Journal of Zhejiang University SCIENCE A
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn


# An application-layer based centralized information access control for VPN*

OUYANG Kai (欧阳凯), ZHOU Jing-li (周敬利), XIA Tao (夏 涛), YU Sheng-sheng (余胜生)

(*School of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan 430074, China*)

E-mail: oykai@mail.hust.edu.cn; ljzhou@mail.hust.edu.cn; xiatao@mail.hust.edu.cn; ssyu@mail.hust.edu.cn

Received Dec. 23, 2004; revision accepted Apr. 4, 2005

**Abstract:** With the rapid development of Virtual Private Network (VPN), many companies and organizations use VPN to implement their private communication. Traditionally, VPN uses security protocols to protect the confidentiality of data, the message integrity and the endpoint authentication. One core technique of VPN is tunneling, by which clients can access the internal servers traversing VPN. However, the tunneling technique also introduces a concealed security hole. It is possible that if one vicious user can establish tunneling by the VPN server, he can compromise the internal servers behind the VPN server. So this paper presents a novel Application-layer based Centralized Information Access Control (ACIAC) for VPN to solve this problem. To implement an efficient, flexible and multi-decision access control model, we present two key techniques to ACIAC—the centralized management mechanism and the stream-based access control. Firstly, we implement the information center and the constraints/events center for ACIAC. By the two centers, we can provide an abstract access control mechanism, and the material access control can be decided dynamically by the ACIAC's constraint/event mechanism. Then we logically classify the VPN communication traffic into the access stream and the data stream so that we can tightly couple the features of VPN communication with the access control model. We also provide the design of our ACIAC prototype in this paper.

**Key words:** Virtual private network, Access control, Tunneling, Centralized management, Stream
**doi:**10.1631/jzus.2006.A0240 **Document code:** A **CLC number:** TP393.02

## INTRODUCTION AND BACKGROUND

In the last decade, as the Internet becomes a popular low-cost backbone infrastructure, many organizations and companies use it to establish their secure private network, which is known as VPN (Virtual Private Network) technology (Cohen, 2003). Generally, VPN implements confidentiality of data, message integrity and endpoint authentication by the security protocols (such as IPSec: IP Security (Kent and Atkinson, 1998) and TLS/SSL: Transport Layer Security/Secure Socket Layer (Dierks and Allen, 1999)), and implements the private addressing by the tunneling technique. Due to the tunneling of VPN server shown in Fig.1, however, vicious users can bypass the control of the firewall by the use of VPN Server and compromise interior servers. Furthermore, even though VPN has endpoint authentication to prevent invalid users from accessing those servers, the whole interior servers' topology would be exposed to every trusting user because there is no access control model for VPN. Hence, establishing access control model for VPN is the key technique of high-security VPN architecture.

Traditionally, the research on access control is classified into two aspects: access control model and the security policy architecture. The classic access model includes the B-LP model, the Biba model (Verschuren *et al*., 1992), RBAC (Role Based Access Control) (Sandhu *et al.*, 1996) and CDAC (Content Dependent Access Control) (Moffett and Sloman, 1991). Spencer *et al*.(1999) presented the Flask security architecture—the operating system security

* Project (No. 60373088) supported by the National Natural Science Foundation of China

architecture. This research was the prelude to the dynamical security policy framework. Bertino *et al*.(2002) described a core specification language of an extensible access control system, called MACS (Multipolicy Access Control System), along which different access control policies can co-exist.

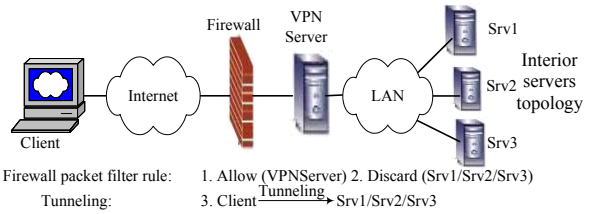


**Fig.1 The basic framework for VPN**

There were further researches on the access control technique for VPN in recent years. Jason *et al*.(2003) presented an object-oriented information model of IPSec policy designed to facilitate agreement on the content and semantics of IPSec policy, and to enable derivations of task-specific representations of IPSec policy such as storage schema, distribution representations, and policy specification languages used to configure IPSec-enabled endpoints. Sanchez and Condell (2002) proposed a protocol called SPP, which can systematically resolve IPSec policies with Policy Servers. Those researches were almost all focused on IP-layer, but few were focused on application-layer.

This paper focuses on the application-layer based access control model for VPN. Compared with IP-layer based access control model, it has two advantages as follows:

(1) Fine-grained. In the application-layer, we could implement not only the control work in IP-layer, but also more complex access constraints for VPN (e.g. the access decision is made according to the cooperation of the user name, client source, destination, time and the type of application-layer protocol). Further, we can parse the content and context of an application-layer protocol (such as HTTP) and implement the access decision.

(2) Guard against network virus and intrusion. In application-layer, we can establish the corresponding constraints based on the virus characteristic and the IDS (Intrusion Detection System) rule to protect the internal servers. We cite for instance a well-known SQL Injection for the Web-based database system. If we set a constraint in the application-layer to inspect application traffic and filter all dangerous packets on VPN server, we can disconnect the routine between the client and the Web-based database system. When we set a constraint in the IP-layer, it is hard to inspect the content of application traffic.

Hence, we present a novel application-layer based access control model—ACIAC (Application-layer based Centralized Information Access Control), aiming at providing an efficient, flexible and secure application-layer based access control model for VPN. To achieve the goal, we present two new ACIAC techniques: centralized management and the stream-based access control. We first implement the information center and the constraints/events center for ACIAC. The two centers can be used to reveal the dynamic multi-decision mechanism for ACIAC, which is composed of the features of multi-access control models, such as UBAC, RBAC and CDAC. Then, we logically classify the VPN communication traffic into the access stream and the data stream so that we can tightly couple the features of VPN communication with the access control model.

The remainder of the paper is organized as follows. In Section 2, we detail the essential features of ACIAC and discuss the control of ACIAC. Section 3 presents the prototype of ACIAC and discusses the relationship between the logical modules and the key techniques of the prototype. In Section 4, we describe the current related researches. Section 5 summarizes this paper.

## ESSENTIAL OF ACIAC

Based on the theory of set and relation, we describe the terminology, rules, control and feature of ACIAC in this section.

### Definition and terminology

**Definition 1** Stream *S* refers to as all communication traffic traversing the VPN server, and can be classified into Access Stream (*AS*) and Data Stream (*DS*). The traffic traversing the VPN server when the users access VPN or the tunnel is established is *AS*. After one tunneling is established, all traffic in this

tunneling is *DS*. Any stream belonging to one particular user is expressed as *u*(*S*/*AS*/*DS*).

Through the above classification, ACIAC can achieve the fine-grained access control and avoid the unnecessary execution of constraints. We constrain *AS* to implement the control of the access to the interior server and constrain *DS* to implement the internal control of one particular server.

**Definition 2** All kinds of sets and chains in ACIAC are defined as follows:

(1) All the attributes of objects in any VPN stream should be described as the information set {*Info*}.

(2) All the constraints in any VPN stream should be described as the constraint set {*Cons*}.

(3) The event set {*Event*} is a series of special actions, which usually cause one special constraint chain to change dynamically.

(4) The information set in any constraint/event can be classified into the subject set {*Sub*} and the object set {*Obj*}. {*Sub*} represents the information that must be decided dynamically in one constraint. {*Obj*} represents the static information in one constraint, which is configured or decided by the administrator.

(5) Chain *List*(*x*) is the management form, by which ACIAC can organize {*Info*} with a special logic relationship. *x* refers to subjects, objects, constraints or events.

(6) The user set *U* describes all the users registered in VPN, and the role set *R* defines different groups of users. Each group of users has the same logical privilege. Before any user can execute any operation, he must be authenticated and authorized *A*(*u*). In addition, there is a special user *Admin*, the superuser for ACIAC.

(7) The actions triggered by the results of the constraints are described as {*Action*}. In ACIAC model, there are four types of actions: *allow*, *discard*, *filter* and *pend*. *allow*: The streams can be transmitted between clients and the internal servers through the VPN server. *discard*: Because of the privilege restriction, the streams cannot be transmitted through the VPN server, which results in the disconnection of this tunneling or the logout of this user. *filter*: Because of the streams' invalid content or particular requirements, the streams cannot be transmitted through the VPN server, and cannot cause the disconnection of this tunneling or the logout of this user. *pend*: When one constraint cannot be executed because the current conditions are not satisfied, ACIAC marks this case as a token *pend*, which cannot be evidence for access control.

**Definition 3** In ACIAC, we formulate the information element as <*name*, *value*, *List*(*R*, *Access*), *List*(*U*, *Access*)>. *name* is the name of the information element. *value* is the abstract description of the element's value, whose type is decided when *name* is defined. *Access* is the control privilege set of one element, *Access*={*none*, *read*, *write*, *own*}. *List*(*R*, *Access*) is the relationship between the role set and the access privilege to the element. *List*(*U*, *Access*) is the relationship between the user set and the access privilege of the element.

**Definition 4** The privilege level is formulated as $Level(x)=\{r\in R\}\times\{u\in U\}$. In ACIAC, we adopt the hybrid judgment mechanism, which includes the role privilege and the user privilege. The access to one information element $info_1$ by a user $u_1$ belonging to the role $r_1$ is allowed on both the conditions below:

(1) $r_1$ has enough privilege to access $info_1$ and $info_1$'s attribution does not include the *none* privilege for $u_1$;

(2) $r_1$ has not enough privilege to access $info_1$ but $info_1$'s attributions definitely include the *read*, *write* or *own* privilege for $u_1$.

Otherwise, this access is not allowed.

**Definition 5** The formula for one stream process is expressed as:

$$\{\forall u\in A(u),u(List(s))\subset\{Sub\}\}\xrightarrow[u(List(e))\subset\{Event\}]{u(List(c))\subset\{Cons\}}\{u(List(o))\subset\{Obj\}\}.$$

When one constraint or event registers in ACIAC, it must provide its subject list *List*(*s*), object list *List*(*o*) and constraints'/events' routine for the ACIAC model. In one stream transmitting process, according to the stream's attribution, ACIAC gets all information and call every constraint's and event's routine in the current *u*(*List*(*c*)) and *u*(*List*(*e*)).

**Rules**

**Rule 1** The constraint's or event's control routine for any stream must be executed only if the user, to whom the stream belongs, is authenticated and authorized.

$$\{\forall u\in A(u),\ \forall s\in u(s),\ \forall c\in u(List(c))\cup\forall e\in u(List(e))e\}\Rightarrow Exec(c\cup e).$$

**Rule 2** Any modification to the value of any information element in any stream is done only if the stream's privilege is not less than that of the information element. The same judgment applies to the elements of {*Cons*} and {*Event*}.

$$\{\forall u\in A(u),\ \forall s\in u(S),\ \forall i\in\{Info\},\ Level(s)\geq Level(i)\}\Rightarrow Modify(i).$$

**Rule 3** One constraint may have many instances, but one user has one instance of one constraint at most.

$$\{\forall u\in A(u),\ \forall c\in u(List(c))\}\Rightarrow\{u(Entity(c))\equiv 1,\ Entity(c)\geq 1\}.$$

**Rule 4** {*Event*} is managed centrally by the ACIAC model and shared by all valid privilege users. When one user $u_1$ is authenticated and authorized, ACIAC organizes $u(List(e))$ for $u_1$ according to Rule 2 and every event in $u(List(e))$ is triggered by the user's stream $u(S)$. $u_1$ does not own any event in $u(List(e))$, but refers to event.

$$\{\forall u\in A(u),\forall s\in u(S)\}\xrightarrow{triger}\{\forall e\in u(List(e))\},$$
$$\{\forall e\in\{Event\}\}\Rightarrow\{u(Entity(e))\equiv Entity(e)\equiv 1\}.$$

**Rule 5** After one user $u_1$ is authenticated and authorized, $u_1$'s constraint list $u(List(c))$ is managed through the user's events $u(List(e))$ created by Rule 4.

$$\{\forall u\in A(u),\ \forall e\in u(List(e))\}\xrightarrow{e}Manage(u(List(c)).$$

**Rule 6** After one user $u_1$ is authenticated and authorized, he will maintain a copy of $u(List(s))$ expressed as <*name*, *value*> for his subject information $u(List(s))$ involved in the constraints list and events list. The same subject element has different values in different user space, but the same object element has the same value in different user space.

$$\{\forall u_1\in A(u),\forall u_2\in A(u),\forall s_1\in u_1(List(s)),\ \forall s_2\in u_2(List(s)),name(s_1)=name(s_2)\}\Rightarrow value(s_1)=value(s_2),$$

$$\{\forall u_1\in A(u),\forall u_2\in A(u),\forall o_1\in u_1(List(o)),\ \forall o_2\in u_2(List(o)),name(o_1)=name(o_2)\}\Rightarrow value(o_1)=value(o_2).$$

**Rule 7** In the process of ACIAC executing the constraints list $u(List(c))$ to control the transmission of the current stream, if the result of any constraint is *discard* or *filter*, ACIAC need not implement the remainder of the constraints in $u(List(c))$. This stream can be passed only if all the result s are *allow* or *pend*.

$$\{\forall u\in A(u)\xrightarrow{\forall c\in u(List(c))}\forall action\notin\{discard,filter\}\}\Rightarrow Pass(u(S)).$$

**Control mechanism**

ACIAC is the stream based control model. We take the process of one user using VPN for example; ACIAC includes these logic control modules: the entry of the user, the establishment of tunneling, the data stream control and the logout of the user.

1. The entry of the user

After the user $u_1$ is authenticated and authorized, ACIAC will dynamically create a new valid ID $A(u_1)$ for $u_1$ (the valid ID is the global, exclusive and irreversible value), initialize the user's event list $u(List(e))$, add 1 to the referenced counter of any event in $u(List(e))$, and call every event's registered routine. When one user enters VPN, ACIAC must trigger the initial event (usually the first event) to build the user's constraints list $u(List(c))$, copy the constraints' and events' subject elements <*name*, *value*> to the user's private space, and reset the values of those copies.

2. The establishment of tunneling

After one user $u_1$ enters the VPN architecture, $u_1$ can establish one tunnel to access one internal server. When one tunnel is established, ACIAC first checks the validity of this request, then calls routines for events and constraints in $u(List(e))$ and $u(List(c))$.

3. The data stream control

After the tunnel is established, ACIAC can control the traffic between the client and the internal server. Because all data streams must pass through the VPN server, ACIAC can analyze those streams content and context through events and constraints. The mechanism of the analysis is similar to that of Content Dependent Access Control (CDAC) model and the

Context Based Access Control (CBAC) model. We take the access control for an internal Web server as an example. After the tunnel from the client $client_1$ to the Web server $serv_1$ is established, ACIAC will filter some special URLs requested by $client_1$ and analyze the request or response packets according to the characteristics of the virus and intrusion. Furthermore, ACIAC can determine whether $client_1$ is allowed to access a resource according to the sequence of status. ACIAC exclusively manages one set of detailed access controls to a data stream so that those controls can only be applied in one unique tunnel, and avoid unnecessary control operations in other tunnels.

4. The logout of the user

When one valid user $u_1$ wants to leave the VPN session, ACIAC will call the events' routines in $u(List(e))$ and subtract 1 from the referenced counters of those events. Usually, the last event ACIAC called is the destruction event, in which ACIAC will clear all session information of $u_1$ and all related resources.

One demonstration of ACIAC shown in Fig.2 consists of one user $u_1$ who has been authenticated and authorized, one tunnel $tunneling_1$, $u_1$'s events list $u(List(e))$ and $u(TUN(List(e)))$, and $u_1$'s constraints list $u(List(c))$ and $u(TUN(List(c)))$. It shows the relationship among information, subjects list, objects list, constraints list, events list and user/role (all interactions must be done under the control of those above rules).

It also indicates that ACIAC has two centralized repositories, one to store all information and the other to store events and constraints. Both the subject and the object come from the information repositories. After ACIAC creates one new valid ID for $u_1$, he will

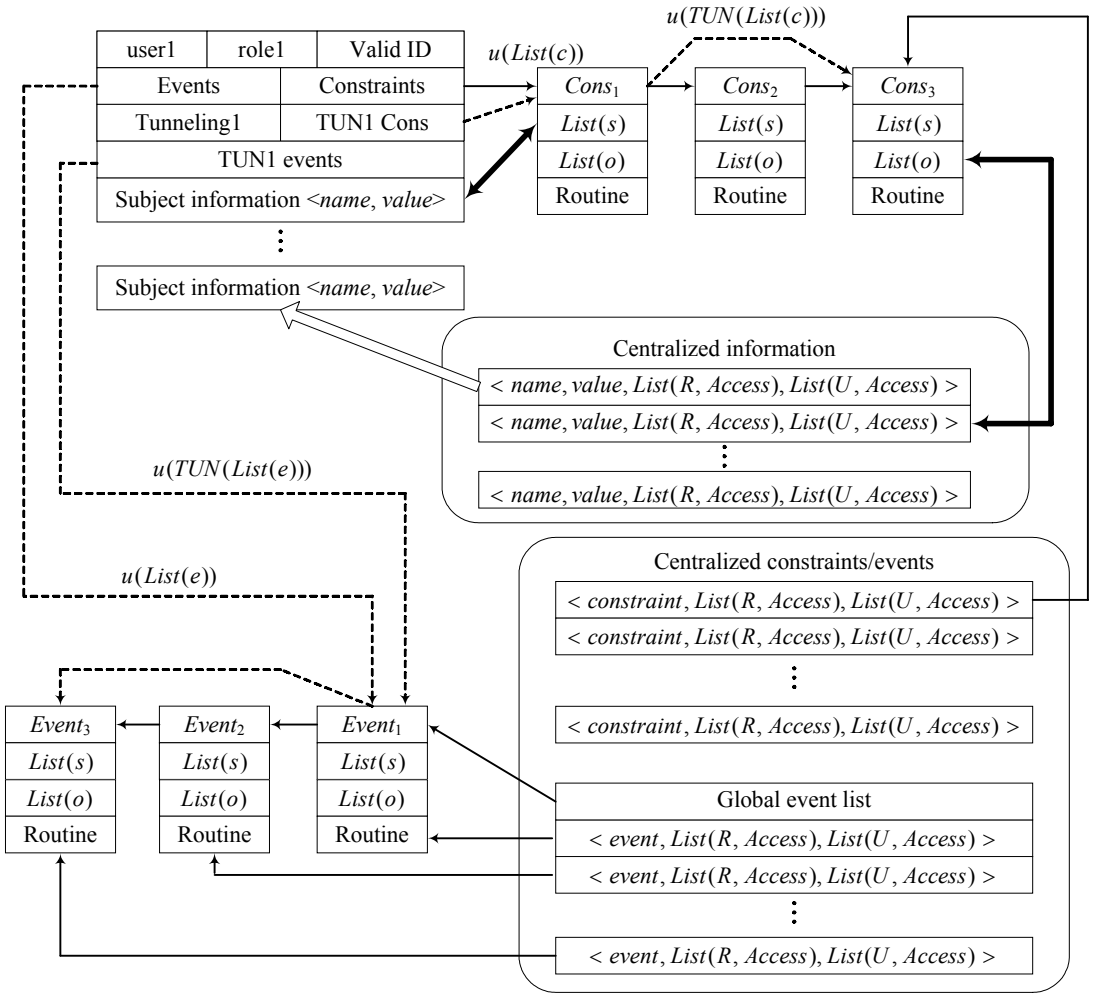


**Fig.2 One demonstration of ACIAC**

copy the needed subject elements from the information center to his private space and reset the value of every subject element. When a constraint or event needs the object, ACIAC retrieves the object from the information repository. ACIAC stores the registered constraints and events information as <*constraint*, *List*(*R*, *Access*), *List*(*U*, *Access*)>/<*event*, *List*(*R*, *Access*), *List*(*U*, *Access*)> in the constraints/events center. ACIAC also maintains a global events entity list *List*(*e*). As *u*(*List*(*e*)) shows in Fig.2, in $u_1$'s initialization process, ACIAC creates the referenced event list ($Event_1$ and $Event_3$) for $u_1$, and $u_1$ creates his own constraints list ($Cons_1$, $Cons_2$ and $Cons_3$) through his initial event.

When $tunneling_1$ has been established, ACIAC will modify $u_1$'s constraints list *u*(*List*(*c*)) and events list *u*(*List*(*e*)), as *u*(*TUN*(*List*(*c*))) and *u*(*TUN*(*List*(*e*))) shown in Fig.2. Furthermore, different tunnels may have different constraint/event lists for the data stream verification and filtering. Each distinct constraint/event list is a unique subclasses instance of *u*(*List*(*c*))/*u*(*List*(*e*)).

**Characteristics**

From the above discussion, we see that the ACIAC model is tightly coupled with the characteristics of VPN communication with the access control technique. It provides a fine-grained access control mechanism for VPN server and integrates the features of RBAC/UBAC, CDAC and CBAC.

We can conclude describing the characteristics of ACIAC as follows:

(1) The centralized object management. We formulate the concrete resources as information elements and abstract these access controls to constraints and events. Both information and constraints/events are stored in ACIAC's repositories, which we can expediently manage and avoid antinomies in different control routines, for example, when both $Cons_1$ and $Cons_2$ need the value of $info_1$ to determine the access privilege, and $info_1$ is currently changed by the administrator. The value of $info_1$ should be conveyed to them automatically, because both $Cons_1$ and $Cons_2$ access $info_1$'s value from ACIAC's centralized information repository.

(2) The stream based access control. According to Definition 1, ACIAC divides all the VPN communication traffic into the control stream and the data stream. The objective of this division is to implement fine-grained access control and avoid unnecessary constraint operations so as to improve the performance of ACIAC. We will prove this claim in the follows paragraphs.

Assumption: in one tunnel, the average duration of one stream transmitting through the VPN server is $T_{\text{trans}}$, the average duration of one access control (constraint or event) is $T_{\text{control}}$ and the total number of access controls is *Num*.

When there is no access control, the duration of one transition is:

$$T_{\text{no\_control}}=T_{\text{trans}}.$$

When there are access controls but the access control model does not distinguish the properties of streams, the duration of one transition is:

$$T_{\text{total}}=T_{\text{trans}}+Num\times T_{\text{control}}.$$

From the above discussion, we know that the operation of any access control can lower the VPN server's performance.

But when we use ACIAC to control the stream, the *AS* number of access controls is *Num*(*AS*(*List*(*c*))+*AS*(*List*(*e*))) and the *DS* number of access controls is *Num*(*DS*(*List*(*c*))+*DS*(*List*(*e*))), so the duration of one transition is:

$$T_{DS}=T_{\text{trans}}+Num(DS(List(c))+DS(List(e)))\times T_{\text{control}},$$
$$T_{AS}=T_{\text{trans}}+Num(AS(List(c))+AS(List(e)))\times T_{\text{control}}.$$

Distinctly, the duration of one transition in ACIAC is less than that in other models without dividing traffic. Based on the analysis of VPN access control, there is little control used in both *AS* and *DS*. Hence, ACIAC can reduce the loss of performance caused by the access control model.

(3) The event-driven dynamic management mechanism. ACIAC uses event-driven model to implement its management mechanism. After the user is authenticated and authorized, ACIAC organizes the event list for him. During the whole lifetime of the user's session, his status is controlled by the event-driven model. In ACIAC, the status of the user is very simple:

$$Status=\{init, update, work, terminal\},$$

where *init* denotes that when the user becomes a validated user, ACIAC will do the initialization for him, *update* means that during the user's VPN session, ACIAC updates his constraints/events list and the value of any information element, *work* means that the user is doing normal work, such as accessing internal servers, *terminal* is a special *update*, in which ACIAC will terminate the user's session and reclaim all resources allocated for the user.

(4) The user/role based hybrid privilege mechanism. Apparently, if we just use the user privilege to manage the access level, we need a very complicated privilege management; if we just use the role privilege to manage the access level, it is hard to satisfy the requirements of all users. Hence, ACIAC uses the hybrid privilege mechanism as described in Definition 4. For example, there is one role, $r_1$, which contains two users $u_1$ and $u_2$, and the other role, $r_2$, and two information elements $info_1$ and $info_2$ accessed only by $r_2$. If we want to allow $u_1$ to access the $info_1$, but do not want to expose $info_1$ to other users of $r_1$ (such as $u_2$) nor give $u_1$ the privilege to access $info_2$, ACIAC can achieve the requirement through setting the access privilege to $info_1$ for $u_1$ definitely.

Ongoing research activities in RBAC are as follows. Steinmuller and Safarik (2001) extended RBAC with states aimed at include the notion of states and state transitions into the RBAC model and to view changes of components of RBAC model as transitions between states of one access control policy. Ferraiolo *et al.*(2001) provided the first proposed NIST standard for RBAC. Furthermore, Wolf *et al.*(2003) showed how RBAC concepts can be applied to model cases in which identification mechanisms can be used as a parameter to be evaluated in access control.

## PROTOTYPE OF ACIAC

In this section we will provide the module design of our ACIAC prototype and discuss all the modules of the prototype and the relationship among these modules.

As shown in Fig.3, the modules of the ACIAC prototype include the Scheduler, the Valid User Manager (VUM), the Access Streams Parse Controller (ASPC) and the Administration Module (AM). They cooperate with each other to implement the access control technique for VPN.

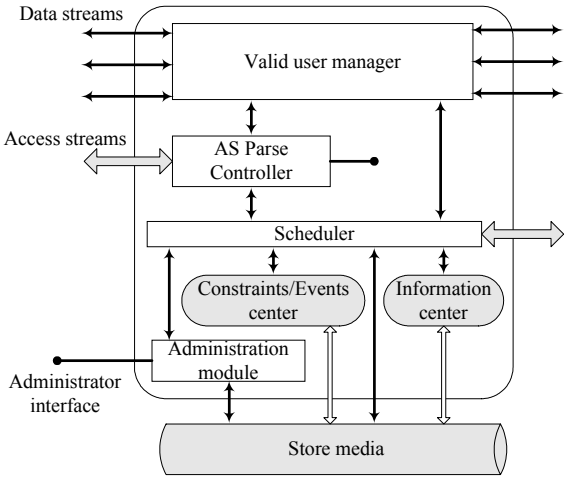


**Fig.3 The prototype of ACIAC**

### Scheduler

The functional organizations of the ACIAC scheduler include the Valid ID Creator, the Global Event Controller, the Privilege Level Controller and Synchronous Controller, as shown in Fig.4.

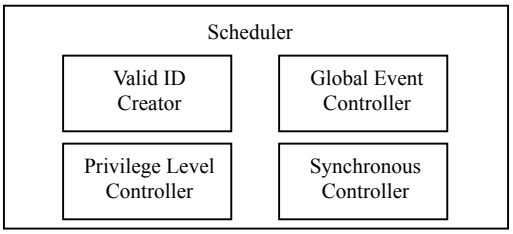


**Fig.4 The functional organizations of Scheduler**

After one user is authenticated and authorized, ACIAC calls the Valid ID Creator to create a new ID for him. The new ID is created by a collision-free hash function whose seed is composed of the current time and a random number. Since the event-entities list is a global list, which can be referred to by valid users under the privilege level control, ACIAC uses the Global Event Controller to manage it. Moreover, the Privilege Level Controller implements the ACIAC's hybrid judgment mechanism, which is described in Definition 4. Because the ACIAC prototype is a multi-threads/multi-processes system and there are many shared resources, we use the Synchronous Controller to implement the consistency and

integrality of those shared resources, such as information center and constraints/events center.

**AS parse controller**

The functionality of ASPC is straightforward. As shown in Fig.5, ASPC classifies *AS* into the unknown *AS* and the known *AS*. The unknown *AS* means that ACIAC does not know who owns the *AS* because there are no valid IDs in the *AS*. Hence, ASPC will notify the VPN Authentication Server through its Authentication Interface. After the user is authenticated and authorized, ASPC can notify the Scheduler to create a new ID and do initialization for the user. If access stream is a known *AS*, ASPC uses its Callback Controller to notify VUM, then VUM will find the user owning this stream, and call his $u(List(e))$ and $u(List(c))$ to deal with this stream.

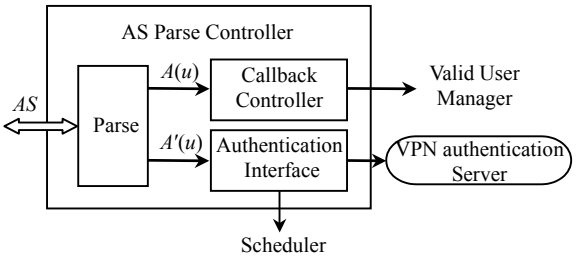


**Fig.5 The basic flow of AS Parse Controller**

**Valid user manager**

The internal organization of VUM is shown in Fig.6. VUM manages every valid user in the form of the user session entity. Each user entity has its own constraints list, events list, subjects list, the Tunneling Controller and the Objects Cache. The Tunneling Controller in VUM is different from the VPN tunneling. The object of the Tunneling Controller is to optimize the access control mechanism for data streams, by which VUM can avoid unnecessary access control operation in one tunneling.

The Object Cache is used to store the copy of the recently used object elements <*name*, *value*>, which is similar to the elements in subjects list. Compared with {*Obj*} in ACIAC, object elements, one user's use of $u(\{Obj\})$ is very limited and always centers on a small scope. Hence, VUM can save some cost used to search objects in the information center by the Objects Cache.

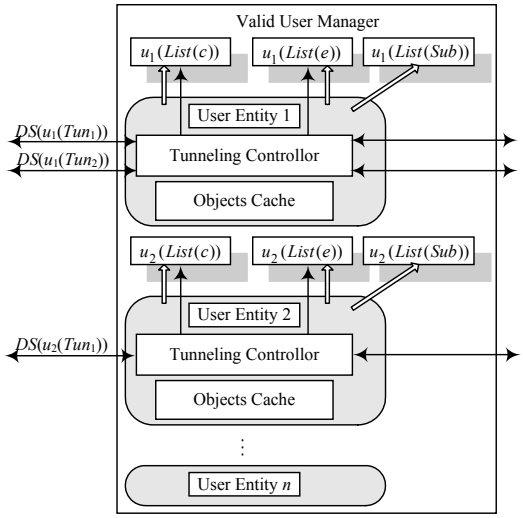


**Fig.6 The internal organizations of Valid User Manager**

**Administration module**

The internal organizations of AM include the Web Interface, the Users/Roles Administration, the Events/Constraints Administration, the Information Administration, the Notification Mechanism and the Store Interface, as shown in Fig.7.

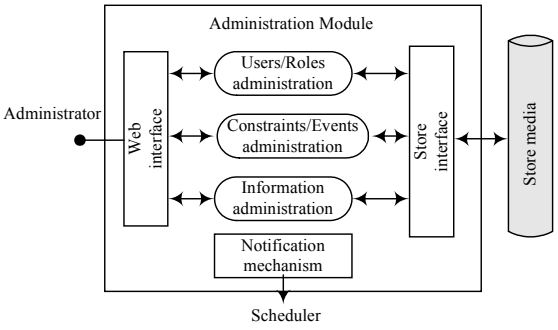


**Fig.7 The internal organizations of Administration module**

The Web Interface provides the Web browser-based administration mechanism for the administrator to manage the registration/deregistration and modification of ACIAC's users, roles, events, constraints and information for administrator. The Store Interface provides the transform mechanism from the media form to the memory form for these resources. ACIAC adopts the XML (Extensible Markup Language) technique to implement the management of these resources. Any change of those resources can trigger

AM's Notification Mechanism, which can notify the Scheduler of the change of some resources. The Scheduler can synchronize the change if necessary.

## RELATED RESEARCHES AND ANALYSIS

Access control for network, by the broadest definition, is to implement the ultimate goal of all network security—granting access when appropriate and denying it when inappropriate. In this section, we will discuss the recent researches on access control technique for VPN and compare them with ACIAC.

As we have shown in the "Characteristic" session, the conventional RBAC and CDAC can provide sufficient level of fine-grained access control for all users' AC requirements. A separate AC module is required for RBAC and CDAC to manage the access control to a server providing different application services. It is an administrator's nightmare to maintain all applications' access control without a centralized architecture.

The existing application-layer firewall can hardly work with any type of VPN technology as all data transmitted in VPN tunnels are encrypted. The application-layer firewall also lacks centralized fine-grained access control, which is the core of our ACIAC.

Guo *et al*.(2003) presented a policy-based network management system for IP VPN in the ICCT 2003 conference. Its basic object is to implement an IP-layer based access control for VPN through PBNM (Policy-Based Network Management) (Wang, 2000). PBNM mainly includes four components: policy management tool, policy repository, Policy Decision Point (PDP) and Policy Enforcement Point (PEP). And it is based on FreeS/WAN IPSec that is a Linux implementation of the IPSec protocols. To implement this system, Guo *et al*.(2003), therefore, mainly provided three mechanisms in the PDP component to implement this system: the Service Level Agreement (SLA) Management, the Key Management and the Internet Key Exchange (IKE) Management. They also provided a high-level policy definition language for the system in order that the administrator adds and changes policies in the policy repository, and designs an object oriented information model to represent the IP VPN management policies. Generally they put emphasis on the network management level of IP VPN. Ku *et al*.(2002) presented the design and implementation of Web-based IP-VPN policy deployment manager (PDM), which was developed aiming at helping ISP network administration in VPN system deployment and management. Yague *et al*.(2003) presented the application of Semantic Web concepts and technologies to the access control area. They designed the Semantic Access Control Model (SAC) that uses different layers of metadata to take advantage of the semantics of the different components relevant to the access decision. Compared with the above researches, ACIAC has one outstanding characteristic: ACIAC is not the access model used in VPN, but the access model tightly coupled with VPN, in which every access decision is based on the VPN stream.

On the other hand, considering that traditional access control mechanisms have little ability to support or respond to the detection of attacks. Ryutov *et al*.(2003) presented a generic authorization framework that supports security policies that can detect attempted and actual security breaches and can actively respond by modifying security policies dynamically. We also consider the disadvantage of the current intrusion detection and anti-virus systems working in isolation from access control for the application the systems aim to protect. Hence, in the ACIAC design, we use the features of the CDAC/CBAC working mechanism to implement the intrusion detection and anti-virus functions by the ACIAC's constraints/events mechanism. Furthermore, because these protection mechanisms for the internal application servers always work after the VPN tunneling is established, we can optimize the access decision by classifying the constraints/events into *AS*'s and *DS*'s.

## CONCLUSION

The application-layer based centralized information access control model (ACIAC) is the outcome of our application-layer based VPN architecture. Based on the analyses of current access control models and the working features of VPN, we present the design of ACIAC for VPN and discuss the definitions, rules and control mechanism of ACIAC. Firstly,

ACIAC is based on the VPN communication stream so that it could tightly couple with VPN. Secondly, any access control can be decided by users' attributes, subjects' attributes or objects' attributes. Thirdly, ACIAC is not only the access control model but also the intrusion detection and anti-virus system.

Furthermore, we provide our prototype of ACIAC. In this prototype, we detail the design of the logical modules and point out the key techniques for implementing an efficient and flexible system.

Compared with other related researches, we think our application-layer based centralized information access control model is a novel tool for research on the VPN access control and management.

## ACKNOWLEDGEMENT

These ideas resulted from many helpful discussions with Zhang Ming, Liu Wei and Dong Li-jun. We would also like to thank Tang Fang and Guo Hui for their valuable suggestions to improve the paper. The researchers in the System Architecture Department of Huazhong University of Science and Technology thank the National Natural Science Foundation for its support and funding.

## References

Bertino, E., Catania, B., Ferrari, E., Perlasca, P., 2002. A System to Specify and Manage Multipolicy Access Control Models. Policies for Distributed Systems and Networks, p.116-127.

Cohen, R., 2003. On the establishment of an access VPN in broadband access networks. *Communications Magazine*, *IEEE*, **41**(2):156-163.

Dierks, T., Allen, C., 1999. The TLS Protocol Version 1.0. RFC2246.

Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramouli, R., 2001. Proposed NIST standard for role-based access control. *ACM Trans. Inform. and System Security*, **4**(3):224-274. [doi:10.1145/501978.501980]

Guo, X., Yang, K., Galis, A., Cheng, X., Yang, B., Liu, D., 2003. A Policy-based Network Management System for IP VPN. Communication Technology Proceedings. ICCT 2003, **2**:1630-1633.

Jason, J., Rafalow, L., Vyncke, E., 2003. IPSec Configuration Policy Information Model. RFC3585.

Kent, S., Atkinson, R., 1998. Security Architecture for the Internet Protocol. RFC2401.

Ku, H., Son, H.G., Facsko, J., Tyrrell, J., Haines, A., 2002. Web-based Policy Deployment Management System. Proceedings of Policies for Distributed Systems and Networks, p.240-243.

Moffett, M.D., Sloman, M.S., 1991. Content-dependent access control. *ACM SIGOPS Operating Systems Review*, **25**(2):63-70. [doi:10.1145/122120.122125]

Ryutov, T., Neuman, C., Dongho, K., 2003. Integrated access control and intrusion detection for Web servers. *IEEE Trans. on Parallel and Distributed Systems*, **14**(9): 841-850. [doi:10.1109/TPDS.2003.1233707]

Sanchez, L., Condell, M., 2002. Security Policy Specification Language. Internet Draft, http://www.csie.nctu.edu.tw/~jkzao/Publication/draft-ietf-ipsec-spsl-01.pdf.

Sandhu, R.S., Coyne, E.J., Feinstein, H., Youman, C., 1996. Role-based access control models. *IEEE Computer*, **29**(2):38-47.

Spencer, R., Smalley, S., Loscocco, P., Hibler, M., Andersen, D., Lepreau, J., 1999. The Flask Security Architecture: System Support for Diverse Security Policies. Proceedings of the Eighth Security Symposium, p.123-139.

Steinmuller, B., Safarik, J., 2001. Extending Role-based Access Control Model with States. EUROCON'2001, International Conference on Trends in Communications, **2**:398-399.

Verschuren, J., Govaerts, R., Vandewalle, J., 1992. Simultaneous Enforcement of the Bell-LaPadula and the Biba Security Policy Models in an OSI-distributed System. ICCS/ISITA'92, Singapore, p.257-263.

Wang, C., 2000. Policy-based Network Management. Communication Technology Proceedings. ICCT 2000, **1**:101-105.

Wolf, R., Keinz, T., Schneider, M., 2003. A Model for Content-dependent Access Control for Web-based Services with Role-based Approach. Database and Expert Systems Applications, Proceedings 14th International Workshop, p.209-214.

Yague, M.I., Mana, A., Lopez, J., Troya, J.M., 2003. Applying the Semantic Web Layers to Access Control. Proceedings of Database and Expert Systems Applications, p.622-626.