BANDS: AN INTER-DOMAIN INTERNET SECURITY POLICY MANAGEMENT SYSTEM FOR IPSEC/VPN

Yanyan Yang[1], Zhi (Judy) Fu[2], and S. Felix Wu[1]
[1]Department of Computer Science, University of California, Davis, CA 95616, USA, {yyyang,sfwu}@ucdavis.edu, [2]Network and Infrastructure Research Lab (NIRL), Motorola Labs, USA, jfu@labs.mot.com

Abstract: IPSec/VPN is widely deployed for users to remotely access their corporate data. IPSec policies must be correctly set up for VPN to provide anticipated protection. Manual policy setup is unscalable, inefficient and error-prone. Automated policy generation to comply with and enforce high-level security policies is desired but difficult, especially in an inter-domain environment when a VPN traverse multiple domains. This paper presents a distributed framework and protocol, BANDS, for inter-domain policy negotiation and generation. The BANDS architecture consists of two phases: AS (Autonomous System) route path discovery and an inter-domain collaborative protocol for policy negotiation among the autonomous systems discovered in the first phase. Each AS conceptually has one security requirement server responsible for the task of inter-domain policy negotiation. Following this two-step process in BANDS, a set of distributed security policies (for the implementation of policy enforcement) will be automatically negotiated/generated based on decentralized and predefined security requirements.

Key words: Inter-domain Security Management, Security Policy Management, IPSec/VPN

1. INTRODUCTION

1.1 Security Management for a Remote/Mobile Layer-3 Network Node

We had encountered various difficulties when we plugged in our laptop computers in a foreign domain during a trip, as shown in Figure 1. We did not know

the security policies along the route path from our laptops (then being connected to a remote layer-3 network) or we did not even know what the route path was. In our home layer-3 network, our own laptops might be protected by an intrusion detection system as well as some other security counter measures, but, in a foreign domain, we in general know very little. When our traffic caused conflicts with "some" security policies along the route path, we did not know what the root cause was (or even target security gateway). Maybe our destination server was down during that time. Or, possibly, our SSH requests to certain restricted sites have been dropped by a particular IPSec/Firewall/NAT gateway on the way. In [7], we have discussed how such conflicts can occur in the context of IPSec/VPN and firewall.

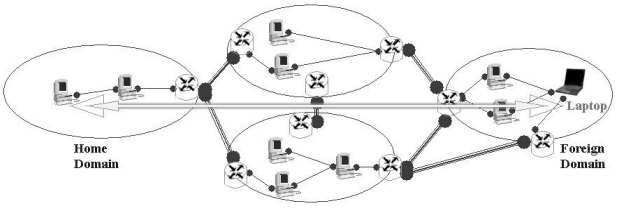

Figure 1. Security management for a remote network node

What we really want is a plug-and-play Internet security management solution. When we plug in a computer, a laptop, or any layer-3 addressable node under some Internet/VPN access points, within a couple minutes or shorter, our connections to some corresponding destinations will be established correctly and securely.

One short-term solution to this problem is to establish a bi-directional secure virtual tunnel from the remote laptop to a "home agent" (similar to MobileIP) in the home layer-3 network. This tunnel may be a L2VPN, a remote IPSec tunnel, a L2TP tunnel, a SSH tunnel, or their combination. This approach has two major shortcomings. First, the traffic will go through sub-optimal route paths, as they must travel through the home network in both directions. Second, with this approach, the traffic from the remote laptop will be properly protected by the home network security gateways because the security gateways will treat them as "trusted home traffic". But, if the laptop (even if it is at home) wants to connect to a "new" corresponding destination under a different administrative domain, it is not clear whether some new "policy conflicts" will arise as we don't know the security policies at the other end of the communication at that particular moment.

1.2 IPSec/VPN Security Policy Management

Internet has been more and more "dynamic" in many ways. In order to provide secure communications for end-to-end connections, IPSec (Internet Security Protocol Suite) [10] policies are widely deployed in firewalls/gateways to restrict access or selectively enforce security operations. Currently, most commercial IPSec implementations require a manual policy configuration process, which is inefficient and error-prone. Some IPSec management products (such as CISCO or NETSCREEN) provide a policy distribution system to allow a centralized policy server to distribute policies to all the IPSec devices in the same administrative domain as shown in Figure 2. However, for all the venders we have talked to, none of their products can provide any "correctness" assurance [7] about the IPSec/VPN policy rules stored in the policy repository in the first place. In other words, currently, even in an intra-domain environment, commercial IPSec/VPN policy tools cannot generate or validate a set of provably correct policy rules.

For the purpose of security policy formal analysis, in [1][7], we proposed a separation principle between security policy and requirement. Based on this principle, system administrators can unambiguously specify each individual security model (or individual security requirement) in our policy language. Then, the collection of such security requirements can be formally and efficiently analyzed for its correctness and completeness properties. Furthermore, a set of IPSec security policy rules will be automatically produced for all the PDP (Policy Decision Point) / PEP (Policy Enforcement Point) devices.

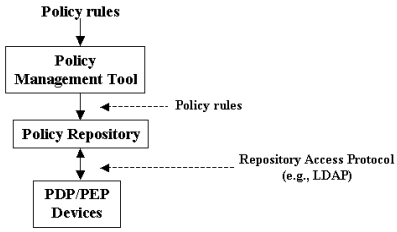

Figure 2. A sample policy distribution system

While our earlier work [1] provides a rigorous framework for security policy management in an intra-domain environment, we do not yet have a good solution for securing end-to-end connections across multiple administrative domains. Therefore, the key contribution in this paper is a distributed and yet scaleable architecture and protocol for inter-domain IPSec/VPN security policy management. The rest of the paper is structured as follows. Section 2 defines the problem of inter-domain security policy management. In Section 3, we briefly review the related work. Then, we present how to solve our target problem as well as different components under our solution in Section 4. Section 5 gives an example scenario to illustrate the cooperation and interoperation of each component under the BANDS architecture. We will also present some preliminary performance evaluation results in Section 6. Finally, in Section 7, we summarize the paper and outline some future works.

2. TERMINOLOGY AND PROBLEM DEFINITION

2.1 Security Policy versus Requirement

Traditionally there is no rigorous definition of security requirements and security policies. As a result, the relationship between them is so vague that the correctness of security policies cannot be formally and automatically substantiated. The needs to distinguish high-level security requirements and low-level policies were addressed in [4][5]. Once the high-level requirements are specified/modified, it should be possible to determine what kind of low-level policies must be created/changed.

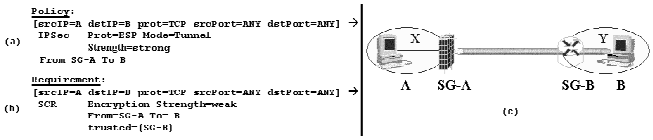

Figure 3. An example of policy (a), requirement (b) under certain network topology (c)

Therefore, under the BANDS architecture, a two-level security requirement/policy model is used. The word "policy" means "How should a network entity or a policy domain handle a particular flow of packets" in the context of

IPSec/VPN. In other words, policy is the command interface between a system administrator and a network device such that the human administrator can instruct the device to perform certain IPSec/VPN related operations. Once a "policy" rule is defined, a network device can unambiguously process the packet flow, including both packet headers and payloads. The left in Figure 3 is an example of policy. It specifies that the TCP traffic from A to B will be encrypted through an ESP Tunnel SA using the triple DES algorithm. And, the unidirectional SA starts at the security gateway SG-A and ends at B itself. This "low-level" policy specifies how SG-A and B should process the TCP packets from A to B.

On the other hand, in the context of this paper, the word "requirement" means "How should a sequence of information bits (the original payload) be handled from the source to the destination" regardless of any possible IP/IPSec header transformation on the route path (e.g., IPSec Transport or Tunnel mode, NAT or NAPT, IP fragmentation and de-fragmentation). In other words, "requirement" (sometimes we call it "high-level" policy) expresses the administrator's (or the user's) intentions about the security of some end-to-end information bits across different administrative domains without concerning low-level security operations. For instance, in the above SCR (Security Coverage Requirement) shown in Figure 3, the system administrator specifies that the TCP traffic between A and B must be encrypted between SG-A and B. And, furthermore, it might be OK to let SG-B to examine the content (for the purpose of intrusion detection, for example).

Policy and requirement are not one-to-one mapping. Usually, one requirement can be satisfied by a set of low-level security policy rules. As shown in [1][7], it is possible to find multiple security policy sets and any one of them can satisfy the target security requirement equally well.

In BANDS, IPSec/VPN security requirements have four different types:

- **Access Control Requirement (ACR):** ACR is related to a security gateway or firewall's access control function to some trusted traffic.
- **Security Coverage Requirement (SCR):** SCR applies security mechanism to prevent traffic from being compromised during the transmission across certain area. It requires the security protection to cover all links and nodes within the certain area. Various algorithms of authentication and encryption can be specified as the parameters in low-level policies.
- **Content Access Requirement (CAR):** Some network nodes may need to access the content of certain traffic, yet the content cannot be viewed if an encryption tunnel is built to across it. For example, a content access policy can be defined to deny all the encrypted traffic.
- **Security Associate Requirement (SAR):** Security Associations (SA) must be formed to perform desired security functions, thus there is a need to specify that some node desire to or not to set up SA with other nodes. Network peers are allowed to build SA unless explicitly disallowed.

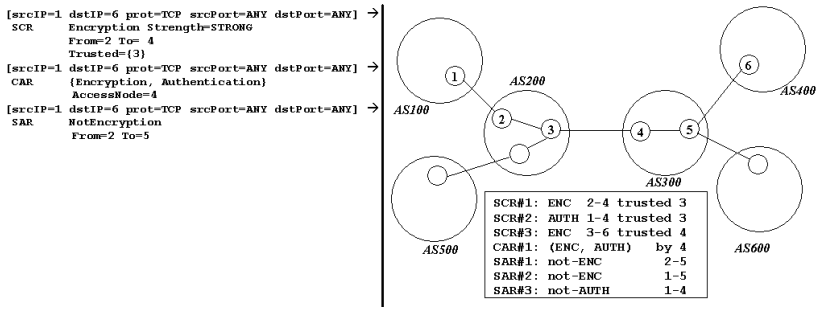

Figure 4. An end-to-end flow with 7 security requirements

Based on the definitions, the following policy solution shown in Figure 5 (we use a linear picture for an intuitive view) satisfies all the requirements. In order to provide flow protection to satisfy ENC SCRs, two encryption tunnels are built between 1 and 4, and between 4 and 5. Similarly, to satisfy AUTH SCR, two authentication tunnels are built between 1 and 3, and between 3 and 4. Obviously, all the tunnels are built to guarantee the certain protection, without violating any of the CARs and SARs. Due to the size limit, we only show the formal definition of one of the policies as follows:

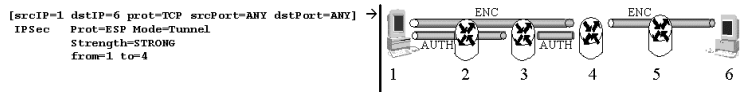

Figure 5. The policy solution

However, the following scenario as shown in Figure 6 may happen sometimes. Each policy satisfies its corresponding requirement, while putting all the policies together may cause conflicts. In this example, the flow is tunneled to 3 with authentication. On the other hand, it is tunneled from 2 to 4 with encryption before the authentication tunnel exits. It's easy to see that the authentication function applies at 1 and will not be de-applied at the tunnel 2 to 4. Thus, the traffic will be encapsulated by 1 for authentication and be encapsulated again by 2 for encryption. When 4 decapsulates and finds out that the destination is 3, the flow will be sent back to 3. Eventually, 3 will decapsulate the flow and send it to its destination. As a result, the flow is sent in plaintext from 3 to 4 because of the tunnel interaction, which violates the original security intentions (requirements). This is one of the reasons to avoid overlapping tunnels in BANDS architecture.

Figure 6. Overlapping Tunnels that cause conflicts

2.2 Inter-Domain Security Requirement Engineering

Under the BANDS framework, a system administrator and a user will use the SRSL (Security Requirement Specification Language) to specify one or more requirements related to either a particular domain (such as AS) or an information flow/bundle. Then, based on all the requirements we have in the Internet, in [7], we show that we can automatically and efficiently generate a set of IPSec security policy rules such that all requirements will be satisfied. Furthermore, if there are any conflicts among the requirements such that it is impossible to find a policy set to satisfy all the requirements, our program can detect such a case as well.

However, in an inter-domain environment such as Internet, it is practically and politically impossible, very inefficient, and un-scaleable to "collect all the requirements in the whole system" and then perform the task of requirement analysis and policy generation. One trick we can do under this situation is to determine the exact "route" path from the source of the information flow to the destination. Then, we can collect the requirements along the route path and then determine how to set up the security policy rules to satisfy all the security requirements along the route path. Furthermore, if BANDS detects that it is impossible to satisfy all the requirements along one particular route path, it might be able to try another route path (for example, in the case of multi-homing). Finally, we need to worry about "routing dynamics" in the Internet. Whenever the route is changed, BANDS needs to decide whether our current policy set has been affected or not. If necessary, we

need to re-collect the requirements and re-compute the policies such that the security will not be affected by the routing dynamics.

3. RELATED WORK

Currently, there are mainly two works related to our research, MSME project at BBN Technologies and one of our research work --- Celestial system at NCSU.

– BBN's MSME architecture

MSME (Multidimensional Security Management and Enforcement) [2] is a research project being conducted at BBN technologies. It presents multidimensional architecture to allow each member in the distributed system to maintain its own policy management system while enabling him to exchange and resolve policies with other members of the coalition. MSME uses one level policy model to achieve the correctness of policy management, because there were vague definitions of security requirements and policies. The high-level security requirement is defined in an abstract format and somehow is mapped into a binding of the implementation, i.e. security policy. When the policy agreements are exchanged, the policies are complied to determine any conflict and to resolve it (if any). It is easy to see that, in MSME architecture, security requirements are exposed when the agreements are exchanged. Yet, in the network nowadays, people anticipate to keep the requirement information sharing as minimal as possible.

– Celestial system

The other research work dealing with security policy management is called Celestial system [3] at NCSU, which was designed to automatically discover security policies along the network path and dynamically configure security mechanisms across the network. Similarly, the Celestial system does not have explicit definition between security requirements and security policies. It just addresses every node's security capabilities and policies, and the receiver computes the corresponding policy strategy. Furthermore, the Celestial architecture is an unscalable and pure centralized security management system, indeed.

4. BANDS: A SECURITY POLICY MANAGEMENT SYSTEM ARCHITECTURE

4.1 Architecture overview

As we addressed in Section 2, there is a need to separate high-level requirements from low-level policies. Therefore, the ultimate aim is to be able to define high-level requirements beforehand and to automatically generate the low-level policies. On the other hand, the information shared must be kept as minimal as possible because we require only relevant information to be exposed. In order to achieve maximum autonomy, the principle of providing policy implementers with everything they need to know to satisfy the relevant requirements, but nothing more, should be respected. Furthermore, today's Internet acts like a huge distributed system. Therefore a pure centralized network management model is not ideal enough to provide reliable and scalable service, which could also guarantee the minimal information sharing. Therefore, the architecture that we developed adopts a hybrid structure of centralized and distributed systems. One of the important roles in the architecture is that every domain (i.e. AS) contains a Requirement Server (RS), which is responsible for cooperation and policy negotiation with other RSs in a distributed environment, as shown in Figure 7. Based on the architecture of RS, a two-phase

policy negotiation process is preceded by each RS to generate correct policies automatically.

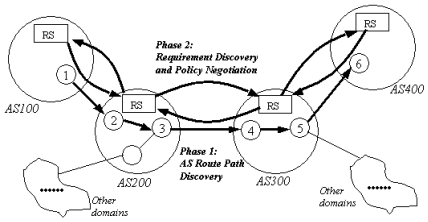

Figure 7. An architecture example in a multiple-domain environment

The first step of the overall protocol is to discover the AS route path, in order to get the RS of each AS involved along the path ready for the incoming negotiation process. This phase is called "route path discovery", explained in Section 4.4. Based on the discovered AS route path, each RS should be able to identify the IP addresses (e.g. with DNS servers or LDAP) of other RSs along the path, such that the RS in the original AS could find out the corresponding security requirements along the path and exchange those which are relevant to each other. Each RS needs to make queries to its "neighbor" RS along the path for requirement discovery request. This requirement discovery phase is followed by policy negotiation. When the RS receives the negotiation message from the remote RS, it computes the corresponding policies using the automatical policy generation algorithm, direct approach [1], as explained in Section 4.5. Eventually, each RS notifies its local routers the security policies for a certain flow. Hence, under BANDS, we only introduce and add a requirement server in each domain, which stores routing information, maintains requirement information and performs policy negotiation, while routers remain unchanged. From the routers' point of view, the operations of BANDS operation are transparent, because the routers still carry out the regular router operations. A router performs its corresponding action only when it receives the policies from its RS.

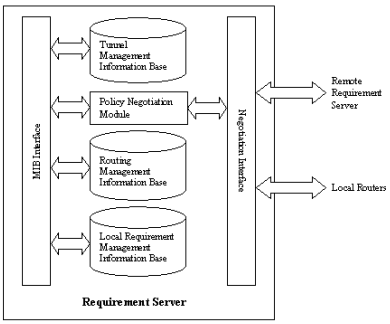

Figure 8. The architecture of a Requirement Server

Figure 8 illustrates each AS's RS's architecture, which has several sub-components to interoperate with each other to precede the requirement discovery and policy negotiation with remote RSs and has interfaces to access its routing information, requirement information and tunnel information databases.

4.2 Components

The functionality of the components shown in Figure 8 is described in the following sections.

a) Routing Management Information Base (Routing MIB) and Local Requirement Management Information Base (Local Requirement MIB)

All the routing information of local routers is stored in each RS's Routing MIB such that the RS will be able to calculate the route path within the AS and the AS path across the AS. The RS needs to maintain periodical routing update from each of the routers in the AS through its Information Base Interface. Similarly, all the security requirements of local routers are stored in each RS's Local Requirement MIB. Together with the information in the Routing MIB, the Local Requirement MIB provides the RS the capability of computing the security policy for each local router for different flows. The RS also needs to maintain periodical requirement update from each of the routers in the AS through its MIB Interface. And the consistent maintenance is implemented using SNMP.

Each router runs an SNMP agent to maintain a local database of its security/routing requirements. Each RS, as SNMP management station, must have some SNMP management software, which is running one or more processes to communicate with the SNMP agents within the local domain (i.e. in the same AS) using SNMP protocol, in order to query the state of an agent's local requirement/routing information (under PKI infrastructure if necessary).

b) Tunnel Management Information Base (Tunnel MIB)

Since each RS knows what local routers establish what kinds of tunnels with what remote routers, every RS should be able to make query to each of its neighbors to find out what are the existing tunnels that are related to certain flows. In another word, with the information in Tunnel MIB, the RS is capable of building a tunnel map to certain flow, with which it could easily find out all the relevant security requirements and all the relevant existing tunnels to run the direct approach algorithm. In addition, the RS needs to maintain periodical tunnel update from each of the remote RSs because it must delete relevant information from its Tunnel MIB when some tunnel has been withdrawn.

c) Policy Negotiation Module (PolNegM)

With PolNegM, the RS negotiates with the remote RS for a set of security policies. Through Negotiation Interface, the PolNegM not only receives the policy negotiation requests from the remote RS and responds with its relevant requirement information to the remote RS, but also informs the appropriate security policy to each of the routers, which will participate along the route path. It will need to access the Routing MIB and the Local Requirement MIB to obtain the routing information for the flow and to gather the requirement details for each of the local routers on the route path of the flow. After collecting all the information, including the routing information and existing tunnel information, the PolNegM runs algorithm for each of the security coverage requirements to obtain the policy solution and then notifies both the local router and the remote router for the SA establishment.

4.3 Interface between Modules

a) MIB interface

The Information Base Interface provides the access interface between the database and the Policy Negotiation Module in the requirement server module. When the PolNegM needs to access the Routing MIB, it sends the query to the MIB Interface. The interface forwards the message to Routing MIB and sends the response back to PolNegM. The same procedure applies to the communication

between PolNegM and Local Requirement MIB, and between PolNegM and Routing MIB.

b) Negotiation Interface

The Negotiation Interface provides the communication interface between the requirement server and the remote requirement server to negotiate security policy. When the RS contacts the remote RS to exchange requirements and to negotiate the policies, the Negotiation Interface forwards the negotiation message to the remote RS's PolNegM component through its Negotiation Interface

4.4 AS Route Path Discovery

The overall protocol consists of two phases, "AS route path discovery" phase and "collaborative policy negotiation protocol" phase. Before an end-to-end connection can be established securely, the RS in the original AS needs to initiate the "AS router path discovery" phase to explore the AS route path to discover all the RSs that will be involved in the subsequent policy negotiation phase.

Depending on what routing mechanism is used, the route path discovery strategies could vary. For instance, Border Gateway Reservation Protocol (BGRP) [11] is an inter-domain aggregated resource reservation protocol for unicast traffic, in which a sink tree is built for each of the stub domains to perform a destination-based reservation aggregation as shown in the example in Figure 9. If we use it under BANDS architecture, the overall protocol starts a route path discovery phase by sending a BGRP PROBE message to the destination. After the initiator gets a GRAFT message back, the exact AS route path has been probed and reserved. Each RS that will be involved in the following protocol then needs to sustain the flow information and launch the PolNegM to prepare for the negotiation.

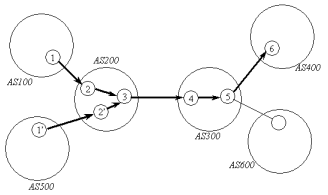

Figure 9. An example of a BGRP reservation sink tree rooted at router 6

4.5 Algorithm: direct approach

Before we dive into the next section for the detailed collaborative negotiation protocol, we will briefly introduce our automatic policy generation algorithm in each RS under BANDS, i.e. direct approach presented in [1], an efficient and scalable way for calculating policies. The algorithm considers one SCR at one time and takes in other relevant requirements (e.g. CARs and SARs) as parameters. By knowing the exact route path and the existing tunnels along the path, it computes the solutions to satisfy one SCR and related requirements without violating any existing tunnels.

To ensure that the generated policy satisfies one SCR without violating corresponding CARs and SARs, the algorithm starts with the initial (node) graph with full connection. To remove CAR conflicts, it eliminates all the links crossing the nodes that have CARs. Then it deletes all the links that starts or ends at the distrusted nodes. Finally among the rest of the edges, it uses Dijkstra shortest path algorithm to get the final tunnel solution. To better understand the algorithm, we use the previous scenario (SCR #2) in Section 2.1 as an example. Figure 10 (a) presents

the initial primary graph and Figure 10 (b) shows the graph after CAR conflict check (i.e. after removing all the edges crossing node 4). Then Figure 10 (c) takes away all of the distrusted edges (i.e. edges that starts or ends at node 2, the distrusted node) and Figure 10 (d) comes with the final solution.

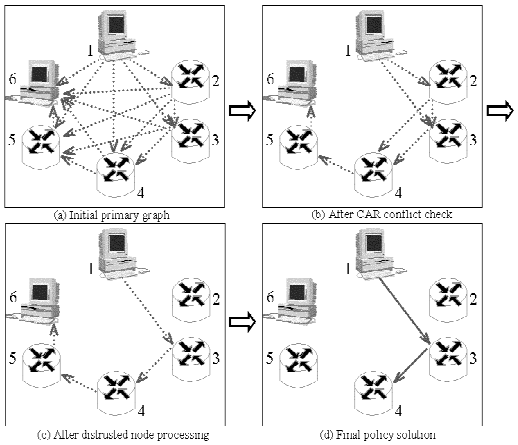

Figure 10. An example of direct approach

4.6 The Collaborative Negotiation Protocol

The architecture is designed to provide reliable and secure end-to-end connections in a distributed environment. On one hand, the exact AS route path for a certain flow needs to be discovered. On the other hand, after the route path is explored, the sub-components of each RS along the path must collaborate and negotiate to figure out the appropriate tunnel (policy) solutions to guarantee that all the security requirements along the route path are satisfied. Our collaborative negotiation protocol consists of two steps.

1. Requirement Discovery Phase

```
Procedure RequirementDiscovery(flow)
1.  /* Get the route path of the flow through MIB Interface */
2.  routepath =MIBI_Get_Route_Path(flow)
3.  /* Check for all CARs on the path through MIB Interface*/
4.  CARList = MIBI_Get_CAR_List(flow, routepath)
5.  /* If any CARs, send to previous Requirement Server through Negotiation Interface */
6.  For every CAR in CARList
7.      NI_Send_Req(CAR, PreRS)
8.  /* Check for all SCRs on the path through MIBI */
9.  SCRList = MIBI_Get_SCR_List(flow, routepath)
10. /* Requirement Aggregation */
11. For every SCR in SCRList
12.    if SCR. strength <. flow. strength
13.    then SCR. strength = flow. strength
14.    if flow.ReqAggregate = 1
15.   then SCR.from = PreRS
16.          NI_Send_Req(SCR, PreRS)
17.          MIBI_Remove_Req(SCR)
18.          flow.ReqAggregate = 0
19. If no SCR between myself and next RS
20. then flow.ReqAggregate = 1
End of Procedure
```

```
Procedure GetRequirements(Req, flow )
1.  /* For CAR, insert into Local Requirement MIB */
2.  /* For SCR, if it is for myself, change SCR.from and insert into Local Requir
3.  /* otherwise send to previous RS */
4.  if (Req is SCR and ReqAggregate = 0)
5.  then Req.from = MIBI_Get_Reg_From(Req, flow)
6.  MIBI_Insert_Req(Req)
End of Procedure
```

Figure 11. The pseudo code of Requirement Discovery

After the AS route path is discovered, the RS in the original AS should start phase II to find out the requirements. The RS in the first AS sends out a "Requirement Discovery" message to the RS in the next AS, including the target security requirements for this end-to-end flow. It is easy for the RS to find out what

are the local requirements for this certain flow in its Local Requirement MIB. The RS launches its PolNegM to prepare for the incoming negotiation request from remote RS. When the "Requirement Discovery" message reaches another RS in another AS, the RS needs to be prepared to participate in the negotiation and find out what are the related requirements for the flow. Furthermore, if the flow requirements could be aggregated with certain local requirements, the RS should be able to make the required aggregation and notify the RS, that may be affected accordingly, to update the requirements. If a router on the route path is the one which has CARs, its RS needs to advertise its CARs to its previous RS through its Negotiation Interface and the latter propagates them to every RS along the path. Therefore, for CARs, every RS on the path knows who are the ones that need to access the content for certain flow. For SARs, because SAR is the security requirement that two network entities cannot establish Security Associate between each other, both RSs should maintain the SARs. When the destination RS gets the "Requirement Discovery" message, it will respond a confirmation message to the sender and this step finishes. The pseudo code of the phase is as follows in Figure 11. It should be noted that solid synchronization mechanism should ensure the inter-domain requirement consistency.

2. Policy Negotiation Phase

Next, the initiator RS sends out a "Policy Negotiation" message to the destination RS. Accordingly, the PolNegM checks its Routing MIB to get the route path and queries Local Requirement MIB to obtain the relevant requirements. With the interact tunnel information in the Tunnel MIB, the PolNegM runs the direct approach algorithm to determine what policy set it will be using. After the solution is obtained, the RS must inform its neighbor and the next RS on the route path, the intended policy solution, which will be considered as the existing tunnel as the input parameters for the direct approach algorithm. If there is no appropriate solution available, the RS must send an error message back to the original RS to either adjust the requirements or withdraw the negotiation process. The pseudo code of this phase is as above in Figure 12.

```
Procedure PolicyNegotiation(flow)
1.  /* Generate policies for each SCR */
2.  /* Get the route path of the flow through MIB Interface */
3.  routepath = MIBI_Get_Route_Path(flow)
4.  /* Check for all SCRs on the path through MIB Interface */
5.  SCRList = MIBI_Get_SCR_List(flow, routepath)
6.  /* Check for all the existing interacted tunnels through MIB Interface */
7.  ExistTunnels = MIBI_Get_Exist_Tunnels(flow, routepath)
8.  /* Run the direct approach to generate the tunnel */
9.  /* If no solution found, send a fail message to previous RS */
10. For every SCR in SCRList
11.     policy = NonoverlappingDirectApproach(flow, SCR, ExistTunnels)
12.     if (policy = NULL) NI_Send_Fail(PreRS)
13.     Insert_Policy(policy, PolicyList)
14. /* If a confirmation is received, notify local routers of policies */
15. /* otherwise send a fail message back */
16. If a confirmation is received
17. then NI_Notify_Router(policy)
18. else NI_Send_Fail(PreRS)
End of Procedure
```

Figure 12. The pseudo code of Policy Negotiation

When the "Policy Negotiation" message reaches the receiver and no error occurs, the receiver responds with a confirmation message to the sender. As a result, every RS along the path knows the solution plan is feasible and it will inform the corresponding local router so that the router could initiate Internet Key Exchange (IKE) to exchange the session key between itself and the certain remote router and then set up IPSec Security Associate with the remote router. The specifications of IKE SA and IPSec SA establishment are in [9] [10].

The overall protocol finishes when the corresponding routers start to establish Security Associations with remote routers. Apparently, phase I guarantees the determination of the exact route path and make resource reservation on it if necessary. Next, all the RSs participate in the policy negotiation, including

discovering all of the relevant requirements and calculating the policies. Thus certain security mechanism and functions could be applied to corresponding area/links to protect the flow. The security requirement information revealed in the protocol are just those CARs on the route path that every RS must know to avoid CAR violation during the policy computation and the existing tunnel information that is only related to the flow.

5. EXAMPLE SCENARIO

The following example demonstrates the whole operation of the various modules in the architecture and explains how the collaborative negotiation protocol works among the network peers. Suppose that there are 6 autonomous systems in the network using BGRP, each of which has an RS and some routers. Also, we have some security requirements[1] at relevant routers as three SCRs (ENC, ordinary, 2, 4, {3}), (ENC, strong, 3, 6, {4}) and (AUTH, middle, 1, 4, {3}), one CAR (ENC, AUTH, 4) and three SARs (2, 5, ENC), (1, 4, AUTH) and (1, 5, ENC).

1. Discover the route path

Router 1 needs to communicate with router 6 through a strong encryption mechanism, in our example. It initiates a route path discovery phase by sending a BGRP PROBE message to router 6. After router 1 gets a GRAFT message back from router 6, the exact path has been probed and reserved as shown in Figure 13.

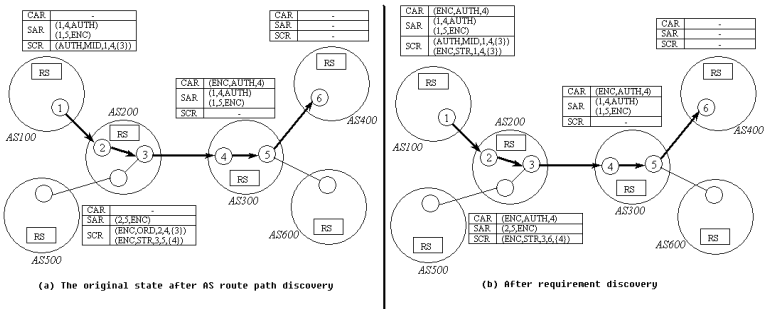

Figure 13. Example scenario: The original state after AS route path discovery (a) and after requirement discovery (b)

2. Discover the requirements

Now the RS in AS100 should begin the second phase with the requirement discovery. It sends the "Requirement Discovery" message to next RS in AS200 to see if there is any related requirement the latter has to share with it. The RS in AS200 figures out that one of its SCR (ENC, ORD, 2, 4, {3}) could be aggregated with the original flow requirement to be (ENC, STR, 1, 4, {3}). As the SCR.from is router 1, the RS in AS200 must inform the RS in AS100 of the newly aggregated requirement so that the latter must consider this new requirement when calculating the policy while the RS in AS200 must not. When the RS in AS300 receives the message and notices that there is a CAR related to the flow (ENC, AUTH, 4), therefore, it passes the CAR to its previous RS (in AS200) and the RS propagates it to the RS (in AS100) through its Negotiation Interface. Until the message reaches the other end (the RS in AS400), the requirement discovery part finishes with the last RS sending back a confirmation message.

[1] For simplicity, we use the abbreviation MID, ORD, STR for MIDDLE, ORDINARY, STRONG respectively as one of the parameters of the requirements in the figure.

3. Negotiate the policies

The RS in AS100 gets the confirmation message and thus it could run the PolNegM to calculate the policy solutions shown in Figure 14 (a) (we use a linear picture for a intuitive view). Similarly, the RS in AS200 runs the algorithm in the PolNegM to figure out the policy solution shown in Figure 14 (b). Therefore, the overall security policy solution is as follows in Figure 14 (c).

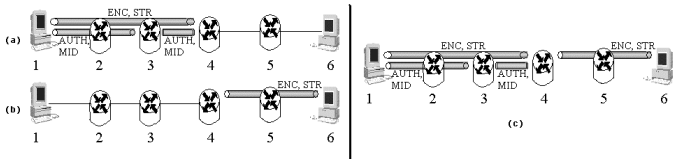

Figure 14. Example scenario: The policy solution computed by the RS in AS100 (a) and AS 200 (b) and the final policy solution (c)

6. EVALUATION

We have implemented a simulation program for the BANDS architecture. Under a simulated network topology, our program takes security requirements file as input and outputs IPSec/VPN policy rules. According to our preliminary performance results shown in Figure 15, while the number of security requirements increased dramatically as well as the number of messages transmitted during the whole process, the number of automatically produced security policies (the actual number of IPSec tunnels that will be built) increased linearly with the number of requirements. Because in the direct approach, one SCR requirement is considered at a time to compute the corresponding policy solution, our algorithm could be considered as a requirement-based algorithm. Certain requirement-based aggregation will be performed during "requirement discovery phase", as we show in the pseudo code. In summary, the BANDS framework from our simulation-based evaluation is indeed very scaleable and practical. Currently, our simulation program does not simulate the dynamic routing aspect of BANDS, and therefore, the route discovery part is omitted. More real network experiments involving routing are with development, in order to test the correctness and scalability of our architecture.

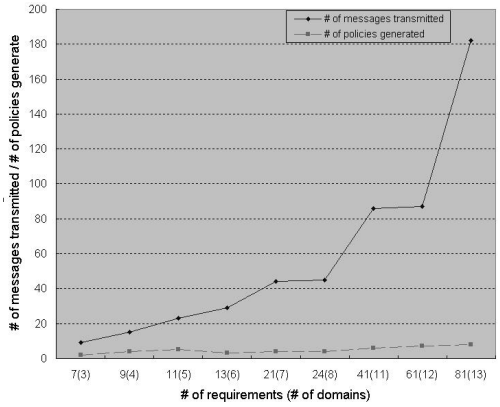

Figure 15. Experimental results

7. CONCLUSIONS

We presented a distributed architecture and a collaborative negotiation protocol for inter-domain Internet security policy management. We introduced the separation principle between security policies and requirements such that we can express the security intentions unambiguously. Furthermore, under the BANDS framework, the low-level policies are automatically generated with high-level requirements defined in advance. As a result, with the requirement server in each autonomous system in an inter-domain environment, we could manage the local requirements and calculate the corresponding policies for a certain flow. The overall protocol includes "AS route path discovery" phase and "collaborative negotiation protocol". After the AS route path is discovered, each RS along the AS path participates in the requirement discovery phase, followed by policy negotiation phase, which are both called the overall collaborative policy negotiation protocol. During the negotiation phase, the requirement information shared is kept as minimal as possible, so that only requirements that are pertinent to the flow are revealed to others.

While the performance simulation we have so far is only for static routing on some given end-to-end connections, our preliminary results demonstrate that our framework is very scaleable with respect to the number of automatically generated policy rules in an inter-domain networking environment. In the future, we will extend our evaluation to a prototype implementation on a routing network test-bed.

REFERENCES

[1] Z. Fu and S. F. Wu, "Automatic Generation of IPSEC/VPN Policies in an Intra-Domain Environment", 12th International Workshop on Distributed Systems: Operations & Management (DSOM 2001), October 15-17, 2001, Nancy, France.

[2] M. Condell, G. Patz, R. Krishnan, C. Lynn, "MSME Architecture", December 17, 2001, BBN Technologies.

[3] C. Xu, F. Gong, I. Baldine, C. Sargor, F. Jou, S. F. Wu, Z. Fu, H. Huang, "Celestial Security Management System", DARPA Information Survivability Conference and Exposition(DISCEX2000),IEEE Computer Society Press, Proceedings, pp.162-172, vol. 1.

[4] Moffett, J. D., "Requirements and Policies", Position paper for Workshop on Policies in Distributed Systems, 15-17 November 1999, HP- Laboratories, Bristol, UK.

[5] Moffett, J. D., Sloman, M. S., "Policy Hierarchies for Distributed Systems Management", IEEE Journal on Selected Areas in Communication, 11(9), 1404-1414.

[6] Z. Fu, "Automatic Generation of Security Policies", Technical Report, http://shang.csc.ncsu.edu/secpolicy.pdf.

[7] Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, "IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution", IEEE Policy 2001 Workshop, Jan. 2001.

[8] S. Blake et al., "An architecture for differentiated service," RFC 2475, Internet Engineering Task Force, Dec.1998.

[9] Dan Harkins et al., "Proposal for the IKEv2 Protocol", Internet Draft, <draft-ietf-ipsec-ikev2-02.txt>, April 2002.

[10] S. Kent et al., "Security Architecture for the Internet Protocol", RFC 2401, Internet Engineering Task Force, November 1998.

[11] P. Pan, E, Hahne, and H. Schulzrinne, "BGRP: A Tree-Based Aggregation Protocol for Inter-domain Reservations", Journal of Communications and Networks, Vol. 2, No. 2, June 2000, pp. 157-167.