```c
int main(int argc, char **argv, char **envp) {
    gid_t gid;
    uid_t uid;
    gid = getegid();
    uid = geteuid();
    setresgid(gid, gid, gid);
    setresuid(uid, uid, uid);
    char *b_1;
    b_1 = NULL;
    asprintf(&b_1, "echo %s", argv[1]);
    system(b_1);
}
```

El programa corre con privilegios especiales por el SetUID. Además en la linea 11 system(b_1), se ejecutará el comando dado por el buffer b_1 con esos permisos. Nosotros tenemos total libertad de poner lo que querramos en el buffer, ya que el mismo toma la entrada por consola y la guarda en el buffer luego de un *"echo "*. Un posible ataque sería llamar al programa de la siguiente manera.

```
./main Usted ha sido atacado por el gato; /bin/bash
```

De esta manera el system() ejecutará primero el echo avisandole al programa que los días felices se acabaron y luego abrirá una terminal en la que tendremos los nuevos privilegios obtenidos con el SetUID. Desde esa terminal podemos hacer lo que querramos.

Desde la consola el comando se vería de la siguiente manera:

```
echo "Usted ha sido atacado por el gato"; /bin/bash
```

Una solución que hará que el comportamiento del programa sea el mismo, pero sin la vulnerabilidad podría ser la siguiente:

```
int main(int argc, char **argv, char **envp) {
    gid_t gid;
    uid_t uid;
    gid = getegid();
    uid = geteuid();
    setresgid(gid, gid, gid);
    setresuid(uid, uid, uid);
    char *b_1;
    b_1 = NULL;
    asprintf(&b_1, "%s", argv[1]);
    exec("/bin/echo", b_1);
}
```

De esta manera estamos forzando a ejecutar echo con b_1 como argumento lo cual si b_1 = "Usted ha sido atacado por el gato; /bin/bash" en la consola sería algo así:

```
/bin/echo "Usted ha sido atacado por el gato; /bin/bash"
```

Esa entrada es tomada en su totalidad como un string que quiere ser impreso. Por lo que el ";" no tiene ningun efecto.