

Zadanie 6: Zraniteľnosť programov

Cieľom šiesteho zadania je oboznámiť sa s problematikou zraniteľnosti aplikácií a možnosťami detekcie daných zraniteľností.

Buffer overflow zraniteľnosť

Prvý príklad zraniteľnosti som si vybral samozrejme buffer overflow.

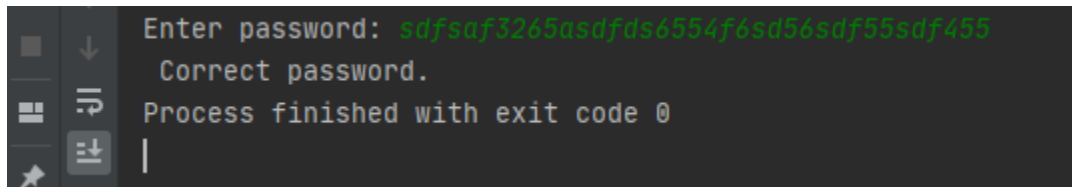
```
static void vulnerable_function() {
    int password_check = 0;
    char password_buf[MAX_PASSWORD_LENGTH] = {0};

    printf("Enter password: ");
    gets(password_buf);

    if (strcmp(password_buf, DEFAULT_PASSWORD_PLAIN) == 0)
        password_check = 1;

    if(password_check)
        printf("Correct password.");
    else
        printf("Incorrect password.");
}
```

Po zadani dostatočne dlhého hesla (input string) pretečie premena `password_buf`. Týmto sa prepíše miesto v pamäti kde je premenná `password_check` a teda podmienka a kontrola bude vždy správna keď hodnota bude iná ako nula.



```
Enter password: sdfsqf3265asdfs6554f6sd56sdf55sdf455
Correct password.
Process finished with exit code 0
```

Buffer overflow ošetrenie

Hlavným problémom bolo použitie funkcie `gets`, ktorá nekontroluje dĺžku vstupu. Funkcia `fgets` rieši tento nedostatok.

```
static void secure_function() {
    char password_buf[MAX_PASSWORD_LENGTH] = {0};

    printf("Enter password: ");
    fgets(password_buf, MAX_PASSWORD_LENGTH, stdin);

    if (strncmp(password_buf, DEFAULT_PASSWORD_PLAIN, MAX_PASSWORD_LENGTH) ==
0)
        printf("Correct password.");
    else
        printf("Incorrect password.");
}
```

Po osetreni vidime, ze heslo je spravne detekovane ako nespravne.

```
Enter password:sdfsaf3265asdfs6554f6sd56sdf55sdf455
Incorrect password.
Process finished with exit code 0
```

Format string zraniteľnosť

Jazyk C ma zranitelnost pri formatovani stdout cez printf, utocnik moze zadat formaty ako 5p alebo 5n ktorym si necha vypisat call stack pamatove hodnoty. Vďaka čomu môže neskôr exekutovať práve podľa týchto hodnôt a volat funkcie a instrukcie programu.

```
static void vulnerable_function() {
    char user_input[MAX_INPUT_LENGTH] = {0};

    printf("Input: ");
    fgets(user_input, MAX_INPUT_LENGTH, stdin);

    printf(user_input);
}
```

```
Input: %p %P %p %p %p
0000000000000000 P 00007FFD372C5940 0000000000061FC582520502520702520
Process finished with exit code 0
```

Format string ošetrovanie

Riesenie pouzije spravne fprintf funkcie pomocou %s .

```
static void secure_function(){
    char user_input[MAX_INPUT_LENGTH] = {0};

    printf("Input: ");
    fgets(user_input, MAX_INPUT_LENGTH, stdin);

    printf("%s", user_input);
}
```

Po ošetrovaní sa zadávaný input neprejaví ako formát, ale ako hodnota charakterov.

```
Input:%p %p %p %p %p
%p %p %p %p %p
Process finished with exit code 0
```

Sprintf zraniteľnosť

Sprintf je nebezpečná funkcia v jazyku C.

```
static void vulnerable_function() {
    char buffer[BUFFER_SIZE];
    int check = 0;

    sprintf(buffer, "%s", "This string is too long!");

    printf("check: %d", check);
}
```

Nastalo pretečenie a tam, kde by mala byť hodnota 0 je 1936269415.

```
check: 1936269415
Process finished with exit code -1073741819 (0xC0000005)
```

Sprintf ošetrenie

Lepšie je používať snprintf.

```
static void secure_function() {
    char buffer[BUFFER_SIZE];

    int length = snprintf(buffer, BUFFER_SIZE, "%s%s", "long-name",
"suffix");

    if (length >= BUFFER_SIZE) {
        printf("String truncation!");
    }
}
```

File opening zraniteľnosť

Ak nekontrolujeme či file existuje, útočník môže vytvoriť symbol link, napríklad na subor s heslami.

```
static int vulnerable_function() {
    if (!access(MY_TMP_FILE, F_OK)) {
        printf("File exists!\n");
        return EXIT_FAILURE;
    }
    /* At this point the attacker creates a symlink from /tmp/file.tmp to
    /etc/passwd */
    FILE * tmpFile = fopen(MY_TMP_FILE, "w");
    if (tmpFile == NULL) {
        return EXIT_FAILURE;
    }
    fputs("Some text...\n", tmpFile);
    fclose(tmpFile);

    return 0;
}
```

File opening zraniteľnosť

Kontrolovať či file existuje.

```
static int secure_function() {
    int fd;
    FILE* f;

    /* Odstrániť možné symlinks */
    unlink(MY_TMP_FILE);

    fd = open(MY_TMP_FILE, O_WRONLY|O_CREAT|O_EXCL, FILE_MODE);
    if (fd == -1) {
        perror("Failed to open the file");
        return EXIT_FAILURE;
    }

    f = fdopen(fd, "w");
    if (f == NULL) {
        perror("Failed to associate file descriptor with a stream");
        return EXIT_FAILURE;
    }
    fprintf(f, "Hello, world\n");
    fclose(f);

    return EXIT_SUCCESS;
}
```