Report homework assignment 1

Carlo Leo 546155

The presented function language has been equipped with security permissions adding four abstracted operations regarding socket where each operation requires a permission to be executed by a function. Mapping is being shown in the following table:

Abstracted Operation	**Permission Required**
OpenSocket	Create
SendData	Write
ReadData	Read
CloseSocket	Destroy

Permissions are being represented by defining a new specific algebraic type *(type perm)*, instead each operation was added to language extending the syntax by means of a constructor. In addition to that, to allow the definition of a function together with its permissions set (static protection domain), a permissions list parameter was added to the *Fun constructor*. Since the language follows scope static rules, in order to have function permissions at run time, when an expression which corresponds to a function definition is being evaluated, to *closure* was added a permissions list field which corresponds to that declared.

Permissions are being checked by stack inspection algorithm so we need to keep track of all active functions in a thread during its execution.
Due the language uses scoping static rules to resolve unknown names within a function when it is being called, we cannot use the same environment to check permissions, because it represents the environment at declaration time adding actual function parameters whereas dynamic protection domain depends on execution flow.

So a new type *(permsStack = (perm list) list)* has been defined to keep track of all active functions permissions. Before a body of a function will be evaluated, its permissions list is pushed on it creating a new one. Then, if the function performs some operation which requires a permission P , eval will perform them only if all elements of the new one permsStack instance contain P by invoking checkPerms which computes whether P belongs to their intersection. When the body evaluation ends, eval will keep evaluating the rest of the program using the previous instance of the permsStack.
To do so also the eval function was changed by adding secEnv : permsStack parameter.