‘Blockhub’: Blockchain-based Software Development System for Untrusted Environments

Denis Ulybyshev, Miguel Villarreal-Vasquez*, Bharat Bhargava, Ganapathy Mani*
Computer Science Department, CERIAS
Purdue University
West Lafayette, United States of America
dulybysh, mvillar, bbshail, manig@purdue.edu

Steve Seaberg, Paul Conoval, Robert Pike, Jason Kobes
Northrop Grumman
McLean, United States of America
Steve.Seaberg, Paul.Conoval, Robert.Pike, Jason.Kobes@ngc.com

***Abstract*— To ensure *integrity, trust, immutability* and *authenticity* of software and information (cyber data, user data and attack event data) in a collaborative environment, research is needed for cross-domain data communication, global software collaboration, sharing, access auditing and accountability. Blockchain technology can significantly automate the software export auditing and tracking processes. It allows to track and control what data or software components are shared between entities across multiple security domains. Our blockchain-based solution relies on role-based and attribute-based access control and prevents unauthorized data accesses. It guarantees integrity of provenance data on who updated what software module and when. Furthermore, our solution detects data leakages, made behind the scene by authorized blockchain network participants, to unauthorized entities. Our approach is used for data forensics/provenance, when the identity of those entities who have accessed/ updated/ transferred the sensitive cyber data or sensitive software is determined. All the transactions in the global collaborative software development environment are recorded in the blockchain public ledger and can be verified any time in the future. Transactions can not be repudiated by invokers. We also propose modified transaction validation procedure to improve performance and to protect permissioned IBM Hyperledger-based blockchains from DoS attacks, caused by bursts of invalid transactions.**

Keywords—blockchain; access control; privacy; data provenance; collaborative software development

I. INTRODUCTION

Blockchain technologies can be categorized in two groups based on their architecture and controls applied to the participants. Permissionless blockchains are the networks in which any node can act as a verifier of the network without previous authorization (e.g. Bitcoin, Ethereum). On the other hand, permissioned blockchains are networks that require authorization from a centralized authority or consortium, which imposes identity management and role-based access control (e.g. Hyperledger). Blockchain technologies can also be categrozied as public or private blockchains depending on whether access control is applied to the network. Whether a blockchain is permissionless or permissioned, there are several key technical challenges that limit the powerful potential of blockchain technology. One of the challenges that we address in this paper is data privacy. Multiple untrusted entities can access, update or transfer software modules in collaborative software development environment. We proposed a ‘WAXEDPRUNE’ mechanism in [1], that extends methodology of a privacy-preserving data exchange between services in SOA, when each service can access only those data items the service is authorized for [2], [3], [4]. Data exchange model uses role-based and attribute-based access control.

In this paper, we aim to combine ‘WAXEDPRUNE’ framework with blockchain-based technology that guarantees integrity of provenance data. This novel application of blockchain technology allows tracking and controlling what data or software modules are shared between entities across multiple security domains. It is used for effective data forensics/provenance in cross-domain data communication networks and in global software collaboration environments with multiple untrusted writers. In our permissioned blockchain network, the attribute-based access control is used, in addition to role-based access control. Client attributes, that are evaluated by access control kernel, include cryptographic capabilities of a web browser, authentication method, type of the client’s device, context and trust level, which is constantly re-calculated [1]. Furthermore, our blockchain application inherits the data leakage detection feature [12] from ‘WAXEDPRUNE’ [1]. It allows to detect several types of data leakages, made by authorized blockchain network participants behind the scene to unauthorized entities outside of blockchain network.

Another challenge that we address in this paper is performance of blockchain-based technologies. We aim to improve the performance of IBM Hyperledger Fabric platform, which is a permissioned blockchain. Modified transaction validation procedure involves less amount of communications and drops invalid transactions at earlier phase. This is also used to protect Hyperledger network from DoS attacks, caused by bursts of invalid transactions.

II. RELATED WORK

Compared to regular centralized databases, blockchain provides more trust, robustness and fault tolerance due to its immutability [5]. Blockchain provides disintermediation i.e., instead of relying on a central administrator, the transactions can be verified by the collaborators that can view the digital ledger and provide consensus. However, recent security

*Both authors contributed equally and are considered co-first authors

Approved For Public Release #18-1247

breaches show that blockchain technology is prone to hacking. The Krypton blockchain, Ethereum-based blockchains, coded in golang, is hacked by "51 crew" [16]. They were able to replicate and manufacture their own blockchain, and push that into production as a real Krypton chain with large amounts of hashing power and DDoS attack on Krypton chain nodes. Thus, in some cases, it is not necessary to use blockchain [17].

Microsoft has announced that in collaboration with Blockstack Labs, ConsenSys and developers from all over the world, it is working on an open-source, blockchain-based identity system that allows applications and services to interoperate across blockchains, clouds and organizations [18].

Recent study [9] demonstrates that trust and immutability are provided through provenance on blockchain technology, where smart contracts can be created. This increases trust and reduces the need for a third party intervention in decentralized systems. Our approach utilizes data provenance with blockchain technology.

III. Core Design

A. Blockchain-based cross-domain data and software sharing

We use blockchain technology for secure cross-domain software development, global software collaboration, sharing and audit. Blockchain allows tracking and controlling what data items and software components are shared between what entities across multiple security domains. It guarantees that transactions cannot be repudiated by invokers. Recording provenance and the identity determination for those entities who have accessed/ updated/ transferred the sensitive software modules are provided. Blockchain guarantees integrity of provenance records that are used for forensics in case of detected data/software leakages. Distributed data storage mechanism using blockchains with smart contracts provides access auditing, tracking capabilities and software leakage prevention. This mechanism supports tamper-resistant software supply chain that allows to develop and exchange software components across multiple entities, e.g. across multiple departments of a company.

The core idea is that any two untrusted collaborators, e.g. X and Y, can securely share software in a controlled manner via smart contracts running in the blockchain network that regulates the access to software modules (see Fig.1). Every access to the software module is granted or rejected based on the policies established in these smart contracts and every successful request and transfer of software module is logged as a record in the blockchain distributed ledger. Software modules are stored in encrypted form as a Software Bundle (SB), which is a self-protected structure that incorporates encrypted software modules, access control policies and policy enforcement engine (see Fig.2). Thus, in addition to smart contracts, running in the blockchain network, there are access control policies, embedded into SBs that control the access to software modules. SB mechanism is built on top of Active Bundle [2], [3] technology for secure data transfer between untrusted entities. For each software request, the following steps are executed:

(a) Registration of software attribute and ID based on software location: Collaborators in the software distribution network provide SB identifier (ID), linked to the software storage location, to the blockchain distributed ledger. Software modules are stored in SBs in encrypted form. Collaborator is

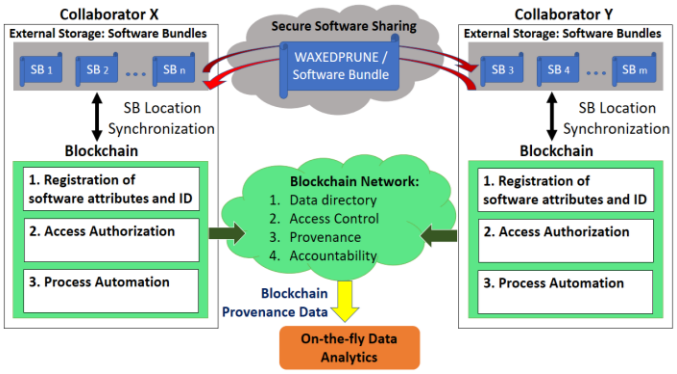

Figure 1. Blockchain-based software management system

able to find required software module in the distributed storage using the stored ID information.

(b) Smart Contract management: Each collaborator is able to manage access control policies in the smart contract for its own software module when the ID of SB is registered in the blockchain network. Software owners decide who can access what particular software module via the distributed ledger using policies in smart contracts.

(c) Secure software distribution through 'WAXEDPRUNE' solution: collaborators are not allowed to download software directly from the repository. They are required to pass authentication, access control policy and attributes evaluation. To provide confidentiality, integrity, leakage detection, role-based and attribute-based access control, software modules are stored in a non-relational database in SB in the form of key-value pairs with encrypted values. Here is the example of key-value pair:

*{"ab.Module1Code" : "***Enc***(#include<stdio.h>; int main() {int var1=1; ... })" }*

Value for a given software module, e.g. source code in C language, is '*#include<stdio.h>; int main() {int var1=1; ... }*' and it is stored in encrypted form. Each separate software module is encrypted with a separate AES symmetric key, which is generated on-the-fly based on execution flow, depending on SB modules, authentication code and resources: authentication certificate and access control policies. Details are given in [2]. When collaborator X requests a software module SMi from an SB, hosted by another collaborator Y (see Fig.1), the identity of X is verified. In the authentication phase [2], [3], X presents its X.509 certificate signed by a trusted Certificate Authority (CA) to the SB, hosted by Y. If authentication passes, then X's attributes, including browser's cryptographic capabilities, are evaluated and enforced. Cryptographic capabilities assume existence and support of certain cryptographic libraries in the collaborator's browser [10], [11]. If the check of attributes for X passes then evaluation of applicable access control policies starts. It is determined what data can be disclosed to X. Based on that evaluation, decryption keys are derived to decrypt accessible values from key–value pairs. SB has a built-in tamper-resistance mechanism, based on the digest of the SB modules and their resources. Furthermore, our blockchain-based solution inherits the data leakage detection feature [12] from 'WAXEDPRUNE' framework.

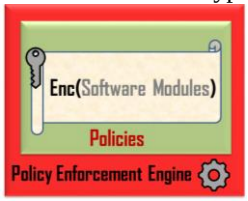

Figure 2. Software Bundle

It supports data leakage detection/prevention for multiple types of data leakages, made by authorized blockchain network insiders behind the scene to unauthorized entities outside of blockchain network. The solution relies on data protection, provided by SB. To address plaintext leakages, we rely on digital/visual watermarks, as well as on partial data disclosure. Details of our data leakage detection solution are available in [12]. SB is written in Java and implemented as a JAR-file. Prototype demo video [7] is available.

(d) Automated Process: Blockchain contains smart contracts that are triggered when a transaction is invoked. These smart contracts automatically run a series of jobs that will complete the software sharing process, from verifying the access authorization to starting software transfer in case when authorization has been granted by both smart contract and policy enforcement engine of SB. Every request for software that is authorized and executed, is recorded in the blockchain public ledger. Thus, integrity of provenance data is guaranteed.

Adversary Model: Blockchain network participant can be malicious in terms of: (a) corrupting transactions log records (provenance data) in order to hide and/or repudiate transactions; (b) leaking data, for which participant is authorized, to unauthorized entities outside of blockchain network.

Assumption: entity that hosts/executes SB has trusted hardware, trusted operating system and trusted Java Virtual Machine.

B. Protection against DoS attacks in permissioned blockchains

In permissioned blockchain [13], there are three major nodes (see Fig. 3): (1) **Client service** invokes a transaction on behalf of a user, submits it to the transaction endorsers and broadcasts the transaction to the sorting module; (2) **Committing peers** commit the transactions and keep the copy of a blockchain. Some of these peers can have a special role of Endorsers. Before the commitment of a transaction, an endorsing peer checks the validity of the transaction; (3) **Sorting module** orders transactions in a chronological order and acts as a communication channel between client services and communicating/endorsing peers. This channel outputs the same message to all connected peers in the same chronological order. These communicated messages are nothing but transactions to be included in the blockchain. For efficiency, the blocks are created with a batch of transactions and sorting module imposes a deterministic ordering of transactions in each block.

Clients can create a dense set of invalid transactions to make network busy. These bursts of transactions can create a bottleneck in Endorsers thus stopping the transmission of valid transactions from other clients. In the baseline (old) workflow the client service creates a transaction and broadcasts it to Tx Endorsers of its choice. The Endorsers simulate the invoked transaction, check if it is valid and if it adheres to the endorsement policies. After the validation procedure, each Endorser produces an endorsement signature. Client receives the endorsed transactions with Read/Write set and sends it to the Sorting Module. The Sorting Module delivers the transaction to Tx Committers. Tx Committers validate the transactions read set again with database before committing. The block is added to the blockchain with transaction marked as valid or invalid based on the validation. We propose the new workflow which eliminates client communication with the

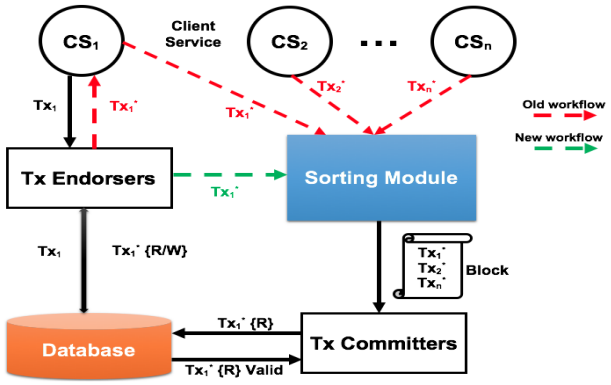

Figure 3. Permissioned Hyperledger-based blockchain platform architecture

Sorting Module. The Tx Endorsers will directly send the endorsed transactions to the Sorting Module. In this way, we create a proper order when multiple clients create transactions at the same time. This order avoids confusion and excessive use of resources even if there was a burst of transactions. The sorting module knows the list of Tx Endorsers and waits for all of them to confirm the transaction. Here we use the majority endorsement principle. This solution is scalable, since Tx Endorsers and Sorting Module process clients' invoked transaction requests in parallel. Only transactions that fulfill the endorsement policies are passed directly to the Sorting Module (Ordering Service). Invalid transactions are discarded.

IV. Evaluation

We evaluated performance overhead of our 'Blockhub' prototype that combines IBM Hyperledger Fabric platform [13] and 'WAXEDPRUNE' framework [1]. In our experiment, there are three web services in IBM Hyperledger network. In the first experiment, they host software modules in the form of SBs. In the second experiment, they have a pointer to a google cloud instance that hosts SBs. Services are deployed on top of IBM Hyperledger network in NodeJS framework, provided by "Marbles" open-source project [6]. Service 1 sends a request for software module 1 to Service 2. Firstly, this data request is registered in the blockchain network and smart contracts are triggered (see Fig.1). Then the request is redirected to corresponding SB, which contains the requested software module. The overall transaction latency includes blockchain transaction processing time and Round-Trip Time (RTT) for data request, processed by SB. Data request RTT for SB is measured between the moment when data request processing is finished by blockchain network and the moment when data (software source code), retrieved from SB, are received by the client (Service 2). Details of the 'WAXEDPRUNE' installation are covered in [8]. In addition, we measure chaincode validation time, which is part of blockchain transaction processing time. We ran the data request 50 times and compute average values of chaincode validation time, blockchain transaction latency and RTT for SB.

Experimental setup 1
SB hosting Hardware: Intel Core i7, CPU 860 @2.8GHz x8, 8GB DRAM
OS: Linux Ubuntu 14.04.5, kernel 3.13.0-107-generic, 64 bit

Experimental setup 2: Google cloud instance with the following characteristics:

SB hosting Hardware: Intel(R) Xeon(R) CPU@ 2.30GHz
OS: Linux Debian 4.9.65-3+deb9u2 (2018-01-04) x86_64, kernel 4.9.0-5-amd64, 64 bit

Blockchain platform *IBM Hyperledger Fabric ver. 1.0.x* is used in both experiments.

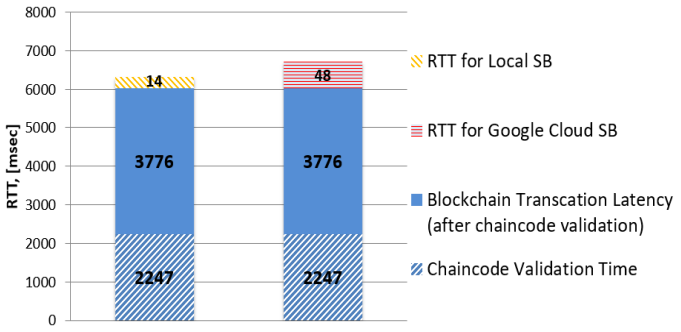

Figure 4. Transaction latency for local / Google cloud Software Bundle

As we see in Fig.4, blockchain transaction latency is way greater than SB RTT. Overall blockchain transaction latency, including chaincode validation time, is 6.02 [sec]. Benefits, provided by SB, such as attribute-based access control and leakage detection, impose only 0.23% overhead when SB is hosted by the service in the blockchain network, and 0.8% overhead when hosted by Google cloud instance. IBM announced that new Hyperledger Fabric version 1.1 [14] will have way better performance. Microsoft Coco blockchain platform [15] provides transaction latency 0.152 sec, but it is not open-sourced yet. Once it is open-sourced, we will be able to integrate it in our framework and improve its performance.

V. Conclusion

We presented a comprehensive solution for collaborative software development that allows to track and control what software components are shared between entities across multiple security domains. The solution combines 'WAXEDPRUNE' [1] framework with permissioned blockchain, IBM Hyperledger Fabric [13]. 'WAXEDPRUNE' provides flexible attribute-based access control and capabilities of detecting leakages, made by authorized blockchain network participants (malicious insiders) to unauthorized entities outside of blockchain network. The overhead, imposed by 'WAXEDPRUNE', does not exceed 0.8%.

We also proposed a modified transaction validation procedure in order to improve performance and to protect permissioned IBM Hyperledger-based networks from DoS attacks, caused by bursts of invalid transactions.

Future Work

We plan to finalize integration of our 'WAXEDPRUNE' framework [1] with IBM Hyperledger Fabric and perform more experiments to test 'Blockhub' prototype. 'Blockhub' will be compared with the baseline solution, which will be 'WAXEDPRUNE' [1] for secure data and software exchange extended with capabilities of blockchain-based recording of provenance data for all the transactions, blockchain replication and with verifiability of every transaction any time in the future. In addition, modified transaction validation procedure for IBM Hyperledger Fabric will be tested.

Approved For Public Release #18-1247

Acknowledgment

This work was partially supported by a Corporate Partners Award of Purdue Computer Science department. This work was also partially supported by the Northrop Grumman Cybersecurity Research Consortium. We thank Prof. Leszek Lilien (Purdue University, Western Michigan University) and Aala Oqab-Alsalem for their help and valuable feedback.

References

[1] D. Ulybyshev, B. Bhargava, M. Villarreal-Vasquez, D. Steiner, L. Li, J. Kobes, H. Halpin, R. Ranchal, A. Oqab-Alsalem, "Privacy-Preserving Data Dissemination in Untrusted Cloud", IEEE CLOUD 2017, pp. 770-773.

[2] R. Ranchal, "Cross-domain data dissemination and policy enforcement," PhD thesis, Purdue University, 2015.

[3] L. B. Othmane, "Active bundles for protecting confidentiality of sensitive data throughout their lifecycle," PhD thesis, Western Michigan University, 2010.

[4] R. Ranchal, B. Bhargava, P. Angin, and L. B. Othmane, "Epics: A framework for enforcing security policies in composite web services," *IEEE Transactions on Services Computing*, 2018.

[5] "Blockchains vs centralized databases", March 17, 2016. Available: https://www.multichain.com/blog/2016/03/blockchains-vs-centralized-databases/, accessed: May 2018

[6] "Marbles Demo," Available: https://github.com/IBM-Blockchain/marbles, accessed: May 2018

[7] D. Ulybyshev and B. Bhargava, "Secure dissemination of EHR demo video,"Available:https://www.dropbox.com/s/4wg3vuv52j4s16v/NGCRC-2017-Bhargava-Demo1.wmv?dl=0, accessed: May 2018

[8] D. Ulybyshev, B. Bhargava, L. Li, J. Kobes, D. Steiner, H. Halpin, B.An, M.Villarreal, R.Ranchal, T.Vincent, "Secure dissemination of EHR in untrusted cloud," Tutorial, Purdue University, 2016.

[9] G. Gordon, "Provenance and authentication of oracle sensor data with block chain lightweight wireless network authentication scheme for constrained oracle sensors," 2017

[10] "W3C web cryptography API," 2018. Available: https://www.w3.org/TR/WebCryptoAPI/, accessed: May 2018

[11] "Web authentication: an API for accessing scoped credentials," Available: http://www.w3.org/TR/webauthn/, accessed: May 2018

[12] D. Ulybyshev, B. Bhargava, A. Alsalem, "Secure Data Exchange and Data Leakage Detection in Untrusted Cloud," ICACCT 2018, in press

[13] "IBM Blockchain 101: Quick-start guide for developers," 2016. Available: https://www.ibm.com/developerworks/cloud/library/cl-ibm-blockchain-101-quick-start-guide-for-developers-bluemix-trs/index.html, accessed: May 2018

[14] "About Fabric performance testing report," 2017. Available: https://lists.hyperledger.org/pipermail/hyperledger-fabric/2017-October/001895.html, accessed: May 2018

[15] "Announcing the Coco Framework for enterprise blockchain networks," 2017. Available: https://azure.microsoft.com/en-us/blog/announcing-microsoft-s-coco-framework-for-enterprise-blockchain-networks/, accessed: May 2018

[16] "Krypton Abandons Ethereum-Based Blockchain After 51% Attack", May 31, 2017. Available: http://krypton.rocks/blog/krypton-abandons-ethereum-based-blockchain-after-51-attack/, accessed: May 2018

[17] G. Greenspan, "Avoiding the Pointless Blockchain Project", 11/22/2015, Available: http://www.multichain.com/blog/2015/ 11/avoiding-pointless-blockchain-project/, accessed: May 2018

[18] "Microsoft building Open Blockchain-based Identity System With Blockstack, ConsenSys. Available: https://bitcoinmagazine.com/articles/microsoft-building-open-blockchain-based-identity-system-with-blockstack-consensys-1464968713/, accessed: May 2018