Exam 2 Report

Alfred Murabito

March 1, 2020

Abstract

This report answers the three questions in the CIS 634 Exam 2. The first two questions are answered with the code in the HOL/exam2Script.sml source file. The last question answers the question if discretionary access control can truly be deemed safe.

Acknowledgments: I received no assistance with this exercise.

Contents

1 Executive Summary

All requirements for this assignment have been satisfied. Formulated proofs for the access control schemes presented in the exam questions 1 and 2 are answered in sections 1 and 2 respctively. Section 3 describes discretionary access control and whether it can be truly deemed safe.

2 Question 1

2.1 Problem Statement

The following is the formal proof to allow Mary access to her account in order to purchase tickets. The token Mary uses for gain access to her account is her password. The principals of interest are Mary and the Kennedy Center, *Roles KennedyCenter*.

people `= Mary`

roles `= KennedyCenter`

passPrinc `= Users people`

principals `= PR passPrinc | Pass passPrinc | Role roles`

The resource being protected is Mary's account, represented by the following datatype.

accounts `= AC people`

The final conculsion of the theorem is that the account is satisfied by the Kripke structure.

$\vdash$ `(`*M*`,`*Oi*`,`*Os*`) sat prop (AC Mary)`

2.2 Relevant Code

The access control proof involves four assumptions in the ML variables defined below. The password is the token that represents the authority Kennedy Center's permission for a user to access their account and purchase a ticket.

The final theorem's conculusion when all four assumptions are true is the account proposition *prop (AC Mary)* The theorem is proved with a forward proof using the ACL *SPEAKS_FOR* and *CONTROLS* inference rules

```
val a1 = ``(Name (PR (Users Mary))) says (prop (AC Mary))
          :(accounts,principals,'d,'e)Form``
val a2 = ``(Name (Role KennedyCenter)) controls ((Name (PR (Users Mary)))
           controls (prop (AC Mary)))
                   :(accounts,principals,'d,'e)Form``

val a3 = ``((Name (Pass (Users Mary))) speaks_for (Name (Role KennedyCenter)))
         :(accounts,principals,'d,'e)Form``

val a4 = ``(Name (Pass (Users Mary))) says ((Name (PR (Users Mary))) controls
         (prop (AC Mary))):(accounts,principals,'d,'e)Form``

val [a1_thm,a2_thm,a3_thm,a4_thm] = map ACL_ASSUM [a1,a2,a3,a4]

val thm1 = SPEAKS_FOR a3_thm a4_thm
val thm2 = CONTROLS a2_thm thm1
val exam2_1_thm = CONTROLS thm2 a1_thm
```

2.3 Execution Transcript

```
> val a1 = ‘‘(Name (PR (Users Mary))) says (prop (AC Mary)):(accounts,principals,’d,’e)Form‘‘
val a2 = ‘‘(Name (Role KennedyCenter)) controls ((Name (PR (Users Mary))) controls (prop (AC Mary
       :(accounts,principals,’d,’e)Form‘‘
val a3 = ‘‘((Name (Pass (Users Mary))) speaks_for (Name (Role KennedyCenter))):(accounts,principa
val a4 = ‘‘(Name (Pass (Users Mary))) says ((Name (PR (Users Mary))) controls (prop (AC Mary))):(

val [a1_thm,a2_thm,a3_thm,a4_thm] = map ACL_ASSUM [a1,a2,a3,a4]

val thm1 = SPEAKS_FOR a3_thm a4_thm
val thm2 = CONTROLS a2_thm thm1
val exam2_1_thm = CONTROLS thm2 a1_thm;
# # # # # # # # # # val a1 =
   Name (PR (Users Mary)) says prop (AC Mary):
   term
val a1_thm =
    [.] |- (M,Oi,Os) sat Name (PR (Users Mary)) says prop (AC Mary):
   thm
val a2 =
   Name (Role KennedyCenter) controls Name (PR (Users Mary)) controls
 prop (AC Mary):
   term
val a2_thm =
    [.]
|- (M,Oi,Os) sat
   Name (Role KennedyCenter) controls Name (PR (Users Mary)) controls
   prop (AC Mary):
   thm
val a3 =
   Name (Pass (Users Mary)) speaks_for Name (Role KennedyCenter):
   term
val a3_thm =
    [.]
|- (M,Oi,Os) sat
   Name (Pass (Users Mary)) speaks_for Name (Role KennedyCenter):
   thm
val a4 =
   Name (Pass (Users Mary)) says Name (PR (Users Mary)) controls
 prop (AC Mary):
   term
val a4_thm =
    [.]
|- (M,Oi,Os) sat
   Name (Pass (Users Mary)) says Name (PR (Users Mary)) controls
   prop (AC Mary):
   thm
```

```
val exam2_1_thm =
    [....] |- (M,Oi,Os) sat prop (AC Mary):
   thm
val thm1 =
    [..]
|- (M,Oi,Os) sat
   Name (Role KennedyCenter) says Name (PR (Users Mary)) controls
   prop (AC Mary):
   thm
val thm2 =
    [...] |- (M,Oi,Os) sat Name (PR (Users Mary)) controls prop (AC Mary):
   thm
>    thm


Process HOL finished
```

3 Question 2

3.1 Problem Statement

Problem 2 involves an access control scenario where Don must give his pin, account number, and id in order to gain access to his account. The authority role in this situation is taken by the BankOfRiches and the resources to protect are the *richAccts* datatypes.

The token in this example was the pin of the client and the token states that the meeting of Don, his bank number, and Don's id has control over the resource Don's riches account.

```
people2 = Don

roles2 = BankOfRiches

pinIdPrinc = Client people2

RichesPrincipals =
    PR2 pinIdPrinc
  | PIN pinIdPrinc
  | ID pinIdPrinc
  | Role2 roles2
  | BN num

richAccts = RAC pinIdPrinc

⊢ (M,Oi,Os) sat prop (RAC (Client Don))
```

3.2 Relevant Code

The proof of this formula invovles similar setps to problem one except the assumption must have Don, Don's ID, and the bank number as principals wanting access to Don's account.

```
val b1 = ``(Name (Role2 BankOfRiches)) controls
          ((Name(PR2(Client Don))) meet
                  (Name(ID(Client Don))) meet
                  (Name(BN 4789111238734609))
          controls (prop(RAC(Client Don))))
                  :(richAccts,RichesPrincipals,'d,'e)Form``

val b2 = ``((Name(PIN(Client Don))) speaks_for (Name(Role2 BankOfRiches)))
          :(richAccts,RichesPrincipals,'d,'e)Form``

val b3 = ``(Name(PIN(Client Don))) says
           ((Name(PR2(Client Don))) meet
                   (Name(ID(Client Don))) meet
                   (Name(BN 4789111238734609))
                   controls (prop(RAC(Client Don))))
                   :(richAccts,RichesPrincipals,'d,'e)Form``

val b4 = ``(Name(PR2(Client Don)) meet Name(ID(Client Don))
```

```
          meet Name(BN 4789111238734609))
                says (prop(RAC(Client Don))):(richAccts,RichesPrincipals,'d,'e)F

val [b1_thm,b2_thm,b3_thm,b4_thm] = map ACL_ASSUM [b1,b2,b3,b4]

val thm1_2 = SPEAKS_FOR b2_thm b3_thm
val thm2_2 = CONTROLS b1_thm thm1_2
val exam2_2_thm = CONTROLS thm2_2 b4_thm
```

3.3 Execution Transcript

```
> val b1 = ``(Name (Role2 BankOfRiches)) controls ((Name(PR2(Client Don))) meet
                  (Name(ID(Client Don))) meet
                  (Name(BN 4789111238734609))
    controls (prop(RAC(Client Don))))
:(richAccts,RichesPrincipals,'d,'e)Form``;
# # # # val b1 =
   Name (Role2 BankOfRiches) controls
 Name (PR2 (Client Don)) meet Name (ID (Client Don)) meet
 Name (BN 4789111238734609) controls prop (RAC (Client Don)):
   term
> val b2 = ``((Name(PIN(Client Don))) speaks_for (Name(Role2 BankOfRiches))):(richAccts,Rich
val b2 =
   Name (PIN (Client Don)) speaks_for Name (Role2 BankOfRiches):
   term
> val b3 = ``(Name(PIN(Client Don))) says ((Name(PR2(Client Don))) meet
                (Name(ID(Client Don))) meet
                (Name(BN 4789111238734609))
     controls (prop(RAC(Client Don))))
 :(richAccts,RichesPrincipals,'d,'e)Form``;
# # # # val b3 =
   Name (PIN (Client Don)) says
 Name (PR2 (Client Don)) meet Name (ID (Client Don)) meet
 Name (BN 4789111238734609) controls prop (RAC (Client Don)):
   term
> val b4 = ``(Name(PR2(Client Don)) meet Name(ID(Client Don)) meet Name(BN 4789111238734609)
     says (prop(RAC(Client Don))):(richAccts,RichesPrincipals,'d,'e)Form``;
# val b4 =
   Name (PR2 (Client Don)) meet Name (ID (Client Don)) meet
 Name (BN 4789111238734609) says prop (RAC (Client Don)):
   term
> val [b1_thm,b2_thm,b3_thm,b4_thm] = map ACL_ASSUM [b1,b2,b3,b4];

val thm1_2 = SPEAKS_FOR b2_thm b3_thm;
val thm2_2 = CONTROLS b1_thm thm1_2;
val exam2_2_thm = CONTROLS thm2_2 b4_thm;
val b1_thm =
```

```
    [.]
|- (M,Oi,Os) sat
   Name (Role2 BankOfRiches) controls
   Name (PR2 (Client Don)) meet Name (ID (Client Don)) meet
   Name (BN 4789111238734609) controls prop (RAC (Client Don)):
   thm
val b2_thm =
    [.]
|- (M,Oi,Os) sat
   Name (PIN (Client Don)) speaks_for Name (Role2 BankOfRiches):
   thm
val b3_thm =
    [.]
|- (M,Oi,Os) sat
   Name (PIN (Client Don)) says
   Name (PR2 (Client Don)) meet Name (ID (Client Don)) meet
   Name (BN 4789111238734609) controls prop (RAC (Client Don)):
   thm
val b4_thm =
    [.]
|- (M,Oi,Os) sat
   Name (PR2 (Client Don)) meet Name (ID (Client Don)) meet
   Name (BN 4789111238734609) says prop (RAC (Client Don)):
   thm
> > val thm1_2 =
    [..]
|- (M,Oi,Os) sat
   Name (Role2 BankOfRiches) says
   Name (PR2 (Client Don)) meet Name (ID (Client Don)) meet
   Name (BN 4789111238734609) controls prop (RAC (Client Don)):
   thm
> val thm2_2 =
    [...]
|- (M,Oi,Os) sat
   Name (PR2 (Client Don)) meet Name (ID (Client Don)) meet
   Name (BN 4789111238734609) controls prop (RAC (Client Don)):
   thm
> val exam2_2_thm =
    [....] |- (M,Oi,Os) sat prop (RAC (Client Don)):
   thm
>
```

4 Question 3

I believe that systems can be classified as safe using sicretionary access control schemes using the algorithm in the provided article. Discretionary Access Control involves parties disclosing access control objects and defining their own rules to grant access to protected systems. From my understanding of Kripke structures and the HOL theorem proving tools at engineers disposal, I beleive using DAC and formal proofs and logic systems can be trusted as safe.

A Source for exam2Script.sml

The following code is from *HOL/exam2Script.sml*

```
(* ========================================================================== *)
(*  Exam2 Theory: Access Control Mechanisms
*)
(*  Author: Alfred Murabito
*)
(*  Date: 2/29/2020
*)
(* ========================================================================== *)

structure exam2Script = struct

(* only necessary when working interactively
app load ["acl_infRules","aclrulesTheory","aclDrulesTheory","exam2Theory"]
open acl_infRules aclrulesTheory aclDrulesTheory exam2Theory
*)

open HolKernel boolLib Parse bossLib
open acl_infRules aclrulesTheory aclDrulesTheory

val _ = new_theory "exam2"

val _ =
Datatype
`roles = KennedyCenter`

val _ =
Datatype
`people = Mary`

val _ =
Datatype
`passPrinc = Users exam2$people`

val _ =
Datatype
`principals = PR passPrinc | Pass passPrinc | Role exam2$roles`

val _ =
Datatype
`accounts = AC exam2$people`

(* Question 2 Datatypes *)
val _ =
Datatype
```

```
‘people2 = Don‘

val _ =
Datatype
‘roles2 = BankOfRiches‘

val _ =
Datatype
‘pinIdPrinc = Client exam2$people2‘

val _ =
Datatype
‘RichesPrincipals = PR2 pinIdPrinc | PIN pinIdPrinc | ID pinIdPrinc | Role2 roles

val _ =
Datatype
‘richAccts = RAC pinIdPrinc‘

(* Problem 1 Ticket Assumptions
‘‘(Name (PR (Users Mary))) says (prop (AC Mary)):(accounts, principals, 'd, 'e)Form

‘‘(Name (Role KennedyCenter)) controls ((Name (PR (Users Mary))) controls (prop
:(accounts, principals, 'd, 'e)Form‘‘

‘‘((Name (Pass (Users Mary))) speaks_for (Name (Role KennedyCenter))):(accounts,

‘‘(Name (Pass (Users Mary))) says (prop (AC Mary)):(accounts, principals, 'd, 'e)For
*)


val a1 = ‘‘(Name (PR (Users Mary))) says (prop (AC Mary)):(accounts, principals, '
val a2 = ‘‘(Name (Role KennedyCenter)) controls ((Name (PR (Users Mary))) contro
                  :(accounts, principals, 'd, 'e)Form‘‘
val a3 = ‘‘((Name (Pass (Users Mary))) speaks_for (Name (Role KennedyCenter))):(
val a4 = ‘‘(Name (Pass (Users Mary))) says ((Name (PR (Users Mary))) controls (pr

val [a1_thm, a2_thm, a3_thm, a4_thm] = map ACL_ASSUM [a1, a2, a3, a4]

val thm1 = SPEAKS_FOR a3_thm a4_thm
val thm2 = CONTROLS a2_thm thm1
val exam2_1_thm = CONTROLS thm2 a1_thm

val _ = save_thm("exam2_1_thm", exam2_1_thm)


val b1 = ‘‘(Name (Role2 BankOfRiches)) controls ((Name(PR2(Client Don))) meet
```

```
val b2 = ``((Name(PIN(Client Don))) speaks_for (Name(Role2 BankOfRiches))):(richAccts

val b3 = ``(Name(PIN(Client Don))) says ((Name(PR2(Client Don))) meet
                                                                    (Name(ID
                                                                    (Name(BN
                                                                    controls
                                                                        :(ri

val b4 = ``(Name(PR2(Client Don)) meet Name(ID(Client Don)) meet Name(BN 478911123873
                says (prop(RAC(Client Don))):(richAccts,RichesPrincipals,'d,'e)Form`

val [b1_thm,b2_thm,b3_thm,b4_thm] = map ACL_ASSUM [b1,b2,b3,b4];

val thm1_2 = SPEAKS_FOR b2_thm b3_thm;
val thm2_2 = CONTROLS b1_thm thm1_2;
val exam2_2_thm = CONTROLS thm2_2 b4_thm;

val _ = save_thm("exam2_2_thm", exam2_2_thm)

val _ = print_theory "-";

val _ = export_theory();

end (* structure *)
```