LANGUAGED-BASED TECHNOLOGY FOR SECURITY
Homework Assignment 1

Due: April 9 2021

In this homework, we will extend the interpreter of the simple functional language into an interpreter of a functional language which includes a dynamic mechanism for enforcing access control checks, along the line of the so-called *stack inspection* algorithm.

Learning objectives

Goals of the homework are

- to appreciate - if it was nott clear from the lectures – the main ideas of stack inspection;
- to understand the capabilities and the limitations of stack inspection;
- to understand the trade-offs in the design and implementation of stack inspection.

The assignment

In this homework you will build an interpreter a simple functional language equipped with security permissions. Each function definition is equipped with a set of permissions (e.g. {read, write}, {}) over a set of security relevant actions. We also assume that the language is equipped with a primitive construct to check a permission. The interpreter executes if permissions are enabled; otherwise, execution fails.

dobbiamo estendere questo linguaggio per affrontare il problema della stack inspection che vuol dire che ogni funzione è caratterizzata da un insieme di permessi. Ad esempio possiamo avere una funzione che può non avere permessi, quindi non può eseguire alcuna operazione sulle risorse oppure può eseguire entrambe le operazioni di lettura e scrittura o può ad esempio eseguire soltanto un'operazione di lettura. Questo è in analogia con la visione che abbiamo detto quando abbiamo presentato la stack inspection, che i permessi hanno una semantica a insiemi cioè quali sono l'insieme delle operazioni disponibili ad un dato istante. Assumiamo che non solo ho i permessi a livello del linguaggio, la possibilità di definire funzioni con diversi livelli di permessi e poi posso avere la possibilità di andare a controllare, di mettere delle guardie che sono concettualmente simili a quelle che abbiamo visto nel caso di Java qui supponiamo di avere un operazione di check che va a controllare a livello del linguaggio va a controllare se è possibile garantire quel permesso che è il parametro del check. A questo punto l'idea è che l'interprete esegue le funzionalità se la funzione ha diritto di accedere alle risorse altrimenti fallisce. Le operazioni privilegiate solo opzionali, se lo volete fare le fate, ma non sono obbligatorie, quindi obbligatorio è soltanto la check enabling e disabling sono opzionale se voi avete capito e volete implementarle sì, però non sono obbligate quindi tutto il discorso delle dell'operazione privilegiate che abbiamo visto nel caso di java non è obbligatorio, è opzionale quindi lo potete fare ma non è richiesto. Il costrutto primitivo, per fare i controlli sui permessi è inteso come costrutto sintattico, quindi esattamente come quello che succede in java, che una guardia si chiama demand permission, potete chiamarla come vi pare.

Dobbiamo seguire un approccio simile all'inline del codice fanno esattamente vedere le stesse caratteristiche, se vi ricordate nell'inline del codice anche lì la prima cosa che dovevamo fare dovevamo capire quali erano le azioni rilevanti che avevano a che vedere con la sicurezza, perché il il meccanismo di inline del security automata dipendeva dalle azioni che erano disponibili, per cui sicuramente bisogna partire da quell idea che abbiamo visto in quell esercitazione in modo particolare, questo mi permette di rispondere alla prima domanda: come partireste? La prima cosa che dovete fare è partire dalla sintassi, definire quindi la sintassi del linguaggio, comprendere come nella sintassi del linguaggio possono essere messi i domini di protezione, ma per fare questo dovete capire qual'è l'insieme delle azioni che assumete nel linguaggio, che sono rilevanti per la sicurezza e ovviamente potete astrarre nel senso possono essere azioni di lettura, di scrittura, di apertura quindi simulando come se avessi sotto un filesystem, evidente che non vi stiamo chiedendo di implementare poi il fail system ma vi stiamo chiedendo di avere un insieme di operazioni rilevanti per la sicurezza e i permessi hanno a che vedere con quelle azioni, e quindi simile esattamente al meccanismo di inlining del codice. Quindi potremmo implementare delle primitive bulk, cioè vuote, cioè solamente a titolo di esempio. Il punto di partenza sono le operazioni primitive per dare la sintassi e poi a questo punto, quando definiamo una funzione, dobbiamo anche indicare i permessi. Vuol dire che il meccanismo di dichiarazione di una funzione è un po' più articolato rispetto a quello che è è presente sia nel linguaggio che abbiamo visto a lezione sia nel meccanismo del inline che abbiamo studiato nell esercitazione, esattamente quindi, quando dichiarava una funzione, una funzione ha associato anche i relativi permessi. Quindi dovete definire i permessi, le funzioni, le operazioni, una volta fatto questo il passo successivo e questa è tutta sintassi, quindi vuol dire che definire tutta la sintassi in termini della simulazione all'interno di ocaml di queste caratteristiche, una volta fatto questo, dovete definire quali sono i valori perché siete in un linguaggio funzionale e quindi ogni esecuzione definisce un valore e dovete definire quali sono i valori una volta definito quali sono i valori, dovete comprendere quali sono le strutture a run time che devono essere presenti per supportare l'esecuzione quindi l'interprete del linguaggio.

Quando voi definite la sintassi dovete anche estendere il linguaggio col costrutto sintattico per andare a controllare i permessi, se volete implementare anche le operazioni privilegiate, nella sintassi dovete prevedere l'operazione di abilitazione di un permesso e disabilitazione di un permesso perché sono tutte scelte che poi avranno poi a che vedere con l'esecuzione dei programmi scritti, quindi dovete fare la definizione della struttura sintattica in modo coerente con quello che avete scelto per l'implementazione del linguaggio e quindi per la struttura del run time relativa. Il punto di partenza è il codice che ho dato io e il codice della esercitazione. Probabilmente crerei un ulteriore struttura che mi simula un po' lo stack del set di permissions delle varie funzioni. Concettualmente, vuol dire che l'interprete usa l'ambiente come come stack per i legami più un'altra struttura che ha a che vedere con i permessi. Sostanzialmente vuol dire che l'interprete deve avere un security manager da invocare. che fa parte del real time. Adesso quello che uno deve capire è come questo security monitor interagisce con tutte le strutture del linguaggio, perché noi siamo in un linguaggio di programmazione che ha scoping statico quindi dovete a questo punto andare a vedere quali sono le caratteristiche del linguaggio che si amalgano bene con la scelta di avere la stack inspection e quali altre caratteristiche non si amalgano bene. La sperimentazione che vi sto chiedendo di fare è: noi abbiamo un linguaggio di programmazione, ci vogliamo aggiungere un meccanismo per controllare i permessi, il linguaggio però di programmazione aveva già delle caratteristiche, quindi che cosa vuol dire? Vuol dire che aveva già una nozione di esecuzione, allora dobbiamo fare in modo di introdurre un nuovo aspetto che però si deve integrare con le caratteristiche esistenti. Vuol dire che se i permessi sono tutti ok, il risultato dell'esecuzione del programma con i permessi tutti ok è esattamente quello che uno avrebbe avuto senza instrumentare con i permessi. Vuol dire che le caratteristiche del linguaggio devono essere rispettate anche dall'aspetto aggiuntivo dei vari controlli. L'aspetto aggiuntivo di controllo degli accessi che viene messo evita di eseguire delle operazioni quando non ci sono i permessi per farlo. altra domanda Quando definiamo una funzione dobbiamo anche indicare i permessi e quindi dovendo indicare i permessi dovete avere la l'aspetto linguistico di definizione dei permessi a livello del linguaggio e poi sotto come sono implementate, Sintatticamente quando definisco una porzione gli devo dire qual è l'insieme dei permessi e notate vi sto già dicendo: usate una semantica a insiemi per i permessi e quindi vi sto già dando un po' di informazione su come deve essere, poi l'altra parte, cioè come questi permessi devono essere presenti nel run time. Quando un utente definisce un permesso, dobbiamo controllare che il permesso sia sensato, cioè corrisponde a una determinata ontologia? No Significa che dovrei avere un meccanismo strutturale per definirmi i permessi e questo deriva dal fatto che, ad esempio, nel caso di Java i permessi hanno una struttura che vive all'interno della gerarchia di Java cioè derivano da delle classi e sono organizzati a livello della gerarchia, quindi dice, se noi usiamo la stessa analogia di java, dobbiamo avere una struttura di permessi più articolata, un albero, un grafo o quello che sia.