Secure Systems Development

Assignment 3, Option A

(Due for submission on 18th May 2024)

Although their long-term utility and sustainability remains to be seen, cryptocurrencies are a hugely interesting application of many modern cryptographic principles. The first cryptocurrency, Bitcoin, was announced in 2008 and launched in 2009. This gave it a considerable first-mover advantage. Although it is no longer the most sophisticated cryptocurrency system in existence, it is the most widely used.

The aim of this assignment is to develop a working Bitcoin blockchain viewer application. This should be similar in spirit to existing viewers such as the Blockchain Explorer, although the scope and complexity is expected to be much smaller. The application should display blocks as they are mined by the Bitcoin system. For each block, it should display:

- The date and time that the block was added to the blockchain, formatted in a human readable form (e.g. 1st April 2024 at 20:40)
- A list of the transactions in the block, and the value of each transaction
- The nonce that was used to successfully hash the block, and the difficulty level
- Verify that that hash does indeed match the hash included in the block!

How do I do this?

There are four main steps involved:

- Study the types of messages that are transmitted between nodes in the Bitcoin network, and write code to parse them
- Make your program connect to the Bitcoin network and start receiving broadcasts when new transactions are sent or new blocks are mined
- Download details of the event as it occurs, and display some interesting information to the user (for example, you could display the value of a transaction, the hash and nonce of a block etc.)
- Put this all in a loop to keep it running forever!

Bitcoin transactions

You can read about the Bitcoin transaction format here:

https://en.bitcoin.it/wiki/Protocol_documentation

The basic gist is all messages sent over the network use a specific datagram format, consisting of:

- The magic number **`0xD9B4BEF9`**, encoded in little endian format, i.e. as **`0xF9BEB4D9`**
- 12 bytes, indicating the *type* or (*command*) of transaction being send, with zeros as padding at the end if the type is less than 12 bytes long
- The *length* of the payload in bytes, encoded as a 32-bit value
- A *checksum*, the first four bytes of the double SHA256 hash of the payload (the double SHA256 of a value is the SHA256 of the SHA256 of this value!)
- The *payload* itself

The website contains detailed information about the encoding of each of the payloads that can be used in the payload field.

Connecting to the Bitcoin network

To make the initial connection, you first need to know the IP address of at least one existing Bitcoin node to connect to. If you don't know any, you can find them by doing a DNS lookup on a special Bitcoin lookup domain, which will get you list of reliable IP addresses for Bitcoin nodes. One example of such a domain is **`seed.bitcoin.sipa.be`**, and you can find others by googling.

Once you have an IP, connecting involves:

- Sending a **`version`** payload to the node
- The node will reply by sending back the **`version`** payload, and a **`verack`** payload
- Your program should send another **`verack`** back
- The node will then being sending you *events* as they occur!

Note that once you have a working connection, you can also send a **`getaddr`** payload to the node, and it should respond with an **`addr`** payload, telling you about all the other nodes it knows. You can store this information in a file, which allows you to build up a database of known IP addresses. Note that you can also use the ping payload to check if a node is still online. (Note: storing IP addresses like this isn't needed for the assignment, I'm including the information for those who are interested! You can work on a hard-coded IP address for the assignment!)

Getting events

The node you connect to should send events to your machine as they occur, in real time. The events come in the form of the **`inv`** payload. Each inv contains a list of *inventory vectors*, each of which is basically an alert about an event that might interest you. Each inventory vector is 36 bytes long, and contains two things:

- A 32-bit value indicating the *type* of event (for example, 1 indicates a transaction, 2 a block)
- The *double hash* of the transaction or block, which will be 32 bytes long

You can ask for the full transaction or block by sending a **`getdata`** payload, containing the list of inventory vectors that interest you. You will get **`tx`** or **`block`** payload back in response.

These structures will contain detailed information about the transaction or block.

What should my program look like?

The program can be written in any language of your choice. However, it must be reliably cross-platform and run on multiple operating systems (try running it on a Linux VM to verify this).

The form of user interface the program provides is up to you. It can be command line based only, or it could present a GUI or web UI. Some marks will be awarded for the quality of information presentation, but unless the system is particularly strange to use, this will be very open ended and easy to gain marks on.

The application **must** be developed using an online version control repository service. This could be GitHub, Codeberg, GitLab or any similar service. The submission must consist of a link to this repo. All documentation for the system should exist only in this repository. The documentation should consist of instructions explaining how to build and run the application. This should include information on any dependencies required (interpreters/compilers/runtimes etc.) and how to install them. Furthermore, the code should be well formatted and commented. The documentation should also have a brief overview of the internal architecture of the codebase, and some information about how to use it.

How do I submit?

The assignment should be submitted via Brightspace before 11:59PM on Saturday 18th May 2024.

As noted above, the submission on Brightspace should consist only of a link to a version control repository containing your source code. Take care not to make modifications to your code after the submission deadline!

Note that this assignment is worth 30% of the total mark for this class.

How will the assignment be marked?

Marks will be awarded for implementing the following:

- Up to **10 marks** for connecting to the Bitcoin network, and receiving the inv payloads
- Up to **10 marks** for successfully parsing the details of the blocks as they arrive
- Up to **5 marks** for displaying this information in a manner that is understandable to somebody using the system
- Up to **5 marks** for clean code and documentation. This includes commenting, a README file explaining how to run the code, and a sane decomposition of the program into functions/classes/etc.

Good luck!