实验五:安全代理协议与实现

周辛宁 计74 2017013684
林俊峰 计73 2017011303

一. 实验简述

Browser (Client) < --- > Local < --- > Server < --- > Web
本次实验中的安全代理协议主要在 `Local` 和 `Server` 之间起到作用.

二. 实验设计思路

2.1. Socks5通信

使用python2的socket库和StreamRequestHandler实现代理，依据socks5协议进行通信，配置认证方式为用户名/密码，支持IPV4, IPV6 和 域名三种地址格式

2.2. 安全通信设计

为了加强 `Local` 和 `Server` 之间的通信安全, 我们考虑采用了对称加密+非对称加密算法组合的方式, 具体的选择为:
对称加密算法: 基于密钥 (KEY)的置换表算法
非对称加密算法: RSA加密 (公钥KU, 私钥KR)
(以上算法均由手工实现)

2.2.1. 安全通信流程设计

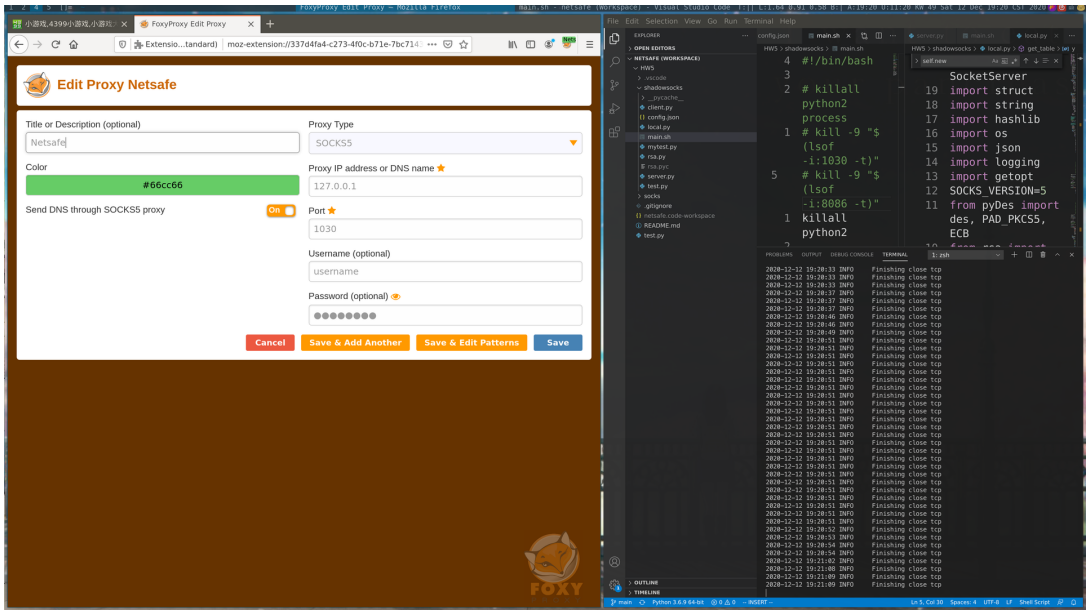
假设 `Local` 要与 `Server` 之间进行通讯 (以下简称为L和S)

1. L与S基于实现给定的KEY_0加密交换彼此的公钥 `KUl` 和 `KUs` .
2. L用S的公钥对一条含有L标识和一个随机生成的临时交互号N1的信息加密发送给S.
   a. E_{KUs}[L+N1]
3. S对信息解密, 验证L后, 将N1+1, 并使用L的公钥对信息加密并返回.
   a. E_{KUl}[S+(N1+1)]
4. L对信息解密, 验证S和N1+1后, 随机生成密钥KEY, 并返回加密信息M:
   a. M = E_{KUs}[E_{KRl} [KEY]]
5. S使用自己的私钥和L的公钥对M进行解密, 获得KEY.
6. 之后双方的通信使用KEY进行对称加密.

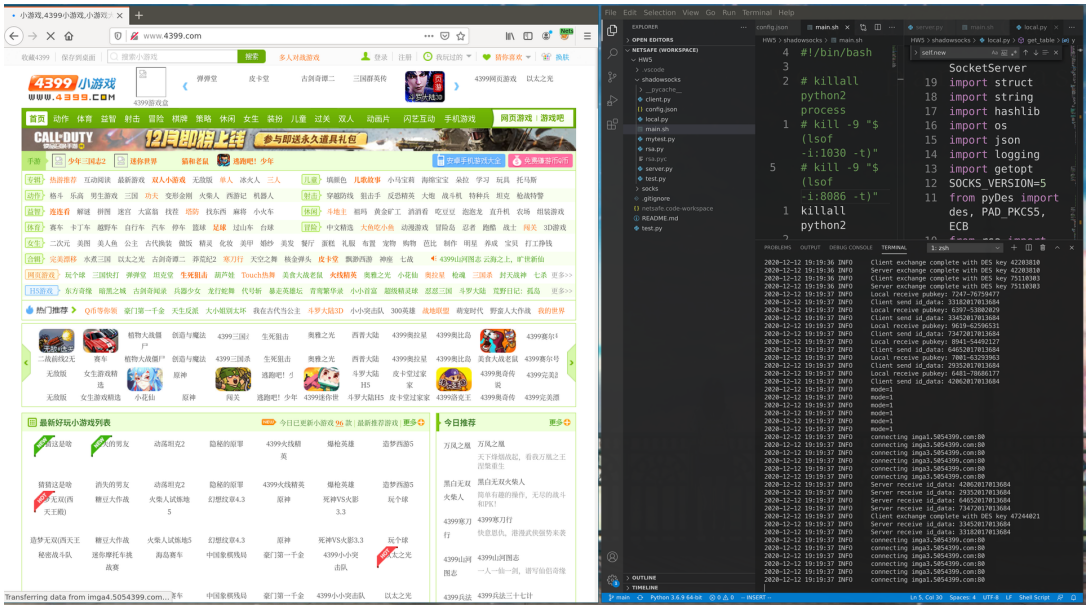
思考: 在此流程设计中 最容易受到攻击的其实是第一步公钥的交换, 在实际场景中可能会采用如"公开发布", "公钥证书"...等方式进行公钥的交换.

三. 试验结果

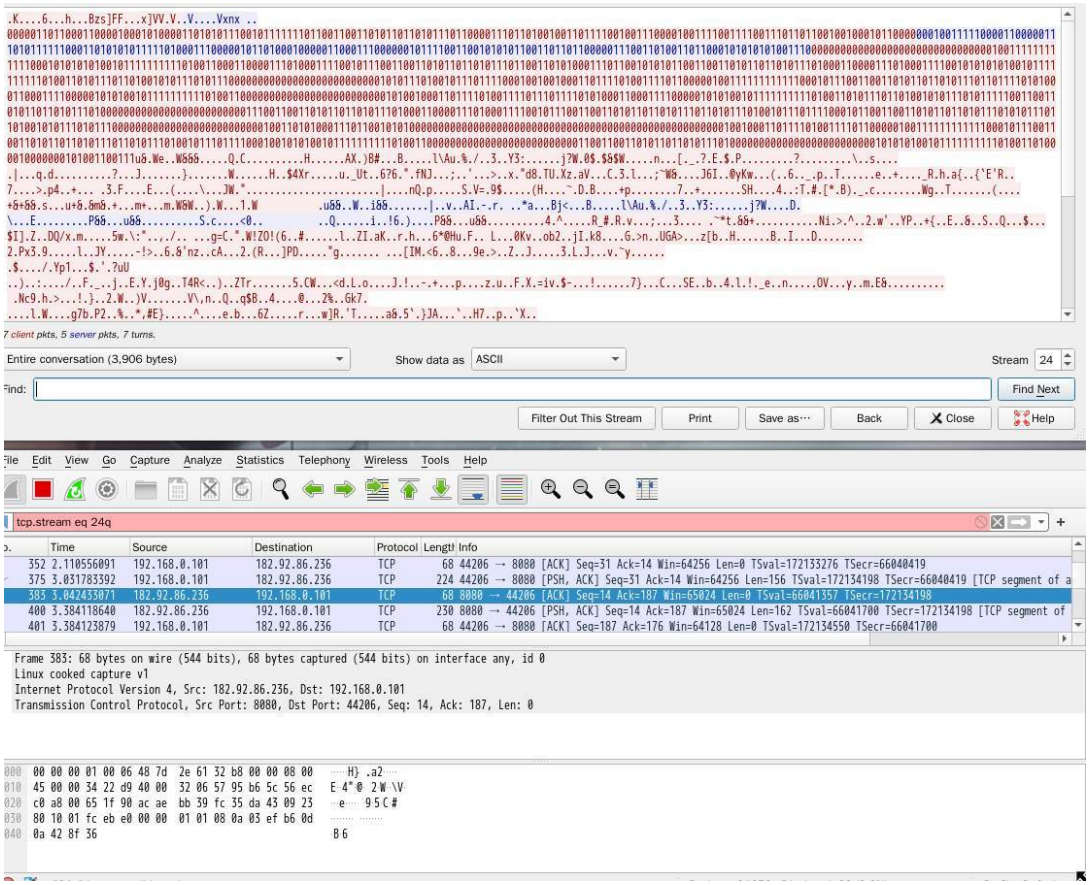
使用foxyproxy能够通过身份认证连接上本地的代理:



使用代理后能正常访问 http://www.4399.com:



通信被成功加密



四. 代码使用说明

运行环境

```
# python2.7
sudo apt install libgmp-dev
pip2 install pyasn1 gmpy numpy pyDes gevent
```

```
# local端
# 默认配置写在 config 文件中，可以通过命令行参数覆盖
python2 local.py -l <代理端口> -s <服务器地址> -p <服务器端口>
# server端
python2 server.py -p <服务器端口>
# 浏览器端
使用firefox浏览器，foxyproxy插件配置socks5代理，用户名username，密码password
```