Practical Trade-Offs in Integrity Protection for Binaries via Ethereum

Oliver Stengele
oliver.stengele@kit.edu
Institute of Telematics, KIT
Karlsruhe, Germany

Jan Droll
jan.droll9@kit.edu
Institute of Telematics, KIT
Karlsruhe, Germany

Hannes Hartenstein
hannes.hartenstein@kit.edu
Institute of Telematics, KIT
Karlsruhe, Germany

Abstract

Ensuring the integrity of executable binaries is of vital importance to systems that run and depend on them. Additionally, supply-chain attacks and security related bugs demonstrate that binaries, once deployed, may need to be revoked and replaced with updated versions.

Recently, blockchain ecosystems have garnered broad attention as middlewares for decentralised solutions to existing problems. Stengele et al. [4] presented a concept how the Ethereum blockchain and peer-to-peer network can be used to ensure the integrity of binaries with timely, accurate, and machine-readable revocations. In this work, we show this concept in practice with a user client implementation in Go and demonstrate how revocations and updates can reliably reach a user client within minutes. We show the client's ability to ensure the integrity of multiple binaries and continuously monitor the Ethereum blockchain for updates and revocations via an unmodified Ethereum client. We also examine the trust relations and trade-offs through our use case. Since the user client fully relies on an Ethereum client as a gateway, the latter's resilience against malicious actors is crucial to consider in a practical deployment.

CCS Concepts: • **Security and privacy → Software and application security**; • **Networks → Network services**; • **Computer systems organization** → *Reliability*.

Keywords: Blockchain, integrity protection, revocation

ACM Reference Format:
Oliver Stengele, Jan Droll, and Hannes Hartenstein. 2020. Practical Trade-Offs in Integrity Protection for Binaries via Ethereum. In *21st International Middleware Conference Demos and Posters (Middleware '20 Demos and Posters), December 7–11, 2020, Delft, Netherlands.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3429358.3429374

Middleware '20 Demos and Posters, December 7–11, 2020, Delft, Netherlands

ACM ISBN 978-1-4503-8202-1/20/12…$15.00
https://doi.org/10.1145/3429358.3429374

1 Introduction

Conventional solutions for ensuring the integrity of binaries generally revolve around hash fingerprints or digital signatures. The latter can be attached to and distributed with binaries but revocation can be cumbersome, unreliable [2], and imprecise as a single key can be used to sign multiple binaries. By contrast, hash fingerprints can individually be revoked but they require a reliable and secure infrastructure to be distributed independently of binaries. With blockchains as a middleware, such an infrastructure not only becomes available to anyone but their use can also obviate the need to trust a single hosting provider. In this work, we examine the trade-offs we encountered while realising a concept for publishing and revoking integrity protecting information for binaries via Ethereum that we since named "Palinodia".

Two prominent blockchain-based solutions for binary integrity protection are Contour [1] and Chainiac [3]. Contour is a lightweight construction based on Merkle trees that uses the Bitcoin blockchain as a tamper-resistant and highly available storage and distribution middleware. Chainiac is a comprehensive software creation and publication logging solution that ensures source-to-binary correspondence through independently performed reproducible builds in addition to integrity. Neither Contour nor Chainiac focus on the ability to revoke integrity protecting information for binaries.

2 Palinodia

In previous work [4], a concept was proposed on how the Ethereum blockchain can serve as a middleware to facilitate the publication and revocation of integrity protecting information for software binaries and how the underlying peer-to-peer network can be viewed as a reliable broadcast medium. Smart contracts are used to establish unique and persistent identities for software that can be managed by and transferred between software developers and maintainers. These contracts also store integrity protecting information and verifiably link it to the respective software identity.

While the previous work showed how to publish and revoke integrity protecting information for binaries on the Ethereum blockchain in principle, we now provide a working user client and explore the practical consequences and trade-offs that come with obtaining and using blockchain-based information in this demonstration. We show the "burden" a user bears in such a decentralised architecture. Rather

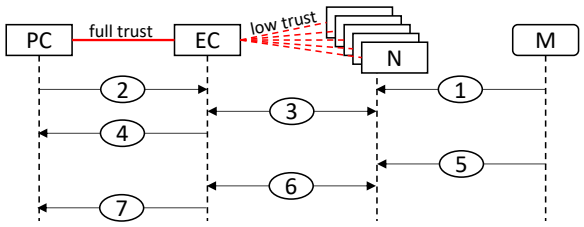

Figure 1. Interactions of Palinodia client (PC), Ethereum client (EC), peer-to-peer nodes (N), and a software maintainer (M). **(1)** M publishes integrity protecting information to a smart contract. **(2)** After obtaining a binary and extracting metadata, PC instructs EC to obtain relevant chain data. **(3)** EC interacts with Ns to obtain and verify chain data. **(4)** EC returns verified chain data to PC. PC determines integrity of the binary. **(5)** M issues revocation via transaction. **(6)** EC obtains revocation from Ns. **(7)** EC forwards revocation to PC. Steps (3) and (6) depend on sync mode of EC.

Table 1. Performance of the Palinodia client and Geth (v1.9.22, default settings) in light and fast/full operating mode on the same device connected to the Ropsten test network (as of Oct 2020). **Rows**: *incoming* network traffic for validating a binary (NT-V) and maintaining synchronisation with the blockchain (NT-M); time spent by respective client during validation (TTV) and revocation (TTR) of a binary. Timing of revocation begins with issuance and includes time for the transaction to be included in a block and propagated to Geth. NT-V, TTV, TTR averaged over 100 binaries.

Client	Palinodia	Geth light	Geth fast/full
RAM	40 MB	275 MB	1.4 GB
Disk	70 MB	500 MB	90 GB
NT-V	0 B	1.5 MB	0 B
NT-M	0 B	12 MB/day	450 MB/day
TTV	4 ms	400 ms	13 ms
TTR	1 ms	20 s	20 s

than implementing the functionality to interact with the Ethereum network, we opted to use a preexisting client with Go Ethereum (Geth) as a gateway as depicted in Figure 1. This choice already opens up various deployment scenarios as the Palinodia and Ethereum clients do not necessarily have to run on the same device and one Ethereum client can serve multiple applications and devices.

Ethereum clients like Geth have several methods for obtaining and maintaining the current state of the blockchain, ranging from fast and lightweight methods to a full synchronisation that can take several days and consume gigabytes of persistent storage. Table 1 shows the resource consumption and performance of the Palinodia client and Geth while publishing and revoking integrity protecting information via the Ropsten test net. Generally, we find the Palinodia client to be even more lightweight than Geth in light mode and it is not surprising to see that, once Geth has obtained and verified the necessary blockchain data, the Palinodia client can perform its function quickly and efficiently.

Each synchronisation mode presents a trade-off between effort and trust. The more effort the client exerts in verifying the data obtained from the Ethereum network, the less chances malicious nodes have to feed it false information, and vice versa. With the full synchronisation mode, Geth obtains and validates all blockchain state data and must therefore not query any other nodes for data pertaining to validation or revocation of binaries. However, with the significant requirements regarding hardware and bandwidth, operating such a full Geth node is quite challenging for an average user. Running Geth in light mode is far more feasible, but since it relies on multiple full nodes to obtain and partially verify state data on demand, it leaks to them some information regarding the software a user has installed. Additionally, light clients can be tricked into accepting invalid information with substantial effort by a malicious actor while a full node is immune to such an attack. Users of Palinodia must strike a balance between their hardware capabilities and their need to defend themselves against malicious actors. We evaluated these multi-faceted options empirically and present the results interactively.

In summary, blockchains as a public middleware for recording and disseminating information present a new paradigm with unique goals and challenges in practical applications in order to preserve and benefit from their decentralised structure. This demonstration presents quantitative results that a) show that the Palinodia client is relatively lightweight compared to the Ethereum client, and b) make the trade-offs between trust and performance explicit and, thus, enable some form of risk management.

Acknowledgments

This work was supported by the German Federal Ministry of Education and Research within the framework of the project KASTEL_ISE in the Competence Centre for Applied Security Technology (KASTEL).

References

[1] M. Al-Bassam and S. Meiklejohn. 2018. Contour: A practical system for binary transparency. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, Cham, 94–110.

[2] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson. 2015. An end-to-end measurement of certificate revocation in the web's PKI. In *Proceedings of the 2015 Internet Measurement Conference*. Association for Computing Machinery, New York, NY, USA, 183–196.

[3] K. Nikitin, E. Kokoris-Kogias, P. Jovanovic, N. Gailly, L. Gasser, I. Khoffi, J. Cappos, and B. Ford. 2017. CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1271–1287.

[4] O. Stengele, A. Baumeister, P. Birnstill, and H. Hartenstein. 2019. Access Control for Binary Integrity Protection using Ethereum. In *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*. Association for Computing Machinery, New York, NY, USA, 3–12.