

信息安全原理课程设计报告

2151140 王谦 信息安全

信息安全原理课程设计报告

- 一、摘要
- 二、问题及需求分析
 - 问题分析
 - 需求分析
 - 综合需求
 - 网络拓扑结构图示
- 三、功能及流程设计
 - 1.实验环境
 - 2.配置购买
 - 3.安装宝塔面板
 - 3.1 配置所需环境
 - 3.2安装宝塔面板
 - 3.3配置LAMP/LNMP
 - 4.部署网页
 - 5.部署MySQL数据库
 - 5.1在CentOS服务器上部署数据库
 - 5.2通过phpMyAdmin，对数据库进行管理
 - 6.配置DNS解析
 - 7.配置SSL加密
 - 8.在CentOS中设置SSH连接
 - 8.1安装OpenSSH服务
 - 8.2启动OpenSSH服务
 - 8.3设置开机自启动
 - 8.4配置防火墙规则
 - 8.5连接SSH
 - 9.安装OpenSSL、配置libssh2
 - 9.1 安装OpenSSL
 - 9.2 编译 libssh2
 - 9.3 运行libssh2代码文件
 - 10.编写C++代码
 - 11.通过C++代码和SSH命令，对服务器进行管理和配置
 - 11.1 查看SSH是否安装
 - 11.2 查看SSH配置文件
 - 11.5 查看已连接到系统的SSH会话
- 四、安全风险分析和安全策略设计
 - 安全风险分析
 - 安全策略设计
- 五、参考文献

考虑若干智能手机为用户终端，数据存储和web服务部署在一个linux服务器上，服务器与终端间应用层至少采用TLS保护，设计一个基于ssh对该服务器远程配置和管理的原型系统。

(1) 摘要；(2) 问题及需求分析；(3) 功能及流程设计；(4) 安全风险分析与安全策略设计；

一、摘要

1. 配置基础实验环境

使用CentOS 7.9云服务器+宝塔面板、使用154.64.254.120公网IP、使用www.tj-icarus.top 作为域名、在服务器配置LNMP (Nginx 1.22.1、MySQL 5.7.44、Pure-Ftpd 1.0.49、PHP 7.4.33、phpMyAdmin 5.1)

2. 部署web服务

在linux 服务器部署web服务 我们通过创建站点、配置DNS解析、添加html等web源代码文件，在linux服务器部署了web服务。

3. 配置TLS保护https连接

考虑若干智能手机为用户终端，服务器与终端间应用层至少采用TLS保护 我们假定用户终端为若干智能手机，通过配置SSL，并要求强制使用HTTPS安全连接，使得服务器与终端间应用层部署了SSL VPN，实现了对连接的保护。

4. 实现SSH远程管理系统

设计一个基于ssh对该服务器远程配置和管理的原型系统。开放linux服务器的ssh服务，通过OpenSSL、libssh2编写C++程序，实现了对服务器进行配置和管理的代码，得到了原型系统。

5. 正确性验证

我们验证了linux服务器提供的服务必须使用安全连接，验证了ssh服务的开放，并基于OpenSSL、libssh2 编写了多种配置和管理代码，检验了系统的正确性。网站开放至7月9日，在此之前均可访问。

6. 参与度和原创性

本次实验原创性极低，极大篇幅地参考了学长的设计<https://github.com/ChestnutSilver/Linux-SSH-System>，服务器的配置和代码方面也有很多参考。关于参与度，整体由我一人完成。

其中服务器来自[雨云\(rainyun.com\)](http://rainyun.com)

域名注册和SSL证书来自[阿里云-计算，为了无法计算的价值\(aliyun.com\)](http://aliyun.com)

二、问题及需求分析

问题分析

1. 问题背景：

- 智能手机作为用户终端，数据存储和Web服务器部署在一个Linux服务器上。
- 服务器与终端间的应用层至少采用TLS保护。
- 需要设计一个基于SSH的远程配置和管理系统。

2. 问题的主要方面：

- **数据传输安全：** 确保智能手机和服务端之间的数据传输安全。
- **服务器配置和安全：** 确保Linux服务器的安全配置和稳定运行。
- **远程管理：** 通过SSH对服务器进行远程管理和配置，确保管理通道的安全。
- **用户终端安全：** 确保智能手机作为用户终端的安全性。
- **网络结构和架构设计：** 确保系统的网络结构合理、可靠和安全。

需求分析

1. 系统需求：
- **LNMP环境：** 服务器需配置LNMP环境，包括Nginx 1.22, MySQL 5.7, Pure-Ftpd 1.0.49, PHP 7.4, phpMyAdmin 5.2。

◦ **域名和公网IP：** 智能手机需要访问Linux服务器提供的Web服务，需要配置域名和公网IP。

◦ **TLS保护：** 服务器和终端间的应用层需采用TLS保护，需申请和设置SSL证书，建立SSL隧道，实现安全传输。

◦ **远程管理：** 需使用OpenSSL、libssh2等，通过编写C++程序，实现对服务器的远程配置和管理。
2. 功能需求：
- **Web和数据库服务：** 提供Web服务和数据库访问。

◦ **用户访问：** 智能手机用户可以访问服务器提供的Web服务。

◦ **数据加密：** 数据在传输过程中进行加密保护。

◦ **远程配置和管理：** 通过SSH进行远程配置和管理。

综合需求

1. 系统环境配置：
- 配置LNMP环境，确保Web和数据库服务正常运行。

◦ 配置域名和公网IP，确保用户终端可以访问。
2. 数据传输和存储安全：
- 采用HTTPS协议，确保数据传输安全。

◦ 使用SSL隧道，保护内部通信。

◦ 数据库加密存储，确保数据在服务器端的安全。
3. 远程管理安全：
- 使用强密码和公钥认证机制，确保SSH连接的安全。

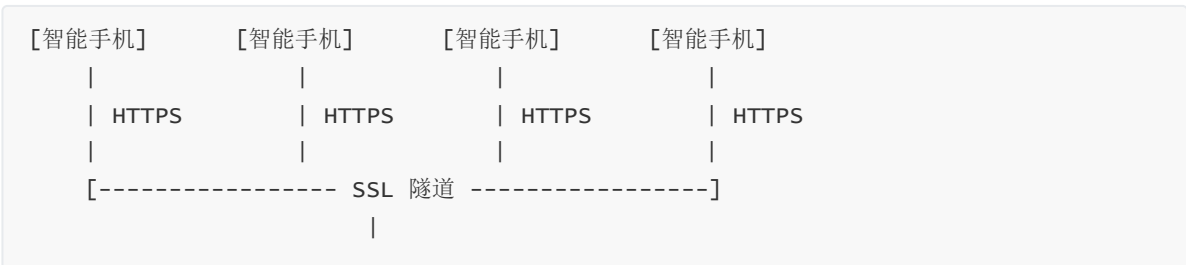
◦ 配置防火墙和入侵检测系统，保护服务器安全。

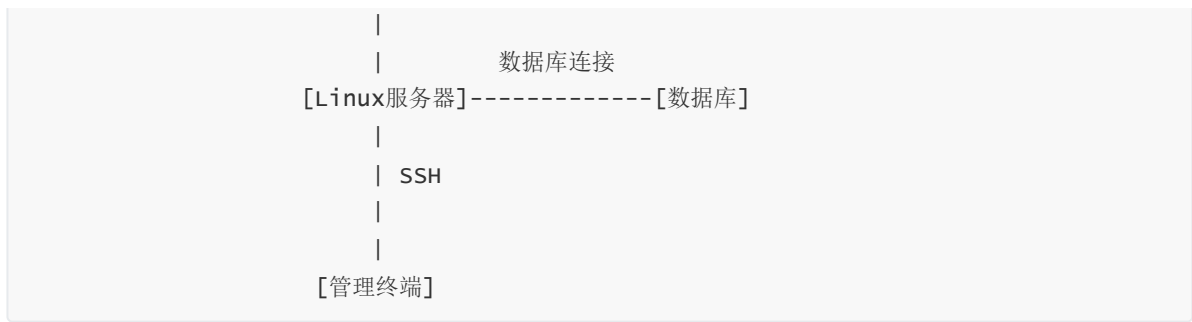
◦ 定期更新和备份，确保系统的稳定性和数据的可恢复性。
4. 用户终端安全：
- 确保用户终端的应用来源可信，并定期更新。

◦ 教育用户提高安全意识，防范钓鱼攻击和恶意软件。

网络拓扑结构图示

1. **用户终端：** 多个智能手机，通过HTTPS连接到系统。
2. **服务器：** 一个Linux服务器，提供Web和数据库服务。
3. **数据库：** 连接到Linux服务器的数据库。
4. **SSL隧道：** 在智能手机和Linux服务器之间建立SSL隧道，以确保HTTPS连接的安全性。
5. **互联网连接：** Linux服务器通过互联网与远程管理终端进行SSH连接。
6. **管理终端：** 通过SSH连接到Linux服务器进行远程管理。
7. **数据流动和通信：** 清晰标识智能手机、Linux服务器、数据库和管理终端之间的数据流动和通信路径。





三、功能及流程设计

1.实验环境

服务器：CentOS 7.9云服务器+宝塔面板

IP：使用154.64.254.120公网IP

域名：www.tj-icarus.top

LNMP：Nginx 1.22.1、MySQL 5.7.44、Pure-Ftpd 1.0.49、PHP 7.4.33、phpMyAdmin 5.1

网站开放至7月9日，在此之前均可访问。

2.配置购买

购买实验环境所需的服务器、公网IP、域名，使用免费SSL服务。

3.安装宝塔面板

3.1 配置所需环境

通常用端口和20250端口，可以直接通过文件导入规则。购买云服务器实例，使用CentOS 7.9系统，如下图所示：

配置安全组策略，要求能够实现数据库、web访问、https、ftp、ssh连接等功能，因此开放相应的端口，如下图所示：设置安全组的进站规则，其中，重点关注20250、3306、21、20、22、888、80、443、8888端口的设置。

配置出站规则，重点关注21、443、5880、80端口。

创建防火墙规则

是否启用

动作

允许

源地址(IPv4)

0.0.0.0

端口

20250,3306,21,20,22,888,80,443,5880,8888

协议

TCP

规则说明:

动作为允许相当于加入白名单,丢弃则相当于黑名单

除了直接写ip之外,源地址还支持CIDR/范围/逗号分割(不能混用)

如: 0.0.0.0/0(代表所有IP)或0.0.0.0-255.255.0.0或114.114.114.114,191.191.191.191

端口支持范围格式和逗号分割(范围和逗号不能混用)

如: 1000:2000(代表1到2千的端口)或22,80,443分割格式

案例: 若想让除了123.123.123.123能访问,其他IP都不能访问,则需添加2条规则(一黑一白),并把白名单规则拖到上方位置

确定

3.2安装宝塔面板

宝塔面板的作用是能够方便快捷地对linux系统进行操作。我的服务器在购买时默认安装了。

3.3配置LAMP/LNMP

选择配置LNMP。

LNMP的介绍如下图所示： 我们使用宝塔面板，安装LNMP环境。

安装完成，如下图所示：

最近使用入口 宝塔SSH终端

软件名称	开发商	说明	价格/天	到期时间	位置	状态	首页显示	操作
Nginx 1.22.1	官方	轻量级，占有内存少，并发能力强(可选Tengine/openresty)	免费	--				设置 卸载
MySQL 5.7.44	官方	MySQL是一种关系数据库管理系统(支持a11sq1/greatsq1/mariadb)，需要多版本共存请使用Docker应用【MySQL多版本管理】插件	免费	--				设置 卸载
PHP-7.4.33	官方	PHP是世界上最好的编程语言	免费	--				设置 卸载
Pure-Ftpd 1.0.49	官方	PureFTPd是一款专注于程序健壮和软件安全的免费FTP服务器软件	免费	--				设置 卸载
phpMyAdmin 5.1	官方	著名Web端MySQL管理工具	免费	--				设置 卸载
宝塔SSH终端 1.0	官方	完整功能的SSH客户端，仅用于连接本服务器	免费	--				设置 修复 卸载

4.部署网页

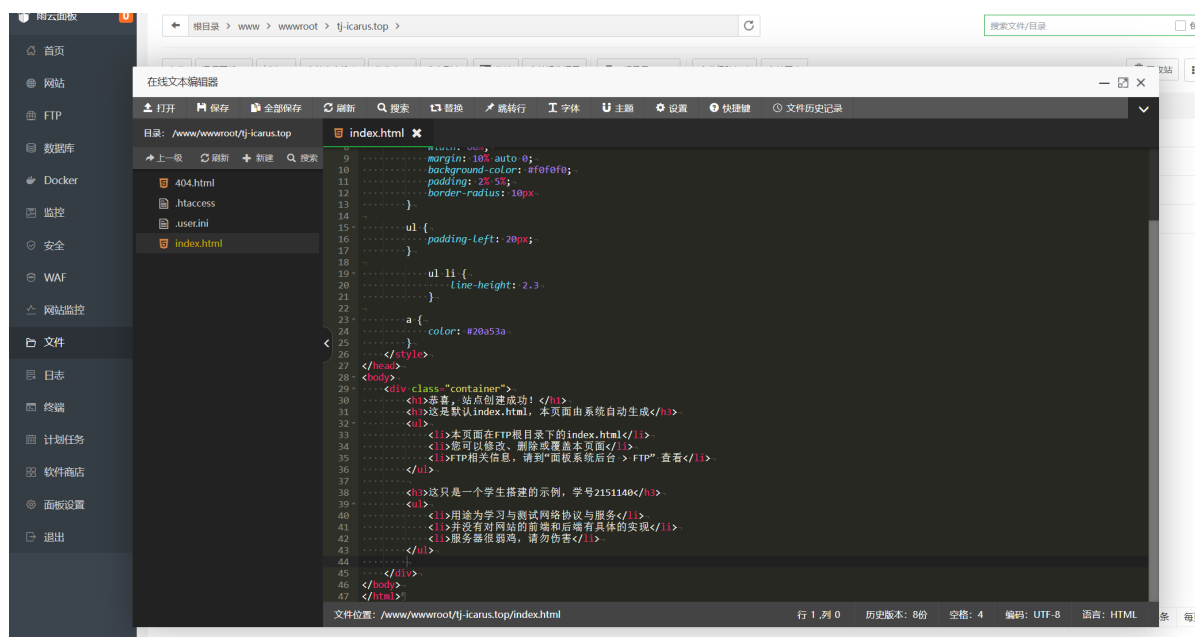
添加站点，使得web服务得以成功部署，如下图所示：



访问我们添加的站点，如下图所示：



我们修改index.html文件：

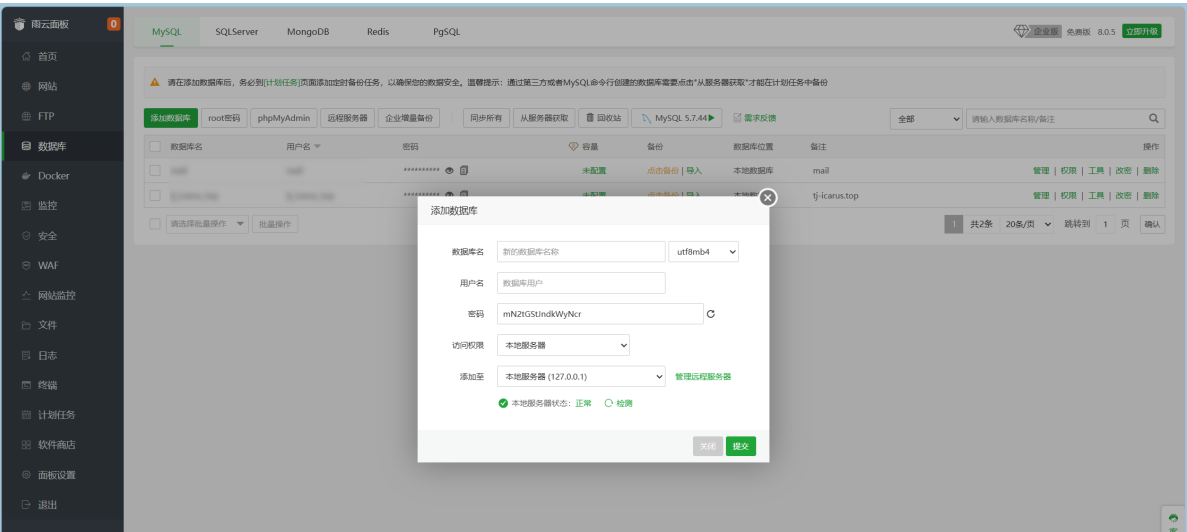


可以看到，多了我们刚刚添加的内容，说明这就是网站依赖的源代码文件：



5.部署MySQL数据库

5.1在CentOS服务器上部署数据库



5.2通过phpMyAdmin，对数据库进行管理

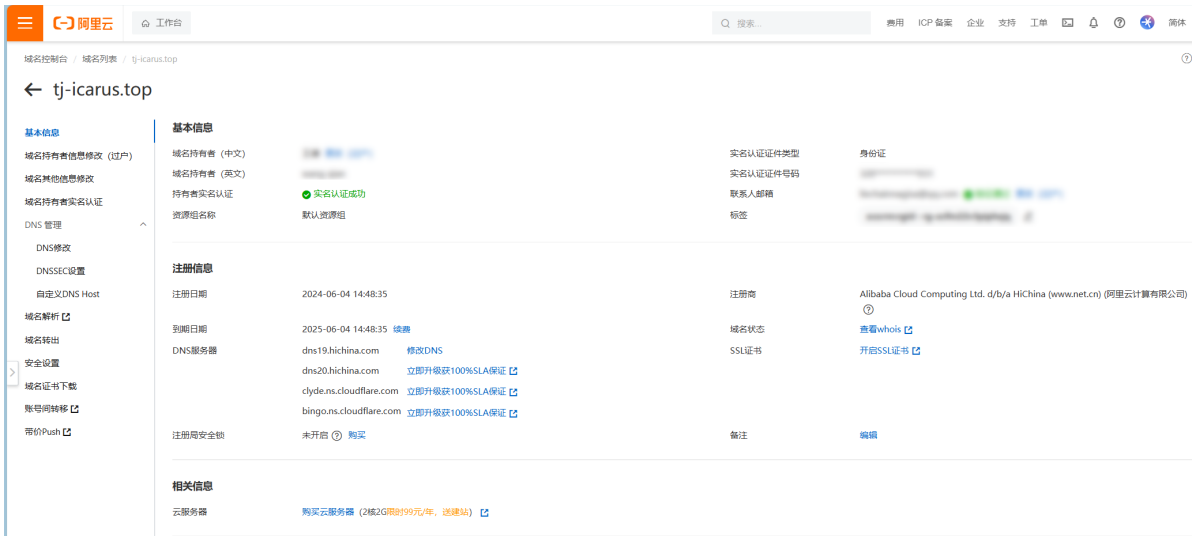


6.配置DNS解析

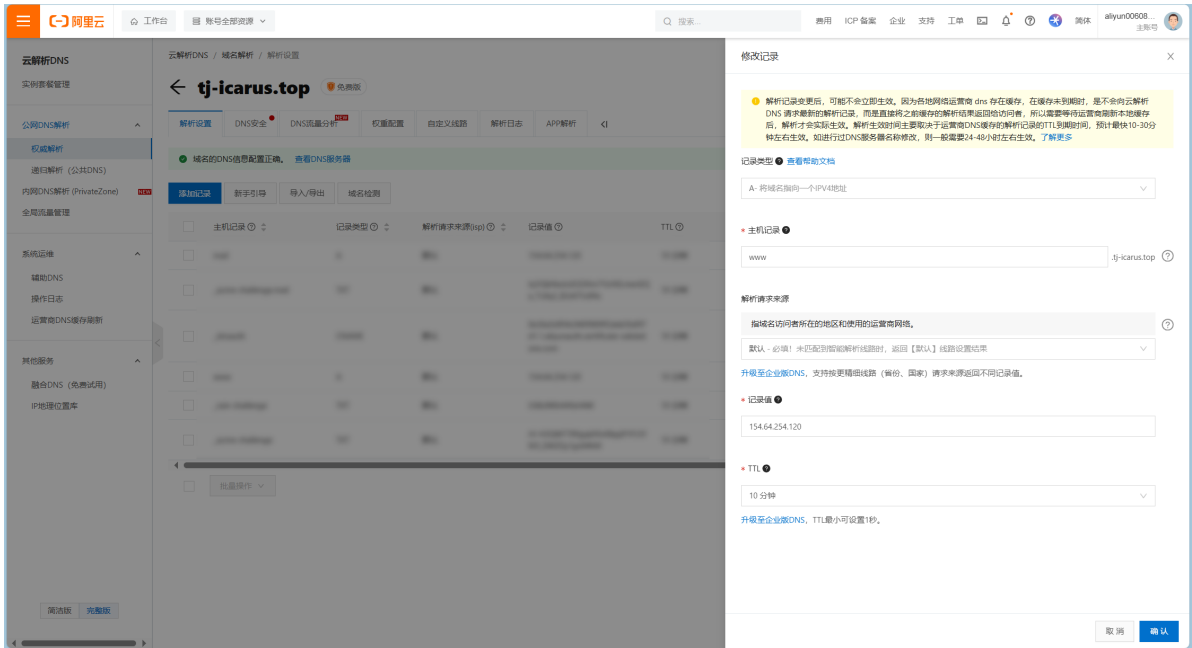
我们还应该使用一个域名，这样，智能手机用户就可以更方便地访问我们的网站，也更方便

域名需要先进行注册，这一过程可能需要等待3天左右：

我是从阿里云注册的域名tj-icarus.top，其他服务器提供商处也可以注册。



使用阿里云提供的免费DNS解析服务，为上面的域名添加DNS解析记录，指向我们的IPv4地址：



域名为: www.tj-icarus.top

需要注意的是，内地的服务器需要进行工信备案才能解析，而服务器至少需要3个月才能备案，因此我选择的是香港的服务器，阿里云的香港服务器有些贵，我用的是别的运营商。

通过cmd窗口，ping一下我们的域名，发现确实是154.64.254.120在响应，说明域名配置成功了：

```
C:\Windows\system32>ping www.tj-icarus.top

正在 Ping www.tj-icarus.top [154.64.254.120] 具有 32 字节的数据:
来自 154.64.254.120 的回复: 字节=32 时间=73ms TTL=42
来自 154.64.254.120 的回复: 字节=32 时间=55ms TTL=42
来自 154.64.254.120 的回复: 字节=32 时间=55ms TTL=42
来自 154.64.254.120 的回复: 字节=32 时间=55ms TTL=42

154.64.254.120 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 55ms, 最长 = 73ms, 平均 = 59ms
```

7.配置SSL加密

我们还要对域名申请SSL证书，来提升网站的安全性：

可以在注册域名的阿里云那里进行申请，我的服务器供应商不是阿里云，但是那里也可以提供免费SSL证书服务。



首先我们要下载下来证书，我们选择Nginx，因为之前配置的环境就是它：

rapid dv单域名SSL证书秒杀200元，每天限量10个，先到先得！

数字证书管理服务 / SSL 证书

SSL 证书

接到Digicert 关于中国区SSL证书价格调整通知，阿里云数字证书管理服务将于2024年6月10日起调整Digicert品牌下部分产品的价格，具体调整方案请参考[阿里云数字证书管理服务公告](#)。

证书管理 免费证书 上传证书 CSR管理 订单退款管理

查看免费证书变更公告

立即购买 创建证书 全部状态 请输入域名

证书	品牌/算法	状态	绑定域名
cert-11541246	DigiCert 免费版 SSL	已签发	tj-icarus.top
资源ID:cas-ivauto-xbaypf 标签:未设置标签			

证书下载

如果您需要技术专家提供1对1在线技术支持，进行配置、部署SSL证书，可点击购买“[部署服务](#)”。

更多信息，请参见[下载SSL证书到本地](#)



常用工具: [查看证书](#)

请根据您的服务器类型选择证书下载：

服务器类型	证书格式	操作
Nginx	pem/key	帮助 下载
Tomcat	pfx	帮助 下载
Apache	crt/key	帮助 下载
IIS	pfx	帮助 下载
JKS	jks	帮助 下载
其他	pem/key	下载
根证书下载	crt/cer	查看文档

下载到本地并解压，可以看到这些文件：

课程设计SSH > 13613756_tj-icarus.top_nginx

名称	修改日期	类型	大小
 tj-icarus.top.key	2024/6/9 16:34	KEY 文件	2 KB
 tj-icarus.top.pem	2024/6/9 16:34	PEM 文件	4 KB

我们通过宝塔面板，可以很方便地部署SSL证书，输入密钥和PEM格式的证书，强制HTTPS，如下图所示：

站点修改[tj-icarus.top] -- 添加时间[2024-06-09 15:54:09]

域名管理

子目录绑定

网站目录

访问限制

流量限制

伪静态

默认文档

配置文件

SSL

PHP

重定向

反向代理

防盗链

防篡改

安全扫描

网站日志

网站告警

其他设置

当前证书 - [已部署SSL]

商用SSL证书

测试证书

Let's Encrypt

证书夹

证书分类: 其他证书

认证域名: tj-icarus.top、www.tj-icarus.top

强制HTTPS: ☒

证书品牌: Encryption Everywhere DV TLS CA - ...

到期时间: 2024-09-06, 剩余75天到期

到期提醒: ☐ 到期提醒配置

密钥(KEY)

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAmFszHoTJrW/ud11mDEZD5xj
EKVezM4JyS19liqbsDoJkpgEv
YF+yOn0cPI7AQVLN4Pkrd3BOBNSnfW7/AxoLAV9
7l0Ebla0tgJilvXEj5i8FjNhj
RDL0Jf6x3p6m6dfVzL5iUoE4QsLYz4XCnn2lnz3Cc
gOobJVAantaxutoDHROyH2
zHrOiYGolFt4lWy9k454XLxc/zLDUeF8CBL4l2tFH5o
4pl0CvOED1sC8iEc97Y9x
WmchzyfY8Zs8NOVT4gneZM1Yhphpm0xfjiUGpk

证书(PEM格式)

-----BEGIN CERTIFICATE-----
MIIGBzCCBO+gAwIBAgIQB/MfsAUH3/K+0QwIXT
bdnjANBgkqhkiG9w0BAQsFADBu
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMR
GlnaUNlcnQsSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuYy29tMS0wKwYDVQQDEyRFb
mNyeXB0aW9uIEV2ZXJ5d2hlcmUg
RFYgVExTIEBIC0gRzlwHhcNMjQwNjA5MDAwM
DAwWhcNMjQwOTA2MjM1OTU5WjAY
MRYwFAYDVQQDEw10ai1pY2FydXMudG9wMIIBIj

保存

下载证书

关闭SSL

• 粘贴您的*.key以及*.pem内容, 然后保存即可[帮助]。

• 如果浏览器提示证书链不完整,请检查是否正确拼接PEM证书

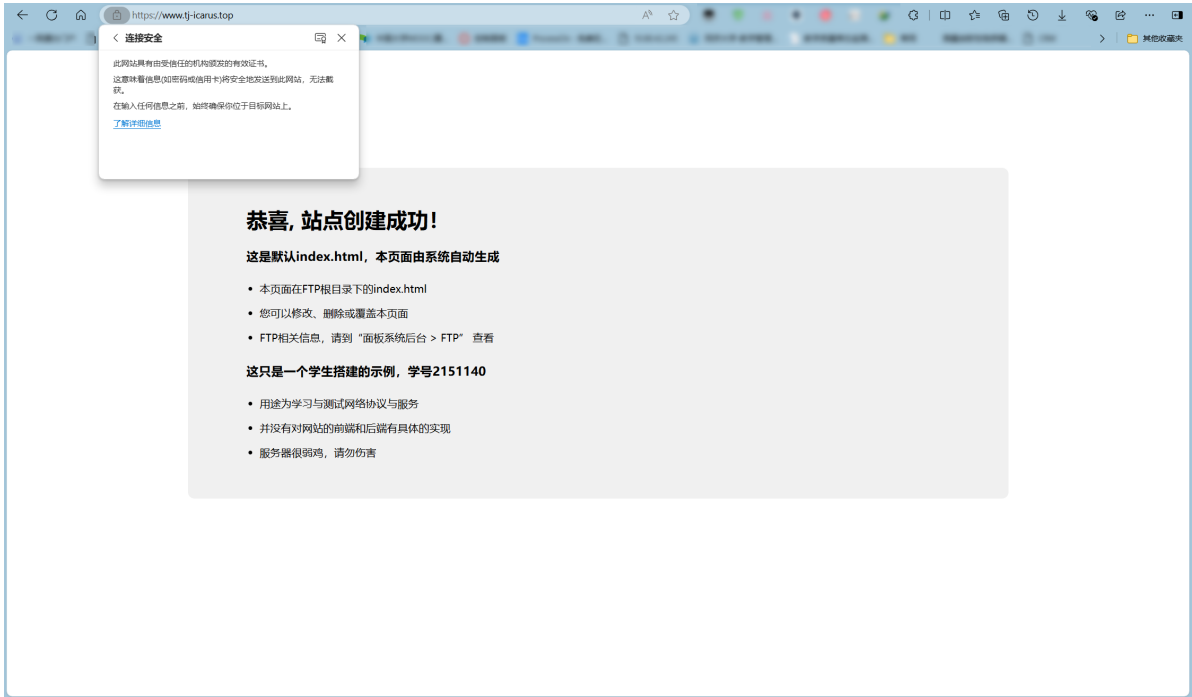
• PEM格式证书 = 域名证书.crt + 根证书(root_bundle).crt

• 在未指定SSL默认站点时,未开启SSL的站点使用HTTPS会直接访问到已开启SSL的站点

• 如开启后无法使用HTTPS访问, 请检查安全组是否正确放行443端口

我们在输入http的情况下，被强制修改为https了，并且建立了安全连接。

Edge和Chrome都表明，连接是安全的：





8.在CentOS中设置SSH连接

8.1安装OpenSSH服务

在终端中输入以下命令以安装OpenSSH服务:

```
sudo yum install openssh-server
```

```
[root@instance-ntiw4bu ~]# sudo yum install openssh-server
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
* base: mirrors.aliyun.com
* extras: mirrors.aliyun.com
* updates: mirrors.aliyun.com
Package openssh-server-7.4p1-23.el7_9.x86_64 already installed and latest version
Nothing to do
[root@instance-ntiw4bu ~]#
```

8.2启动OpenSSH服务

安装成功后, 执行以下命令启动OpenSSH服务:

```
sudo systemctl start sshd.service
```

8.3设置开机自启动

启动之后, 需要设置OpenSSH服务开机自启动, 以便系统重启后服务能自动恢复。执行以下命令设置开机自启动:

```
sudo systemctl enable sshd.service
```

```
Last login: Sun Jun  9 16:47:57 2024 from localhost
[root@instance-ntivw4bu ~]# sudo yum install openssh-server
-bash: sudo yum install openssh-server: command not found
[root@instance-ntivw4bu ~]# sudo yum install openssh-server
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
* base: mirrors.aliyun.com
* extras: mirrors.aliyun.com
* updates: mirrors.aliyun.com
Package openssh-server-7.4p1-23.el7_9.x86_64 already installed and latest version
Nothing to do
[root@instance-ntivw4bu ~]# sudo systemctl start sshd.service
[root@instance-ntivw4bu ~]# sudo systemctl enable sshd.service
```

8.4配置防火墙规则

如果系统有开启防火墙，必须配置防火墙规则以允许SSH连接。

8.5连接SSH

现在，可以使用SSH客户端连接到CentOS服务器了。在本地电脑终端中使用以下命令连接 SSH：

```
ssh username@remote_ip_address
```

其中，"username" 替换为 CentOS 服务器上的用户名，"remote_ip_address" 替换为 CentOS 服务器的 IP 地址。

```
C:\Users\ >ssh root@154.64.254.120
root@154.64.254.120's password:
Permission denied, please try again.
root@154.64.254.120's password:
Last failed login: Sun Jun  9 17:11:08 CST 2024 from 111.187.100.124 on ssh:ntty
There were 3 failed login attempts since the last successful login.
Last login: Sun Jun  9 16:49:38 2024 from localhost
[root@instance-ntivw4bu ~]#
```

9.安装OpenSSL、配置libssh2

9.1 安装OpenSSL

要编译libssh2，必须先编译好 OpenSSL 的静态库，直接从 <http://slproweb.com/products/Win32OpenSSL.html>

下载已经编译好的包含 lib 和 include 文件的安装包即可。

访问该网站点击下载完整的安装包，注意，不要下载 light 版本，因为 light 版本不带 lib 和 include。

"We would like to transfer a donation to Win32 OpenSSL, but only without Paypal. Can we transfer directly? Please send me the bank details with a quote." My bank account is 000012345 and my routing number is 000012345. I look forward to your donation. Mah...that's amazing! I've got the same combination on my legging!

Win32/Win64 OpenSSL Screenshot



Screenshot of the Win32/Win64 OpenSSL Installer.

Download Win32/Win64 OpenSSL

Download Win32/Win64 OpenSSL today using the links below

File	Type	Description
Win64 OpenSSL v3.1.1 Light EXE MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.1.1 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.1.1 EXE MSI	217MB Installer	Installs Win64 OpenSSL v3.1.1 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.1.1 Light EXE MSI	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.1.1 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.1.1 EXE MSI	175MB Installer	Installs Win32 OpenSSL v3.1.1 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.1.1 Light for ARM (EXPERIMENTAL) EXE MSI	6MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.1.1 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.1.1 for ARM (EXPERIMENTAL) EXE MSI	170MB Installer	Installs Win64 OpenSSL v3.1.1 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.2.2 Light EXE MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.2.2 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.2.2 EXE MSI	202MB Installer	Installs Win64 OpenSSL v3.2.2 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.2.2 Light EXE MSI	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.2.2 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.2.2 EXE MSI	163MB Installer	Installs Win32 OpenSSL v3.2.2 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.2.2 Light for ARM (EXPERIMENTAL) EXE MSI	6MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.2.2 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.2.2 for ARM (EXPERIMENTAL) EXE MSI	159MB Installer	Installs Win64 OpenSSL v3.2.2 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.1.6 Light EXE MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.1.6 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

9.2 编译 libssh2

从 <https://www.libssh2.org/> 下载 1.10.0 版本（而非1.11.0及以上版本）的源码包，解压出来后，用 VS2019 打开文件夹下的win32\libssh2.dsw。此时会提示升级项目，点击确定升级即可。但是VS2022及之后不再支持升级项目，这一点需要注意。

升级完成后，项目成功加载。因为我们要用 OpenSSL lib 的方式编译，且要 Release 版本的，所以我只用这个做演示，大家自己需要什么版本自己编译即可。首先双击 OpenSSL LIB Release | Win32，编辑该项目属性，在 C/C++-> 常规-> 附加包含目录 中，添加 OpenSSL 的 include 路径 D:\OpenSSL-Win32\include（根据实际进行设置）。

然后将运行库更改为多线程(/MT)

然后添加OpenSSL-Win32 安装目录下的 lib\VC 文件夹，并将 libcrypto32MT.lib 和 libssl32MT.lib 附加到依赖。

至此全部配置完毕了，编译一下项目，会生成libssh2.lib文件。

这里如果VS2022无法升级项目，也可以尝试去github中找别人已经编译好的，我自己就是这样操作的。

9.3 运行libssh2代码文件

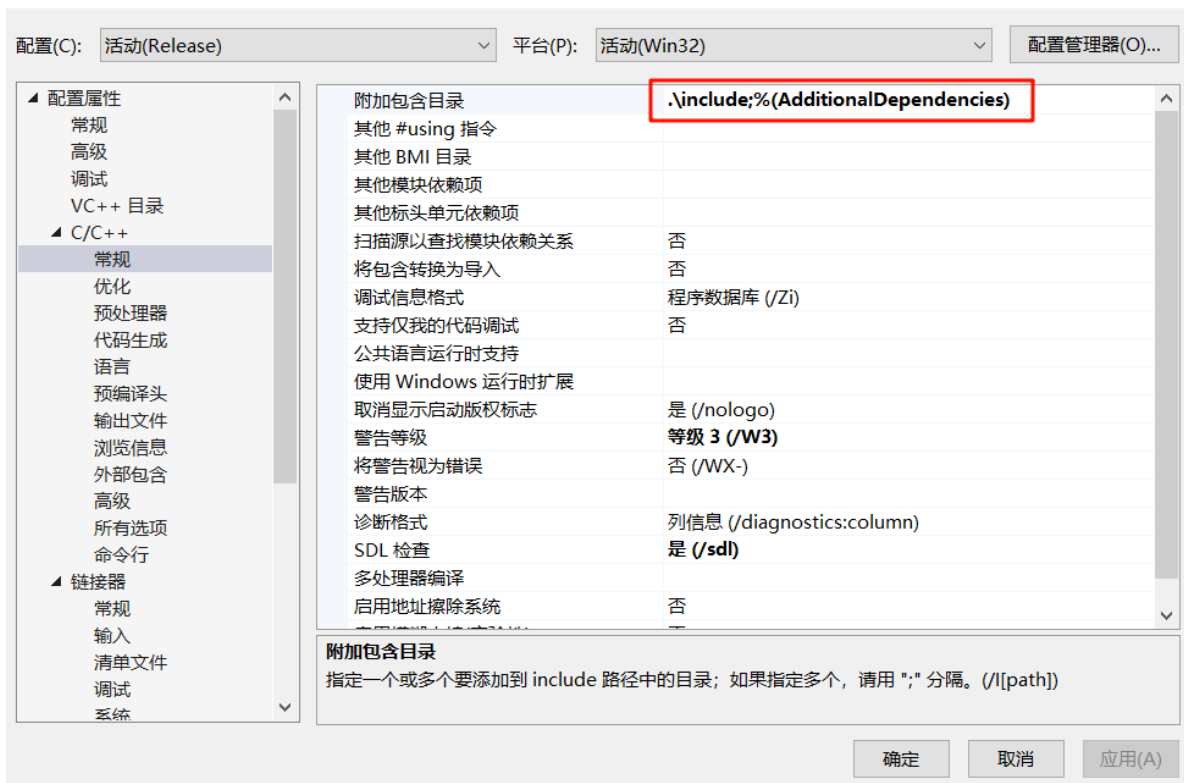
接下来我们新建一个 Win32 控制台的空项目，新建一个 cpp 文件：然后复制我们刚刚编译好的 libssh2 的库文件和所需的头文件。在新建的 Win32 项目目录下新建两个目录，一个叫 include，一个叫 lib。

(1) 复制我们刚才生成好的 libssh2-1.7.0\win32\Release_lib\libssh2.lib 到新建Win32项目的lib目录下。

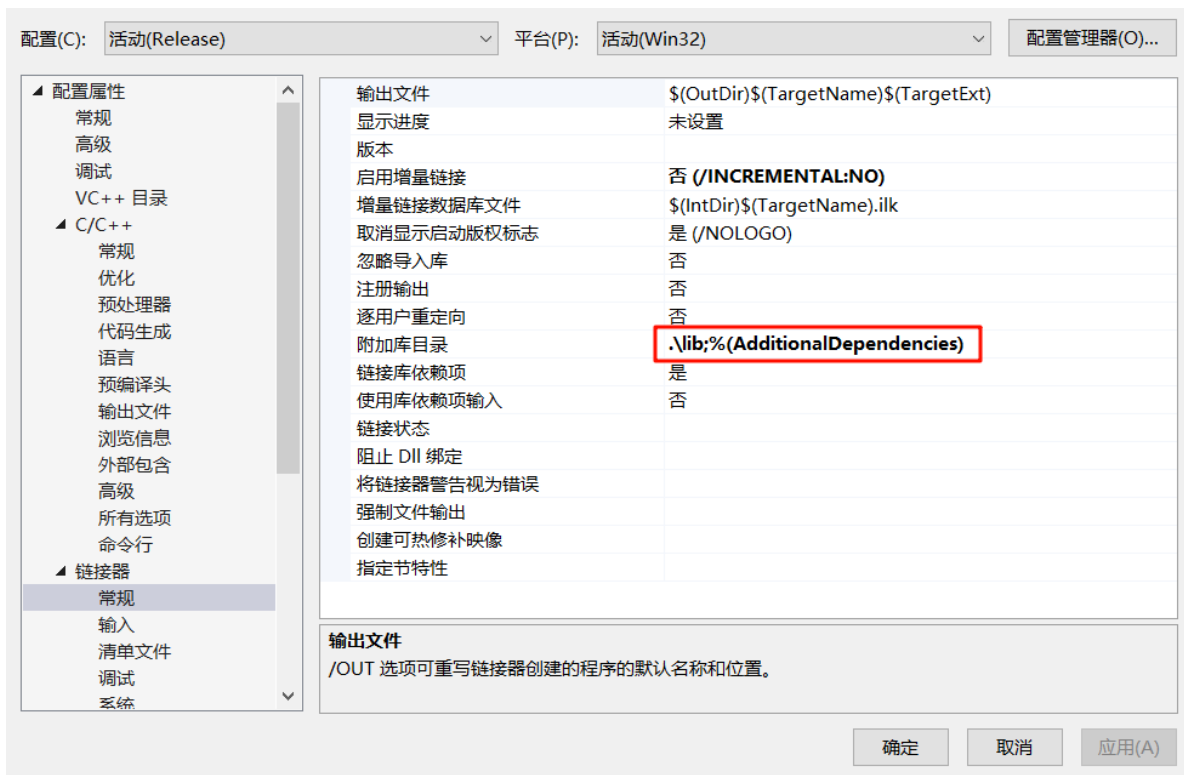
(2) 复制libssh2-1.7.0\include目录下所有文件到新建的Win32项目的include目录下。

(3) 复制libssh2-1.7.0\win32\libssh2_config.h文件到新建的Win32项目的include 目录下。复制完成后的目录结构如下：

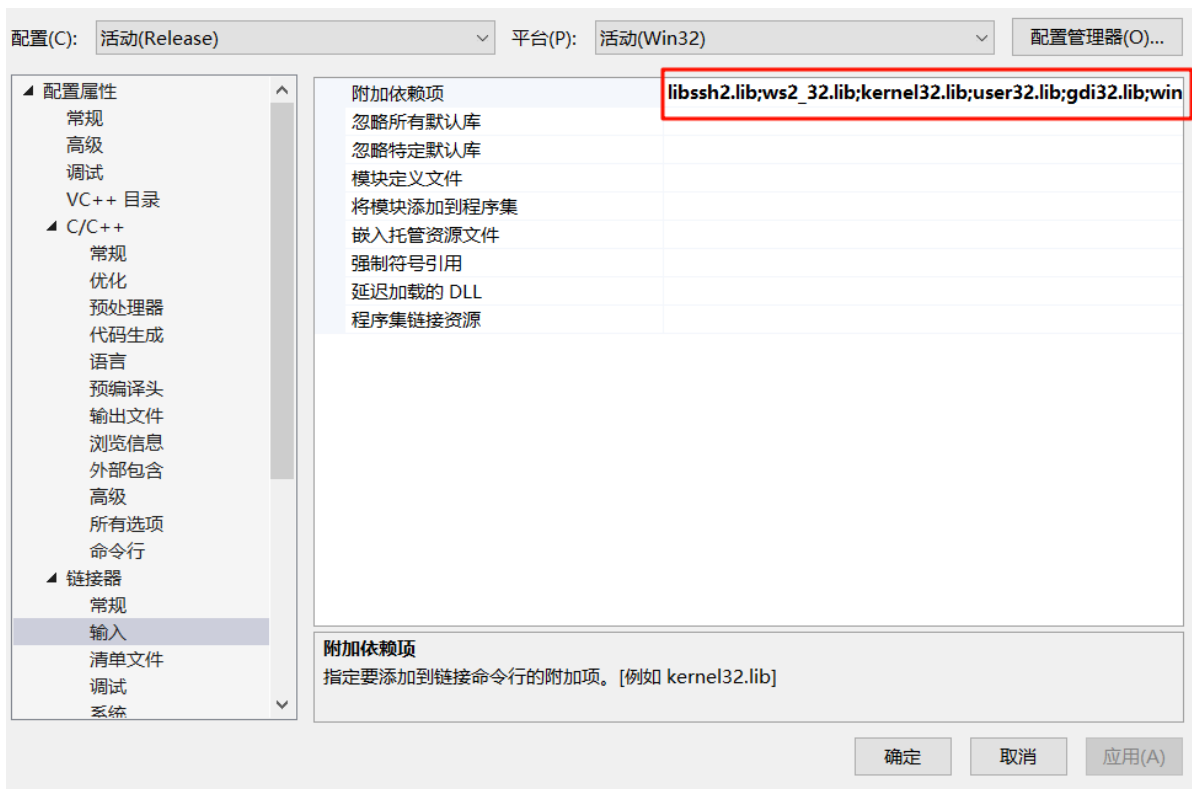
将测试项目的解决方案配置改为Release，将include目录添加到项目的“附加包含目录”中：



将 lib 目录添加到项目的“附加库目录”中：



添加 libssh2.lib、ws2_32.lib 到“附加依赖库”中：



在代码的第一行加上一句 `#define _WINSOCK_DEPRECATED_NO_WARNINGS`，禁用警告。

10.编写C++代码

本代码极大程度地参考了学长的内容。

```
static int waitsocket(int socket_fd, LIBSSH2_SESSION* session)
{
    struct timeval timeout;
    fd_set fd;
    fd_set* writefd = NULL;
    fd_set* readfd = NULL;
    int dir, rc;

    timeout.tv_sec = 10; /* 超时时间10秒 */
    timeout.tv_usec = 0;

    FD_ZERO(&fd); /* 清除fd集合 */
    FD_SET(socket_fd, &fd); /* 将socket_fd添加到fd集合中 */

    /* 确定socket阻塞的方向 */
    dir = libssh2_session_block_directions(session);
    if (dir & LIBSSH2_SESSION_BLOCK_INBOUND)
        readfd = &fd;
    if (dir & LIBSSH2_SESSION_BLOCK_OUTBOUND)
        writefd = &fd;

    /* 等待socket准备好 */
    rc = select(socket_fd + 1, readfd, writefd, NULL, &timeout);
    return rc;
}
```



```

/* 打印错误信息并进行清理 */
static void handle_error(const char* message, LIBSSH2_SESSION* session, int
sock)
{
    fprintf(stderr, "%s\n", message);
    if (session) {
        libssh2_session_disconnect(session, "正常关闭, 感谢使用");
        libssh2_session_free(session);
    }
    if (sock != -1) {
#ifdef WIN32
        closesocket(sock);
#else
        close(sock);
#endif
    }
    libssh2_exit();
    exit(1);
}

int main(int argc, char* argv[])
{
    const char* hostname = "154.64.254.120"; /* 默认主机名 */
    const char* commandline = "rpm -qa|grep ssh"; /* 默认命令 */
    const char* username = "root"; /* 默认用户名 */
    const char* password = "q5bIePVbo0xxxxxxxx"; /* 默认密码 (后半打码) */
    unsigned long hostaddr;
    int sock = -1;
    struct sockaddr_in sin;
    const char* fingerprint;
    LIBSSH2_SESSION* session = NULL;
    LIBSSH2_CHANNEL* channel = NULL;
    int rc;
    int exitcode;
    char* exitsignal = (char*)"none";
    int bytecount = 0;
    size_t len;
    LIBSSH2_KNOWNHOSTS* nh;
    int type;

#ifdef WIN32
    WSADATA wsadata;
    int err;

    /* 初始化winsock */
    err = WSStartup(MAKEWORD(2, 0), &wsadata);
    if (err != 0) {
        fprintf(stderr, "WSAStartup失败, 错误码: %d\n", err);
        return 1;
    }
#endif

    if (argc > 1)
        hostname = argv[1];
    if (argc > 2)

```

```

    username = argv[2];
if (argc > 3)
    password = argv[3];
if (argc > 4)
    cmdline = argv[4];

if ((rc = libssh2_init(0)) != 0)
    handle_error("libssh2初始化失败", session, sock);

hostaddr = inet_addr(hostname);

/* 创建一个TCP套接字并连接到指定的主机和端口（默认22端口） */
sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock == -1)
    handle_error("创建套接字失败", session, sock);

sin.sin_family = AF_INET;
sin.sin_port = htons(22);
sin.sin_addr.s_addr = hostaddr;
if (connect(sock, (struct sockaddr*)&sin, sizeof(struct sockaddr_in)) !=
0)
    handle_error("连接失败", session, sock);

/* 创建一个libssh2会话实例 */
session = libssh2_session_init();
if (!session)
    handle_error("libssh2会话初始化失败", session, sock);

/* 设置会话为非阻塞模式 */
libssh2_session_set_blocking(session, 0);

/* 开始会话握手 */
while ((rc = libssh2_session_handshake(session, sock)) ==
LIBSSH2_ERROR_EAGAIN);
if (rc)
    handle_error("建立SSH会话失败", session, sock);

/* 初始化已知主机 */
nh = libssh2_knownhost_init(session);
if (!nh)
    handle_error("已知主机初始化失败", session, sock);

/* 从文件中读取所有已知主机 */
libssh2_knownhost_readfile(nh, "known_hosts",
LIBSSH2_KNOWNHOST_FILE_OPENSsh);

/* 将所有已知主机存储到文件中 */
libssh2_knownhost_writefile(nh, "dumpfile", LIBSSH2_KNOWNHOST_FILE_OPENSsh);

/* 获取会话的主机密钥指纹 */
fingerprint = libssh2_session_hostkey(session, &len, &type);
if (!fingerprint)
    handle_error("获取主机密钥指纹失败", session, sock);

struct libssh2_knownhost* host;

```

```

#if LIBSSH2_VERSION_NUM >= 0x010206
    /* 检查主机密钥 */
    int check = libssh2_knownhost_checkp(nh, hostname, 22, fingerprint, len,
LIBSSH2_KNOWNHOST_TYPE_PLAIN | LIBSSH2_KNOWNHOST_KEYENC_RAW, &host);
#else
    /* 1.2.5或更旧版本 */
    int check = libssh2_knownhost_check(nh, hostname, fingerprint, len,
LIBSSH2_KNOWNHOST_TYPE_PLAIN | LIBSSH2_KNOWNHOST_KEYENC_RAW, &host);
#endif

    fprintf(stderr, "主机检查: %d, 密钥: %s\n", check, (check <=
LIBSSH2_KNOWNHOST_CHECK_MISMATCH) ? host->key : "<无>");
    libssh2_knownhost_free(nh);

    /* 使用密码进行认证 */
    if (strlen(password) != 0) {
        while ((rc = libssh2_userauth_password(session, username, password)) ==
LIBSSH2_ERROR_EAGAIN);
        if (rc)
            handle_error("密码认证失败", session, sock);
    }
    else {
        /* 使用公钥文件进行认证 */
        while ((rc = libssh2_userauth_publickey_fromfile(session, username,
"/home/user/.ssh/id_rsa.pub", "/home/user/.ssh/id_rsa", password)) ==
LIBSSH2_ERROR_EAGAIN);
        if (rc)
            handle_error("公钥认证失败", session, sock);
    }

    /* 打开会话通道并执行命令 */
    while ((channel = libssh2_channel_open_session(session)) == NULL &&
libssh2_session_last_error(session, NULL, NULL, 0) == LIBSSH2_ERROR_EAGAIN) {
        waitsocket(sock, session);
    }
    if (!channel)
        handle_error("打开通道失败", session, sock);

    while ((rc = libssh2_channel_exec(channel, commandline)) ==
LIBSSH2_ERROR_EAGAIN) {
        waitsocket(sock, session);
    }
    if (rc != 0)
        handle_error("执行命令失败", session, sock);

    /* 循环读取命令输出并打印到标准错误输出 */
    while (1) {
        char buffer[0x4000];
        do {
            rc = libssh2_channel_read(channel, buffer, sizeof(buffer));
            if (rc > 0) {
                fwrite(buffer, 1, rc, stderr);
                bytecount += rc;
            }
            else if (rc != LIBSSH2_ERROR_EAGAIN) {
                fprintf(stderr, "libssh2_channel_read返回 %d\n", rc);
            }
        } while (rc < 0);
    }
}

```

```

    }
} while (rc > 0);

if (rc == LIBSSH2_ERROR_EAGAIN)
    waitsocket(sock, session);
else
    break;
}

/* 获取并打印命令退出状态 */
while ((rc = libssh2_channel_close(channel)) == LIBSSH2_ERROR_EAGAIN)
    waitsocket(sock, session);

if (rc == 0) {
    exitcode = libssh2_channel_get_exit_status(channel);
    libssh2_channel_get_exit_signal(channel, &exit_signal, NULL, NULL, NULL,
NULL, NULL);
}

if (exit_signal)
    fprintf(stderr, "\n收到信号: %s\n", exit_signal);
else
    fprintf(stderr, "\n退出码: %d 读取字节数: %d\n", exitcode, bytecount);

libssh2_channel_free(channel);

shutdown:
    libssh2_session_disconnect(session, "正常关闭, 感谢使用");
    libssh2_session_free(session);

#ifdef WIN32
    closesocket(sock);
#else
    close(sock);
#endif

    fprintf(stderr, "完成\n");
    libssh2_exit();

    return 0;
}

```

示例代码包含固定的主机名、用户名、密码和要运行的命令。

像这样运行：

>ProjectSSH 127.0.0.1 "user" "password" "commandline"

四、安全风险分析和安全策略设计

安全风险分析

1. 数据传输安全风险

- **风险描述：** 用户终端（智能手机）与Linux服务器之间传输的数据可能被窃听或篡改。
- **潜在影响：** 数据泄露、篡改、伪造攻击等。
- **威胁等级：** 高

2. 服务器安全风险

- **风险描述：** Linux服务器可能受到未经授权的访问、恶意软件攻击或漏洞利用。
- **潜在影响：** 服务器被攻陷、数据丢失或被盗、服务中断。
- **威胁等级：** 高

3. 远程管理安全风险

- **风险描述：** 管理终端通过SSH连接服务器进行远程配置和管理，可能会遭遇中间人攻击或暴力破解攻击。
- **潜在影响：** 管理权限被窃取、服务器配置被篡改、系统瘫痪。
- **威胁等级：** 中

4. 证书管理风险

- **风险描述：** SSL/TLS证书可能被伪造或被不当使用。
- **潜在影响：** 不可信的连接、用户数据被窃取、身份伪造。
- **威胁等级：** 中

5. 用户终端安全风险

- **风险描述：** 用户终端（智能手机）可能被恶意软件感染，导致用户数据泄露。
- **潜在影响：** 数据泄露、账户被盗用。
- **威胁等级：** 高

安全策略设计

1. 数据传输安全策略

- **TLS加密：** 强制所有用户终端与服务器之间的通信采用HTTPS协议，确保传输数据的机密性和完整性。
- **SSL隧道：** 建立SSL隧道保护内部通信，防止中间人攻击。
- **证书管理：** 使用受信任的证书颁发机构（CA）申请和管理SSL/TLS证书，定期更新和更换证书。

2. 服务器安全策略

- **防火墙配置：** 配置防火墙，限制对服务器的访问，仅允许必要的端口（如80, 443, 22）开放。
- **系统更新和补丁：** 定期更新服务器操作系统和应用软件，及时应用安全补丁。
- **入侵检测系统（IDS）和入侵防御系统（IPS）：** 部署IDS和IPS来监控和防御异常活动和潜在的攻击行为。
- **数据备份：** 定期备份服务器上的重要数据，确保在遭遇攻击或数据丢失时能够恢复。

3. 远程管理安全策略

- **SSH加固：** 使用强密码或公钥认证机制，禁用密码认证，强制使用公钥认证，并限制SSH登录尝试次数。
- **双因素认证（2FA）：** 为SSH登录配置双因素认证，增加额外的安全层。
- **IP白名单：** 配置SSH仅允许特定IP地址进行访问，防止未经授权的访问。

4. 证书管理策略

- **证书颁发和管理**：使用受信任的证书颁发机构（CA）申请和管理SSL/TLS证书，定期更新和更换证书。
 - **证书透明度（CT）**：配合使用证书透明度机制，防止证书被伪造或被不当使用。
5. **用户终端安全策略**
- **应用安全**：确保用户终端上的应用来源可信，并定期更新应用以修复已知漏洞。
 - **终端防护**：建议用户安装和使用防病毒软件，并定期进行系统扫描。
 - **用户教育**：教育用户关于钓鱼攻击和社交工程的风险，提高用户的安全意识。
6. **日志和监控策略**
- **日志记录**：对所有访问和操作进行详细的日志记录，保存并定期审计。
 - **实时监控**：实时监控系统和网络活动，及时发现和响应安全事件。

功能角色	安全风险	安全策略	优先级
管理员 (A)	配置工具（用户终端）(1)	使用强密码和双因素认证，限制IP访问，定期审计	高
管理员 (A)	测度装置 (2)	加密通信，使用安全协议，配置防火墙和入侵检测系统	高
管理员 (A)	服务器 (3)	定期更新系统和应用，使用IDS和IPS，定期备份数据	高
管理员 (A)	数据库 (4)	数据加密存储，定期备份，严格访问控制	高
普通用户 (B)	用户终端 (5)	使用HTTPS安全访问，定期更新终端软件 and 操作系统	中

五、参考文献

[1] [ChestnutSilver/Linux-SSH-System: 同济大学信息安全原理课程设计大作业. \(SSH原型系统, 2023\) Comprehensive Assignment of Course Design of Information Security Principles in Tongji University. \(github.com\).](#)

[2] [域名、企业和服务器备案的条件和备案流程 备案\(ICP Filing\)-阿里云帮助中心 \(aliyun.com\).](#)

[3] [linux ssh用root用户登录 - 讲击的davis - 博客园 \(cnblogs.com\).](#)

[4] [猫头虎博客: SSH连接失败ssh: connect to host port 22: Connection refused”解决大揭秘-腾讯云开发者社区-腾讯云 \(tencent.com\).](#)

[5] [Visual Studio 2022 Preview 不支持dsp\(dsw\) - QZLin - 博客园 \(cnblogs.com\).](#)

[6] [Win32/Win64 OpenSSL Installer for Windows - Shining Light Productions \(slproweb.com\).](#)