IFC Lab

This report details the implementation of an online dating service designed in Troupe, which enables users to finely control the visibility of their profiles through custom-written discovery agents. The service allows users to specify, via these agents, how much of their profile information is accessible to others, ensuring precise security management and controlled declassification within the system.

Design of Server

The server is used for receiving and storing user profiles, executing discovery agents to evaluate matches, and communicating potential matches back to the clients.

When the server starts, it initialises an empty database (db) to store user profiles. Each profile includes user-specific data including level, name, year of birth, gender, and interests.
When receiving a new profile, the server uses the accompanying discovery agent to assess compatibility with existing profiles in the database. The agent function takes another user's profile as input and decides whether there is a match based on custom logic provided by the user. We use the ‘map’ function here to run all the clients to match.

After determining matches, the server communicates these to the respective clients. If both parties' agents agree to a match, the server sends a "NEWMATCH" message to each client. This message contains the profile information that the other party has allowed to be visible according to the security policies enforced by their discovery agents.

Design of Client

Clients create their profiles with security labels that reflect their privacy preferences. They then send these profiles to the server along with a discovery agent and their process identifier (pid).

After submitting their profiles, clients enter a loop where they continuously listen for "NEWMATCH" messages from the server, indicating successful matches.

Security Implementation

In 'RunAgent' Function part, we set 'pini authority' to both of the input matches. Otherwise, due to the two profiles, the current blocking level will be {client1,client2}, which is higher than target level {client1} or {client2}.

```
fun Runagent(dbprofile,dbagent,dbpid) =let

                val dbmatch = let pini authority val dbmatch =
                            dbagent profile
                            in dbmatch
                            end

                val match = let pini authority val match =
                            agent dbprofile
                            in match
                            end

    in (match, dbmatch, pid, dbpid)
    end
```

In 'SendNewMatch' function part, there are two places that also use security labels . One is 'bool'. 'Bool' is used as a condition in an if statement, and the label should be less than the current blocking level, so use declassify to the lowest label{}.

```
fun Sendnewmatch(match,dbmatch,pid,dbpid) = let

    val (bool1,profile1) = declassifydeep(match, authority, `{}`)
    val (bool2,profile2) = declassifydeep(dbmatch, authority, `{}`)
    (*>> The provided target blocking level is lower than the current pc
       | the current pc: {client1,client2}
       | target blocking level: {client1}
    *)

    in if bool1 andalso bool2 then
           let
                val _ = printString "Match found\n"
                val (lev1, name1, year1, gender1, interests1) = profile1
                val (lev2, name2, year2, gender2, interests2) = profile2
                val profile1 = let pini authority
                val (_, profile1) = declassifydeep(match, authority, lev2)
                in profile1
                end
                val profile2 = let pini authority
                val (_, profile2) = declassifydeep(dbmatch, authority, lev1)
                in profile2
                end
                val _ = send (dbpid, ("NEWMATCH", profile1))
                val _ = send (pid, ("NEWMATCH", profile2))
            in ()
            end
```

In addition, when the match is successful, the label of the passed match needs to be lowered to the other label {client1} or {client2}, so that the other has the access.

Solution

By making the server and clients, the terminals of each client display that they have received the other's profile with their own label, while the server display 'match found' after receiving two profile.

```
root@86ecea58dd89:/code/code/dating# make dating-client1
/Troupe/bin/troupec dating-client1.trp -o ./out/dating-client1.js
node /Troupe/rt/built/troupe.js -f=./out/dating-client1.js --id=../ids/id-client1.json --trustmap=../trustmaps/dating-client1.json --aliases=aliases.jso
n
>>> Main thread finished with value: ffb7201e-06c4-427c-b9ca-56afbb22b772@{}%{}
"Profile and agent sent to the server."
"Waiting for match response..."
("Match response received:"@{}%{}, ({client2}@{client1}%{client1}, "client2"@{client1}%{client1}, 1992@{client1}%{client1}, false@{client1}%{client1}, [
"reading"@{client1}%{client1}, "traveling"@{client1}%{client1}]@{client1}%{client1})@{client1}%{client1})@{}%{}
"Waiting for match response..."
```

Terminal of client1

```
root@86ecea58dd89:/code/code/dating# make dating-client2
/Troupe/bin/troupec dating-client2.trp -o ./out/dating-client2.js
node /Troupe/rt/built/troupe.js -f=./out/dating-client2.js --id=../ids/id-client2.json --trustmap=../trustmaps/dating-client2.json --aliases=aliases.jso
n
>>> Main thread finished with value: 501ce80a-32ee-4622-bfed-ee918e3d09fb@{}%{}
"Profile and agent sent to the server."
"Waiting for match response..."
("Match response received:"@{}%{}, ({client1}@{client2}%{client2}, "client1"@{client2}%{client2}, 1992@{client2}%{client2}, true@{client2}%{client2}, ["
reading"@{client2}%{client2}, "traveling"@{client2}%{client2}]@{client2}%{client2})@{client2}%{client2})@{}%{}
"Waiting for match response..."
```

Terminal of client2

```
root@86ecea58dd89:/code/code/dating# make dating-server
/Troupe/bin/troupec dating-server.trp -o ./out/dating-server.js
node /Troupe/rt/built/troupe.js -f=./out/dating-server.js --id=../ids/id-server.json --trustmap=../trustmaps/dat
ing-server.json --aliases=aliases.json
Running node with identifier: QmTGjANw6zEhJthd8UBAynr6XYLmN5E5rKWToVb4Nw9Jp9
>>> Main thread finished with value: ()@{}%{}
Server started

New profile received

Server started

New profile received

Match found

Server started
```

Terminal of server

Malicious Client

We implemented client3 as our malicious client. Its working principle is as follows: at the beginning, it works the same as a benign client. It sends a normal profile to the server and then receives the match profile from the server. However, the malicious client will send the match profile to the server as long as it receives the match profile. In this case, the malicious client can send as much genuine data as possible and obtain almost all of the match profiles. A notable point is that when the malicious client receives a profile, it should declassify it to {}, replace lev to client3 and then raise it to client3. So that it can be sent correctly.