



中国科学院大学

University of Chinese Academy of Sciences

《软件与系统安全》

实验 1A & 1B 讲解

助教：刘鹏

中国科学院大学 · UCAS

Contents

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

● 实验内容介绍

- 1A: 嗅探器设计与实现
- 1B: 安全文件传输软件设计与实现

● 实验实施步骤

● 作业提交方式

- 不要压缩为RAR、7Z 等格式，压缩包不要加密。
- 命名方式见下页PPT
- 所有代码必须开源，GitHub/Gitee/GitLab 有完整记录

● 实验考核及评分准则

- 共有 59 人选课，最终将根据完成情况按排名打分。

Commit Guides

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

1A: 嗅探器设计与实现

文件夹	张三_202018008829001_EX1A
1. 源码文件夹	张三_202018008829001_ex1A_src
2. 演示视频	张三_202018008829001_ex1A_play.mp4
3. 文档	张三_202118008829001_ex1A_report.pdf/docx

1B: 安全文件传输软件设计与实现

文件夹	张三_202018008829001_EX1B
1. 源码文件夹	张三_202018008829001_ex1B_src
2. 演示视频	张三_202018008829001_ex1B_play.mp4
3. 文档	张三_202118008829001_ex1B_report.pdf/docx

无需 README, 想说的全放在文档里即可。Git 链接必须写进去。

实验 1A: Introduction

Contents

Exp I

ISTs I

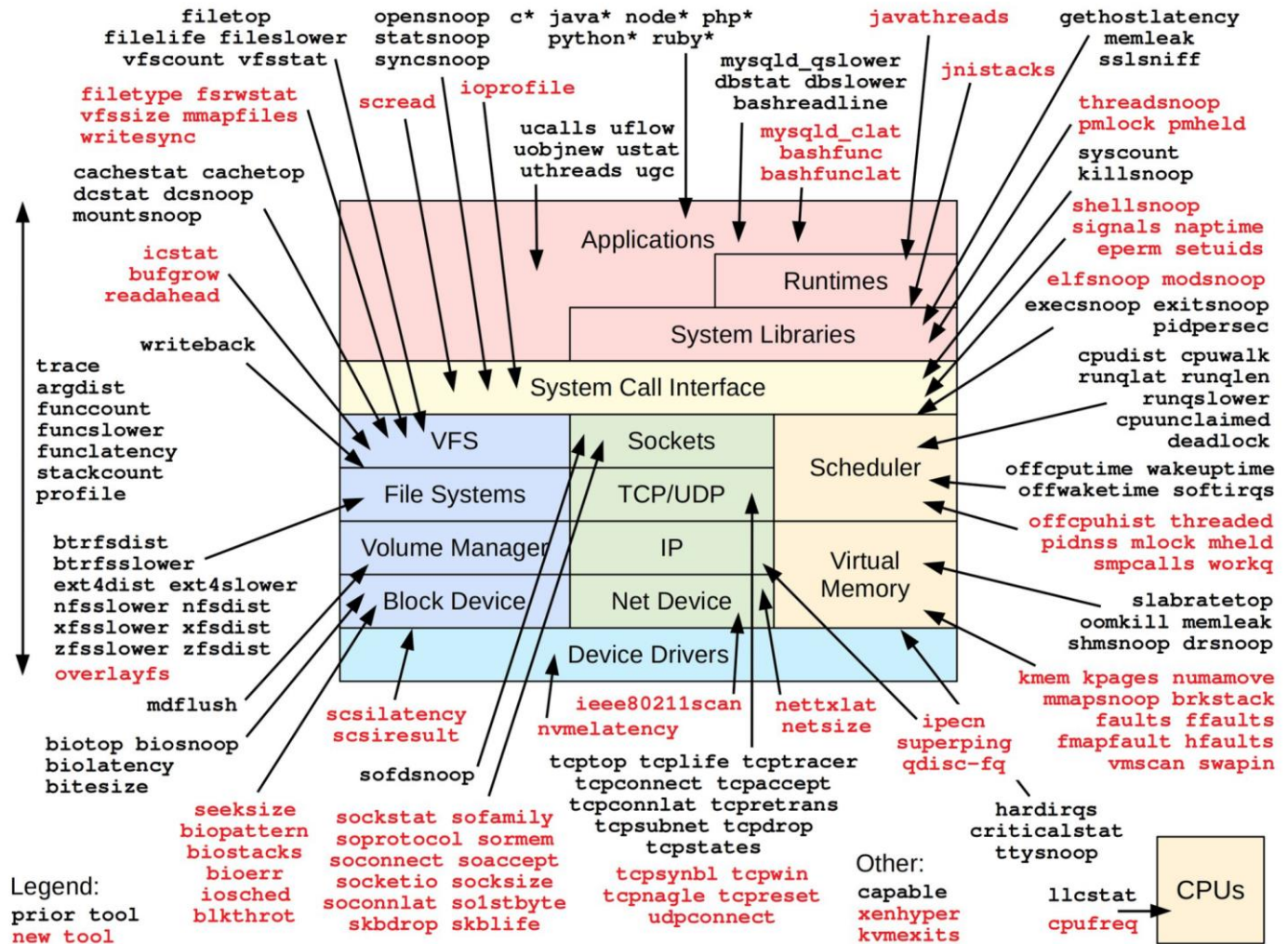
Prompt 1

Exp II

ISTs II

Prompt II

Q&A



实验 1A: Introduction

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

```
0[ 0.0%] 8[ 0.0%] 16[ 0.0%] 24[ 0.0%]
1[ 0.0%] 9[ 0.0%] 17[ 0.0%] 25[ 0.6%]
2[ 2.0%] 10[ 0.0%] 18[ 0.6%] 26[ 0.7%]
3[ 0.0%] 11[ 0.0%] 19[ 0.0%] 27[ 0.0%]
4[ 0.0%] 12[ 0.0%] 20[ 0.0%] 28[ 0.0%]
5[ 0.0%] 13[ 0.0%] 21[ 0.0%] 29[ 0.0%]
6[ 0.0%] 14[ 0.0%] 22[ 0.0%] 30[ 0.0%]
7[ 0.0%] 15[ 0.0%] 23[ 0.0%] 31[ 2.0%]

Mem[|||||] 2.28G/23.3G Tasks: 97, 380 thr; 1 running
Swp[|] 1.01M/7.89G Load average: 0.00 0.00 0.00
Uptime: 19:37:33

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
 2657 gdm        20    0 3163M  159M 88000  S   2.0   0.7   0:52.00 /usr/bin/gnome-shell
975007 newton     20    0 29804  5352  3664  R   2.0   0.0   0:00.19 htop
 2012 root        20    0  276M  11744  9352  S   0.7   0.0   0:06.28 /usr/sbin/httpd -DFOREGROUND
 2132 apache     20    0 2730M 20572  7048  S   0.7   0.1   0:28.90 /usr/sbin/httpd -DFOREGROUND
 2887 mysql      20    0 2056M  399M 35752  S   0.7   1.7   0:35.57 /usr/libexec/mysqld --basedir=/usr
 3028 gdm        20    0 3163M  159M 88000  S   0.7   0.7   0:00.80 /usr/bin/gnome-shell
    1 root        20    0  235M 14244  9028  S   0.0   0.1   0:29.23 /usr/lib/systemd/systemd --system --deserialize 83
 1173 root        20    0  247M  133M  132M  S   0.0   0.6   0:57.78 /usr/lib/systemd/systemd-journald
 1485 rpc      20    0 67200  5288  4560  S   0.0   0.0   0:00.15 /usr/bin/rpcbind -w -f
 1487 root      16   -4  147M  2928  2148  S   0.0   0.0   0:36.51 /sbin/auditd
 1488 root      16   -4  147M  2928  2148  S   0.0   0.0   0:00.77 /sbin/auditd
 1489 root      16   -4 48560  3296  2916  S   0.0   0.0   0:11.65 /usr/sbin/sedispatch
 1490 root      16   -4  147M  2928  2148  S   0.0   0.0   0:08.19 /sbin/auditd
 1529 rtkit      21    1  198M  3680  3328  S   0.0   0.0   0:01.50 /usr/libexec/rtkit-daemon
 1530 root      20    0  213M 14056 11888  S   0.0   0.1   0:01.33 /usr/sbin/sss -i --logger=files
 1531 root      20    0  452M 12704 10880  S   0.0   0.1   0:00.09 /usr/sbin/ModemManager
 1532 libstorag 20    0 19740  1948  1792  S   0.0   0.0   0:00.30 /usr/bin/lsm -d
 1533 root      20    0 50264  5116  4052  S   0.0   0.0   0:00.08 /usr/sbin/smartd -n -q never
 1534 root      20    0  122M  5520  4740  S   0.0   0.0   0:15.90 /usr/sbin/irqbalance --foreground
 1535 dbus      20    0 92808  7848  5996  S   0.0   0.0   1:39.22 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile
 1536 root      20    0  542M 15304 12808  S   0.0   0.1   0:00.37 /usr/libexec/udisks2/udisksd
 1537 root      20    0 79252  7116  6232  S   0.0   0.0   0:02.38 /usr/lib/systemd/systemd-machined
 1538 avahi     20    0 85332  5184  4660  S   0.0   0.0   0:13.90 avahi-daemon: running [emc-redhat.local]
 1540 polkitd   20    0 1998M 33216 19572  S   0.0   0.1   0:42.23 /usr/lib/polkit-1/polkitd --no-debug
 1542 root      20    0 86204 11204  9524  S   0.0   0.0   0:00.02 /usr/bin/VGAuthService -s
 1543 root      20    0  285M 12188 10072  S   0.0   0.0   1:54.80 /usr/bin/vmtoolsd
 1544 root      20    0 18308  2108  1936  S   0.0   0.0   0:00.01 /usr/sbin/mcelog --ignoreudev --daemon --foreground
 1547 root      20    0  122M  5520  4740  S   0.0   0.0   0:00.00 /usr/sbin/irqbalance --foreground
 1548 root      20    0  542M 15304 12808  S   0.0   0.1   0:00.00 /usr/libexec/udisks2/udisksd
 1552 root      20    0  452M 12704 10880  S   0.0   0.1   0:00.00 /usr/sbin/ModemManager

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice +F9Kill F10Quit
```

Linux htop 命令输出

实验 1A：嗅探器设计与实现

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 网络嗅探器

- 介绍：网络嗅探器，Network Packet Sniffer (NPS)，是一种专门用来进行网络流量侦听的工具。数据包嗅探器是研究网络行为学的基础工具。
- 本次实验中的嗅探器与常见的 Wireshark 等工具不同
 - Wireshark 等工具实现了强大的数据包协议分析功能，属于 Network Protocol Analyzer (NPA).
 - 请各位思考，如何识别某种特定类型的应用层报文。

- 温馨提示

- 实验中必须实现 NPS 功能（否则不及格）
- 实验中尽量完善 NPA 功能（否则分不高）
- 如果有同学可以基于 Wireshark 做良好的二次开发，也会有加分。

Demo: Wireshark

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

Standard 3-pane Packet Browser

Packet List

No.	Time	Source	Destination	Protocol	Length	Info
58	0.741594	52.98.42.226	10.0.0.3	TCP	60	993 → 14171 [ACK] Seq=787 Ac
59	0.761188	n169-1.mail.139.com	10.0.0.3	TLSv1.2	570	Certificate, Server Hello Don
60	0.762015	10.0.0.3	n169-1.mail.139.com	TLSv1.2	372	Client Key Exchange, Change C
61	0.771623	120.241.25.80	10.0.0.3	TLSv1.2	102	Application Data
62	0.773022	10.0.0.3	10.0.0.3	TLSv1.2	93	Application Data
63	0.810902	fe80::20c:29ff:fec6:8dde	10.0.0.3	ICMPv6	86	Neighbor Solicitation for 240
64	0.816277	n169-1.mail.139.com	10.0.0.3	TLSv1.2	280	New Session Ticket, Change Ci
65	0.818329	120.241.25.80	10.0.0.3	TCP	60	993 → 14006 [ACK] Seq=486 Ac
66	0.821549	120.241.25.80	10.0.0.3	TLSv1.2	106	Application Data
67	0.821765	10.0.0.3	120.241.25.80	TLSv1.2	127	Application Data
68	0.864797	10.0.0.3	n169-1.mail.139.com	TCP	54	14220 → 993 [ACK] Seq=490 Ac
69	0.888710	17.56.9.14	10.0.0.3	TLSv1.2	119	Application Data

Packet Details

- > Frame 68: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{295CF37B-0794-4165-9C0B-03B9D06D6F52}, id 0
- > Ethernet II, Src: Giga-Byt_ad:01:74 (b4:2e:99:ad:01:74), Dst: OpenWRT.lan (08:0c:29:c6:8d:de)
- > Internet Protocol Version 4, Src: 10.0.0.3 (10.0.0.3), Dst: n169-1.mail.139.com (120.232.169.1)
- > Transmission Control Protocol, Src Port: 14220, Dst Port: 993, Seq: 490, Ack: 4839, Len: 0

Packet in Binary

```
0000 00 0c 29 c6 8d de b4 2e 99 ad 01 74 08 00 45 00  ..)...
0010 00 28 b1 24 40 00 40 06 00 00 0a 00 00 03 78 e8  -(.$@.0.
0020 a9 01 37 8c 03 e1 13 ec 47 c1 86 20 55 89 50 10  ..7.....
0030 03 ff 2c 07 00 00                                ...,
```

Demo: CommView

Contents

Exp I

ISTs I

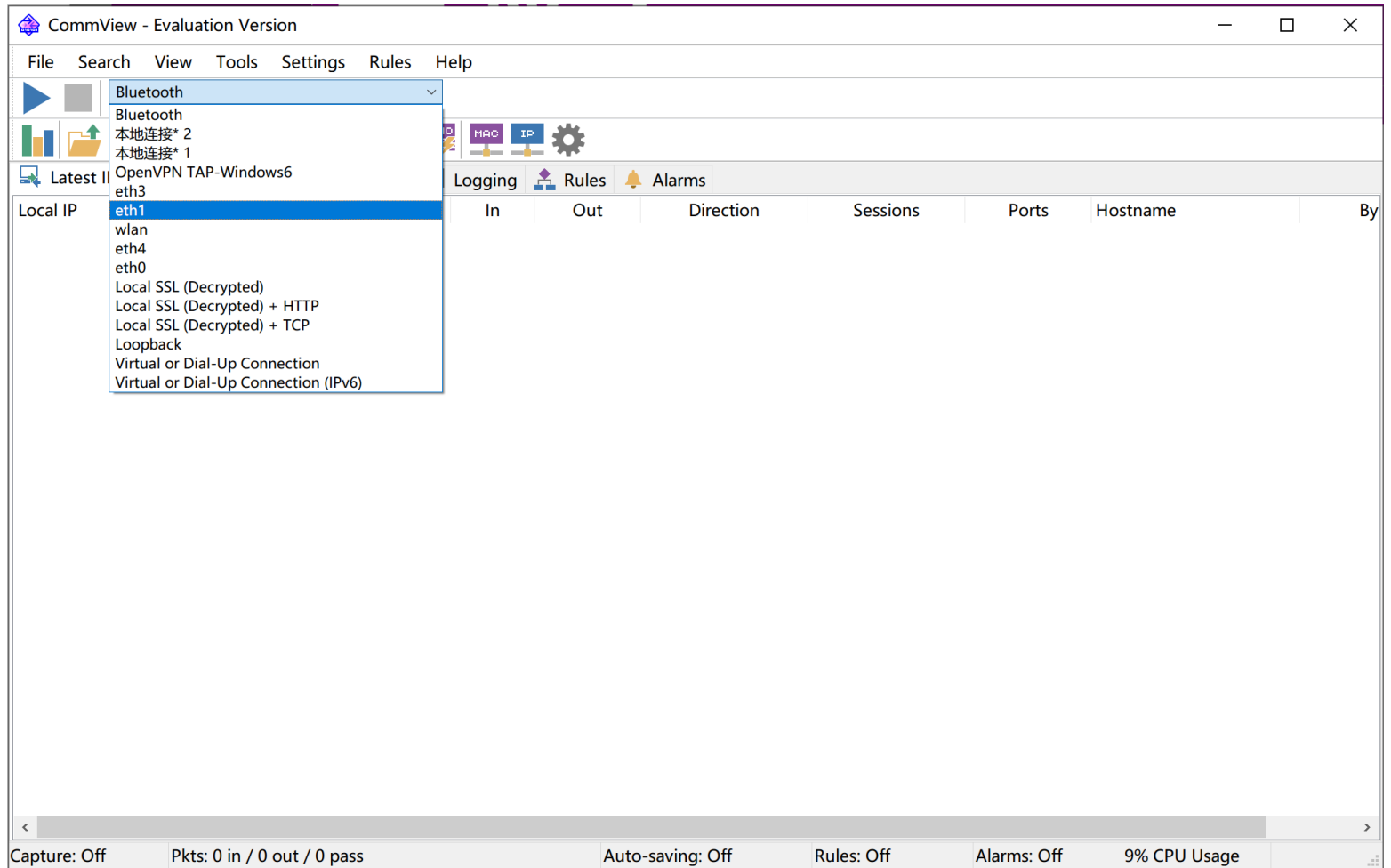
Prompt I

Exp II

ISTs II

Prompt II

Q&A



Demo: CommView

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

CommView - Evaluation Version

File Search View Tools Settings Rules Help

eth0

Latest IP Connections Packets VoIP Logging Rules Alarms

Local IP	Remote IP	In	Out	Direction	Sessions	Ports	Hostname	Bytes	Process
10.0.0.3	192.168.137.1	151	90	In	0	13040,netbios-ns,3285	NEWTON-PC-5	32,859	System
10.0.0.88	239.255.255.250	0	156	Pass	0	36959,ssdp		56,030	
fe80::0488:ea4a:3d20:...	ff02::00fb	0	47	Pass	0	5353		15,897	
10.0.0.124	224.0.0.251	0	47	Pass	0	5353		14,957	
10.0.0.3	223.5.5.5	9	9	Out	0	domain	public1.alidns.com	2,027	System
fe80::dc1d:a108:76fa:c...	ff02::0001:0003	0	8	Out	0	llmnr		1,030	System
10.0.0.3	224.0.0.252	0	8	Out	0	llmnr		870	System
10.0.0.3	140.143.51.110	6	4	In	0	7446		3,066	System
10.0.0.3	58.215.175.52	70	119	Out	1	https,netbios-ns		171,490	System
10.0.0.88	255.255.255.255	0	6	Pass	0	9999		3,324	
2400:dd01:103a:4008:...	ff02::0001:ffc2:9191	0	6	Out	0			516	
10.0.0.3	17.56.9.14	2	2	In	0	14674,14005		228	System
10.0.0.3	172.217.161.174	2	2	In	0	14608,14609		228	System
fe80::020c:29ff:fec6:8d...	ff02::0001:ff02:293b	0	2	Pass	0			172	
10.0.0.3	52.98.40.66	1	1	In	0	14111		114	System
10.0.0.3	52.109.124.129	1	1	In	0	14638		114	System
fe80::020c:29ff:fec6:8d...	ff02::0001:ff99:735f	0	1	Pass	0			86	
10.0.0.3	52.139.250.253	1	1	In	0	14681		114	System
10.0.0.3	120.232.169.1	10	10	Out	1	imaps	n169-1.mail.139.com	6,986	System
10.0.0.3	106.75.93.163	1	1	Out	0	https		143	System

Capture: Off Pkts: 758 in / 832 out / 385 pass Auto-saving: Off Rules: Off Alarms: Off 3% CPU Usage

实验 1A 要求

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 务必 OSI 五层模型下的全部抓包

- 深入理解抓包层次与分析层次

- iOS 的 Thor App 可以在安装了系统描述文件之后，抓取 HTTP/HTTPS 的应用层包。同样因为安装了根证书性质的描述文件，故 Thor 能解析 HTTPS 中的内容。但具体到某些传输控制协议的内容，Thor 无能为力。这告诉了我们什么道理？

- 要有一定的协议过滤能力

- Wireshark 等软件的协议过滤器支持逻辑演算，所以该软件里含有逻辑推导的组件。基本的协议过滤需要支持筛选 HTTP、TCP/UDP、IPv4/v6、ICMP 等不同类型、层次的数据包。libpcap 提供了数据包筛选功能。

- 有一定的流追踪能力（加分项）

- 基于 IP+Port 的 TCP 流
 - 某进程产生的所有 TCP 流

实验 1A 提示

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- **基于图形化编程的重要性**

- 现代化的软件基本都提供图形化交互界面。
- 图形化的界面对于分析流的时序图具有重要帮助作用。
- 在现代图形 API 的帮助下开发图形化程序并不难。
- 有关现代图形库的帮助，参见 Prompt II.

- **抓包的系统 API**

- libpcap
- winPcap: 基于 Windows NT 内核定制的 libpcap
- 请大家自行学习上述 lib/dll 的用法，然后在代码中实现调用。

Linux 协议栈

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

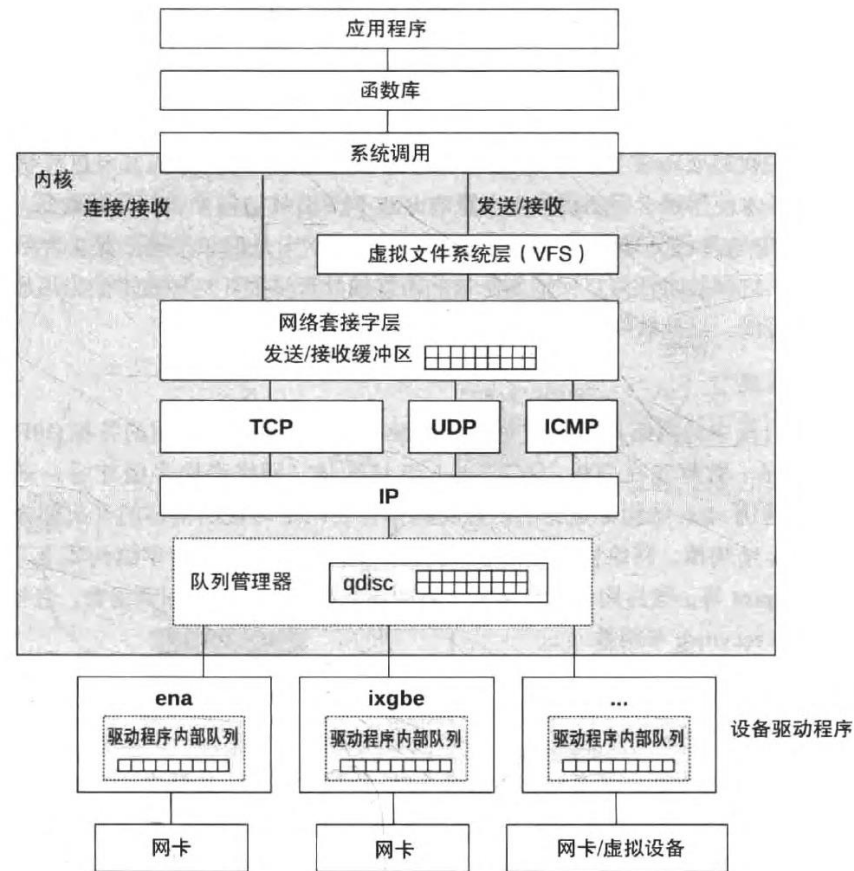


图 1. Linux 网络协议栈

Linux BPF Internals

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 网络传输速度不断提升，以太网已经实现 100Gbps 甚至是 400Gbps 的速率，如何让抓包工具能匹配网速呢？性能是个关键问题。
- BPF/eBPF (2022) 已经是一个技术名称综合，与 LLVM 不是底层虚拟机一样，不可“望文生义”。利用 BPF 可以在内核态编程，并且有以下主要优势：
 - User-defined programs 用户可编程
 - Limited and secure kernel access 内核依旧安全
 - A new type of software 与 VM、Docker 一样已经为一种新技术

<https://www.usenix.org/conference/lisa21/presentation/gregg-bpf>

利用 Linux BPF 开发监控软件

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

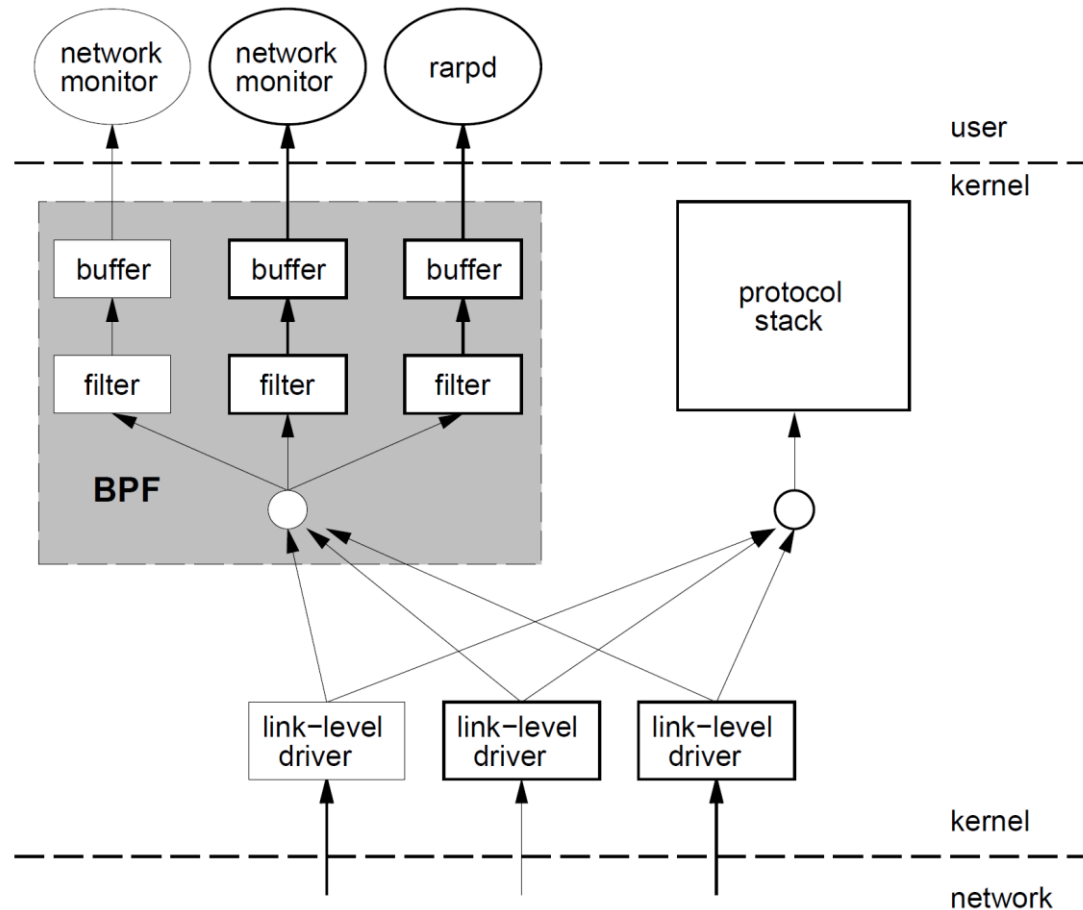


图 2. BPF 概览

BPF 与 传统编程对比

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

	Execution model	User defined	Compilation	Security	Failure mode	Resource access
User	task	yes	any	user based	abort	syscall, fault
Kernel	task	no	static	none	panic	direct
BPF	event	yes	JIT, CO-RE	verified, JIT	error message	restricted helpers

图 3. 用户态、内核态与 BPF 编程对比

BPF kernel Verifier

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

check_subprogs	check_helper_mem_access
check_reg_arg	check_func_arg
check_stack_write	check_map_func_compatibility
check_stack_read	check_func_proto
check_stack_access	check_func_call
check_map_access_type	check_reference_leak
check_mem_region_access	check_helper_call
check_map_access	check_alu_op
check_packet_access	check_cond_jump_op
check_ctx_access	check_ld_imm
check_flow_keys_access	check_ld_abs
check_sock_access	check_return_code
check_pkt_ptr_alignment	check_cfg
check_generic_ptr_alignment	check_btf_func
check_ptr_alignment	check_btf_line
check_max_stack_depth	check_btf_info
check_tp_buffer_access	check_map_prealloc
check_ptr_to_btf_access	check_map_prog_compatibility
check_mem_access	check_struct_ops_btf_id
check_xadd	check_attach_modify_return
check_stack_boundary	check_attach_btf_id

Reference 1A

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- https://www.youtube.com/watch?v=5Z2AU7QTH4&ab_channel=USENIX
- libpcap官方仓库: <https://github.com/the-tcpdump-group/libpcap>

实验 1B：安全文件传输软件

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- Transfer File over Network with Security
 - 安全地通过网络传输文件，简称 FTPS
 - 不要求写一个带身份验证的类 SMB 协议
 - 不要求利用内核 RDMA 等高级功能
- 温馨提示
 - 建议使用 Visual Studio 2019 实现图形化部分
 - Windows 的 WinForm、WPF 图形库
 - macOS 的 Cocoa 图形库
 - Linux 的各种图形库 (Wxwidgets, xxQT)
 - 由于用来判卷的电脑是 Windows 系统，因此用其他 OS 的同学，需要提供编译成功、运行成功的视频。

现代 TCP 报头

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

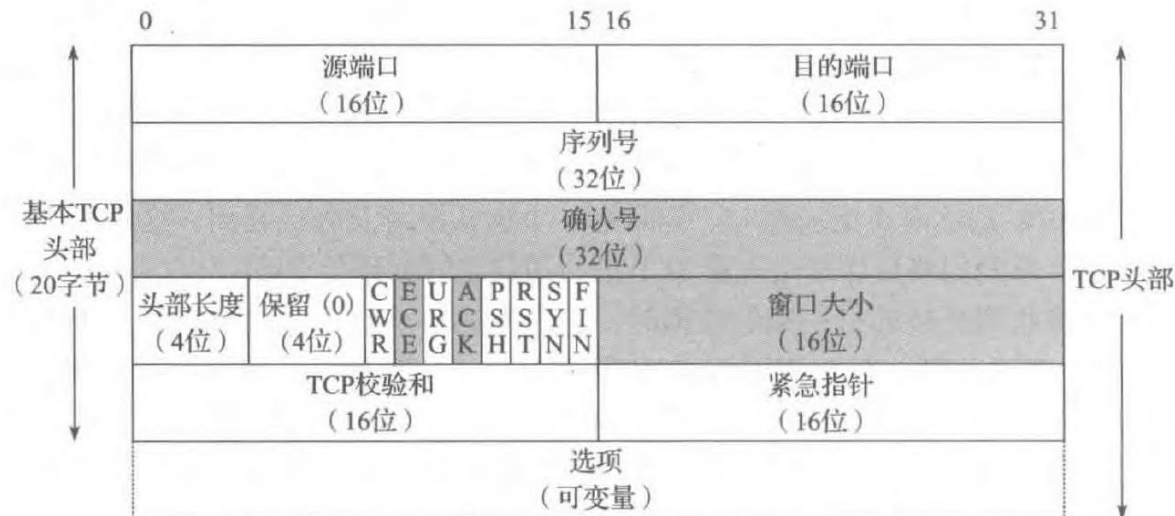


图 12-3 TCP 头部。它的标准长度是 20 字节，除非出现选项。头部长度 (Header Length) 字段以 32 位字为单位给出头部的大小 (最小值是 5)。带阴影的字段 (确认号 (Acknowledgment Number)、窗口大小 (Window Size) 以及 ECE 位和 ACK 位) 用于与该报文段的发送方关联的相反方向上的数据流

下知地理：TCP 可靠字节流

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- TCP 协议本质是带累积正向确认的滑动窗口协议
 - TCP 虽然会给网络层提供 Segment，即报文段，但 TCP 本身是面向字节流的，并非是传输片段。
 - TCP 会自动分片
 - TCP 利用反馈、重传机制，在有损信道上实现可靠传输。
- 如何利用 TCP 传输一个固定大小的文件？
 - 建立连接
 - 发送方直接用 Socket 发送文件，比如以 4KB 为单位读取本地文件在读了 N 个 4KB 之后，终于读取到了 EOL 标志，则最后发送 $(\text{FileSize} - 4 * N)$ KB 的数据，然后发送 FIN，关闭连接
 - 接收方只需要傻傻接收，然后把数据包组合起来即可恢复出原文件。

如何在应用层做文章？

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 上述场景的局限性在于传输的终结由发送方控制
 - 在传输完成之前，只有发送方知道文件的具体大小。
 - 文件是否接收完是由发送方关闭连接所确定的。
- 如何克服上述弊端？
 - 发送方通过自定义应用层协议，提前告知接收方需准备多少磁盘空间（比如 N kB）来存储要接收的数据。
 - 接收方根据 N 的值，从某一个包开始，截取包内容，并往本地的文件描述符写入数据，一旦接收到了 N kB 的数据，就完成接受，并回复接收完成的确认。此后也不需要关闭连接。
 - 接收方回复已完成接收后，发送方关闭打开的文件，并对其文件描述符引用计数减一。

Server 端 ACL 配置

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 自定义服务端业务代码
 - 业务端进程需具备严格的访问控制。服务端只允许某个目录被 Client 读取、写入。
 - ACL 主要针对以下项目进行权限控制
 - 用户 User
 - 用户组 Group
 - 默认属性 Mask
 - 另有 setfacl 之外的方法可实现服务器端的访问控制。
 - Server 端的可读写目录配置需以 shell 脚本或其他形式给出**明确体现**。

应用层协议设计

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 需要建立长连接，代码中要明确指示 keep alive
 - 以下行为是不允许的：Client 发起一次请求并得到响应之后，即关闭 TCP 连接。需要有守护进程维护连接，直至 Client 被手动关闭。
- 需要在 Report 明确写出应用层协议的设计原理
 - 图文并茂地叙述协议设计的思路
 - 时序图
 - 协议头部信息描述图（类似 TCP 头部）
 - 解释协议能准确运行的合理性

加分项

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 文件断点续传
- 基于公钥的身份验证
- 支持同时上传、下载多个文件
- 支持服务端根据用户身份，控制传输带宽

提示： 协议的实现方法

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 仿照 TCP 报头的样式进行实现
 - 划分字段，不同的值代表不同的行为
 - 收到报头，按照 struct 进行解析，然后按照对应的逻辑完成对应的任务。
 - 特别注意，统一
- 用 JSON 编码/解析器完成
 - 这种方法与 struct 报头的方案在理论上是等价的，但是更加现代化。
 - 优势：JSON 是服务器端和客户端都理解的描述语言，因此只要定义了双方的通信解析方法，就可以通过 Json 格式的数据来传输控制报文，无需做字节级的包头解析。
 - 劣势：用 JSON 做控制平面的数据传输固然方便，但实现控制平面与数据平面的分离是颇具挑战的。

仿 TCP 等报文头的设计

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

包类型名称	类型解释	包类型定义
REGISTER_REQUEST	注册请求	00
REGISTER_RESPONSE	注册响应	01
LOGIN_REQUEST	登陆请求	11
LOGIN_RESPONSE	登陆响应	12
CATALOG_REQUEST	远程文件目录请求	21
CATALOG_RESPONSE	远程文件目录响应	22
FILE_REQUEST	文件下载请求	31
FILE_METADATA	文件元数据通知	32
FILE_CONTENT	文件内容	33

技术分享建议

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 先描述好一个问题。
 - 描述一个能让大家感同身受的编程、设计问题，不要一上来就讲你怎么做的。
- How 比 What 重要。
 - 要有不同技术的比较。
- 一定要有 Best Practice 或方法论总结

Thanks

- Thank you for your listening!
- contact information
 - Email: liupeng19@mails.ucas.edu.cn