Exam 1

Bharath Karumudi

August 22, 2019

Abstract

This project is to demonstrate the capabilities of implementing constructing and deconstructing HOL Terms using the tools and techniques - LaTeX, AcuTeX, emacs and ML.

Each chapter documents the given problems with a structure of:

1. Problem Statement
2. Relevant Code
3. Execution Transcripts
4. Explanation of results

Acknowledgments: Professor Marvine Hamner and Professor Shiu-Kai Chin who taught the Certified Security By Design.

Contents

Chapter 1

Executive Summary

All requirements for this project are statisfied specifically, and by using HOL solved the below questions:

1. Question 1
2. Question 2
3. Question 3

Chapter 2

Question 1

2.1 Problem Statement

Consider the following situation. Jack is taking an Amtrak train to visit his parents over the winter break from school, University of the Brilliant. He purchased his ticket online and has printed it out. Jack arrives at the train depot and boards the train. As the train pulls out of the depot the conductor approaches and asks for Jacks ticket. Using the programming and mathematical languages you have learned in CIS634 (emacs, LaTex, HOL, etc.) generate a report for Amtraks Security & Operations manager giving a formal proof for why Jack should be allowed to complete his trip.

2.2 Relevant Code

```
1. Jack says <travel, train>                          Jack's Request
2. Amtrack controls (Jack controls <travel, train>)   Access Policy
3. ticket => Amtrack                                  Trust Assumption
4. ticket says (Jack controls <travel, train>)        Jack's Ticket
5. <travel, train>                                    1,2,3,4 ticket rule
```

```
(*****************************************************************************)
(*  Author: Bharath Karumudi                                                 *)
(*  Date: Aug 20, 2019                                                       *)
(*****************************************************************************)


structure question1Script = struct

open HolKernel boolLib Parse bossLib
open acl_infRules aclrulesTheory aclDrulesTheory


(*****************************************************************************)
(* Create New Theory                                                         *)
(*****************************************************************************)

val _ = new_theory "question1";

(*****************************************************************************)
(* Defining instructions                                                     *)
(*****************************************************************************)

val _ =
Datatype
`commands = travel | deny`
```

```
(*****************************************************)
(* Define some names of people who will be principals *)
(*****************************************************)
val _ =
Datatype
`staff = Jack | Amtrack`


val term1=``((Name Jack) says (prop travel)):(commands,staff,'d,'e)Form``;
val term2=``((Name Jack) controls (prop travel)):(commands,staff,'d,'e)Form``;
val term3 = ``((prop travel) impf (prop deny)):(commands, staff, 'd,'e)Form``;
val term4=``((Name Ticket) speaks_for (Name Amtrack)):(commands,staff,'d,'e)Form``;

val question1Thm=
let
val thm1 = ACL_ASSUM term1
val thm2 = ACL_ASSUM term2
val thm3 = ACL_ASSUM term3

val thm4 = CONTROLS thm2 thm1
val thm5 = ACL_MP thm4 thm3
in
SAYS ``Name Jack`` thm5
end;

val _ = save_thm("question1Thm", question1Thm)

(*************************************************************************)
(* Exporting Theory                                                      *)
(*************************************************************************)

val _ = print_theory "-";

val _ = export_theory();

end (* structure *)
```

2.3 Execution Transcripts

1

```
------------------------------------------------------------------
       HOL-4 [Kananaskis 11 (stdknl, built Sat Aug 19 09:30:06 2017)]

       For introductory HOL help, type: help "hol";
       To exit type <Control>-D
------------------------------------------------------------------
[extending loadPath with Holmakefile INCLUDES variable]
> > > > # # # # # # # # # ** types trace now on
> *** Globals.show_assums now true ***
> # # # # # # # # # ** Unicode trace now off
> # # <<HOL message: Defined type: "commands">>
> # # # <<HOL message: Defined type: "staff">>
> val term1 =
   ‘‘Name Jack says (prop travel :(commands, staff, ’d, ’e) Form)‘‘:
   term
> val term2 =
   ‘‘Name Jack controls (prop travel :(commands, staff, ’d, ’e) Form)‘‘:
   term
> val term3 =
   ‘‘(prop travel :(commands, staff, ’d, ’e) Form) impf
  (prop deny :(commands, staff, ’d, ’e) Form)‘‘:
   term
> val term4 =
   ‘‘((Name (Ticket :staff) speaks_for Name Amtrack)
     :(commands, staff, ’d, ’e) Form)‘‘:
   term
>
*** Emacs/HOL command completed ***

> # # # # # # # # # # val question1Thm =

[((M :(commands, ’b, staff, ’d, ’e) Kripke),(Oi :’d po),(Os :’e po)) sat
 Name Jack controls (prop travel :(commands, staff, ’d, ’e) Form),
 ((M :(commands, ’b, staff, ’d, ’e) Kripke),(Oi :’d po),(Os :’e po)) sat
 (prop travel :(commands, staff, ’d, ’e) Form) impf
 (prop deny :(commands, staff, ’d, ’e) Form),
 ((M :(commands, ’b, staff, ’d, ’e) Kripke),(Oi :’d po),(Os :’e po)) sat
 Name Jack says (prop travel :(commands, staff, ’d, ’e) Form)]
|- ((M :(commands, ’b, staff, ’d, ’e) Kripke),(Oi :’d po),
    (Os :’e po)) sat
   Name Jack says (prop deny :(commands, staff, ’d, ’e) Form):
   thm
>
```

2.3.1 Explanation of Results

The above results shows that the requirements are satisfied.

Chapter 3

Question 2

3.1 Problem Statement

Consider the following situation. Alice has a lock box at her bank, the Bank of Riches. To access her lock box Alice needs an official state-issued identification card, typically her drivers license, DL#2507, and the key to her lock box,#113. Alice wants to store a valuable document in her lock box. She proceeds to her bank and presents her credentials in order to access her lock box. Using the programming and mathematical languages you have learned in CIS634 (emacs, LaTex, HOL, etc.) generate a report for the Banks manager giving a formal proof for why Alice should have access. Hint: the key will act as a ticket while the drivers license can be checked against an access control list (ACL).

3.2 Relevant Code

1. Face says <enter, lockerroom>	Request
2. Bank controls ((Alice controls <113,lockerroom>) Alice controls <enter, lockerroom>	Jurisdiction
3. Bank says ((Alice controls <113, lockerroom>) Alice controls <enter, lockerroom>	Access Policy
4. DMV controls (Face => Alice)	Jurisdiction
5. License says (Face => Alice)	Drivers License
6. License => DMV	Trust Assumption
7. Bank controls (Alice controls <113, lockerroom>)	Jurisdiction
8. key says (Alice controls <113, lockerroom>)	Ticket
9. key => Bank	Trust Assumption
10.<enter, lockerroom>	1,2,3,4,5,6,7,8,9 ID \& Ticket Rule

```
(*****************************************************************************)
(*  Author: Bharath Karumudi                                                 *)
(*  Date: Aug 20, 2019                                                       *)
(*****************************************************************************)


structure question1Script = struct

open HolKernel boolLib Parse bossLib
open acl_infRules aclrulesTheory aclDrulesTheory


(*****************************************************************************)
(* Create New Theory                                                         *)
(*****************************************************************************)
```

```
val _ = new_theory "question1";

(*************************************************************************)
(* Defining instructions                                                 *)
(*************************************************************************)

val _ =
Datatype
`commands = travel | deny`

(*****************************************************)
(* Define some names of people who will be principals *)
(*****************************************************)
val _ =
Datatype
`staff = Jack | Amtrack`


val term1=``((Name Jack) says (prop travel)):(commands,staff,'d,'e)Form``;
val term2=``((Name Jack) controls (prop travel)):(commands,staff,'d,'e)Form``;
val term3 = ``((prop travel) impf (prop deny)):(commands, staff, 'd,'e)Form``;
val term4=``((Name Ticket) speaks_for (Name Amtrack)):(commands,staff,'d,'e)Form``;

val question1Thm=
let
val thm1 = ACL_ASSUM term1
val thm2 = ACL_ASSUM term2
val thm3 = ACL_ASSUM term3

val thm4 = CONTROLS thm2 thm1
val thm5 = ACL_MP thm4 thm3
in
SAYS ``Name Jack`` thm5
end;

val _ = save_thm("question1Thm", question1Thm)

(*************************************************************************)
(* Exporting Theory                                                      *)
(*************************************************************************)

val _ = print_theory "-";

val _ = export_theory();

end (* structure *)
```

3.3 Execution Transcripts

1

```
------------------------------------------------------------------------
       HOL-4 [Kananaskis 11 (stdknl, built Sat Aug 19 09:30:06 2017)]

       For introductory HOL help, type: help "hol";
       To exit type <Control>-D
------------------------------------------------------------------------
[extending loadPath with Holmakefile INCLUDES variable]
> > > > # # # # # # # # # ** types trace now on
> *** Globals.show_assums now true ***
> # # # # # # # # # ** Unicode trace now off
>
>
> # # <<HOL message: Defined type: "commands">>
> # # # <<HOL message: Defined type: "people">>
> # # <<HOL message: Defined type: "roles">>
> # # <<HOL message: Defined type: "keyPrinc">>
> # # <<HOL message: Defined type: "principals">>
> val question2Thm =
    []
|- ((M :(commands, 'b, principals, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   Name (PR (Role Commander)) controls
   (prop go :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   reps (Name (PR (Staff Alice))) (Name (PR (Role Commander)))
     (prop go :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Staff Alice)) quoting Name (PR (Role Commander)) says
   (prop go :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   (prop go :(commands, principals, 'd, 'e) Form) impf
   (prop nogo :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   ((Name (Key (Role CA)) speaks_for Name (PR (Role CA)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Role CA)) says
   ((Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (PR (Role CA)) controls
   ((Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Staff Bank)) quoting
   Name (PR (Role (Operator :roles))) says
   (prop nogo :(commands, principals, 'd, 'e) Form):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***
```

3.3.1 Explanation of Results

The above results shows that the requirements are satisfied.

Chapter 4

Question 3

4.1 Problem Statement

Discretionary access control poses real-world challenges in security. This problem is intended to help you explore what those challenges are and some potential ways to mitigate negative impacts due to these challenges. Consider the 2005 paper, On Safety in Discretionary Access Control, written by Li and Tripunitara at Purdue University. Ive uploaded this paper to our course website for you. You can use this paper, your textbook, or other resources at your disposal. But remember, whatever you use you will be on the clock to get your exam done. Given what you now know, this question asks you to consider whether or not you believe that safety in existing discretionary access control schemes being decidable means that these schemes are in fact safe. Or, if this merely means that we can decide what the level of safety in these schemes is. You are to include a 1 page summary in your report that expresses your thinking about this topic including a sentence that summarizes your conclusion.

4.2 Summary

As mentioned by Ninghui et. al in the research paper On safety in Discretionary Access Control Safety analysis decides whether rights can be leaked to unauthorized principals in future states. Safety analysis was shown to be undecidable in the HRU scheme. Safety analysis, first formulated by Harrison, et. al for the access matrix model has been recognized as fundamental problem in access control. Solworth and Sloan safety is undecidable in DAC and use this assertion as the motivation for introducing a new access control scheme based on labels and relabeling that has decidable safety properties. DAC cannot be equated to the HRU scheme for the following reasons: First, the HRU scheme can be used to encode schemes that are not DAC schemes; therefore, the fact that safety is undecidable in the HRU scheme should not lead one to conclude that safety is undecidable in DAC. Second, features in DAC cannot always be encoded in the HRU scheme. For example, some DAC schemes require that each object be owned by exactly one subject; thus removal of a subject who has the ownership of some objects requires the transfer of ownership to some other subject (often times the owner of the subject being removed) so that this property is maintained. Both the removal of the subject and the transfer of ownership of objects it owns occur in a single state-change. The algorithm by authors, suggests that safety in these DAC schemes can be efficiently decided and there is no need to invent new access control schemes with decidable safety as the primary goal. The authors show that the proposed implementation of DAC schemes in the Solworth- Sloan scheme has significant deficiencies. Two particular limitations that discussed are the lack of support for removing subjects and objects and the inability to ensure that an object has only one owner, as required by DAC schemes. Solworth and Sloan presented a new DAC scheme based on labels and relabeling rules. To our knowledge, the work by Solworth and Sloan was the first to directly address safety in DAC. There is considerable overhead in implementing a relatively simple DAC scheme (SDCO) in the Solworth-Sloan scheme. For each object, we need to create a set of labels whose size is linear in the number of the subjects in the state. We also need to create a set of tags whose size is exponential in the number rights in the system. These tags are used to define groups, and the therefore, the number of entries in all the sets of patterns is also exponential in the number of rights in the system. This is considerable overhead considering the simplicity of SDCO, and the fact that one can directly implement it, with efficiently decidable safety. Conclusion, the existing general DAC schemes is decidable and there is no need to invent new DAC schemes with decidable safety as the primary goal.

4.2.1 Conclusion

The existing general DAC schemes is decidable and there is no need to invent new DAC schemes with decidable safety as the primary goal.

Chapter 5

Appendix A: Question 1

The following code is from the file question1Script.sml

```
(*****************************************************************************)
(*   Author: Bharath Karumudi                                                *)
(*   Date: Aug 20, 2019                                                      *)
(*****************************************************************************)


structure question1Script = struct

open HolKernel boolLib Parse bossLib
open acl_infRules aclrulesTheory aclDrulesTheory


(*****************************************************************************)
(* Create New Theory                                                         *)
(*****************************************************************************)

val _ = new_theory "question1";

(*****************************************************************************)
(* Defining instructions                                                     *)
(*****************************************************************************)

val _ =
Datatype
`commands = travel | deny`

(*****************************************************)
(* Define some names of people who will be principals *)
(*****************************************************)
val _ =
Datatype
`staff = Jack | Amtrack`


val term1=``((Name Jack) says (prop travel)):(commands,staff,'d,'e)Form``;
val term2=``((Name Jack) controls (prop travel)):(commands,staff,'d,'e)Form``;
val term3 = ``((prop travel) impf (prop deny)):(commands, staff, 'd,'e)Form``;
val term4=``((Name Ticket) speaks_for (Name Amtrack)):(commands,staff,'d,'e)Form``;

val question1Thm=
let
val thm1 = ACL_ASSUM term1
```

```
val thm2 = ACL_ASSUM term2
val thm3 = ACL_ASSUM term3

val thm4 = CONTROLS thm2 thm1
val thm5 = ACL_MP thm4 thm3
in
SAYS ``Name Jack`` thm5
end;

val _ = save_thm("question1Thm", question1Thm)

(*****************************************************************************)
(* Exporting Theory                                                          *)
(*****************************************************************************)

val _ = print_theory "-";

val _ = export_theory();

end (* structure *)
```

Chapter 6

Appendix B: Question 2

The following code is from the question2Script.sml

```
(*****************************************************************************)
(*   Author: Bharath Karumudi                                                *)
(*   Date: Aug 21, 2019                                                      *)
(*****************************************************************************)


structure question2Script = struct

open HolKernel boolLib Parse bossLib
open acl_infRules aclrulesTheory aclDrulesTheory

(*****************************************************************************)
(* Create New Theory                                                         *)
(*****************************************************************************)
val _ = new_theory "question2"

(*****************************************************************)
(* Defining instructions - go, nogo *)
(*****************************************************************)
val _ =
Hol_datatype
`commands = go | nogo`


(*******************************************************)
(* Defining Principals  - Alice and Bob                *)
(*******************************************************)
val _ =
Hol_datatype
`people = Alice | Bank`

(*****************************************************************************)
(* Define roles                                                              *)
(*****************************************************************************)
val _ =
Hol_datatype
`roles = Commander | CA`

(*****************************************************************************)
(* Define principals that will have keys                                     *)
(*****************************************************************************)
val _ =
```

```
Hol_datatype
`keyPrinc = Staff of people | Role of roles | Ap of num`

(*****************************************************************************)
(* Define principals as keyPrinc and keys                                    *)
(*****************************************************************************)
val _ =
Hol_datatype
`principals =  PR of keyPrinc | Key of keyPrinc`

(*****************************************************************************)
(* Proof for question2Thm                                                    *)
(*****************************************************************************)
val question2Thm =
let
 val th1 = ACL_ASSUM``((Name (PR(Role Commander))) controls (prop go)):(commands,principals
 val th2 = ACL_ASSUM``(reps(Name(PR (Staff Alice)))(Name(PR(Role Commander))) (prop go)):(c
 val th3 = ACL_ASSUM``((Name(Key(Staff Alice))) quoting (Name((PR(Role Commander)))) says (
 val th4 = ACL_ASSUM``((prop go) impf (prop nogo)):(commands,principals,'d,'e)Form``
 val th5 = ACL_ASSUM``((Name(Key(Role CA))) speaks_for (Name(PR(Role CA)))):(commands,princ
 val th6 = ACL_ASSUM``((Name(Key(Role CA))) says ((Name(Key(Staff Alice)) speaks_for (Name
 val th7 = ACL_ASSUM``((Name(PR(Role CA))) controls ((Name(Key(Staff Alice)) speaks_for (Na
 val th8 = SPEAKS_FOR th5 th6
 val th9 = CONTROLS th7 th8
 val th10 = IDEMP_SPEAKS_FOR``(Name((PR(Role Commander))))``
 val th11 = INST_TYPE[``:'a`` |-> ``:commands``] th10
 val th12 = MONO_SPEAKS_FOR th9 th11
 val th13 = SPEAKS_FOR th12 th3
 val th14 = REPS th2 th13 th1
 val th15 = ACL_MP th14 th4
 val th16 = SAYS ``((Name(Key(Staff Bank))) quoting (Name(PR(Role Operator)))):principals
 val th17 = DISCH(hd(hyp th7)) th16
 val th18 = DISCH(hd(hyp th6)) th17
 val th19 = DISCH(hd(hyp th5)) th18
 val th20 = DISCH(hd(hyp th4)) th19
 val th21 = DISCH(hd(hyp th3)) th20
 val th22 = DISCH(hd(hyp th2)) th21
in
 DISCH(hd(hyp th1)) th22
end;

val _ = save_thm("question2Thm",question2Thm)


(*****************************************************************************)
(* Exporting Theory                                                          *)
(*****************************************************************************)

val _ = print_theory "-";

val _ = export_theory();

end (* structure *)
```

Chapter 7

Appendix C

Ninghui Li , Mahesh V. Tripunitara, On Safety in Discretionary Access Control, Proceedings of the 2005 IEEE Symposium on Security and Privacy.