Network Working Group
Internet-Draft
Intended status: Experimental
Expires: December 3, 2007

J. Bi
J. Wu
L. Xie
CERNET
June 2007

A Signature based Source Address Validation Method for IPv6 Edge Network

draft-bi-sava-solution-ipv6-edge-network-signature-00.txt

Status of this Memo

Copyright Notice

Abstract

Today's Internet is suffering from reflection-type DoS/DDoS attacks, such as TCP SYN flood, Smurf attack, DNS reflection, etc. These attacks often rely on the IP source address spoofing. Current source address spoofing prevention methods have a coarse granularity, and the attackers can still in some cases launch reflection-type DoS/DDos attacks or other attacks while being difficult to trace. Aiming to solve this problem, this document proposes a fine-granularity source address spoofing prevention method, which can prevent the attackers from forging source addresses that belong to the same edge network to send packets to somewhere outside the IPv6 edge network. The proposed method includes source address authentication using session key and hash digest algorithms and replay attack prevention by combining a sequence number method with a timestamp method.

Table of Contents

1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Problem Statement

The source address spoofing prevention approaches based on filtering (e.g. ingress filtering) can validate the source address in real-time. Current approaches based on filtering, however, only validate the IP source address according to the network prefix, so these approaches cannot prevent IP source address spoofing within edge network. Thus, they leave some room to launch reflection-type DoS/DDoS attacks and make traceback uncertain. For instance, if the Internet were to deploy ingress filtering on a large scale, an attacker would be prevented from forging an IP source address belonging to any other edge network, defeating most reflection-type DoS/DDoS attacks. However, even with ingress filtering [RFC2827], the attacker can still forge an IP source address that belongs to the same edge network and attack the victim using the reflection-type DoS/DDoS. This type attack scenario is shown in Fig.1:

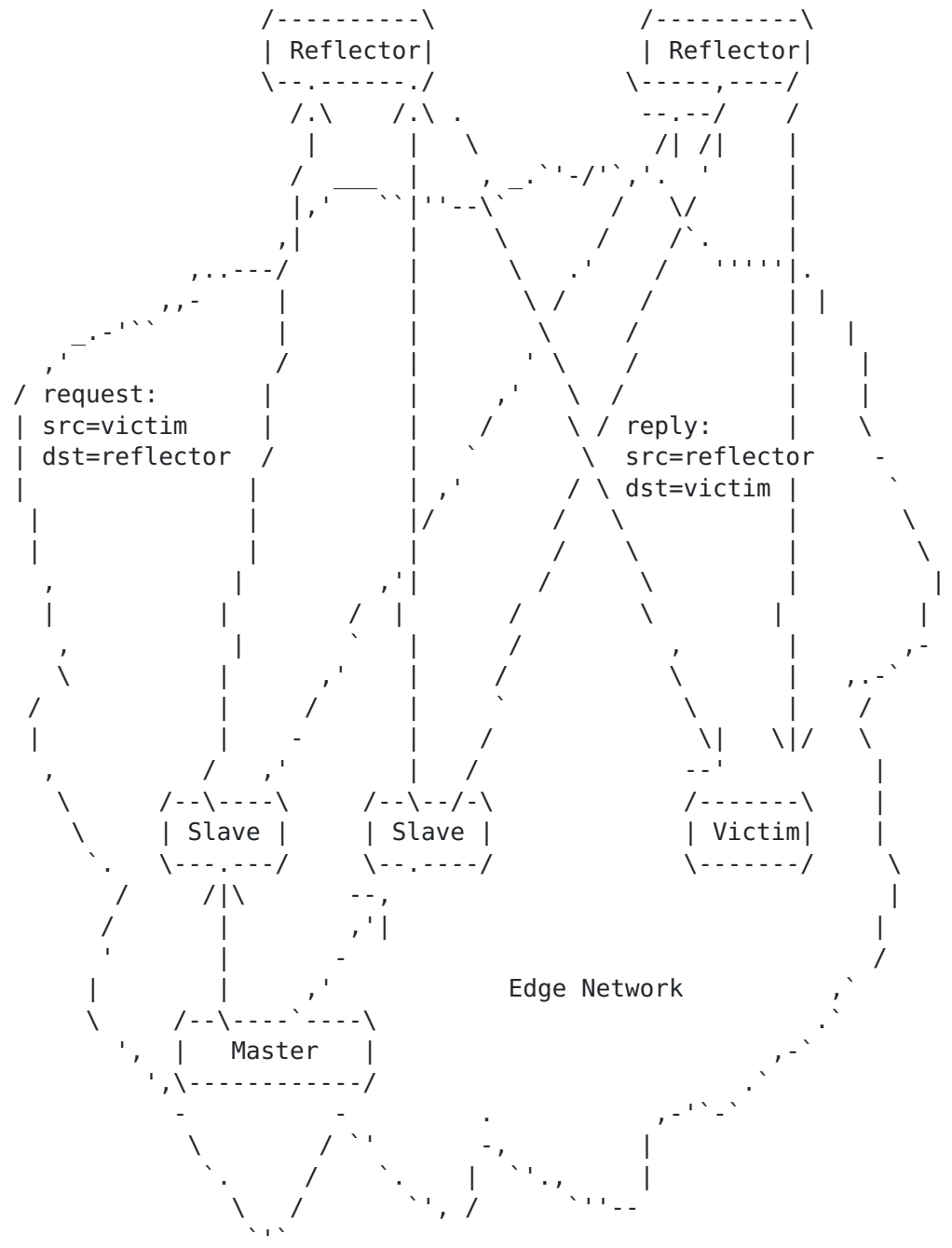

Figure 1: Reflection-type DDoS attack in the same edge network

This type of security threat may be more serious in IPv6 networks, because the IPv6 edge network may be much larger than in IPv4. In this situation, the ingress filtering may be insufficient and a fine granularity source address spoofing prevention method needs to be developed. For this purpose, this document proposes a suite of

mechanisms to prevent source address spoofing in the edge network.

The edge network this document addresses is the network behind the first 3-layer device (router, 3-layer switch, etc.). The edge network uses the border 3-layer device to communicate with other networks. This document aims to prevent source address spoofing when the host in the edge network sends packets to somewhere outside this edge network. In addition, replay attacks are handled. If only forged IP source addresses can be identified, a replay packet can still be sent to somewhere outside the edge network since its source address is valid. Therefore, identifying packets with forged source addresses anti-replay attack defense are two essential components of an integrated source address spoofing prevention method.

3. Related Work

Current work on security problems in edge networks is concentrated on access control, such as IEEE 802.1x [IEEE-802-1x], NAC [NAC], NAP [NAP] and TNC [TNC].

IEEE 802.1x is a standard for port-based network access control, which provides authenticated network access to 802.11 wireless networks and to wired Ethernet networks. The authentication of 802.1x simply controls the enabling of a switch port and in some cases the allocation of a VLAN. If the user with the legitimate account and password accesses the network, the corresponding switch port will be enabled for transit. If the user does not have the legitimate account and password or there is no user to access the network, the corresponding switch port will stay closed. 802.1x uses EAP (Extensible Authentication Protocol) [RFC3748] to carry out its authentication process. EAP is an authentication framework which supports multiple authentication methods. EAP typically runs directly over data link layers such as IEEE 802, without requiring IP.

NAC (Network Admission Control) is proposed by Cisco. NAC ensures that the access hosts are compliant with the security policies. Any noncompliant hosts cannot access the network, and they will be isolated until brought into compliance.

NAP (Network Access Protection) is another security access mechanism proposed by Microsoft. NAP platform provides a suite of integrity-checking methods to check the 'healthy' state of access hosts. Hosts that do not conform policies defining end-host security stance cannot access the network, and will be placed in quarantine until they are compliant.

TNC (Trusted Network Connection) is a framework of trusted network connection supported by about 60 members of TCG (Trusted Computing Group) [TCG]. TNC provides user identity authentication, host trust level validation and network accessibility control on the various trust levels of end systems.

These techniques mentioned above are all used to control hosts' access to the network. These techniques range from the simplest, 802.1x, that simply controls access through a switch port to more complex offerings such as NAC, NAP and TNC, that offer more complicated network access controls such as security policy enforcement, etc.

All of these techniques only have effect when hosts try to access the network. Once the host accesses the network successfully, the

subsequent packets sent from this host are typically unauthenticated, and source IP address spoofing becomes possible. Therefore, in the field of edge network security, it is not sufficient simply to ensure that the access process is secure. In order to defeat reflection-type DoS/DDoS attacks mentioned in the problem statement Section, it is necessary to validate every packet originating in the edge network and prevent the source IP address spoofing in the edge network.

4. The Solution

4.1. Overview

This document proposes a fine-granularity source address spoofing prevention method, which can prevent the attackers from forging source address that belongs to the same edge network to send packets to somewhere outside the IPv6 edge network. In this approach, if the attacker sends packets to somewhere outside the edge network by forging the IP address of another host or replays the victim's packets, these malicious packets will be filtered. Moreover, the method can work with other effective but coarser-granularity methods such as ingress filtering or SPM[Bre05] to form a multi-fence defense architecture for the next generation Internet.

The main idea is that: we deploy a security gateway to carry out the authentication algorithm as shown in Figure 1. The authentication algorithm includes two main mechanisms: the source address validation mechanism which verifies the source address using a session key a hash digest algorithm (such as MD5, SHA-1, etc.), and the anti-replay mechanism which combines the sequence number method with the timestamp method.

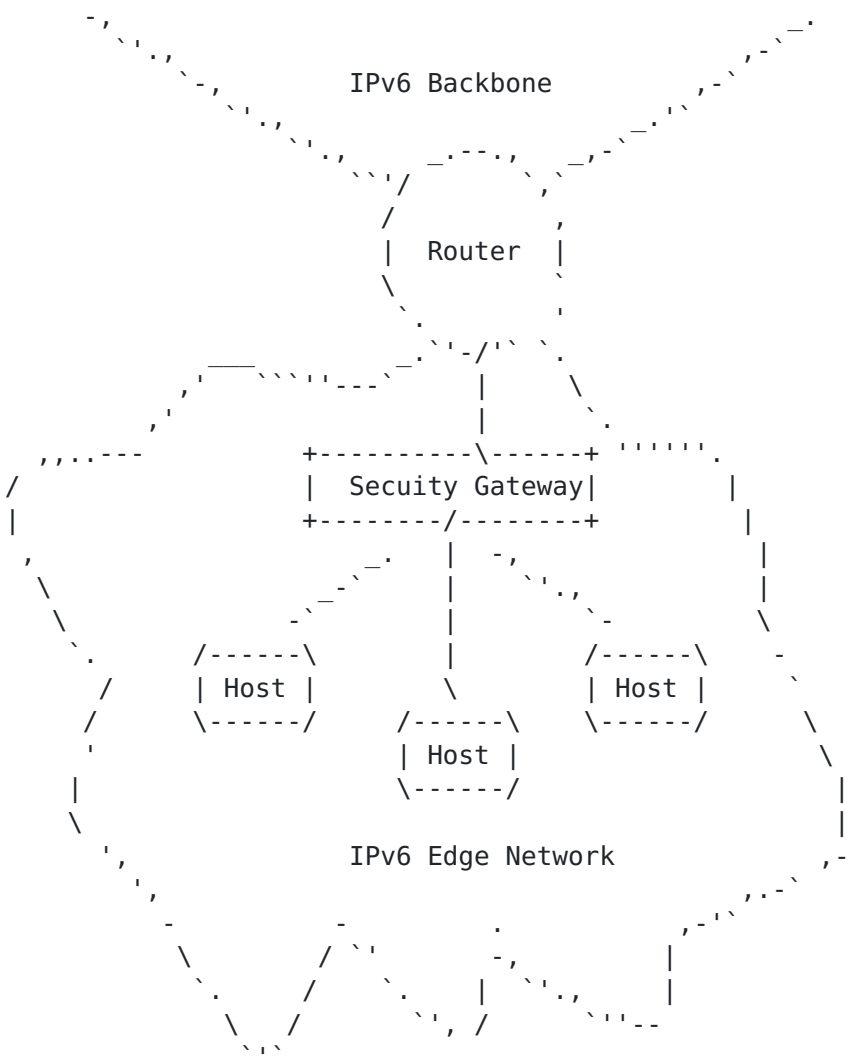

Figure 2: The deployment scenario

Initially, each host in the edge network needs to be authenticated by the security gateway. This action also binds the host's IP address and its session key which is shared by the host and the security gateway. If the access authentication succeeds, each packet sent to somewhere outside the edge network will carry a signature. Then, the packet passes through the source address validation and the anti-replay check of the security gateway. If both these verifications succeed, the packet will be forwarded to the edge 3-layer device of the edge network; otherwise, the packet will be dropped.

The approach mainly includes the following steps:

1. When a host wants to access the Internet, it should firstly carry out the access authentication. This process can use the existing access authentication mechanism such as RADIUS [RFC2865]Kerberos [RFC1510], etc.

2. The host generates a session key and sends it to the security gateway via some key exchange mechanisms such as IKE [RFC2409], IKE2 [RFC4306]. The security gateway binds the session key and the host's IP address.

3. When the host sends packets to somewhere outside the edge network, it needs to generate one signature for each packet using the hash digest algorithm. The signature is carried in a new IPv6 extension header, named 'source address validation header'.

4. The security gateway authenticates the signature carried in the packet to validate the source address.

5. The security gateway identifies the replay packets by checking whether the timestamp of the packet is expired and the sequence number of the packet is increasing.

In above steps, network access authentication and the key exchange mechanisms can use the existing approaches. Next, this document mainly discusses the source address validation algorithm and the anti-replay algorithm.

4.2. The Source address validation algorithm

This document chooses the algorithm which verifies the source address using the session key and hash digest algorithm (such as MD5, SHA-1, etc.) as the authentication method. In this algorithm, the host certifies its ownership of a certain IP address via showing the security gateway its secret session key which is shared with the security gateway as shown inFigure 3.

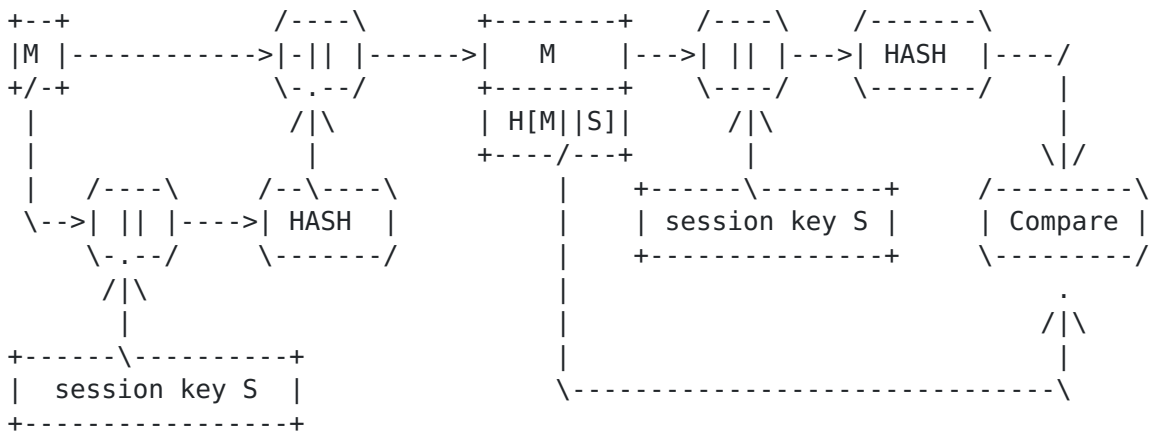

Figure 3: The authentication process of source address validation

The authentication process shown in Figure 3 is as follows:

1. Host A sends a packet M to the security gateway B. M carries a signature H[M||S] which is computed by the hash digest function using the session key S and the certain part of the packet M (source address, timestamp, sequence number, etc.) as the input.

2. When the security gateway B receives the packet M, B can re-compute the hash value HB according to the packet M, since B also knows the session key S. If HB is equal to the signature H [M||S] carried in the packet M, the security gateway B can confirm that the packet has a valid source address; Otherwise, B can conclude that the packet's source address is forged and then drops it.

4.3. The Anti-replay Algorithm

Our anti-replay algorithm combines the timestamp method and the sequence number method. Both the timestamp method and the sequence number method are prevailing anti-replay algorithms. The timestamp method works as follows: when the host A sends a packet M to the security gateway, the packet M is marked with a timestamp Ta, which represents the sending time of the packet M. Once the security gateway receives the packet M, it reads its local time Tb. If |Tb-Ta|>T, where T is the admission time window, the security gateway can conclude that the packet M is a replay of a previoualy-sent packet and drops it. However, it is hard to synchronize the clocks of the host and the gateway exactly. Moreover, network latency is uncertain. Therefore, the admission time window T is always larger

than the real transmission time of the packet. This feature could make the timestamp method insecure for anti-replay. When |Tb-Ta| is less than T, the packet should be a non-replay one. But afterwards, if the replay packet is received in the margin time (T-|Tb-Ta|), the security gateway will incorrectly regard it as a normal packet.

The main idea behind the sequence number method is: when the host A sends packets to the security gateway, each packet carries an incremental sequence number. If the latest packet's sequence number is greater than the previous one, the packet is normal; otherwise, the packet is a replay. This method requires the packets coming in order. If the packets come out of order, one can employ a sequence number window to deal with it, as with IPSec. Out-of-sequence arrival of packets is mainly due to the load balancing policy or queuing in routers or other 3-layer devices. This document deals with the situation behind the first 3-layer device, thus we can assume that the packets sending to security gateway from hosts in the edge network are always in order. (Or at the very least, out-of-sequence arrival is so rare that the rate of packet discard due to sequence error is low compared to other causes of packet loss) However, even though the packets come in order, this method may not identify some replay packets when the sequence number is used in a cycle way. For example, assuming the length of the sequence number is 16 bits, once the sequence number reaches the maximum 65535, it will return to 0 and increase as the previous cycle. In this case, if the attacker keeps a packet of the nth cycle and replays it in the (n+1) th cycle, the security gateway will not identify the replay packet.

In order to overcome the drawbacks of the timestamp and sequence method, we combine these two methods to prevent the replay attack. The timestamp method can use the sequence number mechanism to identify the replay packets in the admission time window T, and the sequence method can avoid the confusion between the normal and the replay packet by limiting the period of the sequence number cycle within the admission time window T.

5. IPv6 Source Address Validation Header

This approach utilizes a new IPv6 extension header to carry the signature, the sequence number and other useful information. We call this new extension header 'source address validation header'.

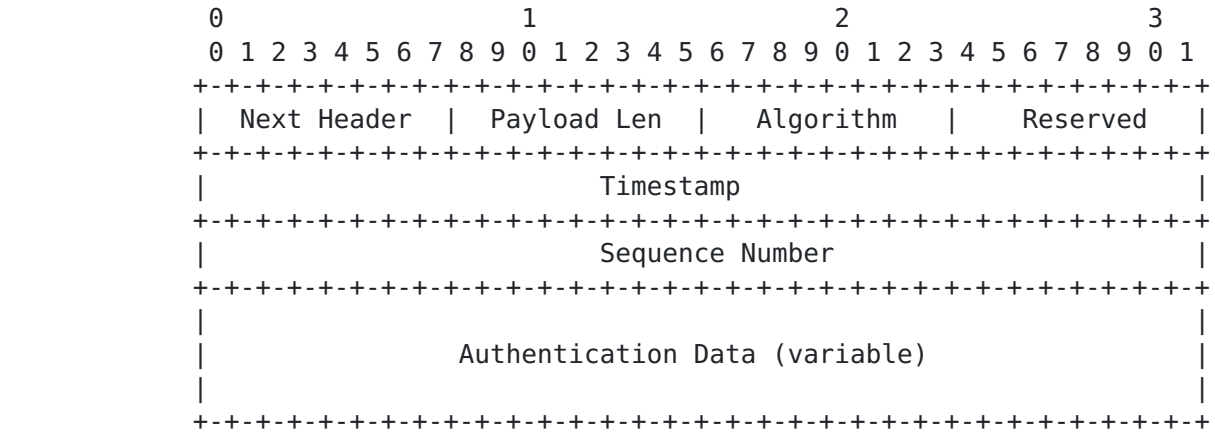

Figure 4: The format of the IPv6 source address validation header

The format of the extension header is shown in Figure 4:

- Next Header: 8-bit. Indicate either the type of the next extension header or the protocol type of the payload (TCP/UDP).
- Payload Len: 8-bit. Length of the source address validation header in 8-octet units, not including the first 8 octets.
- Algorithm: 8-bit. Point out the hash digest algorithm. For example, MD5 is set to 1.
- Reserved: 8-bit. Reserved for future use. Zero on transmit.
- Timestamp: 32-bit. It is used for anti-replay as described above.
- Sequence Number: 32-bit. It is used for anti-replay as described above.
- Authentication Data: 128 bits if using MD5 as the hash digest algorithm. The authentication data is computed by the hash digest

algorithm. The input of the hash digest algorithm includes: IPv6 source address, timestamp, sequence number and session key.

The usage of IPv6 source address validation header is as follows:

1. Each packet sent to somewhere outside the edge network needs to carry a source address validation header. The header is inserted after the IPv6 header and before all the other extension headers.

2. The security gateway conducts the following verifications when receiving the packet:

-Drop the packet directly if there is no source address validation header in the packet.

-Check the authentication data in the source address validation header as described above. Drop the packet if the check is failed.

-Confirm that the timestamp is not expired and the sequence number is greater than the previous. If that is not true, drop the packet.

3. If the whole process in the step 2 goes well, the security gateway needs to remove the source address validation header from the packet. Considering the partial deployment, if we don't remove the source address validation header, the hosts in other edge network may drop the packet due to misunderstanding of the new extension header. This step is unnecessary if our approach has been globally deployed.

6. Performance and Effectiveness Evaluation

The performance is a primary requirement of any source address validation mechanism. In our approach, this requirement reduces to the question of whether the performance of the hash digest algorithm is sufficient. Experimental result show that the performance of MD5 is about 1.63 Gbps (204.346MB/s * 8),which is evaluated in the platform of Intel P4 2.0G CPU and 512M memory. This performance can meet the requirements of most edge networks. We should note that this result is derived from the MD5 algorithm being implemented in the software. If we implement the MD5 algorithm using hardware, we will get higher performance. We conclude therefore, that the source address validation algorithm in our approach is completely feasible.

In order to testify the effectiveness of our anti-replay algorithm which integrates timestamp and sequence number methods, we conducted several experiments to compare the effects of the timestamp, sequence number and our anti-replay algorithm. The experiments show that the effectiveness of timestamp method seriously relies on the size of admission time window T. As the margin of the admission time window T is increased, the effectiveness of the timestamp method decreases. In the experiment to test effectiveness of the sequence number method, the accuracy of replay packets identification in the (n+1)th sequence number increasing cycle is linear with the packet number recorded in the nth sequence number increasing cycle. The experimental result shows that almost all packets recorded can be replayed successfully under conditions of persistent replay. We repeated the above two simulation experiments to test the effectiveness of our anti-replay algorithm which integrates the timestamp and the sequence number methods. Even though the admission time window T has a large margin, with the help of sequence number method, all replay packets are identified successfully. In the (n+1)th sequence number increasing cycle, when attacker replays the packets recorded in the nth sequence number increasing cycle, the security gateway can identify nearly 100% replay packets since the intervals between replay packets and normal packets exceed the admission time window T.

Our experiments show that the source address validation algorithm this document proposed prevent the attackers from forging source address that belongs to the same edge network to send packets to somewhere outside the IPv6 edge network effectively. The anti-replay algorithm this document proposed can overcome the shortcomings of both timestamp and sequence number methods, and form a more effective, fine-grain anti-replay mechanism.

7. An Application Case

Multihoming to upstream ISPs is increasingly common in the Internet. Multihoming refers to the phenomenon that one edge network accesses to the Internet through multiple upstream ISPs. In the multihoming environment, a host in the edge network may have several ISP-assigned IP addresses and the edge network may have several outbound links to different ISPs, which makes the source address spoofing prevention complicated. This document proposes a solution for the multihoming scenario as shown in Figure 5.

In our solution, each edge router of the edge network is equipped with a security gateway. Each security gateway is responsible for handling the addresses assigned from the ISP to which that security gateway is attached. For example, in Figure 5, the security gateway A just handles the addresses assigned from ISP A. Other than this, the security gateway performs the same functions as the single-homed situation described above.

Another problem is that the security gateway may be confused in the multihoming situation since outbound packets are routed by the destination address. For example, in Figure 5, when host A sends packets using the addresses based on prefix A (PrefixA:PrefSite: hostA) as the source addresses, the packets could be sent to the edge router B connected to ISP B, and the security gateway B will be confused and take the wrong action. In order to solve this problem, packets originating in the edge network are passed through a source routing domain (shown in Figure 5) to which all edge routers are connected. These routers would choose a route based on the destination and source addresses of a packet, selecting the appropriate edge router for the given source address. In this way, our source address spoofing prevention method can work well in the multihoming scenario.

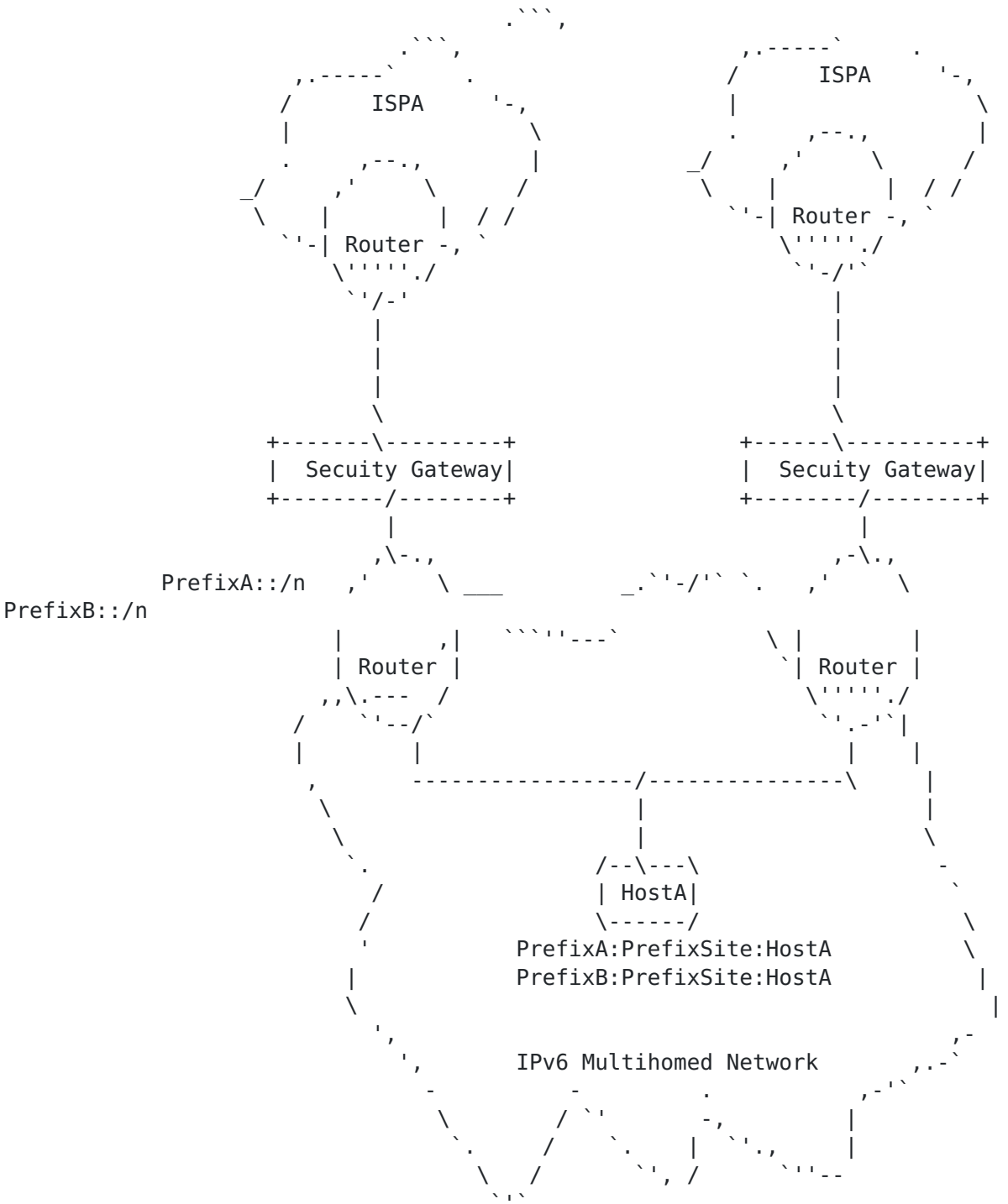

Figure 5: A application case for the multihomed site

8. Acknowledgements

The authors would like to acknowledge the contributors who provided helpful inputs on earlier versions of this document and Mark Williams who provided editorial support.

9. References

[Bre05] Bremler-Barr, A. and H. Levy, "Spoofing Prevention Method", Infocom 2005, 2005.

[IEEE-802-1x] "IEEE Standard for Local and metropolitan area networks: Port-Based Network Access Control", 2001.

[NAC] "NAC", http://www.cisco.com/en/US/netsol/ns617/networking_solutions_sub_solution_home.html .

[NAP] "NAP", http://www.microsoft.com/technet/network/nap/default.mspx .

[RFC1510] Kohl, J. and B. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[TCG] "TCG", https://www.trustedcomputinggroup.org/home .

[TNC] "TCG Trusted Network Connect TNC Architecture for Interoperability", 2005.

Authors' Addresses

Jun Bi
CERNET
Network Center, Tsinghua University
Beijing 100084
China

Email: junbi@cernet.edu.cn

Jianping Wu
CERNET
Network Center, Tsinghua University
Beijing 100084
China

Email: jianping@cernet.edu.cn

Lizhong Xie
CERNET
Network Center, Tsinghua University
Beijing 100084
China

Email: xlz@netarchlab.tsinghua.edu.cn

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).