Information System Security

Bitcoin

ECE PARIS
ÉCOLE D'INGÉNIEURS

Details

- **The objective of this lab is to test the reference tool for blockchain : bitcoin**
- **Download the tool on your computer from the following website : https://bitcoin.org/bin/bitcoin-core-0.17.1/bitcoin-0.17.1-i686-pc-linux-gnu.tar.gz**
- **Transfer this package to your “debian vulnerable” with SFTP**
- **Decompress it (tar zxvf)**
- **This tool contains :**
  - Bitcoind : validation daemon of transactions for official blockchain
  - Bitcoin-cli : command line to interact with this daemon
  - Bitcoin-qt : graphical tool for portfolio management

Description

- **The synchronization daemon with the blockchain (bitcoind) and the graphical client work on a default directory : /home/security/.bitcoin**
  - Launch the synchronization daemon “bitcoind”
- **Question 1 : What’s going on when launching the bitcoind daemon?**
  - Waiting for the end of synchronization may take a few weeks and it will be impossible to test anything. You can interrupt it with ctrl+c command.
- **Question 2 : Why does it takes so much time?**

Description

- **Bitcoin has three modes :**
  - Mainnet : When run with no arguments, all Bitcoin Core programs default to Bitcoin's main network (official network)
  - Testnet : is a test network on which a new blockchain has been launched, but on which you can create money. The testnet network is difficult to exploit because it has the blockchain has the same size than mainnet
    - ✓ The daemon connects to it by launching it with the -testnet option. This has the effect of creating a subdirectory in /home/security/.bitcoin/testnet that contains a structure similar to the main network
  - regtest: is not a network because the daemon will not sync to an existing blockchain. By launching the demon with this option, a regtest directory is created. It is possible to simulate the operation of bitcoin transactions on a number of nodes that you control
- **In this lab, we only use the regtest mode**
  - Start the bitcoind daemon in regtest mode

Description

- **To interact with the synchronization daemon, we use the client bitcoin-cli, which allows to pass instructions by RPC to the daemon. To interact with the regtest, you have to run bitcoin-cli with the -regtest option**
  - Bitcoin-cli -regtest -h : list RPC commands
  - Bitcoin-clid -regtest getinfo : gives information on node
  - Bitcoin-clid -regtest getpeerinfo : gives information on peer to peer network
  - Bitcoin-cli -regtest getmininfo : gives information on mining state
  - Bitcoin-cli -regtest getbalance : gives the amount of the wallet

Description

- **Bitcoin-cli -regtest generate <value>: simulates the generation of <value> blocks**
  - With the -regtest option, the daemon does not connect to any node and does not mine. The call of “generate” is used to simulate a successful mining

- **Question 3 : Indicate the command in order to get a balance of 50BTC on your account**
  - A block must have 100 confirmations before that reward can be spent, so we generate 101 blocks to get access to the coinbase transaction from block #1
  - In regtest mode only the first 150 blocks pay a reward of 50 bitcoins

Creation of a local transaction

- **Question 4 : Create a new bitcoin address and transfer 10BTC on this address**
  - Create a new address and store it in a shell variable $NEW_ADDRESS
  - Transfer 10BTC to this new address
  - Display the transaction with “listunspent” command
    - ✓ You should see two transactions : If we had spent those BTC to someone else, that second transaction would not be displayed in our list of transactions
  - Create a new block to confirm the transaction above (takes less than a second) and clear the shell variable (use unset)
  - Display the transaction with “listunspent” command

Creation of a raw transaction

- **Question 5 : Create a custom transaction using createrawtransaction to send 49.9999BTC**
  - Create a new address and store it in a shell variable $NEW_ADDRESS
  - Create a raw transaction with two arguments : the first argument (a JSON array) references txid and vout parameters of the previous transaction (use “listunspent” to find it) and the second argument (a JSON object) contains the address (public key hash) and number of bitcoin (49.9999BTC) we want to transfer
    - ✓ bitcoin-cli -regtest createrawtransaction '[{ "txid": "$TXID", "vout": $VOUT }]' '{ "$NEW_ADDRESS": 49.9999 }'
  - Save the raw transaction in shell variable $RAW_TX
- **The transaction will include a fee of 0.0001BTC (because our input was 50BTC and we spent 49.9999BTC)**

Sign and send a transaction

- **Question 6 : Use the signrawtransactionwithwallet RPC to sign the transaction created by createrawtransaction**
  - bitcoin-cli -regtest signrawtransactionwithwallet $RAW_TX
  - Save the returned “hex” raw format signed transaction to shell variable $SIGNED_RAW_TX
- **Question 7 : Send the signed transaction to the connected node using the sendrawtransaction RPC**
  - bitcoin-cli -regtest sendrawtransaction $SIGNED_RAW_TX
  - Generate a block to confirm the transaction and clear our shell variables