



华中科技大学

区块链技术与应用实验报告

姓 名： 侯竣

学 院： 网络空间安全学院

专 业： 密码科学与技术

班 级： 密码 2101 班

学 号： U202116003

指导教师： 代炜琦

分数	
教师签名	

2023 年 11 月 20 日

目 录

1. Fabric 实验	1
1.1 实验目的.....	1
1.2 实验内容及结果.....	1
1.2.1 任务 1.....	1
1.2.2 任务 2.....	3
1.3 实验中的问题.....	5
1.4 实验总结及建议.....	5
2.Ethereum 实验	7
2.1 实验目的.....	7
2.2 实验内容及结果.....	7
2.2.1 任务 1.....	7
2.2.2 任务 2.....	8
2.3 实验中的问题.....	10
2.4 实验总结及建议.....	10

1.Fabric 实验

1.1实验目的

- ✧ 本实验的目的是让学生将从书本中学到的有关区块链的知识应用到实践中。在 fabric1.4 的架构下，使用 docker 的容器服务搭建一个具有 5 个节点的简单联盟链，了解基本的共识，出块，部署、调用智能合约（chaincode）的功能。
- ✧ 本实验共有两个任务，第一个任务是使其自己尝试如何搭建一个 fabric1.4 的基础区块链网络，第二个任务是让学生了解在 fabric 的架构下如何去编写、调用智能合约(chaincode)。学生需要了解其基本原理，并根据简单的业务需求（投票）来设计、实现 chaincode。

1.2实验内容及结果

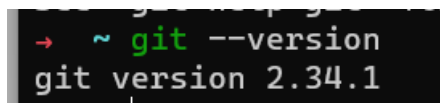
1.2.1 任务 1

这个任务是搭建一个具有 5 个节点的联盟链，这里使用的是 docker compose 搭建方法

根据指导手册上的要求，分别安装 git, curl 和 docker, go lang 如下图所示：

有几点需要注意的：

1. Docker 不能 apt 安装，可以直接去 docker 官网 <https://www.docker.com/> 查看安装方法
2. Golang 也不能 apt 安装，需要去官网下载压缩包安装，并且加入 profile 环境变量 export=go bin 的路径



```
→ ~ git --version
git version 2.34.1
```

图 1-1 git 安装

```

➔ ~ curl --version
curl 7.81.0 (x86_64-pc-linux-gnu) libcurl/7.81.0 OpenSSL/3.0.2 zlib/1.2.11 brotli/1.0.9 zstd/1.4.8 libidn2/2.3.2 libpsl/
0.21.0 (+libidn2/2.3.2) libssh/0.9.6/openssl/zlib nghttp2/1.43.0 librtmp/2.3 OpenLDAP/2.5.16
Release-Date: 2022-01-05
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sftp smb smb
s smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB PSL SPNE
GO SSL TLS-SRP UnixSockets zstd

```

图 1-2 curl 安装

```

➔ ~ docker version
Client: Docker Engine - Community
Cloud integration: v1.0.33
Version: 24.0.2
API version: 1.43
Go version: go1.20.4
Git commit: cb74dfc
Built: Thu May 25 21:52:17 2023
OS/Arch: linux/amd64
Context: default

Server: Docker Desktop
Engine:
Version: 24.0.2
API version: 1.43 (minimum version 1.12)
Go version: go1.20.4
Git commit: 659604f
Built: Thu May 25 21:52:17 2023
OS/Arch: linux/amd64
Experimental: false
containerd:
Version: 1.6.21
GitCommit: 3dce8eb055cbb6872793272b4f20ed16117344f8
runc:
Version: 1.1.7
GitCommit: v1.1.7~g860f061
docker-init:
Version: 0.19.0
GitCommit: de40ad0

```

图 1-3 docker 安装

```

➔ ~ docker compose version
Docker Compose version v2.18.1

```

图 1-4 docker compose 安装

```

➔ ~ go version
go version go1.21.1 linux/amd64

```

图 1-5 go 安装

然后 git clone fabric 库:

git clone <https://github.com/hyperledger/fabric-samples>

下载安装脚本到刚刚 fabric-samples 中:

wget

[https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.s](https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh)
h

执行: `sudo ./install-fabric.sh docker` 拉取镜像

```
./test-network x + v
+ fabric-samples git:(main) sudo ./install-fabric.sh docker
[sudo] password for nolan:

Pull Hyperledger Fabric docker images

FABRIC_IMAGES: peer orderer ccenv tools baseos
====> Pulling fabric Images
====> docker.io/hyperledger/fabric-peer:2.5.4
2.5.4: Pulling from hyperledger/fabric-peer
Digest: sha256:c2e735a3cb8250c47ed3589294b3f0c078004ec001b949710f334736651e106d
Status: Image is up to date for hyperledger/fabric-peer:2.5.4
docker.io/hyperledger/fabric-peer:2.5.4
====> docker.io/hyperledger/fabric-orderer:2.5.4
2.5.4: Pulling from hyperledger/fabric-orderer
Digest: sha256:1a7144705b435062f4be2886de98d0408165cf51326ec721c3f58b40bf8bf139
Status: Image is up to date for hyperledger/fabric-orderer:2.5.4
docker.io/hyperledger/fabric-orderer:2.5.4
====> docker.io/hyperledger/fabric-ccenv:2.5.4
2.5.4: Pulling from hyperledger/fabric-ccenv
Digest: sha256:b7d312c0f8d4530c6ff4b3ef7277f0338f486d0b617c3012415a32308cbc2254
Status: Image is up to date for hyperledger/fabric-ccenv:2.5.4
docker.io/hyperledger/fabric-ccenv:2.5.4
====> docker.io/hyperledger/fabric-tools:2.5.4
2.5.4: Pulling from hyperledger/fabric-tools
Digest: sha256:b3aea8173802186244663334b4196c415cb267d77ea8c617fdd30652c5a732c4
Status: Image is up to date for hyperledger/fabric-tools:2.5.4
docker.io/hyperledger/fabric-tools:2.5.4
====> docker.io/hyperledger/fabric-baseos:2.5.4
2.5.4: Pulling from hyperledger/fabric-baseos
Digest: sha256:cab2ed27ed8d27e4cacde447a1de51884f0bd9e103a6119e74521e1b90b6d106
```

图 1-6 拉取 fabric 镜像

然后进入 `test-network` 执行 `./network.sh up` 启动容器

```
+ test-network git:(main) ./network.sh up
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb'
LOCAL_VERSION=v2.5.4
DOCKER_IMAGE_VERSION=v2.5.4
[+] Building 0.0s (0/0)
[+] Running 5/5
✓ Network fabric_test Created 0.0s
✓ Container orderer.example.com Started 1.0s
✓ Container peer0.org1.example.com Started 0.8s
✓ Container peer0.org2.example.com Started 0.8s
✓ Container cli Started 1.2s
CONTAINER ID IMAGE COMMAND NAMES CREATED STATUS POR
TS
f6c329ada7bb hyperledger/fabric-tools:latest "/bin/bash" cli 1 second ago Up Less than a second
172e9478c217 hyperledger/fabric-peer:latest "peer node start" peer0.org1.example.com 1 second ago Up Less than a second 0.0
fa3610a2d8b2 hyperledger/fabric-peer:latest "peer node start" peer0.org2.example.com 1 second ago Up Less than a second 0.0
e51ebec7cb2e hyperledger/fabric-orderer:latest "orderer" orderer.example.com 1 second ago Up Less than a second 0.0
```

图 1-7 启动区块链网络

如图所示，容器启动成功，说明区块链网络搭建完毕

1.2.2 任务 2

这个任务要求在 fabric 网络下编写 chaincode 实现投票程序

接任务 1，需要在网络上创建 channel，执行以创建 channel

`./network.sh createChannel`

然后编写合约，按照指导手册上给出的 API 接口文档，分别实现 interface 的两个函数，即 `init` 和 `invoke` 函数，以及所需的投票功能函数：

在这里我实现的是 VoteUser, GetUserVote, GetAllVotes 三个函数, 分别代表给某个用户投票, 获取某个用户的投票情况和获取所有用户的投票情况

然后部署合约:

```
./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go
```

注意需要修改../asset-transfer-basic/chancode-go 为刚刚编写的合约目录

部署完成后, 再编写客户端, 分别实现上面三个函数的功能客户端操作, go run main.go 运行:

```
→ vote-app git:(master) x go run main.go
2023/11/27 20:05:56 输入数字选择操作:
2023/11/27 20:05:56 1 : 查询所有投票结果
2023/11/27 20:05:56 2 : 投票
2023/11/27 20:05:56 3 : 查询某个用户的投票结果
1
[
  {
    "id": 1,
    "username": "hj",
    "votes": 1
  },
  {
    "id": 0,
    "username": "nolan",
    "votes": 1
  }
]
2023/11/27 20:06:01 输入数字选择操作:
2023/11/27 20:06:01 1 : 查询所有投票结果
2023/11/27 20:06:01 2 : 投票
2023/11/27 20:06:01 3 : 查询某个用户的投票结果
```

图 1-8 查询所有用户投票情况

如图为查询操作, 刚刚已经给 hj 和 nolan 投了两票, 接下来对 nolan 投票并查询 nolan 的票数, 结果为 2, 增加一票:

```

2023/11/27 20:07:18 输入数字选择操作：
2023/11/27 20:07:18 1 : 查询所有投票结果
2023/11/27 20:07:18 2 : 投票
2023/11/27 20:07:18 3 : 查询某个用户的投票结果
2
2023/11/27 20:07:22 输入投票的目标用户名：
nolan
2023/11/27 20:07:26 输入数字选择操作：
2023/11/27 20:07:26 1 : 查询所有投票结果
2023/11/27 20:07:26 2 : 投票
2023/11/27 20:07:26 3 : 查询某个用户的投票结果
3
2023/11/27 20:07:27 输入查询的目标用户名：
nolan
{
  "id": 0,
  "username": "nolan",
  "votes": 2
}
2023/11/27 20:07:59 输入数字选择操作：
2023/11/27 20:07:59 1 : 查询所有投票结果
2023/11/27 20:07:59 2 : 投票
2023/11/27 20:07:59 3 : 查询某个用户的投票结果
|

```

图 1-9 用户投票

实验完成

1.3 实验中的问题

1. 注意 golang 和 docker 的版本，直接 apt 安装的话版本可能比较老，出现版本兼容性问题，golang 需要去官网下载最新的压缩包配置环境变量安装，docker 需要添加 docker 社区仓库源安装
2. 部署后没有创建 channel 是运行不了的，查看官网文档发现需要 createChannel
3. Docker 拉取比较慢，需要换国内源

1.4 实验总结及建议

总结：

在实验的第一部分，我成功地在 fabric1.4 架构下，使用 docker compose

搭建了一个包含 5 个节点的简单联盟链。在这个实践过程中，我理解并应用了区块链的基本概念，包括共识、出块等，还学习了如何配置和运行一个区块链网络。

在实验的第二部分，探索了智能合约（chaincode）的创建和调用。编写了一个投票的服务端，设计和实现了一个简单的智能合约。实现了查看投票和增加投票两个功能，这部分类似于一个分布式的 k-v 数据库

建议：

1. 实验手册有些部分给的不是特别详细，此外可以增加一些搭建方式，例如官方文档中给出的 bring up the tset network，链接：
https://hyperledger-fabric.readthedocs.io/en/release-2.5/test_network.html#bring-up-the-test-network
这种方式安装更方便一些
2. 部分工具版本比较老，fabric 最新的 release 是 2.5 版本，可以考虑升级一下实验手册的内容

2.Ethereum 实验

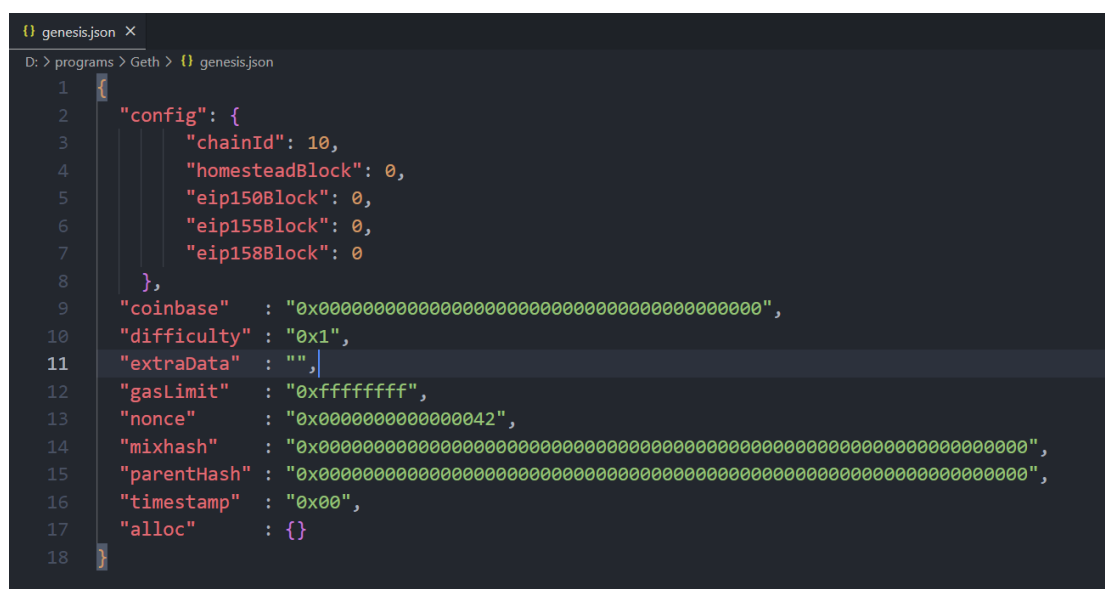
2.1 实验目的

- ✧ 本实验的目的是让学生将从书本中学到的有关区块链的知识应用到实践中。在 Geth 环境下，自行搭建一个私有网络，并掌握以太坊的基本命令，学习编译和调用以太坊智能合约，并最终复现冲入漏洞。
- ✧ 本实验共有两个任务，第一个任务是使其自己尝试如何搭建一个以太坊的多节点私有网络，第二个任务是让学生了解在以太坊的架构下如何去编写、调用智能合约。学生需要了解冲入漏洞的基本原理，并设计代码来复现重入漏洞攻击。

2.2 实验内容及结果

2.2.1 任务 1

首先下载 geth，下载完毕后找到创世块文件 genesis.json



```
1 {
2   "config": {
3     "chainId": 10,
4     "homesteadBlock": 0,
5     "eip150Block": 0,
6     "eip155Block": 0,
7     "eip158Block": 0
8   },
9   "coinbase" : "0x0000000000000000000000000000000000000000000000000000000000000000",
10  "difficulty" : "0x1",
11  "extraData" : "",
12  "gasLimit" : "0xffffffff",
13  "nonce" : "0x0000000000000042",
14  "mixhash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
15  "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
16  "timestamp" : "0x00",
17  "alloc" : {}
18 }
```

图 2-1

如图，调整 difficulty=0x1 加快挖矿速度

然后输入命令:

geth --datadir ./data0 init genesis.json

初始化创世块

```
D:\programs\Geth> geth --datadir data0 init genesis.json 11/28/2023 06:09:28 下午
INFO [11-28|18:09:51.626] Maximum peer count          ETH=50 LES=0 total=50
INFO [11-28|18:09:51.642] Set global gas cap          cap=50,000,000
INFO [11-28|18:09:51.645] Using LevelDB as the backing database
INFO [11-28|18:09:51.645] Allocated cache and file handles database=D:\programs\Geth\data0\geth\chaindata cache=
16.00MiB handles=16
INFO [11-28|18:09:51.721] Using LevelDB as the backing database
INFO [11-28|18:09:51.730] Opened ancient database      database=D:\programs\Geth\data0\geth\chaindata\ancien
t/chain readOnly=false
Fatal: Failed to write genesis block: database contains incompatible genesis (have d4e56740f876aef8c010b86a40d5f56745a11
8d0906a34e69a9c8c0db1cb8fa3, new 2720038ef46044a7a895296b85745294340ecfcdcc32f8c9e9802129aeb62890)
```

图 2-2

再输入命令：`geth --http --nodiscover --datadir ".\data0" --port 30303 --http.api db,eth,net,web3 --http.corsdomain "*" --networkid 1001 --ipcdisable --allow-insecure-unlock console 2>>geth.log --rpc.enableddeprecatedpersonal`

启动以太坊

```
D:\programs\Geth
8d0906a34e69a9c8c0db1cb8fa3, new 2720038ef46044a7a895296b85745294340ecfcdcc32f8c9e9802129aeb62890)
D:\programs\Geth> geth --http --nodiscover --datadir ".\data0" --port 30303 --http.api db,eth,net,web3 --http.corsdomain "*" --networkid 1001 --ipcdisable --allow-insecure-unlock console --rpc.enableddeprecatedpersonal
INFO [11-28|18:09:55.021] Maximum peer count          ETH=50 LES=0 total=50
INFO [11-28|18:09:55.034] Set global gas cap          cap=50,000,000
INFO [11-28|18:09:55.042] Allocated trie memory caches clean=154.00MiB dirty=256.00MiB
INFO [11-28|18:09:55.042] Using LevelDB as the backing database database=D:\programs\Geth\data0\geth\chaindata cache=
512.00MiB handles=8192
INFO [11-28|18:09:55.073] Using LevelDB as the backing database database=D:\programs\Geth\data0\geth\chaindata\ancien
t/chain readOnly=false
INFO [11-28|18:09:55.089] Disk storage enabled for ethash caches dir=D:\programs\Geth\data0\geth\ethash count=3
INFO [11-28|18:09:55.089] Disk storage enabled for ethash DAGs dir=C:\Users\KuJyo\AppData\Local\Ethash count=2
INFO [11-28|18:09:55.090] Initialising Ethereum protocol network=1001 dbversion=8
INFO [11-28|18:09:55.122] -----
INFO [11-28|18:09:55.122] Chain ID: 1 (mainnet)
INFO [11-28|18:09:55.122] Consensus: Beacon (proof-of-stake), merged from Ethash (proof-of-work)
INFO [11-28|18:09:55.122]
INFO [11-28|18:09:55.122] Pre-Merge hard forks (block based):
INFO [11-28|18:09:55.122] - Homestead: #1150000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homestead.md)
INFO [11-28|18:09:55.122] - DAO Fork: #1920000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/dao-fork.md)
INFO [11-28|18:09:55.122] - Tangerine Whistle (EIP 150): #2463000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tangerine-whistle.md)
INFO [11-28|18:09:55.123] - Spurious Dragon/1 (EIP 155): #2675000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon.md)
```

图 2-3

私有链搭建完成

2.2.2 任务 2

随后输入命令 `personal.newAccount()` 创建 3 个账户

```
D:\programs\Geth
Repeat passphrase:
INFO [11-28|18:12:12.511] Your new key was generated
WARN [11-28|18:12:12.511] Please backup your key file!
10-12-09.849209200Z--df10d4959be29ec06f92e06be8826f5d8d942942
WARN [11-28|18:12:12.511] Please remember your password!
"0xdf10d4959be29ec06f92e06be8826f5d8d942942"
> eth.accounts
["0xdf10d4959be29ec06f92e06be8826f5d8d942942"]
> personal.newAccount
function github.com/ethereum/go-ethereum/internal/jsre.MakeCallback.func1()
> personal.newAccount()
Passphrase:
Repeat passphrase:
INFO [11-28|18:13:03.586] Your new key was generated
WARN [11-28|18:13:03.586] Please backup your key file!
10-13-01.042983100Z--549a520f56759f646e40da86748bfcad11d1e4b1
WARN [11-28|18:13:03.586] Please remember your password!
"0x549a520f56759f646e40da86748bfcad11d1e4b1"
> personal.newAccount()
Passphrase:
Repeat passphrase:
INFO [11-28|18:13:11.500] Your new key was generated
WARN [11-28|18:13:11.505] Please backup your key file!
10-13-08.993147700Z--9c4c96e2917e1375d31df781901ec939d7ee636a
WARN [11-28|18:13:11.505] Please remember your password!
"0x9c4c96e2917e1375d31df781901ec939d7ee636a"
> eth.accounts
["0xdf10d4959be29ec06f92e06be8826f5d8d942942", "0x549a520f56759f646e40da86748bfcad11d1e4b1", "0x9c4c96e2917e1375d31df781901ec939d7ee636a"]
> |
```

图 2-4

然后让三个账户挖矿获取以太坊币，解锁账户 1 和账户 2，并设置第三个账户为矿工持续挖矿

随后编写合约连接 remix.

选择账户 1，部署 victim 合约

然后选择账户 2(攻击者)，部署 attack 合约

先用 victim 合约存钱 5 ETH，用攻击者合约攻击获得 ETH 6，如图:

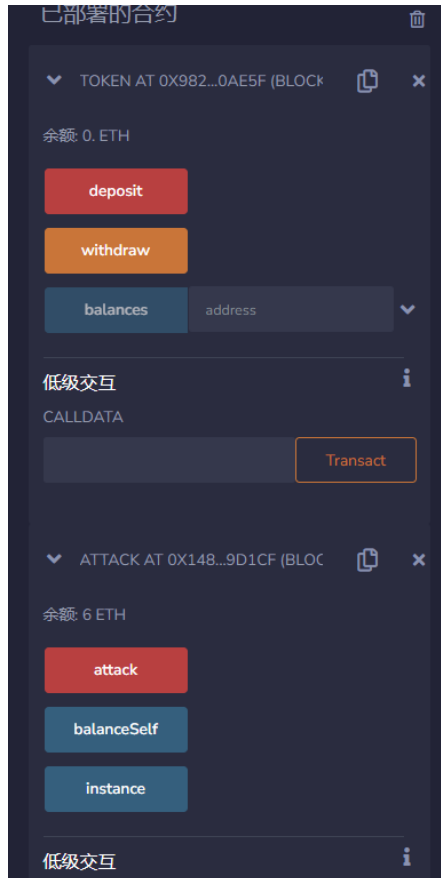


图 2-5

重入攻击成功!

2.3 实验中的问题

1. 部署合约时，发现合约一直是 pending 状态，查看发现是没有挖矿，于是创建第三个账户，设置 EtherBase 然后 miner.start()挖矿，顺便调整了 difficulty
2. personal 指令无法使用，查阅资料发现：启动时需要加上 --rpc.enabledprecatedpersonal 参数

2.4 实验总结及建议

在实验的第一部分，在 Geth 环境下成功地搭建了一个私有网络。这个过程中，我不仅掌握了以太坊的基本命令，也学习了如何配置和运行一个多节点的以太坊私有网络。

在实验的第二部分，我深入学习和实践了以太坊智能合约的编写和调用。值得一提的是，我们还了解了重入漏洞的基本原理，并成功地设计了合约代码来复现重入漏洞攻击。

在实验过程中最麻烦的是版本问题，还有一些合约启动参数不是特别清晰，所以做这个实验的时候踩了特别多的坑，建议实验手册能够指定相对应的版本，减少这类坑的出现。

此外这个实验中的 `geth` 有 `win` 的预构建文件安装包(在 `geth` 官网)，其实可以下载 `win` 的版本本地部署，考虑到绝大部分同学都是 `windows`，可以在实验手册上加上这个安装方法。