ACHIEVING DATA SECURITY IN THE BONUS SHARING SYSTEM "BONUSHARING"

Dolhanenko. O.D.
Scientific supervisor–senior lecturer Shirokopetleva M.S.
Kharkiv National University of Radio Electronics
(61166, Kharkiv, Nauky avenue,14, Program Engineering department,
tel. (057) 702-14-46)

The implementation of data security is arguably one of the most important aspects when developing a modern system. All application functions that somehow manipulate, store or view personalized or sensitive data should be brought to standard so that they guarantee confidentiality, integrity and availability. Since man-in-the-middle attacks could try to insert/remove data in a dataflow, it is important to follow such principles in all system data-related functions, communication and data storage. Building a safe and reliable system is not an easy task, but can be simplified by following basic methodologies and concepts created by professionals. It is recommended to implement well-known methods of security, as achieved in the "BonuSharing" system.

The security matter as come up early while developing the "BonuSharing" system. To guarantee data confidentiality, integrity and availability [1], the following aspects of system security have been paid attention to:

- securing the communication between all services and clients
- encrypting database fields which hold sensitive data
- storing keys in a secure place
- creating a secure virtual wallet for users

The system consists of microservices which communicate through a message queue. Using microservices for the backend was important, since the system should be very adaptive to peak loads and future expansions. Since using HTTPS does not allow to easily add new services (because the load balancer requires a reconfigure), it was chosen to use a messaging queue as a primary means of communication. With that the system can run partially locally and partially in the cloud, which is very important for testing purposes.

The system uses the central Gateway service to mediate secure communications between microservices and clients. The Gateway encrypts own communication with every microservice and client, performing the key exchange using the Diffy-Hellman algorithm and keeping the encryption keys in RAM.

Database level security is achieved via Spring attribute converters, which encrypt and decrypt database columns. Enabling this column conversion is done simply, by annotating the corresponding field in Spring JPA. This way any number of fields can be encrypted with different encryption algorithms or keys.

Encrypting the communication plus encrypting the database fields improve the security. However, if encryption keys will be exposed, there will no

longer be confidence in the overall system security. As mentioned above, the keys for communication encryption and decryption are stored in RAM, thus preventing anyone from getting them easily. But as for database level security, the services need to know the static encryption passwords on startup. Placing encryption keys into the source code or into a plain text file is very insecure. T he solution is to use a separate server with Vault software installed on it as a keystore.

Vault software allows to secure, store and tightly control access to tokens, passwords, certificates, encryption keys for protecting secrets and other sensitive data using a UI, CLI, or HTTP API [2]. It is based on provisioning temporary tokens to services, which can later be used to access a particular key from the Vault key database. It is very easy to deactivate or refresh any token in the system, improving the security. The communication with the Vault server is done through HTTPS, which prevents tokens and sensitive data from being stolen.

Securing the users virtual wallet against foreign application intrusion is also important. Since there is a function of buying and selling bonuses – the system hosts wallets which show the amount of money (balance) the user owns within the system. The communication and database columns are encrypted, but if the users balance is just a number and the database decryption and encryption keys get somehow temporarily or fully compromised, anyone with the right knowledge can change that number, thus illegally intrude into the system. To prevent such events, instead of the balance number the wallet holds a chain of transactions, each transaction pointing to the next. Anytime there is a request to get the balance, it is calculated from the very beginning. It is also very hard to fake a transaction request, because each transaction gets validated by the wallet service, asking the requester to provide an encrypted reason for the transaction. If the requester fails to decrypt the request or the wallet does not receive or fails to decrypt the response – the transaction is not valid and the administrators get notified right away. Inserting or removing a transaction record manually from the database is also irrelevant, as such actions break the overall chain hash (which is also stored and re-calculated upon every valid transaction addition) and the system immediately rolls back the chain to the state when it was valid.

All in all, the “BonuSharing” system is being developed with security in mind at the same time making the system architecture scalable.

List of contents

1. Data security: Definition, Explanation and Guide. URL: https://www.varonis.com/blog/data-security/

2. Vault. Manage secrets and protect sensitive data. URL:https://www.hashicorp.com/products/vault/.