Exam 2

Bharath Karumudi

September 5, 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

Sally purchases a new computer from a reputable company with the operating sys- tem and applications such as web browsers already installed. Upon setting up her computer, Sally types the web address of her favorite Internet bookstore (Good- Books.com) into her web browser. She connects with the bookstores web site and logs onto her account, which is handled by a secure portion of the web site that relies on a private and public key pair. Because she is a careful computer user, she verifies that she has connected to the real GoodBooks.com site by having her web browser authenticate the identity of the site. Her browser reports the following: The identity of this web site has been verified by TrueSignatures, Inc., a certificate authority you trust for this purpose. Using her browser, she looks at the public-key certificate of GoodBooks.com and sees that it is signed by TrueSignatures, Inc. She then makes her selections, places her order, enters her credit-card information, and then leaves the site.

2.2 Relevant Code

1. K TrueSignatures => TrueSignatures	Trust assumption
2. K TrueSignatures says K GoodBooks => GoodBooks	Public key Certificate
3. TrueSignatures controls K GoodBooks => GoodBooks	Jurisdiction
4. TrueSignatures says K GoodBooks => GoodBooks	1, 2 speaks for
5. K GoodBooks => GoodBooks	3,4 controls

```
(*****************************************************************************)
(*  Author: Bharath Karumudi                                                 *)
(*  Date: Sep 03, 2019                                                       *)
(*****************************************************************************)


structure question1Script = struct

open HolKernel boolLib Parse bossLib
open TypeBase isainfRules optionTheory cipherTheory stringTheory


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

val _ = new_theory "question1";

(*****************************************************************************)
(* Proof for GoodBooks pub Key signed by TrueSignatures - Digital Certificate *)
```

```
(*****************************************************************************)

val question1Thm=

 TAC_PROOF(
([],
``! signature.signVerify (pubK TrueSignatures) signature
(SOME "pubK GoodBooks") <=>
(signature =
sign (privK TrueSignatures) (hash (SOME "pubK GoodBooks")))``),
ASM_REWRITE_TAC[signVerify_one_one]);

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
------------------------------------------------------------------
> # # # # # # # # <<HOL message: inventing new type variable names: 'a>>
val question1Thm =
    []
|- !(signature :(string digest, 'a) asymMsg).
     signVerify (pubK (TrueSignatures :'a)) signature
       (SOME "pubK GoodBooks") <=>
     (signature =
      sign (privK TrueSignatures) (hash (SOME "pubK GoodBooks"))):
   thm
> > > # # # # #
*** Time taken: 0.001s
```

2.3.1 Explanation of Results

The above results shows that the requirements are satisfied.

Chapter 3

Question 2

3.1 Problem Statement

In this, we need to show the formal proof for Simple checking rule. Consider a situation where Alice wishes to purchase goods from Bob, her local grocer. For simplicity, we assume that Alice and Bob are both depositors of the same bank. Alice wishes to pay Bob, but she does not have sufficient cash on hand (or she chooses not to use it). Instead, she writes a check to Bob, which is in fact an order to the bank to debit the given amount from her account and to pay it to Bob. Alice hands the check to Bob. Because Bob wishes to be paid, he heads to the bank, endorses Alices check by signing the back of it, and presents the check to the teller for payment. Note the equation in text was considered as: P is Alice and Q is Bob.

3.2 Relevant Code

```
Signature Bob | Signature Alice says
                  (<pay amount, Bob> ^ <debit amt, acct Alice>)
Signature Alice => Alice
Signature Bob => Bob
<amt covered, acct Alice>
<amt covered, acct Alice> > Allice controls
                  (<pay amount, Bob> ^ <debit amt, acct Alice>)
Bob reps Alice on (<pay amt, Bob> ^ <debit amt, acct Alice>)
```

```
(*****************************************************************************)
(*  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 - pay, debit *)
(*****************************************************************************)
val _ =
Hol_datatype
`commands = pay | debit`
```

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

(*************************************************************************)
(* Define roles                                                          *)
(*************************************************************************)
val _ =
Hol_datatype
`roles = payer | payee`

(*************************************************************************)
(* 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 payer))) controls (prop pay)):(commands,principals,'c
 val th2 = ACL_ASSUM``(reps(Name(PR (Staff Alice)))(Name(PR(Role payer))) (prop pay)):(comm
 val th3 = ACL_ASSUM``((Name(Key(Staff Alice))) quoting (Name((PR(Role payer)))) says (prop
 val th4 = ACL_ASSUM``((prop pay) impf (prop debit)):(commands,principals,'d,'e)Form``
 val th5 = ACL_ASSUM``((Name(Key(Role payee))) speaks_for (Name(PR(Role payee)))):(commands
 val th6 = ACL_ASSUM``((Name(Key(Role payee))) says ((Name(Key(Staff Alice)) speaks_for (Na
 val th7 = ACL_ASSUM``((Name(PR(Role payee))) controls ((Name(Key(Staff Alice)) speaks_for
 val th8 = SPEAKS_FOR th5 th6
 val th9 = CONTROLS th7 th8
 val th10 = IDEMP_SPEAKS_FOR``(Name((PR(Role payer))))``
 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 Bob))) quoting (Name(PR(Role Operator)))):principals P
 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 *)
```

3.3 Execution Transcripts

1

```
> # # # <<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 payer)) controls
   (prop pay :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   reps (Name (PR (Staff Alice))) (Name (PR (Role payer)))
     (prop pay :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Staff Alice)) quoting Name (PR (Role payer)) says
   (prop pay :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   (prop pay :(commands, principals, 'd, 'e) Form) impf
   (prop debit :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   ((Name (Key (Role payee)) speaks_for Name (PR (Role payee)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Role payee)) says
   ((Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (PR (Role payee)) controls
   ((Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Staff Bob)) quoting
   Name (PR (Role (Operator :roles))) says
   (prop debit :(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

This question asks you to consider what you know about access control, particularly discretionary access control as weve discussed, and delegation. As you consider this you need to briefly state your opinion about whether or not delegation adds to the decidability of access control, and most importantly if delegation adds to the security of access control.

4.2 Summary

In the discretionary access control (DAC) model, every object (such as files and folders) has an owner, and the owner establishes access for the objects. Many Operating systems such as Windows and most Unix-based systems, use the DAC model. A common example of the DAC model is the NTFS used in Windows. NTFS provides security by allowing users and administrators to restrict the access to files and folders with permissions.

Microsoft systems identify users with SIDs and every object includes a discretionary access control list (DACL) that identifies who can access it in a system using the DAC model. The DACL is a list of Access Control Entries. Each ACE is composed of a SID and the permissions granted to the SID.

If user create a file, they are designated as the owner and have explicit control over the file. As the owner, users can modify the permissions on the object by adding user or group accounts to the DACL and assigning the desired permissions. The DAC model is significantly more flexible than the Mandatory Access Control (MAC) model.

An inherent flaw associated with the DAC model is the susceptibility to Trojan horses. These Trojan horses injects the malware and when required they work on behalf of the user (delegation) unknowingly and make the damage to the system. Beside malware, if a user want to delegate the privileges, the delegation comes with trust and trustworthy. If the user is not trustworthy, then the delegation will create a loop-hole in the system security.

4.2.1 Conclusion

In my opinion, delegation, especially with DAC, may potentially creates the damage to the system in long run.

Chapter 5

Appendix A: Question 1

The following code is from the file question1Script.sml

```
(*****************************************************************************)
(*   Author: Bharath Karumudi                                                *)
(*   Date: Sep 03, 2019                                                      *)
(*****************************************************************************)


structure question1Script = struct

open HolKernel boolLib Parse bossLib
open TypeBase isainfRules optionTheory cipherTheory stringTheory


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

val _ = new_theory "question1";

(*****************************************************************************)
(* Proof for GoodBooks pub Key signed by TrueSignatures - Digital Certificate *)
(*****************************************************************************)

val question1Thm=

 TAC_PROOF(
([],
``! signature. signVerify (pubK TrueSignatures) signature
(SOME "pubK_GoodBooks") <=>
(signature =
sign (privK TrueSignatures) (hash (SOME "pubK_GoodBooks")))``),
ASM_REWRITE_TAC[signVerify_one_one]);

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 - pay, debit *)
(*****************************************************************)
val _ =
Hol_datatype
`commands = pay | debit `


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

(*****************************************************************************)
(* Define roles                                                              *)
(*****************************************************************************)
val _ =
Hol_datatype
`roles = payer | payee`

(*****************************************************************************)
(* 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 payer))) controls (prop pay)):(commands,principals,’
 val th2 = ACL_ASSUM‘‘(reps(Name(PR (Staff Alice)))(Name(PR(Role payer))) (prop pay)):(comm
 val th3 = ACL_ASSUM‘‘((Name(Key(Staff Alice))) quoting (Name((PR(Role payer)))) says (prop
 val th4 = ACL_ASSUM‘‘((prop pay) impf (prop debit)):(commands,principals,’d,’e)Form‘‘
 val th5 = ACL_ASSUM‘‘((Name(Key(Role payee))) speaks_for (Name(PR(Role payee)))):(commands
 val th6 = ACL_ASSUM‘‘((Name(Key(Role payee))) says ((Name(Key(Staff Alice)) speaks_for (Na
 val th7 = ACL_ASSUM‘‘((Name(PR(Role payee))) controls ((Name(Key(Staff Alice)) speaks_for
 val th8 = SPEAKS_FOR th5 th6
 val th9 = CONTROLS th7 th8
 val th10 = IDEMP_SPEAKS_FOR‘‘(Name((PR(Role payer))))‘‘
 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 Bob))) quoting (Name(PR(Role Operator)))):principals P
 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

https://www.techotopia.com/index.php/Mandatory,_Discretionary,_Role_and_Rule_Based _Access_Control