PiScan

简介

对于给定的ip段与port进行扫描， 并对探测得到的端口进行协议、指纹、设备、蜜罐识别， 以json形式输出。

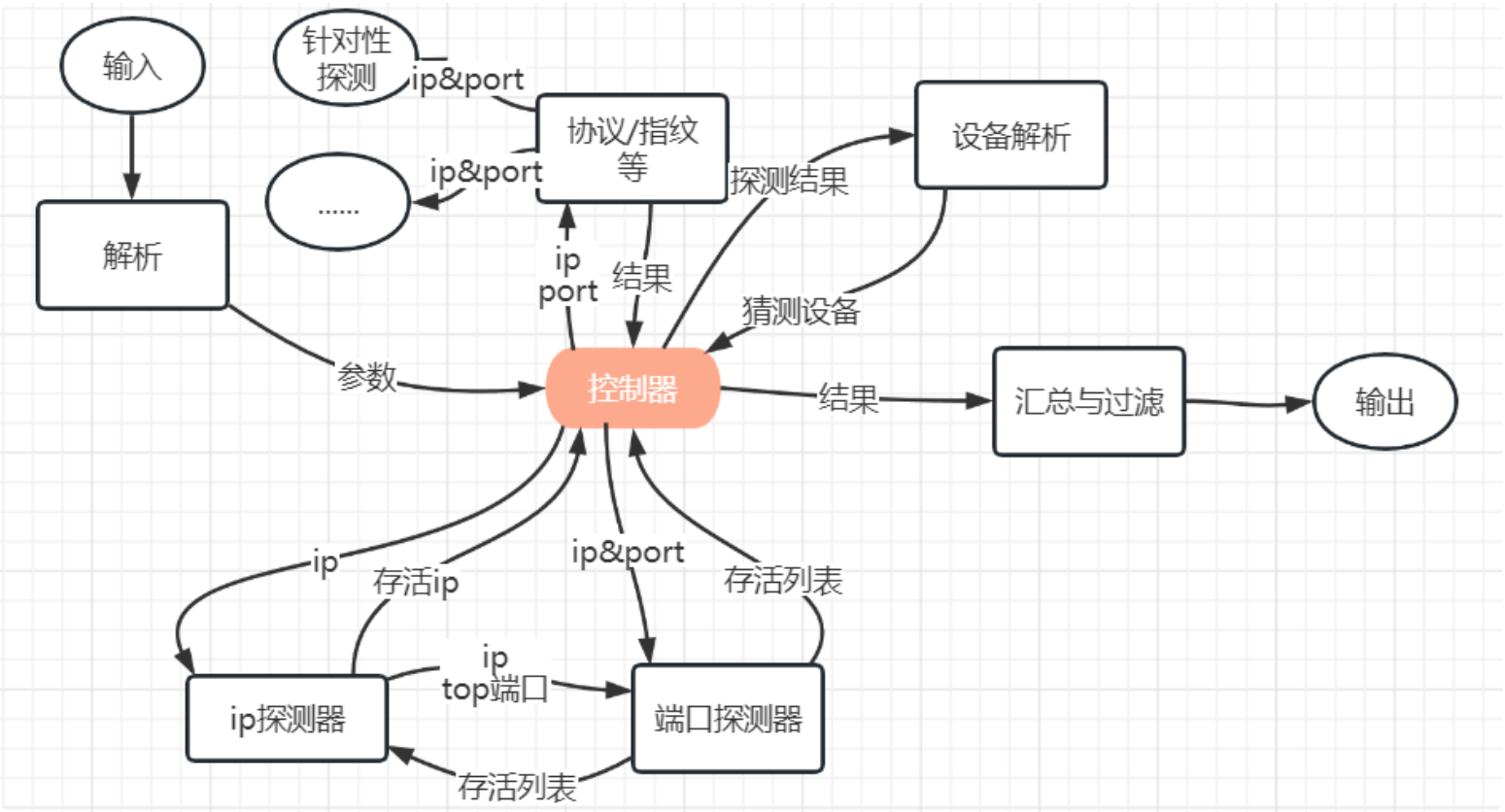
JSON

```JSON
{
  "113.30.191.68": {
    "services": [
      {
        "port": 1022,
        "protocol": "ssh", //协议
        "service_app": ["openssh/7.4"] //指纹
      },
       ...
    ],
    "deviceinfo": null, //设备
    "honeypot": ["2222/kippo"] //蜜罐
  },
  "ip": {...}
}
```

示例

```
python main.py -i YOURIP -o OUTFILE
```

模块



1. 解析器

   接收输入， 解析为参数并传给控制器

```
usage: main.py [-h] [-i IP] [-o OUTPUT] [-tp TOP_PORT_NUM]
[-atp ALIVE_TOP_PORT_NUM] [--proxy PROXY] [-t THREADS]

optional arguments:
  -h, --help            show this help message and exit
  -i IP, --ip IP        IP range to scan (default
'211.22.90.1,211.22.90.152')
  -o OUTPUT, --output OUTPUT
                        Output filename
  -tp TOP_PORT_NUM, --top-port-num TOP_PORT_NUM
                        scan port(default top 1000)
  -atp ALIVE_TOP_PORT_NUM, --alive-top-port-num
ALIVE_TOP_PORT_NUM
                        scan port when alive(default top
2500)
  -t THREADS, --threads THREADS
                        Number of threads (default 1750)
```

2. ip探测器

   采用多种技术判断：

   - icmp包
   - 与端口探测器结合， 当探测到端口开放时判定存活

3. 端口探测器

   - syn扫描

4. 协议&指纹

   nmap进行扫描， 同时加入我们的自定义指纹逻辑

Python

```python
Port = int
Protocal = str
Service = str, Version
def api(ip: str, ports: List[Port]) -> List[Tuple[Port, Protocal, List[Service]]]: # 蜜罐将放入Service中
  pass

# 指纹库: Rule(protocal, service, Union[hit_regex, hash], Union[version_regex, version], send_content_id)
#         Send_content(send_content_id, content)
```

5. 设备识别

   根据之前扫描出的结果判断设备

6. 过滤器

   去除不在目标列表中的结果，汇总为json