on. Figure 1 contains a flowchart visualizing these requirements.

We are interested in finding out how many users use the keys stored on their GitHub account as well as how many are using keys not stored on their account. There is one additional way to receive a digital signature on a commit. Commits can be created from the GitHub website itself, this is done via the edit option on the repository or through GitHub's online IDE. Any commit created this way will be signed by GitHub's own GPG key.

3 RELATED WORK

The World of Code project is an ongoing work with various associated publications. The focus of the current work on the project is data quality and collection. [3, 4]. This is our most closely related work, however we are focused on security topics within World of Code. There are a variety of other works looking at GitHub as a topic for security. Much of this work seems to be focused on sentiment analysis by either investigating GitHub issues or discussions. One work from the University of South Florida developed a model for sentiment analysis across a variety of platforms used for discussion [2]. A similar paper *Security and Emotion: Sentiment Analysis of Security Discussions on GitHub* [6] also employs a natural language model to investigate sentiment patterns on discussions relating to security on GitHub. While these works are interested in security on the platform, their focus is on sentiment analysis rather than user activity and integrity. We did manage to find two papers looking into the security of the GitHub platform in regards to its Copilot language model. [1, 5]. These works are focused on the security of code on the platform and found that a significant portion of code generated by Copilot is insecure. These works are focused on user interaction with secure systems, however they do not have any interest in cryptography large data.

4 METHODOLOGY

The majority of our work involved using command line tools work with data on World of Code as well as using Python for data analysis and visualization. Our work involved three major steps for analysis, looking at Global commit information, looking at commit information of users known to have keys, and comparing information found on them to data we have collected via the GitHub API.

4.1 Global Commits

We first set out to get a general scan of signed commits on GitHub. This was done to compare against our known key ownership values gathered from GitHub's API. We used the method defined in the World of Code tutorial for parsing large numbers of commits at once. This data is gathered via command line tools on the World of code servers, the general command used was:

```
for i in {0..10}; do perl /lookup/lstCmt.perl 3 $i;
 done | grep -B 4 "gpgsig ——BEGIN PGP SIGNATURE——"
```

The grep command was changed to search for both SSH and GPG signatures, as well as finding all signatures over the range. We ran the search over the first 10 of 127 maps to reduce search time.

4.2 Known Key Owners

Once this data collection was done, we ended up with 66,095 users from WoC that we knew had keys on GitHub. Using this information, we began our analysis. First, we examined some general metrics about the data, such as total and average number of commits. In examining this data, we found several users who did not have commits associated with their accounts on WoC or who for other reasons were unable to give us data. We removed these users and ended up analyzing the 48,664 remaining users.

We followed this up by taking a random sample of at least 10,650 commits to analyze what kinds of commits are out there. Because we were constrained by time, we could not check every single commit in this portion of the analysis, but instead we chose this target number of commits to afford us a 99% confidence level with a 1.25% margin of error in our analysis. In the end, we randomly collected 13,934 total commits in case there were any problems with the chosen commits, but we did not run into any such problems.

4.3 General WoC Users

After this analysis was complete, we also took a random sample of 12,220 WoC users to compare our data to. In essence, we wanted a similar sample size to compare random WoC users with users with known keys. To collect this data, we generated 32 random numbers summing to 12,220. We used those random numbers to select a random sample from each of the 32 maps **/da1_data/basemaps/gz/A2cFullV?** where the ? represents which number map it is. Using this random selection of users, we conducted the same analysis. While the initial analysis was complete at the time of writing this paper, unfortunately, the signed-key analysis was not. We report on the findings we do have.

4.4 Comparisons

We then investigated the usage information of GitHub keys among the commits of our known users. First we gathered all keys for each users known to be in world of code and stored this into a csv format along with their Author name and GitHub user name, then all user commits were parsed into lists by Author name along with all Key IDs pulled from their accounts, we used the Python pgpy package to parse the signatures found on their signed commits to create a set of commits associated with an Author. We then go through each World of Code author and find out how many commits each key has, and if each key is found in our database of user keys.

5 RESULTS

Our scan of global commit data included 366,877,605 commits from World of Code. Out of these commits, 57,967,631 are signed with GPG keys and 111,490 of them are singed with SSH keys. In total around 16% of our commit sample is signed. The vast majority of these signatures are GPG which makes up 99.81% of the signed commits we found. This shows a large percentage of commits with signatures when compared to the 1.2% of known users with signing keys collected through our previous work. These commits are taken from World of Code at random and contain historical collections of data, meaning many of the commits are likely to be duplicated. We looked into a more specific search of specified user groups in order to better understand user behaviour around commit signatures.

5.1 Analysis of Users with Known Keys

When looking at our 48,664 users who we knew had keys, we found that in total they had 210,037,164 commits. This averages out to about 4,316 commits per a user, but this number is only scale purposes as the top 4 users in our data set account for 52.9% of the commits and the top 3,395 users in our dataset account for 90% of the commits. In fact, taking out just the top 4 users, the total number of commits drops to 98,966,799 and the average number of commits per user drops to 2,033.84.

At the low end, many users had only a single commit. At the high end, dependabot[bot][1], which is owned by GitHub and is used to automatically perform software and security checks, had 83,714,239 commits. The top 20 users and their number of commits are shown in Table **??**. Looking at the results, most of the top 20 users are bots. The top real user is likely many users who have used GitHub's template for setting up commits, using "Your Name" and "<you@example.com>" instead of their actual information. Similarly, the user with 2,250,350 commits is likely a collection of users who just left the fields blank, and the user with username a is also likely a collection of users using a common letter instead of their real information. It's not until we get to Anudit Nagar that we find a non-bot user we can look up on GitHub. Even then, examining his account yields information that 1 million of his commits come from a script used to generate 1 million commits. While interesting in its own right, we continue past him to Brian Chan who is the first non-bot user on this list that has conventional commits and can be found in GitHub. We discuss the implications of this list in Section 6.

[1]https://www.darkreading.com/application-security/supply-chain-attackers-escalate-with-github-dependabot-impersonation

Number of commits	Username
83,714,239	dependabot[bot]
11,348,161	Upptime Bot
8,578,591	dependabot-preview[bot]
7,429,374	github-actions[bot]
5,110,040	Renovate Bot
2,913,684	Your Name
2,775,512	greenkeeper[bot]
2,601,652	renovate[bot]
2,511,525	GitHub Action
2,316,184	readme-bot
2,250,350	
2,048,455	BuildTools
1,906,497	pyup-bot
1,448,683	Automated
1,376,363	github-actions[bot]
1,370,611	a
1,092,256	Linux Build Service Account
1,049,828	snyk-bot
1,001,949	Anudit Nagar
900,106	Brian Chan

Table 1: The top 20 users by number of commits.

One natural question we had while doing the analysis was how many users are actually signing their commits. So we took a random sample of 13,934 commits and analyzed whether those commits were signed or not. In total, we found that only 4994 (0.36%) of those commits were signed. This comes as a surprise considering these users are known to have keys, so we expected to find a much higher percentage of signed commits.

5.1.1 Comparison to WoC Users. Finally, we wanted to compare our findings for users with known keys to random World of Code users. To do so, we first randomly selected 12,220 users from WoC and then we ran the same type of analysis. We found that these users had a total of 431,100,724 commits for an average of about 35,278 commits per user. Again, we show the top 20 users and their commit count in Table 2. Unlike the previous list, there are much fewer bots in this list. However, most of these top users have some kind of script that generates trivial commits, circumventing the typical use of commits.

5.2 Analysis of User Keys

We then investigated all commits of a sampling of users known to have keys in order to better understand which keys they were using to sign commits. We first collected all of the signed commits from each of these users then compared the key ids of all of these commits with our known keys for each user as well as our known keys that belong to GitHub. We found that 62% of these users are only using GitHub's keys for signatures, and that only 24% of these commits are using keys that could be found in our database. Out of all of the commits 55% were signed using GitHub's keys. Users with personal signing keys are the most active, with 10.98 commits on average compared to an average of 5.24 commits on average for users who only use GitHub for signatures. Users have about 1.12

Number of commits	Username
83,714,239	dependabot[bot]
11,008,817	Maxim-Costa
9,999,999	Lotsa Commits by Abheek Dhawan
8,901,562	Ivan Buiko
7,791,996	Jackson
5,859,080	Haocheng Hu
5,172,113	Gerrit User 1062513
5,138,378	Commit
5,110,040	Renovate Bot
4,710,838	Tom Forbes
4,641,881	Florian Lejosne
4,564,065	Gerrit User 1111084
3,925,560	Ubuntu
3,684,002	OTube
3,579,205	Lassi Haasio
3,354,883	IThinkImOKAY
3,220,104	MaximCosta
2,913,684	Your Name
2,742,342	Jason Calabrese
2,672,945	Syed Umar Arfeen

Table 2: The top 20 WoC users by number of commits.

Username	Commits	Github	Known	Unknown
Josh Dolitsky	770	0	0	770
Andrew Fergusson	421	79	342	0
Madeline Miller	324	0	324	0
dnitsch	316	88	228	0
rafern	315	0	315	0
Wilfried Chauveau	310	101	209	0
Andrew Johnson	288	81	0	207
Leonardo Benicio	246	83	163	0
Tyler Richards	190	190	0	0
Galdan Moulinneuf	182	13	0	169
Andrés Herrera	126	27	48	51
Zhiwei	113	113	0	0
MinionAttack	96	0	96	0
Pat Nafarrete	82	82	0	0
Ladhari Oussama	80	80	0	0
Luke Martin	77	11	0	66
CmdC0de	77	77	0	0
Julien	73	32	41	0
Loïc Houpert	73	73	0	0
Timo Anttila	73	0	73	0
Andrey Tsibin	70	38	0	32

Table 3: The top 20 users by number of commits, including categories for commits with GitHub signatures, commits signed with keys found in our database, and commits signed with keys we do not have recorded

keys on average, trending towards a single key per account. Table 3 shows the top 20 user accounts by number of commits and the counts of which categories their commits fall into.

6 DISCUSSION

6.1 Actually Signing Commits

In our analysis, we find that only about 15.8% of commits analyzed were signed. Even worse, in our sample of commits from users who we know have keys, only 0.36% of commits were signed. One reason for this large discrepancy is that some users with a lot of commits, especially bots like dependabot, sign all their commits, contributing an immense number of signed commits. It is best to view the first percent, 15.8%, as being closer to the actual number of commits being signed while looking at the second percent, 0.36%, as a representation of the percent of users contributing to these signed commits. This is interesting because it shows that there are a number of users who have signing keys who are either not using them or selectively using them.

Future Work. This leads us to recommend future work explore how users are using their signing keys to further understand why some users are not using their keys everywhere. We suspect that some users might have keys specifically tied to only part of their account, such as for work or specific projects at their job, but we believe future work can shed light on this area. A large portion of the signatures we found can also be attributed to GitHub's signature system, these signatures are generated by GitHub when a commit is created on the GitHub website, finding over half of the commits in our sample of users known to have keys were solely signed in this fashion was surprising. This indicates that a large amount of signed commits from actual users are not from any personal key, which also helps to explain why key ownership is low compared to the percentage of signed commits on the platform.

6.2 Bots and Attack Vectors

As seen in our analysis, a majority of the commits we saw, both signed and unsigned, were through the use of bots. While this may help in production workflow, it defeats one of the biggest advantages of commits: knowing who is actually making the changes. While there may be specific circumstances leading to needing to use bots to handle the commits, it is still interesting that it opens up other attack vectors for allowing commits to show up as verified even if they were not made by a legitimate party. For example, in the current GitHub signature verification system, an attacker needs to either steal a private key from an active account or phish a user into uploading the attacker's private key. Once this is completed, an attacker would then be able to use that key to sign commits and make them look verified by the user they are targeting. With the use of bots to sign and push commits to GitHub, the previous attack still works for the bot. However, there is also the opportunity for attackers to gain access to these bots, allowing them to bypass the need to steal or phish a private key. Additionally, if a bot or selection of bots are used across many projects, attackers may opt to create fake bots that look like the real bots as a way to attack users. The end goal of such an attack may be aiming to trick users into adding this malicious bot to an important project where the attackers can then cause some damage. Importantly, the security of verified commits is now in the hands of the team who is leveraging

and/or developing the bot instead of being fully under GitHub's control.

Future Work. We believe future work can focus on further exploring how these bots are used in production workflows and how many bots are used by more than one project at a time. If bots are mostly used on a single project, then it would provide less of a security risk if a single bot was compromised.

6.3 Trivial Commits

In our work, we also find millions of what we call trivial commits. Such commits are created by scripts whose primary goal is to artificially inflate the number of commits. This can be done for many reasons, but in every case the commits added in this way are not substantive. There is usually no informative commit message, code being pushed, or useful trend in the commit frequency. While not a problem on its own, if any future work aims to analyze commits we recommend that they keep in mind the so-called trivial commits and commits made by bots. These might not be the commits that these researchers want in their dataset.

7 FUTURE WORK

In addition to the future work already outlined in the discussion, we suggest that researchers could develop a tool that scans through a repository and then creates a report of the signature status of its commits. Doing this will allow developers to more quickly assess the validity of commits on their projects. Having developers do this would greatly increase the security on their projects. If developers did this they would know for certain if the right person was committing to their GitHub projects. Such a tool may also eliminate or reduce the need for bots pushing commits if these bots are used to verify users. Performing in depth analysis projects with certain thresholds for popularity or reputability such as those with a high number of forks or stars would be a strong next step for this work. This would likely reduce the number of bad commits and ensure that the data being investigated is related to real development environments.

7.1 Limitations

Our two sources of data, the GitHub API and World of Code, were collected via different sources and at different times. This means there is likely some discrepancy between statistics found on either source, especially so when making comparisons between the two. Our dataset from the GitHub API is also incomplete as API limitations prevent us from gathering data more quickly. So our current analysis is only on a portion of that data. Data on GitHub itself is also dangerous especially in the case of commits, many commits on GitHub are trivial commits or obscured by bots, making impactful user analysis more difficult. Finally, while we do have a script running to collect random commits on World of Code and analyze them for signatures, we were not able to complete this script in time for this analysis.

8 CONCLUSION

Signature usage on GitHub is a nuanced metric. At face value around 16% of commits on GitHub are signed, however upon further investigation we found a large number of those commits are likely to have come from bot accounts or are signatures created by GitHub themselves. Individual users are significantly less likely to have created a signing and the ones who do have signing keys are not likely using them to sign every commit. We also found that it is dangerous to investigate all commits on GitHub as there are a number of commits that are generated for trivial purposes and do not contribute to any actual code base. More work on the topic can be done, especially work that more specifically looks into known popular repositories. Many commits are either not signed or not signed with a registered key. To this end we have begun development of a tool to scan a repositories commits and create a report of the status of all of its commits. This is to help developers create more secure and accountable code bases moving forward.

9 CONTRIBUTIONS

(1) Parker Collier
Previous work on Git Key collection, key comparisons and analysis, development of repository scanning tool, writing.
(2) John Sadik
Developing tools to gather users and commits from World of Code, random sampling of commits and users from World of Code for analysis, mapping commits to users for GitHub, exploratory analysis, and writing.
(3) Anthony Roman
Data collection and writing.

REFERENCES

[1] Yujia Fu, Peng Liang, Amjed Tahir, Zengyang Li, Mojtaba Shahin, and Jiaxin Yu. 2023. Security Weaknesses of Copilot Generated Code in GitHub. arXiv:2310.02059 [cs.SE]

[2] Sameera Horawalavithana, Abhishek Bhattacharjee, Renhao Liu, Nazim Choudhury, Lawrence O. Hall, and Adriana Iamnitchi. 2019. Mentions of Security Vulnerabilities on Reddit, Twitter and GitHub. In *IEEE/WIC/ACM International Conference on Web Intelligence* (Thessaloniki, Greece) *(WI '19)*. Association for Computing Machinery, New York, NY, USA, 200–207. https://doi.org/10.1145/3350546.3352519

[3] Yuxing Ma, Chris Bogart, Sadika Amreen, Russell Zaretzki, and Audris Mockus. 2019. World of code: an infrastructure for mining the universe of open source VCS data. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 143–154.

[4] Yuxing Ma, Tapajit Dey, Chris Bogart, Sadika Amreen, Marat Valiev, Adam Tutko, David Kennard, Russell Zaretzki, and Audris Mockus. 2021. World of code: enabling a research workflow for mining and analyzing the universe of open source VCS data. *Empirical Software Engineering* 26 (2021), 1–42.

[5] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2022. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*. 754–768. https://doi.org/10.1109/SP46214.2022.9833571

[6] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. 2014. Security and emotion: sentiment analysis of security discussions on GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 348–351. https://doi.org/10.1145/2597073.2597117