# 网络攻防课程项目1 网络攻防扫描工具集成平台

[网络攻防课程项目1 网络攻防扫描工具集成平台](#网络攻防课程项目1-网络攻防扫描工具集成平台)  
 [扫描器选择](#扫描器选择)  
 [Nmap](#nmap)  
 [Nikto](#nikto)  
 [其他扫描器](#其他扫描器)  
 [项目开发框架选择](#项目开发框架选择)  
 [源代码](#源代码)  
 [项目测试](#项目测试)  
 [扫描本机](#扫描本机)  
 [扫描百度](#扫描百度)  
 [项目打包](#项目打包)  
 [项目上传](#项目上传)

## 扫描器选择

### Nmap

Nmap（Network Mapper）是一个开源的网络扫描和安全审计工具。它被广泛用于网络发现、主机探测和安全评估。

**Nmap的主要特点:**

**主机发现**：

Nmap 可以快速检查网络上的活动主机，识别哪些设备在线。

**端口扫描**：

能够扫描目标主机的开放端口，以确定哪些服务在运行。支持多种扫描技术，如 TCP SYN 扫描、UDP 扫描等。

**服务和版本检测**：

Nmap 可以识别开放端口所对应的服务类型及其版本信息，帮助用户了解网络环境。

**操作系统检测**：

Nmap 能够通过分析响应包来推测目标主机的操作系统类型及其版本。

**脚本引擎**：

Nmap 包含一个强大的脚本引擎（NSE），用户可以编写或使用现成的脚本来执行更复杂的任务，例如漏洞检测和信息收集。

**图形用户界面**：

虽然 Nmap 主要通过命令行使用，它还提供了一个图形用户界面（Zenmap），便于新手用户使用。

### Nikto

Nikto 是一个开源的网页服务器扫描器，用于发现潜在的安全漏洞和配置错误。它能够检测多种类型的漏洞，包括过时的软件版本、常见的安全问题以及服务器配置问题。

**全面扫描**：

* Nikto 能够扫描服务器的多个方面，包括文件和脚本的漏洞、HTTP 头信息、服务器版本等。

**插件支持**：

* Nikto 支持自定义插件，可以扩展其功能以适应特定的安全需求。

**配置错误检测**：

* 检测常见的配置错误，例如不安全的目录权限以及未加密的敏感信息。

**扫描结果报告**：

* Nikto 提供详细的扫描结果，并支持以多种格式导出报告，包括 HTML、CSV 和 TXT。

**更新频率**：

* Nikto 的漏洞数据库定期更新，以包含最新的漏洞信息。

### 其他扫描器

我也寻找了其他可用的扫描器，比如openvas(gvm),metasploit等等，但是由于想开发一个轻量的扫描器，并且缺少linux开发的经验，就选择了nmap和nikto两个比较简单的扫描器进行集成开发

## 项目开发框架选择

项目选择使用Python的Flask框架开发

## 源代码

**app.py**

主程序，定义了扫描器的函数以及处理扫描信息的函数

from flask import Flask, render\_template, request  
import subprocess  
  
app = Flask(\_\_name\_\_)  
  
@app.route('/')  
def index():  
 return render\_template('index.html')  
  
@app.route('/scan', methods=['POST'])  
def scan():  
 scanners = request.form.getlist('scanners') # 获取多个扫描器  
 target = request.form['target']  
 reports = []  
  
 for scanner in scanners:  
 if scanner == 'nmap':  
 command = f'nmap {target}'  
 elif scanner == 'nikto':  
 command = f'nikto -h {target} 5'  
 else:  
 return "Invalid scanner", 400  
  
 result = subprocess.run(command, shell=True, capture\_output=True, text=True)  
 report = format\_report(scanner, result.stdout)  
 reports.append(report) # 直接使用 HTML 格式的报告  
  
 return render\_template('report.html', reports=reports)  
  
def format\_report(scanner, output):  
 if scanner == 'nmap':  
 return format\_nmap\_report(output)  
 elif scanner == 'nikto':  
 return format\_nikto\_report(output)  
 else:  
 return "未知扫描器的输出"  
  
def format\_nmap\_report(output):  
 lines = output.splitlines()  
 report = "<h3>Nmap 扫描报告</h3>"  
   
 # 提取关键信息  
 host\_info = ""  
 open\_ports = []  
   
 for line in lines:  
 if line.startswith("Nmap scan report for"):  
 host\_info = line  
 elif "open" in line:  
 open\_ports.append(line.strip().split()) # 按空格分割，便于后续表格显示  
   
 # 添加主机信息  
 report += f"<p><strong>主机信息:</strong> {host\_info}</p>"  
   
 # 添加开放端口信息为表格  
 if open\_ports:  
 report += "<h4>开放端口:</h4>"  
 report += "<table border='1' cellpadding='5' cellspacing='0' style='border-collapse: collapse;'>"  
 report += "<tr><th>端口</th><th>状态</th><th>服务</th></tr>"  
 for port\_info in open\_ports:  
 port = port\_info[0] # 端口  
 state = port\_info[1] # 状态  
 service = port\_info[2] if len(port\_info) > 2 else "未知" # 服务  
 report += f"<tr><td>{port}</td><td>{state}</td><td>{service}</td></tr>"  
 report += "</table>"  
 else:  
 report += "<p>没有开放的端口。</p>"  
   
 return report  
  
def format\_nikto\_report(output):  
 lines = output.splitlines()  
 report = "<h3>Nikto 扫描报告</h3>"  
   
 # 提取关键信息  
 vulnerabilities = []  
  
 for line in lines:  
 if line.strip(): # 只处理非空行  
 # 检查特定关键字以提取漏洞信息  
 if "created without the httponly flag" in line:  
 vulnerabilities.append("未设置httponly的Cookie: " + line)  
 elif "not present" in line:  
 vulnerabilities.append("缺少安全头: " + line)  
 elif "found" in line:  
 vulnerabilities.append("发现不常见的HTTP头: " + line)  
 elif "contains" in line:  
 vulnerabilities.append("robots.txt条目: " + line)  
 elif "returned a non-forbidden" in line:  
 vulnerabilities.append("robots.txt中返回非禁止访问的路径: " + line)  
  
 # 添加漏洞信息为表格  
 if vulnerabilities:  
 report += "<h4>发现的漏洞:</h4>"  
 report += "<table border='1' cellpadding='5' cellspacing='0' style='border-collapse: collapse;'>"  
 report += "<tr><th>漏洞描述</th></tr>"  
 for vuln in vulnerabilities:  
 report += f"<tr><td>{vuln}</td></tr>"  
 report += "</table>"  
 else:  
 report += "<p>没有发现漏洞。</p>"  
   
 return report  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

**index.html**

<!DOCTYPE html>  
<html lang="zh">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">  
 <title>扫描器选择</title>  
</head>  
<body>  
 <div class="container">  
 <h1>选择扫描器</h1>  
 <form action="{{ url\_for('scan') }}" method="post">  
 <label for="target">目标网址：</label>  
 <input type="text" id="target" name="target" required placeholder="例如: example.com">  
   
 <fieldset>  
 <legend>选择扫描器：</legend>  
 <div class="checkbox-group">  
 <label><input type="checkbox" name="scanners" value="nmap"> Nmap</label>  
 <label><input type="checkbox" name="scanners" value="nikto"> Nikto</label>  
 </div>  
 </fieldset>  
   
 <button type="submit">开始扫描</button>  
 </form>  
 </div>  
</body>  
</html>

扫描器标签页，可以输入IP地址，以及选择扫描器

**report.html**

<!DOCTYPE html>  
<html lang="zh">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">  
 <title>扫描报告</title>  
</head>  
<body>  
 <div class="container">  
 <h1>扫描报告</h1>  
 {% for report in reports %}  
 <div class="report-content">{{ report|safe }}</div>  
 {% endfor %}  
 <a href="{{ url\_for('index') }}">返回</a>  
 </div>  
</body>  
</html>

报告扫描结果的网页

**style.css**

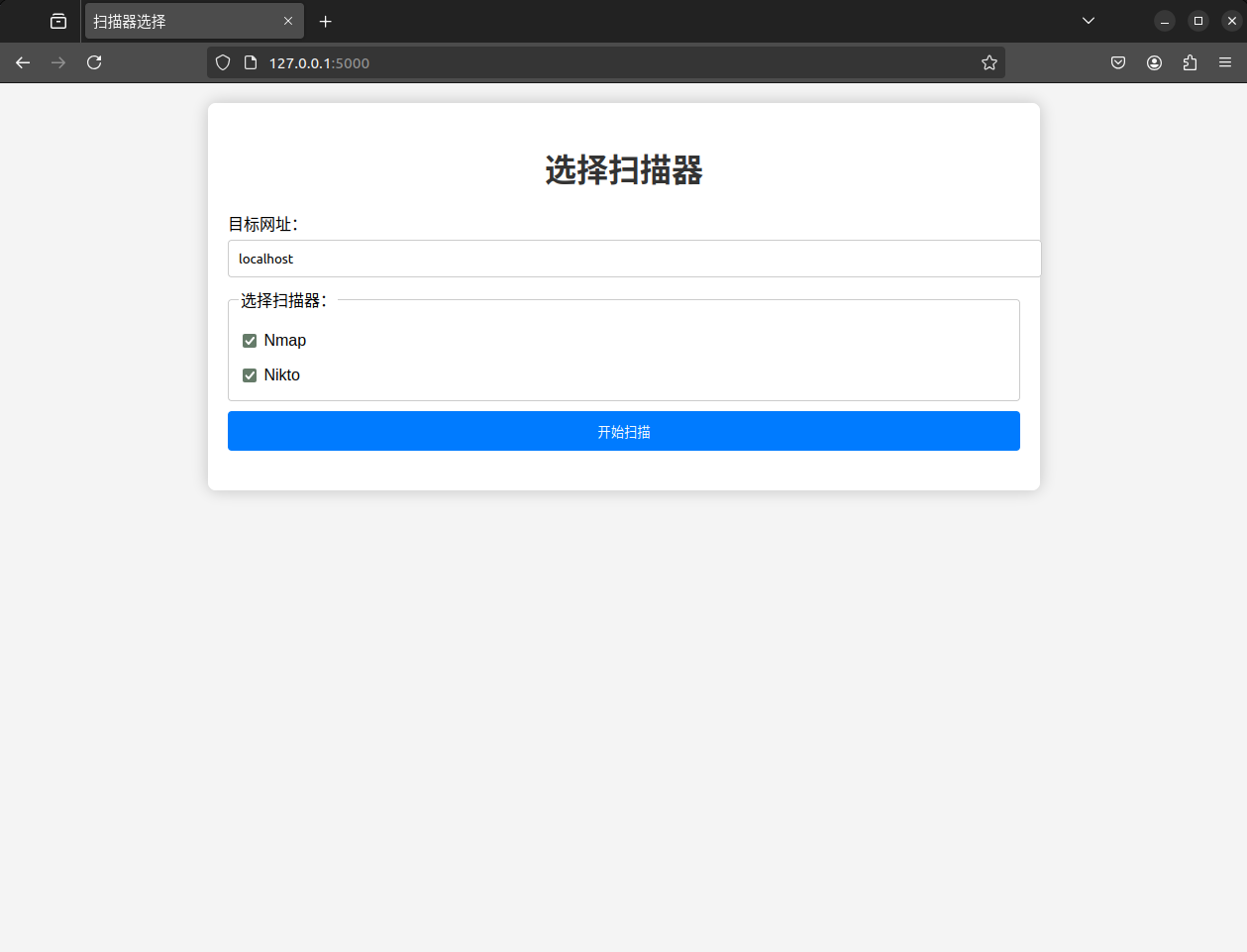
一些简单的样式，用于美化界面

body {  
 font-family: Arial, sans-serif;  
 background-color: #f4f4f4;  
 margin: 0;  
 padding: 20px;  
}  
  
.container {  
 max-width: 800px;  
 margin: auto;  
 background: white;  
 padding: 20px;  
 border-radius: 8px;  
 box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);  
}  
  
h1 {  
 color: #333;  
 text-align: center;  
 margin-bottom: 20px;  
}  
  
h3 {  
 color: #007BFF;  
 margin-top: 30px;  
 border-bottom: 2px solid #007BFF;  
 padding-bottom: 5px;  
}  
  
h4 {  
 color: #555;  
 margin-top: 20px;  
}  
  
form {  
 margin: 20px 0;  
}  
  
label {  
 display: block;  
 margin: 10px 0 5px;  
}  
  
input[type="text"] {  
 padding: 10px;  
 width: 100%;  
 border: 1px solid #ccc;  
 border-radius: 4px;  
 transition: border-color 0.3s;  
}  
  
input[type="text"]:focus {  
 border-color: #007BFF;  
 outline: none;  
}  
  
fieldset {  
 border: 1px solid #ccc;  
 border-radius: 4px;  
 padding: 10px;  
 margin: 10px 0;  
}  
  
.checkbox-group {  
 display: flex;  
 flex-direction: column;  
}  
  
.checkbox-group label {  
 margin-bottom: 5px;  
}  
  
button {  
 padding: 10px 15px;  
 background-color: #007BFF;  
 color: white;  
 border: none;  
 border-radius: 4px;  
 cursor: pointer;  
 width: 100%;  
 transition: background-color 0.3s;  
}  
  
button:hover {  
 background-color: #0056b3;  
}  
  
.report-content {  
 background-color: #f9f9f9;  
 padding: 15px;  
 border-radius: 4px;  
 overflow-x: auto;  
 white-space: pre-wrap;  
 margin-top: 15px;  
}  
  
table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 15px;  
}  
  
table, th, td {  
 border: 1px solid #ccc;  
}  
  
th {  
 background-color: #007BFF;  
 color: white;  
 padding: 10px;  
 text-align: left;  
}  
  
td {  
 padding: 10px;  
 background-color: #f9f9f9;  
}  
  
td:hover {  
 background-color: #f1f1f1;  
}  
  
@media (max-width: 600px) {  
 .container {  
 padding: 15px;  
 }  
}

## 项目测试

使用应用扫描

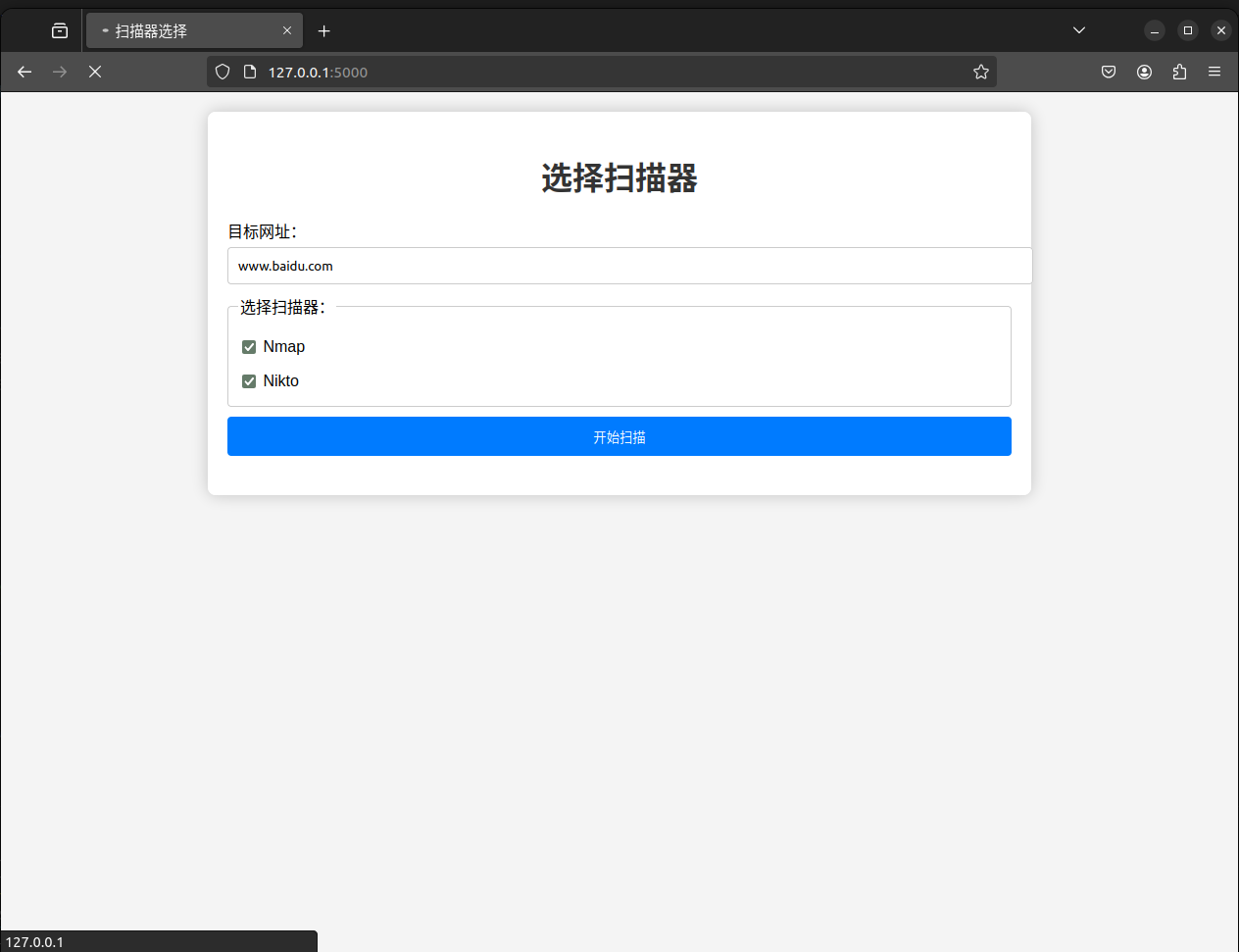
### 扫描本机



**扫描结果**



### 扫描百度



**扫描结果**





## 项目打包

使用docker进行项目打包

**Dockerfile**

FROM ubuntu:24.04  
  
ENV DEBIAN\_FRONTEND=noninteractive  
  
# 更新包列表并安装必要的工具  
RUN apt-get update && \  
 apt-get install -y python3 python3-pip python3-venv nmap nikto && \  
 apt-get clean && \  
 rm -rf /var/lib/apt/lists/\*  
  
# 设置工作目录  
WORKDIR /app  
  
# 复制依赖文件  
COPY requirements.txt .  
  
# 创建虚拟环境并安装依赖  
RUN python3 -m venv venv && \  
 . venv/bin/activate && \  
 pip install --no-cache-dir -r requirements.txt  
  
# 复制应用代码  
COPY . .  
  
# 暴露应用的端口  
EXPOSE 5000  
  
# 设置环境变量以确保Flask使用正确的主机和端口  
ENV FLASK\_APP=app.py  
ENV FLASK\_RUN\_HOST=0.0.0.0  
  
# 运行Flask应用  
CMD ["venv/bin/python", "-m", "flask", "run"]

使用Ubuntu作为基础镜像，安装扫描工具和创建虚拟环境，安装python依赖，最后执行应用

## 项目上传

本项目已经上传到dockerhub

欢迎下载试用

[网址](https://hub.docker.com/repository/docker/yang20031012/my_scanner/general)