

Debunking Apollo. Again.

TLDR

1. There are no changes to the handling of private transactions in Apollo Olympus 2.0. All private transactions have been exported to a csv file which is available for download on our website: https://www.jelurida.com/sites/default/files/private_transactions.csv
2. The database level mixing they announce is referring to a central “mixing” server operated by their online wallet as a closed source mixer which can likely de-anonymize all transactions. In fact anybody can do that as we demonstrate here by solving the two challenges posted by Bitfi and an Apollo supporter:
<https://www.jelurida.com/sites/default/files/bitfi-challenge.pdf>
<https://www.jelurida.com/sites/default/files/godzilla-challenge.pdf>
3. The node IP masking is another centralized solution supported by servers operated by the Apollo foundation.
4. We will not be updating the snakeoil node since it has fulfilled its purpose and we do not intend to maintain indefinitely a customized Apollo node. Instead we have published the complete list of “private” transactions up until now (April 3, 2019) and the small snippet of code that anyone can use to recreate or update this list (see below the technical part).

In the last few days there has been a lot of social media activity coming from Apollo Community members (humans and bots) on Twitter, Telegram and other channels claiming that we intentionally spread “fud” about the Apollo Currency project. The reason for this was an article in CCN.com pointing out some “red flags” about Apollo and using as reference the “snake oil” server that is accessible through our website <https://www.jelurida.com/apollo-snake-oil>. Shortly after that, the author of the article stated on Twitter that he had not done proper research and would be correcting the article. CCN also submitted an apology to the Apollo project <https://www.ccn.com/ccn-an-apology-to-apollo-john-mcafee-and-our-readers>. In the meantime, we and our community members have been called “fudsters”, “haters,” and much worse, on various social media channels.

So, in the interest of straightening things out, we felt compelled to take a closer look into an issue which we frankly considered something of the past.
But first - a little context for the people not familiar with the whole story:

The Apollo project <https://apollocurrency.com/en/> was launched last year and it is a clone of Nxt. Being one of many such clones, it specifically attracted our attention because of two things:

- They had copied the code and launched a new blockchain but they had forgotten three key requirements: to quote the proper license, to release the source code, and to include the proper copyright notices of Nxt developers and Jelurida <https://www.jelurida.com/jelurida-apollo-public-statement-per-the-IPL>.
- They didn't give any credit to Nxt and presented Apollo as if they developed all the available features - which were the result of our 5 years of development and production experience.

And while we fought to correct the IP violation (and succeeded at the end) - there was little we could do about the second issue which is partly a question of decency and moral integrity - things that cannot be enforced by law.

Apollo also claimed (and still does) that they are on a “privacy mission” and that they have

implemented “private transactions” on their blockchain. Naturally, people were asking us “Is that true, is it safe to use this feature?” And they were asking us because, well, who better to ask than the Nxt developers who actually wrote the codebase?

So, we took the time to look at the code and discovered that the so-claimed “private transactions” are actually very visible on the blockchain and they are only not visible in the UI. To prove this, we had created the “snakeoil” - a modified version of Nxt to interact with the Apollo blockchain, to show that actually the “private” transactions are not so private and to remind our community members to always double check and do as much research as possible when it comes not only to Apollo, but to any other cryptocurrency project. Shortly after that, Apollo started “explaining” that this feature is only at the “API level” - a statement ridiculous unto itself. The illusion of privacy is even worse than the lack of privacy. Those who do not understand what “API” means may think that it takes a quantum computer or highly advanced cryptographic knowledge to reveal their transactions, while in reality, a beginner developer can do it with 10 lines of code.

Today, apparently this issue is brought into light again in connection with the above mentioned article. Apollo says that now the privacy is on the “Database level,” which again, we didn’t know exactly how to interpret. So once again, we had to spend time to look into their code and see if they have actually accomplished their privacy mission. But not even close....

What we found was the following:

1. Nothing in the “private transactions” has changed. If they knew this feature doesn’t work - why do they continue to keep it?
2. We came to understand that this “Database level” privacy is achieved by using a centralized mixer, which again raises the question - where is the privacy in that and what is the definition of privacy as understood by Apollo? To keep complete control over users’ privacy and funds? This totally defeats the purpose of the blockchain - send your funds to some “mixer”, trust it completely, and hope to receive it in another account after the mixing...

In a Medium article they state:

“Database-Level Coin Mixing

Olympus Protocol 2.0 adds coin mixing feature which offers an extra layer of privacy and security. The function filters a sender’s transaction within Apollo’s database. It mixes the transaction with others that were sent in the same time period—the effect is to completely mask the funds’ destination. On the blockchain, the coin mixing feature shows a sender’s transaction but not the destination.”

<https://medium.com/@apollocurrency/-fc4d9980877c>

So “within Apollo’s database” actually mean their own servers.... If this mixing all happens on a database owned by Apollo - they and/or their computers inevitably know the destination of the mixing funds, and even if we assume that they don’t keep logs or record the recipients now, nothing can stop them from doing so in the future. But this is not the only problem with this “mixer” - its algorithm is so trivial that it is possible to manually track all transactions. We were able to do that for two different examples as documented here (one of them going through the “mixer” twice):

<https://www.jelurida.com/sites/default/files/bitfi-challenge.pdf>

<https://www.jelurida.com/sites/default/files/godzilla-challenge.pdf>

This is also fundamentally different from the fully decentralized coin shuffling as pioneered by Nxt more than 4 years ago, which does not rely on any centralized third party. They should have stuck to it. Even with the “mixer” being closed source and us having no knowl-

edge of its exact algorithm - it was straight forward to track the transactions that went through it.

To support our findings, we include below a full technical report.

We will not be commenting on Apollo's "autoupdate" anti-feature (automatic update for cryptocurrency software ?!?) or the closed source binary doing "IP - masking", because they go out of the scope of this article.

We will be posting this article also on nxtforum and r/NXT and we invite all independent developers to join the conversation and review our conclusions. We will also be reaching out to CCN with these findings and hope that they will respect our right to respond to the their latest article.

Apollo analysis, based on the latest code published on Github as of 2019-04-02, as of last master branch commit:

<https://github.com/ApolloFoundation/Apollo/commit/a4ddeacaa06e473be75ffd07fb79283c6271b439>

1. There has been no change in the processing of the private transactions, those of type 0, subtype 1 (“PrivatePayment”) compared to our last review of this code in 2018.

2. By making the following trivial code modification (new code in yellow) in TransactionType.java:

```
private static final BufferedWriter writer;
static {
    try {
        writer = Files.newBufferedWriter(Paths.get("private_transactions.csv"));
        writer.write("height,timestamp,transaction_id,sender,recipient,amount,fee");
        writer.newLine();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}

final void apply(TransactionImpl transaction, Account senderAccount, Account recipientAccount) {
    long amount = transaction.getAmountATM();
    long transactionId = transaction.getId();
    if (!transaction.attachmentIsPhased()) {
        senderAccount.addToBalanceATM(getLedgerEvent(), transactionId, -amount, -transaction.getFeeATM());
    } else {
        senderAccount.addToBalanceATM(getLedgerEvent(), transactionId, -amount);
    }
    if (recipientAccount != null) {
        recipientAccount.addToBalanceAndUnconfirmedBalanceATM(getLedgerEvent(), transactionId, amount);
    }
    if (transaction.getType() == Payment.PRIVATE) {
        try {
            writer.write(String.format("%d,%d,%s,%s,%s,%d,%d",
                transaction.getHeight(),
                transaction.getTimestamp(),
                transaction.getStringId(),
                Convert.rsAccount(senderAccount.getId()),
                Convert.rsAccount(recipientAccount.getId()),
                amount,
                transaction.getFeeATM()));
            writer.newLine();
            writer.flush();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
    applyAttachment(transaction, senderAccount, recipientAccount);
}
```

we were able to export to a CSV file a complete list of all 19k private payment transactions up until now. The list is available for download here: https://www.jelurida.com/sites/default/files/private_transactions.csv

3. Comparing the result of installing the binary release:

<https://github.com/ApolloFoundation/Apollo/releases/download/1.26.3/ApolloWallet-linux-1.26.3.sh>

with the source repository obtained by cloning the Apollo Github:

git clone <https://github.com/ApolloFoundation/Apollo.git>

the binary release contains two important additional components, the source code of which is not provided:

```
root@dinosaur:/usr/local/ApolloWallet/ApolloWallet# ls -l tor/
total 17868
-rw-r--r-- 1 root staff 4251904 Jan 24 17:33 geoip
-rw-r--r-- 1 root staff 2378805 Jan 24 17:33 geoip6
-rw-r--r-- 1 root staff 2575848 Jan 24 17:33 libcrypto.so.1.1
-rw-r--r-- 1 root staff 330128 Jan 24 17:33 libevent-2.1.so.6
-rw-r--r-- 1 root staff 451728 Jan 24 17:33 libssl.so.1.1
-rwxr-xr-x 1 root staff 5297944 Jan 24 17:33 tor
-rw-r--r-- 1 root staff 2707304 Jan 24 17:33 tor-gencert
-rw-r--r-- 1 root staff 1375 Jan 24 17:33 torify
-rw-r--r-- 1 root staff 272632 Jan 24 17:33 tor-resolve
-rw-r--r-- 1 root staff 8999 Jan 24 17:33 torsocks

root@dinosaur:/usr/local/ApolloWallet/ApolloWallet# ls -l secureTransport/
total 8248
-rw-r--r-- 1 root staff 277 Apr 2 16:03 apltrans000.log
-rwxr-xr-x 1 root staff 111 Jan 24 17:33 runClient.sh
-rwxr-xr-x 1 root staff 8431576 Jan 24 17:33 securenodexchg
-rw-r--r-- 1 root staff 279 Jan 24 17:33 servers.json
```

The **tor** subdirectory apparently contains a tor installation, to allow for proxying peer connections over tor. This ability is *already present* in Nxt, except that we don't bundle tor itself in our releases. Instead, we expect advanced users to install and configure a tor proxy themselves. We have not tried to find out if the tor binaries and libraries bundled with the Apollo installer have been customized in any way, which could necessitate their bundling instead of relying on a user-provided tor proxy.

The **secureTransport** subdirectory is more interesting, as it contains a closed source binary, securenodexchg. When the Apollo server is started using the run-secure-transport.sh script, which in turn calls secureTransport/runClient.sh, the securenodexchg binary is executed (with root privileges), to apparently start a server and create a local 10.75.110.0 tunnel interface which proxies the peer networking requests from the Apollo server to one or more of the IP addresses hardcoded in secureTransport/servers.json:

```
root@dinosaur:/usr/local/ApolloWallet# more ApolloWallet/run-secure-transport.sh
#!/bin/bash

# WARNING: java still bypasses the tor proxy when sending DNS queries and
# this can reveal the fact that you are running Apl, however blocks and
# transactions will be sent over tor only. Requires a tor proxy running
# at localhost:9050. Set apl.shareMyAddress=false when using tor.

if [ -x jre/bin/java ]; then
    JAVA=./jre/bin/java
else
    JAVA=java
fi

unamestr=`uname`
if [[ "$unamestr" == 'Linux' ]]; then

    cd secureTransport
```

```
sudo ./runClient.sh
cd ..
fi

xdock="

if [[ "$unamestr" == 'Darwin' ]]; then
xdock=-Xdock:icon=./favicon.ico
fi

${JAVA} $xdock -DsocksProxyHost=10.75.110.1 -DsocksProxyPort=1088 -Dapl.runtime.
mode=desktop -Dapl.enablePeerUPnP=false -jar Apollo.jar
```

```
root@dinosaur:/usr/local/ApolloWallet# more ApolloWallet/secureTransport/runClient.sh
killall -KILL securenodexchg
pkill -KILL securenodexchg
./securenodexchg --mode 0 --shuffle --unique 20001 &
```

```
root@dinosaur:/usr/local/ApolloWallet# file ApolloWallet/secureTransport/securenodexchg
ApolloWallet/secureTransport/securenodexchg: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/Linux 3.2.0, BuildID[sha1]=dded4cd869ee69c82c9d3b8d4e97a4b636a16e9c, not stripped
```

```
root@dinosaur:/usr/local/ApolloWallet# more ApolloWallet/secureTransport/servers.json
{
  "servers": [
    "51.15.249.23:25000",
    "163.172.182.242:25000",
    "51.15.228.121:25000",
    "163.172.187.100:25000",
    "51.15.217.97:25000",
    "212.47.254.71:25000",
    "51.15.249.27:25000",
    "163.172.172.121:25000",
    "51.15.236.93:25000"
  ]
}
```

Output from running securenodexchg:

```
[2019-04-02 17:26:28.185835][info]: Unique mode of operation is enabled
[2019-04-02 17:26:28.186066][info]: UniqueModePort: 20001
[2019-04-02 17:26:28.186118][info]: starting singularity locker
[2019-04-02 17:26:28.186174][info]: Running in 'CLIENT' operation mode
host: 51.15.249.23
port: 25000
host: 163.172.182.242
port: 25000
host: 51.15.228.121
port: 25000
host: 163.172.187.100
port: 25000
host: 51.15.217.97
port: 25000
host: 212.47.254.71
port: 25000
host: 51.15.249.27
port: 25000
host: 163.172.172.121
port: 25000
```



```
host: 51.15.236.93
port: 25000
[2019-04-02 17:26:28.186511][info]: Shuffle routine activated
[2019-04-02 17:26:28.186565][info]: 163.172.172.121:25000
[2019-04-02 17:26:28.186601][info]: 163.172.182.242:25000
[2019-04-02 17:26:28.186635][info]: 51.15.217.97:25000
[2019-04-02 17:26:28.186665][info]: 51.15.249.23:25000
[2019-04-02 17:26:28.186700][info]: 51.15.249.27:25000
[2019-04-02 17:26:28.186734][info]: 163.172.187.100:25000
[2019-04-02 17:26:28.186767][info]: 51.15.228.121:25000
[2019-04-02 17:26:28.186800][info]: 51.15.236.93:25000
[2019-04-02 17:26:28.186833][info]: 212.47.254.71:25000
[2019-04-02 17:26:29.188347][info]: connecting: 163.172.172.121 : 25000, entity: 0
[2019-04-02 17:26:29.188577][info]: final socket receive buf: 425984
[2019-04-02 17:26:29.188629][info]: final socket send buf: 425984
[2019-04-02 17:26:29.188676][info]: shifting to: 1
[2019-04-02 17:26:29.218258][info]: json data: {"tunaddr":"10.75.110.75","netmask":"255.255.255.0"}


[2019-04-02 17:26:29.218462][info]: tunaddr: 10.75.110.75
[2019-04-02 17:26:29.218506][info]: netmask: 255.255.255.0
[2019-04-02 17:26:29.218542][info]: raising tun interface
opening :
[2019-04-02 17:26:29.219801][info]: Tunnel device id: 6
```

Apparently, this securenodexchg tunneling proxy is what implements the so-called IP-Masking Apollo feature. We can only question the sanity of someone running a closed source binary, as root, that creates a tunnel interface and establishes network connections, on a machine used for handling cryptocurrency. As for any “privacy” such a proxy can provide - it is all at the mercy of those running the servers defined in secureTransport/servers.json. Which we assume is the Apollo Foundation.


4. Regarding the “database level mixing” touted by the Apollo team, we were unable to find anything in the published source code that can act as a mixer, other than the well known and fully copied from Nxt without changes coin shuffling. We now assume that this feature description refers to a centralized mixing service that the Apollo Foundation runs. We also noticed that when using the online Apollo wallet, available at <https://apollowallet.org/>, when trying to send a “private” transaction, the UI offers a choice of “Use Mixer”:


Send Apollo ✕

Recipient  

Amount  Apollo

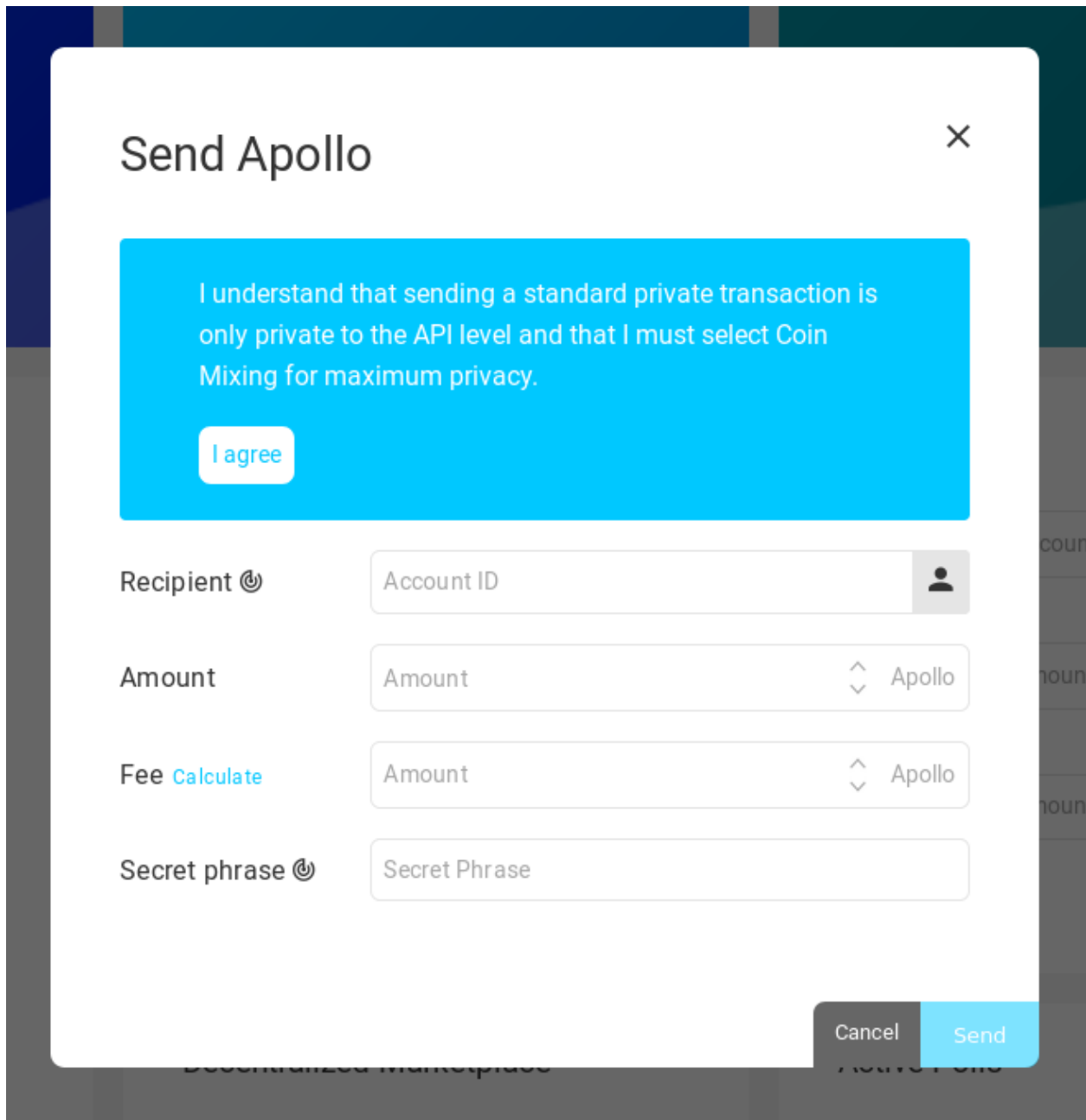
Use Mixer

Fee [Calculate](#)  Apollo

Secret phrase 

Cancel Send

However, no such “Use Mixer” checkbox is available when running a local installation, either the one produced by the binary installer:



or the one produced by compiling from latest source:

Privately send Apollo

Recipient ⓘ

Recipient Account ⓘ

Amount

Amount ⓘ Apollo

Fee Calculate

Minimum Fee ⓘ Apollo

Passphrase ⓘ

Cancel Submit

DATE ⓘ TYPE AMOUNT FEE ACCOUNT HEIGHT CONFIRMATION

View More Account Ledger

Copyright © 2017-2018 Apollo Foundation. Apollo Version: 1.27.3

Copyright © 2016-2017 Jelurida IP B.V.

Copyright © 2013-2016 The Nxt Core Developers.

We are unable to investigate further how the “Use Mixer” feature works, given that it is only present on the online Apollo wallet, but as this mixer is not part of the decentralized blockchain consensus, is not open source, and is not available for other independent parties to run, relying on such centralized mixer for privacy ultimately requires full trust in those running the servers, i.e. the Apollo Foundation.