目录

**[JKscan设计说明书 1](#_Toc20155)**

**[一、 引言 1](#_Toc30531)**

[1.1 编写目的 1](#_Toc9474)

[1.2 应用功能 1](#_Toc14189)

**[二、 技术框架 2](#_Toc14378)**

[2.1 技术分析 2](#_Toc11560)

[2.2 模块实现 3](#_Toc19196)

**[三、 效果展示 4](#_Toc5061)**

[3.1 使用说明 4](#_Toc18572)

[3.2 实例展示 5](#_Toc23297)

[3.3 性能测试 7](#_Toc28749)

JKscan设计说明书

# 引言

## 编写目的

本设计说明书的编写目的是为了详细阐述网络空间资产测绘工具的设计目标和功能需求，以便于开发人员、项目管理人员和最终用户对该工具有一个全面、深入的理解,为使用用户提供使用指南和参考。此工具的主要目标是开发一个能够对指定的IP地址范围进行网络空间资产测绘的工具，包括主机存活性探测、端口开放探测、协议识别、指纹识别、设备识别和蜜罐识别等功能。

## 应用功能

主机存活性探测：在用户提供的测绘目标范围内，利用有效手段探测出存活主机，并记录存活主机的IPv4地址。

端口开放探测：对存活主机进行端口开放状态探测，并记录存活主机开放端口信息。

协议识别：对存活主机开放端口进行服务探测，并记录已知开放端口的协议信息。如果协议未知，该项为 unknown。如果没有协议识别结果信息，该项为null。

指纹识别：对已知端口的开放服务进行指纹识别，并记录其指纹信息列表。指纹识别结果格式为：应用（服务）名称/版本信息，例如：[“nginx/1.1.1” ,… ,“WordPress/N”]，“N”：表示无精确版本信息。如果没有指纹识别结果信息，该项为null。

设备识别：对存活主机进行探测与应用识别，记录存活主机设备信息。设备识别结果格式为：设备类型/产品名称，例如：“webcam/Hikvision”。如果没有设备识别结果信息，该项为null。

蜜罐识别：记录蜜罐信息，包含端口/蜜罐，例如：["22/kippo","80/glastopf","3306/N"]。

所有的测绘结果需要按照特定的JSON格式进行记录，包括IP地址、服务、端口、协议、服务应用、设备信息、蜜罐信息和时间戳等。

本设计说明书将在后续章节中详细阐述以上功能的实现方式和技术要求。

# 技术框架

## 技术分析

根据项目需求，我们设计的网络空间测绘引擎由五个主要模块构成，分别为：解析模块、扫描模块、指纹识别模块、分析模块和调度模块。以下是各个模块的功能与设计：

**解析模块：**解析模块的主要任务是对输入的目标测绘地址进行解析，并分发扫描任务。解析模块需要处理的输入可能是一个IP地址、一个IP地址范围或者一个CIDR块。解析模块首先需要验证输入的格式是否正确，然后将输入解析为一系列的IP地址。这些IP地址将作为扫描任务分发给扫描模块。解析模块需要能够处理大量的输入，并且能够有效地分发任务，以确保扫描模块的工作负载均衡。

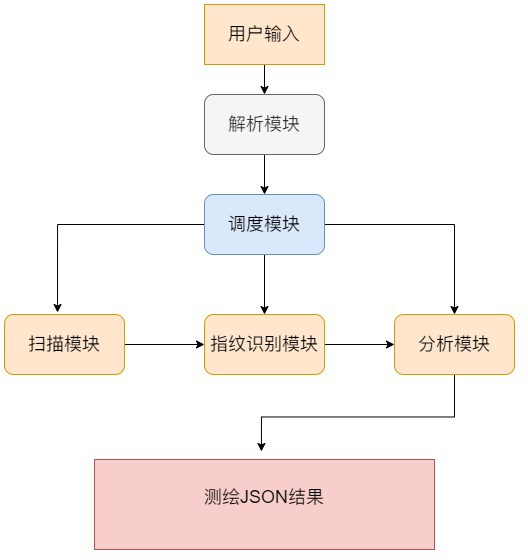
**扫描模块：**扫描模块的主要任务是接收扫描任务，对主机进行ICMP与TCP测活，并对存活的主机进行端口扫描。扫描模块需要实现ICMP Echo Request（ping）和TCP SYN扫描两种测活方法。对于存活的主机，扫描模块需要对其所有端口进行扫描，以确定哪些端口是开放的。扫描模块需要能够处理大量的扫描任务，并且能够快速准确地完成扫描。

**指纹识别模块：**指纹识别模块的主要任务是对开放的端口进行指纹识别，并对其运行的服务、设备类型、网络协议等进行标注。指纹识别模块需要实现多种指纹识别方法，包括但不限于TCP/IP堆栈指纹识别、服务版本指纹识别和设备类型指纹识别。指纹识别模块需要能够处理大量的指纹识别任务，并且能够准确地识别出服务、设备类型和网络协议。

**分析模块：**分析模块的主要任务是对扫描结果进行分析，识别设备类型、蜜罐，并对结果进行汇总输出。分析模块需要实现多种分析方法，包括但不限于设备类型分析、蜜罐分析和结果汇总。设备类型分析需要根据指纹识别模块的结果，识别出设备的类型。蜜罐分析需要根据指纹识别模块的结果，识别出可能的蜜罐。结果汇总需要将所有的分析结果整合在一起，生成一个完整的网络空间测绘报告。

**调度模块：**调度模块是整个系统的核心，它负责协调解析模块、扫描模块、指纹识别模块和分析模块的工作。调度模块需要能够处理大量的任务，并且能够有效地分配资源，以确保整个系统的高效运行。

以上五个模块的设计和实现，将确保网络空间资产测绘工具能够有效、准确地完成网络空间资产的测绘任务。其流程可以用如下流程图说明。



## 模块实现

在本空间测绘引擎中我们使⽤Go语⾔进⾏设计，依托于Go语⾔⾼效的性能和优秀的线程处理，我们实现了⼀套性能⼗分优异的空间测绘引擎。

1.解析模块：Go语言的 net 包提供了解析IP地址和CIDR的功能。我们可以使用 net.ParseIP 和 net.ParseCIDR 函数来解析输入的IP地址和CIDR。解析后的IP地址可以放入一个channel中，作为扫描任务分发给扫描模块。

2.扫描模块：Go语⾔的并发模型使得并发扫描变得⾮常简单。我们可以使⽤goroutine来并发地处理每个扫描任务。Go语⾔的 net 包提供了创建TCP连接的功能，我们可以使⽤ net.DialTimeout 函数来尝试创建⼀个TCP连接，以此来检测⼀个端⼜是否开放。在本模块中，我们借鉴nmap的实现⽅式，实现了⾼性能的主机存活测试和端⼜扫描，并且初步实现了服务识别。

3.指纹识别模块：指纹识别通常需要发送特定的请求并分析响应。在本空间测绘引擎中，依赖正则引擎实现了对Port, Response, Hash进⾏初步分析，并且对于特定的请求再次进⾏指纹分类。如针对HTTP，可以对Header, Body,Title等信息进⾏针对性匹配。

4.分析模块：分析模块需要处理指纹识别模块发送过来的结果。由于结果可能是并发发送的，我们可以使⽤Go语⾔的并发模型来并发地处理这些结果。处理完的结果可以放⼊⼀个struct中，然后写⼊到⼀个JSON⽂件，作为最终的测绘报告。

5.调度模块：调度模块需要协调解析模块、扫描模块、指纹识别模块和分析模块的⼯作。我们可以使⽤Go语⾔的channel和select语句来实现这个功能。每个模块都有⼀个或多个channel来接收任务和发送结果，调度模块可以使⽤select语句来监听这些channel，然后根据收到的消息来调度各个模块的⼯作。

# 效果展示

## 使用说明

首先，使用者需要指定探测对象。这可以通过 -t 或 --target 参数来实现。探测对象可以是单个IP地址，例如 "114.114.114.114"；也可以是一个IP地址段，例如 "114.114.114.114/24" 或 "114.114.114.114-115.115.115.115"；还可以是一个URL地址，例如 "https://www.baidu.com"；甚至可以是一个包含多个目标的文件地址，例如 "file:/tmp/target.txt"。使得使用者可以根据实际需求来选择最适合的探测对象。

若使用者想要进行网段探测，可以使用 --spy 参数启动网段探测模式。在此模式下，工具将自动探测主机可达的内网网段。可以选择探测 "192"、"10"、"172" 网段，也可以选择探测所有网段，即 "all"，或者指定一个特定的IP地址，工具将探测该IP地址所在的B段存活网关。

在进行网络探测时，可能希望只扫描某些特定的端口。这可以通过 -p 或 --port 参数来实现。默认情况下，工具会扫描TOP400的端口。但也可以指定扫描其他端口，例如 "80"、"8080" 或 "8088-8090"。

在完成网络探测后，若希望保存扫描结果。这可以通过 -oJ 参数来实现。使用此参数后，扫描结果将以json格式保存到文件中。例如，您可以使用命令 "./tool --target 114.114.114.114 -oJ output.json" 将扫描结果保存到 "output.json" 文件中。

在进行网络探测时，支持设置一个超时时间。这可以通过 --timeout 参数来实现。例如，您可以使用命令 "./tool --target 114.114.114.114 --timeout 10" 来设置超时时间为10秒。

此外，工具还提供了一种扫描常见端口的功能。这可以通过 --top 参数来实现。默认情况下，工具会扫描TOP400的端口。但也可以指定扫描其他数量的常见端口，例如TOP1000。例如使用命令 "./tool --target 114.114.114.114 --top 1000" 来扫描TOP1000的常见端口。

## 实例展示

输入命令go run ./jkscan.go -t 164.92.167.0/24 -p 1-65535 -oJ outputs/164.92.167.0.json可获得全端口的162.92.167.0/24网段的端口扫描信息，得到的部分json结果如下：

{

## "164.92.167.1": {

"services": [

{

"port": 888, "protocol": "http", "service\_app": null

},

{

"port": 22, "protocol": "ssh", "service\_app": [

"openssh/7.6", "ubuntu/4ubuntu0.6"

]

},

{

"port": 88, "protocol": "http", "service\_app": null

}

],

"deviceinfo": null, "honeypot": null

},

## "164.92.167.10": {

"services": [

{

"port": 22, "protocol": "ssh", "service\_app": [

"openssh/8.9", "ubuntu/3ubuntu0.1"

]

},

{

"port": 3001, "protocol": "http", "service\_app": null

},

{

"port": 80, "protocol": "http", "service\_app": [

"nginx/1.18.0", "express/N", "ubuntu/N"

]

},

{

"port": 9200, "protocol": "http", "service\_app": null

},

{

"port": 3000, "protocol": "http", "service\_app": [

"node.js/N", "express/N"

]

}

],

## "deviceinfo": null,

## "honeypot": null

}

}

运行界面如下：



## 性能测试

针对空间测绘工具jkscan的性能测试结果如下：

**测试环境：**

硬件配置：1核CPU，1GB内存

网络环境：广域网环境，目标机去测试机的网络延迟普遍在50-200ms以内

目标网络：C段网络，包含256个IP地址

测试内容：该网段下所有机器所开服务信息。

全端口扫描：扫描目标网络的所有65535个端口

常用端口扫描：扫描目标网络的常用1000个端口

**测试步骤：**

启动空间探测器，设置目标网络和扫描端口

开始扫描，记录开始时间

扫描完成后，记录结束时间

计算扫描时间，即结束时间减去开始时间

分析扫描结果，包括发现的主机数量、开放的端口数量等

扫描速度：扫描时间除以扫描的IP地址数量和端口数量

发现率：发现的主机数量除以实际的主机数量

准确率：正确识别的端口状态（开放或关闭）的数量除以所有端口的数量

**测试结果：**

全端口扫描：完成扫描所需耗时为40-90分钟，准确率高于98%

常用端口扫描：完成扫描耗时2-5分钟，准确率高于98%。