



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

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 (2022年4月7日23:59前提交到课程网站)
  - 不要压缩为 rar、7zip 等格式，只要 ZIP，不要加密。
  - 文件组织与命名方式见下页
  - 所有代码必须开源，GitHub/Gitee/GitLab 有完整记录。担心代码被抄袭的同学，可以选择在提交之后将 repo 设置为 public.
- 实验考核及评分准则
  - 共有 46 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

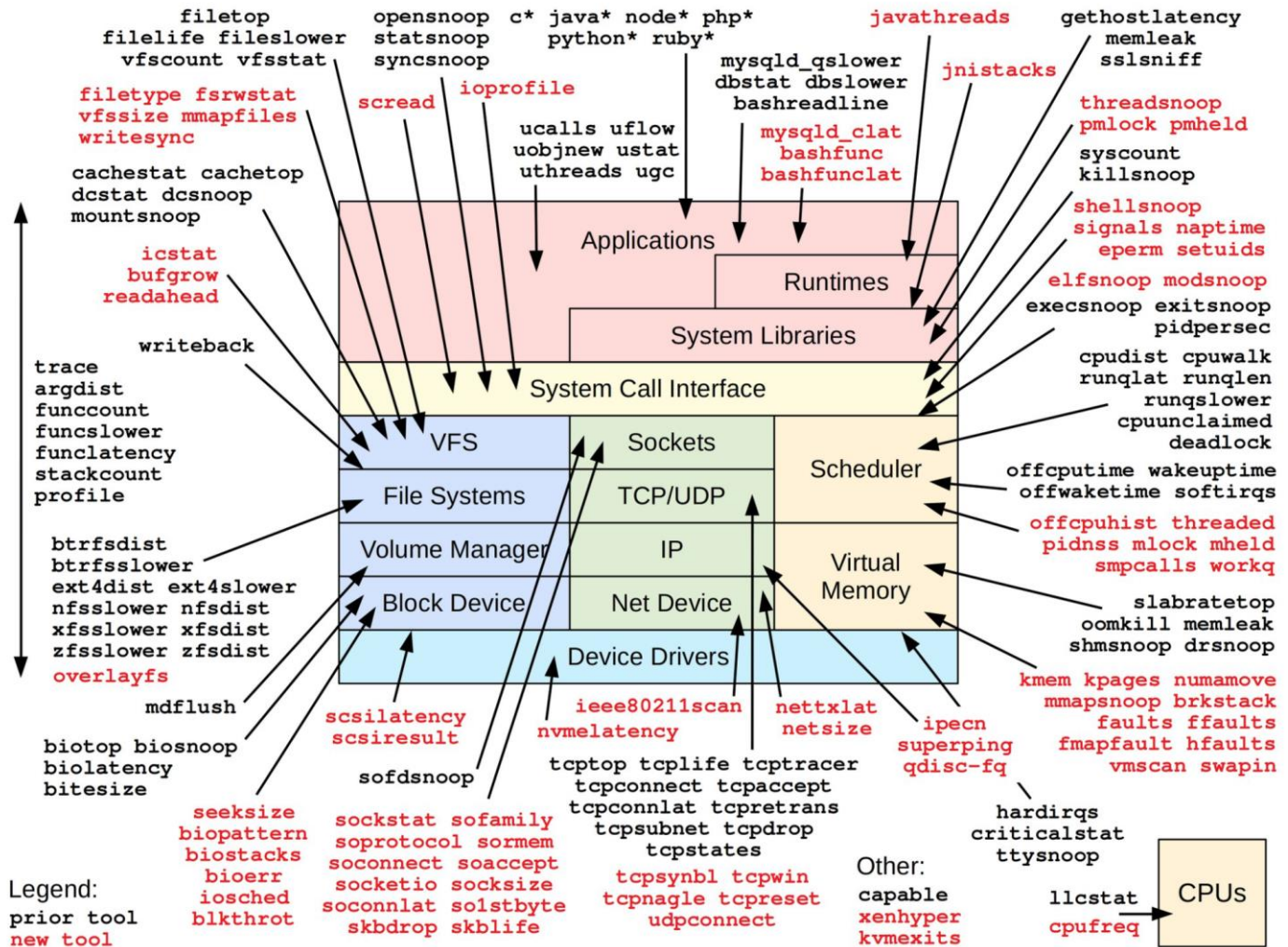
## 1B: 安全文件传输软件设计与实现

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

## 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 --ignorenodev --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

The image shows the Wireshark network protocol analyzer interface. The top pane displays a list of captured packets. The middle pane shows the details of the selected packet (No. 68). The bottom pane shows the raw packet data in hexadecimal and ASCII.

**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 06 00 00 0a 00 00 03 78 e8  (.$@.
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

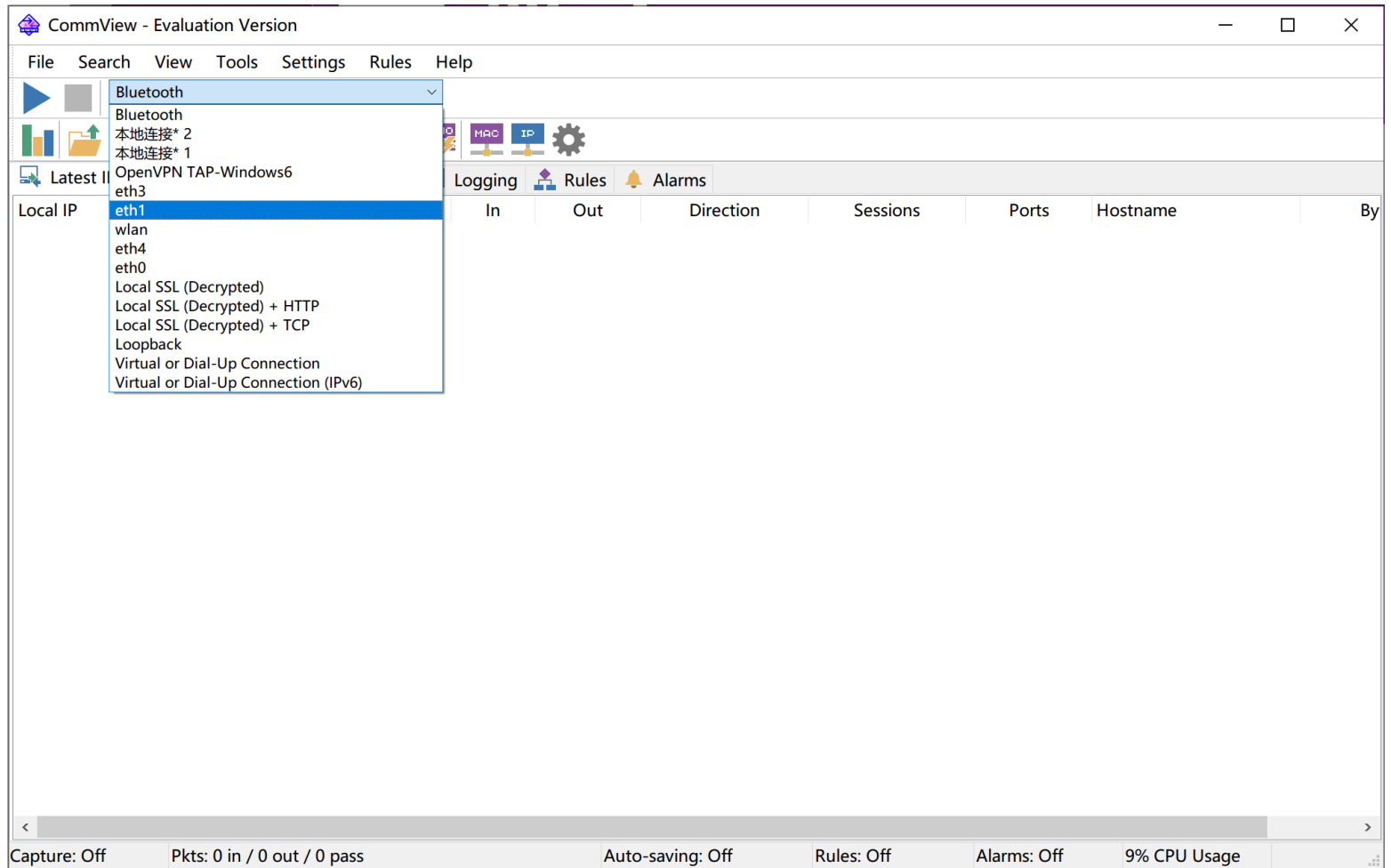
Prompt I

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

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

- 务必理清 OSI 五层模型的概念

- 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

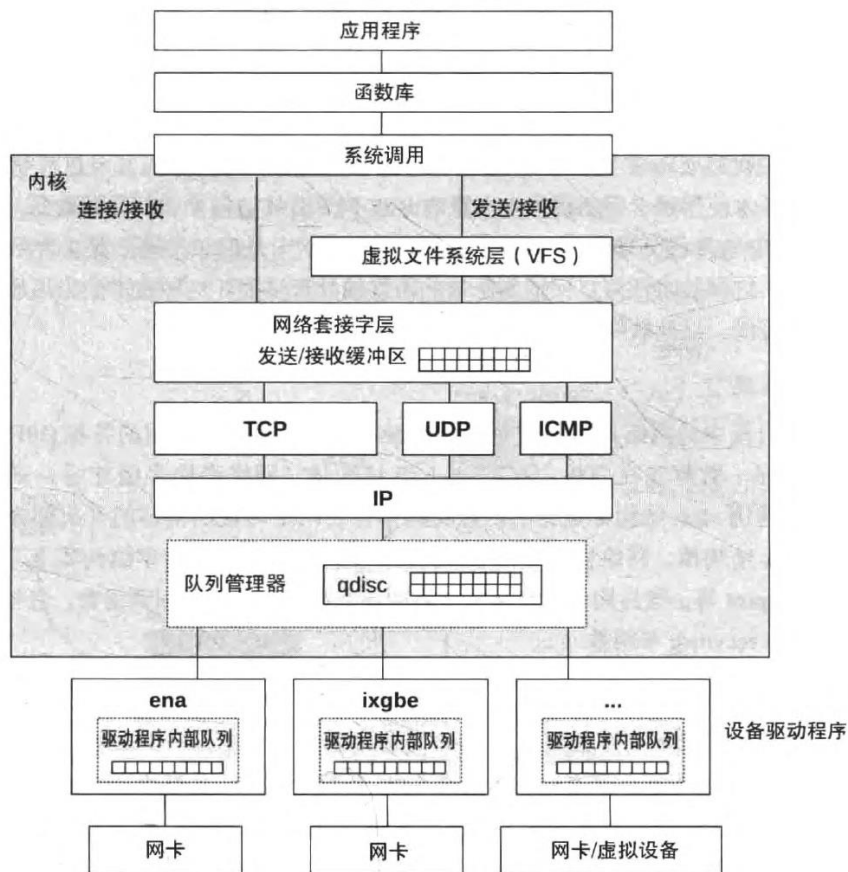
Prompt I

Exp II

ISTs II

Prompt II

Q&A



VFS 抽象层会在 1B 中用到

“一切皆文件”是Linux的基本哲学之一，不仅是普通的文件，包括目录、字符设备、块设备、套接字等，都可以以文件的方式被对待。实现这一行为的基础，正是Linux的虚拟文件系统机制。

图. Linux 网络协议栈

# Linux BPF Internals

Contents

Exp I

ISTs I

Prompt I

Exp II

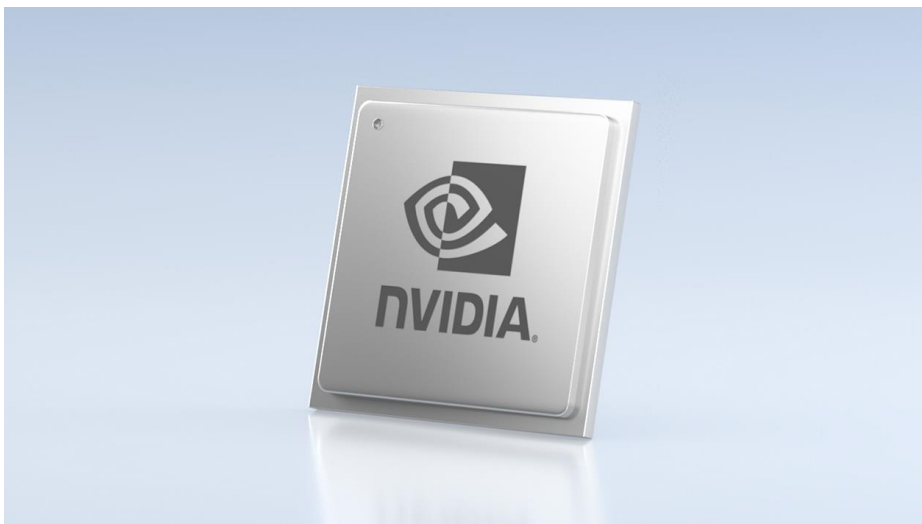
ISTs II

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。



## PRODUCT SPECIFICATIONS

Max Total Bandwidth	400Gb/s
Supported Ethernet Speeds	10/25/40/50/100/200/400GbE
Number of Network Ports	1/2/4
Network Interface Technologies	NRZ (10/25G) / PAM4 (50/100G)
Host Interface	PCIe Gen5.0 x16/ x32
Cards Form Factors	PCIe FHHL/ HHHL, OCP3.0 SFF
Network Interfaces	SFP56, QSFP56, QSFP56-DD, QSFP112, SFP112

# Linux BPF

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- BPF/eBPF 可解决抓包开销问题

- BPF 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 类似，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

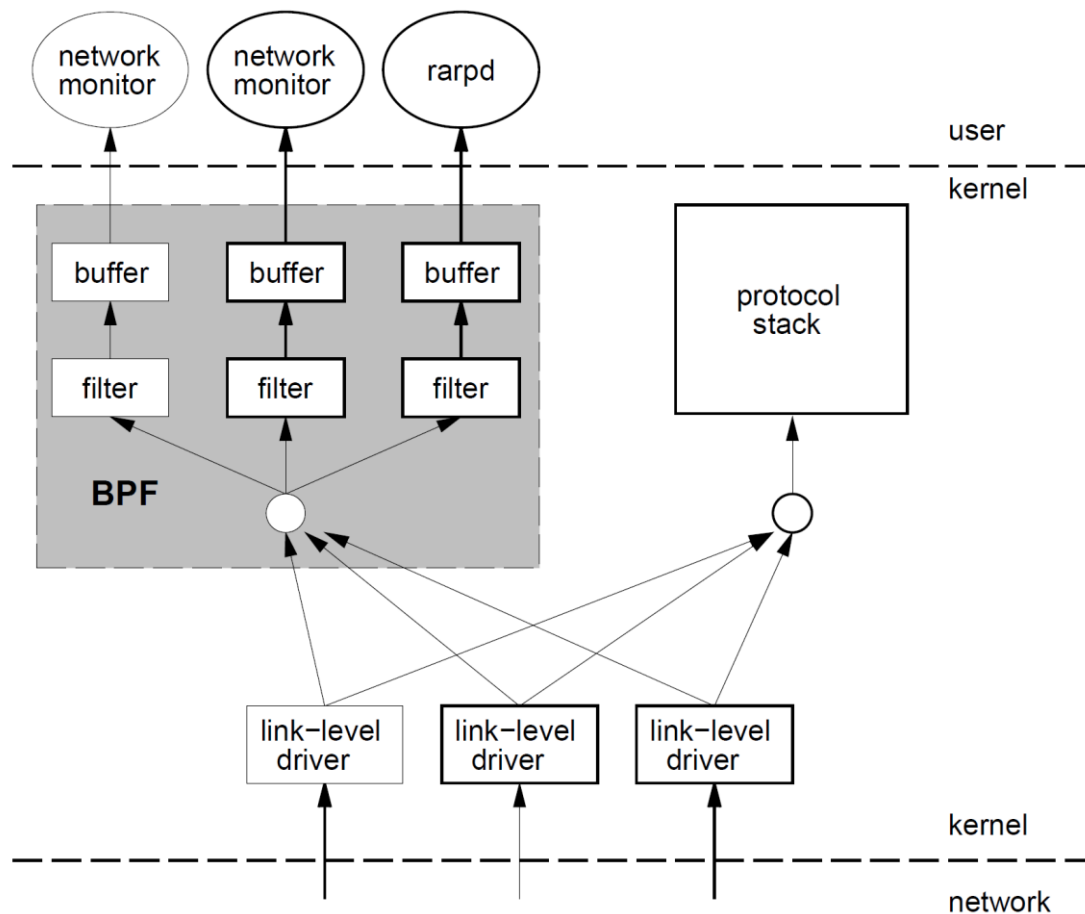


图. BPF 概览

# 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
<b>User</b>	task	yes	any	user based	abort	syscall, fault
<b>Kernel</b>	task	no	static	none	panic	direct
<b>BPF</b>	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

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

# 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](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

- 文件传输是网络的基本功能之一
  - 文件传输属于抽象需求，可细分为多种场景
    - HTTP：通过构造 HTTP 请求实现文件传输（非典型）
    - WebDAV：Http 的扩展，用于在 80 端口进行文件共享
    - FTP：远比 HTTP 复杂，专用于文件传输，不加密
    - SFTP：FTP 的扩展，保持同样的接口，但是支持加密
    - SCP：Secure Copy，基于 SSH 协议在两台主机直接传输
    - SMB：微软独家的 Windows 电脑间的文件共享、打印共享协议，性能安全优异
    - NFS：Unix 系统间的文件系统级共享，权限控制比较弱
  - Note：文件共享有别于块存储共享！
    - 块存储常见于 IDC，与本实验不同，但是功能比较相似
    - IP-SAN、FC-SAN：基于 SAN 存储网络的高性能块共享
    - iSCSI：一种广泛应用的 IP-SAN 技术

# Demo: WinSCP

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

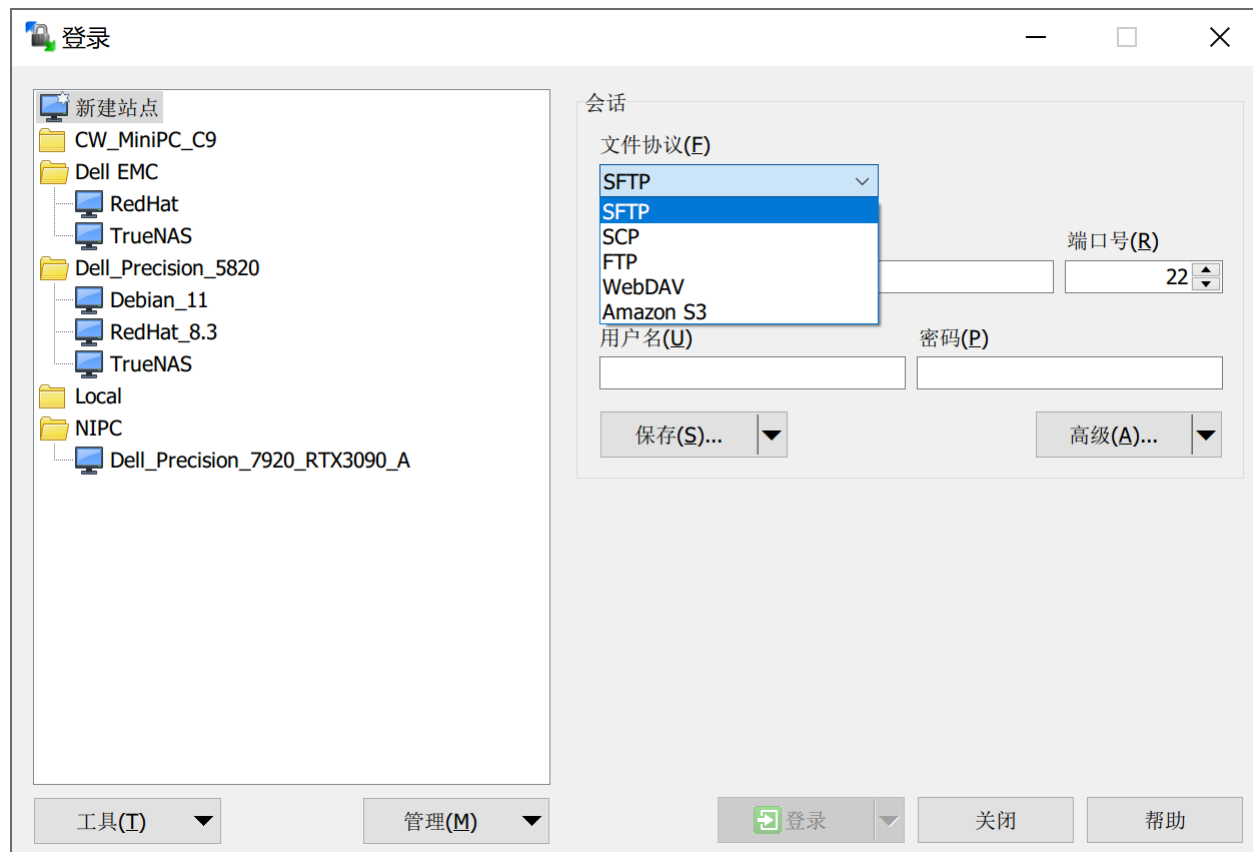


图. WinScp 软件配置界面

# 实验 1B：要求与提示

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 要求实现一种简单但安全的文件传输协议
  - 能保证多用户并发访问时，文件可完整地成功传输
  - 通信全过程可抵御中间人、DDoS、SQL 注入等常见攻击
- 温馨提示
  - 建议使用 Visual Studio 2022 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 (xxQT)
    - 基于 Native API 的图形库封装：wxWdigets
  - 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。
    - 视频里可以讲解一下自己的代码与思路，注意录制时长
    - 视频要展示程序编译、部署、测试的全过程

# 实验 1B：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 (协议的每个模块都要画出时序图)
    - 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

# 功能流程图

Contents

Exp I

ISTs I

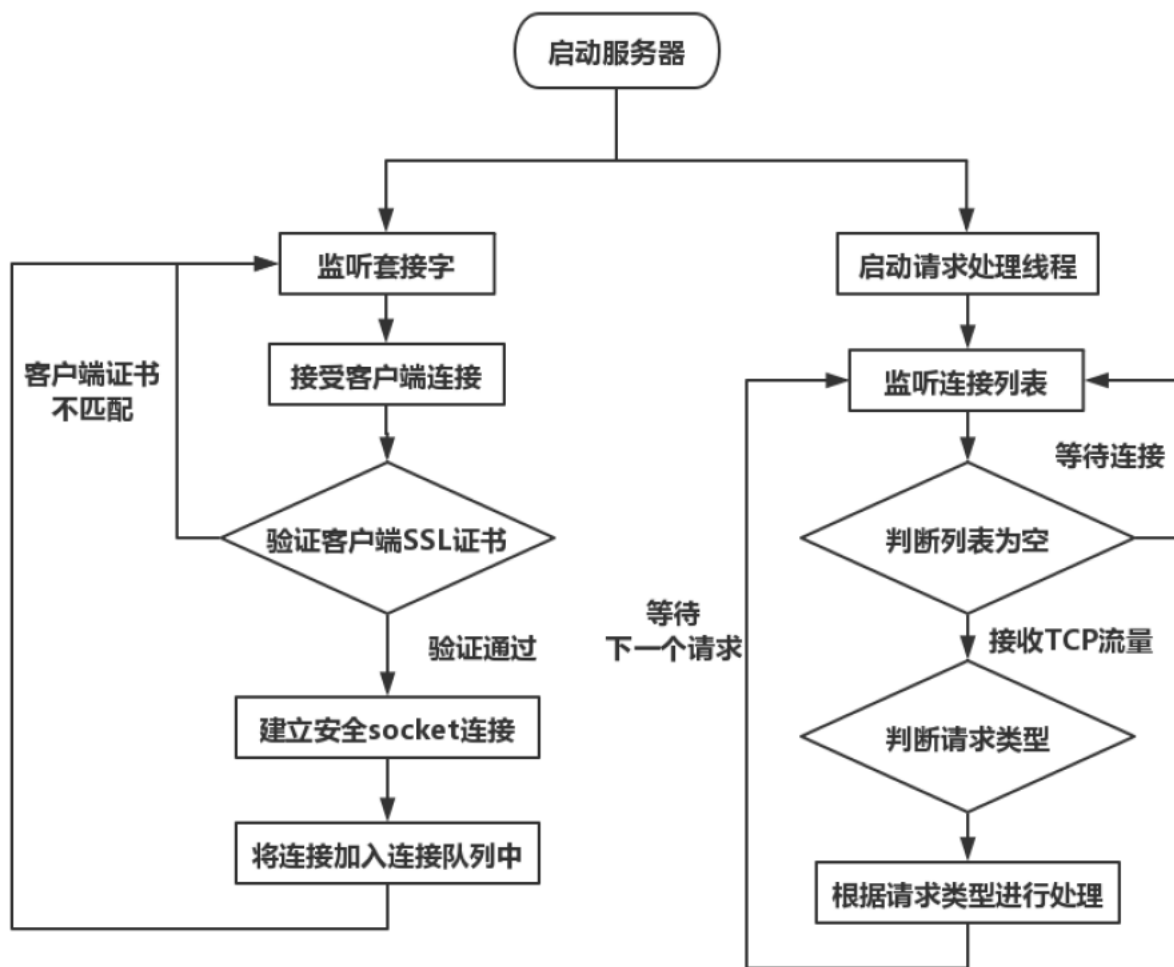
Prompt I

Exp II

ISTs II

Prompt II

Q&A



# 协议时序图

Contents

Exp I

ISTs I

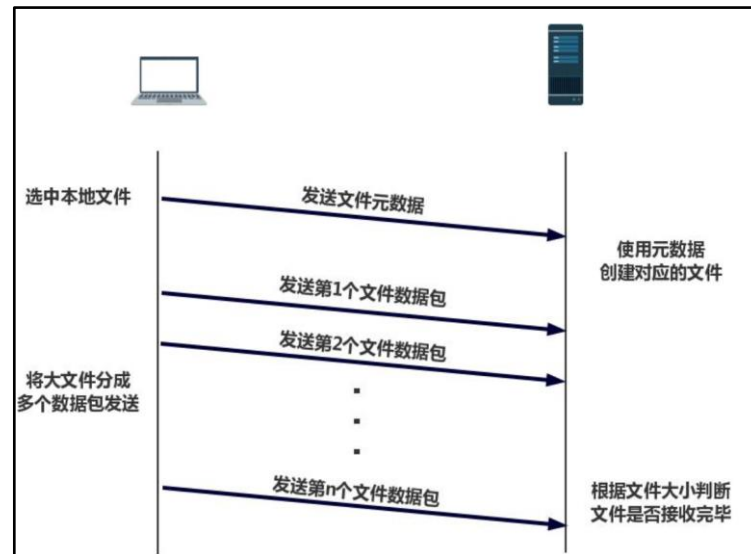
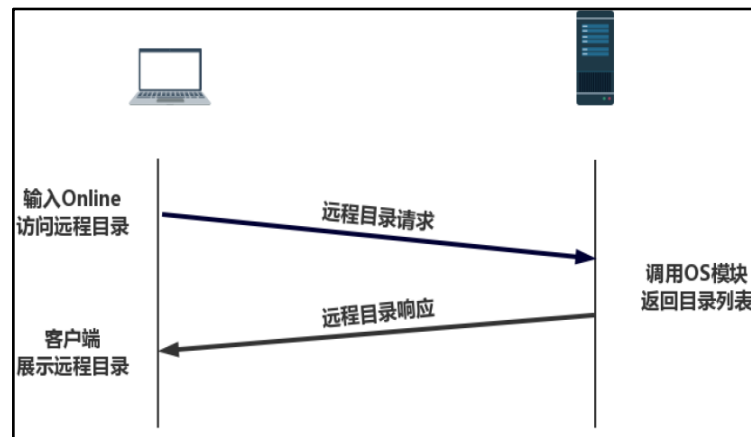
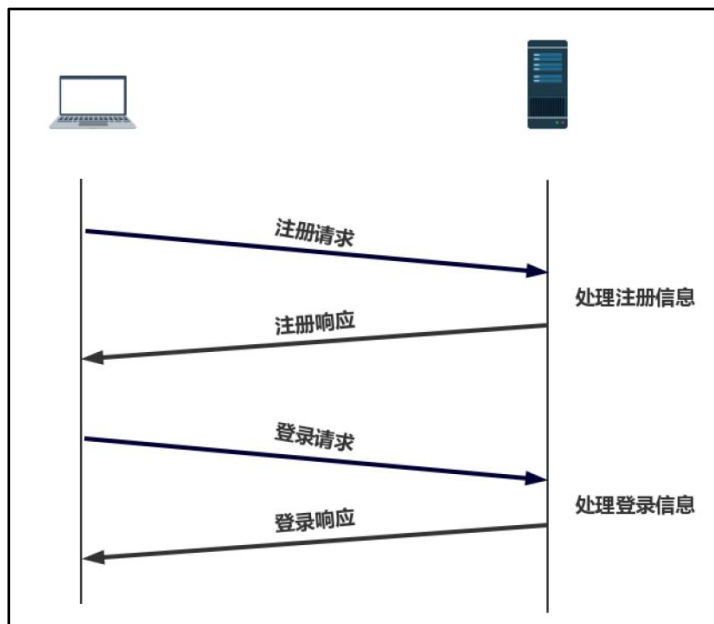
Prompt I

Exp II

ISTs II

Prompt II

Q&A





# 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
  - 双向证书验证亦可
- 支持同时上传、下载多个文件
  - GUI 支持
- 支持服务端根据用户身份，控制传输带宽
  - VIP

# 从 TCP 报头谈协议是什么

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

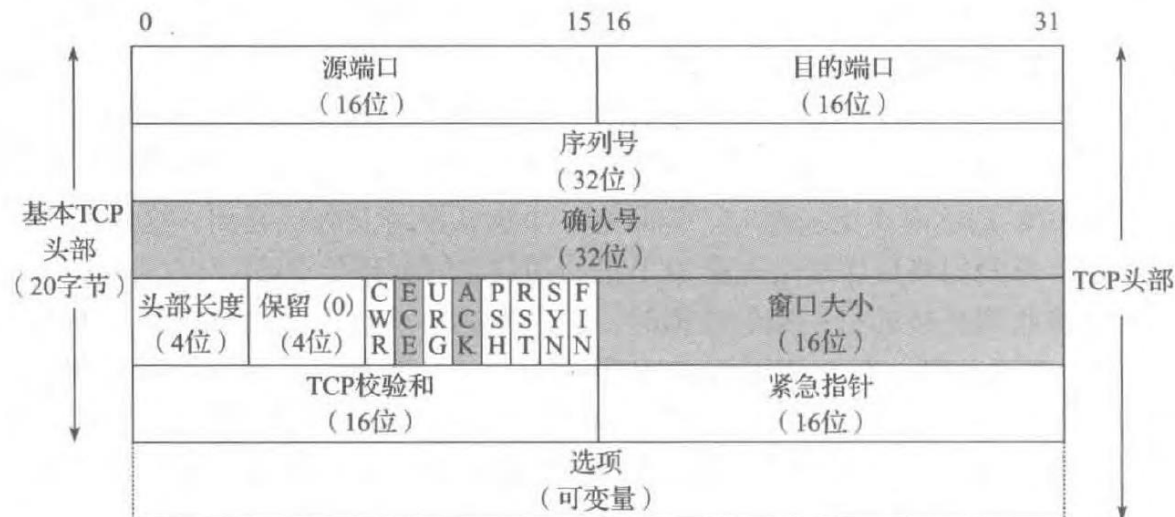


图 12-3 TCP 头部。它的标准长度是 20 字节，除非出现选项。头部长 (Header Length) 字段以 32 位字为单位给出头部的大小 (最小值是 5)。带阴影的字段 (确认号 (Acknowledgment Number)、窗口大小 (Window Size) 以及 ECE 位和 ACK 位) 用于与该报文段的发送方关联的相反方向上的数据流

# 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？
  - 首先建立 TCP 连接：三次握手
  - 发送方用 Socket 发送文件：以 4 KiB 为单位读取本地文件，在读了  $N$  个 4KiB 之后，终于读取到了 EOF 标志，则最后发送  $(FileSize - 4 * N)$  KiB 的数据
  - 发送 FIN，关闭连接：四次挥手
  - 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。

# 如何建立完善的应用层协议

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  $N$  kiB）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  $N$  kiB 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。

# 安全要点：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，以减少 Server 进程被黑客控制后产生进一步的损失
- 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。

# 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，报头将出现在每一个应用层 packet 里，要注意协议的传输开销。

- 用 JSON 编码/解析器完成

- 这种方法与 TCP/IP 的 bitmap + struct 报头方案在理论上是等价的，但是更加现代化、更灵活
- 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。

# 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

# PKI Internals

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- **PKI 技术：解决公钥密码算法中的信任问题**

- 要使用公钥算法，必须明确公钥的归属问题

- Diffie and Hellman 首次公开公钥密码算法时便已意识到此问题，他们设想了一种 online 的可信资料库，但面临许多问题：

- 频繁查询带来的性能开销问题

- 离线使用问题

- 1978 年，Kohnfelder 提出数字证书 (Certificate) 的概念，由证书认证中心 (Certification Authority, CA) 签发证书来解决公钥归属问题。PKI 中的三个实体：

- CA：负责为他人签发证书 (CA 的私钥至关重要！)

- 证书持有者 (Certificate Holder)：有 CA 签发的证书和自己的对应私钥 (而非 CA 的私钥)

- 依赖方 (Relying Party)：安装了 CA 的证书 (CA 的公钥)



# CA 签名过程

Contents

Exp I

ISTs I

Prompt I

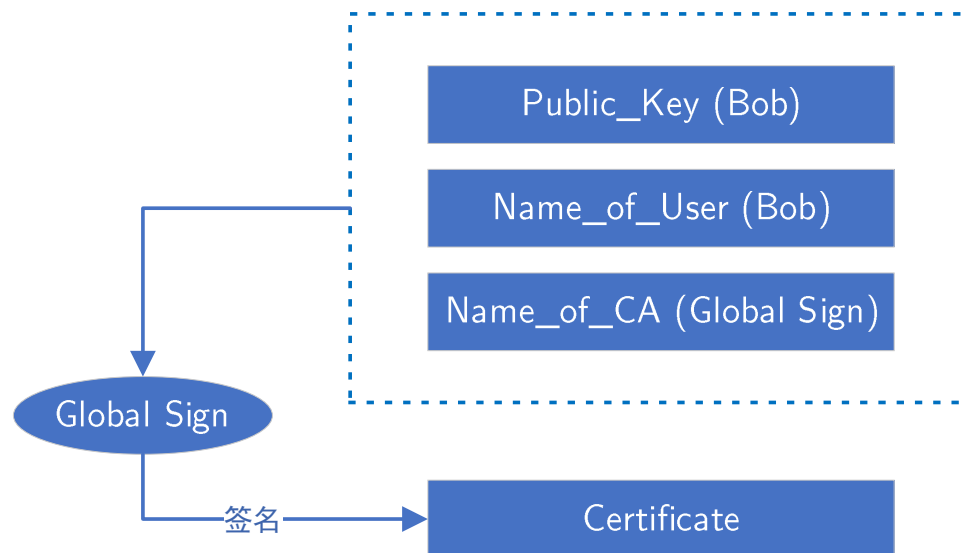
Exp II

ISTs II

Prompt II

Q&A

- 签名过程很严格、很繁琐，需要大量人工参与
  - CA 保证证书申请者与证书所标识的是同一个实体
  - 证书的身份信息也要进行仔细审查
  - 检查 Optional 的信息：电子邮件、IP 地址、DNS 域名



# CA 给自己签发证书的过程

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

# 创建 CA 的信息

# 生成 CA 私钥

```
openssl genrsa -out ca-key.pem 1024
```

# 生成 CA 请求文件

```
openssl req -out ca-req.csr -key ca-key.pem -new
```

# 自签署 CA 的根证书

```
openssl x509  
    -out ca-cert.cer  
    -req -in ca-req.csr  
    -signkey ca-key.pem  
    -days 365
```

# CA 给 Server 签发证书的过程

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

# CA 签发服务器的证书

# 生成服务器私钥

```
openssl genrsa -out server-key.pem 1024
```

# 生成服务器请求文件 (公钥 + 信息)

```
openssl req -out server-req.csr -key server-key.pem -new
```

# 用 CA 的根证书、CA 的私钥签署服务器证书

```
openssl x509
```

```
-out server-cert.cer
```

```
-req -in server-req.csr
```

```
-signkey server-key.pem
```

```
-CA ca-cert.pem
```

```
-CAkey ca-key.pem
```

```
-CAcreateserial
```

```
-days 365
```

# CA 给 Client 签发证书的过程

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

# 创建客户端证书

# 生成客户端私钥

```
openssl genrsa -out client-key.pem 1024
```

# 生成客户端请求文件

```
openssl req -out client-req.csr -key client/client-key.pem -new
```

# 用 CA 的根证书、CA 私钥签发客户端证书

```
openssl x509
```

```
-out client-cert.cer
```

```
-req -in client-req.csr
```

```
-signkey client-key.pem
```

```
-CA ca-cert.cer
```

```
-CAkey ca-key.pem
```

```
-CAcreateserial
```

```
-days 365
```

# 证书使用说明

Contents

Exp I

ISTs I

Prompt I

Exp II

ISTs II

Prompt II

Q&A

- 证书使用者不限于计算机

- 进程、线程等运行时对象之间也可以用证书校验身份

- 以邮件程序为例：

- Alice 给 Bob 发送邮件时，在写好邮件之后，用自己的证书给邮件进行签名 (证书持有者)

- Alice 准备发送时用 Bob 的证书给邮件加密 (证书依赖者)

- 证书需要能撤销

- 证书持有者丢失了自己的私钥

- 需要重新申请新证书

- 废除旧的证书

- 绕过 CA 撤销别人的证书将直接导致服务不可用

# 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: [liup@nipc.org.cn](mailto:liup@nipc.org.cn)