# 安全文件传输

## 需求：



## 方案一：文件传输软件

对于文件的安全传输来讲如果实现一个安全传输的服务端和两个客户端。校园网客户端只写，财政端只读。系统的基本结构如下：



需要实现的功能有：

* 用户界面设计
* 用户的加密认证
* 加密传输
* 文件加密解密（选做？）
* 文件的管理

中间服务器采用https处理一般请求，采用??文件传输协议处理文件请求。

与https的交互：

会话就用token认证：<https://www.cnblogs.com/vovlie/p/4182814.html>

文件传输采用SSL，自写。

登陆：

|  |  |
| --- | --- |
| url | login |
| Data | {“username”:”vergilzhang”, “password”:”cv1314522”} |
| return | {“error”:0,”token”:”dsa89fhoa”}  {“error”:1,”err\_msg”:”wrong password”} |
|  |  |

注销：

|  |  |
| --- | --- |
| url | logout |
| Data | {“username”:”vergilzhang”, “token”:” dsa89fhoa”} |
| return | {“error”:0,”session”:”dsa89fhoa”}  {“error”:1,”err\_msg”:”not login”} |
|  |  |

修改密码：

|  |  |
| --- | --- |
| url | passwd |
| Data | {“username”:”vergilzhang”,  “token”:” dsa89fhoa”, “newpasswd”:”1314522”} |
| return | {“error”:0,”session”:”dsa89fhoa”}  {“error”:1,”err\_msg”:”not login”} |
|  |  |

获取文件列表：

|  |  |
| --- | --- |
| url | File |
| Data | { “token”:” dsa89fhoa”, “command”:”ls”} |
| return | {“error”:0,”session”:”dsa89fhoa”}  {“error”:1,”err\_msg”:”not login”} |
|  |  |

上传文件：

|  |  |
| --- | --- |
| url | File |
| Data | { “token”:” dsa89fhoa”, “command”:”upload”} |
| return | {“error”:0,”session”:”dsa89fhoa”}  {“error”:1,”err\_msg”:”not login”} |
|  |  |

文件传输特性：

V1 简单使用ssl做文件传输服务器

V2 支持断点续传

文件传输提供so库完成,包括服务端以及客户端.

基本设计:

可执行文件接口设计

./zuolinServer -p 10086 -u vergil -t token -d dir

./zuolinClient -p 10086 -u vergil -t token -f file -d dir

so库设计:

zuolinServer:

不面向对象,采用C语言设计模式,直接提供函数.

#define XC\_EVENT\_SYSTEM "event.system"

#define XC\_EVENT\_STREAM "event.stream"

typedef int (\*secft\_func\_pt)(void \*user\_data, void \*data);

bool secft\_strat\_up(int port, string user\_name, string token, string wording\_dir);

void secft\_shut\_down();

string secft\_get\_property(const char\* prop);

int secft\_set\_property(const char\* prop, const char\* value);

string secft\_version();

int secft\_handle\_event(const char \*event\_path, xcast\_func\_pt handler, void \*user\_data);

zuolinClient:

不面向对象,采用C语言设计模式,直接提供函数.

对于客户端来说,应该维护一个文件列表.提供进度,包括是上传还是下载.

struct item{

Path:

Md5:

Offset:

Stream\_type:up | down

items

}

文件传输协议设计:

网络库采用asio库.

packet[file\_name|md5|size|offset|data]

## 方案二：网络文件系统

自定义网络文件系统，在这个文件系统中，校园网用户访问时只写，财政端访问时只读。

表 1 网络文件系统对比

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *加密传输* | *加密认证* | *Posix语义.* | *ACL控制* | *有实际应用* |
| *NFSv3+IPSec* | ✓ | 🗶 | ✓ | 🗶 | ✓ |
| *NFS4+IPSec+Krb* | ✓ | ✓ | ✓ | ✓ | 🗶 |
| *CIFS* | 🗶 | ✓ | ✓ | ✓ | ✓ |
| *CIFS+IPSec* | ✓ | ✓ | ✓ | ✓ | 🗶 |
| *SSHFS* | ✓ | ✓ | ✓ | ✓ | ✓ |
| *AFP+NetATalk* | 🗶 | ✓ | ✓ | ✓ | 🗶 |

如果实现文件系统，则考虑依照SSHFS来实现，基于FUSE实现，代码地址：

<https://github.com/libfuse/sshfs>

如果实现网络文件系统：

* 仿照SSHFS
* 基本的用户界面，仅提供挂载启动的界面，文件管理等功能可以利用文件系统本身的功能

## 对比

* 工作量基本都需要两周。
* 实现文件系统不需要实现用户界面，因为可以调用系统自己的用户界面。
* 加密方式可以都采用SSL
* 文件系统的实现难度较大
* 文件系统的实现方式可以使需要分享的文件没有冗余。