# 软件安全T1——基于Flask框架的远程 控制程序

**小组名称**：G11

|  |  |  |
| --- | --- | --- |
|  | 姓名 | 学号 |
| 组长 | 罗之龙 | 2019302180017 |
| 组员 | 李云龙 | 2019302180016 |
| 组员 | 高建烨 | 2019302180014 |
| 组员 | 张依帆 | 2019302180189 |

**成员分工**：

|  |  |
| --- | --- |
| 罗之龙 | 配置服务端端由，WebUI设计，进程管理，文件管理 |
| 李云龙 | 屏幕截取，开启麦克风、摄像头，获取键盘记录 |
| 高建烨 | 客户端任务分发及执行，WebUI设计，文件管理 |
| 张依帆 | 注册表管理 |

注：每人分别就自己负责的工作部分撰写实验报告

**目录**

[软件安全T1——基于Flask的远程控制程序 1](#_Toc90210977)

[1 简介 2](#_Toc90210978)

[2 工作原理 2](#_Toc90210979)

[2.1 程序基本结构 2](#_Toc90210980)

[2.2 通信模块 3](#_Toc90210981)

[2.3 用户交互模块 4](#_Toc90210982)

[2.3 命令分发与执行模块 5](#_Toc90210983)

[3 功能测试 17](#_Toc90210984)

[3.1 文件管理 17](#_Toc90210985)

[3.2 进程管理 23](#_Toc90210986)

[3.3 注册表管理 26](#_Toc90210987)

[3.4 开启摄像头、麦克风 28](#_Toc90210988)

[3.5 屏幕截取 32](#_Toc90210989)

[3.6 键盘记录 33](#_Toc90210990)

[4 总结 34](#_Toc90210991)

## 1 简介

我们小组基于Flask框架制作了一个远程控制程序（Web应用），实现了控制端(server)远程对被控端(client)进行文件管理、进程管理、注册表管理、开启摄像头、麦克风、屏幕截取、获取键盘记录等功能，拥有良好的人机交互界面。

**关于Flask框架：**

Flask是一个使用Python编写的轻量级Web应用框架，由Werkzeug和Jinja2两大组件构成。Werkzeug是一个WSGI工具包，它实现了请求，响应对象和实用函数。这使得能够在其上构建web框架。Flask框架使用Werkzeug作为其基础之一。Jinja2是Python的一个流行的模板引擎。Web模板系统将模板与特定数据源组合以呈现动态网页。



图 1 Flask通信原理

图 1展示了客户端与基于Flask框架的服务端的通信过程。Flask框架基于Websocket进行通信，Werkzeug负责监听socket请求，并调用Flask App提供的函数来进行逻辑处理。在我们实现的远程控制程序设计中，服务端作为控制端，而客户端作为被控端。

## 2 工作原理

### 2.1 程序基本结构

图 2展示了远程控制程序的总体工作流程，该程序可分为三部分：通信模块、用户交互模块以及命令分发与执行模块。该程序实现远程控制功能的大致流程如下：

1. 向serverhost/hello发送POST请求，请求服务端发送指令
2. 根据与用户的交互选择相应的指令
3. 根据用户选择向Client回复要执行的指令
4. 将指令进行处理后交付给shell执行
5. 将执行结果返回给Client
6. 向serverhost/report发送POST请求 ，发送执行结果
7. 在服务端的人机交互界面上显示输出信息

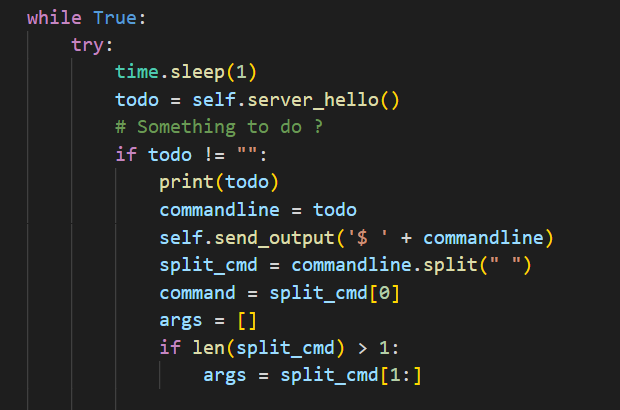


图 2 G11 Remote Controller 核心架构

### 2.2 通信模块

通信模块主要使用基于Flask的框架进行通信。通信的流程如下：

首先将服务端运行在指定的地址及端口，这是服务端已准备好接收和处理http请求。随后运行客户端，客户端在运行时会不断地调用Client类的server\_hello方法向服务端发送请求，请求服务端为其发送指令，如代码 1所示。



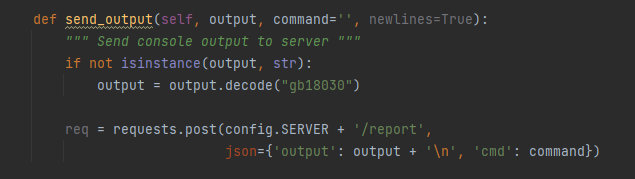
代码 1 Client类主循环

Flask框架依赖于路由完成http请求的分发，我们在服务端程序中注册了一系列url规则的路由，针对每种规则设立了相应的处理函数。如代码 2所示，服务端注册了/hello路由，调用send\_commands函数向客户端发送指令。



代码 2 /helllo路由

接下来客户端会进行指令的分发与执行，执行完毕后，如代码 3所示，客户端会调用send\_output函数将指令执行的节后回传给服务端，服务端随后会在人机交互界面上显示该结果，至此，远程控制功能的一轮通信结束。



代码 3 send\_output函数

### 2.3 用户交互模块

服务端在接收到客户端回传的指令执行结果之后，会将指令执行结果作为模板参数，调用render\_template函数来依照参数重新渲染模板，以达到实时与服务端用户进行交互的结果。

同时，我们将各种远程控制功能封装成四个模块吗，如图 3所示。在用户交互过程中，不必用户输入命令，用户只需点击交互界面中的按钮即可调用相关接口。

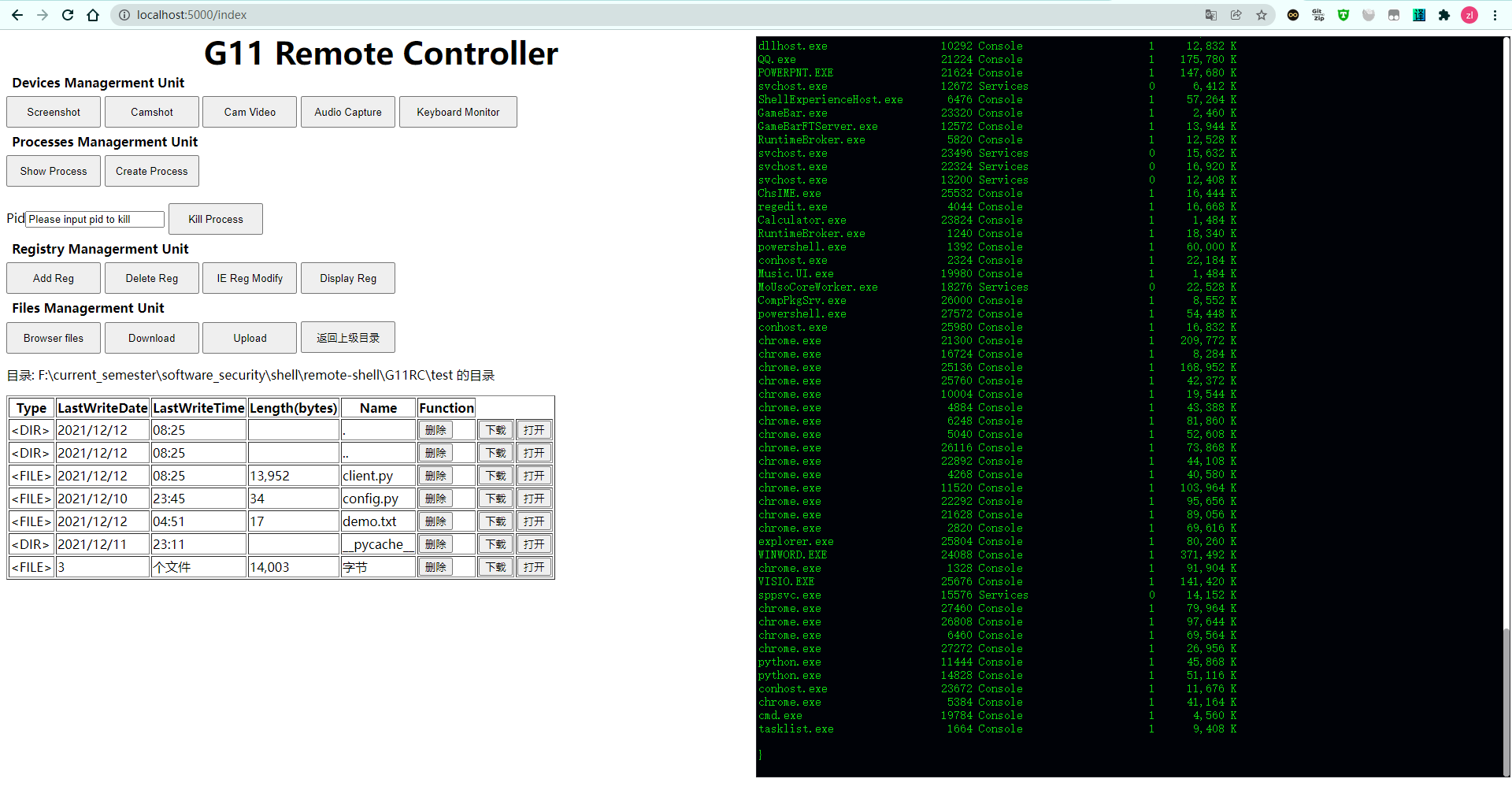


图 3 用户交互界面

### 2.3 命令分发与执行模块

#### 2.3.1 被控端命令分发

被控端命令分发主要就是Client类中的run()函数，根据收到的命令不同，执行相应的处理函数，实现细节如下：

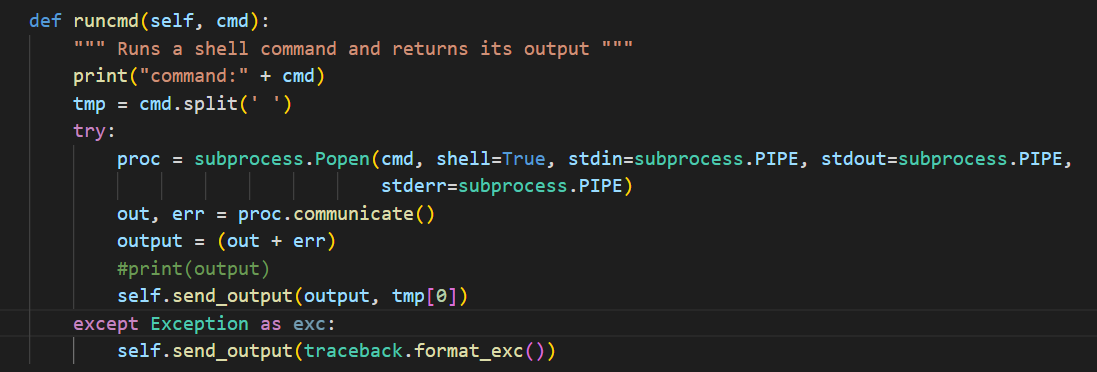


代码 4 命令分发

#### 2.3.2 文件管理

##### 2.3.2.1 常用的文件管理功能

常见的文件管理功能（查看文件、删除文件、创建文件）可以通过基本的shell命令实现，因此，我们只需要在runcmd方法中传入相应的指令，在被控制端通过runcmd执行这一指令，返回执行结果给服务端即可，如代码 5所示。

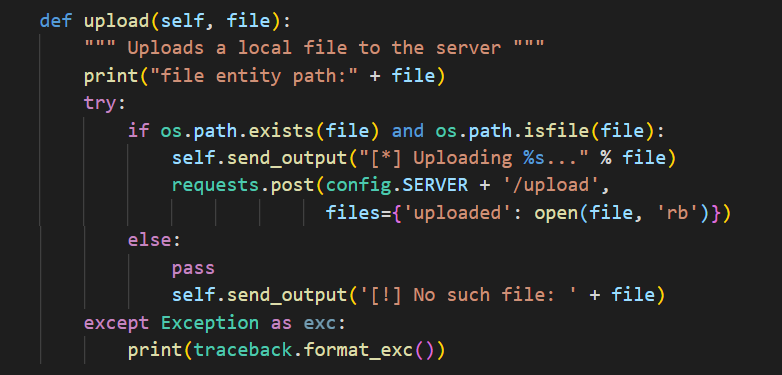


代码 5 runcmd方法

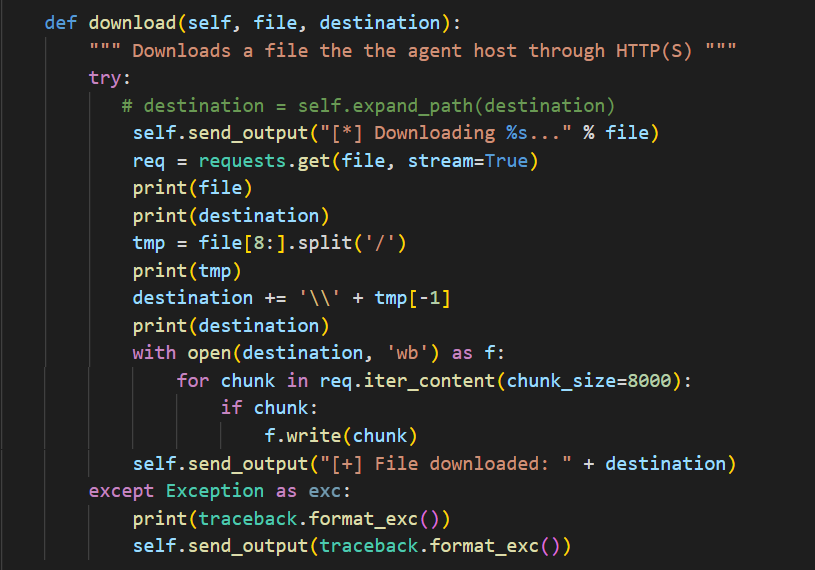
##### 2.3.2.2控制端和被控端文件的上传与下载

实现思路：上传（upload方法）用于从被控端向控制端发送文件，下载（download方法）用于在被控端向控制端请求下载文件。

具体实现如代码 6和代码 7所示。



代码 6 upload方法



代码 7 download方法

#### 2.3.3 进程管理

实现思路：进程管理的实现主要是直接通过在Client端运行shell命令(tasklist, taskkill等指令)，并将指令执行结果呈现给用户。UI设置如图 4 进程管理UI图 4所示。

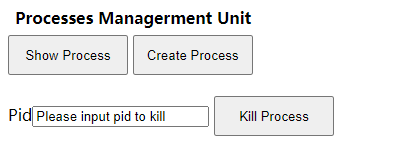
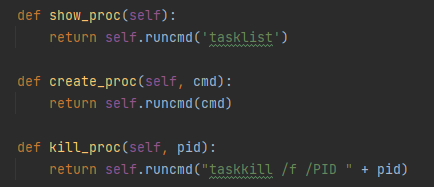


图 4 进程管理UI

具体实现如图 8所示：



代码 8 进程管理

#### 2.3.4 注册表管理

**注册表管理实现的功能：**

对注册表的内容进行获取，获取注册表的子项的名称、时间和数量，以及默认值等相关信息，也能对注册表的子项进行增添和删除，对value值进行修改和查看。

**Windows注册表简介：**

注册表中，左侧每一个基本项包含子项，子项也包含自己的子项，这里得出的子项数目不包括递归得到的子项，右侧的为value值。

Windows注册表的基本项和具体描述如下



图 5 Windows注册表简介

**功能说明以及代码说明：**

此模块的UI设计如图 6所示：

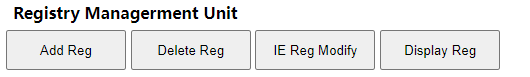


图 6 注册表管理UI设计

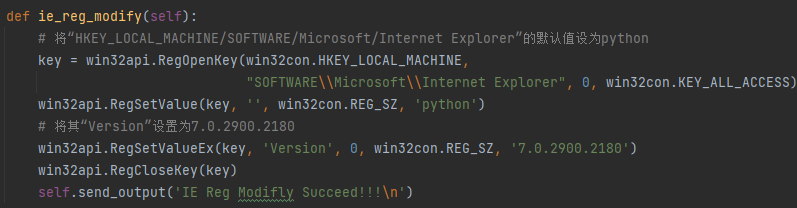
1. 在指定的注册表路径(HKEY\_CURRENT\_USER/SOFTWARE)中添加一个子项test，并且指定它的默认值

如果当前指定的是HKEY\_CURRENT\_USER下的software，权限是ALL\_ACCESS,size是通过RegQueryInfoKey（key）[0]，得到的子项的个数,然后通过for循环在所有的子项中查找，是否有名称为test的子项，根据结果不同设置result的值为ture/false。

具体实现代码如代码 9所示

代码 9 add\_reg

1. 修改指定的注册表项(此处是IE浏览器字段)

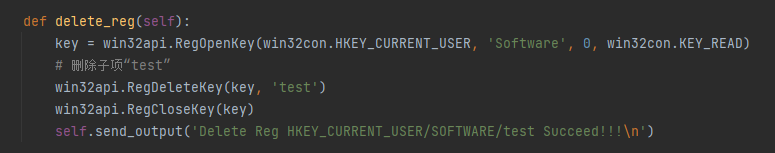


代码 10 ie\_reg\_modify

对Internet Explorer项的默认值和version数据都进行修改，使用了RegSetValue函数，指定要修改的项值和修改内容，其中‘’代表默认值，将默认值修改为python，version也进行修改。

1. 删除特定的注册表项

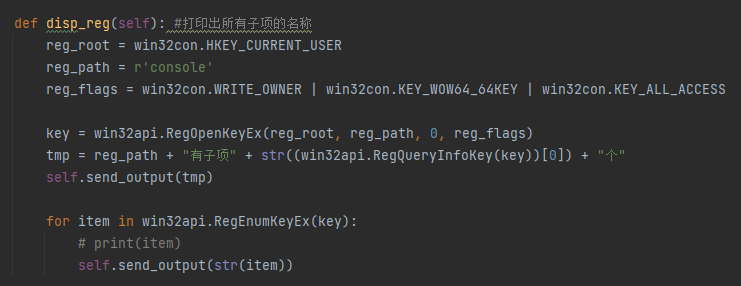
将（1）中创建的test子项删除，使用RegDeleteKey,通过key指定注册表项，通过test指定删除的子项。



代码 11 delete\_reg

1. 打印所有的子项信息

编写函数打印特定位置注册表的子项个数及内容，如代码 12所示。

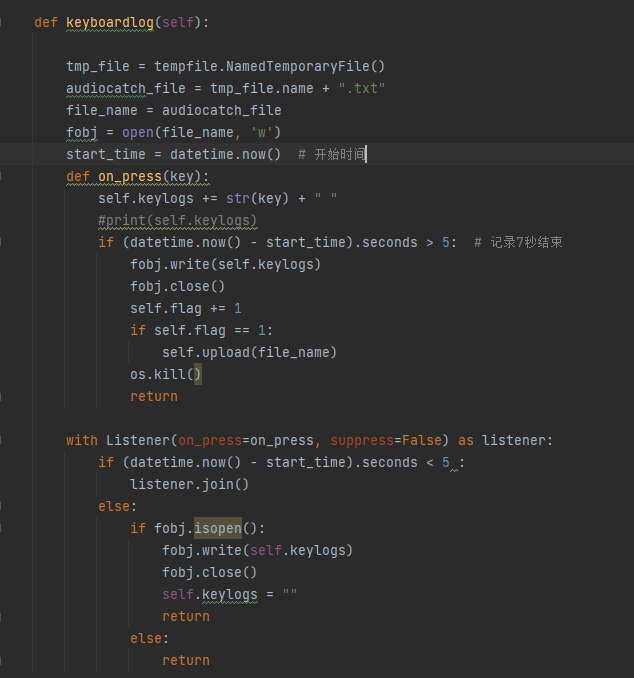


代码 12 disp\_reg

#### 2.3.5 键盘记录

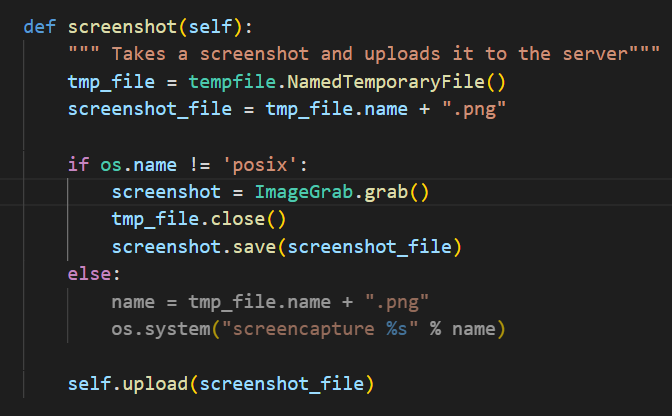
实现思路：通过定义一个类成员字符串变量keylogs，当接收到命令keylogger时，监听定义好的on\_press事件，on\_press事件中将键盘记录转为str追加到keylogs,定义一个计时器，当监听5秒之后，将keylogs写入文件，再调用upload将这个结果返回给server。键盘记录的txt文件存放在file\_from\_client目录下。

实现代码：



代码 13 获得键盘记录方法

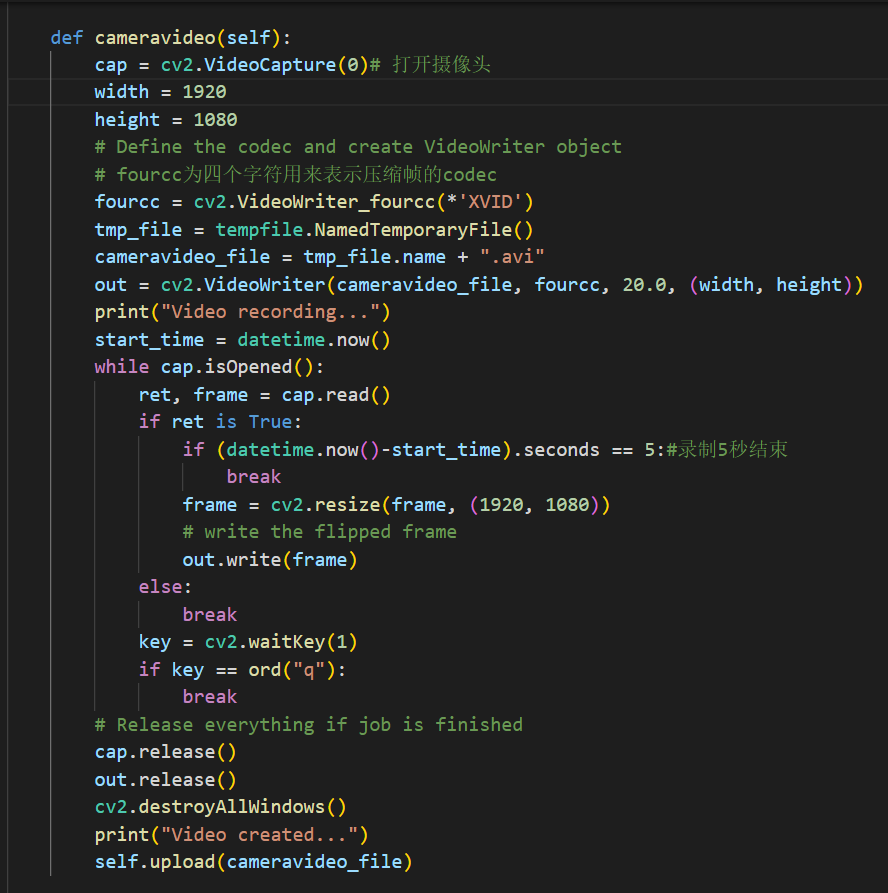
#### 2.3.6 屏幕截取

实现思路：首先通过tempfile库来创建系统临时文件，然后调用PIL库的ImageGrab模块对屏幕截图保存为png文件(系统临时文件)，最后调用文件管理部分实现的功能将屏幕截图文件发送回服务端，具体实现如代码 13所示。

代码 14 screenshot

#### 2.3.7 开启摄像头

实现思路：首先调用cv2库打开摄像头(VideoCapture(0)为笔记本电脑内置摄像头)，设置录制视频的相关参数，fourcc为四个字符用来表示压缩帧的codec，然后调用VideoWriter函数将摄像头录制的视频保存到系统临时文件(加上avi文件后缀)，录制视频的过程放在while循环中，循环结束的条件为(datetime.now()-start\_time).seconds == 5即录制5秒后停止录制，将生成的摄像头录制文件（avi文件）通过文件管理部分的upload功能发送回server端。关于摄像头的调用，我们还实现了另一个调用摄像头来对clint端拍照，并返回图片，思路类似，具体实现如代码 14和代码 15所示。



代码 15 cameravideo



代码 16 camerashot

#### 2.3.8 开启麦克风

实现思路：开启麦克风主要调用pyaudio库和wave库，首先要设置一些录制音频的配置参数例如：数据流块、量化位数、声道数以及采样率等等。然后实例化一个PyAudio对象，我们默认开启麦克风录制10秒的视频，将录制的音频文件保存在系统临时文件(加上.wav文件后缀)中，将此文件通过文件管理部分的upload功能发送回服务端，具体实现如代码 16所示。



代码 17 audiocatch

## 3 功能测试

### 3.1 文件管理

1. 查看被控端运行目录下的文件

点击“Browser files”按钮，遍历文件目录。

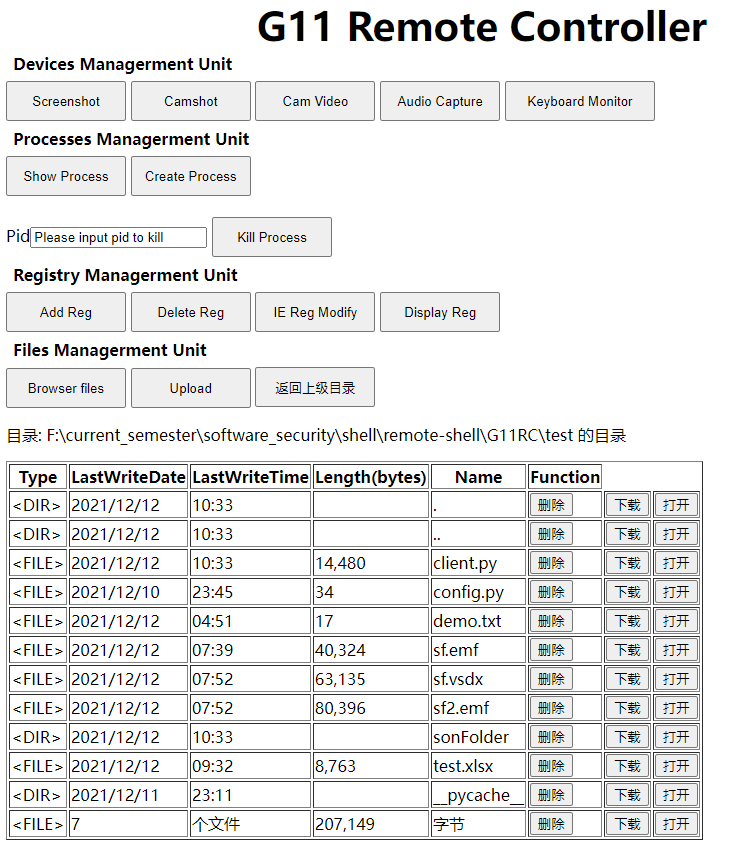


图 7 文件遍历演示

1. 进入子目录

点击文件列表种想要打开的子目录后方的“打开”按钮，即可进入对应子目录并刷新文件列表。

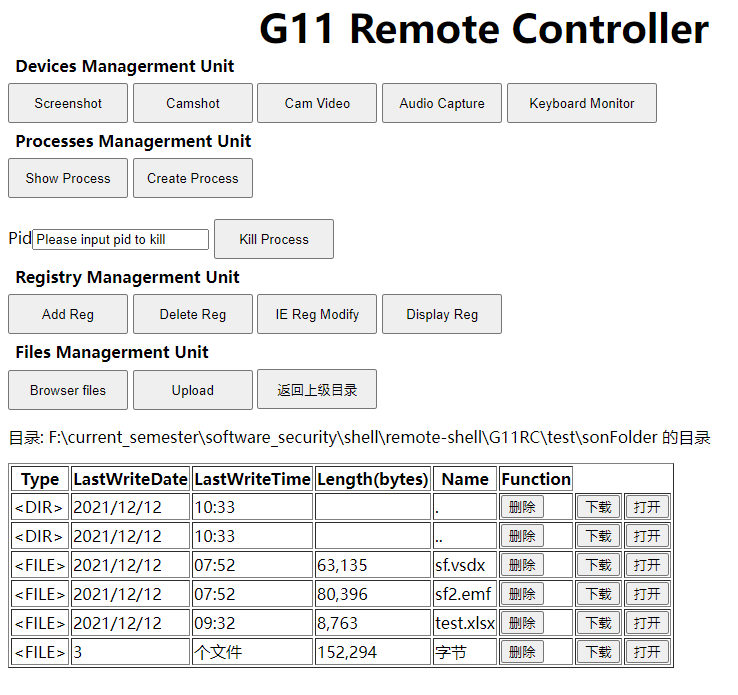


图 8 进入子目录演示

1. 返回上一级目录

点击“返回上级目录”按钮，即可返回上一级目录并刷新文件列表。

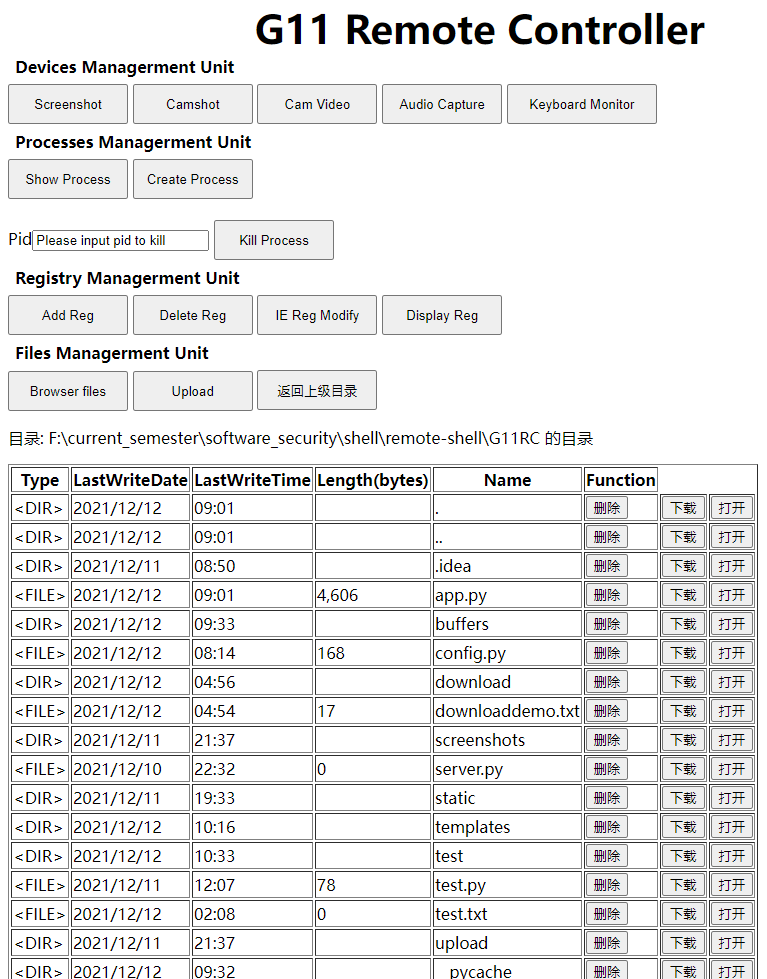


图 9 返回上级目录演示

1. 下载文件

点击想要下载的文件项后方的“下载”按钮，即可将对应文件下载到默认的缓存目录。

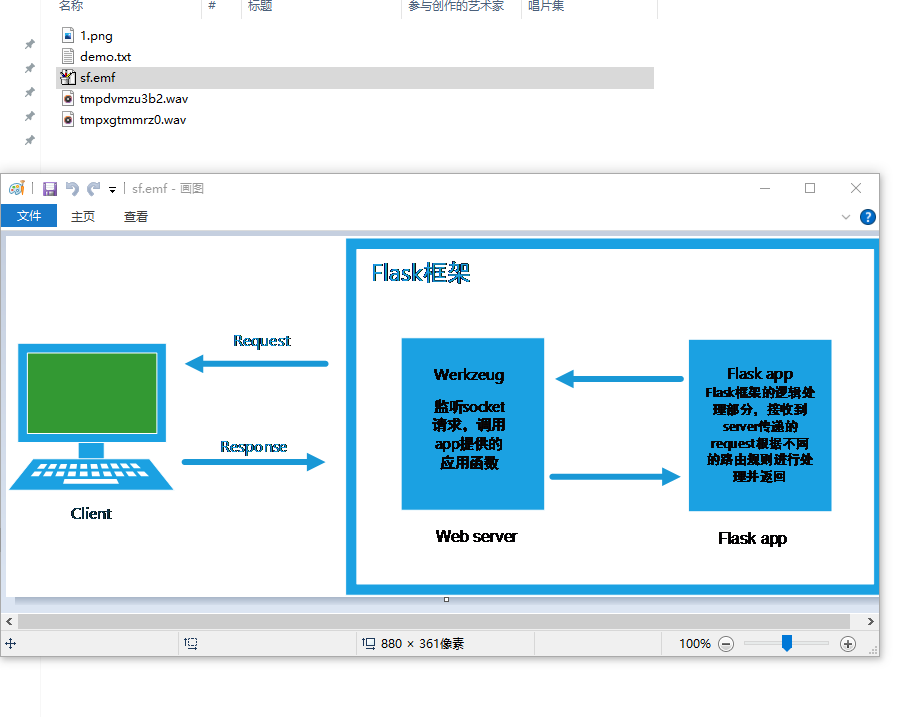


图 10 下载文件演示

1. 上传文件

点击“上传文件按钮”，在弹出的窗口中选择要上传的文件。

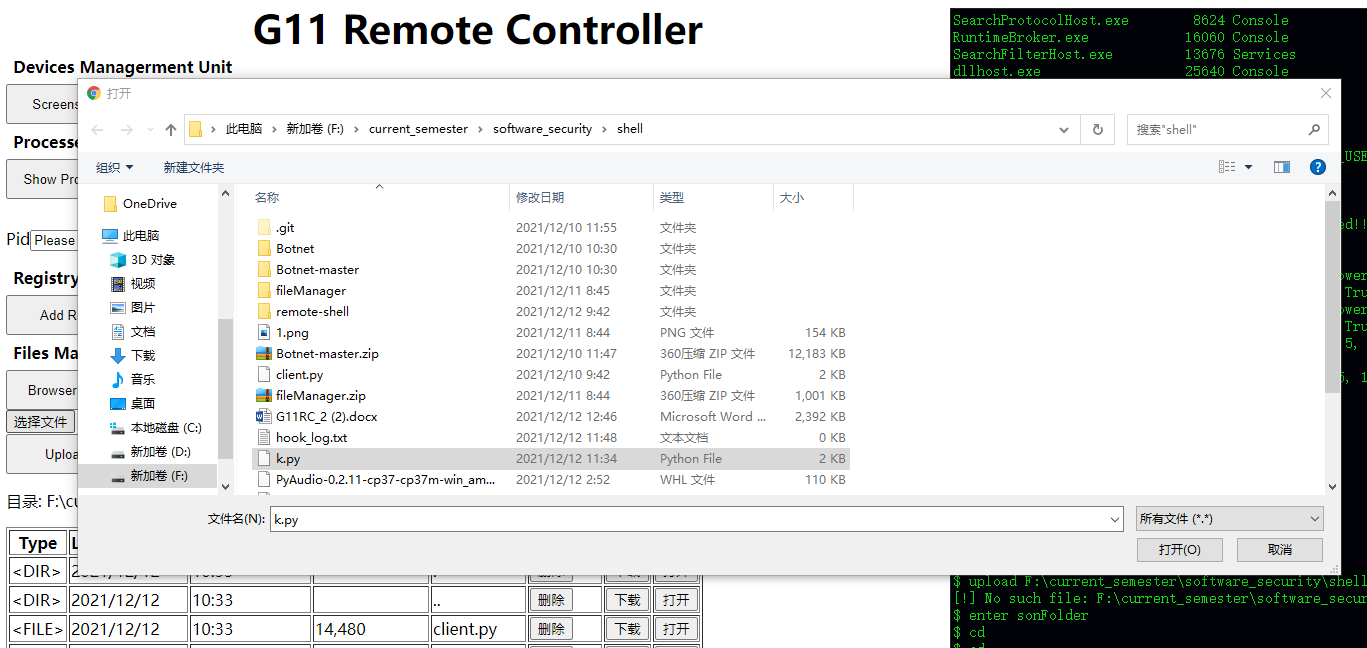


图 11 选择上传的文件

点击“提交”与“Upload”按钮，可将刚才选择的文件传送到被控机。

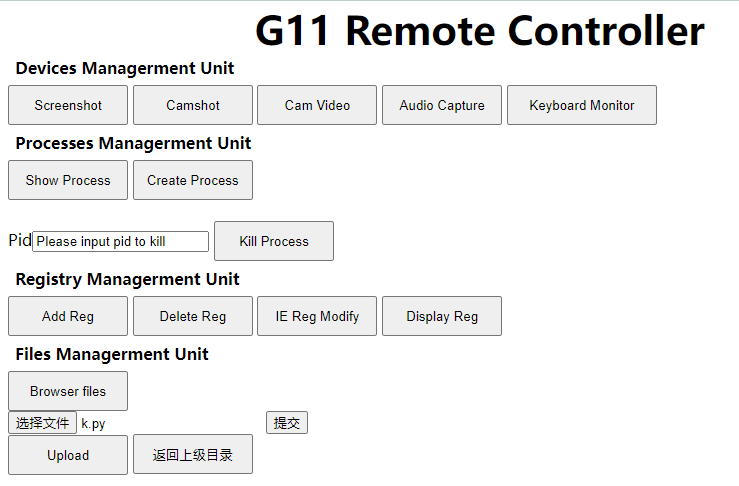


图 12 上传文件演示

在被控机的目标文件夹下可见k.py以被成功传送。

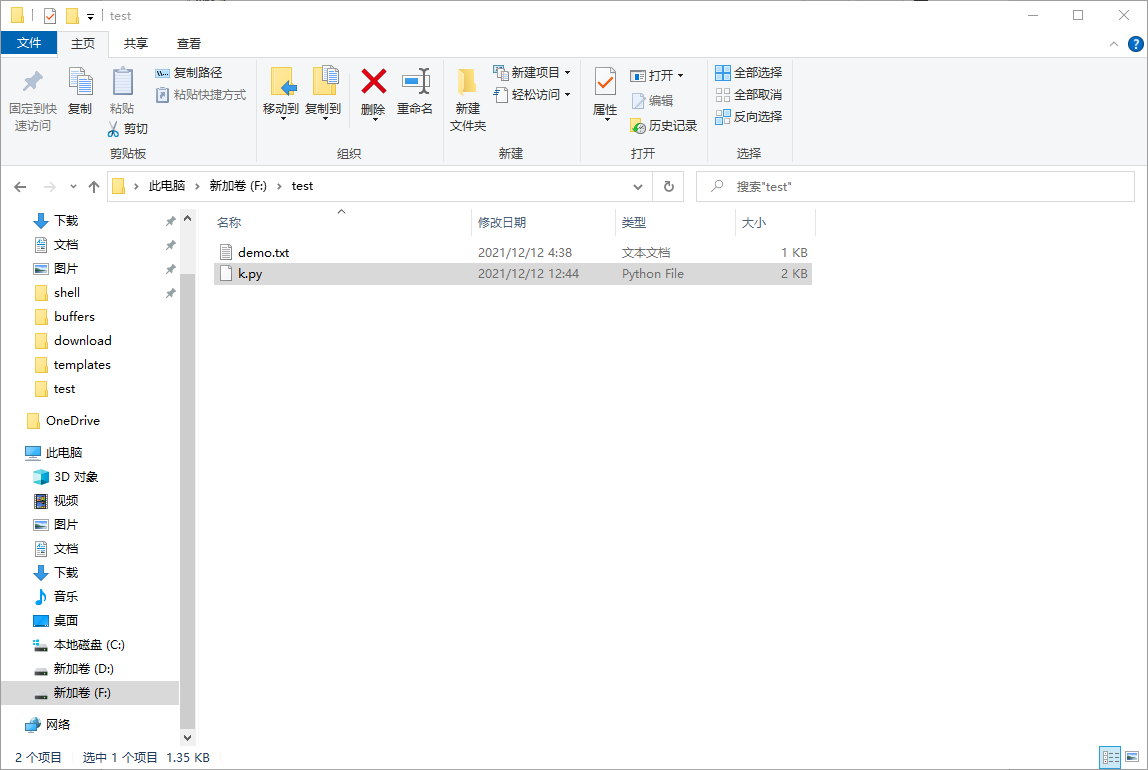


图 13 上传文件效果

### 3.2 进程管理

1. 打印被控端的进程列表

点击“Show Process”按钮，即可在右侧的输出框中显示被控端的进程列表

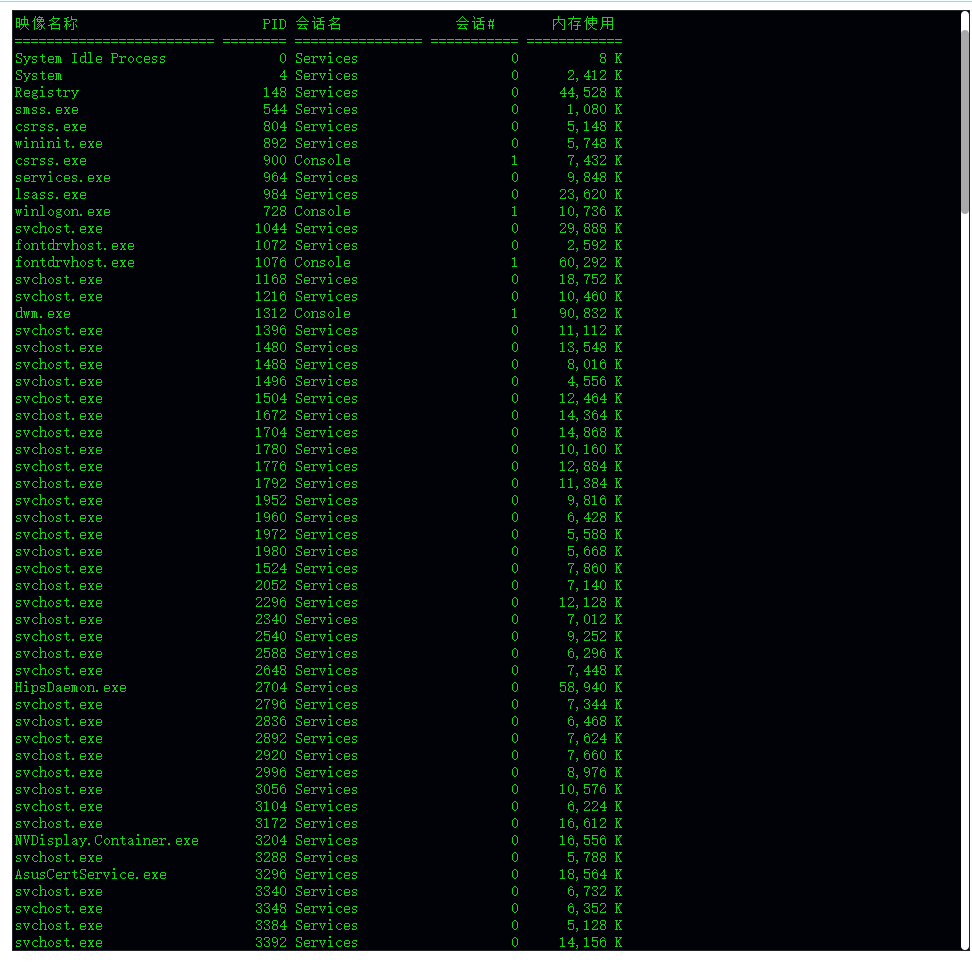


图 14 打印进程列表演示

1. 创建新的计算器进程

点击“Create Process”按钮，即可在被控端打开计算器程序。

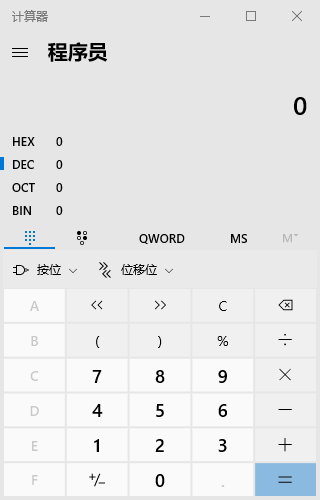


图 15 创建进程演示

同时再次打印被控端的进程列表，可见计算器进程的pid为17960。

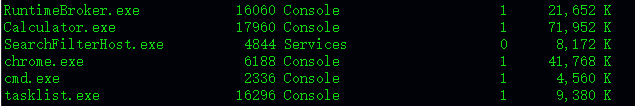


图 16 打开进程验证

1. 关闭第二步打开的计算器进程

输入第二步创建的计算器进程的pid，点击“Kill Process”按钮即可关闭该进程。



图 17 关闭进程演示

在右侧的输出窗口可见PID为17960的进程以被关闭。



图 18 关闭进程成功输出

### 3.3 注册表管理

1. 向指定位置添加注册表项

在HKEY\_CURRENT\_USER/SOFTWARE/处添加注册表项test，将其值设为“G11-注册表管理”。

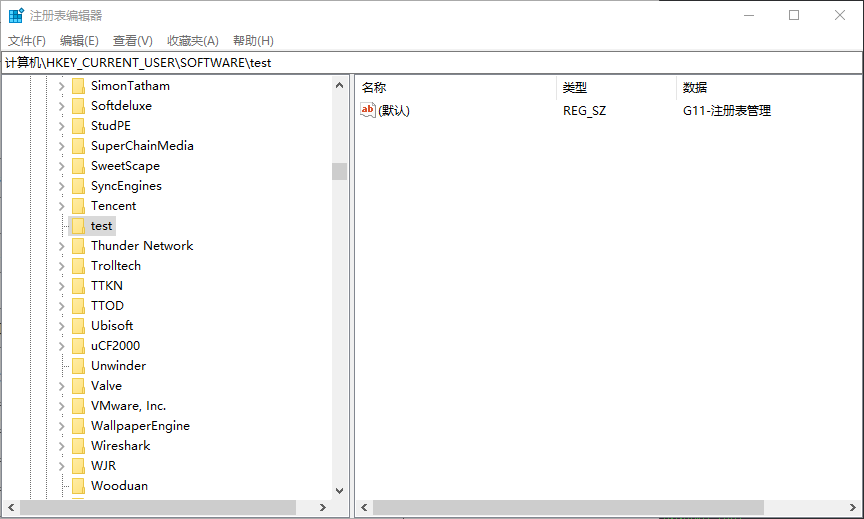


图 19 添加注册表项演示

1. 修改注册表项

修改HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Internet Explorer处的注册表默认值为“python”，“Version”项的值为7.0.2900.2180。

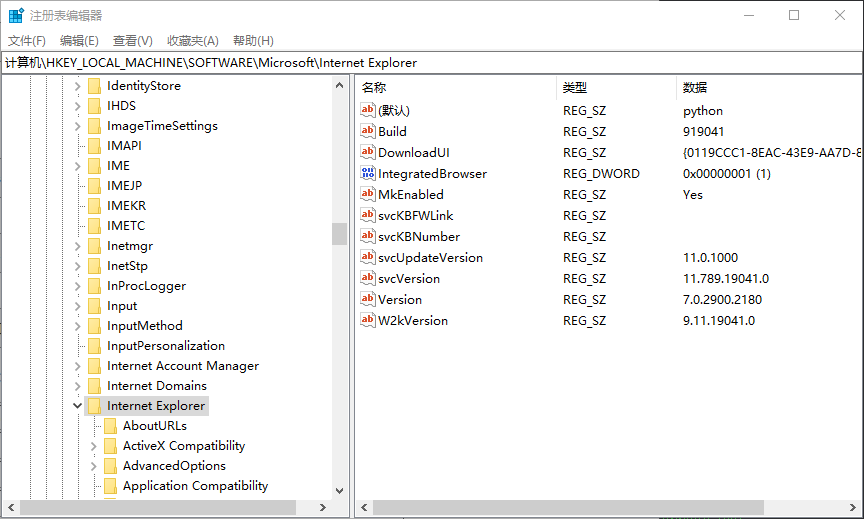


图 20 修改注册表项演示

1. 删除指定位置注册表项

删除(1)中新建的注册表项。可见该注册表项被成功删除。

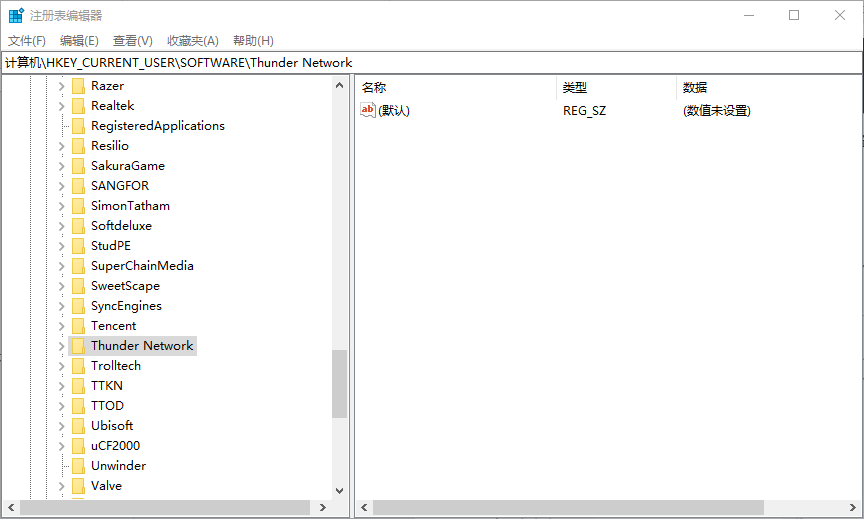


图 21 删除注册表项演示

1. 打印某一位置所有子项

打印HKEY\_CURRENT\_USER/console处的注册表子项的个数及内容。

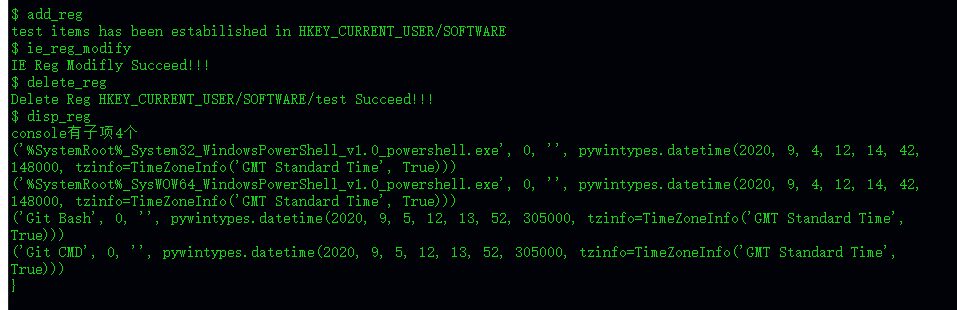


图 22 打印注册表演示

在注册表编辑器中查看对应位置的注册表项，可以验证远程控制程序所打印的注册表项是正确的。

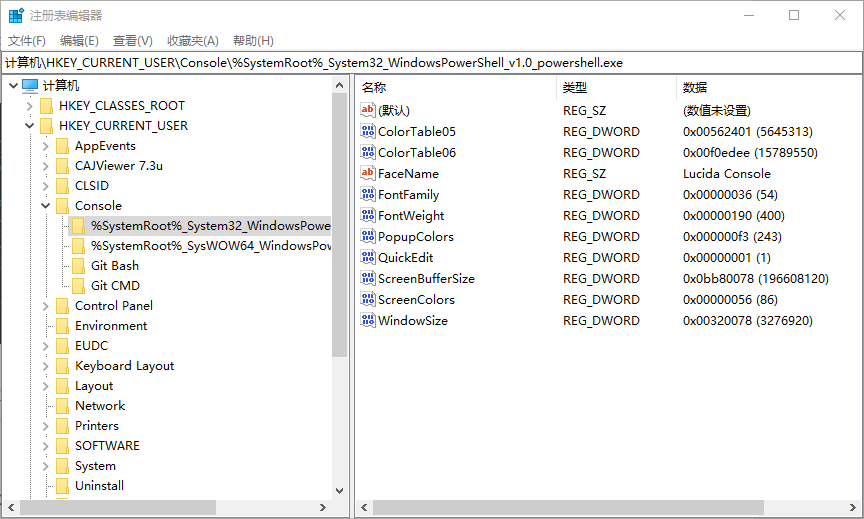


图 23 验证打印的注册表项

### 3.4 开启摄像头、麦克风

1. 控制端在被控端执行摄像头，将摄像头录制的视频文件、摄像头拍摄的照片传输到控制端的file\_from\_client文件夹下

点击Camshot按钮，发现右侧终端有提示信息产生，如图 19所示

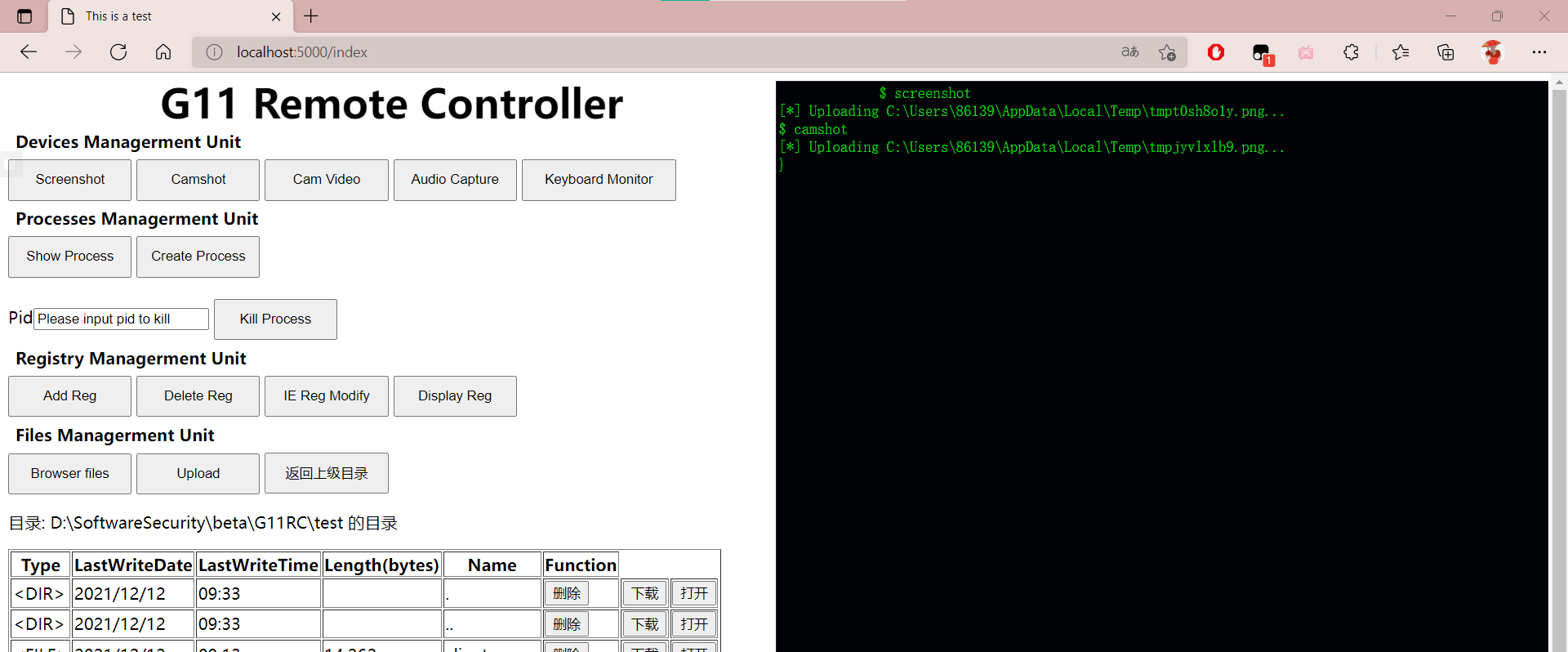


图 24 点击Camshot按钮

在file\_from\_client目录下找到了摄像头拍摄的文件如图 20所示:

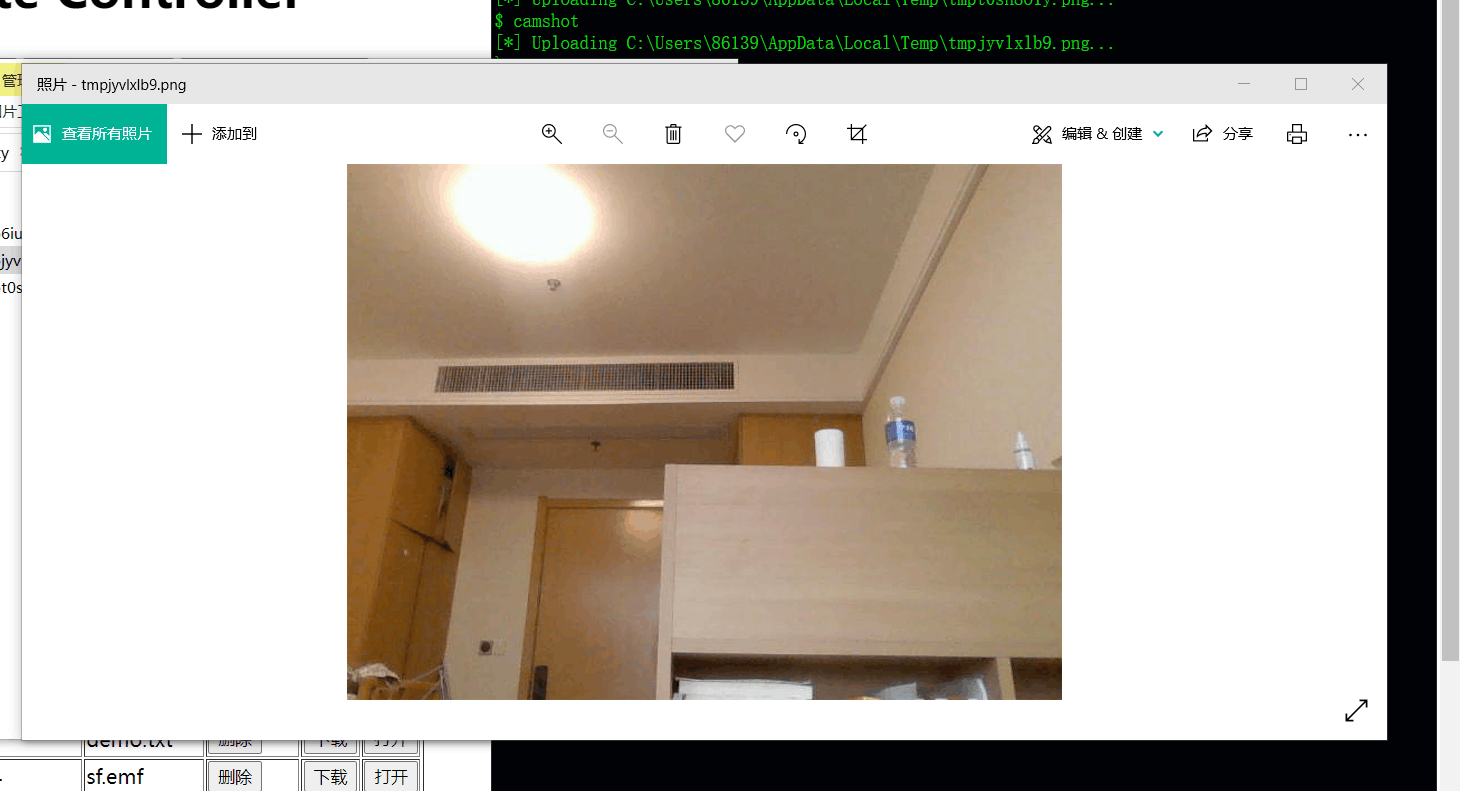


图 25 Camshot功能拍摄的照片

点击Cam Video按钮，发现右侧终端有提示信息产生，如图 26所示

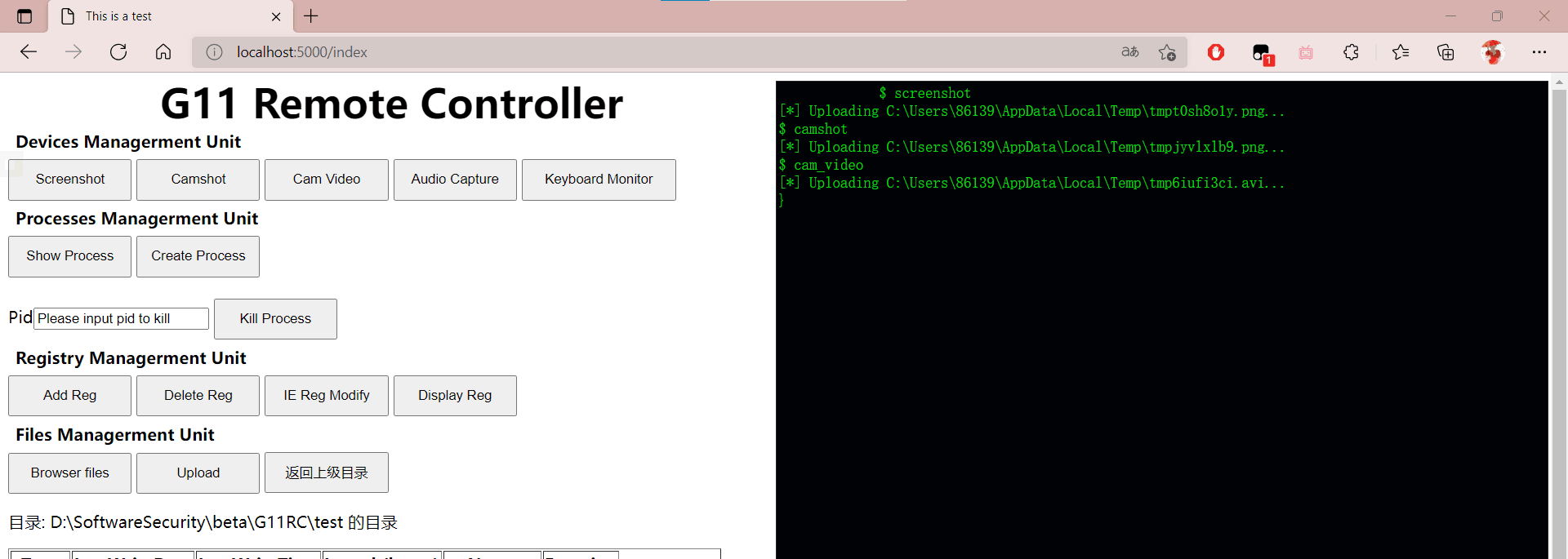


图 26 点击Cam Video按钮

file\_from\_clients文件夹下发现已经存在摄像头录制的avi视频文件:

![F:\QQ\2311339467\Image\C2C\F99R](MX{`1IGF{(}QXO3ZU.png](data:image/png;base64,)

图 27 Cam Video录制的视频

2) 麦克风功能

点击Audio Capture按钮，发现右侧中断有提示信息产生，如图 25所示

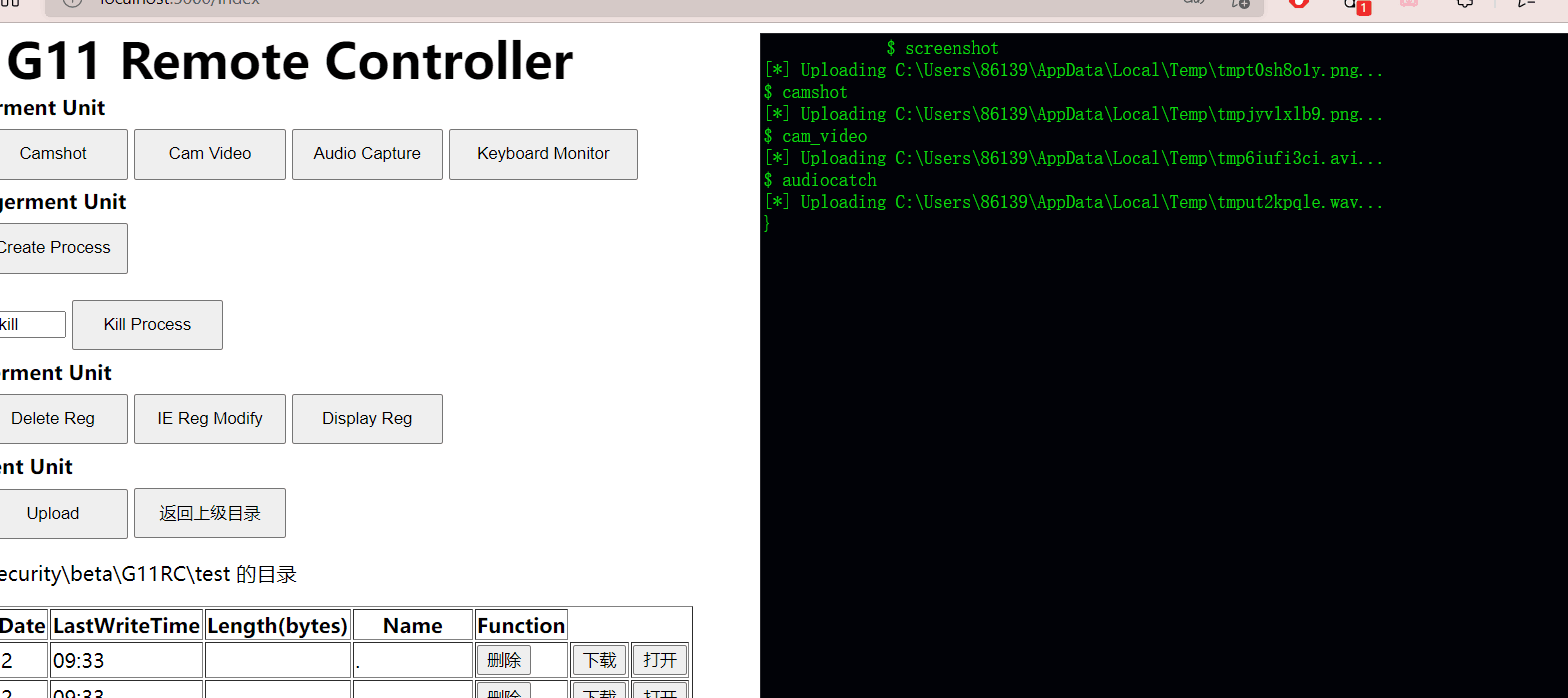


图 28 执行Audio Capture按钮

file\_from\_clients文件夹下发现已经存在摄像头录制的wav音频文件:

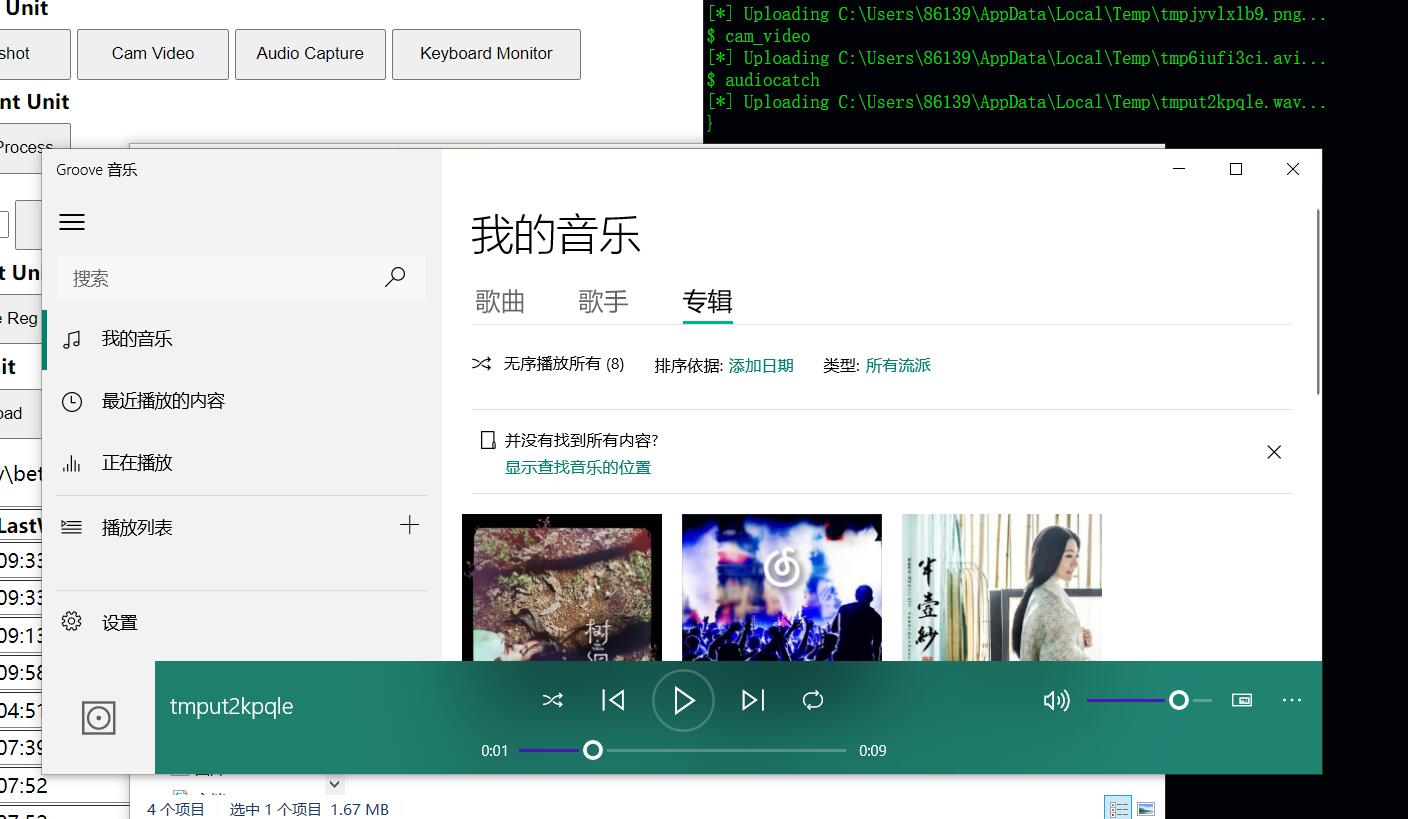


图 29 录制的音频文件

最后经过上述实验，查看此时files\_from\_client目录下的各个文件:

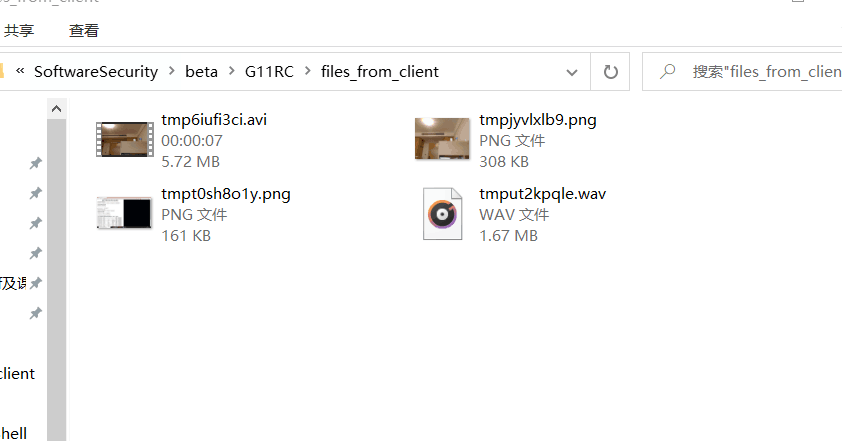


图 30 执行完一系列外设指令后的files\_from\_client文件夹

### 3.5 屏幕截取

**控制端在被控端执行屏幕截取功能，将图像文件传输到控制端。**

点击Screenshot按钮，发现右侧终端有提示信息产生，如图 31所示

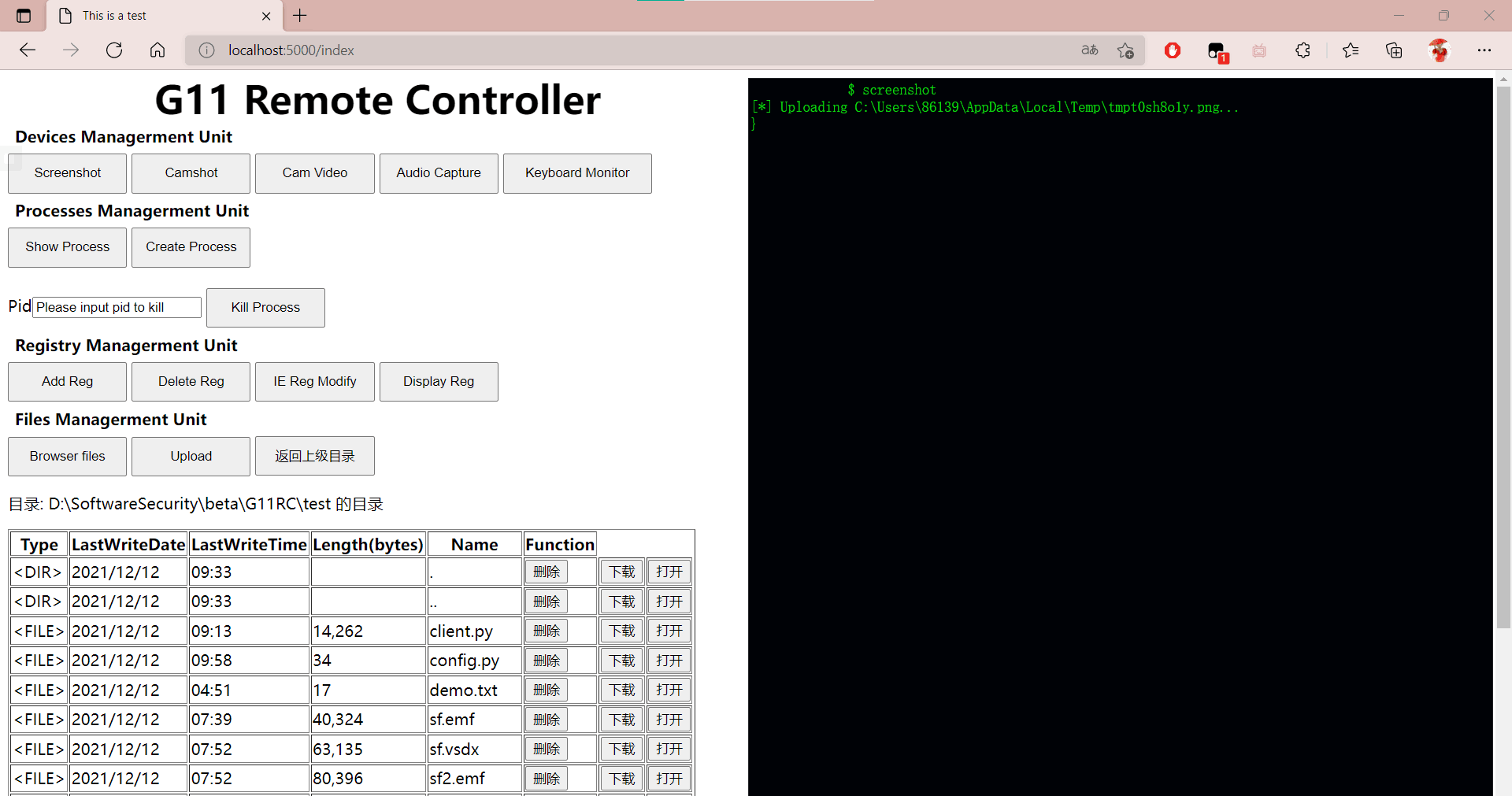


图 31 点击Screenshot按钮

file\_from\_clients文件夹下发现已经存在截屏文件:

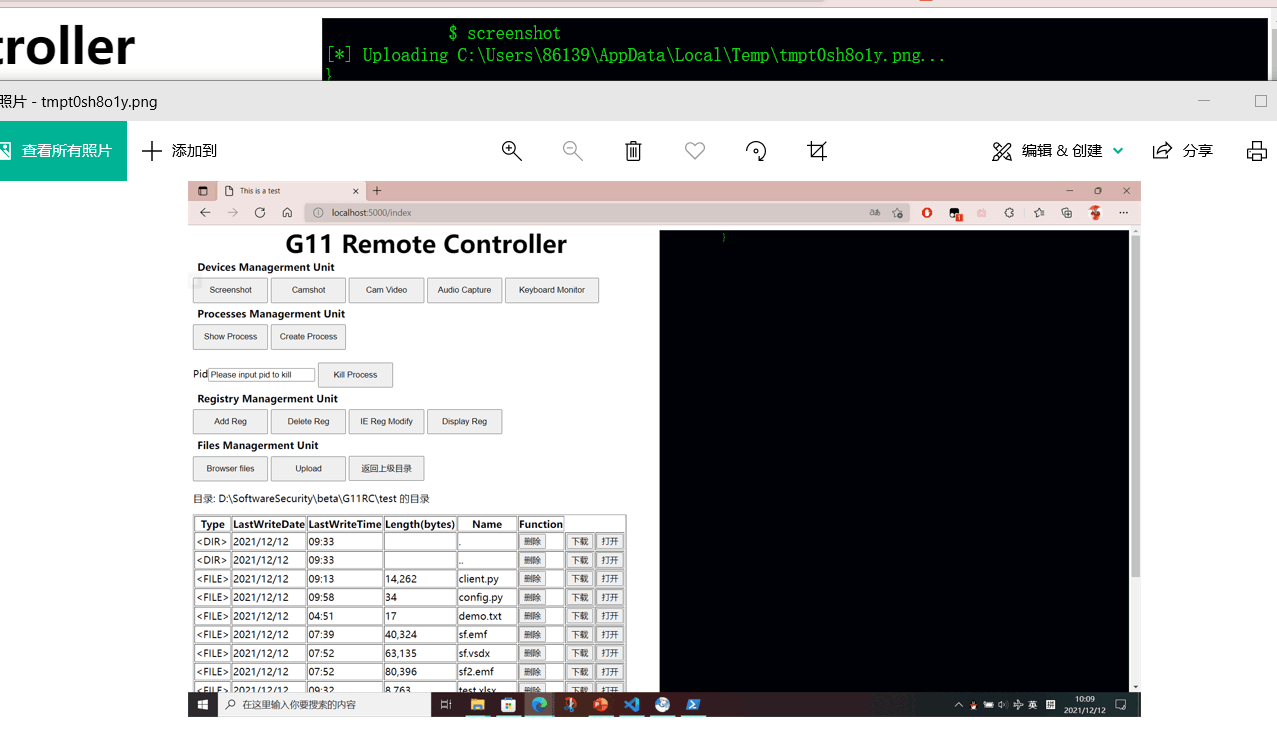


图 32 Screenshot实现的屏幕截屏

### 3.6 键盘记录

1. 控制端开启被控端的键盘记录功能，在被控端手动输入文字，控制端读取被控端的键盘记录，控制端关闭被控端的键盘记录功能

点击Keyboard Monitor按钮，发现右侧终端有提示信息产生，如图 33所示：



图 33 Keyboard Monitor执行

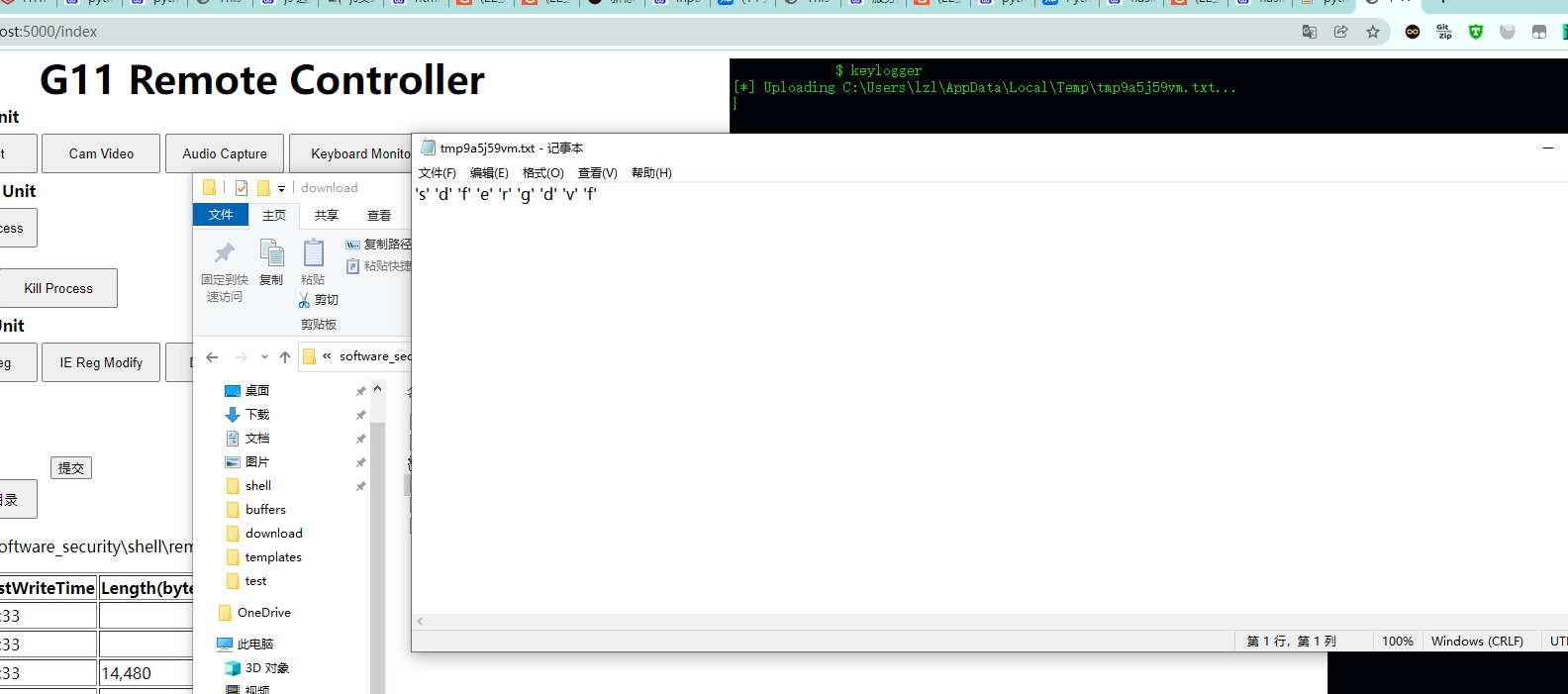
file\_from\_clients文件夹下发现已经存在记录键盘记录的文本文档，如图 34所示：

图 34 获取键盘记录的执行结果

## 4 总结

本次实验我们完成了一个具有多种远程控制功能的web应用程序，并进行了部署和测试。通过本次实验，我们对远程控制程序的结构和通信原理有了更加深入的认识。在开发过程中我们学习到了实现远程控制时权限方面的知识，socket通信的基本知识，flask框架路由的分发和处理机制，调用外部设备的方法以及web前端的UI设计知识，我们小组成员都学习到了很多，同时也对如何防范恶意的远程控制程序有了更深的理解。