Detailed description of the attack scenarios

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. S_1: Node spoofing attack

Node spoofing is when an attacker steals DID credentials exploiting vulnerability V_1 and communicates with another node on behalf of the user. If cryptographic keys are not stored or maintained properly, it could cause the compromise and disclosure of private keys leading to fraudulent transactions or loss of assets. This will lead to the compromising of the integrity and privacy of the operations. Wallet theft uses classic mechanisms such as phishing, which include system hacking, the installation of buggy software, and the incorrect use of wallets. The attacker may exploit the vulnerabilities in the different DID protection schemes to conduct the following attacks: Opening communication channels with multiple nodes, sharing transport data, intercepting information from other nodes, forging of transactions, hampering normal mining operations of other miners and correlating DIDs in the ledger to track single nodes.

2. S_3: Linkability of user identities to their transactions

In BSMD, given that a node will have one unique DID per transaction, it is difficult for an attacker to correlate DIDs in the ledger to track single nodes. In fact, if users were only identified by one DID, an attacker wishing to de-anonymize users may exploit vulnerability V_3 and V_4 to construct the one-to-many mapping between users and DIDs and associate information external to the system with the users. This attack is prevented in BSMD by storing the mapping of a user to his or her DID on that user's node only and by allowing each user to generate as many DIDs as required.

However, many work points to the difficulty in maintaining anonymity where network data on user behaviour is available and illustrates how seemingly minor information leakages can be aggregated to pose significant risks Reid and Harrigan (2012). Using an appropriate network representation, it is possible to associate many DIDs with each other, and with external identifying information. With appropriate tools, the activity of known nodes can be observed in detail. Even more, large centralized nodes such as government, universities and companies are capable of identifying and tracking considerable portions of user activity.

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

In fact, user identities may be linked with their transactions by various deanonymization techniques, such as network flow and temporal analysis, address clustering, transaction fingerprinting, TCP/IP operation of the underlying peer-to-peer network and context discovery (partial node directory associating nodes and their DIDs with off-network information). Attackers may use global network properties, such as degree distribution, to identify outliers. They can use local network properties to examine the context in which a node operates by observing the nodes with which he or she interacts with either directly or indirectly. The dynamic nature of the network also enables attackers to perform flow and temporal analyses by examining the significant flows between groups of nodes over time.

3. S_4: Leakage of the location of the user

BSMD implements techniques which consist of hiding the real location of the user using either K-anonymity or a Differential privacy called Geo-indistinguishability (GeoInd). Unfortunately, these privacy protection measures are not very robust and may lead to privacy leakage of its sender. Attackers may use the weak privacy protection vulnerability V_5 to extract the real location of the user and thus compromise the confidentiality of the data.

4. S_5: Non-deterministic behavior of the contract programming language

Source code of contracts is often not public in contrast to their bytecode. For this reason, bytecode decompilers, analyzers, and automated exploit generators can be utilized by attackers to conduct code analysis and exploit the vulnerabilities V_7 and V_{21} in the program design and writing of *smart contracts*. The threat agents may stand for developers who intentionally introduce semantic bugs in *smart contracts*, bugs that represent backdoors. Most security vulnerabilities in Fabric chaincodes arise from the non-deterministic behavior of Go, which may lead to consensus failure (Huang et al., 2019).

5. S_6: Undesired behavior arising from platform features

Attackers can exploit vulnerabilities V_9 and V_{18} pertaining to design flaws in the blockchain platform, particularly some range query methods so that phantom reads (malicious data) in the code are not detected. Also, by exploiting the read your write vulnerability, attackers may force the system to get into an unexpected behavior.

6. S_7: Smart contract for malicious activities

Attackers may exploit the lack of monitoring of smart contract applications (vulnerabilities V_6 and V_{20}) to leverage *smart contracts* for a variety of malicious activities. On one hand, a Criminal Smart Contract (CSC) can facilitate the leakage of confidential information, theft of cryptographic keys, and various sabotage activities. Such a CSC might pay a reward for (confidential) delivery of a target private key. In most of the existing blockchain platforms, pseudonymous transactions provide a nest for criminal *smart contracts* (Juels et al., 2016).

Also, lack of monitoring of *smart contracts* may allow denial-of-service (DoS) attacks. Attackers could simply introduce *smart contracts* that take a very long time to execute, thus severely reducing the performance of the blockchain (Androulaki et al., 2018). To address DoS attacks from untrusted chaincode, an node in the BSMD can simply abort an execution according to a local policy if it suspects a DoS attack. Due to the permissioned nature of Fabric, detecting clients that try to mount a DoS attack by flooding the

network with invalid transactions should not be challenging. This is due to the fact that the ledger of Fabric contains all transactions, including those that are deemed invalid. One approach would be to black-list such clients according to a policy that could be put in place.

7. S_8: Installation of malware on nodes

Attackers may exploit the security design flaw (vulnerability V_9) pertaining to insufficient chaincode sandboxing to install malicious chaincode. Remote Access Trojan (RAT) malware create a foothold in a corporate network that allows other systems to be scanned and attacked.

The installation of malicious chaincode would be a nontrivial exercise for most threat actors, given the level of access required. However, plausible scenarios exist. For example, an attacker may infiltrate the organization responsible for developing and maintaining the chaincode for an existing ledger, and then publish an update containing the malicious data. Note that the chaincode does not necessarily contain any overtly malicious functionality at the time it is installed on the network. It merely needs to be able to download and execute code from a command-and-control server at some future point in time.

8. S_9: DoS attack on the host

Design miscalculations, or malware can easily result in a DoS attack if host resources are not properly configured (vulnerability V_9), because all containers share kernel resources. If one container can monopolize access to certain resources (memory, or user IDs , CPU, memory, disk I/O), it can starve out other containers on the host, resulting in DoS, whereby legitimate users are unable to access part or all of the system.

9. S_{10}: Elevated privileges gained by the user

If the host system is not configured correctly through the Docker container, an attacker can gain various privileges or can bypass isolation checks by exploiting vulnerability V_9, thus accessing sensitive information from the host. Normally, it should not be possible for an attacker to gain access to other containers or the host. However, since users are not namespaced, any process that breaks out of a container will have the same privileges on the host as it did in the container. In addition, by default, the Docker daemon runs as a root. This can cause potential elevation attacks (elevated privileges gained by user), such as those of the root user, usually through a bug in the application code that needs to run with additional privileges.

10. S_{12}: Execute code remotely with DNS attacks

Node.js and Go can be exploited to execute code remotely using the DNS rebinding vulnerability V_{10}. The attack is possible from a malicious website that accesses the web browser on a computer that has network access to the computer running the Node.js or Go process (Vagg, 2018). The malicious website can use a DNS rebinding vulnerability to trick the web browser and bypass same-origin-policy checks, allowing HTTP connections to the localhost or to a host on the local network. If a process with an active debug port is running on the localhost or on a host on the local network, the malicious website can connect to it as a debugger and get full access to the code execution.

This attack can be used to breach a private network by causing the victim's web browser to access computers at private IP addresses and return the results to the attacker. It can also be employed to use the victim machine for spamming, distributed DoS attacks, or other malicious activities.

11. S_{13}: Remote DoS attack

An attacker may exploit specific security flaws in Node.js as per vulnerability V_7 to conduct a remote DoS attack. Node.js crashing or throwing an exception could be remotely exploited using some of the existing WebSocket clients (Dawson, 2017). For validating nodes of the BSMD system, this attack leads to disruption of some blockchain dependent services. One mitigation is to peer only with white-listed nodes. Methods to prevent volumetric Distributed DoS (DDoS) include on-premise filtering with an extra network device, cloud filtering via redirection of traffic through a cloud when DDoS is detected or through a cloud DDoS mitigation service.

12. S_{14}: BGP Hijacking Attack

To intercept the network traffic of blockchain, attackers either leverage or manipulate BGP routing through vulnerability V_{11}. BGP hijacking typically requires the control of network operators, which could potentially be exploited to delay network messages. Routing attacks, including both node level and network-level attacks, may split the network, or delay the speed of block propagation. Also, internet service providers may intercept connections to conduct network hijacking attacks.

13. S_{15}: Delaying network communications in forkable blockchain systems

A fork can split the consensus group and potentially make the PBFT consensus stall, which can further be aggravated. To manipulate forks, the key strategy is to isolate a group of nodes, i.e. to partition the network for a given duration by delaying network communications between subgroups of nodes. An attacker can exploit vulnerability V_{13}, at the network level, the BGP hijacking, and at the application-level protocol to surround the targeted nodes with ones under the attacker's control. As a result, it would cause the network to fail to establish a common acceptance truth or a unique authoritative chronology blockchain.

Also, in a weakly synchronous network where block propagation and message exchange among committee members can suffer from uncertain delays, tentative blocks can result in a fork. To resolve the forks of tentative blocks, a recovery protocol should run to accept a tentative block if there is any. Specially, the recovery protocol needs to be invoked by a synchronized committee. Forks should be resolved timely and orphan consensus forks should be blocked. To "orphan" a block means to deny it into the main chain.

14. S_{16}: Adversarial Centralization of Consensus Power

A design assumption about the decentralized distribution of consensus power can be violated. In fact, an attacker can exploit vulnerability V_{14} to manipulate and modify the blockchain information. Nodes may be malicious and wish to alter the outcome of the consensus protocol by deviating from it. They may vote wrong, equivocate (tell different nodes different votes), relay wrong votes to the nodes they are connected to and lie about who they are connected to. In Byzantine attacks, a quorum of 1/3 adversarial consensus nodes might cause the protocol being disrupted or even halted. An attacker can conduct the following attacks:

- Forging of transactions by reversing transactions.
- Excluding and modifying the ordering of transactions.
- Hampering normal mining operations of other miners.
- Impeding the confirmation operation of normal transactions.

In terms of security, there are certainly advantages of private blockchains where the miners or validators cannot be anonymous. *Hyperledger* uses its own chaincode to secure transactions and achieve consensus. BSMD is a public closed blockchain, the number and the nodes that participate in consensus are known. An organization pre-selects the participants and thus, they are highly trusted. Therefore, the chances of someone acting maliciously on a network are less because malicious nodes can be identified and fines can be applied to those guilty of such practices.

15. S_{17}: Time-Validation Attacks

An attacker can exploit the timestamping consensus flaw V_{19} by connecting a significant number of nodes and propagating inaccurate timestamps. This action can slow down or speed up the victim node's network time. When such a desynchronized node creates a block, this block can be discarded by a network due to freshness constraints.

16. S_{19}: A network scheduler that thwarts the consensus protocol

Many BFT protocols assume synchronous delivery of messages. However, this assumption and vulnerability V_{17} can be violated by an unpredictable network scheduler, as demonstrated on PBFT protocol in (Miller et al., 2016). At any given time, the designated leader is responsible for proposing the next batch of transactions. If progress is not made, either because the leader is faulty or because the network has stalled, then the nodes attempt to elect a new leader. The PBFT protocol critically relies on a weakly synchronous network for liveness.

First, the scheduler assumes that a single node has crashed. Then, the network delays messages whenever a correct node is the leader, preventing progress and causing the next node in round-robin order to become the new leader. When the crashed node is the next up to become the leader, the scheduler immediately heals the network partition and delivers messages very rapidly among the honest nodes; however, since the leader has crashed, no progress is made here either. This attack violates the weak synchrony assumption because it must delay messages for longer and longer each cycle, since PBFT widens its timeout interval after each failed leader election. On the other hand, it provides larger and larger periods of synchrony as well. However, since these periods of synchrony occur at inconvenient times, PBFT is unable to make use of them.

17. S_{21}: Malleability Attack

In *Hyperledger* network, the ledger of a channel inside the *Hyperledger* limits the accessibility to only members that are part of the channel. The client can choose an endorser of its choice during the transaction proposal phase, due to which the identity of the endorser is disclosed to everyone in the network, including an insider adversary. If an attacker is a member of the channel on *Hyperledger*, as in conventional data sharing schemes, the attacker can eavesdrop all the network traffic inside a channel of *Hyperledger* fabric by exploiting vulnerability V_{15}. The attacker also has access to every transaction present in the ledger. When a sender broadcasts his transaction to other peers, the adversary can modify the content of a transaction i.e. the signature or even modify the receiver identity and then recalculate the transaction hash and further broadcast the transaction. The sender waits for the endorsement of his transaction, which never happens as the transaction hash was modified by the adversary. In this situation, the sender being confused, resend the transaction to the receiver (Andola et al., 2019).

18. S_{22}: Flooding of the nodes of the blockchain network

If no incentive is put in place an attacker can exploit vulnerability V_{22} and initiate operations in quantity in one transaction. This will cause the user to consume a lot of computing resources, and block synchronization for the *active nodes* will be significantly slower compared with the normal situation. An attacker can also initiate a DoS attack on the blockchain. They create a million empty accounts which need to be stored in the state tree. This attack causes a waste of hard disk resources. At the same time, the node information synchronization and transaction processing speed are significantly decreased.

References

Andola, N., Raghav, Gogoi, M., Venkatesan, S., Verma, S., 2019. Vulnerabilities on hyperledger fabric. Pervasive and Mobile Computing 59, 101050. doi:10.1016/j.pmcj.2019.101050.

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., Cocco, S.W., Yellick, J., 2018. Hyperledger fabric: A distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference, Association for Computing Machinery, New York, NY, USA. pp. 1–15. doi:10.1145/3190508.3190538.

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.

Juels, A., Kosba, A., Shi, E., 2016. The ring of gyges: Investigating the future of criminal smart contracts, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery, New York, NY, USA. p. 283–295. doi:10.1145/2976749.2978362.

Miller, A., Xia, Y., Croman, K., Shi, E., Song, D., 2016. The honey badger of bft protocols, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery, New York, NY, USA. p. 31–42. doi:10.1145/2976749.2978399.

Reid, F., Harrigan, M., 2012. An analysis of anonymity in the bitcoin system, in: Security and Privacy in Social Networks. Springer New York, pp. 197–223. doi:10.1007/978-1-4614-4139-7_10.

Vagg, R., 2018. March 2018 security releases. URL: https://nodejs.org/en/blog/vulnerability/march-2018-security-releases/. [Online; accessed 30-March-2020].