برنامه‌نویسی شبکه و بررسی پروتکل HTTP

پروژه اول درس شبکه‌های کامپیوتری

دانشکده‌ی برق و کامپیوتر

دانشگاه صنعتی اصفهان

بهار ۱۴۰۳

۱ معرفی

این پروژه شامل دو قسمت «برنامه نویسی شبکه» و «بررسی بسته‌های پروتکل HTTP » است.

در قسمت اول با استفاده از سوکت‌های TCP به زبان C و در محیط سیستم عامل اوبونتو یک پوسته وارونه (Reverse Shell) ساده نوشته می‌شود که به ما کمک می‌کند دستورات مورد نظر خود را بر روی کامپیوتر کلاینت اجرا کرده و نتیجه آن‌ها را از راه دور مشاهده کنیم.

در قسمت دوم، با استفاده از برنامه وایرشارک (Wireshark) که یک برنامه برای بررسی ترافیک عبوری در شبکه است، بسته‌های ارسالی و دریافتی توسط پروتکل HTTP را به ازای حالت‌های مختلف (ارسال و دریافت فایل‌های کوچک و بزرگ و وبسایت‌هایی که توسط نام کاربری و کلکه عبور محافظت شده هستند) مشاهده و بررسی خواهیم نمود.

۲ برنامه نویسی شبکه

۱.۲ پوسته وارونه

پوسته یا شل (shell) یک برنامه است که منتظر دریافت نام یک برنامه در خط فرمان می‌باشد تا آن را اجرا نماید. اگر شما به پوسته یک کامپیوتر دسترسی داشته باشید عملا مالک آن هستند!. برنامه‌های پوسته انواع و اقسام مختلفی دارند که از معروف‌ترین آنها Bourne shell و C shell هستند. جدول زیر بعضی از دستورات مهم پوسته را نشان می‌دهد:

Commands	Comments	Example
pwd	current working directory	
echo	display a line of text	echo hello world, echo $PATH
ls	list files and directories	ls /home
cd	change directory	cd /home, cd ..
nano, vi, emacs, ...	text editors	nano myfile
cat, nl, head, tail, less, more	current working directory	cat myfile
mv	move or rename	mv myfile testfile
cp	copy	cp testfile /home/Desktop/
rm	remove file (and dir with -rf)	rm testfile
mkdir	make directories	mkdir mydir
locate	search the entire filesystem	locate matlab
wheris	finding binaries	whereis ls
find	the most powerful and flexible	find /etc -type f -name apache2
grep	searching keywords in texts	grep -nri hello testfile
sed	find and replace	sed s/hello/Hello/g testfile
fdik, mkfs	partitioning and formating	

در بسیاری از مواقع، وقتی که می‌خواهیم کد مخربی را بر روی یک دستگاه اجرا کنیم، به آن دستگاه دسترسی مستقیم نداریم. در چنین زمان‌هایی به جای اینکه به دستگاه قربانی متصل شویم، کاری می‌کنیم که قربانی به دستگاه ما متصل شود. برای این کار ابتدا از یک سرور منتظر اتصال می‌مانیم، سپس از طریق فریب دادن فرد قربانی یا دستکاری بسته‌های شبکه، اتصالی از دستگاه قربانی به سرور برقرار کرده و ورودی و خروجی پوسته دستگاه قربانی را به این اتصال هدایت می‌کنیم. به خاطر روند اتصال وارونه این حمله، به آن پوسته وارونه (Reverse Shell) می‌گویند.

۱.۱.۲ ساخت یک پوسته وارونه ساده

یک پوسته وارونه شامل دو بخش سرور و کلاینت است. در ادامه در مورد برنامه‌ای که برای بخش سرور و بخش کلاینت یک پوسته وارونه ساده باید نوشته شود توضیح داده می‌شود.

◄ بخش سرور

برنامه سرور وظیفه ارسال دستورات مخرب به کلاینت‌ها و نمایش خروجی آن‌ها را بر عهده دارد. این سرور باید در شروع کار خود بر روی پورتی که از طریق خط فرمان و بوسیله مدیر سرور مشخص شده شروع به گوش دادن کند. (ممکن است که این پورت از قبل پر باشد و سرور نتواند به آن متصل شود. در اینصورت نیاز است که پس از چاپ یک پیام خطای معنادار برنامه بسته شود.)

قطعه کد زیر دستوری که برای اجرای برنامه سرور می‌بایست در پوسته کامپیوتر سرور استفاده شود را نشان می‌دهد (در این صورت شماره پورتی که سرور بر روی آن گوش می‌دهد ۳۰۰۰ خواهد بود)

```
$ ./server 3000
```

فرض بر این است که کاربرانی که قرار است به این سرور وصل شوند، از قبل آدرس IP و پورت این سرور را می‌دانند. پس از اتصال یک کاربر، سرور پیام زیر را چاپ می‌کند

```
A new client connected to the server:
127.0.0.1 $
```

که در آن 127.0.0.1 آدرس IP کلاینتی هست که به سرور متصل شده است. در این حالت در صورتی که یک دستور معتبر پوسته (shell) بعد از علامت $ توسط کاربر پشت برنامه سرور تایپ شود این دستور برای کلاینت ارسال می‌شود تا آن را بر روی کامپیوتر کلاینت اجرا کند و نتیجه را برای سرور ارسال نماید.

تصویر زیر خروجی برنامه را در شرایطی که دستور lscpu توسط کاربر سرور تایپ شود نشان می‌دهد.

```
$ ./malserver 3000
A new client connected to the server:
127.0.0.1 $lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 60
Model name:            Intel(R) Core(TM) i7-4702MQ CPU @ 2.20GHz
```

```
Stepping:              3
CPU MHz:               800.292
CPU max MHz:           3200.0000
CPU min MHz:           800.0000
BogoMIPS:              4390.61
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              6144K
NUMA node0 CPU(s):     0-7
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
   mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall
    nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl
   xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl
    vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 movbe popcnt
    tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm cpuid_fault epb
   tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep
   bmi2 erms invpcid xsaveopt dtherm ida arat pln pts

127.0.0.1 $
```

توجه داشته باشید که در پیاده‌سازی این برنامه نیازی به ایجاد منطق قبول کردن و پاسخ دادن به درخواست کاربران همزمان نیست و می‌توانید به کاربرانی که قصد اتصال به سرور دارند به صورت پشت سر هم و بدون همزمانی پاسخ دهید اما باید به نکات زیر توجه کنید:

- مدیریت بافر: بسته‌های ارسالی بوسیله سوکت اندازه مشخصی ندارند و می‌تواند بزرگ یا کوچک باشد. اما سوکت TCP مورد استفاده در هر زمان اندازه کوچک و مشخصی از داده را در خود نگه می‌دارد. در این برنامه باید به این نکته توجه کرده و برنامه خود را طوری بنویسید که داده‌ها به طور کامل و صحیح بوسیله سوکت ارسال شوند. به عنوان نمونه دستوری که توسط کاربر سرور درخواست می‌شود ممکن است دستوری مانند dmesg باشد که خروجی کاملا حجیمی تولید می‌کند.
- دقت به مقادیر بازگشتی از توابع: تابع send(.) ، در هنگام نوشتن بر روی سوکت ، مقدار بایت‌های نوشته شده و تابع recv(.) در هنگام خواندن، مقدار بایت‌های خوانده شده را برای ما بازگردانی می‌کند. لازم است که به این مقادیر توجه کرده و بر اساس آن‌ها روند اجرای برنامه را کنترل کنید. recv(.) دقت کنید که در تصویر بالا بعد از آنکه ارسال خروجی دستور توسط کلاینت به اتمام رسیده است، سرور این موضوع را متوجه شده است و مجدد خط فرمان جدید (127.0.0.1 $) را نشان می‌دهد و منتظر دریافت فرمان جدید از کاربر پشت برنامه سرور است.

  راهنمایی: برای تحقق این موضوع باید کلاینت و سرور یک قراردادی را برای تعیین انتهای خروجی دستورات با

یکدیگر توافق کنند (مثلا انتهای پیام با ارسال ۳ کارکتر * توسط کلاینت مشخص شود).
سوال: درباره سایر مدل‌های همزمانی و چالش‌های آنها تحقیق کنید و در گزارش خود درباره آن‌ها به صورت کوتاه توضیح دهید.

◄ بخش کلاینت

برنامه کلاینت وظیفه دریافت پیام‌های سرور، اجرای آن‌ها و برگرداندن پاسخ به سرور را بر عهده دارد. فرض بر این است که برنامه کلاینت از قبل آدرس IP و پورت برنامه سرور را می‌داند. برنامه پس از اتصال به سرور، منتظر دریافت دستور می‌ماند. هر زمان که دستوری از سمت سرور ارسال شد، برنامه کلاینت آن را بدون خبر دادن به کاربر اجرا کرده و خروجی آن را بر روی سوکت به سمت سرور ارسال می‌کند و سپس منتظر دستور بعدی می‌ماند. توجه داشته باشید که اندازه خروجی دستورات نامشخص است و ممکن است بیشتر از حجم بافر سوکت باشد.

سوال: دستور cd را از سمت سرور بر روی دستگاه قربانی اجرا کنید. آیا این دستور به صورت صحیح کار می‌کند؟ اگر خیر، چرا؟

راهنمایی (اجرای دستورات پوسته در برنامه به زبان C):

یکی از روش‌های اجرای دستورات در زبان C استفاده از تابع popen(.) است. این تابع یک کانال ارتباطی یک طرفه بین فرآیند اصلی و یک فرآیند فرعی (که در این حالت یک دستور لینوکس است) می‌سازد. این کار به فرآیند اصلی اجازه می‌دهد تا دستور را طوری اجرا کند که گویی در خط فرمان تایپ شده و از ورودی استاندارد آن دستور خوانده یا در خروجی استاندارد آن دستور بنویسد. شما می‌توانید برای پیاده‌سازی خود از دستور popen(.) یا هر دستور دیگری که این نیاز را برآورده می‌کند، استفاده کنید. برای آشنایی بیشتر با این دستور به صفحه man آن می‌توانید مراجعه کنید. قطعه کد زیر نیز یک مثال از نحوه استفاده از این دستور را نشان می‌دهد:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  FILE *fp;
  char path[1035];
  fp = popen("/bin/ls /", "r");
  if (fp == NULL) {
    printf("Failed to run command\n" );
    exit(1);
  }
  while (fgets(path, sizeof(path), fp) != NULL) {
    printf("%s", path);
  }
  pclose(fp);
  return 0;
}
```

۲.۲ دستور کار

پاسخ به سوالات مطرح شده و نوشتن برنامه‌های سرور و کلاینت که طبق توضیحات بالا عمل می‌کنند جزء موراد تحویلی این قسمت است. توجه کنید که رعایت اصول خوانایی کد و داشتن توضیحات مناسب (comment) مهم است و برای آن نمره در نظر گرفته شده‌است. در ضمن برای کامپایل کردن برنامه می‌بایست یک Makefile بنویسید و به همراه فایل‌های C ، این فایل را نیز ارسال نمایید (توضیحات تکمیلی در قسمت «شیوه تحویل» آمده است).
دانشجویان علاقه‌مند می‌توانند به عنوان یک بخش امتیازی یک نسخه جدید سرور تحت عنوان parserver.c نیز با توضیحی که در ذیل آمده است نیز تحویل دهند.
بخش امتیازی

- استفاده از مدل‌های ارتباط همزمان با بیشتر از یک کلاینت مانند Thread، Process، Thread Pool، Process Pool، Polling و پیاده‌سازی دستور sendall برای ارسال یک دستور خاص به همه دستگاه‌های قربانی.

۳ بررسی پروتکل HTTP توسط برنامه وایرشارک

برنامه وایرشارک (Wireshark) ،نرم‌افزاری متن باز برای آنالیز شبکه است که در سال ۱۹۹۸ و بوسیله Gerald Combs ساخته شد. در حال حاضر، متخصصان شبکه، امنیت و حتی هکرها برای بررسی بسته‌هایی که در شبکه در حال انتقال هستند، از این نرم افزار قدرتمند استفاده می‌کنند. ابتدا، ویدیویی که برای آشنایی با وایرشارک برای شما تهیه شده را مشاهده کرده و سپس با استفاده از این برنامه، اعمالی که در ادامه آمده را انجام داده، از آن‌ها تصویر (اسکرین‌شات) گرفته و در کنار پاسخ به سوالات در گزارش خود قرار دهید.

۱.۳ ارسال و دریافت بسته‌های HTTP

ابتدا وایرشارک را باز کرده سپس پکت های منتقل شده را ضبط کنید پس از آن این لینک را در پنجره مرورگر خود باز کنید. سپس ضبط پکت ها را متوقف کرده و به سوالات زیر پاسخ دهید:

۱. مرورگر شما از کدام نسخه HTTP استفاده می‌کند؟ سرور در پاسخی که برای مرورگر ارسال کرده از چه نسخه‌ای استفاده کرده است؟

۲. مرورگر شما چه زبان‌هایی را به عنوان زبان‌های قابل قبول به سرور ارسال می‌کند؟

۳. آدرس IP دستگاه شما چیست؟

٤. آدرس IP دستگاه مقصد چیست؟

۵. کد وضعیت برگشتی از سمت سرور به مرورگر شما چند است؟

٦. چند بایت محتوا به مرورگر شما برگردانده شده است؟

۷. درخواستی که بوسیله مرورگر ارسال کردید را با استفاده از دستور curl و از طریق ترمینال ارسال کنید. در بسته‌های ضبط شده در وایرشارک چه تفاوتی وجود دارد؟ چرا؟

۲.۳ اسناد طولانی

در مثال قبل محتوای دریافت شده ساده و کم حجم بود. بیایید ببینیم وقتی یک فایل طولانی HTML را دریافت می‌کنیم چه اتفاقی می‌افتد. برنامه وایرشارک را باز کرده سپس بسته‌ها را ضبط کنید، پس از آن، این لینک را در مرورگر خود وارد کنید، ضبط پکت ها را متوقف کنید. سپس بسته‌ها را برای پروتکل HTTP فیلتر کنید. مشاهدات خود را توضیح دهید و بگویید تفاوت این مسئله با مسئله قبلی چیست؟ در ادامه به سوالات زیر نیز پاسخ دهید:

۱. چند پیام درخواست (request) توسط مرورگر شما ارسال شده است؟

۲. کد وضعیت و عبارت در پاسخ دریافتی (response) چیست؟

۳. چند بخش TCP حاوی داده برای حمل پاسخ HTTP نیاز است؟

۳.۳ احراز هویت در HTTP

در نهایت، بیایید از وب‌سایتی بازدید کنیم که با رمز عبور محافظت می‌شود. ابتدا وایرشارک را باز کنید و بسته‌ها را ضبط کنید، سپس این این لینک را در مرورگر خود وارد کنید و نام کاربری و رمز عبور داده شده را وارد کنید. سپس ضبط بسته‌ها را متوقف کرده و بر اساس مشاهدات خود، به سوالات زیر پاسخ دهید:

Username: dm557
Password: network

۱. پاسخ سرور (کد وضعیت و عبارت) در پاسخ به پیام اولیه HTTP GET از مرورگر شما چیست؟

۲. وقتی مرورگر شما برای بار دوم پیام HTTP GET را ارسال می کند، چه فیلد جدیدی در پیام HTTP GET گنجانده شده است؟

۳. نام‌کاربری و رمز عبور ارسالی شما در پیام دیده نمی‌شوند. توضیح دهید که چه اتفاقی برای آن‌ها افتاده است؟ (راهنمایی: به محتوای سرآیند Authorization در درخواست ارسالی دقت کنید.)

٤.۳ دستور کار

پاسخ به سوالات مطرح شده به همراه تصاویر جزء موراد تحویلی این قسمت است.

٤ شیوه تحویل

برای تحویل این پروژه یک پوشه به نام ComputerNetworks_StudentID_Project1 بسازید (به جای StudentID باید شماره دانشجویی خود را قرار دهید) که شامل محتوای زیر باشد:

۱. یک فایل PDF: شامل پاسخ به سوالات هر دو قسمت و تصاویر خواسته شده در قسمت دوم.

۲. یک پوشه با نام Source که در آن کدهای C و Makefile قسمت اول پروژه قرار دارند.

در نهایت این فایل‌ها را فشرده کرده و به صورت یک فایل با نام ComputerNetworks_StudentID_Project1 و فرمت zip در سامانه یکتا ارسال کنید.