Цели и задачи лабораторной работы

Цель лабораторной работы: изучение способов эксплуатации уязвимостей переполнения буфера на стеке и связанных с ними уязвимостей форматной строки.

Задание на лабораторную работу: добиться вывода на экран надписи “Access granted”.

1 Теоретические сведения

См. лекцию об уязвимостях.

2 Поиск и эксплуатация ROP-цепочек

2.1 Утилита ropper

Для поиска гаджетов используются утилиты наподобие Ropper (https://github.com/sashs/Ropper/tree/master) и RopGadget. Данные утилиты способны как осуществлять поиск гаджетов, так и составлять полноценные ROP-цепочки. Далее будет рассмотрена утилита Ropper.

Входными данными для анализа является исполняемый файл/разделяемая библиотека, в котором будет осуществляться поиск (флаг -f). Поскольку в самом исполняемом файле достаточного количества гаджетов может не оказаться, поиск гаджетов может проводиться в библиотеках, которые использует исполняемый файл. Для отображения списка используемых библиотек используется утилита ldd (использование: *ldd <исполняемый файл>*).

Помимо простого поиска гаджетов утилита может осуществлять автоматического составления ROP-цепочки с генерацией заготовки Python-скрипта, выводящего строку, пригодную для подачи в атакуемое приложение.

Ropper генерирует только заготовку скрипта, поскольку утилита не знает 1) размер буфера, который необходимо переполнить до того, как будет перезаписан адрес возврата; 2) адрес загрузки библиотеки.

Адреса загрузки библиотек можно узнать в gdb с помощью команды *info proc mappings* (программа уже должна быть запущена). Известный адрес загрузки библиотеки вставляется в скрипт либо передается в Ropper с помощью аргумента *-I <адрес>*.

Для преодоления буфера перед адресом возврата наиболее простым вариантом является модификация скрипта (см. переменную *rop*, которая изначально равна пустой строке).

2.2 Эксплуатация ROP

Ropper может генерировать только ROP-цепочки для вызовов execve и mprotect. Первый вызов запускает на выполнение другую программу. Второй вызов может использоваться для изменения разрешений для области памяти (позволяет снять заперт на исполнение кода в стеке/куче).

Поскольку для решения задачи Л/Р достаточно добиться вывода “Access granted”, вместо получения экземпляра оболочки /bin/sh (что обычно и является целью shell-кодов, но требует дополнительных действий) можно выполнить скрипт, выводящий требуемую строку. В этом случае, для построения ROP-цепочки нужно передать в Ropper аргумент *--chain “execve cmd=<путь к скрипту>"*. Для упрощения путь к скрипту следует сделать кратным 8 байтам.

Скриптом с точки зрения Unix является текстовый файл, начинающийся со строки вида *#!<путь к интерпретатору>*. Ниже приведен текст скрипта Python3, который ничего не выводит:

```
#!/usr/bin/python3
1+1
```

Поскольку скрипт должен иметь разрешения на выполнение, для файла скрипта необходимо выполнить команду *chmod u+x <путь к скрипту>*.

Итоговый порядок выполнения работы:

1. Написать скрипт на Python/Bash, который выводит “Access granted”.
2. Разрешить выполнение скрипта с помощью chmod.
3. Определить загружаемые библиотеки с помощью ldd, выбрать целевую библиотеку.
4. Найти адрес загрузки библиотеки с помощью gdb.
5. Сгенерировать скрипт генерации ROP-цепочки с помощью Ropper.
6. Поправить скрипт генерации с учетом размера переполняемого буфера.
7. Сгенерировать строку и передать ее на вход программы.

Примечание: в целом, можно получить экземпляр оболочки /bin/sh, но она требует наличия открытого потока ввода, через который будут считываться команды. Если вы перенаправите поток ввода из файла, вводить команды вы не сможете. Есть обходной путь – перенаправить вывод из другой команды, но вот какой – можете подумать сами.