Отчет по пятой домашней работе по дисциплине «Бинарные уязвимости».

Хаукка Станислав Игоревич

Задание:

Найти уязвимость в программе и разработать эксплойт, который бы позволил вызвать функцию system_priv(), которая по легенде предоставляет доступ к системе с наивысшими привилегиями. Функция была доступна при разработке программы и сейчас содержится в ytq, но для релизной версии разработчик решил, что пользователи не должны иметь к ней доступ и добавил проверку.

```
$ cat prog12b.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
struct data {
  char pass[8];
  int user_id;
  int (*get_shell)(unsigned short int, char *);
};
void user_priv(void);
void admin_priv(void);
void system_priv(void);
void get_shell(unsigned short int, char *);

void get_shell(unsigned short int user_id, char *pass) {
  if (user_id == 128) {
    system_priv();
  }
    ```

    ```
  if (user_id == 256 && !strcmp("qwerty", pass)) {
    user_priv();
  }
 if (user_id == 512 && !strcmp("asdfgh", pass)) {
    admin_priv();
  }
}
void user_priv(void) {
  printf("ACCESS GRANTED. You have shell with USER privileges.\n");
}
void admin_priv(void) {
 printf("ACCESS GRANTED. You have shell with ADMIN privileges.\n");
}
void system_priv(void) {
  printf("ACCESS GRANTED. You have shell with SYSTEM privileges.\n");
}
og
void main(int argc, char *argv[]) {
  if (argc != 3) {
    printf("USAGE: ./prog12 <user_id> <password>\n");
    return;
  }
  if (atoi(argv[1]) < 256) {
    printf("ERROR: user id must be more than 256.\n");
    return;
  }
  struct data *auth;
  auth = malloc(sizeof(struct data));

  auth->user_id = atoi(argv[1]);
```

```
  auth->get_shell = get_shell;
  strcpy(auth->pass, argv[2]);

  auth->get_shell(auth->user_id, auth->pass);

  free(auth);
}
```

Проанализировав код, понял

1)необходимо сделать user_id=128 для получения привлегий, но проверка больше 256 не должна нас пускать.

2) аргумент с id пользователя принимется main как текст, затем atoi преобразует его в int(4 байта) и попадает на проверку в get_shell как unsigned short int(2 байта).

Тогда решение выглядит примерно так:

128 в двоичной системе 1000 0000 – это 1 байт – не пропустит проверка больше 256.

Тогда добавим нулей до 2 байт необходимых в short int:

1 0000 0000 1000 0000 = 65664.