کند و کاو سرآیندهای اترنت و IP بسته‌ها

پروژه دوم درس شبکه‌های کامپیوتری

دانشکده‌ی برق و کامپیوتر

دانشگاه صنعتی اصفهان

بهار ۱۴۰۳

۱ معرفی

در پروژه پیشین با عملکرد نرم‌افزار وایرشارک تا حدی آشنا شدید. وایرشارک یک نرم‌افزار رایگان و متن‌باز برای تجزیه و تحلیل ترافیک شبکه است. این ابزار به شما امکان می‌دهد تا بسته‌های داده‌ای را که در شبکه در گردش هستند، رصد، ضبط و فیلتر کنید. وایرشارک برای تشخیص و رفع مشکلات شبکه، آنالیز پروتکل‌های شبکه و امنیت شبکه بسیار مفید است.

در این پروژه ابتدا با ابزار دیگری به نام tcpdump که به ما این امکان را می‌دهد که ترافیک شبکه را تجزیه و تحلیل کنیم آشنا می‌شویم. مانند وایرشارک، tcpdump امکان رصد بسته‌ها را بر اساس فیلترهای مختلف مانند آدرس IP ، شماره پورت و پروتکل فراهم می‌آورد. تفاوت عمده این برنامه با وایرشارک آن است که tcpdump یک برنامه خط فرمان است و محیط گرافیکی ندارد.
در ادامه کتابخانه‌ها و توابعی در C که توسط آنها می‌توان بسته‌ها را مستقیم از کارت شبکه دریافت نمود و سرآیندهای مربوط به لایه‌های مختلف پروتکلی را بررسی کرد معرفی می‌شوند و از شما خواسته می‌شود خودتان نسخه ساده‌ای از ابزارهایی مانند وایرشارک و tcpdump بنویسید که می‌تواند سرآیندهای اترنت و شبکه را بررسی نماید و اطلاعاتی را با توجه به این سرآیندها به کاربر نمایش دهد.

۲ برنامه tcpdump

برای نصب این برنامه بر روی لینوکس خود دستور

```
sudo apt install tcpdump
```

را وارد کنید. سپس برای اجرای آن دستور

```
sudo tcpdump
```

را در ترمینال اجرا نمایید. این دستور گزینه‌های بسیار زیادی دارد که در صفحه راهنمای لینوکسی (man page) می‌توانید در مورد آن‌ها بخوانید. به عنوان مثال یکی از گزینه‌ها D- است که لیستی از رابط‌های دستگاه را چاپ می‌کند. حال می‌توانید روی هر کدام از رابط‌ها که نیاز داشتید بسته‌هارا دریافت کنید.

مثال: در صورتی که رابطی با نام ens33 وجود داشته باشد، با اجرای دستور `sudo tcpdump ens33` فقط بسته‌هایی که از این رابط عبور می‌کنند بر روی ترمینال چاپ می‌شود.

برای اعمال فیلتر بر روی بسته‌های دریافتی و نمایش خروجی فیلتر از عبارت کلی

```
sudo tcpdump 'expression'
```

استفاده می‌شود. با مراجعه به man 7 pcapfilter می‌توانید در مورد نحوه ساختن فیلترهای مختلف آشنایی پیدا کنید.

مثال: دستورات زیر معادل هستند و برای چاپ بسته‌هایی که در آنها لایه حمل و نقل TCP است استفاده می‌شود.

```
sudo tcpdump -v 'proto \tcp'
sudo tcpdump -v 'tcp'
```

مثال: دستور زیر برای فیلتر کردن بسته‌های ارسالی به یک وب سرور استفاده می‌شود.

```
sudo tcpdump -v 'tcp and dst port 80'
```

همانطور که در این دستور مشاهده می‌کنید در عبارت فیلتر می‌توان از and و or برای ساخت عبارت‌های ترکیبی نیز استفاده نمود.

به طور خاص عبارت proto [expr : size] قدرت انتخاب هر قسمتی از سرآیند هر پروتکل مد نظر و وارسی مقدار آن را در اختیار می‌گذارد (proto می‌تواند مواردی مانند tcp, ip, ether و udp باشد و نوع پروتکل را تعیین می‌کند، expr شماره بایت (با شروع از ۰) مد نظر در پروتکلی که توسط proto مشخص شده است و تعیین size اختیاری (با مقدار پیش‌فرض یک) است و تعداد بایت را مشخص می‌کند).

مثال: برای چاپ بسته‌هایی که در آنها لایه حمل و نقل TCP است علاوه بر دستورات قبل از دستور زیر نیز می‌توان استفاده نمود.

```
sudo tcpdump -v 'ip[9] = 6'
```

زیرا در سرآیند IP بایت دهم (با شماره ۹) شماره پروتکل لایه بالایی را مشخص می‌کند که برای TCP مقدار آن برابر با ٦ خواهد بود.

سوال ۱: الف) به دستور tcpdum چه فیلتری باید داده شود تا تنها بسته‌های غیر IP را نمایش دهد؟
ب) دستور tcpdum را در یک ترمینال با فیلتر قسمت الف اجرا کنید. در یک ترمینال دیگر ابتدا محتویات جدول arp لپ‌تاپ خود را از طریق دستور arp نشان دهید. سپس با استفاده از دستور

```
sudo arp -d <ip address>
```

به تعداد لازم این جدول را خالی نمایید (به جای <ip address> باید آدرس IP متناظر با رکوردهای مختلف جدول را قرار دهید). حال به ترمینالی که در آن tcpdum در حال اجرا هست بازگردید و مشاهدات خود را گزارش و توجیه کنید.
ج) لپ‌تاپ خود را از طریق هات‌اسپات گوشی همراه به اینترنت متصل کنید و مشاهدات خود در ترمینالی که در آن tcpdump در حال اجرا هست را گزارش و توجیه نمایید.

۳ دسترسی مستقیم به بسته‌های دریافتی با استفاده از packet socket

در این بخش می‌خواهیم با تعریف سوکت‌هایی از نوع بسته‌ای (packet socket) برای دریافت مستقیم و بررسی بسته‌های شبکه در یک برنامه C استفاده کنیم. برای این منظور طبق روش زیر سوکت ساخته می‌شود:

```
#include<sys/socket.h>
#include<netpacket/packet.h>
#include<net/Etherne.h>  /* Fordata linl layer (DLL) protocols */
sockfd = socket(PF_PACKET, SOCK_RAW,htons(ETH_P_ALL));
```

تابع int socket(int domain, int type, int protocol) یک توصیفگر از جنس سوکت بازمی‌گرداند. آرگومان اول این تابع، خانواده پروتکل است که در اینجا PF_PACKET تنظیم می‌شود. آرگومان دوم، نوع سوکت را مشخص می‌کند و در اینجا با SOCK_RAW مقداردهی می‌شود و برای کار با بسته‌های خام شامل سرآیند لایه لینک مناسب است (در صورتی‌که با سرآیند لایه لینک کاری نداشته باشیم می‌توان از مقدار SOCK_DGRAM استفاده کرد. در این صورت در مسیر دریافت ابتدا سرآیند لایه لینک از بسته دور ریخته می‌شود و سپس به برنامه تحویل داده می‌شود. در مسیر ارسال نیز، بسته بدون سرآیند لایه لینک به سوکت تحویل می‌شود و سرآیند لایه لینک به صورت خودکار قبل از ارسال به بسته اضافه می‌شود). آرگومان سوم به نام protocol نوع پروتکل را مشخص می‌کند و در این جا باید با شماره پروتکل IEEE 802.3 با ترتیب بایت شبکه (big endian) جایگزین شود. لیست پروتکل‌های مجاز در فایل <linux/if_ether.h> آمده است. با مقداردهی این آرگومان به صورت htons(ETH_P_ALL) تمام بسته‌های اترنت می‌توانند توسط سوکت دریافت و یا ارسال شوند (تابع uint16_t htons(uint16_t hostshort) برای

تبدیل ترتیب بایت به صورت big endian استفاده می‌شود). توجه نمایید برنامه‌هایی که از packet socket استفاده می‌کنند، نیاز به دسترسی‌های ویژه sudo برای اجرا دارد.
برای دریافت بسته‌ها از سوکت ایجاد شده، می‌توان از تابع recvfrom به صورت زیر استفاده می‌شود.

```
int sockfd, len;
char buffer[2048];
struct sockaddr_ll phyaddr;
len = sizeof(struct sockaddr_ll);
recvfrom(sockfd,buffer,sizeof(buffer),0,(struct sockaddr *)&phyaddr,&len);
```

فرم کلی این تابع به صورت

```
int recvfrom(int packet_socket, void *buf, unsigned int len, int flags, struct sockaddr
*from, unsigned int *fromlen);
```

است که در آن آرگومان اول شناسه (توصیفگر) سوکت، آرگومان دوم، اشاره‌گر به یک بافر که قرار است بسته درساقتی در آن ریخته شود، آرگوان سوم، طول بافر و آرگومان چهارم برای تنظیم عملکرد تابع از طریق یک کردن تعدادی بیت (پرچم‌های کنترلی) است. همچنین آرگومان پنجم در صورتی‌که NULL داده نشود می‌تواند توسط پروتکل لایه پایین برای در اختیار گذاشتن آدرس مبدا مورد استفاده قرار گیرد (توجه کنید که فارغ از نوع دقیق آدرس مبدا لایه پایین که مثلا از نوع struct sockaddr_in برای آدرس‌های IP و یا نوع struct sockaddr_ll برای آدرس‌های MAC است، آرگومان پنجم می‌بایست به نوع کلی struct sockaddr تغییر نوع [۱] دهد). در نهایت آرگومان ششم اندازه واقعی آدرس که در بافر from ریخفته شده است را نشان می‌دهد.
سوال ۲: با مراجعه به فایل‌های netpacket/packet.h و sys/socket.h ساختار داخلی ساختمان داده‌های struct sockaddr_ll و struct sockaddr را تعیین کنید.

راهنمایی: آدرس فایل‌های سرآیند [۲] به صورت نسبی بر مبنای آدرس پایه /usr/include و یا usr/include/x86_64-linux-gnu می‌باشد.

برنامه ۱: برنامه‌ای به نام dllsniffer.c بنویسید که تمام بسته‌هایی که به کارت شبکه دستگاه شما وارد می‌شود را در مبنای هگزا دسیمال نمایش دهد. هم‌چنین در مورد هر بسته، نوع آن (Outgoing, Incoming, Broadcast, Multicaast) نیز مشخص شود. شکل زیر نمونه خروجی برنامه را نشان می‌دهد:

```
Outgoing:
647002f0f616ac7ba14f4c0f08004500005b8da04000400682c8c0a801664a5b1dcbb50401bb53608bb4eeddc3ef8018004b31f400000101080ab010d198dc16c
9b91703030022d6c2f20b0b4113ce6ab34a8fb4ba03abf31475c04e1a4c97d199eba76289b7b5cadf
Incomming:
ac7ba14f4c0f647002f0f616080045280034bc59400027066d0e4a5b1dcbc0a8016601bbb504eeddc3ef53608bdb801000f5da5500000101080adc16cab4b010d
198
Outgoing:
01005e0000fbac7ba14f4c0f080046c000200000400001024l0ec0a80166e00000fb94040000160000904e00000fb
Broadcast:
ffffffffffff647002f0f61608004500004848c800000111ec8cc0a80101c0a801ff020802080034efca0202000000020000000000000000000000000000000000
0020002000064402111ffffffff0000000000000001
```

سوال ۳: با توجه به شکل فوق،آدرس MAC دستگاهی که برنامه بر روی آن اجرا شده است چیست؟

راهنمایی: برای تعیین نوع بسته دو راه حل وجود دارد. یکی بر مبنای بررسی دستی آدرس‌های MAC در سرآیند اترنت و مقایسه آنها با آدرس‌های Broadcast ، آدرس خود دستگاه و .... و دیگری (روش راحت‌تر) با استفاده از اطلاعات در بخش

[۱] typecast
[۲] header file

sll_pkttype از متغیر phyaddr. در مورد روش دوم می‌توانید با مراجعه به فایل سرآیند netpacket/packet.h اطلاعات لازم را بدست آورید.

برنامه ۲: برنامه قبلی را توسعه دهید به نحوی که در کنار نوع بسته، نوع پروتکل لایه بالاتر (IP یا ARP) نیز نمایش داده شود و این برنامه را dllsniffer2.c نامگذاری کنید.

تصویر زیر نمونه خروجی برنامه فوق را نشان می‌دهد:

```
Upper Protocol: IP, Multicast:
01005e000001647002f0f61608004500001c4f8600000102c8afc0a80101e00000011164ee9b0000000001005e7ffffaac7ba14f4c0f080046c00020
Upper Protocol: IP, Outgoing:
01005e7ffffaac7ba14f4c0f080046c000200000400000102320fc0a80166effffffa940400001600fa04effffffa
Upper Protocol: ARP, Outgoing:
ffffffffffffac7ba14f4c0f080600010800060400001ac7ba14f4c0fc0a80166000000000000c0a80101
Upper Protocol: ARP, Incomming:
ac7ba14f4c0f647002f0f6160806000108000604000264700 2f0f616c0a80101ac7ba14f4c0fc0a801660166effffffa9404000016000000effffffa
Upper Protocol: IP, Outgoing:
647002f0f616ac7ba14f4c0f08004500005ba0da400040066f8fc0a801664a5b1dcaa25e01bbafa5ccdb8dae907280180026587200000101080ade36f50d02a6e
711170303002239562014 91b4fd0acb251ffea9b4c5b525d1b31386097934a88d2e82443c2d15a37f
Upper Protocol: IP, Incomming:
ac7ba14f4c0f647002f0f61608004528005beef5400026063b4c4a5b1dcac0a8016601bba25e8dae9072afa5cd02801800f5c6e700000101080a02a7cadcde36f
50d1703030022f6e79907cbaeeabffdb9f7ce1cec2a4ab7c458f8fdfad740a325f39cbf05dc803409
```

سوال ٤: با توجه به تصویر فوق و با فرض اینکه پیام‌های ARP در واکنش به اجرای دستور مربوط به پاک کردن رکوردهای جدول ARP ایجاد شده است، آدرس MAC مربوط به روتر دروازه [۳] که همسایه دستگاه است چیست؟

راهنمایی: برای نوشتن این برنامه از تغییر نوع اشاره‌گر buffer به نوع *struct ethhdr استفاده نمایید، سپس با دسترسی به متغیر h_proto در داخل این ساختار، نوع پروتکل لایه بالا را بررسی نمایید (این ساختمان داده در فایل سرآیند linux/if_ether.h تعریف شده است. همچنین انواع مقادیری که متغیر h_proto به ازای پروتکل‌های مختلف می‌گیرد نیز در همین فایل آمده است).

سوال ۵: در نوشتن برنامه فوق آیا نیاز به استفاده از تابع ntohs وجود دارد؟ توضیح دهید.

برنامه ۳: برنامه ۲ را توسعه دهید به نحوی که اگر پروتکل لایه بالا IP است، طول سرآیند IP ، طول کل بسته IP بر حسب بایت و همچنین شماره پروتکل بالای سر IP را نیز چاپ کند. این برنامه را dllsniffer_pro.c نامگذاری نمایید. تصویر زیر نمونه خروجی برنامه را نشان می‌دهد:

```
Upper Protocol: IP (header len: 20, total len: 52, proto: 6), Incomming:
00000000000000000000000008004500 0034f29b400040064a267f0000017f0000010050d444fc064542f3c510388010002bfe2800000101080a39c66d2839c66
d28
Upper Protocol: IP (header len: 20, total len: 72, proto: 17), Broadcast:
ffffffffffff647002f0f61608004500004852dd00000111e277c0a80101c0a801ff02080208 0034efca0202000000020000000000000000000000000000000000
0020002000064402111ffffffff0000000000000001
Upper Protocol: IP (header len: 20, total len: 28, proto: 2), Multicast:
01005e000001647002f0f61608004500001c52df00000102c556c0a80101e00000011164ee9b0000000001005e7ffffaac7ba14f4c0f080046c00020
Upper Protocol: IP (header len: 24, total len: 32, proto: 2), Outgoing:
01005e7ffffaac7ba14f4c0f080046c000200000400000102320fc0a80166effffffa940400001600fa04effffffa
```

سوال ٦: با توجه به شماره پروتکل‌ها، پروتکل بعد از پروتکل IP دربسته‌های نشان داده شده در تصویر کدام است؟

برای دسترسی به قسمت‌های مختلف سرآیند IP می‌توانید از ساختار struct ip و یا struct iphdr که در فایل سرآیند netinet/ip.h تعریف شده‌اند استفاده کنید. به این نحو که ابتدا اشاره‌گر buffer را به اندازه طول سرآیند اترنت افزایش داده و سپس آن را به اشاره‌گری از نوع struct ip و یا struct iphdr تغییر نوع دهید.

سوال ۷: ساختار داده struct ip به صورت زیر در فایل netinet/ip.h تعریف شده است:

```
struct ip
{
#if __BYTE_ORDER == __LITTLE_ENDIAN
unsigned int ip_hl:4;               /* header length */
```

[۳] gateway router

```
unsigned int ip_v:4;                /* version */
#endif
#if __BYTE_ORDER == __BIG_ENDIAN
unsigned int ip_v:4;                /* version */
unsigned int ip_hl:4;               /* header length */
#endif
u_int8_t ip_tos;                    /* type of service */
u_short ip_len;                     /* total length */
u_short ip_id;                      /* identification */
u_short ip_off;                     /* fragment offset field */
#define IP_RF 0x8000                    /* reserved fragment flag */
#define IP_DF 0x4000                    /* dont fragment flag */
#define IP_MF 0x2000                    /* more fragments flag */
#define IP_OFFMASK 0x1fff               /* mask for fragmenting bits */
u_int8_t ip_ttl;                    /* time to live */
u_int8_t ip_p;                      /* protocol */
u_short ip_sum;                     /* checksum */
struct in_addr ip_src, ip_dst;      /* source and dest address */
};
```

چند خط اول در این قطعه کد که برای تعریف بخش‌های نسخه IP و طول سرآیند IP هستند جالب و در عین حال مرموز است! در این راستا تحقیقات زیر را انجام دهید:

الف) دلیل وجود شرط‌های قبل از کامپایل [٤] که نحوه ترتیب بایت را مشخص می‌کند چیست؟ در پردازنده‌های intel کدام یک از این شرط‌ها برقرار است؟

ب) منظور از 4: در کنار نام متغیرهای ip_hl و ip_v چیست؟ فرض کنید که ساختار فوق فقط شامل تعریف این دو متغیر بود و سایر قسمت‌ها نبودند. در این صورت خروجی دستور `sizeof(struct ip)` چه عددی می‌شود؟ چرا؟

٤ شیوه تحویل

برای تحویل این پروژه یک پوشه به نام ComputerNetworks_StudentID_Project2 بسازید (به جای StudentID باید شماره دانشجویی خود را قرار دهید) که شامل محتوای زیر باشد:

۱. یک فایل PDF: شامل پاسخ به سوالات ۱ تا ۷.

۲. یک پوشه با نام Source که کدهای C برنامه‌های ۱ تا ۳ (dllsniffer.c, dllsniffer2.c, dllsniffer_pro.c) و Makefile در آن قرار دارد. .

[٤] preprocess

در نهایت این فایل‌ها را فشرده کرده و به صورت یک فایل با نام ComputerNetworks_StudentID_Project2 و فرمت zip در سامانه یکتا ارسال کنید.