Detailed description of the vulnerabilities

Ranwa Al Mallah

LiTrans, Ryerson University, Ontario, Canada

David López

GiiTraL, Universidad Nacional Autónoma de México, Mexico City, Mexico

Bilal Farooq

LiTrans, Ryerson University, Ontario, Canada

1. V_1 - Improper key protection mechanisms

Permissioned blockchains need to follow the cryptographic standards that apply to public blockchains to secure transaction messages and ensure transaction authenticity. They also might need to encrypt data to ensure maximum security. Thus, the cryptographic keys required for these operations should be stored and maintained properly. A private key that is lost cannot be recovered. Also, if stolen, the user's blockchain account will face the risk of being tampered. Since the blockchain is not dependent on any centralized third-party, if the user's private key is stolen, it is difficult to track the attacker's behaviors and recover the modified blockchain information.

Vulnerabilities may exist in the different key protection schemes. Different categories of wallets to store private keys have emerged, self-sovereign wallets, hosted wallets, etc. Users of self-sovereign wallets locally store their private keys in a file on the computer. If the file is lost, stolen or corrupted by a malware, the key cannot be recovered. A basic wallet and a password protection ensures a two-factor authentication. If the file is stolen, the attacker will not know the password. However, if the password is forgotten, it cannot be recuperated and the account either.

Hosted wallets require a centralized party to provide an interface for interaction with the wallet and thus blockchain Web-hosted wallets is a web service that holds the signing key for the individual. The advantage is that it can be accessed from anywhere and also that the password can be reset. However, this requires trust in the web service.

2. V_2 - Unauthenticated data feed

Nodes of the blockchain might provide data reflecting the state of the world beyond the blockchain. They act as cyber physical systems where they sense data from the environment and communicate it across a network. On the other hand, companies, transportation agencies and government, receive the data and

Email addresses: `ranwa.almallah@ryerson.ca` (Ranwa Al Mallah), `dlopezfl@iingen.unam.mx` (David López), `bilal.farooq@ryerson.ca` (Bilal Farooq)

might use it in a retroaction loop to send back commands into the environment for some mobility application. In this context, data of the nodes serve as data feeds and must be authenticated because some nodes may misbehave and inject wrong data, resulting in corrupted outcomes of the mobility applications. Even providers that offer authenticated data feeds using trusted hardware, a vulnerability in trusted hardware may also result in a compromise of the entire data feed.

3. **V_3 - Unsecure cryptographic signature algorithm**

Bitcoin and *Ethereum* blockchains use the Elliptic Curve Digital Signature Algorithm (ECDSA). Vedral and Morikoshi (2007) argue that quantum computers can crack ECDSA by recovering the user's private key. This makes a user vulnerable to identity theft attacks because when using blockchain, the user's private key is regarded as the identity and security credential. On another hand, mathematical complexity does not necessarily guarantee the security of a cryptographic algorithm when it is implemented in a real-world situation. Implementation can dramatically lower the security level.

4. **V_4 - Privacy Threats to User Identity**

Transactions are digitally signed by private keys of users enabling anybody to verify the validity of transactions by corresponding public keys. However, such an approach provides only pseudonymous identities that can be traced to real identities. Various means should be used for obfuscation of user identities to provide unlinkability to users.

5. **V_5 - Weak privacy protection measures**

Users of the BSMD need to disclose their location to receive Location Based Services (LBS). Since the users' behaviors in the blockchain are traceable, the blockchain systems must take measures to protect the transaction privacy of the users. BSMD implements techniques which consist of hiding the real location of the user from the LBS provider. It is possible to still access LBS without disclosing the user's exact location using either K-anonymity or a Differential privacy called Geo-Indistinguishability (GeoInd). However, for maintaining a high level of privacy in GeoInd, the utility must be sacrificed. Thus, the information retrieved from LBS using GeoInd may not be useful unless users sacrifice some degree of privacy. In BSMD, the hybrid approach leaves the user to select the level of privacy. In situations where the exact location is important for getting accurate response, GeoInd is used. LBS provider can reward users for their information if more relaxed parameters for privacy are set. Unfortunately, these privacy protection measures are not very robust and may lead to privacy leakage of its sender.

6. **V_6 - Lack of monitoring of smart contract applications**

Smart contracts can be developed and deployed by pseudonymous malicious people (only public addresses or public keys are known to others in the blockchain). Therefore, a vulnerable smart contract is hard to patch and can easily become out of control once deployed. Because a smart contract is hard to patch for bugs, how to monitor anomalous activities in a smart contract after its deployment and apply appropriate countermeasures should be considered in advance.

Also, smart contract security depends on codified business logic. *Hyperledger Fabric* needs codified business logic in Golang or JavaScript, which are then wrapped around the nodes responsible for this sort of logic with Docker. Poor access management on *smart contracts* or code vulnerabilities, like call stack, stack size limit, reentrancy, malicious libraries, type casts, and open sensitive information, might deviate *smart contracts* intended behaviour to malicious transactions or even unintended take over for further compromise.

7. V_7 - Program design of smart contract

Particularly, as programs running in the blockchain, *smart contracts* may have security vulnerabilities caused by program defects. Atzei et al. (2017) conduct a systematic investigation of the security vulnerabilities of *Ethereum smart contracts* at the level of source code.

On the other hand, BSMD nodes were implemented on *Hyperledger* Indy which is a public-closed blockchain. *Hyperledger* utilizes general-purpose programming languages, e.g., Go, Node.js, and Java, to implement *smart contracts* (called chaincode in *Hyperledger* Fabric). One of the disadvantages is that these languages were not originally designed for writing *smart contracts*. In Yamashita et al. (2019), they surveyed the vulnerabilities associated with chaincodes developed using Go language. They observed there are 14 potential risks.

For example, they highlight the system timestamp vulnerability. It's difficult to ensure timestamp functions are executed at the same time for each peer. Also, regarding the global variables vulnerability, global variables are only global to a single node. If such a node breaks down, global state variables might no longer be consistent over all peers.Finally, Goroutines generate concurrency which leads to non-deterministic behavior in Go Kaiser (2018).

One of the more notable features of the Go programming languages is the ability to import packages from a remote repository, identified by a URL. Fabric encourages the use of this feature in chaincode for loading the packages and other support code. According to Dawson (2017), this represents a latent security issue, it is one which has not resulted in an exploitable vulnerability, but which could potentially be the cause of one if circumstances were to change. The authors present specific security flaws in Node.js. In fact, if a specific function is called, one of its modules will crash or throw an exception depending on the version and thus attackers may exploit this vulnerability,

8. V_8 - Program writing of *smart contracts*

Some *smart contracts* are not adequately written due to misunderstandings of common practices. In Huang et al. (2019), they identified two vulnerabilities named *unhandeled errors* and *unchecked input arguments*. Unhandeled errors might lead to faulty execution. The input arguments should always be checked to avoid accessing a nonexistent element.

9. V_9 - Design flaws of the blockchain platform

Fabric chaincode runs in a secured Docker container that is isolated from the endorsing peer process. However, Shaw (2017) present a main vulnerability in design in that chaincode is accessible on the network and can easily download and install other software packages and security tools. Moreover, it can run for long periods. The use of Docker greatly constrains what the chaincode can do. However, the chaincode still has sufficient freedom, given that it can install arbitrary software within the container, including security tools such as Nmap. Also, it can perform port scans against public or private networks that are visible to the node, it can exploit any vulnerable hosts that are discovered and accept commands from, and exfiltrate results to, a remote command-and-control server and continue executing for a long period.

On the other hand, Huang et al. (2019) identified two vulnerabilities in Fabric chaincodes arising from platform features: Range query risk and read your write. Some range query methods, such as *GetQueryResult*, are not re-executed during the validation phase. For a write statement to take effect, a transaction first has to be committed. Therefore, when reading a value that has already been written during the same transaction, its old value will be retrieved from the ledger.

Finally, containers are more numerous than virtual machines, this implies that a lot of software entities are competing for host resources. If resources are not properly configured, because all containers share kernel resources, one container can monopolize access to certain resources and starve out other containers on the host.

10. V_{10} - DNS rebinding

DNS rebinding is a method of manipulating resolution of domain names. In theory, client-side scripts are only allowed to access content on the same host that served the script, and the same-origin policy prevents anything else from happening. DNS rebinding circumvents this protection by abusing the DNS.

11. V_{11} - Absence of BGP security extensions

BGP (Border Gateway Protocol) is a de-facto routing protocol and regulates how IP packets are forwarded to their destination. Since BGP security extensions are not widely deployed, vulnerabilities in the protocol may expose network operators to Internet prefixes hijacking (Apostolaki et al., 2017).

12. V_{12} - Lack of node monitoring and prevention techniques

Virtual Private Network (VPN) connectivity is required to communicate between private networks spread over different geographical locations. While VPNs are in general secure, lack of prevention techniques that minimize trust and maximize trustworthiness may pose a serious security threat.

13. V_{13} - Poor architecture design

Most blockchain systems are forkable in that they require participants to agree on a chain out of multiple possible branches of blocks. Private ledger network may get split and replicated into parallel fork chains creating ambiguity among child blocks for parallel parent fork chains. This replication may be susceptible to several attacks in parallel fork chains. Non-forkable blockchain design is required to protect against these attacks.

Hyperledger is a private blockchain implementation that has no fork which is expected because its consensus protocol is proven to guaranteed safety. In particular, security means that the non-Byzantine nodes have the same blockchain data. Classic Byzantine tolerant protocols such as PBFT are proven to ensure safety for a certain number of failures to guarantee security. However, violation of the safety property leads to forks in the blockchain.

14. V_{14} - Tolerated power of adversary

In PoW-based blockchains, if a single miner's hashing power accounts for more than 50% of the total hashing power of the entire blockchain, then the 51% attack may be launched. In PoS-based blockchains, 51% attack may occur if the number of coins owned by a single miner is more than 50% of the total blockchain. As BSMD is a BFT-based consensus algorithm, the amount of control an attacker would need is 33.3%. An attacker can exploit this vulnerability to manipulate and modify the blockchain information.

15. V_{15} - Lack of peer privacy

BSMD is implemented on *Hyperledger* Indy, which is a public-closed blockchain for decentralized digital identities. Permissioned blockchains control who participates in validation and in the protocol; these nodes typically have established identities and form a consortium. Validation of transactions occurs through the replicated execution of the chaincode and given the fault assumption underlying BFT consensus. Thus, the security is guaranteed as long as trusted nodes participate in consensus. Membership among the validating nodes running BFT consensus is static and the setup requires manual intervention. Because of lack of peer privacy and constant committee members, the identity of an endorser is known to all members within a channel. Precisely, in a channel, the identity of each peer is known to every other peer. This opens a gateway for attacks on endorsers.

16. V_{16} - Consensus flaws

The consensus protocol used in the implementation of the BSMD is RBFT (Redundant BFT) because Byzantine fault tolerance protocols are not efficient when a fault is encountered. In fact, the current BFTs rely on a special replica called as primary to order the requests. If this primary is a malicious node then it can delay the requests of the transactions. RBFT is a new approach for BFTs inspired by plentum BFT. It runs multiple instances of plentum protocol to ensure that the system is fault tolerant. By doing so, a malicious node may be detected.

17. V_{17} - Asynchronous delivery of messages

Many BFT protocols assume synchronous delivery of messages. Such protocols rely critically on network timing assumptions, and only guarantee liveness when the network behaves as expected. However, network links can be unreliable, network speeds change rapidly, and network delays may even be adversarially induced. This vulnerability motivates asynchronous BFT protocols that can be based on threshold-based cryptography, which enables reliable and consistent broadcast.

18. V_{18} - Unpredictable state

Luu et al. (2016) discovered the unpredictable state vulnerability where the state of the contract is changed before invoking. As the order of transactions' execution depends entirely on miners, Transaction-Ordering Dependent contracts are vulnerable. They also found the randomness bug where the seed is biased by a malicious miner.

19. V_{19} - Timestamping

Usually, besides system time, nodes in PoW and PoS maintain network time that is computed as the median value of the time obtained from the peers. Such a time is often put into the block header, while nodes, upon receiving a block, validate whether it fits freshness constraints. This architecture is vulnerable to de-synchronization attacks. The timestamp of a block can be changed by a malicious miner. If an attacker can modify it, timestamp-dependent contracts are vulnerable.

20. V_{20} - Lack of intrusion prevention/detection mechanisms

Malicious activities should be detected before consensus can be reached. Nodes must monitor, filter blockchain content and/or restrain malicious activity which can degrade acceptance and trust in the blockchain ecosystem. Identifying transactions as malicious is challenging as transactions themselves carry little information about other context.

For example, Criminal Smart Contracts (CSCs) are self-incriminating objects. For a CSCs to be effective, their deployers must advertise, drawing attention to the nature of their contracts. Thus, a sufficiently aware blockchain network can detect the presence of many CSCs and will be incentivized to filter or purge associated transactions. One prevention mechanism is by omitting transactions from blocks when they are flagged by reputable communities as CSCs.

21. V_{21} - Non-deterministic transactions

When executed on top of PBFT consensus, it is important that chaincode transactions are deterministic, otherwise the state of the peers might diverge. A solution to filter out non-deterministic transactions that are demonstrably diverging should be implemented.

22. V_{22} - Absence of incentive and rewards

Incentives are economic rewards for the participants of the network. On one hand, incentives for hosting the network and participating in consensus mechanisms motivate the nodes to maintain the network and secure the transactions. Other rewards are given for sharing information. In BSMD, users own their information so they decide whether to sell it or not. However, the absence of incentive or rewards in the design of the blockchain system may be exploited by malicious nodes to conduct many attacks. For example, if no incentive is put in place, operations can be executed in quantity in one transaction causing nodes of the network to consume a lot of computing power, waste hard disk resources and corrupt the blockchain system.

References

Apostolaki, M., Zohar, A., Vanbever, L., 2017. Hijacking bitcoin: Routing attacks on cryptocurrencies, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE. pp. 375–392. doi:10.1109/sp.2017.29.

Atzei, N., Bartoletti, M., Cimoli, T., 2017. A survey of attacks on ethereum smart contracts sok, in: Proceedings of the 6th International Conference on Principles of Security and Trust - Volume 10204, Springer-Verlag, Berlin, Heidelberg. p. 164–186. doi:10.1007/978-3-662-54455-6_8.

Dawson, M., 2017. Dos security vulnerability. URL: https://nodejs.org/en/blog/vulnerability/oct-2017-dos/. [Online; accessed 20-February-2020].

Huang, Y., Bian, Y., Li, R., Zhao, J.L., Shi, P., 2019. Smart contract security: A software lifecycle perspective. IEEE Access 7, 150184–150202. doi:10.1109/access.2019.2946988.

Kaiser, T., 2018. Chaincode scanner: Automated security analysis of chaincode. URL: https://static.sched.com/hosted_files/hgf18/18/GlobalForum-tobias-kaiser.pdf. [Online; accessed 15-March-2020].

Luu, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A., 2016. Making smart contracts smarter, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery, New York, NY, USA. p. 254–269. doi:10.1145/2976749.2978309.

Shaw, G., 2017. Security assessment technical report. URL: https://wiki.hyperledger.org/download/attachments/2393550/technical_report_linux_foundation_fabric_august_2017_v1.1.pdf. [Online; accessed 25-March-2020].

Vedral, V., Morikoshi, F., 2007. Schrödinger's cat meets einstein's twins: A superposition of different clock times. International Journal of Theoretical Physics 47, 2126–2129. doi:10.1007/s10773-007-9568-y.

Yamashita, K., Nomura, Y., Zhou, E., Pi, B., Jun, S., 2019. Potential risks of hyperledger fabric smart contracts, in: 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), IEEE. pp. 1–10. doi:10.1109/iwbose.2019.8666486.