PROCEEDINGS OF SCIENCE

Formal Design and Verification for A Typical Security Gateway

Ruiyun Wang[1]
The PLA Information Engineering University
Zhengzhou 450001, China
E-mail: wry0068@126.com

Guolei Zhao
The PLA Information Engineering University
Zhengzhou 450001, China
E-mail: glz0371@163.com

Chaowen Chang
The PLA Information Engineering University
Zhengzhou 450001, China
E-mail: ccw@xdja.com

Xuejian Wang
The PLA Information Engineering University
Zhengzhou 450001, China
E-mail: 1368154434@qq.com

To ensure the security of the top-level design of the security gateway, we proposed a method of formally designing and verifying a typical security gateway. Firstly, we designed the typical security gateway's security policy according to its security requirements. Secondly, we formally modeled the security policy and verified the security model's internal consistency by means of BLP model. In the end, we verified the consistency between the security gateway's functional specifications and its security model. To make sure the reasoning procedure's correctness, we used the theorem prover Isabelle/HOL to formally describe the above work and help us deduce. Our work ensures the security of a typical security gateway in terms of top-level design and exerts certain referential significance on formal design of security gateway.

CENet2017
22-23 July, 2017
Shanghai, China

[1]Speaker

PoS(CENet2017)076

1. Introduction

To ensure the safety of connection between the intranet and external network, it generally deploys the security gateway which has encryption function, such as IPSEC VPN[1] and SSL VPN[2], etc. To improve the security of the whole network; however, in recent years, the OpenSSL HeartBleed flaw[3] leads to fatal attacks on products because of implementation vulnerabilities of these security products. To ensure the implementation security of security products, scholars both at home and abroad carried out many researches by using such formal methods as formal specification and formal verification[4]. Klein G, etc. reported on the formal, machine-checked verication of the seL4 microkernel from an abstract specication down to its C implementation[5].The Verisoft project aimed at the pervasive formal verification from the appl i-cation layer over the system level software, comprising a microkernel and a compiler, down to the hardware[6]. Qian Z J, etc. proposed a method for formal design and verification of the oper ating system[7]; however, the majrotiy formalisim research works fall in the scope of operating systemin terms of formalization research on security gateway, with few but not none. Pierre Bieber etc.[8] described the BLP model [9] and B-method [10] to develop a gateway capable of meeting the ITSEC E4[11] requirements. However, with its focus on the evaluation of security gateway, the idea is not applicable to the formal design and formal verification for security gateway. Currently, the research of formal design and formal verification for security gateway at home is relatively small. In response to these realities and needs of research projects, this paper presents a method used to formally design and verify a typical security gateway.

We firstly introduce some basic and important concepts and terms relative to formal design [12]. ① The security policy is the solution to security threats, including laws，regulations and implementing rules related to management, protection and distribution of sensitive information. The security policy consists of a series of strict rules which determine the rules for granting access and basis foraccess control decision; ② the security model is a simple, abstract and unambiguous description of security requirements expressed by the security policy. It provides a framework that associates security policy and implementation of the security policy. While there are lots of classical information security models like BLP model[9],Biba model[13] and secure information flow model [14] and Unwinding and Inference Control model[15]， we chose the BLP model to formally model the security policy of typical security gateway owing to the easy understanding of BLP model itself and familiarity of most projects engineers with BLP model;③formal specification aims at describing the system functions in a simple, unanimous manner.

This paper presents a method which may be used to formally design and verify the typical security gateway. Section 2 gives security policy of the typical security gateway; Section 3 gives security model on the basis of access control rules in BLP model by using Isabelle/HOL[16]; Section 4 gives the consistency verification between functional specification and security model of typical security gateway; Section 5 summarizes our work and gives further research directions.

2. Security Policy of Typical Security Gateway

The security gateway which uses TCP/IP protocol [17-18] to connect the intranet and extranet and protects the confidentiality and integrity of intranet data. Fig. 1 shows the hardware

structure of typical security gateway. The security chip is a hardware platform, the micro-operation system provides the basic task-scheduling and the memory management functions; the configuration module is responsible for parameter configuration of the whole security gateway; the filter module chooses the communication packet which is from communication module and fulfill the configuration rules (port, IP address and etc.) and transmit it to the next module; the data encryption and decryption module completes transformations between plain text and cipher text. It ensures that only the legal packets can be decrypted and transmitted onto the intranet; in addition, only the encrypted packets can be transferred to extranet by detecting the replay attack and integrity of packets. In this sense, the data encryption and decryption module can be seen as a kind of filter.

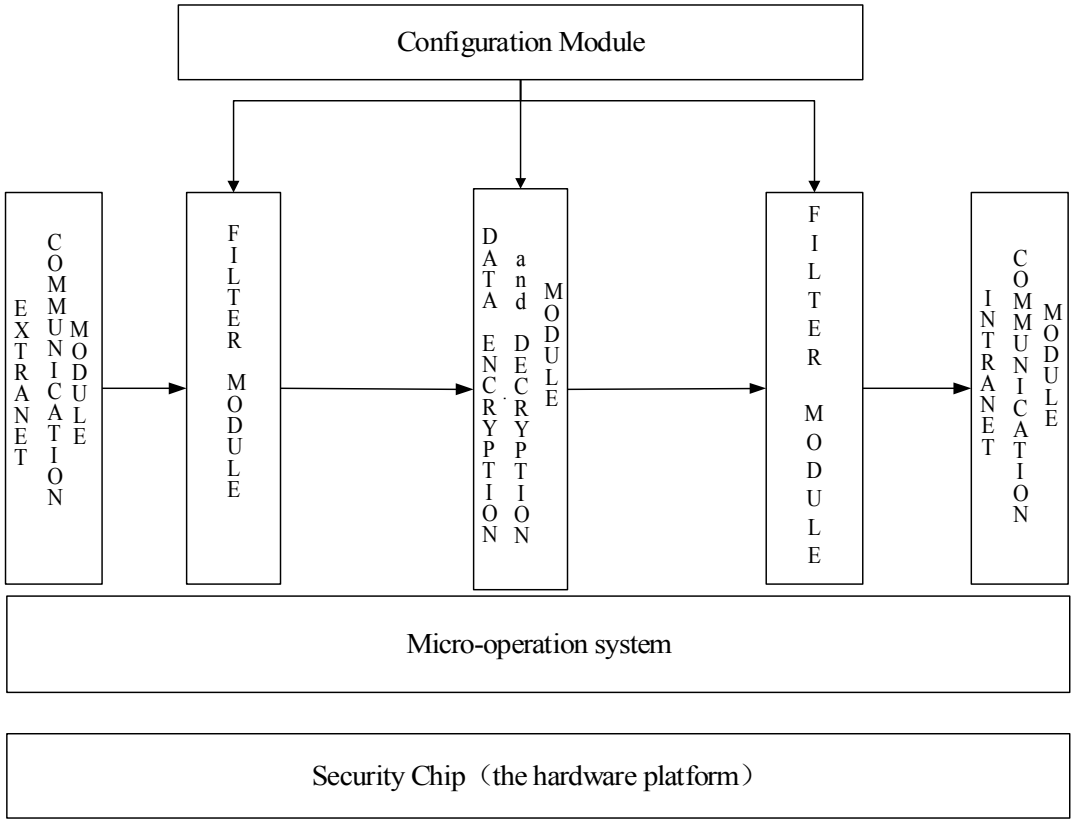

Figure 1: Hardware Structure of Typical Security Gateway

Fig. 1 gives information flows from extranet to intranet. The incoming information is firstly handled by extranet communication module according to TCP/IP protocol and transmitted onto filter module and then the data encryption and description module for further processing. In the end, legal packets from filter module are transmitted onto intranet after being processed by the intranet communication module. The information flows from intranet to extranet is similar to that of above.

In order to make sure that the security gateway has been correctly filtered the exchanged messages between intranet and extranet, it needs to fulfill the following two security requirements:

The information isolation: within the security gateway, the information originating from the intranet or the extranet should be isolated, which means no module has both access to the information originating from intranet and extranet.

The information filtration: the information that is transmitted by security gateway onto the other network has been filtered by filter module, data encryption and decryption module.

In order to meet the above-mentioned security requirements, we define the security levels that could be associated with the objects and subjects of security gateway, and determine whether the information flow between levels should be authorized or not, as is shown below:

high：the security level of information of high confidentiality, information from intranet is associated with this level.

low：the security level of information of low confidentiality, information from extranet is associated with this level.

middle: the security level of configuration information.

In order to enforce the information isolation requirement, the information flow between high and low levels should be forbidden; the information flow from one level to itself and the information flow fromthe middle to high and low level may be allowed. However, in order to avoidinformation that could indirectly flow from high to low level through the middle level, the information flow from high or low to middle level should be forbidden. Table 1 summarizes the authorized information flows ("1"in Table 1,Table 2 and Table 3 represents the information flow from level of the first column to level of the first row is allowed and "null" represents the corresponding information flow is forbidden).

	high	low	middle
high	1		
low		1	
middle	1	1	1

Table 1. Authorized Flows for Isolation

According to the information filtration requirement, the filtered information originating from extranet should be transmitted onto the intranet, indicating that the information may flow from the low to high level. This is inconsistent with information isolation requirement. For solving this problem, we redefine the security levels as levels made of two components, an isolation level(low, high, middle) and a filtration level(in, ok, out)：

<low,in> is the security level of information from extranet and to be filtered by filter modules.

<low,ok> is the security level of information from extranet and filtered by filter modules.

<high,out> is the security level of information filtered by filter modules and to be transmitted onto the intranet.

Table 2 summarizes the new authorized flows.

	<low,in>	<low,ok>	<high,out>	<middle,?>
<low,in>	1			
<low,ok>		1	1	
<high,out>			1	
<middle,?>	1	1	1	1

Table 2. New Authorized Flows for Isolation

In order to enforce the filtration requirement, we introduce two filtration levels f_{tf} and f_{fi} for each filter module f .

<*low,* f_{fi} > is the security level of filter module. No other module may be associated with this level.

<*low,* f_{tf} > is the security level of information to be filtered by f .

In order to let a filter read information be filtered and filter it, the information flow from <*low,* f_{tf} >to <*low,* f_{fi} >should be allowed. The first filter module of security gateway is denoted by f_1 , the second one is denoted by f_2 , and so forth. Then, the information flow from <*low,in*> to <*low,* $f1_{tf}$ > and the information flow from <*low,* $f1_{tf}$ > to <*low,* $f1_{fi}$ > may be allowed. If f_2 is the last filter module, the information flow from <*low,* $f2_{fi}$ >to <*low,ok*> should be allowed. Table 3 summarizes all the authorized flows .

	<low,in>	<low, $f1_{tf}$ >	<low, $f1_{fi}$ >	<low, $f2_{tf}$ >	<low, $f2_{fi}$ >	<low,ok>	<high,out>	<middle,?>
<low,in>	1	1						
<low, $f1_{tf}$ >		1	1					
<low, $f1_{fi}$ >			1	1				
<low, $f2_{tf}$ >				1	1			
<low, $f2_{fi}$ >					1	1		
<low,ok>						1	1	
<middle,?>	1	1	1	1	1	1	1	1

Table 3. Authorized Flows for Isolation and Filtration

Based on the aforementioned definitions, the security gateway's security policy meeting with the two security requirements, the information isolation and the information filtration, is given as follows according to the BLP model: for some functional requirements, the modules of security gateway involved are regarded as subjects, the data flows involved in the functional requirement are seen as objects and we associate each subject and object with appropriate security level. A subject has observation access to some object, if and only if the information flow from the security level of the object to the security level of the subject is authorized; a subject has alteration access to some object, if and only if the information flow from the security level of the subject to the security level of the object is authorized.

3.Security Model

This section presents the formal model of the aforementioned security policy in Section 2 on the basis of the access control rules of BLP model and gives an internal consistency verification of the formal security model. In order to guarantee the correctness of reasoning, Isabelle/HOL is used when describing the formal model and verifying its internal consistency.

3.1 Symbol System of Isabelle/HOL

Isabelle is a generic system for implementing logical formalisms and Isabelle/HOL is the specialization of Isabelle for HOL, which abbreviate Higher-Order Logic and provide interactive verification environment by means of functional programming. HOL is a typed logic. There are type variables, denoted by $'a$ and $'b$ and etc.; the terms formed as $x::'a$, indicating that x is a term of type $'a$. The general HOL datatypes can be defined by datatype

command, as we define the isolation level datatype LEVELi = low|high | middle which means the datatype leveli introduces three constructors low, high and middle； type_synonym command creats the type synonyms, like type_synonym LEVEL = "LEVELi * LEVELf"； the nonrecursive definitions can be made with the definition command, for example, the definition flow: : "* LEVEL (LEVEL) set" where "flow = = {k. fst K = snd K}. The form“｛s． Ps｝”is a definition of set whose elements satisfy the predicate formula P; declaring a constant without definition can be made with the consts command, like consts obj_active::"OBJECT set"； declaring new type without definition can be made with typedecl command, like typedecl OBJECT, a declaration of OBJECT types; declaiming a property can be made with axiomatization command, like axiomatization where obj_not_interaction [simp]: " obj_active ∩ obj_zombie =｛｝".

3.2 Description of Security Model in Isabelle/HOL

To formally describe the security policy of typical security gateway, we create the theory Formal Model. Its axioms correspond to security property of the BLP Model and its operations are corresponding to the rules of BLP model, like creation and delegation of subject and object, change of current level or granting access, etc.

For instance, the proof process for operations granting access and creation of objects are illustrated as follows (the specific meaning of code is given in (**)):

```
theory Formal Model
imports Main
begin
typedecl OBJECT （*the declaration of new type OBJECT *）
Typedecl SUBJECT (*the declaration of new type SUBJECT *)
Datatype LEVELi = low|high|param (*the definition of the isolation level*)
Datatype LEVELf=out|inn|ok|f1fi|f1tf|f2fi|f2tf (*the definition of filtration level*)
type_synonym LEVEL =" LEVELi*LEVELf"
Data-type RIGHT = obs|alt (*right = observation or alteration *)
the definition flow :: "(LEVEL*LEVEL) set "where"flow == {K. fst K = snd K} ∪ {K. fst(fst K) = param} ∪ {((low,inn),(low,f1tf)), ((low,f1tf),(low,f1fi)), ((low,f1fi), (low,f2tf)), ((low,f2tf), (low,f2fi)), ((low,f2fi), (low,ok)), ((low,ok),(high,out)) }" (*the definition of authorized information flows *)
consts obj_active ::"OBJECT set" (*the definition of a set of active objects *)
consts obj_zombie ::"OBJECT set"（*the definition of a set of deleted objects*）
consts subj_active ::"SUBJECT set"（* the definition of a set of active subjects *）
consts subj_zombie ::"SUBJECT set" (*the definition of a set of deleted subjects *)
consts origin :: "(SUBJECT * LEVEL) set" (*used to associate a subject with original level *)
consts c_level :: "(SUBJECT * LEVEL) set" (*used to associate a subject with current level *)
consts clasf :: "(OBJECT* LEVEL) set"（*used to associate a object with classification level *）
consts c_access:: " ((SUBJECT * OBJECT) * RIGHT)set"（*a set of current access of subjects to objects *）
(**********AXIOMS**********)
```

axiomatization where obj_not_ interaction[simp]: " obj_active ∩ obj_zombie ={} "

axiomatization where subj_not_interaction[simp]:" subj_active ∩ subj_zombie = {} "

definition subj_obj1 ::"(SUBJECT*OBJECT)set" where "subj_obj1 == {K. ∃m. m∈ c_access∧ fst m = K ∧ snd m = obs }" （*the set of pairs (subject × object) , where subject was granted access right obs on object *）

definition subj_obj0 :: "(SUBJECT*OBJECT)set" where "subj_obj0 == {K. ∃m n. m∈origin∧ n∈clasf∧ (fst m , fst n) = K ∧(snd n ,snd m) ∈ flow}" (*the set of pairs (subject × object), where the information flow from the classification level of object to the original level of subject is authorized *）

axiomatization where obs1[simp]: " subj_obj1 ⊆ subj_obj0 " (*a subject has observation access to a object, if and only if the information flow from the classification level of the object to the original level of the subject is authorized. ---Simple Security Property of BLP*)

definition subj_obj00 ::"(SUBJECT * OBJECT)set" where"subj_obj00 == {K. ∃m n. m∈c_level ∧ n∈clasf ∧ (fst m , fst n) = K ∧(snd n ,snd m) ∈ flow }" (*the set of pairs (subject × object), where the information flow from the classification level of object to the current level of subject is authorized *）

axiomatization where obs2[simp]: " subj_obj1 ⊆ subj_obj00 " (*a subject has observation access to a object, if and only if the information flow from the classification level of the object to the current level of the subject is authorized. ---* Property of BLP *)

definition subj_obj2 ::"(SUBJECT * OBJECT)set" where"subj_obj2 == {K. ∃m. m∈c_access∧ fst m = K ∧snd m = alt}" (*the set of pairs (subject × object) , where subject was granted access right alt on object *)

definition subj_obj000 ::"(SUBJECT * OBJECT)set" where "subj_obj000 == {K. ∃m n. m∈c_level ∧ n∈clasf ∧ (fst m , fst n) = K ∧(snd m ,snd n) ∈ flow}" （* the set of pairs (subject × object), where the information flow from the current level of subject to the classification level of object is authorized *）

axiomatization where alt[simp]: " subj_obj2 ⊆ subj_obj000 "(* a subject has alteration access to a object, if and only if the information flow from the current level of the subject to the classification level of the object is authorized. ---* Property of BLP *)

(**********OPERATIONS **********)

definition get_access::"SUBJECT⇒OBJECT set ⇒OBJECT set ⇒ ((SUBJECT*OBJECT) * RIGHT)set" where " get_access ss oo_obs oo_alt =

```
( let
        subj_obj3 = {ss} ×oo_obs;
        subj_obj4 = {ss} ×oo_alt
 in (
        if (ss∈subj_active) ∧ (oo_obs ⊆ obj_active)∧ (oo_alt ⊆ obj_active )∧(subj_obj3
⊆
           subj_obj0)∧(subj_obj3 ⊆ subj_obj00)∧(subj_obj4 ⊆ subj_obj000)
       then
            c_access ∪ subj_obj4 ×{alt} ∪subj_obj3 ×{obs}
       else
            c_access
)
```

)" (*If it satisfies conditions （ss∈subj_active) ∧ (oo_obs ⊆ obj_active)∧ (oo_alt ⊆ obj_active)∧(subj_obj3⊆subj_obj0)∧(subj_obj3⊆subj_obj00)∧(subj_obj4⊆subj_obj000),then getting access is OK; otherwise,failure*)

definition create_object :: "SUBJECT ⇒ OBJECT set ⇒LEVEL ⇒OBJECT set"where " creat_object_ok ss oo nn =

(if (ss∈subj_active) ∧ (oo ∩ obj_active = {})∧ (oo ∩ obj_zombie = {})∧(∃K. K∈
c_level ∧ (fst K = ss) ∧ {snd K} ×{nn}⊆ flow)

then

obj_active ∪ oo

else

obj_active

)" (*If it satisfies conditions(ss∈subj_active) ∧ (oo ∩ obj_active = {})∧ (oo ∩ obj_zombie = {})∧(∃K. K∈ c_level ∧ (fst K = ss) ∧ {snd K} ×{nn}⊆ flow)，then creating object oo is OK ,Otherwise, failure*)

3.3 Internal Consistency Verification of the Security Model

The goal of internal consistency verification is that every operation preserves the simple security property and *property. For that purpose, it's sufficient to prove the 5 axioms should hold in states reached after the operation is executed. For instance, the proof of the operation get_access in Isabelle/HOL is illustrated as following:

lemma "({K. ∃m. m∈get_access ss oo_obs oo_alt ∧ fst m = K ∧ snd m = obs } ⊆ subj_obj0)∧({K. ∃m. m∈get_access ss oo_obs oo_alt ∧ fst m = K ∧ snd m = obs } ⊆ subj_obj00)∧({K. ∃m. m∈get_access ss oo_obs oo_alt ∧ fst m = K ∧ snd m = alt}⊆ subj_obj000) "

apply (simp add:get_access_def)

apply (simp add:Let_def)

apply (rule conjI)

apply (insert obs1 obs2 alt)

apply (simp_all add:subj_obj1_def subj_obj2_def)

apply auto

done

Simp method simplifies the first subgoal according to relevant definitions; rule method introduces the rule conjI to subgoal; insert method inserts axioms obs1, obs2 and alt as new assumptions when attacking all subgoals; auto method simplifies all subgoals.

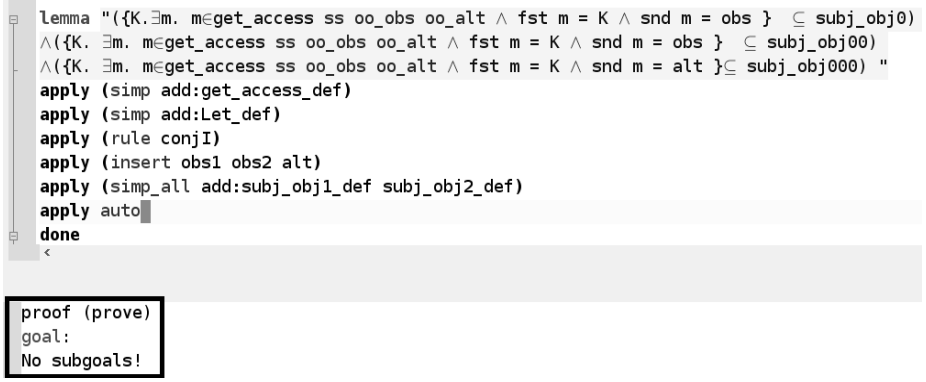

Figure 2: Result of Proof in Isabelle/HOL

Fig. 2 is the result of this proof . "No subgoals" shows that the proof is complete and no more subgoals yield.

4. Consistency Verification Between Functional Specification and Security Model

As is shown in Fig. 1, the hardware structure consists of some modules which have different functions. This section will give the consistency verification between functional specifications of these modules and security model. Due to the lack of space, we only select the filter module to state the consistency verification method.

First of all, we give the formal specification by formally describing the functional specifications. For each function, we search for the module that is supposed to satisfy this function. We regard this module as subject module_name of the security model. We also look for the data flows involved in the functional specification. We regard these data flows as objects data_flow_name1 and data_flow_name2, etc. of the security model. We determine whether in this functional specification data flows are observed, altered, created or deleted. Then, we give the description of the functional specification which has the form: "Subject module_name requires to have access rights rr on objects data_flow_name_1, data_flow_name_2, … data_ flow_name_n, where rr is obs, alt, creation_object or delegation_object" on basis of security model. Furthermore, we give the formal specification in Isabelle/HOL.

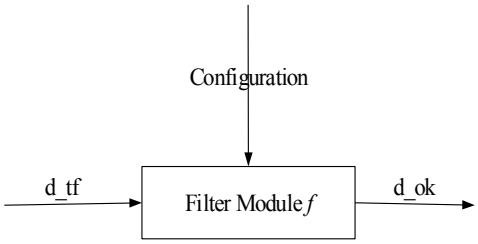

Figure 3: Filter Incoming Messages

As is shown in Fig. 3, for the functional specification "filter incoming message ", the associated description on basis of security model is that subject filter module (noted f)should be authorized to observe objects incoming message to be filtered (noted d_tf) and configuration information (the noted configuration) and alter object incoming data filtered(noted d_ok). Hence, the corresponding formal specification in Isabelle/HOL is get_access f {d_ok, configuration } {d_tf}.

According to the rules of the security model, whether the operation get_access f {d_ok, configuration } {d_tf} is ok is still unknown. In order to solve this problem, the consistency verification between formal specification and security model should be carried out. It's sufficient to verify whether the precondition of this operation is fulfilled. If it's fulfilled, the formal specification is consistent with security model; if not, we should conclude that the functional specification goes against the security policy and this function should be forbidden. Next we give the proof of the operation get_access f {d_ok, configuration } {d_tf} in Isabelle/HOL.

To verify whether the operation "get_access f {d_ok, configuration } {d_tf}" is OK, it's sufficient to verify whether its precondition (f∈subj_active) ∧ ({d_tf,management} ⊆ obj_active)∧({d_ok}⊆obj_active)∧({f}×{d_tf,management}⊆subj_obj0)∧({f}×{d_tf,manag ement}⊆ subj_obj00)∧({f} ×{d_ok }⊆ subj_obj000) is satisfied. The proof in Isabelle/HOL is as following:

```
theory Consistency Verification
imports Main
begin
datatype LEVELi = low|high|param
datatype LEVELf = out|inn|ok|f1fi|f1tf|f2fi|f2tf
datatype RIGHT = obs|alt
type_synonym LEVEL =" LEVELi*LEVELf"
definition flow :: "(LEVEL*LEVEL) set" where"flow == {K. fst K = snd K} ∪{K. fst(fst K) = param} ∪{((low,inn),(low,f1tf)),((low,f1tf),(low,f1fi)),((low,f1fi),(low,f2tf)),((low,f2tf),(low,f2fi)), ((low,f2fi),(low,ok)),((low,ok),(high,out))}"
datatype OBJECT = d_tf |d_ok |management
datatype SUBJECT= f
definition obj_active ::"OBJECT set" where "obj_active == {d_tf,d_ok,management}" (*the definition of a set of active objects *)
consts obj_zombie ::"OBJECT set"
definition subj_active ::"SUBJECT set" where "subj_active == {f}" （* the definition of a set of active subjects *）
consts subj_zombie ::"SUBJECT set"
definition origin:: "(SUBJECT* LEVEL)set" where"origin == {f}×{low}×{f1fi}" (*used to associate subject f with original level （low，f1fi）*)
definition c_level :: "(SUBJECT* LEVEL)set" where "c_level == {f}×{low}×{f1fi} " (*used to associate subject f with current level （low，f1fi）*)
definition clasf::"(OBJECT*LEVEL)set"where"clasf=={d_tf}×{low}×{f1tf}∪{d_ok}×{low}×{f2tf}∪{management}×{param}×{ok} " （ *used to associate a object with classification level *）
consts c_access:: " ((SUBJECT * OBJECT) * RIGHT)set"
definition subj_obj1 ::"(SUBJECT * OBJECT)set" where"subj_obj1 == {K. ∃m. m∈c_access∧ fst m = K ∧ snd m = obs }" （*the set of pairs (subject × object) , where the subject was granted access right obs on object *）
definition subj_obj0 :: "(SUBJECT * OBJECT)set" where"subj_obj0 == {K. ∃m n. m∈origin∧ n∈clasf∧ (fst m, fst n) = K ∧(snd n, snd m) ∈ flow}" (*the set of pairs (subject × object), where the information flow from the classification level of object to the original level of subject is authorized *）
definition subj_obj00 :: "(SUBJECT * OBJECT)set" where"subj_obj00 == {K. ∃m n. m∈c_level ∧ n∈clasf ∧ (fst m, fst n) = K ∧(snd n, snd m) ∈flow }" (*the set of pairs (subject × object), where information flow from the classification level of object to the current level of subject is authorized *）
definition subj_obj2 ::"(SUBJECT*OBJECT)set" where "subj_obj2 == {K. ∃m. m∈c_access∧ fst m = K ∧snd m = alt}" (*the set of pairs (subject × object) , where the subject was granted access right alt on object *)
definition subj_obj000 ::"(SUBJECT * OBJECT)set" where "subj_obj000 == {K. ∃m n. m∈c_level ∧ n∈clasf ∧ (fst m, fst n) = K ∧(snd m, snd n) ∈ flow}" （*the set of pairs (subject × object), where information flow from the current level of subject to the classification level of object is authorized *）
(***the consistency verification between functional specification and security model***)
```

```
lemma " let
            ss=f;
            oo_obs={d_tf, management};
            oo_alt={d_ok}
        in  (ss∈subj_active) ∧ (oo_obs ⊆ obj_active)∧ (oo_alt ⊆ obj_active )∧({ss}
            ×oo_obs⊆subj_obj0 ∧({ss} ×oo_obs ⊆ subj_obj00)∧({ss} ×oo_alt ⊆
subj_obj000))"
apply (simp add:Let_def)
apply (simp add:subj_active_def obj_active_def)
apply (simp only:subj_obj0_def subj_obj00_def subj_obj000_def)
apply (simp only:habilt_def clasf_def flow_def c_level_def)
apply auto
done
```

The result of this proof is shown in Fig. 4."No subgoals" shows that the proof is complete and no more subgoals are yielded.

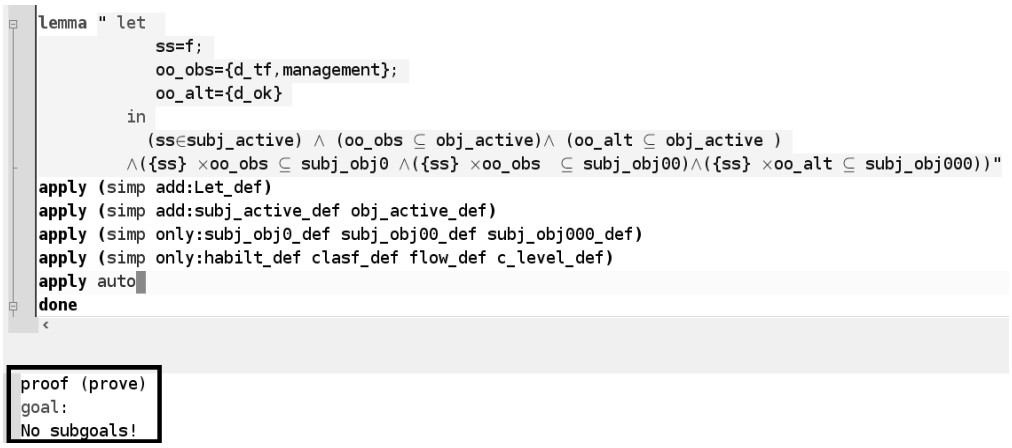

Figure 4: Result of Proof in Isabelle/HOL

5.Conclusion

Formal methods are adopted to analyze and verify information security products in a design level to guarantee the consistency between functional specifications and security model, which is the first step to ensure the implementation safety of the information security product. In order to assure the safety of typical security gateway in the top design level, we proposed a method to formally design and verify typical security gateway. we firstly formally modeled the security policy of typical security gateway on basis of BLP. Secondly we gave the formal specification in Isabelle/HOL by formally analyzing the functional specification. Finally, we gave the consistency verification between the formal specification and the formal security model with the assistance of Isabelle/HOL in order to ensure the safety of functional specification.

On the basis of BLP model, formal descriptions of typical security gateway and its function modules can be made, our future work will continue the research on the formal design and verification of each function module of typical security gateway.

References

[1] A R Viaplana. *Ipsec VPN*[EB/OL].[2017-4-10].http://upcommons.upc.edu/handle/2099. 2/2117.

[2] P Agarwal, S Thirunarayanan, S K Adhya . *Systems and methods for fine grain policy driven clientless SSL VPN access*[J]. 2017.

[3] T Wu, X H Kuang, Z L Fu. *Analysis and Suggestion of the Heartbleed*[J].National Defense Science & Technology, 2014,35(5):27-30. (in Chinese)

[4] D A PELED. *Software Reliability Methods* [M]. Springer,2001.

[5] G Klein, J Andronick, K Elphinstone. *seL4: formal verification of an operating-system kernel*[J]. Communications of the Acm, 2010, 53(6):107-115.

[6] E Alkassar, M A Hillebrand, D Leinenbach. *The Verisoft Approach to Systems Verification*[C]// International Conference on Verified Software: Theories, Tools, Experiments. Springer-Verlag, 2008:209-224.

[7] Z J Qian, H Huang, F M Song. *Research on Consistency Verification of Formal Design and Security Requirments for Operating System*[J].CHINESE JOURNAL OF COMPUTERS, 2014(5):1082-1099. (in chinese)

[8] P Bieber. *Formal techniques for an ITSEC-E4 secure gateway*[C]// Computer Security Applications Conference, 1996. IEEE, 1996:236-245.

[9] D E Bell, A L Lapadul. *Secure computer system: mathematical foundation*, MTR2527[R]. Belford: Mitrecorp,1973.

[10] J R Abrial . *Modeling in Event-B: System and Software Engineering*[M]. Cambridge university Press,2010.

[11] Z Chen, L Huang, L Chen. *ITSEC: An information-theoretically secure framework for truthful spectrum auctions*[C]// Computer Communications. IEEE, 2015:2065-2073.

[12] Q N Shen, S H Qing. *Design for Operating System Security*[M]. Beijing: China Machine Press, 2013. (in Chinese)

[13] W Liu, P Liao, Y Chen. *WEB System Security Technology Based on Operating System Enhancements*[J]. Electric Power Information & Communication Technology, 2016.

[14] P Malacaria, J Heusser. *Information Theory and Security: Quantitative Information Flow*[J]. Lecture Notes in Computer Science, 2010, 6154:87-134.

[15] J A Goguen,J Meseguer. *Unwinding and Inference Control*[C]//Security and Privacy, 1984 IEEE Symposium on. IEEE, 1984:75-75.

[16] T Nipkow, L C Paulson , M Wenzel. *A proof assistant for higher-order logic*[C]// of Lecture Notes in Computer Science. 2008: xiii-xiv.

[17] M C Wood. *Firewall security for computers with internet access and method*[J]. 2017.

[18] A Mejías, R S Herrera, M A Márquez. *Easy Handling of Sensors and Actuators over TCP/IP Networks by Open Source Hardware/Software*[J]. Sensors, 2017, 17(1).