



اهداف تمرین

- آشنایی با پیش نیازهای تمرین ها
- آشنایی با Stack
- آشنایی با حمله ی Buffer Overflow
- آشنایی با حمله ی Format String

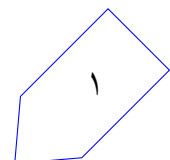
۱. مقدمه

هدف از این تمرین، آشنایی با پیش نیازهای مورد نیاز برای تمرین های این درس و به دست آوردن تجربه در شناسایی نقاط آسیب پذیر برنامه و استفاده از آن نقاط برای حمله است. در این تمرین به شما سه پرونده ی باینری^۱ داده شده است که در هر کدام یک یا چند آسیب پذیری وجود دارد. شما باید این آسیب پذیری ها را شناسایی کرده و برای هر کدام یک برنامه (با C/C++ یا python) یا script نوشته و پرونده ی اجرایی به همراه کد منبع^۲ آن ها را به عنوان پاسخ این تمرین تحویل دهید.

* با تشکر از تیم دستیاران آموزشی

¹binary

²source code



۲. پیش‌نیازها

۱.۲. Gogs

تمامی تمرین‌های شما از طریق این سامانه دریافت می‌گردد. بنابراین شما نیازمند یک حساب کاربری در این سامانه می‌باشید. 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 تولید می‌کند. دستور دوم کلید عمومی‌تان را در ترمینال نمایش می‌دهد. شما می‌بایست به این سامانه ورود کرده و سپس از این قسمت کلید عمومی خود را به حساب خود اضافه کنید. برای دسترسی به پرونده‌های مورد نیاز هر تمرین درس یک مخزن عمومی ساخته شده است. این مخزن از آدرس زیر در دسترس است:

```
git@tarasht.ce.sharif.ir:ce442-971-students/ce442-971-handouts.git
```

همه‌ی پرونده‌های مورد نیاز شما در پوشه‌ی مربوط به تمرین است. برای به‌روزرسانی این مخزن می‌توانید از این دستور استفاده کنید:

```
$ cd ~/ce424-971-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) خود رفته و دستورات زیر را اجرا کنید:

³repository

⁴Commit

```
git clone REPO-URL
```

که REPO-URL ساختاری شبیه به:

```
git@tarasht.ce.sharif.ir:ce442-971-students/ce442-971-"student-id".git
```

۲.۲. Git

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 [این‌جا](#) را مشاهده کنید.

۳.۲. gdb

امکان دیباگ کردن آسان برنامه‌ها را فراهم می‌آورد. اگر برنامه با گزینه‌ی `-g`^۶ کامپایل شده باشد، gdb امکان ردگیری پشته، مشاهده‌ی مقدار و آدرس متغیرها، مشاهده‌ی کد اسمبلی برنامه، توقف برنامه در محل‌های دلخواه و... را در اختیار شما قرار می‌دهد. برای یادگیری gdb می‌توانید [اینجا](#) را مشاهده کنید. هم‌چنین مستند اصلی gdb نیز علی‌رغم طولانی بودن مفید است.

۴.۲. objdump

اطلاعات مربوط به object file را نمایش می‌دهد. شما با استفاده از گزینه‌های آن می‌توانید نوع اطلاعاتی در مورد هر file object که مایل به مشاهده‌ی آن هستید را مشخص نمایید. برای آشنایی با objdump می‌توانید [این‌جا](#) را مشاهده کنید.

^۵version control

^۶option

۳. برنامه‌های آسیب‌پذیر

در پوشه‌ی questions سه برنامه‌ی اجرایی (باینری) به همراه کد منبع آن‌ها با نام‌های prog_vuln1، prog_vuln2 و prog_vuln3 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 عامل لینوکس ساخته شده‌اند، 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.^۷ هدف از این تمرین اجرای Aleph one shellcode است که به صورت 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\x40xcd\x80\xe8\xdc\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^۸ خاموش است. برای خاموش کردن موقت ASLR می‌توانید از دستور

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

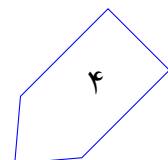
و برای روشن کردن دوباره‌ی آن می‌توانید از دستور

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

استفاده نمایید. توجه کنید که قبل از شروع کار باید حتماً ASLR را خاموش نمایید.

^۷ 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.

^۸Address Space Layout Randomization



۴. توضیحات

- برای پاسخ به این تمرین نیاز است تا با مفاهیم کلی معماری x86 (شامل: ثبات‌ها، پشته و نحوه‌ی اجرای یک برنامه و فراخوانی توابع) آشنایی داشته باشید. در صورت نیاز می‌توانید این مستند را مطالعه نمایید.
- سازوکارهای دفاعی^۹ به کار رفته در هر برنامه، در شکل ۱ قابل مشاهده است.

```
ce40442@SUT:~$ checksec --file prog_vuln1
RELRO      STACK CANARY  NX          PIE         RPATH      RUNPATH     FORTIFY Fortified Fortifiable FILE
Partial RELRO No canary found NX disabled No PIE     No RPATH   No RUNPATH No      0          0          prog_vuln1

ce40442@SUT:~$ checksec --file prog_vuln2
RELRO      STACK CANARY  NX          PIE         RPATH      RUNPATH     FORTIFY Fortified Fortifiable FILE
Partial RELRO No canary found NX enabled  No PIE     No RPATH   No RUNPATH No      0          0          prog_vuln2

ce40442@SUT:~$ checksec --file prog_vuln3
RELRO      STACK CANARY  NX          PIE         RPATH      RUNPATH     FORTIFY Fortified Fortifiable FILE
Partial RELRO Canary found   NX enabled  No PIE     No RPATH   No RUNPATH Yes     0          0          prog_vuln3
```

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_vuln2 سازوکار دفاعی NX^{۱۰} 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.
- در برنامه‌ی prog_vuln3 علاوه بر NX سازوکار دفاعی Canary^{۱۱} 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 عامل 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 سوءاستفاده از آن می‌توانید این مستند را مطالعه نمایید.

^۹defense mechanisms

^{۱۰}Non-executable

^{۱۱}به آن Stack Cookie نیز گفته می‌شود.

۵. تحویل دادنی‌ها

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، exploit2.sh و exploit3.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 source code ممکن است باعث تغییر آدرس‌ها در پرونده‌ی باینری و در نتیجه عدم موفقیت در ماشین تست شود.
 - در صورتی که هر مشکل یا پرسشی داشتید که فکر می‌کنید پاسخ آن برای همه مفید خواهد بود، آن را در فهرست پستی (میلینگ لیست) ارسال نمایید.
 - از فرستادن جواب تمرین به فهرست پستی 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 نهایی مخزن شما پس از انجام این تمرین به شکل زیر است:
- ```
--README.md
--hw1/
 --exploits/
 --exploit1
 --exploit2
 --exploit3
--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 ~/ce442-961-student_id/hw1
git status
git add - -all
git commit -m "Finished my first assignment"
git push origin master
```

