



去中心化信任机 测试指引

2021 年 6 月 6 日

深圳物信链技术有限公司

目录

去中心化信任机.....	3
测试指引.....	3
一、 概述.....	3
二、 信任机用途.....	3
三、 信任机网络操作说明.....	3
1. 环境搭建.....	3
(1) 下载信任机.....	3
(2) 操作说明.....	3
2. 操作命令.....	3
(1) 命令查看.....	3
(2) 账户操作.....	4
(3) 产品操作.....	5
3. 开发接口.....	6
(1) 注册接口.....	7
(2) 验证接口.....	7
四、 信任机应用案例流程图.....	7
五、 源码案例.....	8
1. 下载源码.....	8
2. 源码展示目标.....	8
3. 认证操作.....	8
(1) 通信程序登记注册.....	8
(2) 认证成功操作.....	9
(3) 修改代码后认证失败操作.....	10
4. 源码说明.....	10
(1) 客户端注册认证模块.....	10
(2) 服务端注册认证模块.....	11
(3) 认证模式.....	12
六、 后续.....	12

去中心化信任机

测试指引

一、概述

信任机是机器与机器实现绝对信任的一段程序和内置标准操作接口,通过这些接口,不同机器之间通信时候,可以知道对方一切,无法作假,而且代码公开情况下,无人能干预,比喻我通过修改信任机代码,预留漏洞,只要你修改一个字节,就没有机器会信任,马上被踢出,有人可能会想到在源码中增加伪装响应信息,那可能不会有任何机会,就是量子计算机都无法破解返回包,因为没有算法去计算。

二、信任机用途

1、物联网可信通信: 每个网络通信终端均配置有信任机节点,通信时候可以通过信任机节点查看通信对方版本是否匹配,是否有被病毒感染或伪装,彼此通信 可以采用常规通道通信也可以采用信任机加密通道通信,彼此可以通过信任机传送公钥和对称加密密钥;

2、零信任架构基础平台: 第三方开发商很容易通过二层开发,实现零信任架构的虚拟私密办公空间,所有数据和应用都在网上,但比私有网络还安全私密;

3、区块链应用: 区块链目前的算力挖矿就是一个笑话,很快将成为历史,目前区块链最大的矛盾是去中心化和处理速度此消彼长,想去中心化就要牺牲速度,有了信任机后,噩梦不复存在;

4、区块链的未来: 超比特币的去中心化和超过中心系统的处理速度,如果这绝对不可能,那信任机技术更加不可能;

三、信任机网络操作说明

1. 环境搭建

去中心化信任机测试网目前只提供 **linux 边缘节点**信任机下载,由于测试网带宽只有1Mb,因此认证过程需要几秒钟,请耐心等待。

(1) 下载信任机

<http://www.biotwsl.com/download/tmac.tar>

>tar -zxvf tmac.tar

展开后结果:

```
total 24884
-rwxr-xr-x. 1 root root 2076069 May 27 12:47 gethash.exe
-rw-r--r--. 1 root root 115 Jun 2 15:22 manage.sh
-rw-r--r--. 1 root root 114 Jun 2 15:22 start.sh
-rwxr-xr-x. 1 root root 23391584 Jun 2 18:32 tmac.exe
-rw-r--r--. 1 root root 130 Jun 2 19:22 v001.ver
[root@localhost biot]#
```

Manage.sh 管理操作模式

Start.sh 长期后台运行模式

模式可以自由切换,但同一时间只能以一种模式运行

(2) 操作说明

由于信任机网络是去中心化的,您可以在任一台 linux 机器上运行信任机,都可以操作相同的数据镜像。

2. 操作命令

(1) 命令查看

>sh manage.sh

```

[root@localhost biot]# sh manage.sh
mkdir: cannot create directory '/biot_data': File exists
mkdir: cannot create directory '/biot_data/version': File exists
publickey: 9cb88d772aac2d895763d2b72e6cf931a7c5dda6e85d6c5e7d67f1e175a9bb4c1cf2f78eba66de854cdc6ca7f06fd4f6f5df92b
56f
TMac is running.....

> The node is being authenticated. Please wait.....
AuthSuccessful from 39.108.150.165

```

启动信任机后，需要等待信任机接入信任机网络，这中间会有一个验证过程，需要耐心等待，当出现 AuthSuccessful 时，说明已成功接入信任机网络，后面的 IP 就是该边缘节点的信任机网络见证人。

>cmd ?

查看有哪些命令

```

> The node is being authenticated. Please wait.....
AuthSuccessful from 39.108.150.165
> cmd ?
Usage:
    account ?
    product ?
    exit

```

查看账户操作 > account ?

查看产品操作 >product ?

退出系统 >exit

(2) 账户操作

使用信任机网络，首先要创建自己的账户，然后可以在该账户下增加自己的产品

> account ?

```

> account ?
Usage:
    account add      -uusername      -ppassword
    account show
    account logon    -uusername      -ppassword
    account logout   -uusername
    account del       -uusername
    account chpassword -uusername      -ppassword      -ppassword

```

增加账户:

如增加一个名为 company 密码为 @WSX3edc 的账户

> account add -ucompany -p@WSX3edc

```

> account add -ucompany -p@WSX3edc
Create account successful!

```

提示增加成功

查看账户:

>account show

```

> account show
Account:
    hu1
    hu2
    hu3
    hu7
    hu8
    hu9
    hu10
    company

```

该操作只会展示在当前节点上增加的账户，其他节点增加的不可见。

登录账户:

```
>account logon -ucompany -p@WSX3edc
```

```
> account logon -ucompany -p@WSX3edc
Account company logon Successful!
>
```

在做产品管理时候，必须先要登录账户，然后才可以对当前账户下的产品进行管理。

登出账户:

```
> account logout -ucompany
```

删除账户:

```
>account del -ucompany -p@WSX3edc
```

删除账户时，如该账户下还有产品，是无法删除成功的，需先要删除所有产品后，再删除账户。

修改密码:

```
>account chpassword -ucompany -p@WSX3edc -p@WSX3edc12345678
```

第一个-p 是旧密码，第二个-p 是新设置的密码

(3) 产品操作

有了账户后，就可以管理自己的产品，被管理的产品，通信时候可以验证该产品的哈希值，防止被别人篡改或伪装。

>product ?

产品操作，必须先登录账户

```
> product ?
Usage:
    product show
    product show -pproduct
    product show -pproduct -vversion
    product add -nname -vversion -fdirname&file1,file2
    product del -nname
    product del -nname -vversion
>
```

增加产品:

```
>product add -ntest -v001 -f/home&example.exe
```

```
> product add -ntest -v001 -f/home&example.exe
Add product test successful!
>
```

增加文件时候，可以增加多个，逗号分隔，也可以加入当前目录所有文件 如：*，系统会把指定文件的内容哈希值保存起来。

- n 产品名
- v 版本号
- f 需要登记的文件

查看产品:

```
>product show
```

```
> product show
Product:
    test
>
```

查看一个产品:

```
>product show -ntest
```

```
Product:
  Username:    company
  Name:        test
  Createtime:  2021-06-03 08:51:28
  Versioin:    001
>
```

查看一个产品的一个版本:

```
>product show -ntest -v001
```

```
Product:
  Username:    company
  Name:        test
  Versioin:    001
  Files:
    example.exe: dc8184eb01b0da62d8059964
>
```

删除产品:

```
>product del -ntest
```

```
> product del -ntest
Product version record exists, please delete the version before operation!
>
```

删除产品时, 需要先删除该产品下的所有版本

删除产品的一个版本:

```
>product del -ntest -v001
```

```
> product del -ntest -v001
Del Product version 001 Successful!
> product del -ntest
Del Producttest Successful!
> product show
Product:
>
```

3. 开发接口

开发提供了两个接口:

代码	名称	说明
OpCode	"opcode"	主操作代码
SubCode	"subcode"	子操作代码
OpCode_Register	"Regsiter"	注册操作
SubCode_GetNonce	"GetNonce"	获取注册码
SubCode_AuthNonce	"AuthNonce"	认证注册码
D_Username	"username"	账户名
D_Productname	"pname"	产品名
D_PVername	"Pvername"	版本号
D_Nonce	"nonce"	注册码
D_Content	"content"	备注, 注册时候可以传递密钥
D_EncodeId	"enodeid"	注册码生成节点 ID

D_Ret	"ret"	返回值 Retval_Successful =0 //成功 Retval_Fail =1 //失败 Retval_IdTextErr =2 //id 字符编码错误
-------	-------	--

（1）注册接口

请求：

http://localhost:8080?OpCode=OpCode_Register&SubCode=SubCode_GetNonce&D_Username=company&D_Productname=test&D_PVername=001&D_Content=

返回值：

D_Ret=Retval_Successful&D_Nonce=&D_Content=&D_EncodeId=

接口原理说明：

注册时候需要提供用户名，产品名，版本号 and 需要交换的备注内容，发送到本地信任机，信任机会根据提交的产品版本，进行匹配，如果发现产品版本内容哈希值不匹配或者不存在就返回注册失败，否则发给自己的见证者，生成唯一注册码和见证者 ID 成功返回，见证者机器生成的注册信息会保留 30 秒，超过会自动清除。

（2）验证接口

http://localhost:8080?OpCode=OpCode_Register&SubCode=SubCode_AuthNonce&D_Nonce=&D_Content=&D_EncodeId=

返回值：

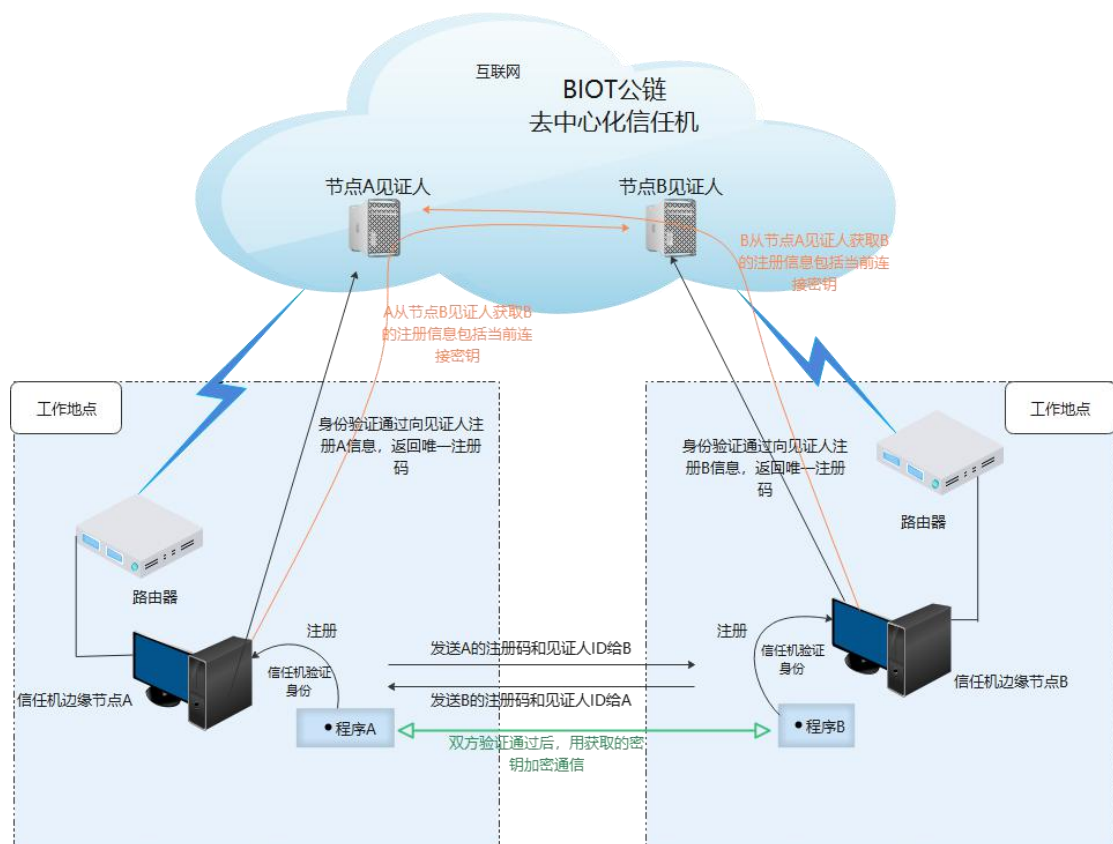
D_Ret=Retval_Successful&D_Content=

接口原理说明：

通过注册码和生成注册码的 ID，发给本地信任机，由该信任机通过信任机网络获取指定机器上的注册信息，如果获取失败，则验证不成功。

四、信任机应用案例流程图

通信程序身份验证和密钥交换应用案例一



五、源码案例

1. 下载源码

<http://www.biotwxl.com/download/example.tar>

下载后展开

```
root@localhost:~# ls -la
-rw-r--r-- 1 root root 345 May 24 20:43 authExample.iml
-rw-r--r-- 1 root root 33 Jun 2 13:43 go.mod
drwxr-xr-x 2 root root 130 Jun 2 13:45 tmac
[root@localhost authExample]#
```

2. 源码展示目标

- (1) 通信程序登记注册
- (2) 通信时候验证程序的完整性，如有被修改将无法连接通信
- (3) 可以通过信任机交换公钥或对称加密密钥

3. 认证操作

- (1) 通信程序登记注册

先编译程序：`go build -o example.exe authExample/tmac`

登录信任机增加产品：

```

[root@localhost biot]# sh manage.sh
mkdir: cannot create directory '/biot_data': File exists
mkdir: cannot create directory '/biot_data/version': File exists
publickey: 64ccf2421f2e4cdc8a1ad6ecd8080585e9758e55c9bbdcc38fbd818a2012ce434e2ecb6d5fb123363b8fed0f21879f5d2a82fb2375eb
TMac is running.....

> The node is being authenticated. Please wait.....
AuthSuccessful from 39.108.61.26
> account logon -ucompany -p@WSX3edc
Account company logon Successful!
> product add -nprodu -v101 -f/home/test/authExample&example
open file failed!, err: open /home/test/authExample/example: no such file or directory
Add product produ successful!
> product add -nprodu -v101 -f/home/test/authExample&example.exe
Add product produ successful!
> product show -nprodu -v101
Product:
  Username:    company
  Name:        produ
  Versioin:    101
  Files:
    example.exe: 5dea6c218e24d1a55d959a8f
>

```

第 1 步: 登录

第 2 步: 增加产品

第 3 步: 查看产品

(2) 认证成功操作

第一步, 在两台机器上同时启动信任机

```

1 biot01  2 tmac_01  3 tmac_02  4 tmac_02
[root@localhost biot]# cd /home/test/authExample/
[root@localhost authExample]# scp example.exe root@192.168.0.203:/home
root@192.168.0.203's password:
example.exe
100% 624
[root@localhost authExample]# cd /biot
[root@localhost biot]# sh manage.sh
mkdir: cannot create directory '/biot_data': File exists
mkdir: cannot create directory '/biot_data/version': File exists
publickey: 64ccf2421f2e4cdc8a1ad6ecd8080585e9758e55c9bbdcc38fbd818a2012ce434e2ecb6d5fb123363b8fed0f21879f5d2a82fb2375eb
TMac is running.....

> The node is being authenticated. Please wait.....
AuthSuccessful from 39.108.61.26

1 biot01  2 tmac_01  3 tmac_02  4 tmac_02
[root@localhost biot]# sh manage.sh
mkdir: cannot create directory '/biot_data': File exists
mkdir: cannot create directory '/biot_data/version': File exists
publickey: 9cb88d772aac2d895763d2b72e6cfd931a7c5dda6e85d6c5e7d67f1e175a9bb4c1cf2f78eba66de854cdc56f
TMac is running.....

> The node is being authenticated. Please wait.....
20.000841849s
The node is being authenticated. Please wait.....
AuthSuccessful from 47.106.87.200

```

第二步: 一台做客户端, 一台做服务器

分别启动好信任机后, 就可以启动客户端和服务端程序:

启动客户端: `/home/example.exe -ucompany -pprodu -v101 -h192.168.0.205`

注意: 启动命令必须写完整路径

```
[root@localhost home]# /home/example.exe -ucompany -pprodu -v101 -h192.168.0.205
Register successful!
Client Auth Successful! Recv Server Info 0
hello
send 6 byte
```

启动服务端: /home/test/authExample/example.exe -ucompany -pprodu -v101

注意: 启动命令必须写完整路径

```
1 tmac_01 x 2 tmac_01 x 3 tmac_02 x 4 tmac_02 x +
[root@localhost biot]# /home/test/authexample/example.exe -ucompany -pprodu -v101
-bash: /home/test/authexample/example.exe: No such file or directory
[root@localhost biot]# cd /home/test/authExample/
[root@localhost authExample]# pwd
/home/test/authExample
[root@localhost authExample]# /home/test/authExample/example.exe -ucompany -pprodu -v101
Confirm that the client authentication is successful ! 0
Server Register successful!
hello
```

第三步: 认证成功正常发生接收数据

提示注册认证成功, 同时发生了"hello", 服务端也收到了"hello"

(3) 修改代码后认证失败操作

```
line = strings.Trim(line, "\r\n")
if line == "exit" {
    fmt.Println("客户端退出")
    return
}
n, err := conn.Write([]byte(line + "\n"))
if err != nil {
    fmt.Println("conn.Write err=", err)
}
fmt.Printf("send %d byte\n", n)
buf := make([]byte, 1024)
n, err = conn.Read(buf)
if err != nil {
    fmt.Println("服务器read err=", err)
    return
}
// 掩藏一行代码, 再编译
fmt.Print(string(buf[:n]))
}
func ClientAuth(conn net.Conn) error{
    //先要向信任机注册, 信任机会用以前登记的程序内容HASH与当前程序匹配, 如果当前程序内容
    id, nonce, err := Register()
    if err != nil {
        fmt.Println("Register fail!", err.Error())
        return err
    }
}
```

掩藏一行代码, 再编译

```
[root@localhost home]# /home/example.exe -ucompany -pprodu -v101 -h192.168.0.205
Register fail! Find item Fail!
[root@localhost home]#
```

4. 源码说明

(1) 客户端注册认证模块

注册认证分三步:

第一步: 先要向信任机注册, 信任机会用以前登记的程序内容 HASH 与当前程序匹配, 如果当前程序内容有改动, 是无法注册成功的

第二步: 注册成功, 把注册信息发送到服务端, 由服务端向信任机确认是否注册成功

第三步: 最后从信任机网络读取服务的注册信息, 确保服务端也是注册成功的

```
func ClientAuth(conn net.Conn) error{
    //先要向信任机注册, 信任机会用以前登记的程序内容HASH与当前程序匹配, 如果当前程序内容有改动, 是无法注册成功的
    id,nonce,err:=Register()
    if err!=nil{
        fmt.Println("Register fail!",err.Error())
        return err
    }
    fmt.Println("Register successful!")

    //注册成功, 把注册信息发送到服务端, 由服务端向信任机确认是否注册成功
    req:=NewRequest("", "")
    req.AddItem(D_EncodeId,id)
    req.AddItem(D_Nonce,nonce)
    _, err= conn.Write([]byte(req.ToString()))
    if err != nil {
        fmt.Println("conn.Write err=", err)
    }
    buf := make([]byte, 1024)
    n:=0
    n, err = conn.Read(buf)
    if err != nil || n<1{
        fmt.Println("Read data from server fail!", err)
        return err
    }
    //最后从信任机网络读取服务的注册信息, 确保服务端也是注册成功的
    rq:=NewRequest("", string(buf[:n]))
    id,err=rq.GetItem(D_EncodeId)
    if err!=nil{return err}
    nonce,err=rq.GetItem(D_Nonce)
    if err!=nil{return err}

    var result string
    result,err=AuthNonce(id,nonce)
    if err!=nil{return err}
    fmt.Println("Client Auth Successful! Recv Server Info",result)
    return nil
}
```

(2) 服务端注册认证模块

注册认证也分三步:

第一步: 先读取客户端注册信息

第二步: 自身也要向信任机网络注册, 让客户端进行确认

第三步: 把注册信息发给客户端, 让客户端确认

```
func ServerAuth(conn net.Conn)error {
    //先读取客户端注册信息
    buf := make([]byte, 1024)
    n, err := conn.Read(buf)
    if err != nil || n<1 {
        fmt.Println("Server read err=", err)
        return err
    }
    var id,nonce string
    req:=NewRequest("",string(buf[:n]))
    id,err=req.GetItem(D_EncodeId)
    if err!=nil{return err}
    nonce,err=req.GetItem(D_Nonce)
    if err!=nil{return err}
    var result string
    //向信任机网络进行确认
    result,err=AuthNonce(id,nonce)
    if err!=nil{return err}
    fmt.Println("Confirm that the client authentication is successful ! ",result)

    //自身也要向信任机网络注册, 让客户端进行确认
    id,nonce,err=Register()
    if err!=nil{
        fmt.Println("Server Register fail!",err.Error())
        return err
    }
    fmt.Println("Server Register successful!")
    //把注册信息发给客户端, 让客户端确认
    req=NewRequest("", "")
    req.AddItem(D_EncodeId,id)
    req.AddItem(D_Nonce,nonce)
    _, err= conn.Write([]byte(req.ToString() + "\n"))
    if err != nil {
        fmt.Println("conn.Write err=", err)
        return err
    }
    return nil
}
```

(3) 认证模式

默认需要双方认证成功, 连接才能建立, 当然这都是由开发者自己定义, 可以灵活管理。

六、后续

目前测试网络由于配置非常低, 带宽也只 1Mb, 只做技术验证用, 年底会上正式网提供正式服务, 公网版本暂时可以免费接入, 如果需要私有化部署, 需要一定费用, 具体情况可以联系公司商务部门, 技术验证过程中如有疑问随时跟我们联系, QQ: 649387564 邮箱: hu119_3@163.com