+

课程设计(论文)说明书

题 目：实现一个入侵检测工具

学 院： 计算机学院

专 业： 网络工程

姓 名： 廖升波

学 号： 20200440109

指 导 教 师： 刘朝晖

2023 年 1 月 4 日

# 目录

[**1. 前言 3**](#_Toc15261)

[1.1 网络安全 3](#_Toc28808)

[1.2国内外研究现状及发展 4](#_Toc31303)

[1.3 入侵检测技术 5](#_Toc12790)

[1.4 研究目的及意义 7](#_Toc12790)

[1.5 所做工作及学习 8](#_Toc12790)

[**2. 系统分析 10**](#_Toc24786)

[2.1 基本要求及需求分析 10](#_Toc5238)

[2.2 系统环境 10](#_Toc13072)

[2.3 系统主要功能 10](#_Toc31778)

[**3. 功能设计 11**](#_Toc32253)

[3.1 设计概要 11](#_Toc15911)

[3.2 结构框图 12](#_Toc21934)

[**4 所遇到的问题及分析解决 1**](#_Toc10172)**3**

[4.1 遇到的问题及分析解决 1](#_Toc16904)3

[**5 系统特色及关键代码 1**](#_Toc8562)**4**

[5.1 主要的数据结构 1](#_Toc11020)5

[5.2 各层数据分析的函数结构 1](#_Toc1148)6

[5.2 主要函数 1](#_Toc1148)8

[**6 测试 22**](#_Toc11432)

[6.1 简单测试 22](#_Toc4383)

[6.2 Ping测试 22](#_Toc4383)

[6.3 筛选测试 23](#_Toc4383)

[6.4 日志文件 24](#_Toc4383)

[**7 结论 25**](#_Toc11432)

[7.1 完成情况 25](#_Toc4383)

[7.2 不足与缺陷](#_Toc7540) 25

[7.3 改进](#_Toc6454) 26

[7.4 结语](#_Toc6454) 27

[7.5 心得体会 2](#_Toc6454)8

[参考文献 2](#_Toc6454)9

**一、前言**

**1.1 网络安全**

**(1).什么是网路安全？**

网络安全指的是通过各种管理手段与技术手段保障网络系统的运行，增强网络数据的保密性及完整性。

**(2).为什么我们要去了解网络安全**

在计算机网络技术的不断发展和普及下，各行各业逐渐加大了对计算机的应用，而计算机由于自身防护能力差，存在严重的漏洞，为网络病毒、网络黑客、不法分子的恶意入侵和破坏提供了可乘之机，而我们去学习网络安全就是去解决以上问题。通过不断的学习，可以实现对计算机问题的深入挖掘、分析和利用，将网络安全风险降到最低，避免人们隐私信息出现泄露、丢失现象，从而更好地保障人们的人身安全和财产安全。

**(3)影响网络安全因素有哪些？**（举例几个）

1. 操作不当。在计算机使用的时候，用户应具备良好的安全意识，保证计算机操作的安全性，否则很容易出现各种各样的网络安全问题。若是在用户操作方面存在问题，容易出现黑客入侵或木马入侵等网络安全问题，或是计算机操作不符合相关的制度规范，造成了网络系统中的安全漏洞，那么黑客就可以利用漏洞入侵网络系统，严重影响到网络系统的安全性。

2. 黑客攻击。在网络系统存在漏洞的情况下，黑客能够利用网络工具入侵计算机系统，使得计算机系统的运行安全面临着较大风险。一般在发生黑客攻击之后，系统中的资源将被占用，使得网络出现堵塞的情况，容易发生网络中断，影响到计算机系统的正常运行。同时，在系统缺乏安全性能的情况下，黑客能够利用非法的手段获取系统的控制权限，使得网络的信息安全面临着威胁。同时，黑客可以将带有病毒的邮件发送给用户，从而影响网络安全；此外，病毒的清除难度大，容易给用户带来较大的损失

**(4).有哪些技术可应用于网络安全**（举例几个）

1. 信息认证技术
2. 防火墙技术
3. 网络信息加密技术
4. 网络入侵检测技术
5. 安全扫描技术

**1.2 国内外研究现状及发展**

2002年，在美国颁布的《美国联邦网络信息安全管理法案》(FISMA) 中,给出了有关网络信息安全的定义，“保护网络信息与网络信息系统,防止未授权的访问、使用、泄露、中断、修改或破坏，以保护:

(1)完整性：即防止对网络信息的不当修改或破坏，包括确保网络信息的不可否认性和真实性;

(2)保密性：即对网络信息的访问和泄露施加授权的约束，包括保护个人隐私和专属网络信息的手段;

(3)可用性：即确保能及时、可靠地访问并使用网络信息；

2005年，在ISO/IEC 17799标准中，定义网络信息安全为:“保持网络信息的保密性、完整性、可用性;另外，也包括其他属性，如真实性、可核查性、抗抵赖性和可靠性。

网络安全技术的发展具有明显的代际发展效应。在内生安全技术之前的网络安全发展主要经历了3个阶段:以阻止入侵为目的的系统加固阶段、以限制破坏为目的的检测响应阶段、以系统顽存为目的的网络容侵阶段。每一阶段的安全技术都呼应了其斯面临的安全问题。在网络规模化部署中，这对已知特征和固化模式的攻击具有重要防护意义，但各阶段均以网络攻防对抗为核心思路，缺乏安全的顶层结构化设计，难以逃脱“道高- -尺，魔高一丈”的安全困境。

第1阶段的安全技术主要通过划分明确的网络边界，利用各种保护和隔离技术手段,例如用户鉴权与认证、访问控制、信息加解密、网络隔离等，在网络边界上部署,防止外部非法入侵与信息泄露，达到系统加固的目的。

第2阶段的安全防护融合了保护、检测、响应、恢复四大技术。此阶段主要采用特征扫描、模式匹配等手段对系统状态进行检测与报警，寻找被植入的恶意代码并进行查杀，找出导致恶意代码可被植入的原因并用补丁的方式进行修补，发现不规范的蓄意行为和特征并加以抑制。此阶段技术高度依赖检测能力，且攻击方发展出对应的伪装欺骗技术，导致不可能发现全部攻击。

第3阶段的安全防护在前两阶段的基础\_上叠加了信息生存技术。此阶段网络在假设漏洞后门可避免，攻击和意外事故已然、必然发生的条件下，通过实时状况感知与响应,，实时调整安全策略，采用自我诊断隔离、还原重构等手段，仍可在限定时间内完成全部关键使命。

然而，在当下的“互联网+”时代，人工智能、物联网、云计算等新兴技术

在推动工业控制系统发生前所未有的转型变革的同时3)，信息安全事件也随之而来。特别是近年来，全球重大高级持续性工业信息安全事件频发。





**1.3 入侵检测技术**

（1）.入侵检测技术

网络安全环境是计算机运行的重要基础，在计算机遭到操作不当、恶意攻击等不安全因素影响时，极有可能出现计算机程序运行异常、数据信息丢失等情况。入侵检测技术主要是指在计算机遭受到不安全因素时，可对潜在的病毒、恶意攻击代码进行有效阻隔、删除，进而保障计算机网络系统安全的一项计算机技术。入侵检测技术可及时检测出计算机网络中存在的安全问题，及时告知防火墙系统并与其结合对不良入侵进行抵抗。入侵检测技术的应用，可有效弥补计算机防火墙技术应用中的诸多不足。

（2）.入侵检测技术的原理

入侵检测技术通过对计算机系统及用户行为的监测、识别、记录、分析,将未经授权的现象或异常状况向相关程序发出警示。计算机网络的入侵检测系统运行前，会对其安全模式予以识别、扫描，可在入侵检测系统运行中发挥有效的监测、记录作用，并逐步积累监测记录获取到的数据信息,达到相应后，系统便会对数据信息予以自动分析,并与安全模式进行比较,进一步对潜 在的安全风险进行有效识别，并对异常状况发出警示。入侵检测技术实现的方法多种样，比如基于神经网络的入侵检测、基于专家系统的入侵检测等,通常而言，入侵检测主要是以执行相关任务的方式进行实现的,具体包括:对计算机系统构造及存在的薄弱部分进行检测;对计算机系统活动、用户行为进行全面检测;对计算机系统异常状况进行统计分析;对重要系统及相关数据信息进行全面评估;对既有进攻活动模式进行识别及呈现，并向相关人员发出警示信息等。

（3）.认识snort

Snort通常被称为轻量级的NIDS, 即占用的资源少, 能运行在不同操作系统上的网络入侵检测系统, 它的免费使用为网络安全时代添上浓墨重彩的一笔。Snort最初的开发目的是做一个数据包嗅探器, 经过不断发展, 如今, 它是一个基于libpcap包的十分有效的网络监控软件, 它能够进行数据嗅探, 内容搜索匹配，协议分析, 日志记录, 检测各种攻击, 如缓冲区溢出, 隐秘端口扫描等, 以及可以发出警报。由于它的速度快, 性能好, 使用方便, 容易管理和配置等优点, 协议分析, 日志记录, 检测各种攻击, 如缓冲区溢出, 隐秘端口扫描等, 以及可以发出警报。由于它的速度快, 性能好, 使用方便, 容易管理和配置等优点, 使Snort成为该系列使用人数最多的产品之一。

Snort的结构由4大软件模块组成, 这些模块使用插件模式与Snort结合, 因此扩展起来非常方便。它的架构决定了它的各种功能, 基本组成模块是数据包嗅探器, 预处理器, 检测引擎和报警输出模块。

Snort规则是基于文本的, 通常存在于Snort程序目录中或者子目录中。规则文件按照不同的组进行分类。Snort规则分为两部分:规则头和规则体。Snort规则头是特征中的重要组成[6], 它定义了规则被触发时该做什么, 对应哪一种协议, 以及包括IP地址、网络、端口在内的源和目的信息。一个规则头分为4个部分:规则行为、协议、源信息和目的信息。规则体的作用是在规则头信息的基础上进一步的分析, 有了它才能确认复杂的攻击。

**1.4 研究目的及意义**

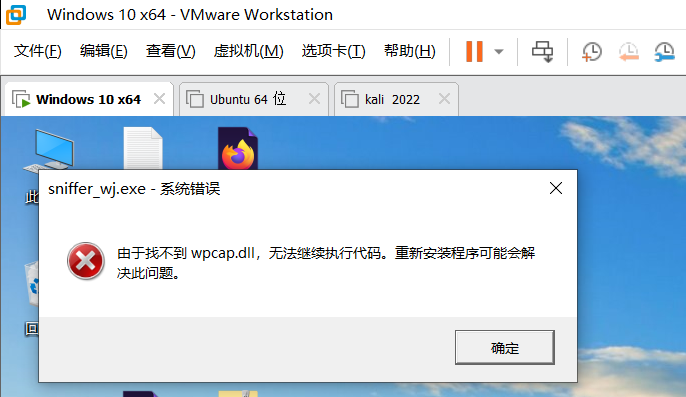
网络技术的发展为人们的生活、铲提供了极大便利，但是也带来很多安全问题。信息安全问题较为复杂、烦琐,信息内容受人为因素、自然因素的影响可能出现丢失、恶意篡改等情况。现阶段,计算机网络系统、信息系统被不法分子窥视，当系统遭受病毒、黑客入侵将造成不可逆转的损失。当前计算机网络安全威胁主要表现在黑客威胁与攻击和病毒入侵。

入侵检测系统在主机保护操作中，主要是监测分析主机在运行中的进程和日志，以此来进行相关入侵行为和信息的筛选，并及时向管理员作出警示。然后，管理员就可以以此为基础进行相应处理措施的制定和实施。在此过程中，检测程序能够实时捕获网络数据(流量)，并对其进行分析，获取数据特征值形成自身的规则库。另外，根据Snort检测规则的可定制化，使Snort能够分析网络当中各种协议类型的数据包，并对其中的非法流量和可疑数据的特征与自身规则库中的规则进行匹配。如匹配成功，则发送给指定的日志文件或提示告警信息,从而实现入侵检测的功能。当然入侵检测系统也会自动进行全部运行日志的采集存储。同时作为管理员需要完全依据日志反馈内容判断其中有无恶意进程，在某些情况下，检测系统也具备自动进行该操作的能力。为有效提升计算机网络的安全防范效果，应当对信息安全体系加以不断优化完善，保证信息安全体系规范性、合理性、科学性。另外，应当积极借助网络技术强化计算机网络安全性。首先，为避免病毒入侵恶意篡改、窃取计算机信息资料，引发经济损失，应当应用安全检测技术和防火墙，阻止恶意信息传输和发送，对网络信息加以保护。其次，防火墙主要是通过隔离方式保护计算机网络，避免病毒入侵。现阶段，防火墙的实际应用具有实用性、便捷性、高效性特点，企业安全管理中应用防火墙能够有效降低病毒入侵几率，确保网络资料信息的安全。最后将安全检测技术和防火墙融合应用，能够避免病毒入侵，提升计算机网络的防范性。但是在这里只是实现了入侵检测技术。

对于入侵检测技术研究的目的，其根本还是为计算机网络运行提供可靠稳定的保障。

**1.5 所做工作及学习**

对于入侵检测技术，我最后是通过结合本课程和协议分析课程的网络编程实现一个小程序来完成的，这个程序主要以嗅探为主，同时参考了snort的一些规则和原理进行检测，程序是使用Visual C++开发的 MFC 程序，开发/编译环境为 Microsoft Visual Studio 2022。此程序是基于winpcap开发的，所以在测试时务必保证自己的计算机安装winpcap。



关于winpcap：它是一款用于网络抓包的专业软件，用于windows系统下的直接的网络编程。winpcap功能全面，适用范围广泛。不仅可以帮助用户快速出色的将网络上的信息包进行抓取和分析，而且还可以用于网络监控、网络扫描、安全工具等各个方面，为用户带来人性化、便捷化的使用体验。

  winpcap提供了一个强大的编程接口，它很容易地在各个操作系统之间进行移植，也很方便进行开发。Winpacap适用范围很广，比如我们常见的

1、网络及协议分析

      2、网络监控

      3、通信日志记录

      4、用户级别的桥路和路由

      5、网络入侵检测系统（用于这次课设）

      6、网络扫描

      7、安全工具等等

**二、系统分析**

**2.1 基本要求及需求分析**

1. 具有嗅探功能。
2. 能够分析数据包，甚至能够对系统日志进行检测和分析。
3. 设定检测的规则，可以参考 snort 的规则。
4. 可以根据用户自己设置的规则进行报警、记录、甚至响应。
5. 能够生成入侵检测系统的日志，记录各种检测到的事件。

**2.2 系统环境**

开发/编译环境为 Microsoft Visual Studio 2022。此程序是基于winpcap开发的，所以在测试时务必保证自己的计算机安装winpcap。其实最开始想基于npcap来写的，因为winpcap很久没更新了，但是由于我的电脑不知道为什么一安装npcap然后一启动就会强制关机然后盘符乱掉，直接怎么都改不回，最后只能重装系统，因为这个我去年重装了两次系统，而且网上关于winpcap网络编程的资料也不少，所以最终选择winpcap。

**2.3系统主要功能**

1. 具有嗅探功能，能对各个网卡进行检测
2. 具有筛选过滤功能，能对TCP、UDP、ICMP等包进行筛选过滤
3. 具有分析功能，能对数据包进行分析
4. 具有日志功能，能记录到检测的事件
5. 管理员能依据日志反馈内容判断其中有无恶意进程

**三、功能设计**

**3.1 设计概要**

（1）.入侵检测原理

**用户当前行为**

**监测**

**入侵检测**

**用户历史行为**

否

**存在入侵**

是

**日志记录**

**管理员处理**

（2）.入侵检测处理过程

网络数据

触发事件

反馈处理

数据整合

**数据分析**

**写入日志**

**数据采集**

**3.2 结构框图**

**Int analyze\_frame()函数获取MAC帧头信息**

Type==0x0806

Type==0x86dd

Type==0x0800

**Int analyze\_ip()函数获取ipv4头信息**

**Int analyze\_ip6()函数获取ipv6头信息**

**Int analyze\_arp()函数获取arp头信息**

下一头0x11

协议号17

**Int analyze\_udp()函数获取udp头信息**

**Int analyze\_icmp6()函数获取icmpv6头信息**

下一头0x06

协议号6

**Int analyze\_tcp()函数获取tcp头信息**

端口80

**Int analyze\_http()函数获取http头信息**

协议号1

**Int analyze\_icmp()函数获取icmp头信息**

**四、所遇到的问题及分析解决**

（1）.怎么获取网卡设备

调用 pcap\_findalldevs()获得网卡接口信息。

int Cmcf6Dlg::lixsniff\_initCap()

{ devCount = 0;

if(pcap\_findalldevs(&alldev, errbuf) ==-1)

return -1;

for(dev=alldev;dev;dev=dev->next)

devCount++; //记录设备数return 0;

}

（2）.怎么打开指定网卡

调用 pcap\_open\_live()打开指定网卡接口，winpcap 将在此接口上侦听数据。然后调用 pcap\_datalink()、pcap\_compile()、pcap\_setfilter()分别检查是否是以太网，并对过滤器进行设置。

由于网络中过来的数据包是不同层次、不同协议的，过滤器的作用就是可以设定一些的规则来查看自己想要的数据包。

（3）.对数据包的捕获和分析

因为主进程是一个对话框，它主要的任务是处理界面交互，而数据捕获是一项后台工作，所以调用CreateThread()创建一个新的线程，再调用 lixsinff\_CapThread()函数在线程中完成数据包的捕获工作。在 lixsinff\_CapThread()中调用 pcap\_next\_ex()函数进行数据包捕获，每到达一个数据包，调用自定义的包处理函数 analyze\_frame()函数完成对捕获数据的解析。

**五、系统特色及代码**

mcf6Dlg.h, mcf6Dlg.cpp 文档主要实现 GUI 以及数据包的抓获。

Protocol.h 实现各协议头的数据结构及类型定义。

utilities.h, utilities.cpp 实现各协议头分析的功能函数。

**5.1.主要的一些数据结构（**Protocol.h**）**

- MAC 帧头信息

typedef struct ethhdr

{ u\_char dest[6]; //6 个字节 目标地址

u\_char src[6]; //6 个字节 源地址

u\_short type; //2 个字节 类型

};

- ARP头信息

typedef struct arphdr

{ u\_short ar\_hrd; //硬件类型

u\_short ar\_pro; //协议类型

u\_char ar\_hln; //硬件地址长度

u\_char ar\_pln; //协议地址长度

u\_short ar\_op; //操作码，1 为请求 2 为回复

u\_char ar\_srcmac[6]; //发送方 MAC

u\_char ar\_srcip[4]; //发送方 IP

u\_char ar\_destmac[6]; //接收方 MAC

u\_char ar\_destip[4]; //接收方 IP

};

- IP 头信息

typedef struct iphdr

{ #if defined(LITTLE\_ENDIAN)

u\_char ihl:4;

u\_char version:4;

#elif defined(BIG\_ENDIAN)

u\_char version:4;

u\_char ihl:4;

#endif

u\_char tos; //TOS 服务类型

u\_short tlen; //包总长 u\_short 占两个字节

u\_short id; //标识

u\_short frag\_off; //片位移

u\_char ttl; //生存时间

u\_char proto; //协议

u\_short check; //校验和

u\_int saddr; //源地址

u\_int daddr; //目的地址

u\_int op\_pad; //选项等

};

- TCP 头信息

typedef struct tcphdr

{ u\_short sport; //源端口地址 16 位

u\_short dport; //目的端口地址 16 位

u\_int seq; //序列号 32 位

u\_int ack\_seq; //确认序列号

u\_short doff\_flag; //头大小、保留位、标志位u\_short window; //窗口大小 16 位

u\_short check; //校验和 16 位

u\_short urg\_ptr; //紧急指针 16 位

u\_int opt; //选项

};

- UDP 头信息

typedef struct udphdr

{ u\_short sport; //源端口 16 位

u\_short dport; //目的端口 16 位

u\_short len; //数据报长度 16 位

u\_short check; //校验和 16 位

};

- ICMP 头信息

typedef struct icmphdr

{ u\_char type; //8 位 类型

u\_char code; //8 位 代码

u\_char seq; //8 位序列号

u\_char chksum; //8 位校验和

};

- IPv6 头信息

typedef struct iphdr6

{ u\_int version:4, //版本

flowtype:8, //流类型

flowid:20; //流标签

u\_short plen; //有效载荷长度

u\_char nh; //下一个头部

u\_char hlim; //跳限制

u\_short saddr[8]; //源地址

u\_short daddr[8]; //目的地址

};

- ICMP6 头信息

typedef struct icmphdr6

{ u\_char type; //8 位 类型

u\_char code; //8 位 代码

u\_char seq; //序列号 8 位

u\_char chksum; //8 位校验和

u\_char op\_type; //选项：类型

u\_char op\_len; // 选 项 ： 长 度 u\_char op\_ethaddr[6]; //选项：链路层地址

};

- 保存数据结构

typedef struct datapkt

{ char pktType[8]; //包类型

int time[6]; //时间

int len; //长度

struct ethhdr\* ethh; //链路层包头

struct arphdr\* arph; //ARP 包头

struct iphdr\* iph; //IP 包头

struct iphdr6\* iph6; //IPV6

struct icmphdr\* icmph; //ICMP 包头

struct icmphdr6\* icmph6; //ICMPv6 包头

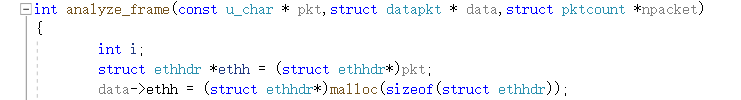
struct udphdr\* udph; //UDP 包头

struct tcphdr\* tcph; //TCP 包头

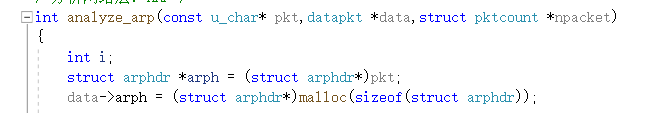
};

**5.2.各层数据分析的函数结构（**utilities.cpp**）**

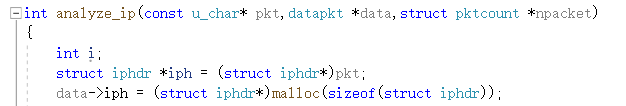
（1）.链路层



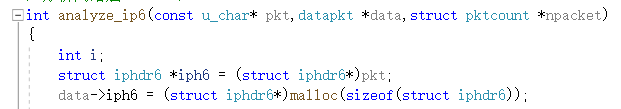
（2）.网络层ARP



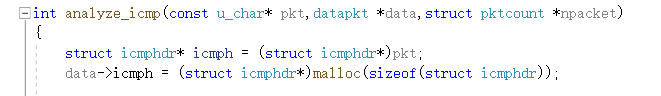
（3）.网络层IP



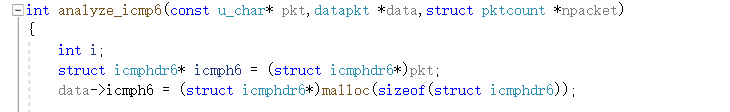
（4）.网络层ipv6



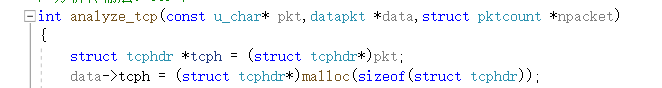
（5）.传输层ICMP



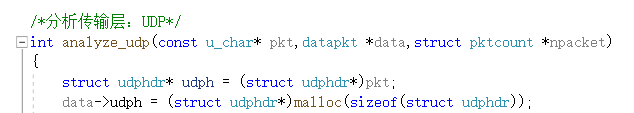
（6）.传输层ICMPv6



（7）.传输层TCP



（8）.传输层UDP



**5.3.主要函数（**mcf6Dlg.cpp**）**

//初始化winpcap

int Cmcf6Dlg::lixsniff\_initCap()

{

devCount = 0;

if(pcap\_findalldevs(&alldev, errbuf) ==-1)

return -1;

for(dev=alldev;dev;dev=dev->next)

devCount++;

return 0;

}

//更新信息

int Cmcf6Dlg::lixsniff\_updateEdit(int index)

{

POSITION localpos,netpos;

localpos = this->m\_localDataList.FindIndex(index);

netpos = this->m\_netDataList.FindIndex(index);

struct datapkt\* local\_data = (struct datapkt\*)(this->m\_localDataList.GetAt(localpos));

u\_char \* net\_data = (u\_char\*)(this->m\_netDataList.GetAt(netpos));

CString buf;

print\_packet\_hex(net\_data,local\_data->len,&buf);

//this-

this->m\_edit.SetWindowText(buf);

return 1;

}

//更新统计数据

int Cmcf6Dlg::lixsniff\_updateNPacket()

{

CString str\_num;

str\_num.Format(\_T("%d"),this->npacket.n\_arp);

this->m\_editNArp.SetWindowText(str\_num);

str\_num.Format(\_T("%d"),this->npacket.n\_http);

this->m\_editNHttp.SetWindowText(str\_num);

str\_num.Format(\_T("%d"),this->npacket.n\_icmp);

this->m\_editNIcmp.SetWindowText(str\_num);

str\_num.Format(\_T("%d"),this->npacket.n\_ip6);

this->m\_editNIp.SetWindowText(str\_num);

str\_num.Format(\_T("%d"),this->npacket.n\_other);

this->m\_editNOther.SetWindowText(str\_num);

str\_num.Format(\_T("%d"),this->npacket.n\_sum);

this->m\_editNSum.SetWindowText(str\_num);

str\_num.Format(\_T("%d"),this->npacket.n\_tcp);

this->m\_editNTcp.SetWindowText(str\_num);

str\_num.Format(\_T("%d"),this->npacket.n\_udp);

this->m\_editNUdp.SetWindowText(str\_num);

str\_num.Format(\_T("%d"),this->npacket.n\_ip);

this->m\_editNIpv4.SetWindowText(str\_num);

str\_num.Format(\_T("%d"),this->npacket.n\_icmp6);

this->m\_editIcmpv6.SetWindowText(str\_num);

return 1;

}

/\*获得源IP\*/

buf.Empty();

if(0x0806== data->ethh->type)

{

buf.Format(\_T("%d.%d.%d.%d"),data->arph->ar\_srcip[0],

data->arph->ar\_srcip[1],data->arph->ar\_srcip[2],data->arph->ar\_srcip[3]);

}else if(0x0800 == data->ethh->type){

struct in\_addr in;

in.S\_un.S\_addr = data->iph->saddr;

buf = CString(inet\_ntoa(in));

}else if(0x86dd == data->ethh->type){

int i;

for(i=0;i<8;i++)

{

if(i<=6)

buf.AppendFormat(\_T("%02x-"),data->iph6->saddr[i]);

else

buf.AppendFormat(\_T("%02x"),data->iph6->saddr[i]);

}

}

pthis->m\_listCtrl.SetItemText(nItem,6,buf);

/\*获得目的IP\*/

buf.Empty();

if(0x0806 == data->ethh->type)

{

buf.Format(\_T("%d.%d.%d.%d"),data->arph->ar\_destip[0],

data->arph->ar\_destip[1],data->arph->ar\_destip[2],data->arph->ar\_destip[3]);

}else if(0x0800 == data->ethh->type) {

struct in\_addr in;

in.S\_un.S\_addr = data->iph->daddr;

buf = CString(inet\_ntoa(in));

}else if(0x86dd == data->ethh->type){

int i;

for(i=0;i<8;i++)

{

if(i<=6)

buf.AppendFormat(\_T("%02x-"),data->iph6->daddr[i]);

else

buf.AppendFormat(\_T("%02x"),data->iph6->daddr[i]);

}

}

//调用 pcap\_dump\_open()先创建一个文件，捕获的数据将会存储到此文件中，后面捕获的数据包将会实时地写入临时文件。文件默认存储在 SaveData 文件中，文件名为存储时的时间，并且在捕获数据结束时，可以选择将此文件存储于指定路径。

CFileFind file;

char thistime[30];

struct tm \*ltime;

memset(filepath,0,512);

memset(filename,0,64);

if(!file.FindFile(\_T("SavedData")))

{

CreateDirectory(\_T("SavedData"),NULL);

}

time\_t nowtime;

time(&nowtime);

ltime=localtime(&nowtime);

strftime(thistime,sizeof(thistime),"%Y%m%d %H%M%S",ltime);

strcpy(filepath,"SavedData\\");

strcat(filename,thistime);

strcat(filename,".lix");

strcat(filepath,filename);

dumpfile = pcap\_dump\_open(adhandle, filepath);

if(dumpfile==NULL)

{

MessageBox(\_T("文件创建错误！"));

return -1;

}

//接收数据，新建线程处理。因为主进程是一个对话框，它主要的任务是处理界面交互，而数据捕获是一项后台工作，所以调用CreateThread()创建一个新的线程，再调用 lixsinff\_CapThread()函数在线程中完成数据包的捕获工作。在 lixsinff\_CapThread()中调用 pcap\_next\_ex()函数进行数据包捕获，每到达一个数据包，调用自定义的包处理函数 analyze\_frame()函数完成对捕获数据的解析。

LPDWORD threadCap=NULL;

m\_ThreadHandle=CreateThread(NULL,0,lixsinff\_CapThread,this,0,threadCap);

if(m\_ThreadHandle==NULL)

{

int code=GetLastError();

CString str;

str.Format(\_T("创建线程错误，代码为%d."),code);

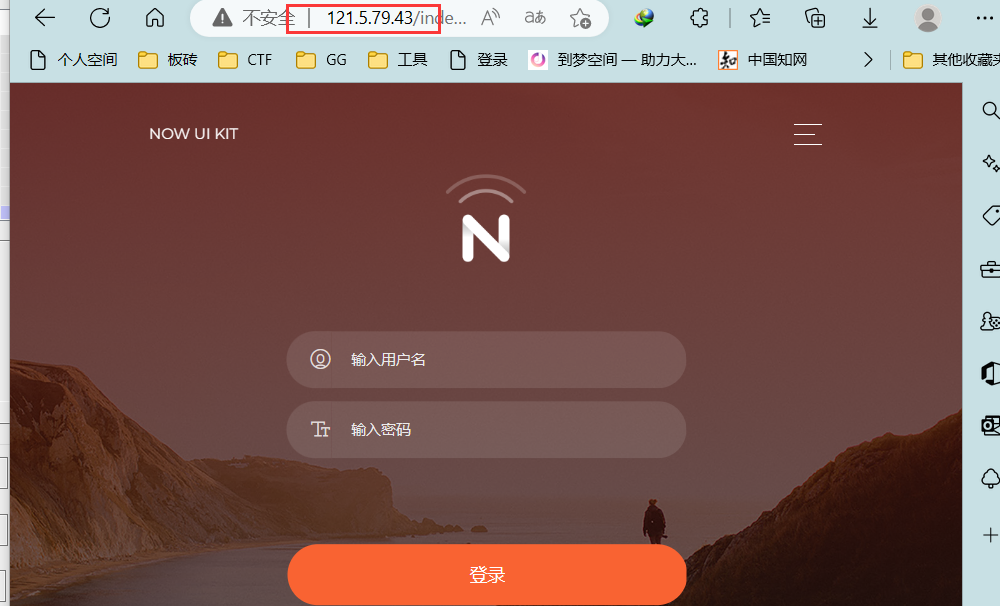
MessageBox(str);

return -1;

}

**六、测试**

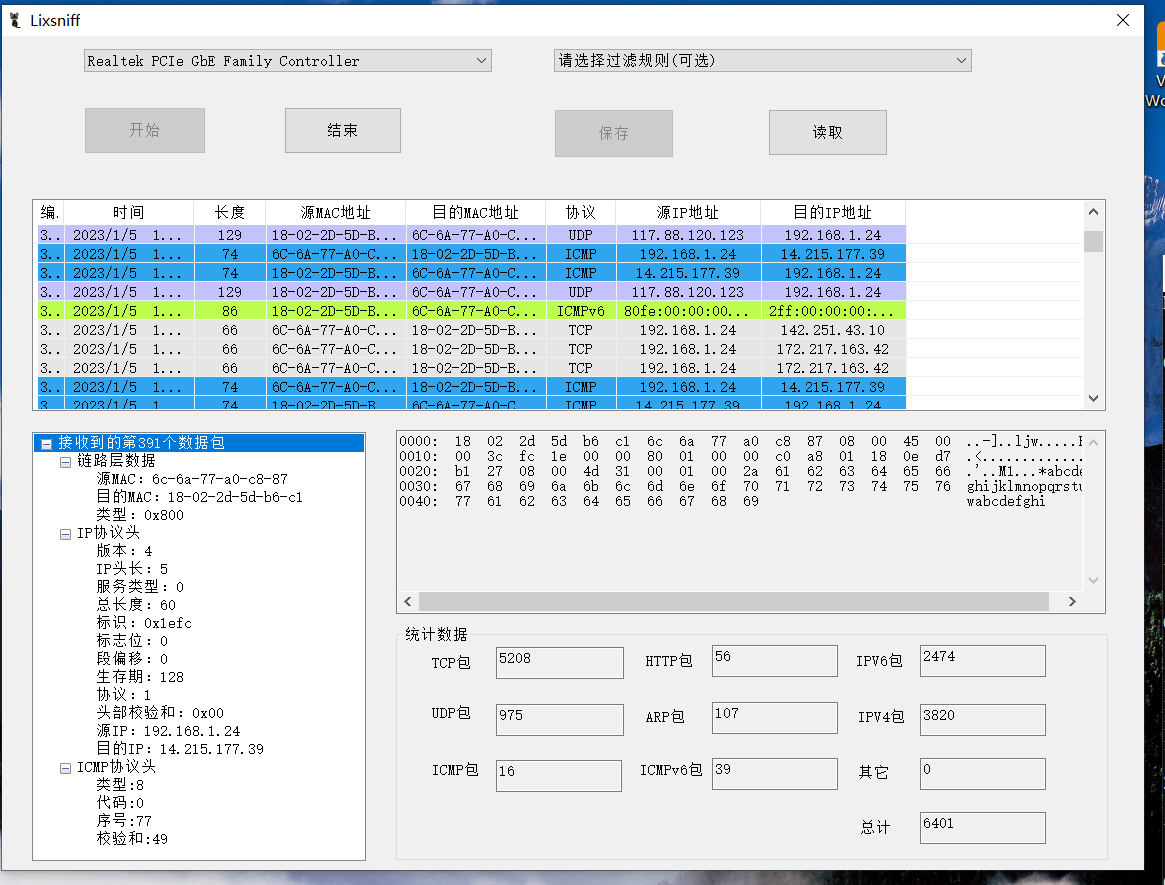
**6.1.简单测试，访问一下自己搭的服务器**





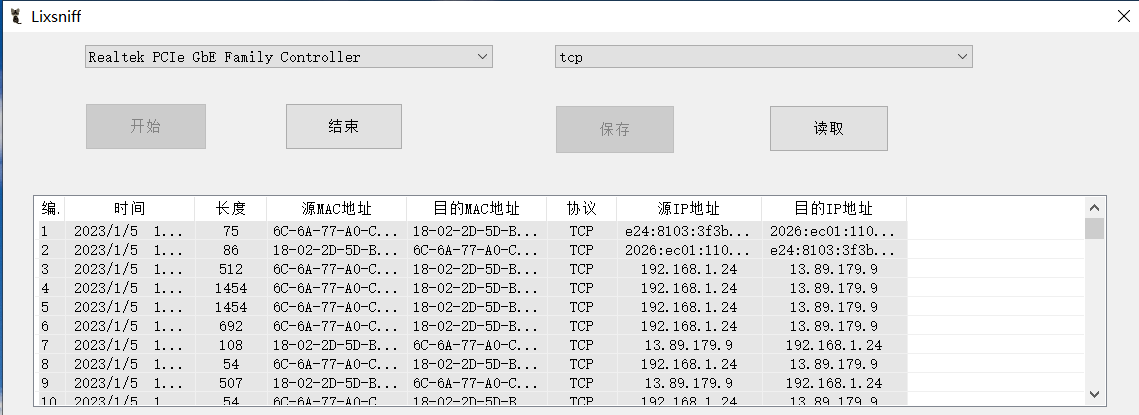
**6.2.ping 测试，可以看到抓到ICMP的百度包**



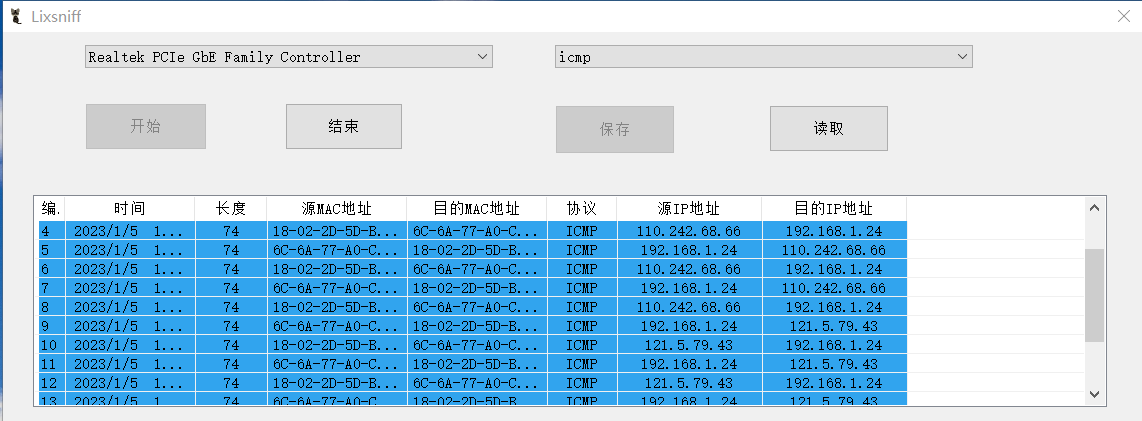


**6.3.筛选过滤规则测试**

筛选TCP的数据包

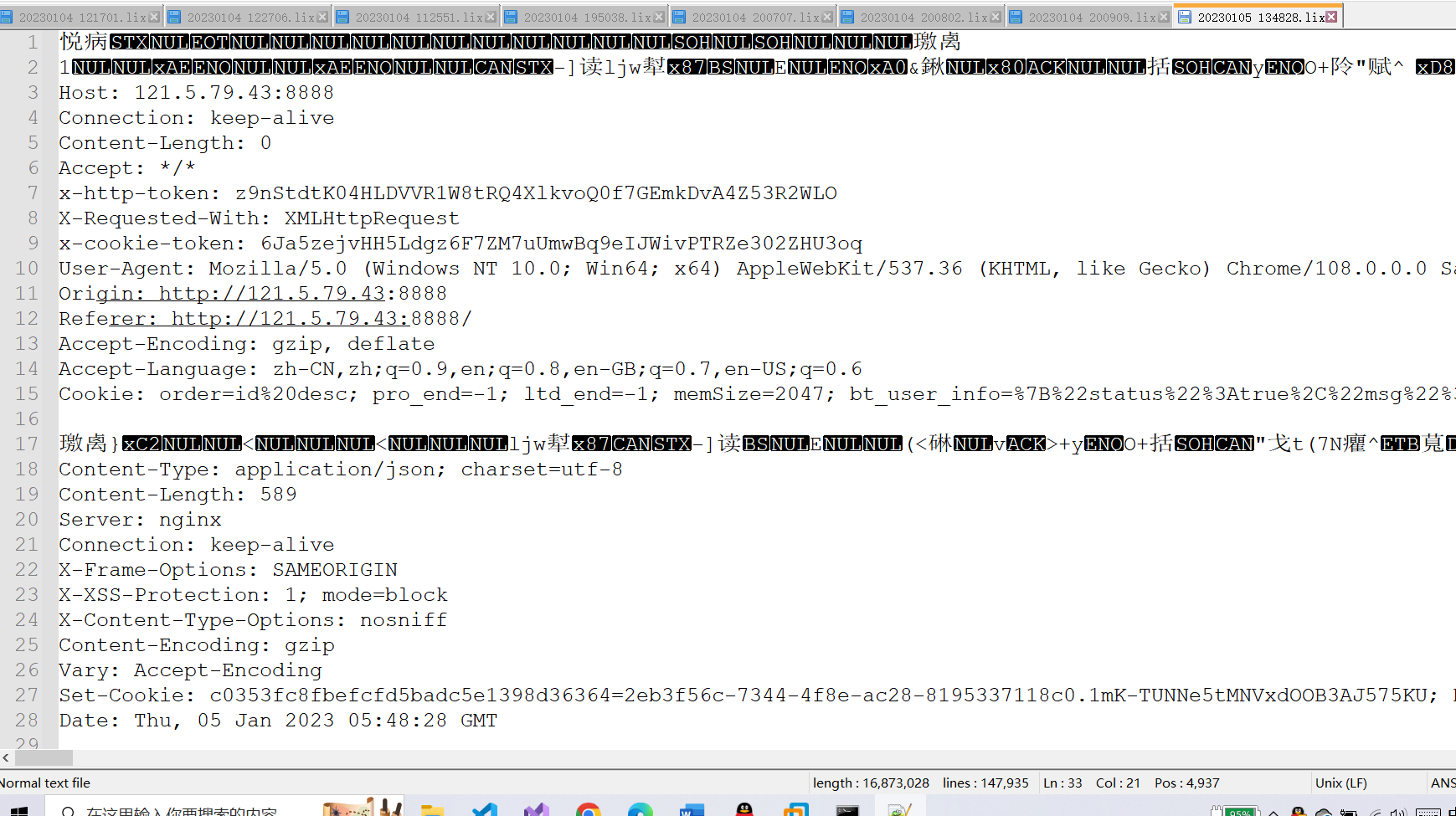


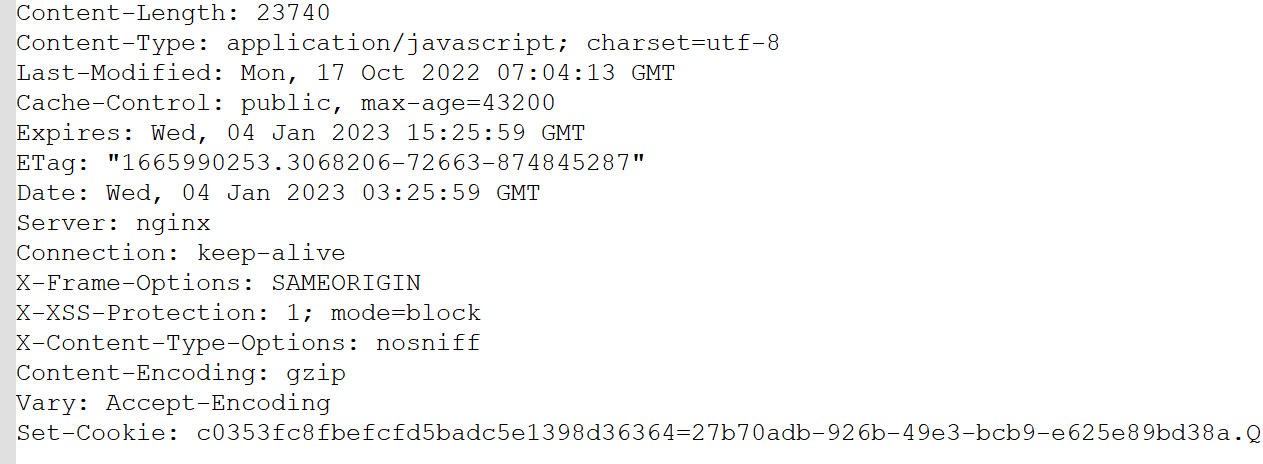
筛选ICMP的包

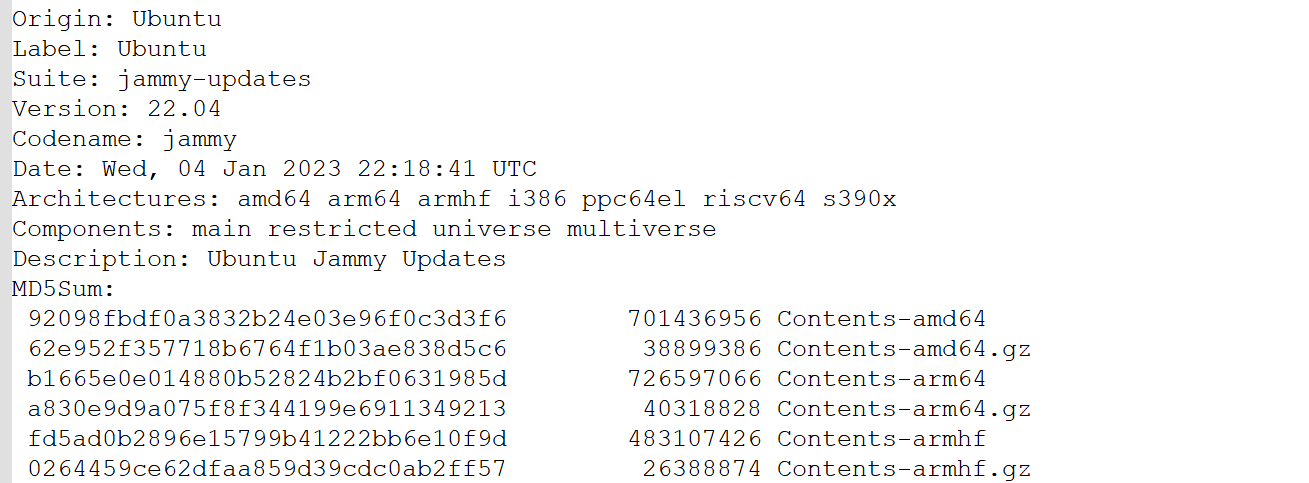


**6.4 日志文件**









**七、结论**

**7.1 完成情况**

1. 具有嗅探功能，主要是对本机进行嗅探监听获取发送给其他进程的数据包的过程。
2. 能够分析数据包，能够对系统日志进行检测和分析，以此来进行相关入侵行为和信息的筛选，然后，管理员就可以以此为基础进行相应处理措施的制定和实施。
3. 设定检测的规则，选择监听的网卡并可以实现对数据包的筛选过滤。

**7.2 不足与缺陷**

(1)有漏报问题.

尽管此程序能完成对于数据信息的筛选，但是不能完全覆盖所有信息识别。这是因为现代的网络环境具有复杂性,不仅数据量浩如烟海，而且在种类上也愈发多元，并处在快速增长中。这给入侵检测技术的研究和应用带来诸多不确定性,使其入侵检测识别的功能，包括采用的算法、应用的数学模型等，在对比之下显得局限。由此,数据库入侵检测技术内嵌于计算机数据库中后，以便能够保证进行常态化的入侵行为识别、防御，- 边也会在未知的新增有害程序或到访行为面前显出无力，出现-定概率的漏报甚至错报问题。

(2)可扩张性不足

一方面，入侵检测技术的研究与设置更新角度而言，于其系统内的功能参数均依靠技术人员的人工设置，所以更新提速慢。而且，系统中一些参数还涉及重复设置，一定程度上要损耗昂贵的人力、资金、时间等成本，使得其技术应用更新自带局限性。在信息发达的当下社会，各种新型木马病毒、技术黑客层出不穷，让入侵检测技术研究开发人员应接不暇，其相应的技术不能及时有效应对(识别、检测)， 所以可扩张性的不足更被突显。另一方面，就用户角度而言，他们往往缺乏有原理依据的技术支撑，仅能按照入侵检测系统的操作说明来操作，而且已习惯于“一键查杀”,不懂得深度应用系统的潜在功能。这就使得一些在日常可以被检测、防御的系统危险因素有了可乘之机，降低了入侵检测系统的实际运行作用，不仅危害了计算机数据安全，还局限了入侵检测技术的应用。

(3)检测效率低

由于入侵检测技术在开发升级过程中,难免有未知领域没能攻克，所以其技术对危险的干扰因素的排除也是有局限性的，就出现了-些入侵识别因素配置不准的现象。这造成计算机要反复进行检测或检测失败的情况出现，也会增加相应成本。入侵检测技术的执行所依托的是对二进制的进项进行识别转化，其转化速度与检测效率呈正相关。显然，当计算机运行负荷较低，即所要处理的信息数据量不大时，相对应的检测操作会更快执行，效率更高;反之，计算机运行负荷高，要处理海量数据，比如收到DDos攻击时，就容易受到干扰和影响检测效率。

**7.3 改进**

(1) 提升数据库的入侵检测

数据写入日志时会出现乱码错乱，在这种背景条件下，入侵检测技术若想获得效率的明显改善，就需以二进制编码为基础,尽快建立和完善数据编码的转换标准,从而能更全面地覆盖计算机网络中可能承载的数据，能够在相应类型数据出现时对其实时记录。由此，入侵检测技术的存在才更有价值，更有效。这种方下的检测效率实现提高。

(2) 使入侵检测进入智能化应用状态

入侵检测技术可融合人工智能技术，以提高数据库自我防御的自动化、智能化水平。例如，基于人工智能的深度分析和学习功能，计算机所遭受过的入侵历史即可转化为系统的研究素材，从而帮助防御系统更加有针对性地进行恶意入侵行为判断、处理。而智能化技术也可以进一步优化数据库入侵检测系统的反应机制，使其各项指令的发送、启动等都更加精准、迅速，为计算机网络运行减负。

（3）结合防火墙技术一起使用

借助于入侵检测监测到的各种类型网络数据、网络地址以及网络端口的打包处理来实现数据的筛选与处理。将被检查数据流中各个数据包的目的地址、源地址、TCP端口号以及TCP链路状态等作为依据，将其和预定义好的规则进行比对，如果数据包与规则相符，防火墙便允许其进入到内部网络中，如果不相符，防火墙便会将相应的数据包删除。同时也可以将指定的过滤编辑应用到特定的网络协议中，在过滤时进行数据包的必要统计、分析与登记，并形成相应的报告。在这样的情况下，用户便可及时发现防火墙的运行状态，并明确安全问题所在。

**7.4 结语**

近年来，随着网络的普遍应用，在我国网络安全事件层出不穷，发生了各种各样类型的网络安全事件,发生的比例呈上升趋势,几乎所用的网络用户访问过携带木马的web网页，绝大多数用户的电脑感染过病毒，超过- -半的用户个人信息被盗窃、被篡改，还有一部分用户被钓鱼网站所欺骗，网络诈骗、网络盜窃等行为使人们防不慎防。为了获取经济利益，某些技术人员通过制造贩卖病毒木马、网络盗窃及诈骗、教授网络攻击技术等行为进行网络犯罪活动，造成了巨大的经济损失和安全威胁，严重影响了我国互联网事业的健康、和谐发展。面对如此严峻的挑战，对网络恶意攻击事件的防范就显得尤为重要，针对防火墙、加密、VPN等技术无法解决的网络安全问题，入侵检测系统却可以利用其先进的分析技术实时、动态地检测出入侵攻击事件，并及时做出响应。这时入侵检测的优越性就体现出来了，它不仅可以对入侵事件进行检测，还可以很好的进行处理，防止入侵事件进一步进行渗透、 攻击，在网络安全防范体系中的成为-一个重要的角色。但还是有许多不足缺陷

应使入侵检测技术朝智能化发展。科学技术的飞速发展，为人们生活带来便利的同时也使得入侵技术朝着多样化智能化的方向发展。我国应加强计算机网络技术入侵检测技术向着智能化方向发展，进一步加强入侵检测技术对异常数据的检测灵敏度。因此，智能化计算机网络入侵技术应集模糊技术、神经网络等主要检测方式为-身,只有这样入侵技术才能在遇到网络入侵的时候对其实现精准的识别，并及时采取有效措施，将入侵风险拒之门外，保护网络系统的平稳运行。

建立计算机全方位网络防御系统.国民经济在国家的宏观调控下，发生了巨大的改变。为不断满足人们和企业的生活和生产需要，信息化技术不得不进行更新迭代，这就使得针对网络安全的入侵技术出现了多样化和智能化的发展。仅仅依靠对入侵技术的检测、分析和处理，尚不能满足网络技术的发展需要，因此，需要建立计算机全方位网络防御系统。通过建立全方位立体式的网络防御系统，可有效提升入侵检测技术对异常数据的识别能力，使得计算机资源充分被利用，进一步维护网络环境的安全稳定运行。

**7.5 心得体会**

这次课设让我学到了不少的东西，主要是本学期协议分析网络编程课程的一些知识和老师讲的snort的一些规则结合使用的学习。对TCP/IP、UDP、ICMP的套接字编程技术也更加熟练，对于一些库函数的使用也更加熟练。这次课设也花了不少时间，主要是数据包捕获这一功能模块花了不少时间，因为主进程是一个对话框，它主要的任务是处理界面交互，而数据捕获是一项后台工作，所以调用CreateThread()创建一个新的线程，再调用 lixsinff\_CapThread()函数在线程中完成数据包的捕获工作。在 lixsinff\_CapThread()中调用 pcap\_next\_ex()函数进行数据包捕获，每到达一个数据包，调用自定义的包处理函数 analyze\_frame()函数完成对捕获数据的解析。在筛选规则这里，对TCP、UDP、HTTP数据包筛选也是非常感谢B站的“去图书馆约我”up的视频的参考。（课设手机查看好像排序很乱的）

网络安全课程也就此结束了，但是我们学习网络安全这条路永远不会停下，最后感谢老师这一学期的辛勤教导，让我们打开了网络安全的大门

**参考文献**

**[1]田里,喻潇,王捷,王晋.基于Snort的网络安全入侵检测预防系统设计[J],2021,29(18):148-151+156.**

**[2]沈春马.大数据背景下计算机网络信息安全问题分析[J].网络安全技术与用,2022,(12):167-169.**

**[3]郑丽华,陈丽卿.基于Snort入侵检测系统的分析与应用[J].福建电脑,2015,31(06):117-119.**

**[4]袁华兵.基于身份加密的智慧校园网络入侵检测系统[J].自动化技术与应用,2022,41(12):96-100.**

**[5]刘红阳.基于Snort的PROFINET入侵检测系统设计[J].工业控制计算机,2019,32(08):10-12+15.**

**[6]李伟,张建锋.基于winPcap的网络行为审计系统设计[J].中国新通信,2018,20(06):43-44.**

**[7]刘凌龙. 基于WINPCAP的网络入侵检测系统设计与实现[D].辽宁大学,2015.**