به نام خداوند بخشنده‌ی مهربان

دانشگاه صنعتی شریف
مدرس: مهدی خرازی

تمرین سری اول*
امنیّت داده و شبکه

دانشکده‌ی مهندسی کامپیوتر
خزان ۱۳۹۸

اهداف تمرین

- آشنایی با پیش‌نیازهای تمرین‌ها
- آشنایی با Stack
- آشنایی با حمله‌ی Buffer Overflow
- آشنایی با حمله‌ی Format String

۱. سرآغاز

هدف از این تمرین، آشنایی با پیش‌نیازهای مورد نیاز برای تمرین‌های این درس و به‌دست آوردن تجربه در شناسایی نقاط آسیب‌پذیر برنامه و استفاده از آن نقاط برای حمله است. در این تمرین به شما سه پرونده‌ی دودویی[1] داده شده است که در هر کدام یک یا چند آسیب‌پذیری وجود دارد. شما باید این آسیب‌پذیری‌ها را شناسایی کرده و برای هر کدام یک برنامه (با C/C++ یا python) یا script نوشته و پرونده‌ی اجرایی را به همراه کد منبع[2] آن‌ها در پاسخ به این تمرین تحویل دهید.

*با تشکر از تیم دستیاران آموزشی

[1] binary
[2] source code

۲. پیش‌نیازها

۱.۲ Gogs

تمامی تمرین‌های شما از طریق این سامانه دریافت می‌گردد. بنابراین شما به یک حساب کاربری در این سامانه نیاز دارید. برای هر یک از شما یک مخزن[۳] خصوصی و شناسه‌ی ورود به سامانه ساخته شده و اطلاعات آن در اختیارتان قرار خواهد گرفت. دستورهای زیر را در ترمینال خود اجرا کنید تا تنظیماتی که برای کامیت‌های[۴] خود استفاده می‌کنید برقرار گردد.

```
git config --global user.name "Your Username"
git config --global user.email "Your Email"
```

سپس نیاز دارید که کلیدهای ssh خود را به منظور احراز هویت از درون ماشین خود تنظیم کنید (البته می‌توانید از ارسال کلید به سامانه‌ی گیت صرف نظر کنید و برای هر بار کامیت و دست‌رسی به مخزن خود از گذرواژه‌ی خود استفاده کنید). برای این کار دو دستور زیر را به ترتیب اجرا کنید:

```
ssh-keygen -N "" -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub
```

دستور اول یک جفت کلید ssh برای شما می‌سازد. دستور دوم کلید عمومی‌تان را در ترمینال نمایش می‌دهد. می‌توانید از این قسمت کلید عمومی خود را به حساب خود اضافه کنید. برای دسترسی به پرونده‌های مورد نیاز هر تمرینِ درس یک مخزن عمومی ساخته شده است. این مخزن از آدرس زیر در دسترس است:

```
git@tarasht.ce.sharif.ir:ce441-981-students/ce441-981-handouts.git
```

پرونده‌های مورد نیاز شما در پوشه‌ی HW1 هستند. برای به‌روزرسانی این مخزن می‌توانید دستور زیر را اجرا کنید:

```
$ cd ~/ce441-981-handouts
$ git pull origin
```

برای اتصال به مخازنی که از قبل برای شما به صورت خصوصی تعریف شده کافی است به مسیر دلخواه (در این مستند فرض کرده‌ایم در مسیر home مخزن‌های گیت را بارگیری می‌کنید) رفته و دستور زیر را اجرا کنید:

```
git clone REPO-URL
```

که REPO-URL به صورت زیر است:

```
git@tarasht.ce.sharif.ir:ce441-981-students/ce441-981-"student-id".git
```

[3] repository
[4] Commit

Git .۲.۲

یک برنامه‌ی مدیریت نسخه[۵] است که به کمک آن می‌توانید روند تغییر پرونده‌ها را دنبال کنید. به عنوان مثال GitHub یکی از سامانه‌های گیت تحت وب است که امکان میزبانی پرونده‌های شما را فراهم می‌کند. در واقع این سامانه فضای تعاملی و اشتراک‌گذاری پرونده‌ها را فراهم می‌سازد. برای افزایش سطح دانش خود نسبت به گیت می‌توانید این صفحه را بخوانید.

gdb .۳.۲

امکان دیباگ کردن آسان برنامه‌ها را فراهم می‌آورد. اگر برنامه با گزینه‌ی[۶] g- کامپایل شده باشد، gdb امکان مشاهده‌ی کد منبع برنامه، ردگیری پشته، مشاهده‌ی مقدار و نشانی متغیرها، مشاهده‌ی کد هم‌گذاری[۷] برنامه، توقف برنامه در محل‌های دلخواه و ... را در اختیار شما قرار می‌دهد. برای یادگیری کار با gdb می‌توانید این مستند را بخوانید. همچنین مطالعه‌ی مستند اصلی gdb نیز مفید است.

objdump .۴.۲

اطلاعات مربوط به object file را نمایش می‌دهد. شما با استفاده از گزینه‌های آن می‌توانید نوع اطلاعاتی را که مایل به مشاهده‌ی آن‌ها در مورد هر object file هستید، مشخص نمایید.

برای آشنایی با برنامه‌ی objdump می‌توانید این صفحه را مشاهده کنید.

[5] version control
[6] option
[7] Assembly

۳. برنامه‌های آسیب‌پذیر

در پوشه‌ی questions سه برنامه‌ی اجرایی (دودویی) به همراه کد منبع آن‌ها با نام‌های prog_vuln1، prog_vuln3 و prog_vuln4 وجود دارد (برای استفاده از objdump نیاز است که پوشه‌ی questions را در مسیر /tmp/hw1/questions قرار دهید). در هر یک از این برنامه‌ها، یک یا چند آسیب‌پذیری وجود دارد که از آن برای اجرای shellcode استفاده خواهید کرد.

این برنامه‌ها برای سامانه‌ی x86 (32 bit) و بر روی سامانه‌ی پایه‌ی لینوکس[۸] ساخته‌شده‌اند، بنابراین برای اجرای این برنامه‌ها در سامانه‌ی x86-64 نیاز به نصب بسته‌های ۳۲ بیتی از مخزن لینوکس است. از پیوند ۱ و پیوند ۲ می‌توانید اطلاعات بیشتری برای نحوه‌ی اجرای برنامه‌های ۳۲ بیتی بر روی سامانه‌ی ۶۴ بیتی کسب نمایید[۹].

هدف از این تمرین اجرای Aleph one shellcode است که به‌صورت زیر می‌توانید آن را تعریف کنید:

```
static char shellcode[] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c"
"\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb"
"\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

گرفتن remote shell در این تمرین مجاز نمی‌باشد و در صورت استفاده از هر shellcodeای که منجر به یک عملیات خراب‌کارانه در هنگام آزمون و نمره‌دهی شود، نمره‌ی شما صفر خواهد شد. شما برای استفاده از آسیب‌پذیری‌های درون برنامه فقط حق استفاده از ابزارهای gdb و objdump را دارید و استفاده از ابزارهایی مانند Metasploit مجاز نیست و هیچ نمره‌ای به آن تعلق نمی‌گیرد. در همه‌ی بخش‌های این تمرین، ASLR[۱۰] خاموش است. برای خاموش کردن موقت ASLR می‌توانید از دستور

```
$ echo 0 | sudo tee /proc/sys/kernel/randomize_va_space
```

و برای روشن کردن دوباره‌ی آن می‌توانید از دستور

```
$ echo 2 | sudo tee /proc/sys/kernel/randomize_va_space
```

استفاده نمایید. توجه کنید که قبل از شروع کار لازم است که ASLR را خاموش نمایید.

[8] Linux OS

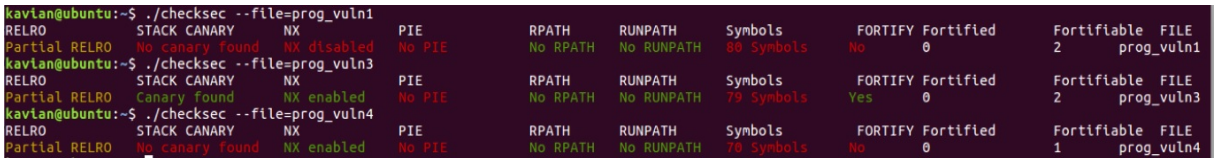
[۹] ممکن است پس از نصب بسته‌های ۳۲ بیتی بازهم مشکلاتی به وجود بیاید، به همین دلیل توصیه می‌شود که از یک ماشین ۳۲ بیتی مجازی استفاده کنید.

[10] Address Space Layout Randomization

۴. توضیحات

- برای پاسخ به این تمرین نیاز است تا با مفاهیم کلی معماری x86 و x64 (شامل: ثبات‌ها، پشته و نحوه‌ی اجرای یک برنامه و فراخوانی توابع) آشنایی داشته باشید. در صورت نیاز می‌توانید این مستند و این را مطالعه نمایید.

- سازوکارهای دفاعی[11] به کار رفته در هر برنامه، در شکل ۱ قابل مشاهده است.



شکل ۱: سازوکارهای دفاعی

- در برنامه‌ی prog_vuln1 هیچ سازوکار دفاعی وجود ندارد و باید سعی کنید از طریق رشته‌ای که وارد می‌کنید، حمله‌ی Buffer Overflow را انجام دهید.

- در برنامه‌ی prog_vuln4 سازوکار دفاعی NX[12] فعال است. این سازوکار باعث می‌شود تا محتویات درون پشته غیرقابل اجرا شوند و نتوانید همانند یک حمله‌ی Buffer Overflow ساده عمل کنید. بنابراین برای حمله به این برنامه باید به دنبال راهکار دیگری باشید. این برنامه از معماری x64 استفاده می‌کند؛ لذا به تفاوت آن با معماری x86 دقت شود. برای حل این برنامه استفاده از ابزارهایی مثل Ropper، ROPgadget و pwntools برای پیدا کردن gadgetها و متصل کردن آن‌ها مجاز است.

- در برنامه‌ی prog_vuln3 علاوه بر NX سازوکار دفاعی Canary[13] نیز فعال است. این سازوکار با قرار دادن یک مقدار تصادفی در پشته و بررسی آن هنگام بازگشت از تابع، باعث می‌شود تا در صورت سرریز بافر، این مقدار بازنویسی شده و بدین ترتیب سامانه‌ی پایه از وقوع سرریز آگاه شده و اجرای برنامه را خاتمه دهد. به همین علت امکان حمله به این برنامه همانند برنامه‌های قبل وجود ندارد. برای دور زدن این سازوکار و حمله به این برنامه می‌توان از آسیب‌پذیری Format String استفاده کرد. برای آشنایی با این آسیب‌پذیری و نحوه‌ی به‌کارگیری[14] آن می‌توانید این مستند را مطالعه نمایید.

[11] Defensive Mechanisms

[12] Non-eXecutable

[13] به آن Stack Cookie نیز گفته می‌شود.

[14] Exploitation

۵. تحویل‌دادنی‌ها

برای هر برنامه شما موظفید که یک گزارش بنویسید و در آن آسیب‌پذیری‌های موجود در برنامه را شرح دهید (شما باید تابعی را که آسیب‌پذیری دارد نام ببرید؛ مانند strcpy یا printf) و نوع آسیب‌پذیری‌ها را بیان نمایید (اگر برنامه چند آسیب‌پذیری دارد باید هر کدام را نام ببرید و بگویید که دامنه‌ی خراب‌کاری هر کدام از آسیب‌پذیری‌ها چیست). هم‌چنین می‌بایست روند کار خود تا اجرای shell را توضیح داده و از مراحل لازم، تصویر تهیه نمایید.

علاوه بر گزارش، باید سه Script (به‌همراه برنامه‌ی اجرایی و کد منبع برنامه) با نام‌های exploit1.sh، exploit3.sh و exploit4.sh را در پوشه‌ای با نام exploits تحویل دهید که بتوان مسیر برنامه‌ی *prog_vuln را به عنوان آرگومان به آن‌ها داد. درحقیقت برای تصحیح این تمرین، دستور

```
$ ./exploit1.sh /path/to/prog_vuln1
```

مورد استفاده قرار می‌گیرد. بنابراین برای گرفتن نمره‌ی این تمرین رعایت این نکات الزامی است.

۶. نکات ضروری

- توجه کنید که شما باید روی پرونده‌های باینری داده شده در این تمرین shell بگیرید و دوباره کامپایل کردن از روی کد منبع ممکن است باعث تغییر نشانی‌ها در پرونده‌ی دودویی و درنتیجه عدم موفقیت در ماشین آزمون‌گر شود.
- در صورتی‌که هر مشکل یا پرسشی داشتید که فکر می‌کنید پاسخ آن برای همه مفید خواهدبود، آن را در میلینگ لیست یا تالار گفت‌وگوی درس[۱۵] مطرح نمایید.
- از فرستادن پاسخ تمرین برای دیگران خودداری کنید.
- تمام برنامه‌ی شما باید توسط خود شما نوشته شده باشد. فرستادن کل یا قسمتی از برنامه‌تان برای افراد دیگر، یا استفاده از کل یا قسمتی از برنامه‌ی فرد دیگری، حتی با ذکر منبع، تقلب محسوب می‌شود.
- به علت اینکه بخشی از نمره‌ی تمرین به صورت خودکار داده می‌شود، ساختار پوشه‌ی تحویل داده شده باید دقیقاً به صورت گفته شده باشد. ساختار نهاییِ مخزن شما پس از انجام این تمرین به شکل زیر است:

```
--README.md
--hw1/
  --exploits/
    --exploit1.sh
    --exploit3.sh
    --exploit4.sh
--report.pdf
```

- همه‌ی پرونده‌های لازم را با همان نامی که در مستند تمرین ذکر شده است، با دستورهای زیر ارسال کنید (فرض شده مخزن خود را در مسیر home قرار داده‌اید):

```
cd ~/ce441-981-student_id/hw1
git status
git add --all
git commit -m "Finished my first assignment"
git push origin master
```

[15] http://ce441-sharif.rf.gd/