Language-Based Security DD2525

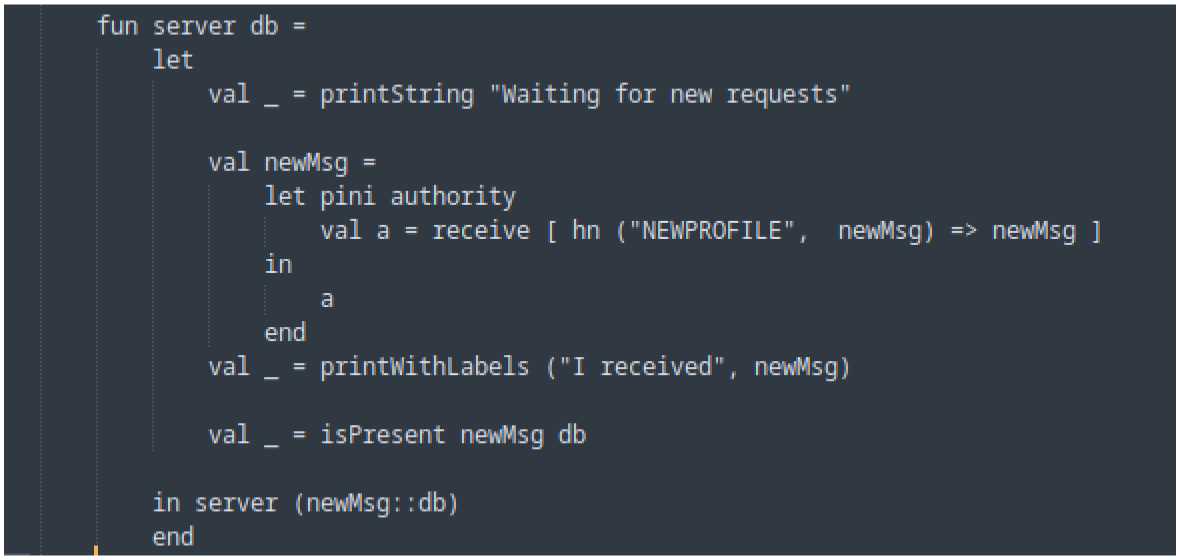
Emmanouil Dimosthenis Vasilakis - edvas@kth.se
Akkogiounoglou Georgios - gakk@kth.se

Assignment

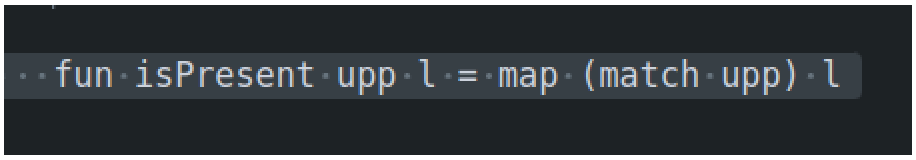
Lab Assignment 1: Information Flow Control

1. Source code:

dating-server.trp: The implementation of the dating server which includes the matching function of the profiles. It listens for new user profiles, processes them to find matches, and updates the database with each new profile received. The server calls itself recursively with the updated database. We define a newMsg and use the let pini authority which blocks the program and waits for a "NEWPROFILE" message. After receiving the message, we return it in the "in" part and then call the isPresent function by providing as the first argument the newMsg and as the second argument the database.



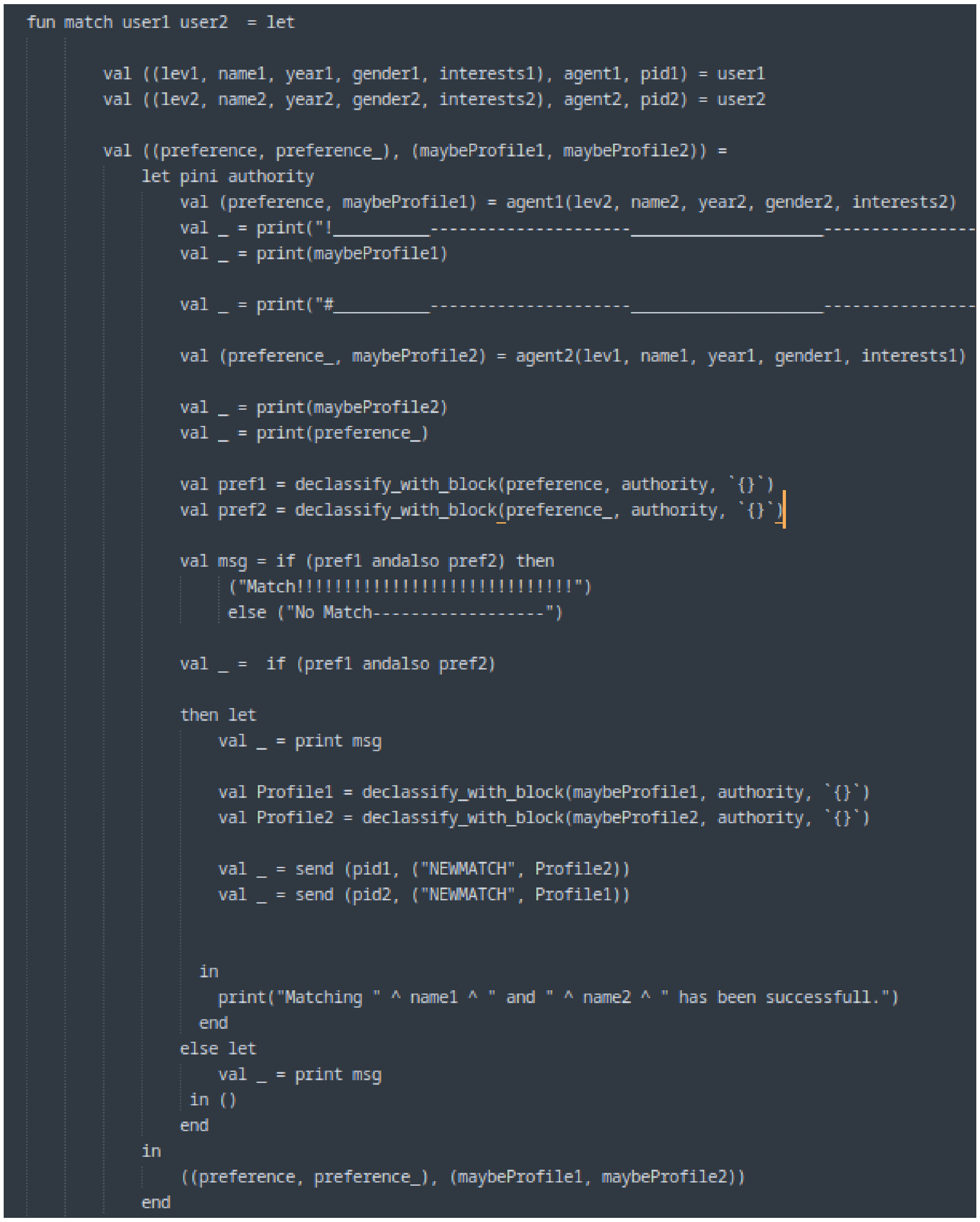
```
fun server db =
    let
        val _ = printString "Waiting for new requests"

        val newMsg =
            let pini authority
                val a = receive [ hn ("NEWPROFILE",  newMsg) => newMsg ]
            in
                a
            end
        val _ = printWithLabels ("I received", newMsg)

        val _ = isPresent newMsg db

    in server (newMsg::db)
    end
```



```
  fun isPresent upp l = map (match upp) l
```

This function is Present is checking the new profile with all the other profiles in the database in order to find matches.

First step was to use destructuring to get all the variables and the agent function from the user1 and user2. We call the agent1 function for the user2 and the agent2 function for the user one following the assignment’s information. After storing the preferences and the maybeProfiles that the agents return, we are using declassification since these information are considered highly sensitive secrets and must be protected with security labels. We are removing these labels with the function declassify_with_block by providing the authority. If the preferences match, we send the corresponding profiles with the send() function to the users.

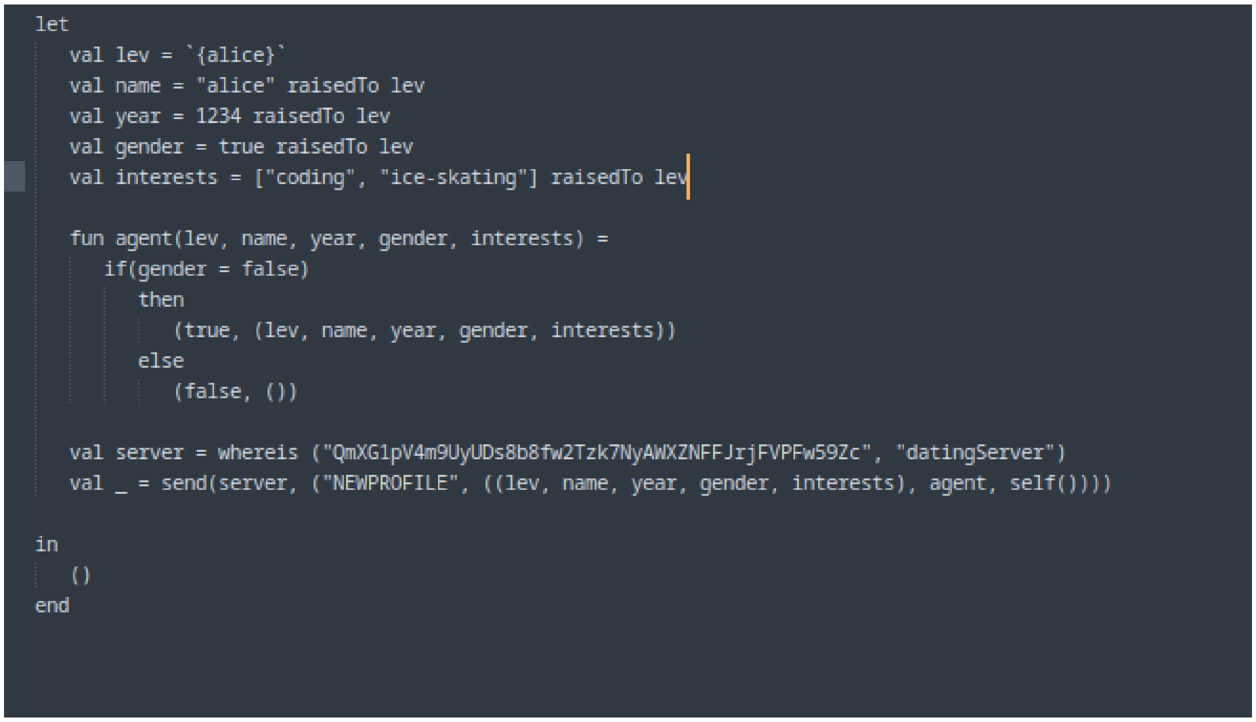


```
fun match user1 user2  = let

        val ((lev1, name1, year1, gender1, interests1), agent1, pid1) = user1
        val ((lev2, name2, year2, gender2, interests2), agent2, pid2) = user2

        val ((preference, preference_), (maybeProfile1, maybeProfile2)) =
            let pini authority
                val (preference, maybeProfile1) = agent1(lev2, name2, year2, gender2, interests2)
                val _ = print("!__________--------------------____________________---------------
                val _ = print(maybeProfile1)

                val _ = print("#__________--------------------____________________---------------

                val (preference_, maybeProfile2) = agent2(lev1, name1, year1, gender1, interests1)

                val _ = print(maybeProfile2)
                val _ = print(preference_)

                val pref1 = declassify_with_block(preference, authority, `{}`)
                val pref2 = declassify_with_block(preference_, authority, `{}`)

                val msg = if (pref1 andalso pref2) then
                     ("Match!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
                     else ("No Match-----------------")

                val _ =  if (pref1 andalso pref2)

                then let
                    val _ = print msg

                    val Profile1 = declassify_with_block(maybeProfile1, authority, `{}`)
                    val Profile2 = declassify_with_block(maybeProfile2, authority, `{}`)

                    val _ = send (pid1, ("NEWMATCH", Profile2))
                    val _ = send (pid2, ("NEWMATCH", Profile1))


                  in
                    print("Matching " ^ name1 ^ " and " ^ name2 ^ " has been successfull.")
                  end
                else let
                    val _ = print msg
                 in ()
                end
            in
                ((preference, preference_), (maybeProfile1, maybeProfile2))
            end
```

Dating-client-alice.trp : Initialization of the client Alice and communication with the server.

Dating-client-bob.trp : Initialization of the client Bob and communication with the server.

Both of them are using the agent function which returns the boolean preference and the profile. In addition they match with the opposite gender. Then it sends the message to the server.



```
let
   val lev = `{alice}`
   val name = "alice" raisedTo lev
   val year = 1234 raisedTo lev
   val gender = true raisedTo lev
   val interests = ["coding", "ice-skating"] raisedTo lev

   fun agent(lev, name, year, gender, interests) =
      if(gender = false)
         then
            (true, (lev, name, year, gender, interests))
         else
            (false, ())

   val server = whereis ("QmXG1pV4m9UyUDs8b8fw2Tzk7NyAWXZNFFJrjFVPFw59Zc", "datingServer")
   val _ = send(server, ("NEWPROFILE", ((lev, name, year, gender, interests), agent, self())))

in
   ()
end
```

2. Our approach was based on the examples that were given during the Troupe lecture and mainly on the password matching exercise. The security labels were only declassified with authority as described above.

3. The implementation of the server was mutual for both of us. We collaborated continuously to understand and make small progress with the implementation every time. Client Alice was made by Emmanouil and client Bob from Giorgos.