

# 网络流量数据可视化系统 使用手册

## 目录

目录 .....	1
文 件 修 订 记 录 .....	2
1. 总体功能描述 .....	3
2. 运行环境 .....	4
3. 系统编译环境 .....	5
4. 系统配置说明 .....	7
5. 系统运行说明 .....	10
6. 系统操作说明 .....	11
6.1. 注册登录操作 .....	11
6.2. 找回密码操作 .....	13
6.3. 网络攻击数据显示 .....	16
6.4. 上传数据操作 .....	18
6.5. 操作流程示意图 .....	22
7. 系统后台监测 .....	23
8. 部分功能支持说明 .....	24
8.1. 数据可视化 .....	24
8.2. 找回密码 .....	24
9. 系统数据库说明 .....	26
10. 配置脚本文件 .....	27
10.1. 数据库配置文件 .....	27
10.2. 邮箱配置文件 .....	28

文件修订记录

版本号	生成日期	作者	修订内容
V1.0	2023-04-10	冯读硕	初始版本

## 1. 总体功能描述

系统采用 Python 的全栈式 Web 框架和异步网络库 Tornado 进行开发，配合 HTML，Javascript，CSS 等实现系统的 Web 框架、HTTP 客户端和服务端。

系统主要实现基于数据可视化技术的网络流量数据分析功能。系统分为服务器后端和用户前端两个部分。服务器后端可以搭建在 windows 或 Linux 平台上。用户可以从 Web 前端注册并登录到该系统中，从而进行数据可视化分析、数据上传分析等操作。

## 2. 运行环境

### 硬件要求

类 别	基本要求
服务器端	CPU 2G 内存 2G 以上；硬盘剩余空间不低于 50G
客户端	手机或电脑 1G 内存及以上；硬盘空间 10G 及以上；需要有摄像头设备的支持。

### 软件要求：

类别	名 称	基本环境
服务器端	操作系统	支持 Windows8 及以上；Linux5.19 及以上
	数据库软件	支持 MySQL8.0.27
客户端	操作系统	Android/Windows8 及以上
	其它软件	Chrome 浏览器；IE 浏览器

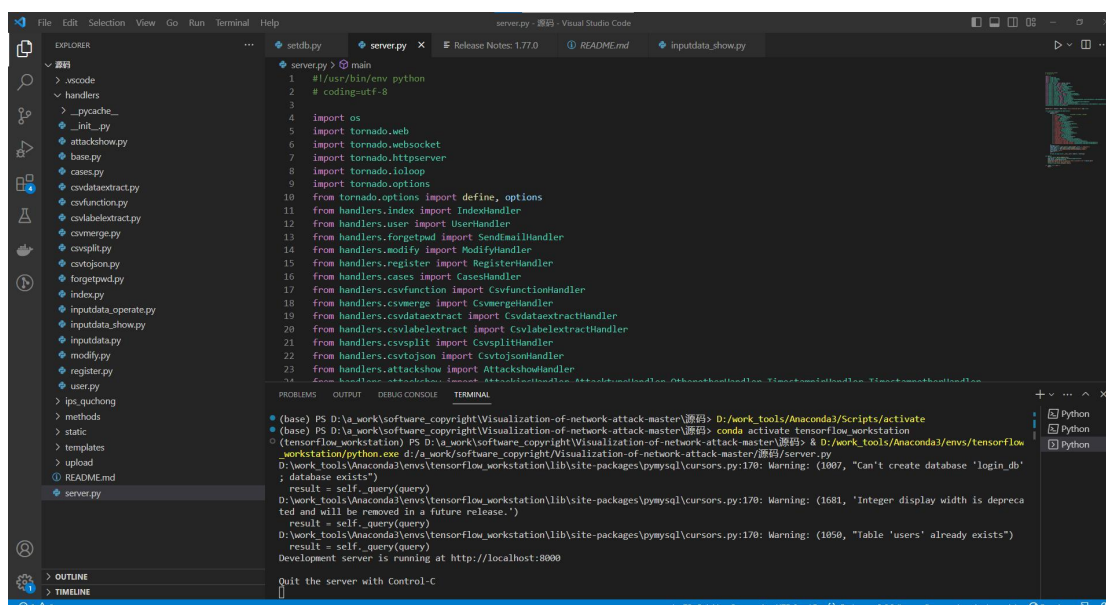
### 3. 系统编译环境

本系统开发期间编译平台的操作系统为 Windows 10 家庭中文版。硬件配置为：

处理器	Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz 2.90 GHz
机带 RAM	16.0 GB (15.7 GB 可用)

本系统使用 Visual Studio Code 进行开发，需要使用相同软件进行开发编译。

编译环境如下所示：



本系统使用的编程语言是 python，版本是 3.6.8。必需的 Python 程序库包括：

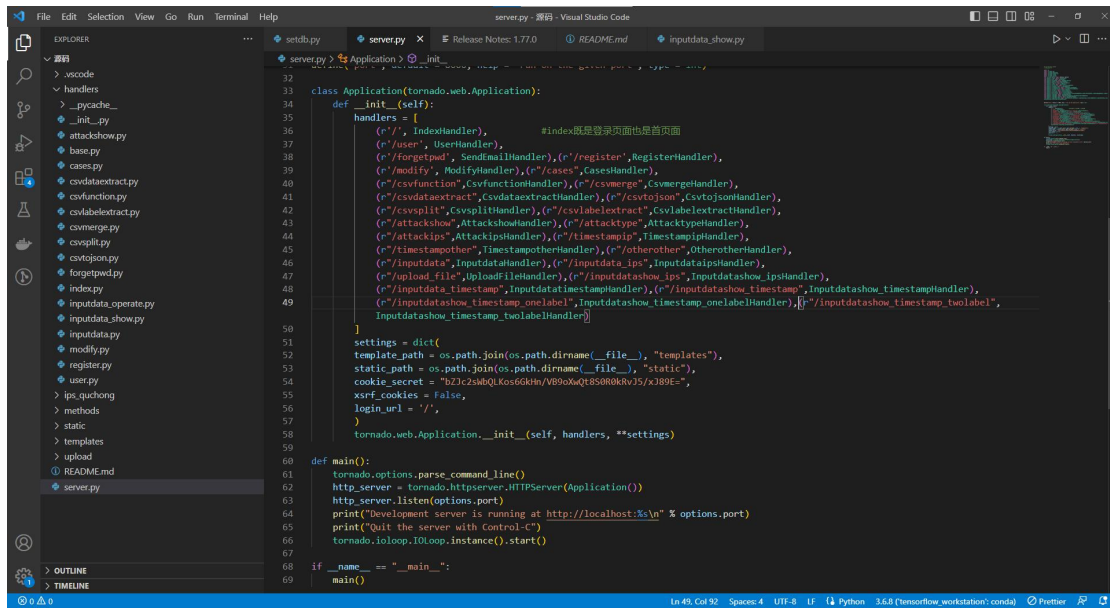
tornado, opencv-python, matplotlib, PIL, pymysql, hashlib, csv, numpy 等。

编译本系统需要对应版本的 Python 和相应的库函数。

本系统使用的数据库版本为 MySQL8.0.27，正常运行本系统需要对应版本的 mysql。

本系统前端部分需要浏览器适配，编译开发阶段测试用的浏览器为 Google Chrome，版本号为 106.0.5249.119。

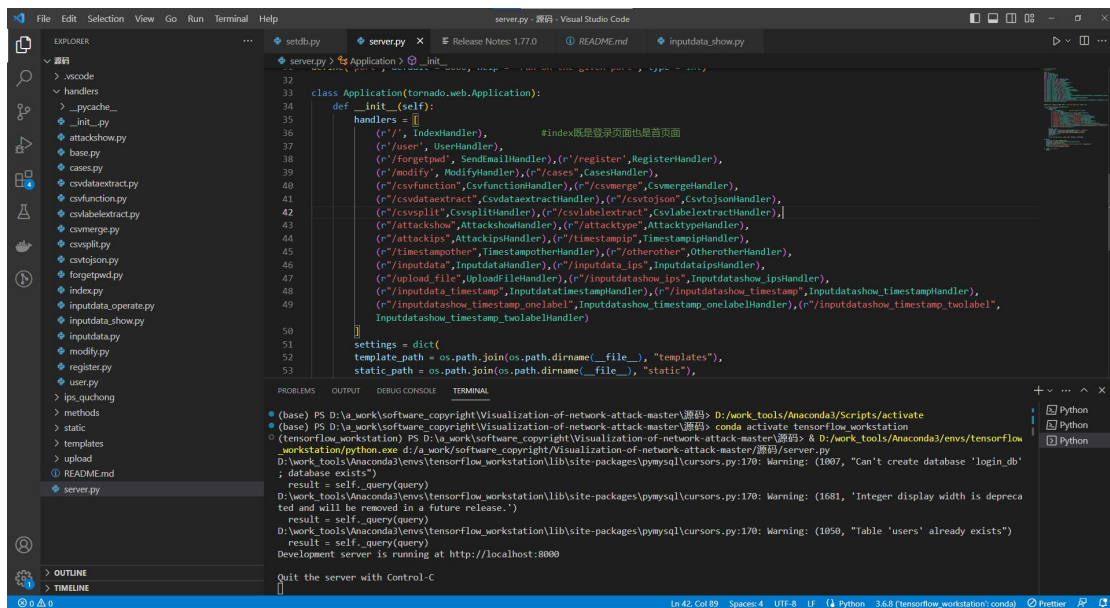
## 入口函数：



```

32
33 class Application(tornado.web.Application):
34     def __init__(self):
35         handlers = [
36             (r"/", IndexHandler), #index既是登录页面也是首页
37             (r"/user", UserHandler),
38             (r"/forgetpad", SendEmailHandler), (r"/register", RegisterHandler),
39             (r"/modify", ModifyHandler), (r"/cases", CasesHandler),
40             (r"/csvfunction", CsvfunctionHandler), (r"/csvmerge", CsvmergeHandler),
41             (r"/csvdataextract", CsvdataextractHandler), (r"/csvtojson", CsvtojsonHandler),
42             (r"/csvsplit", CsvsplitHandler), (r"/csvlabelextract", CsvlabelextractHandler),
43             (r"/attackshow", AttackshowHandler), (r"/attacktype", AttacktypeHandler),
44             (r"/attacksips", AttacksipsHandler), (r"/timestamp", TimestampHandler),
45             (r"/timestampother", TimestampotherHandler), (r"/otherother", OtherotherHandler),
46             (r"/inputdata", InputdataHandler), (r"/inputdata_ips", InputdataipsHandler),
47             (r"/upload_file", UploadFileHandler), (r"/inputdata_show_ips", Inputdata_show_ipsHandler),
48             (r"/inputdata_timestamp", Inputdata_timestampHandler), (r"/inputdata_show_timestamp", Inputdata_show_timestampHandler),
49             (r"/inputdata_show_timestamp_oneLabel", Inputdata_show_timestamp_oneLabelHandler), (r"/inputdata_show_timestamp_twoLabel",
50              Inputdata_show_timestamp_twoLabelHandler)
51         ]
52         settings = dict(
53             template_path = os.path.join(os.path.dirname(__file__), "templates"),
54             static_path = os.path.join(os.path.dirname(__file__), "static"),
55             cookie_secret = "b2c2abQkKos6dhnV89o8nqT8S8n8KvJ5/xJ89C-",
56             xsrf_cookies = False,
57             login_url = '/',
58         )
59         tornado.web.Application.__init__(self, handlers, **settings)
60
61 def main():
62     tornado.options.parse_command_line()
63     http_server = tornado.httpserver.HTTPServer(Application())
64     http_server.listen(options.port)
65     print("Development server is running at http://localhost:%s\n" % options.port)
66     print("Quit the server with Control-C")
67     tornado.ioloop.IOLoop.instance().start()
68
69 if __name__ == "__main__":
70     main()
  
```

## 编译运行成功画面：



```

(base) PS D:\work\software\copyright\Visualization-of-network-attack-master\源码> D:\work_tools\Anaconda3\Scripts\activate
(tensorflow workstation) PS D:\work\software\copyright\Visualization-of-network-attack-master\源码> conda activate tensorflow_workstation
D:\work_tools\Anaconda3\envs\tensorflow_workstation\lib\site-packages\pymysql\cursors.py:170: Warning: (1007, "Can't create database 'login_db' ; database exists")
  result = self._query(query)
D:\work_tools\Anaconda3\envs\tensorflow_workstation\lib\site-packages\pymysql\cursors.py:170: Warning: (1681, "Integer display width is deprecated and will be removed in a future release.")
  result = self._query(query)
D:\work_tools\Anaconda3\envs\tensorflow_workstation\lib\site-packages\pymysql\cursors.py:170: Warning: (1050, "Table 'users' already exists")
  result = self._query(query)
Development server is running at http://localhost:8000

Quit the server with Control-C
  
```

## 4. 系统配置说明

将系统源码放到 Windows 或 Linux 服务器平台上。配置好 Python 环境，安装相应的 Python 库，打开对应的端口号，配置相应的 MySQL 数据库，并根据实际情况进行其他配置。

由于 Tornado 提供了自己的 httpserver，因此运行和部署它与其他 PythonWeb 框架有点不同。不是配置一个 wsgi 容器来查找应用程序，而是编写一个 main() 启动服务器的函数。

```
def main():
    tornado.options.parse_command_line()
    http_server = tornado.httpserver.HTTPServer(Application())
    http_server.listen(options.port)

    print("Development server is running at http://localhost:%s" % options.port)
    print("Quit the server with Control-C")

    tornado.ioloop.IOLoop.instance().start()
```

配置操作系统或进程管理器以运行此程序以启动服务器。

由于 python 的 GIL（全局解释器锁），需要运行多个 python 进程以充分利用多 CPU 机器。通常，最好每个 CPU 运行一个进程。

supervisord 是用 Python 实现的一款很是实用的进程管理工具。使用 supervisor 守护进程，自动重启网站。

supervisor 配置文件中添加：

```
[program:tornado-8000]
command=python /var/www/main.py --port=8000
directory=/var/www
user=www-data
autorestart=true
redirect_stderr=true
stdout_logfile=/var/log/tornado.log
loglevel=info

[program:tornado-8001]
command=python /var/www/main.py --port=8001
directory=/var/www
user=www-data
autorestart=true
redirect_stderr=true
stdout_logfile=/var/log/tornado.log
loglevel=info
```



对于更复杂的部署，建议独立启动进程，并让每个进程侦听不同的端口。当每个进程使用不同的端口时，通常需要一个外部负载均衡器（如 `nginx`）向外部访问者提供一个地址。下面给出一个 `nginx` 的配置文件，`nginx` 和 `tornado` 服务器在同一台机器上运行，四台 `tornado` 服务器在端口 8000-8003 上运行：

```
user nginx;
worker_processes 1;

error_log /var/log/nginx/error.log;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
    use epoll;
}

http {
    # Enumerate all the Tornado servers here
    upstream frontends {
        server 127.0.0.1:8000;
        server 127.0.0.1:8001;
        server 127.0.0.1:8002;
        server 127.0.0.1:8003;
    }

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    access_log /var/log/nginx/access.log;

    keepalive_timeout 65;
    proxy_read_timeout 200;
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    gzip on;
    gzip_min_length 1000;
    gzip_proxied any;
    gzip_types text/plain text/html text/css text/xml
        application/x-javascript application/xml
        application/atom+xml text/javascript;
```

```

# Only retry if there was a communication error, not a timeout
# on the Tornado server (to avoid propagating "queries of death"
# to all frontends)
proxy_next_upstream error;

server {
    listen 80;

    # Allow file uploads
    client_max_body_size 50M;

    location ^~ /static/ {
        root /var/www;
        if ($query_string) {
            expires max;
        }
    }
    location = /favicon.ico {
        rewrite (.*?) /static/favicon.ico;
    }
    location = /robots.txt {
        rewrite (.*?) /static/robots.txt;
    }

    location / {
        proxy_pass_header Server;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Scheme $scheme;
        proxy_pass http://frontends;
    }
}

```

静态文件已经在程序中指定了路径：

```

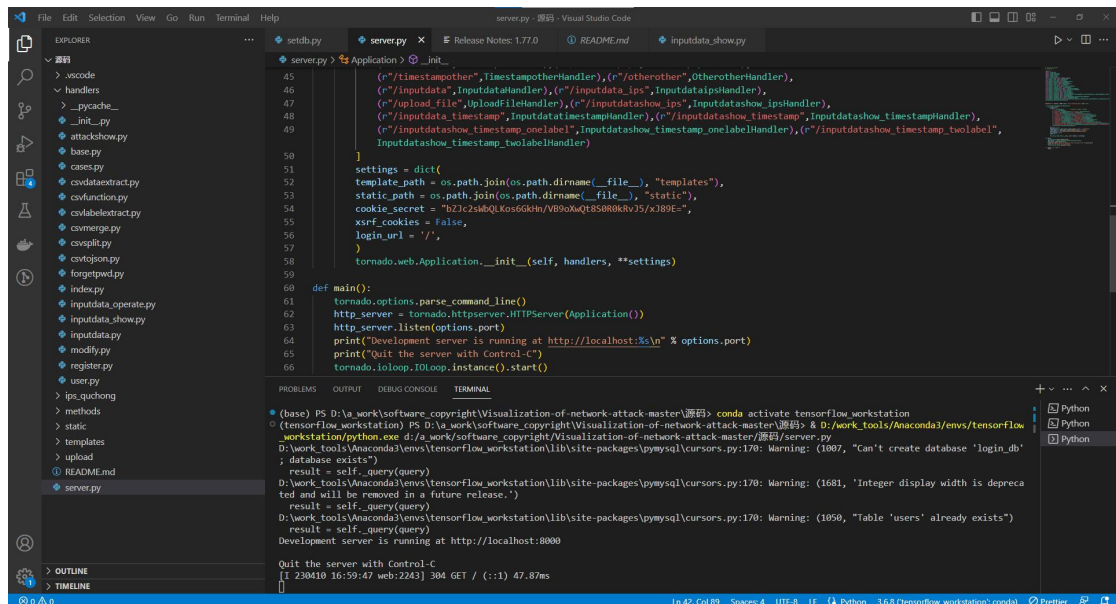
settings = dict(
    template_path = os.path.join(os.path.dirname(__file__), "templates"),
    static_path = os.path.join(os.path.dirname(__file__), "static"),
    cookie_secret = "bZJc2sWbQLKos6GkHn/VB9oXwQt8S0R0kRvJ5/xJ89E=",
    xsrf_cookies = False,
    login_url = '/',
)
tornado.web.Application.__init__(self, handlers, **settings)

```

## 5. 系统运行说明

在部署好服务器端之后，就可以运行服务器端程序，启动服务器。

服务器端运行成功效果如图所示：



```
server.py
45 (r"/timestamp", timestampHandler), (r"/otherother", otherotherHandler),
46 (r"/inputdata", inputdataHandler), (r"/inputdata_ips", inputdata_ipsHandler),
47 (r"/upload_file", upload_fileHandler), (r"/inputdata_show_ips", inputdata_show_ipsHandler),
48 (r"/inputdata_timestamp", inputdata_timestampHandler), (r"/inputdata_show_timestamp", inputdata_show_timestampHandler),
49 (r"/inputdata_show_timestamp_one_label", inputdata_show_timestamp_one_labelHandler), (r"/inputdata_show_timestamp_two_label",
    inputdata_show_timestamp_two_labelHandler)
50
51 settings = dict(
52     template_path = os.path.join(os.path.dirname(__file__), "templates"),
53     static_path = os.path.join(os.path.dirname(__file__), "static"),
54     cookie_secret = "b2c2smbQLKos6Gdhn/V89oXwQt850R0kVJ5/xJ89E=",
55     xsrf_cookies = False,
56     login_url = '/',
57 )
58 tornado.web.Application._init__(self, handlers, **settings)
59
60 def main():
61     tornado.options.parse_command_line()
62     http_server = tornado.httpserver.HTTPServer(Application())
63     http_server.listen(options.port)
64     print("Development server is running at http://localhost:8000")
65     print("Quit the server with Control-C")
66     tornado.ioloop.IOLoop.instance().start()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(base) PS D:\a\work\software\copyright\Visualization-of-network-attack-master\源码> conda activate tensorflow_workstation
(tensorflow_workstation) PS D:\a\work\software\copyright\Visualization-of-network-attack-master\源码> D:\work_tools\Anaconda3\envs\tensorflow_workstation\python.exe d:/a/work/software_copyright/Visualization-of-network-attack-master/源码/server.py
D:\work_tools\Anaconda3\envs\tensorflow_workstation\lib\site-packages\pymysql\cursors.py:170: Warning: (1007, "Can't create database 'login_db'
; database exists.")
  result = self._query(query)
D:\work_tools\Anaconda3\envs\tensorflow_workstation\lib\site-packages\pymysql\cursors.py:170: Warning: (1681, 'Integer display width is deprec
ated and will be removed in a future release.')
  result = self._query(query)
D:\work_tools\Anaconda3\envs\tensorflow_workstation\lib\site-packages\pymysql\cursors.py:170: Warning: (1050, "Table 'users' already exists")
  result = self._query(query)
Development server is running at http://localhost:8000
Quit the server with Control-C
[I 230410 16:59:47 web:2243] 304 GET / (:::1) 47.87ms
```

此时服务器已经启动，端口号为 8000。运行服务器本地登录可以访问 <http://127.0.0.1:8000>，内网登录可以访问 <http://内网 IP:8000>，外网登录可以访问 <http://外网 IP:8000>。

访问登录界面如下所示：



## 6. 系统操作说明

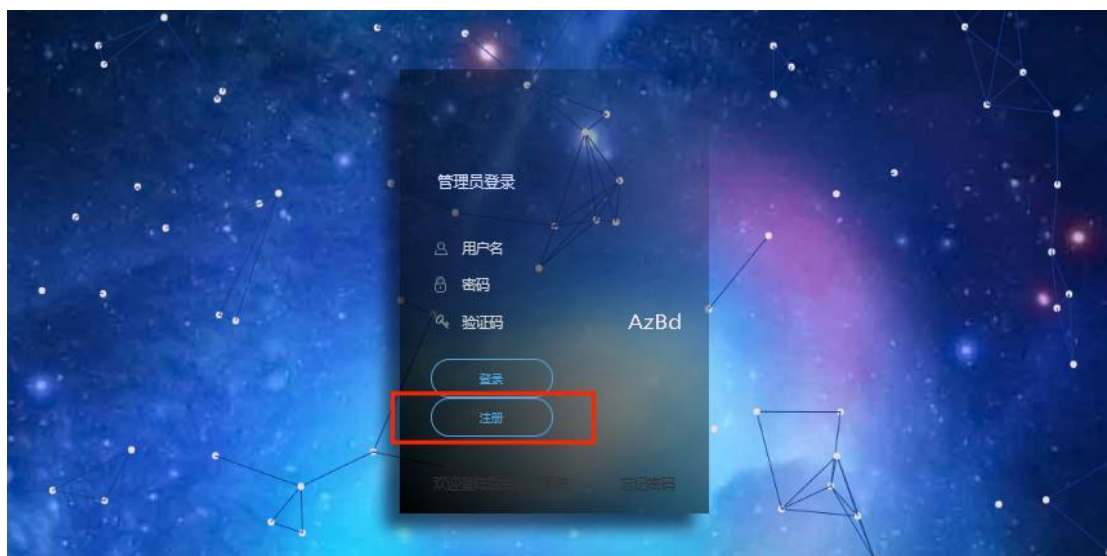
### 6.1. 注册登录操作

系统服务器端运行成功之后，用户即可正常访问前端页面并执行相应操作。

本系统的功能操作需要在用户登录的情况下进行，因此用户可以选择注册账号或是使用已有账号进行登录。注册登录部分的操作说明详见下文。

注：用户可以选择使用 PC 或是移动端设备登录系统进行相关操作，访问时注意保持联网，以保证功能正常运行。

首先对于首次使用的用户，注册账号需要在首页点击注册按钮：



即可进入注册页面，需要用户提供用户名，邮箱，密码和密码确认等信息：

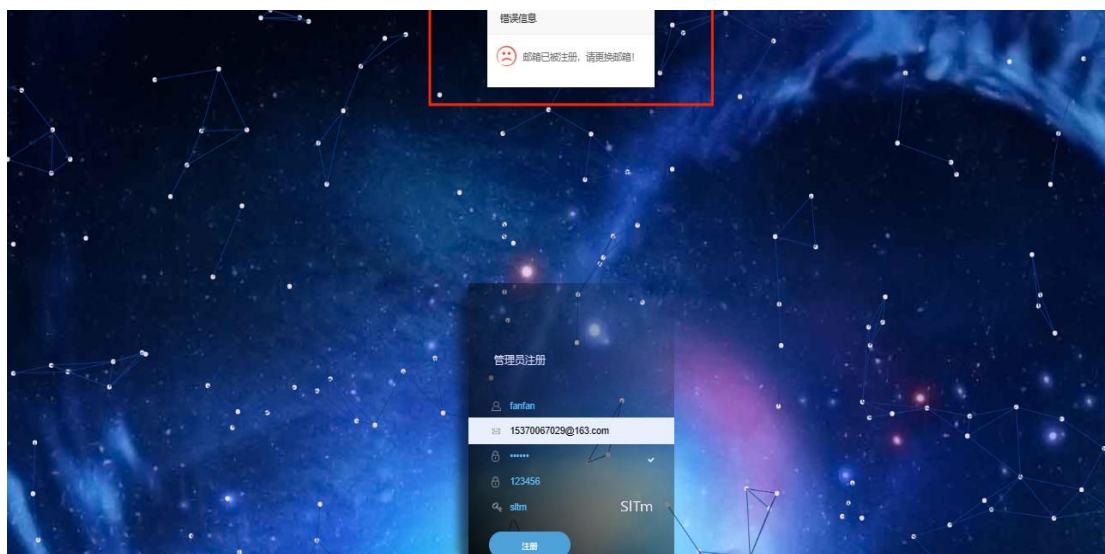
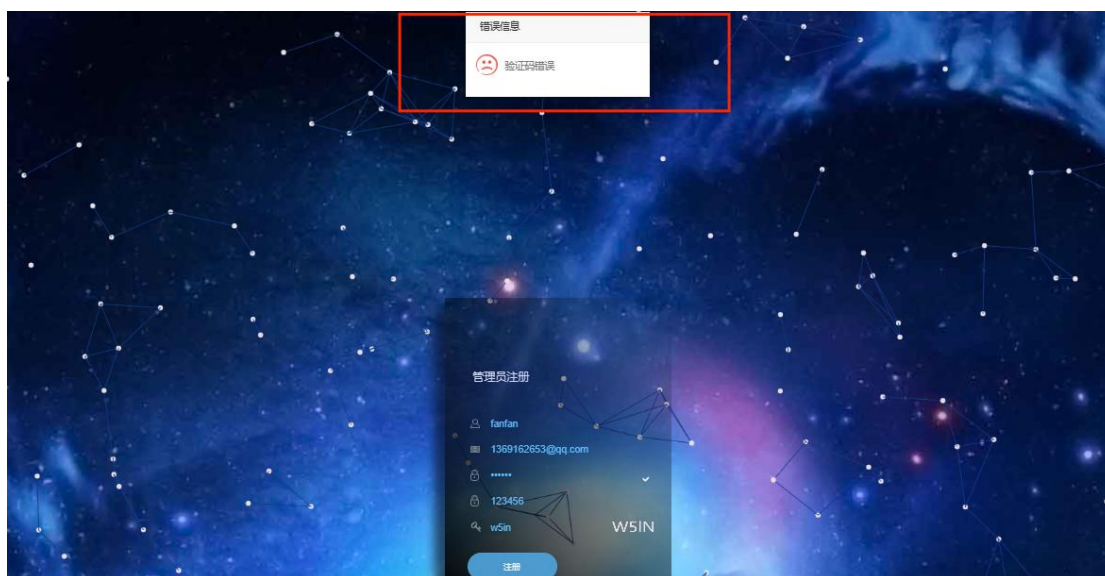




输入对应信息后，即可点击注册按钮，完成注册。



若注册不成功，会有对应的提示，此时根据提示进行相关修改即可。



注册成功的返回信息：



此时用户已经注册成功，即可返回登录（对于已经有帐号的用户，可直接执行这一步）。



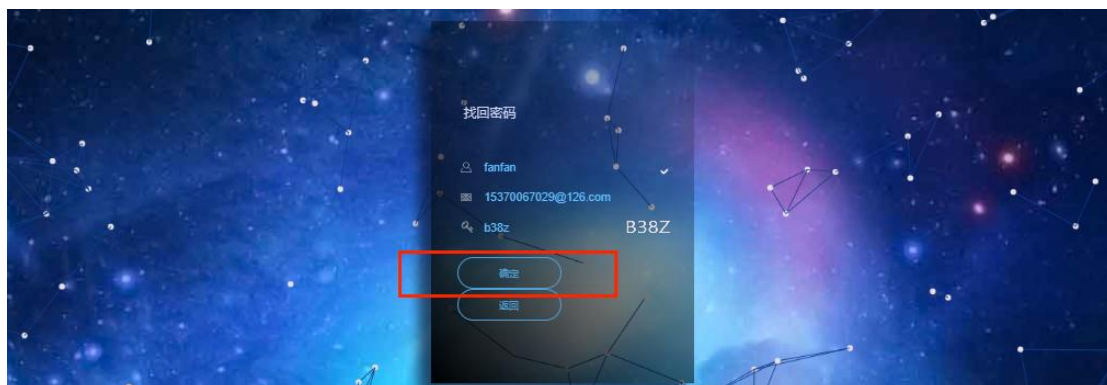
## 6.2. 找回密码操作

对于忘记账号密码的用户，系统提供找回密码功能，如下点击忘记密码按钮。





此时进入找回密码页面，根据要求输入用户名和邮箱，并点击确定按钮发送找回密码邮件。



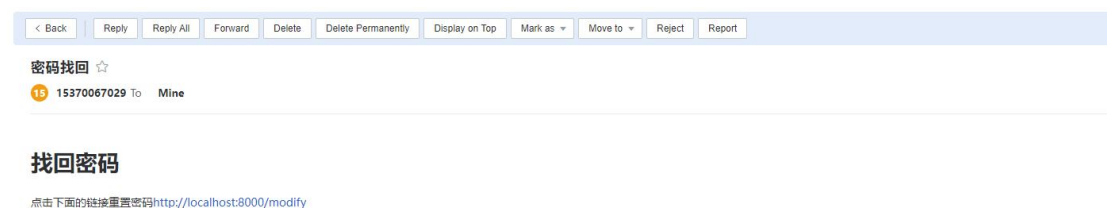
邮件发送过程中出现如下提示，可稍等几秒钟。



邮件发送成功状态提示信息如图：



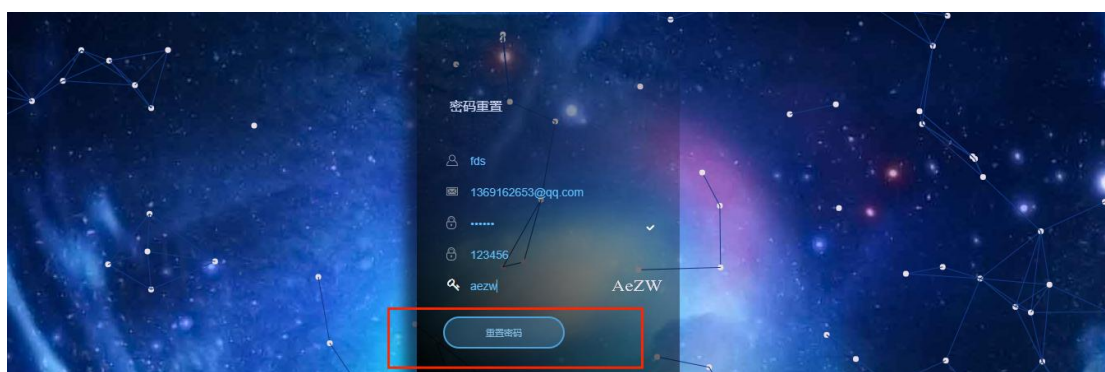
此时检查邮箱中是否收到密码重置的邮件。



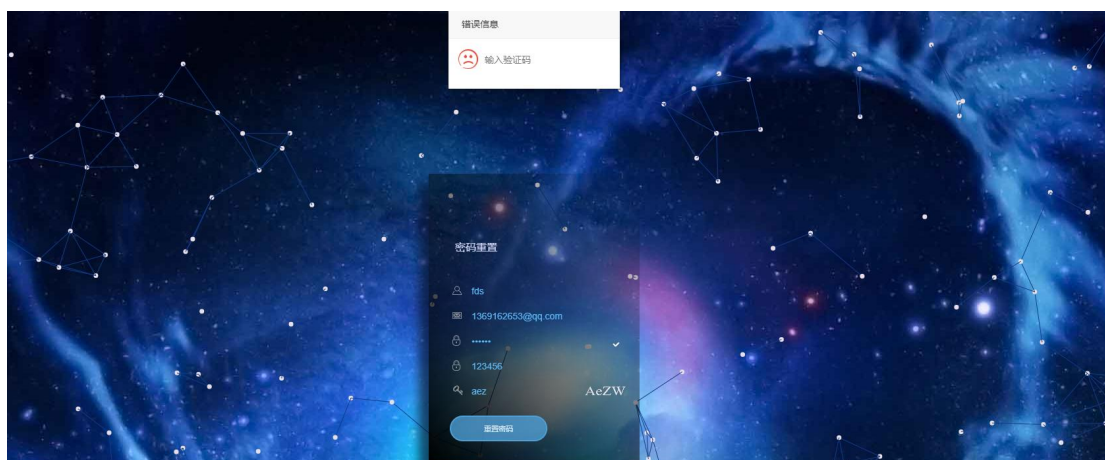
点击密码重置链接，进入密码重置页面：



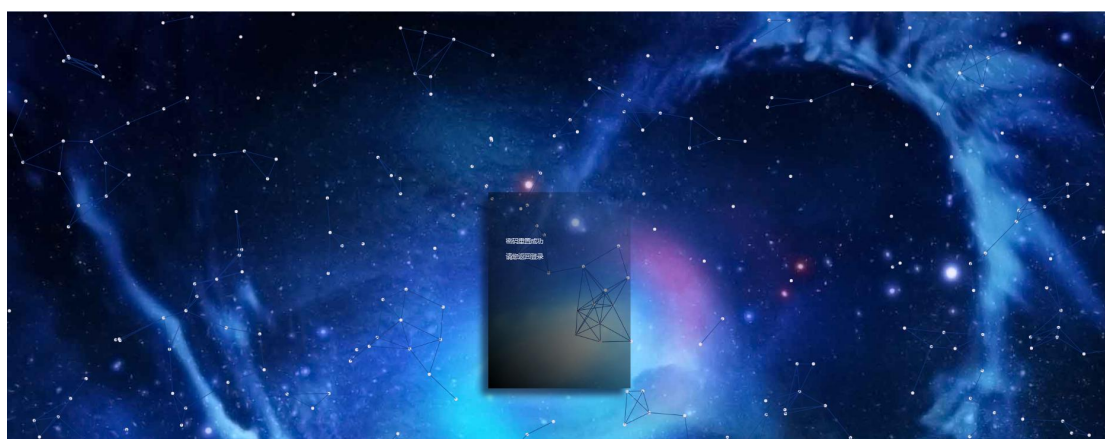
输入对应的用户名，邮箱和密码，进行密码重置：



若遇到错误提示，请按照提示修改对应信息。



密码重置成功。





使用重置的新密码进行登录，成功登录。



登录之后进入主页面，如下所示：



### 6.3. 网络攻击数据显示

用户可根据需求查看网络流量数据，数据来源于 CIC-IDS-2017 数据集，针对这一经典网络攻击数据集进行数据可视化分析操作。



进入网络攻击数据显示模块。

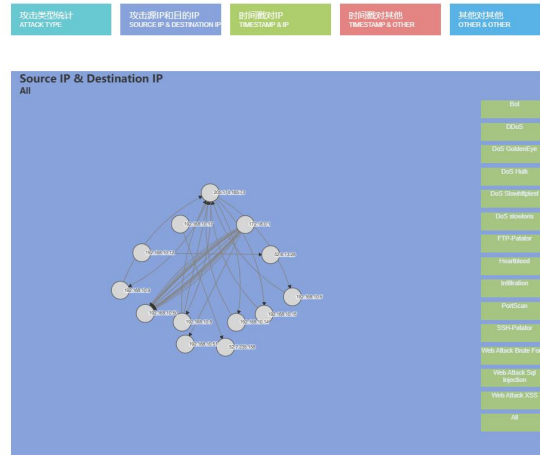


根据需要执行不同数据的可视化操作。

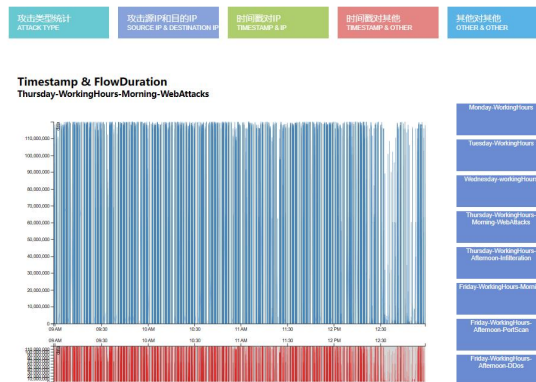
### (1) 攻击类型统计（饼状图和柱状图）



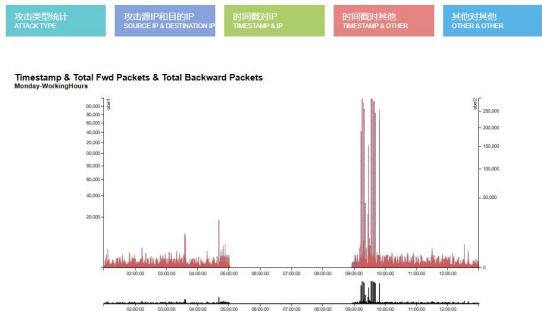
### (2) 攻击源 IP 和目的 IP（力导向图）



### (3) 时间戳对其他，根据数据集中的特征标签来命名坐标轴（折线图）



### (4) 其他对其他，根据数据集中的特征标签来命名坐标轴（折线图）

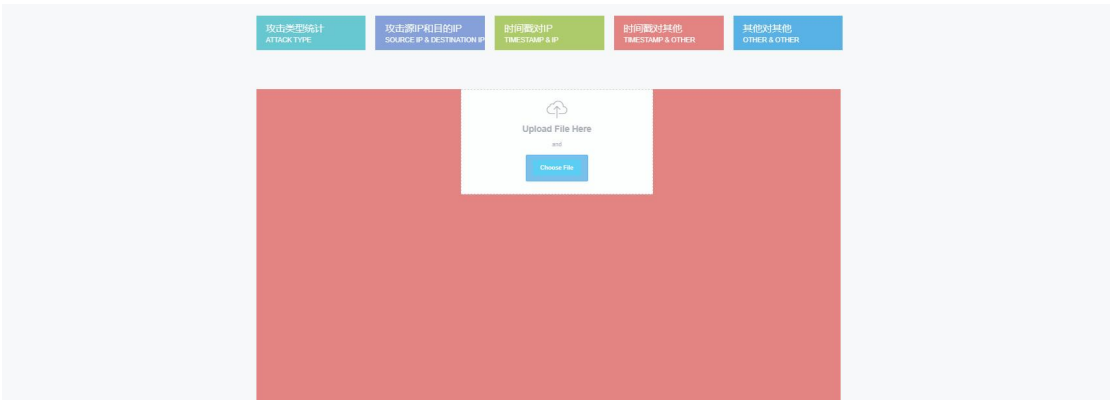


6.4. 上传数据操作

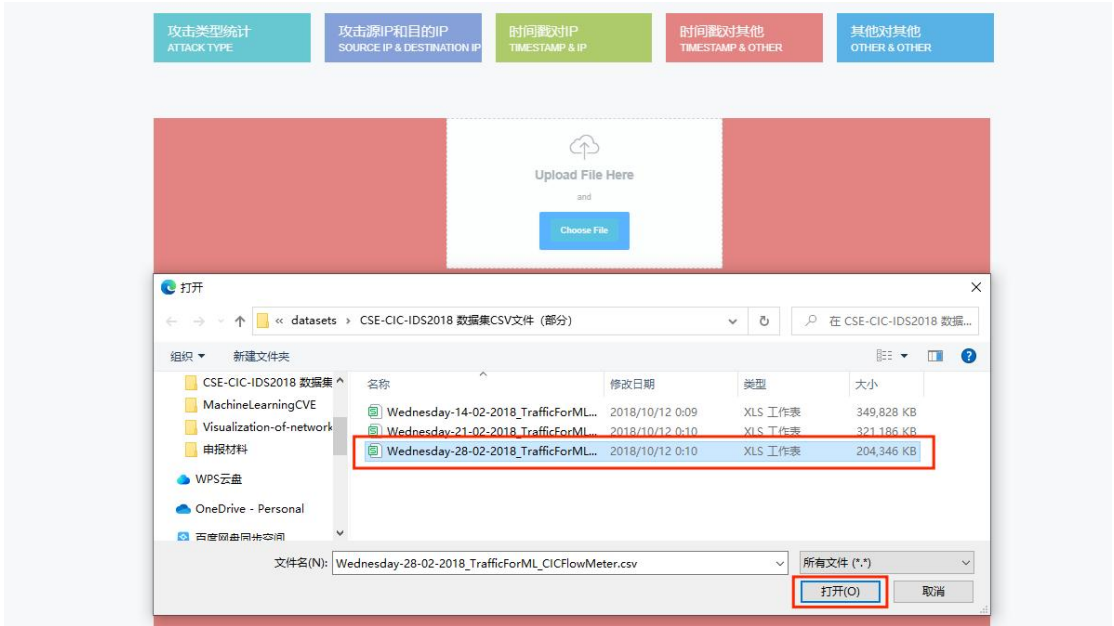
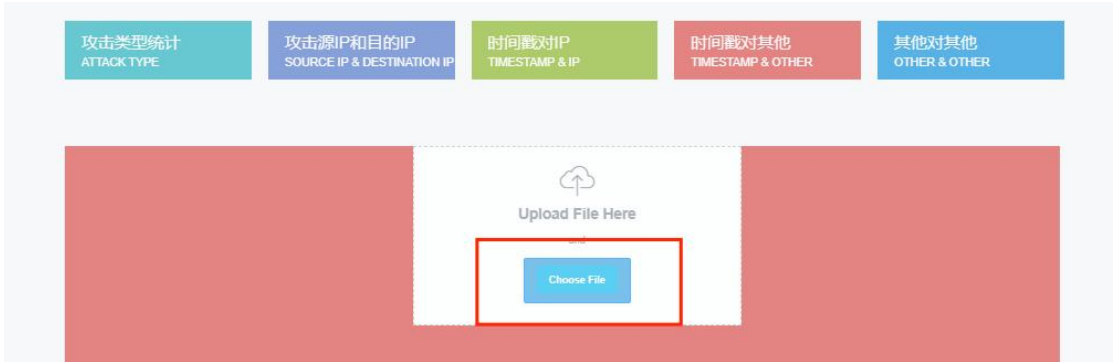
针对用户上传的网络流量数据进行可视化分析。



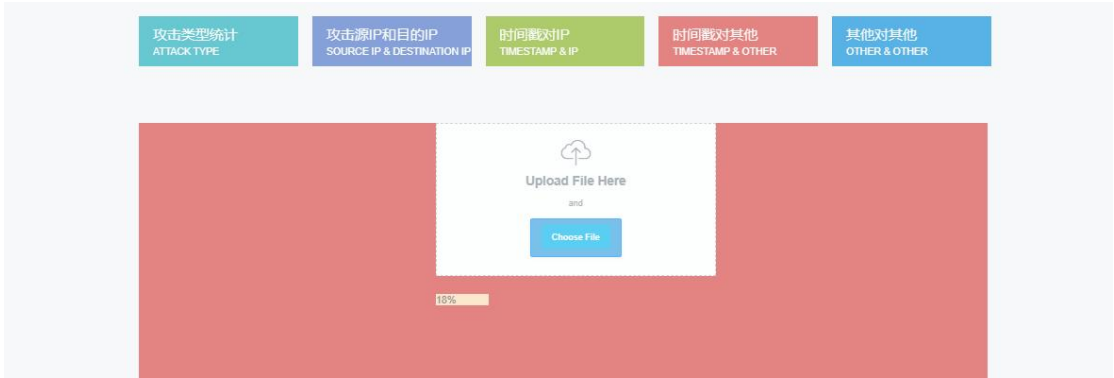
上传数据页面如下所示：



点击“Choose File”按钮选择文件上传：



等待上传。



上传成功。



点击“查看图表”按钮查看图表。



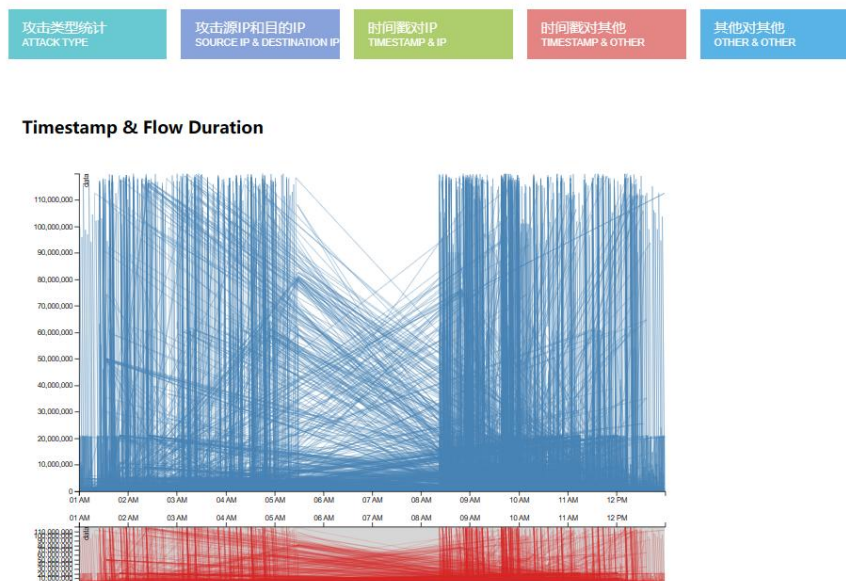
选择对应标签进行可视化。



选择单列标签进行可视化，点击确定。

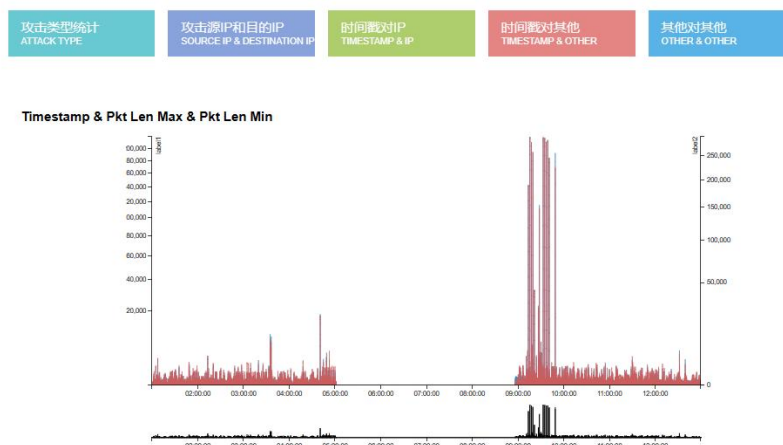


获取单列标签对应的可视化图表。

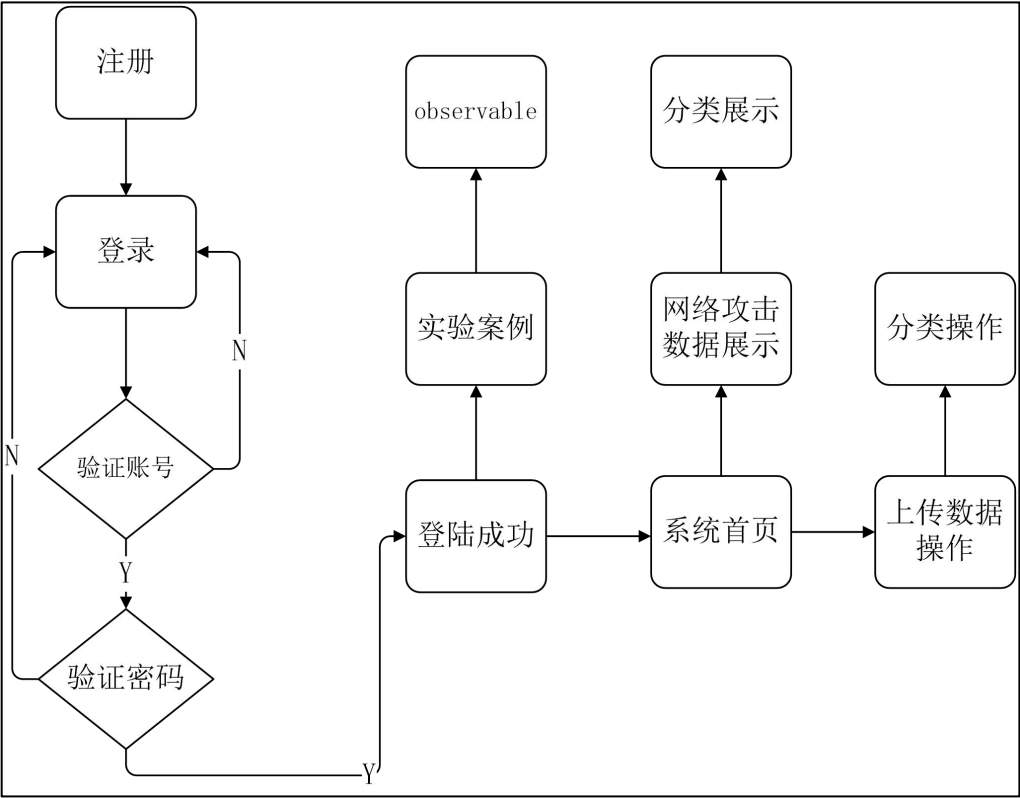


选择双列标签进行可视化，点击确定。

获取双列标签对应的可视化图表。



6.5. 操作流程示意图





## 7. 系统后台监测

系统后台运行成功提示：

```
(tensorflow_workstation) PS D:\a_work\software\copyright\Visualization-of-network-attack-master\源码> & D:/work_tools/Anaconda3/envs/tensorflow_workstation/python.exe d:/a_work/software/copyright/Visualization-of-network-attack-master/源码/server.py
D:\work_tools\Anaconda3\envs\tensorflow_workstation\lib\site-packages\pymysql\cursors.py:170: Warning: (1007, "Can't create database 'login_db'; database exists")
  result = self._query(query)
D:\work_tools\Anaconda3\envs\tensorflow_workstation\lib\site-packages\pymysql\cursors.py:170: Warning: (1681, 'Integer display width is deprecated and will be removed in a future release.')
  result = self._query(query)
D:\work_tools\Anaconda3\envs\tensorflow_workstation\lib\site-packages\pymysql\cursors.py:170: Warning: (1050, "Table 'users' already exists")
  result = self._query(query)
Development server is running at http://localhost:8000

Quit the server with Control-C
```

进入登录界面：

```
[I 230412 13:31:47 web:2243] 200 GET / (:::1) 20.94ms
```

登录：

```
[I 230412 13:32:39 web:2243] 200 POST / (:::1) 1.00ms
[I 230412 13:32:44 web:2243] 304 GET /user (:::1) 11.97ms
```

进行数据可视化查看：

```
[I 230412 13:33:43 web:2243] 304 GET /attackshow (:::1) 23.94ms
[I 230412 13:33:47 web:2243] 304 GET /attacktype (:::1) 1.99ms
[I 230412 13:34:05 web:2243] 304 GET /attacktips (:::1) 2.99ms
[I 230412 13:34:07 web:2243] 304 GET /timestampip (:::1) 2.00ms
[I 230412 13:34:09 web:2243] 304 GET /timestampother (:::1) 1.99ms
[I 230412 13:34:12 web:2243] 200 GET /otherother (:::1) 2.00ms
```

上传文件进行数据可视化操作：

```
[I 230412 13:34:43 web:2243] 304 GET /inputdata (:::1) 1.99ms
[I 230412 13:34:45 web:2243] 304 GET /inputdata.timestamp (:::1) 2.00ms
[I 230412 13:34:49 web:2243] 200 POST /upload_file (:::1) 279.50ms
[I 230412 13:34:49 web:2243] 200 POST /upload_file (:::1) 301.43ms
[I 230412 13:34:49 web:2243] 200 POST /upload_file (:::1) 318.16ms
[I 230412 13:34:50 web:2243] 200 POST /upload_file (:::1) 296.76ms
[I 230412 13:34:50 web:2243] 200 POST /upload_file (:::1) 290.42ms
[I 230412 13:34:50 web:2243] 200 POST /upload_file (:::1) 317.69ms
[I 230412 13:34:51 web:2243] 200 POST /upload_file (:::1) 444.04ms
[I 230412 13:34:51 web:2243] 200 POST /upload_file (:::1) 266.32ms
[I 230412 13:34:51 web:2243] 200 POST /upload_file (:::1) 286.39ms
[I 230412 13:34:52 web:2243] 200 POST /upload_file (:::1) 297.24ms
[I 230412 13:34:52 web:2243] 200 GET /upload_file?task_id=uu_1gtpvb3g71pi0bli4r455kd90&filename=Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv (:::1) 722.87ms
[I 230412 13:34:53 web:2243] 304 GET /inputdatashow.timestamp?filename=Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv (:::1) 2.99ms
[W 230412 13:34:53 web:2243] 404 GET /static/fonts/PFSquareSansPro-Bold.otf (:::1) 0.00ms
static/onenlabel_with_timestamp/Wednesday-28-02-2018_TrafficForML_CICFlowMeter 目录已存在
[I 230412 13:35:14 web:2243] 200 POST /inputdatashow.timestamp_onenlabel (:::1) 13387.70ms
[I 230412 13:35:14 web:2243] 200 GET /inputdatashow.timestamp_onenlabel?csvoutputfile=static/onenlabel_with_timestamp/Wednesday-28-02-2018_TrafficForML_CICFlowMeter\TimestampBwdIATTot.csv&timestamp_format=%d/%m/%Y%20%I:%M:%S&chosedlabel=Bwd%20IAT%20Tot (:::1) 2.99ms
[I 230412 13:35:15 web:2243] 200 GET /static/onenlabel_with_timestamp/Wednesday-28-02-2018_TrafficForML_CICFlowMeter/TimestampBwdIATTot.csv (:::1) 341.09ms
```



## 8. 部分功能支持说明

### 8.1. 数据可视化

在实现 PC 端数据可视化功能时使用的可视化基础是 D3.js，服务器端选用 Python 的 Tornado 框架，数据库选用 MySQL8.0。

#### (1) D3.js

D3.js 是一个可以通过数据来操作文档的 JavaScript 库。D3.js 可以通过使用 HTML、SVG 和 CSS 实现数据可视化。D3.js 严格遵循 Web 标准，使其能够方便地与现代主流浏览器兼容，避免对某些框架的依赖。项目中需要一些特殊的图表，如 IP、数据库拓扑等，它可以帮助快速实现这些图表效果。D3.js 测试了当下主流的各个浏览器，其大部分组件可以在旧的浏览器运行。

#### (2) Tornado

Tornado 是非阻塞式的轻量级的 Web 服务器，使用 Python 编写。其非阻塞的方式和对 epoll 的运用使其速度比较快。Tornado 可以解决高并发，每秒可以处理数以千计的连接，超过了很多主流的 Web 框架，因此 Tornado 对于实时 Web 服务来说是一个较为理想的 Web 框架。

### 8.2. 找回密码

本系统中找回密码功能主要使用邮件验证的方式。用户向服务器提交重置密码的请求，服务器会向用户的邮箱发送重置密码邮件，用户点击重置链接即可重置账号密码。重置密码的邮箱配置需要对应修改为管理员的配置。

```
my_sendername = ""      # 发件人账号名 例如"fengdushuo"
my_sender=""           # 发件人邮箱账号 例如"1224556@163.com"
my_pass = ""           # 发件人邮箱密码 例如"123456"
mail_host = ""         # SMTP服务器 例如"smtp.163.com"
```

重置密码邮箱配置的对应该函数：

```
def send_email(message,subject,to_address):
    ret=True
    try:
        msg=MIMEText(message,'html','utf-8')
        msg['From']=formataddr([my_sendername,my_sender]) # 括号里的对应发件人邮
        箱昵称、发件人邮箱账号
        msg['To']=formataddr(to_address) # 括号里的对应收件人邮
        箱昵称、收件人邮箱账号
        msg['Subject']= Header(subject, 'utf-8').encode() # 邮件的主题

        server=smtplib.SMTP_SSL("smtp.163.com", 465) # 发件人邮箱中的SMTP服
        务器，端口是25
        server.login(my_sender, my_pass) # 括号中对应的是发件人
        邮箱账号、邮箱密码
        server.sendmail(my_sender,[to_address[1],****],msg.as_string()) #
        括号中对应的是发件人邮箱账号、收件人邮箱账号、发送邮件
        server.quit() # 关闭连接
    except smtplib.SMTPException as e: # 如果 try 中的语句没有执行，则会执行下面
    的
        ret=False
        print(e)
        ret=False
    return ret
```

## 9. 系统数据库说明

数据库使用的是 mysql8.0.27，如下：

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 67
Server version: 8.0.27 MySQL Community Server - GPL
```

创建一个数据库 login\_db 对应于注册登录的用户 login\_db 数据库：

```
mysql> show tables;
+-----+
| Tables_in_login_db |
+-----+
| users               |
+-----+
1 row in set (0.01 sec)
```

users 数据表中字段详情：

```
mysql> SHOW FULL COLUMNS FROM users;
```

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
id	int	NULL	NO	PRI	NULL		select,insert,update,references	
username	varchar(20)	utf8mb4_0900_ai_ci	YES		NULL		select,insert,update,references	
password	varchar(40)	utf8mb4_0900_ai_ci	YES		NULL		select,insert,update,references	
email	varchar(40)	utf8mb4_0900_ai_ci	YES		NULL		select,insert,update,references	

```
4 rows in set (0.01 sec)
```

## 10. 配置脚本文件

### 10.1. 数据库配置文件

```
import pymysql
mysqluser="root"
mysqlpassword="19971130"
logindatabase="login_db"
imagedatabase="faceimage"
hostsrc="localhost"
conn =
pymysql.connect(host=hostsrc,port=3306,user=mysqluser,password="19971130",charset='utf8')
cursor = conn.cursor()
createloginsql="CREATE DATABASE IF NOT EXISTS "+logindatabase
createfaceimgsql="CREATE DATABASE IF NOT EXISTS "+imagedatabase
cursor.execute(createfaceimgsql)
cursor.execute(createloginsql)
logconnect = pymysql.connect(      #连接数据库服务器
    user=mysqluser,                #本地 mysql 用户名
    password="19971130",           #本地 MySQL 密码
    host=hostsrc,
    port=3306,
    db=logindatabase,
    charset="utf8"
)
imagconnect = pymysql.connect(      #连接数据库服务器
    user=mysqluser,
    password="19971130",
    host=hostsrc,
    port=3306,
    db=imagedatabase,
    charset="utf8"
)
loginconnect = imagconnect          #连接数据库服务器
loginconn = loginconnect.cursor()   #创建操作游标
loginconn.execute("create table IF NOT EXISTS students(id int(11),stuname
varchar(20),stuid varchar(20)PRIMARY KEY,imagedata MEDIUMBLOB)")
loginconnect = logconnect          #连接数据库服务器
loginconn = loginconnect.cursor()   #创建操作游标
loginconn.execute("CREATE TABLE IF NOT EXISTS users (id int(11) PRIMARY
KEY ,username VARCHAR(20),password VARCHAR(40),email VARCHAR(40))")
```

## 10.2. 邮箱配置文件

```
import smtplib
from email.mime.text import MIMEText
from email.utils import formataddr
from email.header import Header

my_sendername = ""      # 发件人账号名 例如“fengdushuo”
my_sender=""           # 发件人邮箱账号 例如“1224556@163.com”
my_pass = ""           # 发件人邮箱密码 例如“123456”
mail_host = ""         # SMTP 服务器 例如“smtp.163.com”

def send_email(message,subject,to_address):
    ret=True
    try:
        msg=MIMEText(message,'html','utf-8')
        msg['From']=formataddr([my_sendername,my_sender])  # 括号里的对
        # 应发件人邮箱昵称、发件人邮箱账号
        msg['To']=formataddr(to_address)                  # 括号里的对应
        # 收件人邮箱昵称、收件人邮箱账号
        msg['Subject']= Header(subject, 'utf-8').encode() # 邮件的主题

        server=smtplib.SMTP_SSL("smtp.163.com", 465)      # 发件人邮箱中
        # 的 SMTP 服务器，端口是 25
        server.login(my_sender, my_pass)                   # 括号中对应的是
        # 发件人邮箱账号、邮箱密码
        server.sendmail(my_sender,[to_address[1],"***"],msg.as_string())
        # 括号中对应的是发件人邮箱账号、收件人邮箱账号、发送邮件
        server.quit()  # 关闭连接
    except smtplib.SMTPException as e:
        print(e)
        ret=False
    return ret

# 测试 sendemail
# token="khsadlkhdkajdkhksdbanmdbadabhdjasdhsnmj"
# message =""<h1>找回密码</h1>点击下面的链接重置密码<a
# href="http://127.0.0.1:8000/modify?token=""+"token+">http://127.0.0.
# 1:8000/modify?token=""+"token
# ret=send_email(message,"修改密码",["fengdushuo","1369162653@qq.com"])
# if ret:
#     print("邮件发送成功")
# else:
#     print("邮件发送失败")
```