SHadowsocks设计分析文档

2017/12/27

目录

[项目背景和原理 1](#_Toc499830161)

[项目方法 2](#_Toc499830162)

[OOA模型 2](#_Toc499830163)

[1. 需求模型 2](#_Toc499830164)

[系统边界 2](#_Toc499830165)

[2. 基本模型 3](#_Toc499830166)

[3. 辅助模型 4](#_Toc499830167)

[OOD模型 4](#_Toc499830168)

[1. 问题域部分 4](#_Toc499830169)

[2. 数据接口部分 4](#_Toc499830170)

[3. 控制驱动部分 4](#_Toc499830171)

[4. 人机交互部分 4](#_Toc499830172)

[总结 4](#_Toc499830173)

SHadowsocks设计分析文档

2017/12/27

# 项目背景和原理

Shadowsocks是一款比较知名和应用比较广泛的翻墙软件。Shadowsocks的代码质量很高，并且有python编程语言编写的版本，符合面向对象编程的要求。

Shadowsocks的翻墙原理如下列示意图所示：



图1. Shadowsocks原理示意图

Shadowsocks基于Sock5协议完成了一个墙内墙外的通信过程：

1. SS Local把PC的数据包进行加密
2. SS Local转发加密后的数据包到墙外的服务器SS Server
3. SS Server把解密后的数据包转发到目标地址
4. SS Server把返回的数据包进行加密
5. SS Server把加密后的数据包转发到墙内的SS Local
6. SS Local把解密后的数据包返回到PC

上述就是Shadowsocks的基本原理。该软件实现过程涉及到网络通信、命令行交互、加密解密、IO复用等方面，是一个有一定复杂度的软件。

# 项目方法

本次项目主要针对shadoesocks的2.8.2 python版本进行建模分析。在这个项目里面我们主要采用的建模方法是面向对象建模。原因主要有以下几点：

1. 本次分析的软件是用python语言进行编写的。而python是一个面向对象的编程语言，与面向对象建模及其吻合
2. 面向对象建模是目前比较流行的一种建模方法。掌握好这一方法有利于提高以后的编程能力。

因此在采用面向对象建模方法以后，主要流程包括OOA和OOD分析两个阶段。这两个阶段建立的模型分别为OOA模型和OOD模型。

# OOA模型

OOA阶段主要发生在面向对象建模的分析阶段。OOA模型框架的示意图如下：



图2. OOA模型框架

本小节主要按照需求模型、基本模型和辅助模型的顺序一一阐述。

## 需求模型

### 系统边界

在面向对象建模的方法中，第一步需要进行的就是需求分析建模。而需求模型主要是由用况图(use case diagram)和对应的文档构成。在构建需求模型之前，我们需要确定好系统边界。

在系统边界的表示上，我们首先把整个软件系统看成一个单独的系统，把软件之外的参与者先确定出来。

首先对整个系统进行边界确定。其示意图如图3所示。首先整个系统的参与者有以下几个：

1. 客户端：即墙内的PC，利用本系统与墙外的某些服务器通信
2. 服务器端：即墙外的服务，利用本系统与墙内的PC通信
3. DNS服务器：主要是客户端再必要的时候，通过DNS服务器查询服务器的ip地址。



图3. 整体系统的边界

根据以上信息建立需求模型如图4所示下：



图4 需求模型用况图

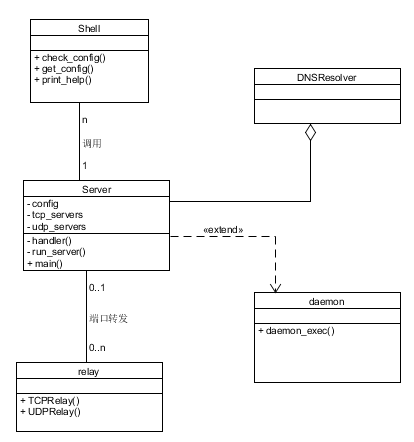
其中每个用况的如下：

|  |  |
| --- | --- |
| 用况名称 | SS Server |
| 用况编号 | 1 |
| 行为陈述 | SS Server 获取服务器配置 启动服务器进程  配置输入端口 开启端口转发  DNS解析  while 未收到关机指令  服务器持续运行  exit |

|  |  |
| --- | --- |
| 用况名称 | 端口转发 |
| 用况编号 | 2 |
| 行为陈述 | 分配输入端口  配置端口密码  for 每个配置的端口  call 开启端口  call 加入TCP转发列表  call 加入UDP转发列表 |

## 基本模型

基本模型是类图



|  |  |  |
| --- | --- | --- |
| 类的总体说明 |  |  |
|  | 类名 | Server |
|  | 解释 | 这个类的主要功能就是分配代理端口，接收用户的访问请求。 |
|  | 一般类 | None |
|  | 主动性 | No |
|  | 持久性 | Yes |
|  | 辅助模型 |  |
|  | 其他 |  |
| 属性说明 |  |  |
|  | { |  |
|  | 名称与数据类型 | config: list |
|  | 属性解释 | 获取服务器配置 |
|  | 多态性 |  |
|  | 组合 | Server端接下来的所有操作都要使用config的内容 |
|  | 其它 |  |
|  | } |  |
|  | { |  |
|  | 名称与数据类型 | tcp\_servers: list |
|  | 属性解释 | 保存了TCP连接列表并进行管理 |
|  | 多态性 |  |
|  | 聚合 | tcp\_servers将tcprelay管理的TCP连接加入，得到TCP连接列表 |
|  | 其它 |  |
|  | } |  |
|  | { |  |
|  | 名称与数据类型 | udp\_servers: list |
|  | 属性解释 | 保存UDP连接列表并进行管理 |
|  | 多态性 |  |
|  | 聚合 | udp\_servers将udprelay管理的UDP连接加入，得到UDP连接列表 |
|  | 其它 |  |
|  | } |  |
| 操作说明 |  |  |
|  | { |  |
|  | 特征标记 | handler(signum: int) |
|  | 操作解释 | 管理服务器各子节点的运行，维持服务器正常运作 |
|  | 主动性方法 | 被动 |
|  | 多态性 |  |
|  | 消息发送 | signum: 向操作系统发送需要终止的进程的进程号 |
|  | 操作流程 |  |
|  | 其他 |  |
|  | } |  |
|  | { |  |
|  | 特征标记 | run\_server() |
|  | 操作解释 | 该操作主要是启动服务器 |
|  | 主动性方法 | 被动 |
|  | 多态性 |  |
|  | 消息发送 | None |
|  | 操作流程 |  |
|  | 其他 |  |
|  | } |  |
|  | { |  |
|  | 特征标记 | main() |
|  | 操作解释 | 主函数，初始化服务器配置，分配服务端口和密码，并启动服务器 |
|  | 主动性方法 | 主动[进程] |
|  | 多态性 |  |
|  | 消息发送 | None |
|  | 操作流程 |  |
|  | 其他 |  |
|  | } |  |

|  |  |  |
| --- | --- | --- |
| 类的总体说明 |  |  |
|  | 类名 | Shell |
|  | 解释 | ShadowSocks软件的内核，定义了配置文件的搜索、检查、提取内容的方法，以及打印  帮助文档的方法等。 |
|  | 一般类 | None |
|  | 主动性 | No |
|  | 持久性 | No |
|  | 辅助模型 |  |
|  | 其他 |  |
| 操作说明 |  |  |
|  | { |  |
|  | 特征标记 | check\_config(config: list, is\_local: boolean) |
|  | 操作解释 | 检查并安装配置。如果有异常（IP地址异常或者加密算法不安全等）会发出警告，  出现错误（密码错误或未指定等）会退出程序。 |
|  | 主动性方法 | 被动 |
|  | 多态性 | 部分代码的执行与否取决于is\_local的值（为true时仅需要密码，为false时需要本地密  码或端口密码） |
|  | 消息发送 | None |
|  | 操作流程 |  |
|  | 其他 |  |
|  | } |  |
|  | { |  |
|  | 特征标记 | get\_config(is\_local: boolean) |
|  | 操作解释 | 获取配置信息 |
|  | 主动性方法 | 被动 |
|  | 多态性 | is\_local为true时获取客户端配置，为false获取服务器配置 |
|  | 消息发送 | None |
|  | 操作流程 |  |
|  | 其他 |  |
|  | } |  |
|  | { |  |
|  | 特征标记 | print\_help(is\_local: boolean) |
|  | 操作解释 | 打印帮助信息 |
|  | 主动性方法 | 被动 |
|  | 多态性 | is\_local为true时打印客户端帮助信息，否则打印服务器帮助信息 |
|  | 消息发送 | None |
|  | 操作流程 |  |
|  | 其他 |  |
|  | } |  |

## 辅助模型

本项目主要用到的辅助模型包括

# OOD模型

OOD阶段是建立在OOA阶段的基础上，对OOA模型作必要的修改和调整或补充某些细节。其模型框架如下图所示：



本小节主要是按照问题域部分、数据接口部分、控制驱动部分和人机交互部分去阐述。

## 问题域部分

问题域部分主要是软件对编程语言的支持、操作系统等方面对设计进行调整。

（1）编程语言方面，python2.6及以后版本的异常处理语句格式如下：

try:

…

Except error as e:

…

在本程序中使用到了这类的语句，因此要求python版本在2.6以上。

## 数据接口部分

数据接口部分包括

## 控制驱动部分

控制驱动部分包括

## 人机交互部分

人机交互部分包括打印帮助信息等。

在shell中对print\_help()方法细化，分别定义了2个方法print\_local\_help()和print\_server\_help()，用于打印不同端的帮助信息。

# 总结

总结