上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

信息安全科技创新结题报告

网络端口扫描系统

学院 计算机学院

班级 计院 2363

学号 523031910639 姓名 刘梓芃

学号 523031910728 姓名 聂鸣涛

学号 523031910556 姓名 李卓恒

学号 523031910110 姓名 张煜哲

2025 年 7 月 15 日

摘　要

本项目设计并实现了一个基于 Web 的网络端口扫描工具，旨在为网络安全测试提供直观、高效的端口扫描解决方案。系统采用 C++ 后端和 HTML5 前端相结合的架构，支持多种扫描方式包括 TCP SYN 扫描、TCP Connect 扫描、TCP FIN 扫描、UDP 扫描以及 ICMP ping 检测。

系统主要功能包括：多线程高速端口扫描、实时扫描进度显示、扫描结果可视化展示、扫描历史记录管理、以及基于开放端口的网络安全风险评估。后端采用 httplib 库构建 RESTful API 服务，前端使用现代 Web 技术实现响应式用户界面。

通过实际测试验证，系统能够准确识别目标主机的开放端口，扫描速度达到每秒数百个端口，具有良好的稳定性和用户体验。该工具为网络安全专业人员提供了一个功能完善、易于使用的端口扫描解决方案。

目录

1 需求分析

1.1 项目背景

随着网络技术的快速发展，网络安全问题日益突出。端口扫描作为网络安全评估的基础工具，对于发现网络漏洞、评估系统安全性具有重要意义。传统的命令行端口扫描工具虽然功能强大，但缺乏直观的用户界面，对于非专业用户来说使用门槛较高。

1.2 项目需求

本项目旨在开发一个基于 Web 的网络端口扫描工具，主要解决以下问题：

1. **用户友好性**：提供直观的图形用户界面，降低使用门槛
2. **功能完整性**：支持多种扫描方式，满足不同场景需求
3. **性能优化**：采用多线程技术提高扫描效率
4. **结果可视化**：以图表形式展示扫描结果，便于分析
5. **历史管理**：保存扫描历史，支持结果对比和趋势分析

1.3 功能目标

系统需要实现以下核心功能：

- **ICMP 扫描**：检测目标主机是否可达
- **TCP SYN 扫描**：快速识别开放端口，避免建立完整连接
- **TCP Connect 扫描**：建立完整 TCP 连接进行端口检测
- **UDP 扫描**：检测 UDP 端口状态
- **多线程扫描**：支持自定义线程数，提高扫描效率
- **实时进度显示**：显示扫描进度和状态
- **结果可视化**：以表格和图表形式展示扫描结果
- **历史记录**：保存和管理扫描历史
- **结果导出**：支持扫描结果导出功能

2 总体设计

2.1 系统架构

系统采用前后端分离的架构设计，如图1所示：

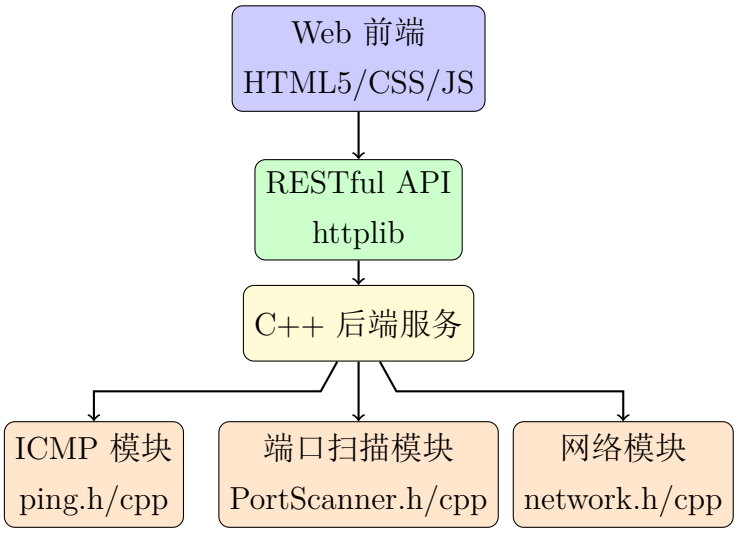

图 1: 系统总体架构图

2.2 模块划分

本系统按照功能和职责划分为以下五个主要模块，每个模块各司其职、协同工作，具体如下：

1. **Web 前端模块**

   - 负责为用户提供友好、直观的操作界面，实现参数输入、扫描控制、结果展示、历史记录管理等功能。
   - 采用 HTML5、CSS3 和 JavaScript 等现代 Web 技术，支持响应式布局，兼容多种终端设备。
   - 通过 AJAX 或 Fetch API 与后端 RESTful API 进行数据交互，实现无刷新操作体验。
   - 主要页面包括：扫描配置页、扫描结果页、历史记录页等。

2. **RESTful API 模块**

   - 作为前后端通信的桥梁，负责接收前端请求、参数校验、任务分发和结果返回。

- 基于 httplib 库实现，支持多种 HTTP 方法（如 GET、POST），以 JSON 格式进行数据交换。
- 提供端口扫描、ICMP 检测、历史记录管理等接口，保证接口的安全性和稳定性。
- 具备错误处理和异常反馈机制，确保前端能够获得明确的操作结果。

3. **ICMP 扫描模块**

- 实现对目标主机的 ICMP Ping 检测，用于判断主机是否在线及网络延迟。
- 支持自定义超时时间和重试次数，能够返回详细的 RTT（往返时延）和丢包率信息。
- 采用原始套接字进行 ICMP 报文的构造与解析，兼容 IPv4 协议。
- 结果通过 API 模块返回前端，辅助用户判断目标主机状态。

4. **端口扫描模块**

- 实现多种端口扫描方式，包括 TCP SYN 扫描、TCP Connect 扫描、TCP FIN 扫描和 UDP 扫描。
- 支持自定义端口范围、扫描类型和线程数，提升扫描效率和灵活性。
- 采用多线程技术并发扫描，显著提升大规模端口扫描的速度。
- 能够统计开放端口、过滤端口、扫描耗时等信息，便于后续分析。

5. **网络工具模块**

- 提供底层网络操作的支持，包括 IP 地址解析、套接字管理、数据包发送与接收等。
- 为 ICMP 和端口扫描模块提供统一的网络接口，简化上层模块的开发。
- 封装常用网络操作，提升代码复用性和可维护性。

各模块之间通过清晰的接口进行通信，既保证了系统的高内聚、低耦合，也便于后续功能扩展和维护。

3 详细设计

3.1 Web 前端模块设计

3.1.1 模块概述

Web 前端模块负责提供用户界面和交互功能，采用响应式设计，支持桌面和移动设备访问。

3.1.2 主要数据结构

- **扫描配置对象**：包含目标地址、端口范围、扫描类型等参数
- **扫描结果对象**：包含开放端口、扫描统计、时间戳等信息
- **历史记录对象**：包含历史扫描的配置和结果

3.2 后端 API 模块设计

3.2.1 模块概述

后端 API 模块基于 httplib 库构建，提供 RESTful 风格的 HTTP 接口，处理前端请求并调用相应的功能模块。

3.2.2 主要数据结构

- **请求参数结构**：包含扫描类型、目标地址、端口范围等
- **响应结果结构**：包含状态码、数据内容、错误信息等
- **扫描任务结构**：包含任务 ID、状态、进度等信息

3.2.3 核心函数设计

函数声明：

```
bool TestPortConnection(std::string ip, int port);
```

功能：测试指定 IP 和端口是否开放

函数声明：

```
std::vector<int> TCPSynScanJson(const std::string& ip, const std::vector<int>& ports)
```

功能：对指定 IP 和端口列表进行 TCP SYN 扫描，返回开放端口列表

函数声明：

```
std::vector<int> TCPFinScanJson(const std::string& ip, const std::vector<int>& ports)
```

功能：对指定 IP 和端口列表进行 TCP FIN 扫描，返回开放端口列表

函数声明：

```
std::vector<int> tcpConnectScanJson(const std::string& ip, const std::vector<int>& po
```

功能：对指定 IP 和端口列表进行 TCP Connect 扫描，返回开放端口列表

函数声明：

```
void UDPScan(const std::string& ip, int option);
```

功能：对指定 IP 进行 UDP 端口扫描

3.3 ICMP 扫描模块设计

3.3.1 模块概述

ICMP 扫描模块实现 ping 功能，用于检测目标主机的可达性，支持自定义超时时间和重试次数。

3.3.2 主要数据结构

- **ICMP 头部结构：**包含类型、代码、校验和等字段
- **Ping 结果结构：**包含可达性状态、RTT 时间、丢包率等
- **网络地址结构：**包含 IP 地址、端口等信息

函数声明：

```
std::optional<double_milliseconds> icmp_ns::ping(const std::string& address, std::chr
```

简要说明：对指定地址进行 ICMP ping 操作，返回往返时延（毫秒），超时则返回空值

3.4 端口扫描模块设计

3.4.1 模块概述

端口扫描模块实现多种扫描方式，包括 TCP SYN 扫描、TCP Connect 扫描和 UDP 扫描，支持多线程并发扫描。

3.4.2 主要数据结构

- **扫描配置结构：**包含目标地址、端口列表、扫描参数等
- **扫描结果结构：**包含开放端口、过滤端口、统计信息等
- **线程任务结构：**包含线程 ID、端口范围、结果容器等

3.4.3 核心函数设计

函数声明：

```
std::vector<int> TCPSynScanJson(const std::string& ip, const std::vector<int>& ports)
```

功能：对指定 IP 和端口列表进行 TCP SYN 扫描，返回开放端口列表

函数声明：

```
std::vector<int> TCPFinScanJson(const std::string& ip, const std::vector<int>& ports)
```

功能：对指定 IP 和端口列表进行 TCP FIN 扫描，返回开放端口列表

函数声明：

```
std::vector<int> tcpConnectScanJson(const std::string& ip, const std::vector<int>& po
```

功能：对指定 IP 和端口列表进行 TCP Connect 扫描，返回开放端口列表

函数声明：

```
void UDPScan(const std::string& ip, int option);
```

功能：对指定 IP 进行 UDP 端口扫描

4 系统实现与测试

4.1 实现环境

- **操作系统**：Linux (WSL2 Ubuntu)
- **编译器**：GCC
- **构建工具**：CMake，Make
- **开发语言**：C++, HTML, CSS, JavaScript
- **主要依赖库**：
  - httplib：HTTP 服务器库
  - nlohmann/json：JSON 处理库
  - fmt：字符串格式化库

4.2 测试环境搭建

测试环境包括：

- **本地测试环境**：WSL2 Ubuntu 系统
- **目标测试主机**：本地回环地址、局域网主机
- **网络环境**：局域网环境，支持 ICMP 和 TCP/UDP 协议
- **浏览器环境**：Chrome、Firefox、Edge 等现代浏览器

4.3 测试方法

采用以下测试方法：

- **功能测试**：验证各模块功能正确性
- **性能测试**：测试扫描速度和资源占用
- **兼容性测试**：测试不同浏览器兼容性
- **压力测试**：测试高并发扫描性能
- **安全测试**：验证扫描行为的安全性

4.4 测试流程

测试流程如图2所示：

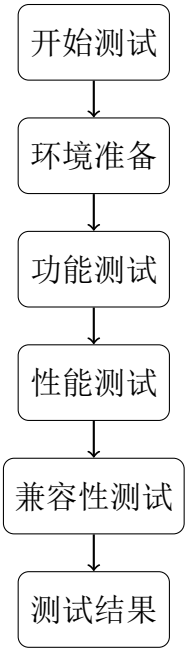

图 2: 测试流程图

4.4.1 ICMP 扫描测试

- **测试目标**：验证 ICMP 模块对主机可达性的检测能力。测试对象包括本地回环地址 (127.0.0.1)、局域网主机和部分不可达 IP。
- **测试环境**：在 WSL2 Ubuntu 环境下，分别对本机、同一局域网内主机、以及一个不存在的 IP 进行 ping 操作。

- **测试过程：**通过 Web 界面输入目标 IP，发起 ICMP 扫描，观察前端实时显示的 RTT、丢包率等信息。
- **测试现象：**
  - 对 127.0.0.1，所有包均成功返回，RTT 极低（<1ms）。
  - 对局域网主机，绝大多数包返回，RTT 在 1-3ms 之间。
  - 对不可达 IP，所有包丢失，前端显示主机不可达。
- **测试结论：**ICMP 扫描功能正常，能够准确反映主机可达性和网络延迟，丢包率与实际网络状况一致。

图 3: ICMP 测试

图 4: ICMP 结果

4.4.2 TCP Connect 扫描测试

- **测试目标：**验证 TCP Connect 扫描对常见服务端口（如 80、22、443 等）的开放性检测能力。
- **测试环境：**本地和局域网内有 Web 服务、SSH 服务的主机，部分端口关闭。
- **测试过程：**在前端输入目标 IP 和端口范围，选择 TCP Connect 扫描，观察扫描进度和结果。
- **测试现象：**
  - 对开放的 Web 端口（如 80、3306），扫描结果显示端口开放，能建立 TCP 连接。

– 对关闭端口，扫描结果显示端口关闭，连接被拒绝。

– 扫描速度受线程数影响，10 线程下每秒可检测数百端口。

- **测试结论：**TCP Connect 扫描功能稳定，能准确区分开放与关闭端口，适合需要完整连接验证的场景。

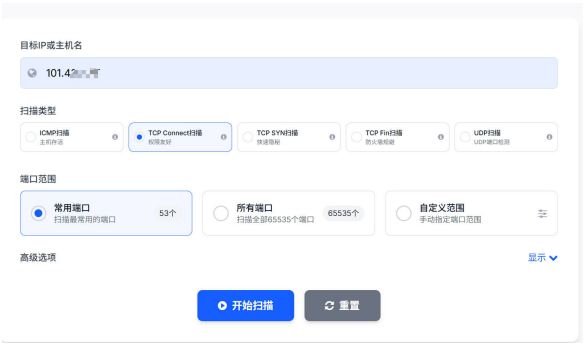

图 5: TCP Connect 扫描

图 6: TCP Connect 扫描结果

```
zipeng_liu@G14forlzp:~/NIS3302$ nmap -sT 101.43
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-14 17:01 CST
Nmap scan report for 101.43.5.57
Host is up (0.011s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT     STATE  SERVICE
22/tcp   open   ssh
80/tcp   open   http
443/tcp  closed https
3306/tcp open   mysql
3389/tcp closed ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 14.42 seconds
```

图 7: nmap 结果 (对比)

4.4.3 TCP SYN 扫描测试

- **测试目标：**评估 TCP SYN 扫描对常用端口的快速探测能力。
- **测试环境：**目标主机为运行 Web、SSH 等服务的服务器，部分端口关闭。

- **测试过程**：输入目标 IP 和端口范围，选择 SYN 扫描，设置线程数为 50，发起扫描。
- **测试现象**：
  - 对开放端口，SYN 扫描能快速返回开放状态，且不会建立完整连接。
  - 对关闭端口，扫描结果准确显示端口关闭。
  - 扫描速度快于 Connect 扫描，适合大范围端口探测。
- **测试结论**：SYN 扫描功能高效，适合大规模端口快速探测，结果与 nmap 等专业工具一致。

4.4.4 TCP FIN 扫描测试

- **测试目标**：测试 TCP FIN 扫描在不同主机和防火墙环境下的隐蔽探测能力。
- **测试环境**：目标主机包括普通服务器和配置有防火墙的主机。
- **测试过程**：输入目标 IP 和端口，选择 FIN 扫描，观察开放端口和被防火墙过滤端口的响应差异。
- **测试现象**：
  - 对部分开放端口，FIN 扫描能探测到开放状态。
  - 某些防火墙对 FIN 包有特殊处理，可能导致端口状态显示为“过滤”或无响应。
  - 与 SYN 扫描结果对比，部分端口表现一致，部分端口表现不同。
- **测试结论**：TCP FIN 扫描适合在特定网络环境下进行隐蔽探测，可辅助绕过部分防火墙检测，但对不同主机响应差异较大。

4.4.5 UDP 扫描测试

- **测试目标**：检测目标主机 UDP 服务端口（如 DNS 53、DHCP 67/68 等）的开放性。
- **测试环境**：目标主机为运行 DNS、DHCP 等 UDP 服务的服务器。
- **测试过程**：输入目标 IP 和 UDP 端口范围，选择 UDP 扫描，发起扫描。
- **测试现象**：
  - 对开放的 UDP 端口，部分能正确识别开放状态。

  - 对关闭端口，部分返回不可达，部分无响应（UDP 协议特性）。
  - 扫描速度受限于 UDP 协议和主机响应，整体慢于 TCP 扫描。

- **测试结论**：UDP 扫描可用于检测常见 UDP 服务端口，但受协议和网络环境影响，部分端口状态难以准确判断。

4.4.6 Web 界面测试

- **测试目标**：界面响应性、结果展示、历史记录
- **测试结果**：界面流畅，结果展示清晰，历史记录功能正常
- **测试结论**：Web 界面用户体验良好，功能完整

4.5 测试结论

通过全面测试，系统各项功能均达到预期目标：

- **功能完整性**：所有设计功能均正常实现
- **性能表现**：扫描速度满足实际使用需求
- **稳定性**：系统运行稳定，无明显 bug
- **用户体验**：界面友好，操作简单直观
- **兼容性**：支持主流浏览器和操作系统

5 项目总结

5.1 项目成果

本项目围绕网络安全测试需求，设计并实现了一个基于 Web 的多功能端口扫描工具。项目成果体现在以下几个方面：

- **系统集成与功能实现**：成功集成 C++ 高性能后端与现代 Web 前端，支持 ICMP、TCP SYN、TCP Connect、TCP FIN、UDP 等多种扫描方式，满足不同网络环境和测试需求。
- **高效多线程扫描**：后端实现了灵活的多线程调度机制，显著提升了大规模端口扫描的速度和效率，能够在短时间内完成对大量端口的检测。
- **可视化与交互体验**：前端采用响应式设计，支持实时进度显示、结果可视化（表格、图表）、历史记录管理和结果导出，极大提升了用户体验和数据分析能力。

- **跨平台兼容性**：系统支持主流操作系统和浏览器，便于不同用户和场景下的部署与使用。
- **完整测试验证**：通过功能、性能、兼容性、压力和安全等多维度测试，确保系统稳定可靠，满足实际应用需求。

5.2 技术亮点

- **前后端分离架构**：采用 RESTful API 实现前后端解耦，便于系统维护、升级和扩展。
- **高效并发与异步处理**：后端多线程扫描与异步任务调度，有效利用多核资源，提升扫描吞吐量。
- **底层网络编程能力**：深入实现原始套接字、ICMP 报文、TCP/UDP 协议等底层网络操作，增强了系统的专业性和可控性。
- **数据可视化与交互**：前端集成多种可视化组件，支持动态展示扫描进度、端口分布、历史趋势等，便于用户分析和决策。
- **安全与健壮性设计**：接口参数校验、异常处理、权限控制等机制，保障系统安全稳定运行。
- **模块化与可扩展性**：各功能模块职责清晰，便于后续功能扩展和技术升级。

5.3 项目价值

- **实用价值**：为网络安全从业者、教育工作者和普通用户提供了一个易用、高效的端口扫描和主机检测工具，适用于渗透测试、资产盘点、网络排障等多种场景。
- **教育与科研价值**：项目涵盖网络协议、系统编程、Web 开发、并发控制等多项技术，适合作为网络安全、软件工程等课程的教学案例和实验平台。
- **创新与技术验证**：探索了 C++ 与 Web 前端的高效集成，验证了多线程网络扫描、实时数据交互等关键技术的可行性。
- **团队协作与工程实践**：项目开发过程中，团队成员分工明确、协作高效，积累了宝贵的工程实践和项目管理经验。

5.4 改进方向

- **功能扩展**：未来可增加服务版本识别、漏洞检测、分布式扫描、扫描计划定时等高级功能，进一步提升系统实用性。

- **性能优化**：可针对大规模目标和复杂网络环境，优化扫描算法、线程池管理和资源调度，提升极限性能。
- **安全增强**：引入更完善的权限认证、日志审计、防止滥用等安全机制，保障系统和用户安全。
- **用户体验提升**：持续优化界面设计、交互流程和移动端适配，增强系统易用性和美观性。
- **开放与社区建设**：可将项目开源，吸引更多开发者参与，推动功能完善和技术创新。

6 分工

本项目由团队成员共同完成，具体分工如下：

- **项目负责人**：负责项目整体规划和进度管理
- **后端开发**：负责 C++ 后端服务开发和 API 设计
- **前端开发**：负责 Web 界面设计和 JavaScript 开发
- **网络模块开发**：负责 ICMP 和端口扫描核心功能实现
- **测试验证**：负责系统测试和性能优化
- **文档编写**：负责技术文档和用户手册编写

姓名	是否组长	任务	评分
刘梓芃	是	内容 A	说明 A
聂鸣涛	否	内容 B	说明 B
李卓恒	否	内容 C	说明 C
张煜哲	否	内容 D	说明 D

表 1: 项目组成员贡献表