Individual Plan

Johan Moritz

February 2022

PROJECT INFORMATION

Preliminary title: Decentralized Distribution of .buildinfo Files

Student: Johan Moritz, jmoritz@kth.se

Examiner at KTH: Mads Dam

Supervisor at KTH: Giuseppe Nebbione

Supervisor at Subset: Peter Jonsson, peter.jonsson@subset.se

Current date: February 7, 2022

Keywords: Reproducible builds, Security, Decentralized trust, buildinfo, P2P

BACKGROUND & OBJECTIVE

Discussions on how to verify the lack of malicious code in binaries go at least as far back as to Ken Thompson's Turing award lecture [1] where he discusses the issues of trusting code created by others. In recent years, several attacks on popular packages within the Free Open Source Software ecosystem (FOSS) have been executed [2] where trusted repositories have injected malicious code in their released binaries. These attacks question how much trust in such dependencies is appropriate. In an attempt to raise the level of trust and security in FOSS, the reproducible builds projects [3] was started within the Debian community. Its goal was to mitigate the risk that a package is tampered with by ensuring that its builds are deterministic and therefore should be bit-by-bit identical over multiple rebuilds. Any user of a reproducible package can verify that it has indeed been built from its source code and was not manipulated after the fact simply by rebuilding it from the package's .buildinfo file. These metadata files for reproducible builds include hashes of the produced build artifacts and a description of the build environment to enable user-side verification. .buildinfo files are by this notion the crucial link to ensure reproducibility, which also means that a great deal of trust is assumed when using them. Current measures for validating .buildinfo files and their corresponding packages involve package

repository managers and volunteers running rebuilderd [4] instances that test the reproducibility of every .buildinfo file added to the relevant package archive. This setup allows users to audit the separate build logs, thus confirming the validity of a particular package. However, because this would be a manual process and the different instances do not coordinate their work, it relies on the user judging on a case-by-case basis whether to trust a package or not. This project seeks to reduce some of that burden from the user while increasing their trust in the software they use by investigating possible decentralized solutions for distributing and proving the correctness of .buildinfo files.

The project is carried out at Subset, an IT-security consulting company focusing on critical systems. They see reproducible builds as a step for raising their trust in FOSS, which is relevant for them to be able to use such software and technologies themselves in their work for their customers.

Necessary background knowledge

Besides the information gained from the literature study and a general understanding of computer science, no additional background knowledge is needed to complete this project.

RESEARCH QUESTION & METHOD

Question To what extent can .buildinfo files be provably verified in regards to upholding user trust?

Objectives Explore the research question through a system with the following requirements:

- Multi-parti verification of .buildinfo files
- Public access of provably verified .buildinfo files
- Minimized risk of single-point-of-failure

Such a system is assumed to increase user trust in .buildinfo files if it exists. The project objectives are then to research, construct and test such a system with current technologies in order to verify the extent that said system lives up to the listed requirements.

Tasks

- Research current state of the art in peer-to-peer and consensus algorithms to make a qualified technology choice as the base for the system. This research involves understanding the functional and technical requirements of the system itself, so the choice is suitable. The main challenge is to understand the consequences of such a technology choice in a production system.

- Construct a decentralized system for distributing .buildinfo files based on the technology choice from the previous task. This is the main production task of the project and, while probably quite straightforward, could be challenging in terms of time management.
- Setup a simulated production environment for testing properties of the system. Test scenarios should be easy to describe and set up to allow for easily rerunning tests. This, though, will add to the complexity of the project.
- Run scenarios on the production environment to test whether the system follows the requirements or not. Scenarios where some nodes in the system are acting maliciously and scenarios with reduced availability should be included.
- Stress-test the system with an estimated production workload. It is probably hard to estimate the actual workload of such a system, so perhaps a general notion of performance will have to do. One such simplification is measuring the retrieval latency of .buildinfo files for an increasing number of requests.

Method

- To realize the first task of finding a suitable base technology for the system, a literature study and a comparison of possible algorithms and solutions must happen. The comparison should especially consider the system requirements to ensure that the technology choice is appropriate.
- Testing the requirements conformity of the system is to be done by judging the threat of availability and integrity compromise in the system. As described in the tasks above, this is done by artificially introducing faults and malicious actors in the testing scenarios. This way, the level of conformity to the system requirements is directly correlated to the results of the tests.

Ethics and Sustainability From an ethics point of view, there is the question of who has the right to validate a .buildinfo file. In other words, whom users should trust. This project does not serve to enact any such policies directly but could assume that the parties involved in running the nodes on the system are globally known and trusted by the users. They could, for example, be institutions or other well-known organizations.

From a sustainability perspective, there is a consideration to be made in that a distributed system could have a large energy consumption. Throughout the project, we should try and reduce such issues and especially choose technologies with this point in mind.

Limitations The project focuses on exploring and understanding the needs and requirements of decentralized technologies for distributing .buildinfo files. By its exploratory nature, a focus will not be on testing and comparing different implementations. Instead, the base technology is chosen based on the literature study, and any analysis is done on the implementation based on that particular choice.

Risks Building and testing a distributed system is not an easy task. That is especially true when cryptographic invariants should be followed. It could very well turn out that implementing the system in practice is too great of a task for the limited time available. In this case, I see two possible ways to still go forward with the project. The primary choice would be to simplify the system as much as possible, in particular, loosen various security requirements that should be present in a production system. If this too turns out problematic, a second option is to instead model the system with a tool such as TLA+ [5] where implementation details would not have to be worried about.

EVALUATION & NEWS VALUE

Hopefully, the project will show that we can build an efficient and resilient system with current-day technologies which enhance the integrity and availability of .buildinfo files. Such results would be highly relevant to solve some of the issues of trusting package archives. They could also possibly be utilized in other areas where the current state-of-the-art is based on centralized distribution. In computer science research specifically, this research is interesting to anyone who wishes their research to be reproducible for others.

Regarding the evaluation of the objectives, the testing tasks produce quantitative measures of the system's performance. One such measure is the highest possible fraction of malicious nodes in the system until the integrity or availability of the system is compromised. These measures can be compared to current methods for distributing .buildinfo files and other, for example, centralized techniques for distributing files.

PRE-STUDY

The literature study will primarily focus on understanding reproducible builds within the Debian project and other distributions. This includes relevant information on why .buildinfo files were introduced, their format and semantics and how they are used today. Earlier research on reproducible builds is in, for example, [6, 2]. That first leg then informs a study on decentralized systems, namely the technologies and algorithms that are in use, their security guarantees and a first estimate on whether they could fulfill the requirements of a system for distributing .buildinfo files. Distributed ledger technologies in general and

blockchain and peer-to-peer networks in particular are to be studied. Some initial sources on this topic are [7, 8, 9, 10, 11]. To find the means for building and testing the actual system, the pre-study will also include working with and automating the use of containers or similar technologies.

CONDITIONS & SCHEDULE

The simulated production environment for testing the system needs a number of virtual or physical machines. Subset supplies the resources for such a setup. Subset's involvement is also in terms of general advice from the external supervisor and to supply knowledge and support in their specialization areas. These areas include IT security and cryptography.

Timeline

Week No.	Task(s)
6	Individual plan hand-in, Pre-study
7	Pre-study
10	Pre-study
11	Pre-study, Implementation
12	Implementation, Pre-study hand in
13	Implementation
14	Implementation
15	Implementation, Testing & analysis
16	Testing & analysis
17	Testing & analysis
18	Additional thesis work time
19	Thesis hand-in, Opposition
20	Thesis re-work, Opposition
21	Thesis re-work, Presentation preparation
22	Presentation
23	Post-presentation corrections, Final hand-in

Glossary

FOSS Free Open Source Software. 1, 2

References

[1] K. Thompson, "Reflections on trusting trust," vol. 27, no. 8, p. 3.

[2] C. Lamb and S. Zacchiroli, "Reproducible builds: Increasing the integrity of software supply chains," pp. 0–0, conference Name: IEEE Software.

[3] Reproducible builds — a set of software development practices that create an independently-verifiable path from source to binary code. [Online]. Available: https://reproducible-builds.org/

[4] Public rebuilderd instances. [Online]. Available: https://rebuilderd.com/

[5] L. Lamport, "Specifying concurrent systems with TLA+," p. 372.

[6] M. Linderud and A. L. Opdahl, "Reproducible builds: Break a log, good things come in trees," p. 120.

[7] E. Daniel and F. Tschorsch, "IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks." [Online]. Available: http://arxiv.org/abs/2102.12737

[8] J. Blähser, T. Göller, and M. Böhmer, "Thine — approach for a fault tolerant distributed packet manager based on hypercore protocol," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1778–1782, ISSN: 0730-3157.

[9] M. N. Ince, M. Ak, and M. Gunay, "Blockchain based distributed package management architecture," in *2020 5th International Conference on Computer Science and Engineering (UBMK)*, pp. 238–242.

[10] J. Liu, B. Li, L. Chen, M. Hou, F. Xiang, and P. Wang, "A data storage method based on blockchain for decentralization DNS," in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 189–196.

[11] N. Zahed Benisi, M. Aminian, and B. Javadi, "Blockchain-based decentralized storage networks: A survey," vol. 162, p. 102656. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1084804520301302