

# 11 基于网络的入侵检测技术

2019年5月30日 16:03

## 1 分层协议模型与TCP/IP协议

- [1.1 TCP/IP协议模型](#)
- [1.2 TCP/IP协议报文格式](#)

## 2 网络数据包的截获

## 3 检测引擎的设计

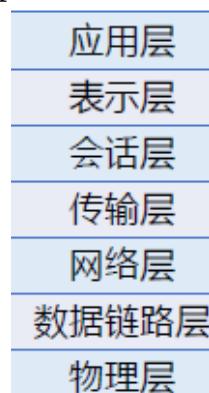
- [3.1 嵌入式规则检测引擎设计★](#)
- [3.2 可编程的检测引擎设计](#)
- [3.3 特征分析与协议分析技术★](#)

## 1 分层协议模型与TCP/IP协议

- [1.1 TCP/IP协议模型](#)
- [1.2 TCP/IP协议报文格式](#)

### 1.1 TCP/IP协议模型

- 为了减少网络协议在设计上的复杂性，大多数的协议采用了层次模型。
- 每一较低层次的模型都向其高一层的协议提供一定的服务，这些服务的具体实现方法对上一层协议而言透明不可见。
- 相邻层协议之间采用原语进行交互，这样的设计提供了各层协议在实现上的独立性。
- 同样层的协议在不同的机器上和操作系统上可能有不同的实现方式，但是只要它
  - 正确实现了与上下协议层的交互界面原语
  - 提供了一致的服务
- 可以确保网络通信的正常进行
- 国际标准化组织（ISO）于20世纪70年代后期指定了开放系统互连参考模型（Open System Interconnection OSI），如下图所示：



- OSI参考模型分为7层：
  - 应用层：提供用户网络分布信息服务的接口，如文件传送、电子邮件服务等
  - 表示层：提供两个应用层协议之间数据表示的语法，如加、解密算法等
  - 会话层：提供应用层实体会话通道的建立和清除以及会话过程的维护等
  - 传输层：提供上面面向应用的高三层和下面面向网络的低三层之间的接口，为会话层提供与具体网络无关的可靠的端对端通信机制。主要有面向连接的服务（字节流）和无连接的服务（数据报）两种类型服务
  - 网络层：建立传输层实体之间的网络（WAN或LAN）连接，包括路由选择等服务
  - 数据链路层：建立于特定网络（LAN）的物理连接上，为网络层提供可靠的传送通道，提供传输错误检测与数据重发
  - 物理层：提供网络端设备接口的物理和电气接口，与物理传输介质直接相连
  - TCP/IP最早起源于1969年美国国防部赞助研究的网络ARPANET—世界上第一个采用分组交换技术的计算机通信网络
- TCP/IP协议模型从更实用的角度出发，形成了具有高效率的四层体系结构，TCP/IP和OSI参考模型对照关系如下图所示：





- TCP/IP协议模型包括四层：

- 主机—网络（网络接口）层：

- TCP/IP模型中的主机—网络（网络接口）层与OSI/ISO的物理层、数据链路层以及网络层的一部分相对应。该层中所使用的协议大多数是各通信子网固有的协议

- 网络互连层（IP层）：

- 是TCP/IP模型的关键部分。功能是使主机把分组发往任何网络，并使各分组独立地传向目的地，这种信息传送的类型称为数据报方式。这些分组到达的顺序可能和发送的顺序不同，因此当需要按顺序发送和接收时，高层必须对分组进行排序。
- 使用IP协议，它把传输层送来的消息组装成IP数据报文，并把IP数据报文传送给主机—网络层
- 网络互连层的主要任务是：
  - 为IP数据报分配一个全网唯一的IP地址，实现IP地址的识别与管理
  - IP数据报的路由机制
  - 发送或接收时使IP数据报的长度与通信子网所允许的数据报长度相匹配

- 传输层：

- 传输层作为应用程序提供端到端通信功能，和OSI/ISO中的传输层相似。该层协议处理网络互连层没有处理的通信问题，保证通信连接的可靠性，能够自动适应网络的各种变化。传输层主要有两个协议——传输控制协议（TCP）和用户数据报协议（UDP）

- 应用层：

- 位于传输层之上的应用层包含所有的高层协议，为用户提供所需要的各种服务，主要包括：
  - 远程登录（Telnet）：用户可以使用异地主机
  - 文件传输（FTP）：用户可以在不同主机之间传输文件
  - 电子邮件（SMTP）：用户可以通过主机和终端相互发送信件
  - Web服务（HTTP）：发布和访问具有超文本格式的HTML的各种信息
  - 域名系统（DNS）：把主机名映射成网络地址

- TCP/IP模型中的应用层与OSI/ISO中的应用层有较大的差别，它不仅包括了会话层及上面的三层的所有功能，而且还包括了应用进程本身在内

## 1.2 TCP/IP协议报文格式

- 每层包含的主要协议类型如下表所示：

TCP/IP协议族及分层结构

应用层	简单邮件传输协议（SMTP）	超文本传输协议（HTTP）	文件传输协议（FTP）	域名服务协议（DNS）	简单网络管理协议（SNMP）
传输层	TCP			UDP	
网络层	IP、ICMP、ARP、RARP				
网络接口层	Ethernet		Token-Ring	100BAST-T	其他

- 网络接口层：

- 实际上，TCP/IP协议并不包括物理层和数据链路层协议，只定义了各种物理协议与TCP/IP之间的接口信息。这些物理网络包括了多种广域网络，如ARPANET、MILNET等

目标主机的以太网地址（第0-3字节）	
目标主机的以太网地址（第4、5字节）	源主机的以太网地址（第0、1字节）

源主机的以太网地址 (第6、7字节)	目的主机的以太网地址 (第8、9字节)
源主机的以太网地址 (第2-5字节)	
帧类型	

- 以太网帧头格式

- ARP和RARP协议：

- 为了使TCP/IP协议与具体的物理网络无关，将物理地址隐藏而统一使用IP地址进行网际通信，就必须提供一种在IP地址和物理地址之间进行映射的机制，对于像以太网这样的具备广播能力的网络，TCP/IP使用
  - 地址解析协议(AddressResolutionProtocol, ARP)，来提供从IP地址到物理地址的映像服务
  - 逆向地址解析协议(ReverseAddressResolutionProtocol, RARP)，来提供从物理地址到IP地址映像服务
- ARP和RARP报文格式：

0	8	16	24	32		
硬件类型		协议类型				
硬件地址长度		操作类型				
源主机硬件地址 (第0~3字节)						
源主机硬件地址 (第4、5字节)		源主机IP地址 (第0、1字节)				
源主机IP地址 (第2、3字节)		目的主机硬件地址 (第0、1字节)				
目的主机硬件地址 (第2~5字节)						
目的主机IP地址						

- IP协议：

- IP协议是TCP/IP协议族的核心协议之一，它提供了无连接数据包传输和网际路由服务
- IP通过互联网传输数据报，它是无连接的，各个IP数据报之间是相互独立的
- 它是不可靠的，不保证投递抵达目的地，对分组的丢失、重复、延迟和顺序错位等情况都不予检测
- IP协议只是做到尽力发送每个数据分组，但是在网络阻塞的情况下，则可能丢弃某些数据报
- IP数据报格式：

0	8	16	24	32		
版本号	报头长度	服务类型	报文总长度			
报文标识		标志	分段偏移量			
生存期		协议号	报头校验和			
源IP地址						
目的IP地址						
选项 + 填充数据						
报文数据						

- 常见网络协议值：

网络协议值	关键字	协议名称
0		保留
1	ICMP	Internet网间控制报文
2	IGMP	Internet网组管理
3	GGP	网关—网关协议
4	IP	IP里的IP (IP in IP)
5	ST	数据流
6	TCP	传输控制协议
8	EGP	外部网关协议
17	UDP	用户数据报协议
29	ISO-TP4	ISO传输协议类4
38	IDPR-CMTP	IDPR控制报文协议
80	ISO-IP	ISO IP协议
89	OSPF	开放式最短路径优先协议
255		保留

- 数据报的分组与重组

- IP协议首先要根据物理网络所允许的最大报文长度对上层协议的数据进行长度检查，必要时将数据报分成若干段后再发送。在数据报分段后，对于每一个段，都要加上IP报头，形成IP数据报。与分段相关的字段包括：
  - 报文标识：数据报的唯一标识
  - 报文总长度：对于每一个数据报的分段，都要重新计算其报文长度
  - 分段偏移量：每一个分段都要设置该字段值，来指明其在原始数据报中的位置，用8B的倍数来表示
  - 标志：如果没有分段，则该标志设定为0；如果数据报进行了分段，则除了最后一个分段将该字段设置为0外，其余各分段都应该设置为1

○ IP数据报的选项：

- 在IP数据报的选项字段中提供了若干选项参数，主要用于控制和测试，如下表所示：

CF	CLASS	NUMBER	长度/B	功能
0	0	0	1	选项表结束（标志数据报头选项字段结束）
0	0	1	1	无操作（作为填充数据，用于在选项字段中排列字节）
1	0	2	11	安全于处理限制（军事应用）
1	0	3	可变	自由源路由
0	0	7	可变	记录路由
1	0	9	可变	限制源路由
0	2	4	可变	Internet时间戳

○ ICMP协议：

- 网际控制报文协议（InternetControlMessageProtocol, ICMP）是用来提供差错报告服务的协议。ICMP是IP协议的一部分，必须包含在每一个IP协议实现中。ICMP报文要通过IP协议发出，具有多种类型，可以提供多种服务
- ICMP的主要优点
  - 为所有控制报文和信息报文提供了一个统一的机制
- ICMP报文格式：



○ ICMP报文类型：

ICMP报文类型值	含义	ICMP报文类型值	含义
0	回送响应	12	数据报参数错误
3	目的不可达	13	时间戳请求
4	报源抑制	14	时间戳响应
5	重定向	17	掩码请求
8	回送请求	18	掩码回应
11	数据报超时		

○ ICMP差错报文：

- ICMP的最基本功能就是提供差错报告传输机制
- ICMP的差错报告都是采用路由器向源主机报告模式，即当路由器发现了IP数据报的错误后，使用ICMP报文向该数据报的源发送主机报告错误情况。同时，发生错误的IP数据报被丢弃，不再转发

○ ICMP的差错报文分为：

- 目的不可达报文
- 超时报文
- 参数出错报文
- ICMP控制报文

○ ICMP请求/应答报文

- 回送请求/响应报文
  - 主要用于测试网络目的结点的可达性
- 时间戳请求/响应报文
  - 主要用于估算源和目的结点间的报文往返时间
- 掩码请求/响应报文
  - 主要用于获取源主机所在网络的IP地址掩码信息

· TCP协议

- TCP的主要功能是在一对高层协议 (UpperLayerProtocol, ULP) 之间在数据报服务的基础上，建立可靠的端对端连接，并提供虚电路服务和面向数据流的传输服务。连接管理分为3个阶段：
  - 建立连接
  - 数据传输
  - 终止连接
- TCP报文格式



- UDP协议
  - 用户数据报协议 (UDP) 提供应用进程之间传送数据报的基本机制
  - UDP主要用于直接使用数据报服务的应用程序，这些应用程序自己提供误码校验以及拥塞控制机制
  - UDP是一种简单的协议机制，通信开销很小，效率比较高，因而比较适合交易型的应用
  - UDP报文格式

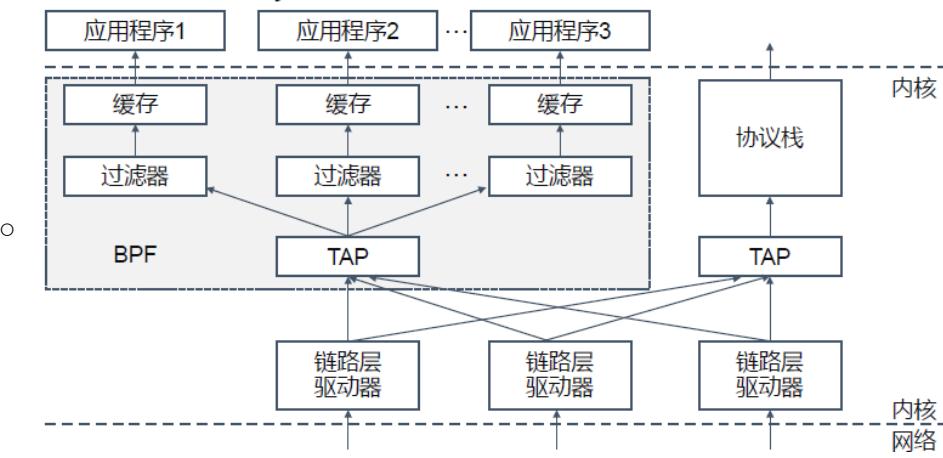


## 2 网络数据包的截获

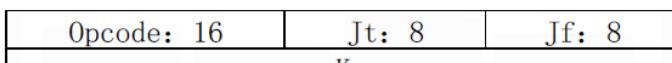
- 网络数据包的截获是基于网络的入侵检测技术的工作基石
- 网络数据包截获方法：
  - 利用以太网络的广播特性
  - 通过设置路由器的监听口或者镜像口来实现

以太网环境下的数据包截获方法：

- 将网卡的工作模式置于混杂 (Promisc) 模式
- 直接访问数据链路层，截获相关数据
- 由应用程序而非上层协议如IP和TCP协议对数据进行过滤处理
- 截获到流经网卡的所有数据
- 不同的操作系统提供的接口功能并不相同，因此直接采用套接字设备的编程代码在不同系统平台上不能通用。这个问题的解决办法之一是使用由美国洛伦兹伯克利国家实验室 (LawrenceBerkeleyNationalLaboratore) 编写的专用于数据报截获功能的API函数库 “Libpcap”
- BPF(BerkeleyPacketFilter)的工作原理



- BPF的具体实现为一个虚拟机。该虚拟机包括一个累加器、一个变址寄存器 (x)、一个临时缓存和一个隐含的程序计数器 (pc)
- 指令格式

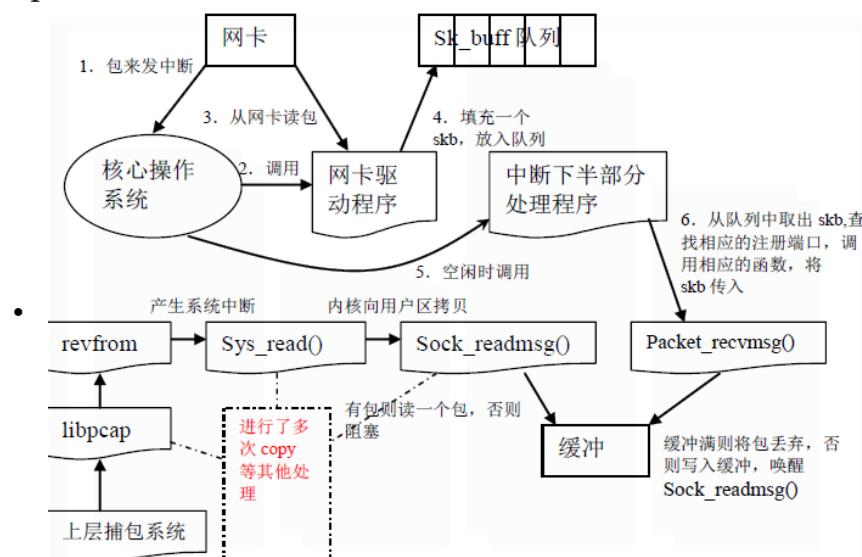


- Opcode(操作码): 指示指令类型和寻址模式
- Jt (真跳转)、Jf (假跳转) : 用于条件跳转指令, 代表真、假情况下的指令跳转偏移量
- K: 通用字段, 可用于各种用途
- BPF数据包过滤机制功能非常强大, 性能也很高
  - 能够尽早丢弃不需要的数据包
  - 避免必须将所有的数据包从内核中复制到用户空间再进行处理这种重复的复制(无效工作)
  - 节约了大量的CPU时间
  - 提高了数据包的截获能力
- Winpcap架构包含3个模块:
  - NPF (NetgroupPacketFilter) : 一个虚拟设备驱动程序文件
    - 过滤数据包, 并把数据包传递给用户态模块, 这个过程中包括了一些操作系统所特有的代码
  - Packetdll: 该动态链接库为WIN32平台提供了一个公共的接口
    - 用于解决这些平台的差异性
  - Wpcapdll: 该动态链接库提供了一个不依赖于操作系统类型的高层接口库
    - 提供了更高层次、抽象的函数
  - Packetdll直接映射了内核过滤模块的调用, 而Wpcapdll提供了更加友好、功能更加强大的函数调用

交换网络环境下的数据包截获方法:

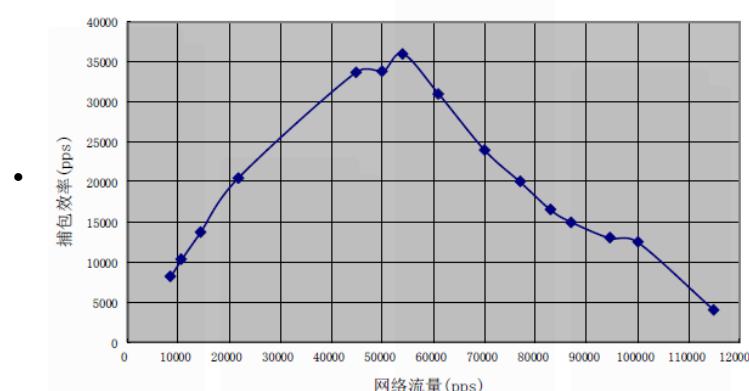
- 常用的方法是利用交换机或者路由器上设置监听口或者镜像口, 此时所有的网络信息数据包除按照正常情况转发外, 将同时转发到镜像端口, 从而达到截获所有网络流量的目的。在实际工作中存在两个问题:
- 随着交换带宽的不断增长, 并非所有的网络流量都会反映在镜像口上
- 并非所有的交换设备都提供类似的镜像口
- 很多IDS系统会选择挂接在流量通常最大的上下行端口上, 用来截获进出内外网的数据流量

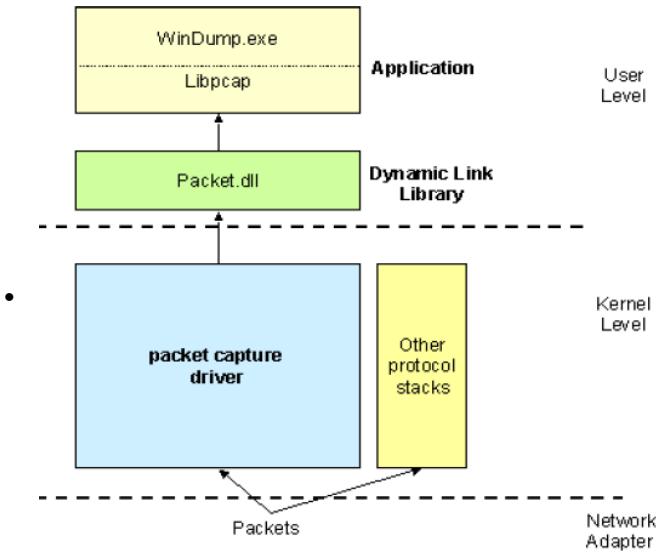
### Libpcap与Zero-copy



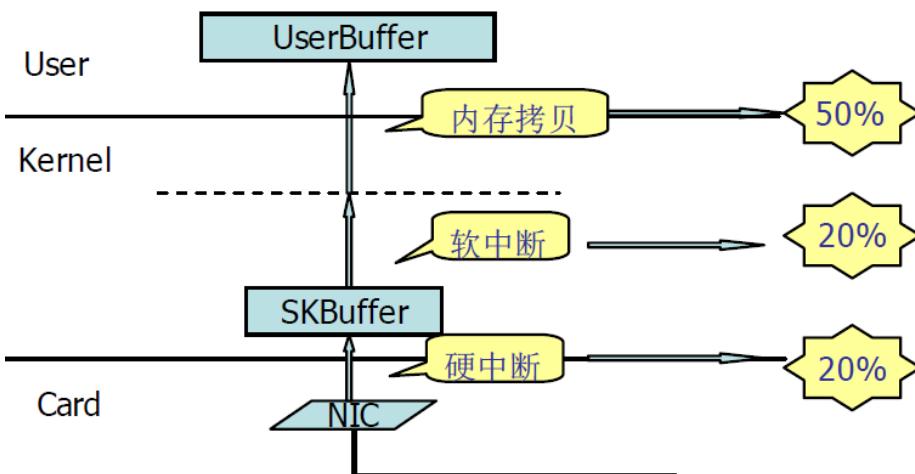
Libpcap报文捕获原理图

libpcap捕包速率与流量关系图

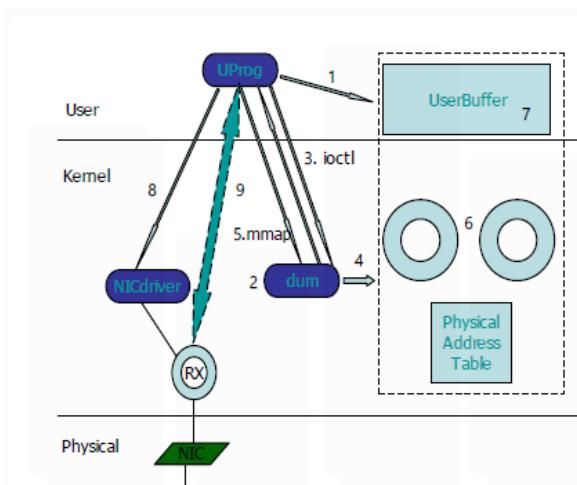




Winpcap报文捕获原理图



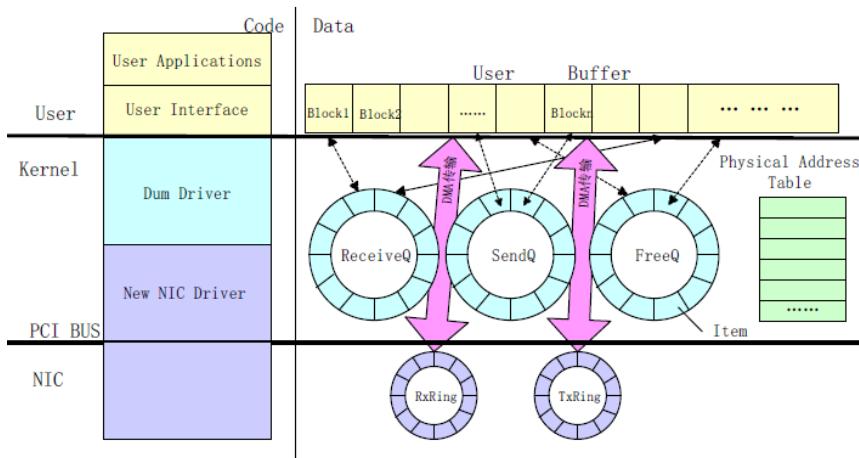
通常情况下内核接收网络数据包的流程图



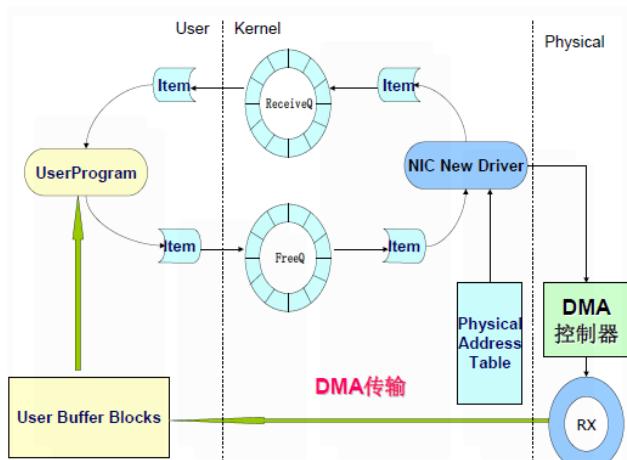
基于零拷贝技术捕包平台流程示意图

- 用户程序分配一块大的由用户和内核共享的内存UserBuff。
- 创建设备/dev/dum。
- 用户程序进行ioctl系统调用，向dum设备发出DUM\_IOCREGBUF命令，并将用户缓冲区UserBuff地址和大小等信息传给dum。
- Dum设备驱动程序执行DUM\_IOCREGBUF命令，在内核分配两块空间：Directory和PhyAddressTable，并初步初始化。然后返回Directory的大小给用户程序。其中Directory包含freeQueue和busyQueue两个环形队列，PhyAddressTable存储UserBuffer中每一页的物理地址。
- 根据Directory的大小系统调用mmap将内核的Directory空间映射到用户内存空间，使用户程序可以直接对Directory进行操作。

- f. 用户程序初始化Directory中两个环形队列，使队列中每一项对UserBuffer中每个2k块的块号。
- g. 用户程序将UserBuffer中的2k小块初始化成包的格式。
- h. 用户程序先后通过两次ioctl调用，分别通知dum一切缓冲区均已就绪，然后将dum的信息传给网卡，通知它用UserDMA的方式开始工作。
- i. 网卡将收到的包直接送入UserBuffer的每一块中，用户程序可以通过接口函数获取数据并进行相应处理。

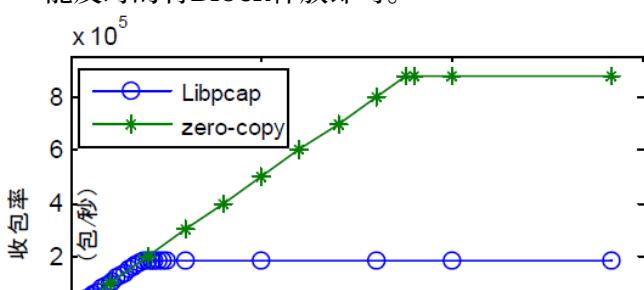


零拷贝平台代码和数据的层次示意图



零拷贝平台收取报文的过程

- 为了说明网卡、Dum、上层应用协作的过程，便于说明问题，假设中断的触发方式是每包中断。
  - a. 数据帧到达网卡RxRing后，发出一次硬件中断
  - b. 收包中断处理程序首先从Dum设备的空闲队列中取出第一个可用的块描述符Item1，根据Item1中的块号查找物理地址表，找到当前可用的空闲块Block1的物理地址，这个地址就是网卡DMA的目的地址
  - c. 然后将这个物理地址写入网卡RxRing上的一个描述符，并将数据包的大小也写入这个描述符。DMA控制器就根据这个描述符上的信息进行一次DMA传输，将RxRing刚收到的数据包传入用户的Block1中
  - d. 接着中断处理程序将刚刚充满数据的Block1对应的Item1挂入ReceiveQ队列
  - e. 上层应用程序一直在不断查询ReceiveQ的状态，一旦发觉已经有新的Item，便将其从ReceiveQ上摘下，根据Item中的块号找到真正存放数据的Block的地址，然后就可以对这块缓冲区中的数据包进行处理了。处理方式由上层应用决定，可以是得到数据包便进行处理，处理完再将这个Block对应的Item挂入空闲队列，表示这个块可以被再使用了；也可以先把这个数据包拷贝到内存的另一个地方，然后就释放Item到空闲队列。不管什么方式，只要上层应用最终能及时的将Block释放即可。



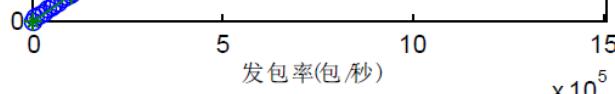


图4 Libpcap与zero-copy捕包性能比较示意图

在相同的实验环境下并且每个包大小为64B时

## 专用网卡

- 曙光NetFirm智能网卡：
- 可编程的专用芯片，既可以实现高性能网络数据处理，也可以进行业务扩展。
- 功能：
  - 线速收发包
  - 报文分类
  - 规则处理
  - 端口过滤、流还原

## 3 检测引擎的设计

- 检测引擎的设计是基于网络入侵检测的核心问题
- 检测引擎分为两大类：
  - 嵌入式规则检测引擎
  - 可编程的检测引擎
- 两种引擎比较
  - 嵌入式规则检测引擎的代码实现对于终端用户隐藏不可见，用户可配置检测规则，但是无法了解嵌入在系统内部的实现机制
  - 可编程检测引擎允许用户采用特定的编程语言或脚本语言，来实现具体的检测模块，然后交由检测引擎处理
  - 可编程的检测引擎设计具备强大的灵活性，需要用户具备更多的专业知识
  - 嵌入式规则检测引擎的设计则更容易使用，用户可以在既定的规则集合中选取特定子集，或者根据条件创建自己的检测规则
- [3.1 嵌入式规则检测引擎设计★](#)
- [3.2 可编程的检测引擎设计](#)
- [3.3 特征分析与协议分析技术★](#)

### 3.1 嵌入式规则检测引擎设计★

- 嵌入式规则检测引擎的设计中存在两个关键问题：
  - 检测规则
  - 引擎架构设计

以Snort为例

- Snort规则可以划分为两个逻辑部分
  - 规则头：包含了规则动作、协议、IP源地址和目的地址、子网掩码以及源端口和目标端口等信息
  - 规则选项：包含警报信息以及用于确定是否触发规则响应动作而需检查的数据包区域位置的相关信息
- Snort规则头的主要组成字段
  - 规则动作
    - 定义了当前数据包满足所有在规则中指定的属性特征的情况下，所应采取的动作，它位于规则的首位，包含三种基本动作类型和新增的两种类型：
      - Alert：使用选定的报警方式生成警报信号，然后记录当前数据包
      - Log：记录当前数据包
      - Pass：丢弃（忽略）当前数据包
    - Activate：激活指定的特定Dynamic类型的检测规则
    - Dynamic：在特定Activate类型规则触发后，得到激活
  - 协议：主要支持TCP、UDP和ICMP协议
  - IP地址
  - 端口号
  - 方向操作符
- Snort规则的基本选项类型
  - Msg：指定规则触发时需要记录的警报消息文本

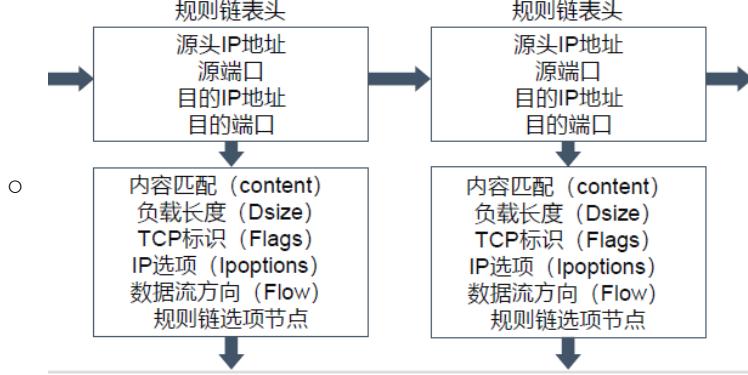
- Logto: 用来通知系统将触发当前规则的所有数据包记录到一个指定的输出日志文件中
- Ttl: 设定用于测试目的的特定TTL值, 主要用于对Traceroute探视活动的测试
- Id: 用于精确匹配IP数据包头中的分组ID字段值
- Dsize: 用于测试数据包负载的大小
- Flow: 指定当前网络数据流的方向及状态信息
- Content: 允许用户在数据包载荷中搜索特定模式的内容信息, 并触发响应动作
- Content-list: 用来替换对Content关键字的多次使用行为
- Uricontent: 与content类似, 但不同之处在于uricontent中指定的特征模式搜索工作仅限定于HTTP请求报文中的URL字段内容
- Offset: 是content选项关键字的修饰符, 用于设定从数据包负载起始位置处计算的模式匹配的开始搜索位置
- Depth: Content规则选项的修饰符, 设定从搜索起始位置开始计算所的模式匹配的最大搜索深度
- Nocase: 用于使得“content”选项中的大小写敏感性失效
- Flags: 测试各种TCP标识符在当前数据包里是否进行了设置
- Seq: 对TCP序列号字段的测试, 检查当前数据包的TCP序列号是否等于指定值
- Ack: 测试TCP包头的确认字段是否等于指定值
- Itype: 测试ICMP数据包的类型字段
- Icode: 用来测试ICMP数据包中的代码值
- Session: 从TCP对话中提取用户数据
- Icmp\_id: 用于测试一个ICMP ECHO数据包的ID字段是否等于指定值
- Icmp\_seq: 检测ICMP的序列号字段
- Ioption: 搜索某一特定的选项是否存在
- Rpc: 查看Rpc请求, 并自动解析出该请求中涉及的程序、过程和程序版本号信息
- Resp: 实现对满足特定规则的网络流量的响应动作, 能够主动关闭可能的入侵
- Snort的系统结构分为三部分:
  - 协议解析器
    - 协议解析引擎的全部工作都是围绕着网络协议栈中的各层协议展开的, 包括了从数据链路层协议向上到TCP协议层的定义
    - 协议解析器中涉及的主要数据结构为Packet
    - Packet数据结构主要分为三类:
      - 指示原始数据包截获信息的字段
      - 用于存放当前数据包进行协议解析后所得信息的字段
      - 各种标识字段
  - 规则检测引擎
    - 三个模块:
      - [规则链表构造模块](#)
      - [预处理模块](#)
      - [规则匹配模块](#)
  - 日志/警报系统
    - 规则匹配模块所触发的规则动作, 具体功能实现由日志/警报系统完成。
    - 日志/警报系统的各种响应功能, 也是通过各种插件模块来实现
    - 三种日志模式:
      - 关闭: 获取更好的运行性能
      - 以可读格式记录数据包: 有利于快速分析
      - 以Tcpdump二进制形式记录数据包: 提高系统运行性能

## 规则链表构造模块

- 主要功能是读入配置文件, 逐条解析检测规则, 并最终形成内存中的二维规则链表结构。该链表结构分为两部分:
  - 链表头: 包含多个规则中的共有属性
  - 链表选项: 不同的检测属性
- 同一个规则链表头的下面可以链接多个规则链表选项结点
- Snort2.0中引入了快速规则匹配模块

- 通过规则中的目的端口和源端口值来将整体的大规则集合有效划分为多个子规则集合
- 在原有的二维规则链表结构的基础上，再构建一层新的数据结构，以便用于快速规则匹配

- 规则链表逻辑结构



### 预处理模块

- 主要作用就是对当前截获的数据包进行预处理操作，以方便后续规则匹配模块对数据包的处理操作
- 预处理模块提供的有利于规则匹配的重要功能
  - 数据包分片重组及数据流重组功能
  - 协议规范化/解码功能

### 规则匹配模块

- 完成规则链表的构造工作和必要的预处理操作后，规则匹配模块的工作就是对于每一个当前数据包，遍历整个规则链表结构，并调用对应的插件模块进行各种处理。一旦规则匹配模块搜索到一个与解析后的数据包匹配的规则，则触发定义好的规则动作并返回

## 3.2 可编程的检测引擎设计

- 允许用户采用特定的编程语言或脚本语言，来实现具体的检测模块
- 可编程入侵检测引擎设计的要点就在于用于实现入侵检测功能的脚本语言的定义及其与引擎架构的接口设计

## 3.3 特征分析与协议分析技术★

- 特征分析技术
  - 最早应用在早期的网络入侵检测系统中
  - 工作原理
    - 建立在字符串匹配的简单概念上。在最初的特征分析技术工作过程中，每次输入检测引擎的数据包，将会与单个特征进行逐个字符的匹配操作
  - 最初的基于特征分析的入侵检测系统完全不考虑网络数据包中所含的协议格式化信息。而是将输入数据包视为一个无序无结构的随机数据流，企图仅仅依靠简单的字符串匹配操作完成所有的检测任务
  - 优点
    - 在小规则集合情况下工作速度快
    - 检测规则易于编写、理解并且容易进行定制
    - 对新出现的攻击手段具备快速升级支持能力
    - 对底层的简单脚本攻击，具备良好的检测性能
    - 对所发生的攻击类型，具备确定性的解释能力
  - 缺点
    - 随着规则集合规模的扩大，检测速度迅速下降
    - 各种变种的攻击行为，易于造成过度膨胀的规则集合
    - 较易产生虚警信息
    - 仅能检测到已知的攻击类型，前提是该种攻击类型的检测特征已知
- 协议分析技术
  - 将输入数据包视为具有严格定义格式的数据流，并将输入数据包按照各层协议报文封装的反向顺序，层层解析出来。然后，再根据各层网络协议的定义对各层协议的解析结果进行逐次分析
  - 协议分析技术利用预先定义好的关于协议字段的期望值或合理值的详细信息，来判断是否出现了恶意的网络流量。与特征分析技术依赖于已知的攻击特征来

检测非法活动不同

○ 优点

- 具备良好的性能可扩展性，特别是在规则集合规模较大的情况下
- 能够发现最新的未知安全漏洞
- 较少出现虚警信息

○ 缺点

- 在小规模集合情况下，初始的检测速度相对较慢
- 检测规则比较复杂，难以编写和理解并且通常是由特定厂商实现
- 协议复杂性的扩展以及实际实现的多样性，容易导致规则扩展的困难
- 对发现的攻击行为类型，缺乏明确的解释信息