Shim6: Reference Implementation and Optimization

Jun Bi, Ping Hu, and Lizhong Xie

Network Research Center, Tsinghua University,
Beijing, 100084, China
junbi@tsinghua.edu.cn

Abstract. Shim6 is an important multihoming solution. This paper studies shim6 from several perspectives, including shim6 protocol implementation, shim6 mechanism optimization and security enhancement. In order to provide a shim6 research platform, we implement shim6 protocol on the Linux 2.6 platform as one of the first reference implementations. Based on this research platform, we refine the shim6 address switching mechanism, which reduces shim6 address switching time greatly. In addition, we propose an enhanced shim6 security mechanism to defeat reflection-type DoS/DDoS attacks launched from the multihomed site, by preventing source address spoofing in the multihomed site.

Keywords: Multihoming, Shim6, IPv6.

1 Introduction

Multihoming to upstream ISPs is a requirement today for most IPv4 networks in the Internet, especially for the enterprise networks. With the deployment of IPv6 network, more and more devices can access to the Internet, and more and more sites will have the requirement of multihoming. IPv6 multihoming will be a very common phenomenon undoubtedly.

Currently, the shim6 mechanism [1] is an important multihoming approach. This paper studies shim6 from several perspectives, including shim6 protocol implementation, shim6 mechanism optimization and IPv4/IPv6 transition utilizing shim6. In order to provide a shim6 research platform, we implement shim6 protocol on the Linux 2.6 platform, which is one of the first reference implementations in the world. Based on this research platform, we refine the shim6 address switching mechanism, which reduces shim6 address switching time greatly. In addition, we propose an enhanced shim6 security mechanism to defeat reflection-type DoS/DDoS attacks launched from the multihomed site, by preventing source address spoofing in the multihomed site. In order to explore the utilization of shim6 in other research fields, we propose a mechanism called MI46 to optimize IPv4/IPv6 inter-operation using simplified shim6.

The rest of this paper is organized as follows: Section 2 describes our Linux-based shim6 implementation; Section 3 and section 4 present the enhancements on shim6 protocol, including shim6 address switching mechanism refinement and shim6 security mechanism enhancement. Section 5 concludes the paper.

A. Das et al. (Eds.): NETWORKING 2008, LNCS 4982, pp. 302–313, 2008.

2 Reference Implementation of Shim6

Multihoming refers to the phenomena that one network end node accesses to the Internet through multiple network paths mainly due to the consideration of fault resilience. For the purpose of accessing to the Internet via multiple network paths, the multihomed network end node often possesses several addresses. Once the current network path fails, the multihomed network end node can immediately switch to another address and use another network path to communicate.

In the shim6 approach, a new 'SHIM6' sub-layer is inserted into the IP stack in end hosts that wish to take advantage of multihoming. The shim6 sub-layer is located within the IP layer between the IP endpoint sub-layer and IP routing sub-layer. With the shim6, hosts have to deploy multiple provider-assigned IP address prefixes from multiple ISPs. These IP addresses are used by applications and if a session becomes inoperational, shim6 sub-layer can switch to using a different address pair. The switch is transparent to applications as the shim6 layer rewrites and restores the addresses at the sending and receiving host.

For the purpose of transport layer communication survivability, the shim6 approach separates the identity and location functions for IPv6 addresses. In shim6, the identifier is used to uniquely identify endpoints in the Internet, while the locator is used to perform the role of routing. There is a one-to-more relationship between the identifier and locator. The shim6 layer performs the mapping function between the identifier and the locator consistently at the sender and the receiver. The upper layers above the shim6 sub-layer just use the unique identifier to identify the communication peer, even though the locator of the peer has changed. Hence, when the multihomed host switches to another locator, the current transport layer communication does not break up since the identifier is not changed.

Due to the lack of the implementation, there are a lot of problems on shim6 which nobody can give some clear answers. Therefore, in order to start our research on shim6, we first implement shim6 protocol based on Linux 2.6.x platform.

The control plane of shim6 mainly refers to the four handshakes process. The function of first two handshakes are to confirm the peer whether deploys shim6, and that of other two handshakes are to exchange the address list with the peer.

The data plane of shim6 includes two main functions: rewriting the source and destination addresses of each packet, adding (the sender) /removing (the receiver) the payload extension header to/from the packet.

The control plane of shim6 is implemented in the user space while the data plane is implemented in the kernel space using netfilter mechanism [2] of Linux. That's because the four handshakes function of control plane is more complicated than the rewriting packets' addresses function of data plane. If the four handshakes function is implemented in the kernel space, it may bring bad influence to the efficiency of the kernel. However, if we want to modify the addresses of every packet, we must implement the function of rewriting packets' addresses in the kernel space. And it doesn't bring too much weight to kernel since the function of rewriting packets' addresses is very simple. But, the data plane relies on the addresses mapping table that is generated by the control plane. So, we implement modules of communication

between the kernel and user space using the Linux netlink mechanism[3], in order to transfer the addresses mapping table from the control plane to the data plane.

3 Optimization of Shim6

In our experimental platform of shim6, we explore some researches on optimization of shim6 and get some achievements. The address switch mechanism of shim6 is defined in REAP (REAchability Protocol)[4]. REAP contains two parts: failure detection mechanism which detects failures between two communicating hosts, and address-pair exploration mechanism which locates another operational address-pair if a failure occurs.

The failure detection mechanism uses two timers (Keepalive Timer, Send Timer) and Keepalive message to detect the failures between two communicating hosts. The concrete process is as follows:

1. When host A sends data packets to host B, a send timer starts at the same time. The suggested timeout value is 10s.
2. When host B receives data packets sent from host A, a Keepalive timer starts simultaneously. The suggested timeout value is 3s. If host B sends no packets to host A within the keepalive interval (less than Keepalive timeout value) after receiving data packets from host A, host B sends a Keepalive message to host A.
3. If host A receives no packets (including data packets and Keepalive packets) from host B before the send timer is timeout, it can be determined that there is a failure of current address pair. And then the addresses exploration mechanism sets up to locate another operational addresses pair.

The address-pair exploration mechanism uses the Probe message to examine the reachability of the address-pair. Specifically, if host A wants to choose another address-pair with host B, host A needs to send a series of Probe messages to host B until receiving the Probe message sent from B. Upon receipt of the Probe message from host B, host A can conclude that the new address-pair is reachable from A to B. After receiving the first Probe message from host A, host B applies the same algorithm to confirm that the address-pair is reachable from B to A. If the test fails, another address-pair is selected to do the next address exploration. REAP defines some suggested default values associated with the exploration process. The Initial Probe Timeout which specifies the interval between initial attempts to send probes is 0.5s. Then the exploration process uses the exponential backoff procedure to increase the time between every probe if there is no response. So, each increase doubles the time. If the exploration process reaches Max Probe Timeout (60 seconds), it will continue sending at this rate until a suitable response is received.

From the above description, we can see the time of shim6 address switching can be expressed as the following formula:

Shim6 address switching time = failure detection time + next address-pair exploration time

The failure detection time is the value of Send timer timeout (10s). The next address-pair exploration time may grow exponentially. If the multihomed hosts have numerous addresses and the address failure rate is high, the shim6 address switching time will be very time consuming. For example, if the multihomed host A and B have Na and Nb addresses respectively, then under the worst circumstance each host would be required to detect about Na*Nb times to examine the next operational address-pair.

In the original shim6 address switching mechanism, the shim6 next address-pair exploration process starts until the failure detection has finished, that is, after 10s, the hosts start to explore the reachability of the next address-pair. In this paper, we explore from the SCTP[5] protocol to refine the shim6 address switching mechanism, whose main idea is: during the failure detection, the hosts maintain the reachability of next address-pair simultaneously. In this way, once the failure occurs, the hosts can immediately switch to next operational address-pair. The refined shim6 address switch mechanism can shorten the shim6 address switching time to exactly 10s (that is the failure detection time).

In the refined shim6 address switch mechanism, we use the Heartbeat message to maintain the reachability of next address-pair. The details are as follows:

While host A and B communicate with each other using the current address-pair, meanwhile host A and B send Heartbeat request to each other periodically.

Upon receipt of the Heartbeat request, the host sends back the Heartbeat acknowledgement.

The host maintains a counter to record the number of Heartbeat requests that have been sent but not received Heartbeat acknowledgements. When the counter value reaches a certain threshold, this address-pair is marked un-reachable. Then another address-pair is selected to examine the reachability using the same method.

We design several experiments to verify the effectiveness of the refined shim6 address switching method regarding the address number of each multihomed host and address failure possibility as the variables. The experimental result shows that when the multihomed hosts have numerous addresses, and these addresses are at a relatively high probability of failure, the original shim6 address switching time significantly increases. But the refined shim6 address switching time remains 10s constantly. Thus, in this situation, the refined shim6 address switching mechanism can reduce the address switching time effectively.

4 Source Address Spoofing Prevention for Multihomed Site

4.1 Problem Statement

Today's Internet is suffering from reflection-type DoS/DDoS attacks. 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 hard to trace. The approaches based filtering, such as ingress filtering [6][7], can validate the source address in real-time. But the current approaches based on filtering just validate the IP source address according to the network prefix. Thus, these approaches can not

prevent the IP source address spoofing in the edge network granularity, and leave some room to launch reflection-type DoS/DDoS attacks. For instance, with the ingress filtering, the attacker can still forge the IP source address that belongs to the same multihomed site and attack the victim using the reflection-type DoS/DDoS. This type attack scenario is shown in Figure 1.

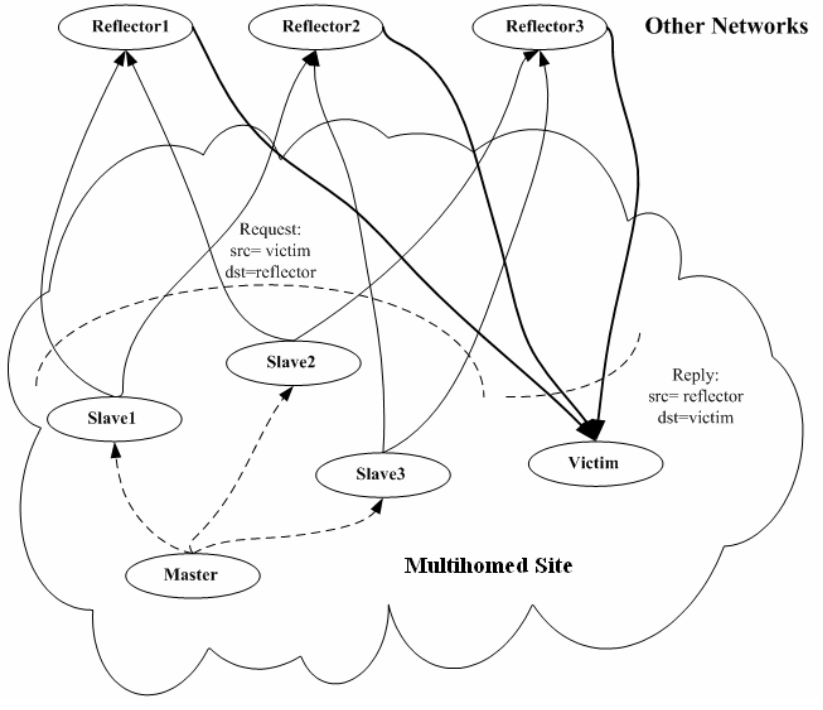


Fig. 1. Reflection-type DDoS attack in the same multihomed site

This type security threats may be more serious with the IPv6 network deployment, because the IPv6 multihomed site may be much larger than the one of IPv4. In this situation, the ingress filtering is not sufficient. This paper aims to prevent the source address spoofing when the host in the multihomed site sends packets to somewhere outside this multihomed site. In addition, the replay attack must be handled. Even if the forged IP source address packets can be identified, the replay packet can still be sent to somewhere outside the multihomed site since its source address is valid.

4.2 The Algorithms

4.2.1 Source Address Validation Algorithm

In our solution, each edge router of the edge network is equipped with a security gateway to carry out the authentication algorithm. Each security gateway is responsible for handling the addresses assigned from the ISP to which the security gateway is

attached. For example, in Figure 2, the security gateway A just handles the addresses assigned from ISP A. When host A sends packets using the addresses based on prefix A (PrefixA:PrefSite:hostA) as the source addresses, the packets may be sent to the edge router B connected to ISP B, and the security gateway B will be confused and take wrong action. In order to solve this problem, packets originating in the multihomed site are passed through a source routing domain (see Figure 2), 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.

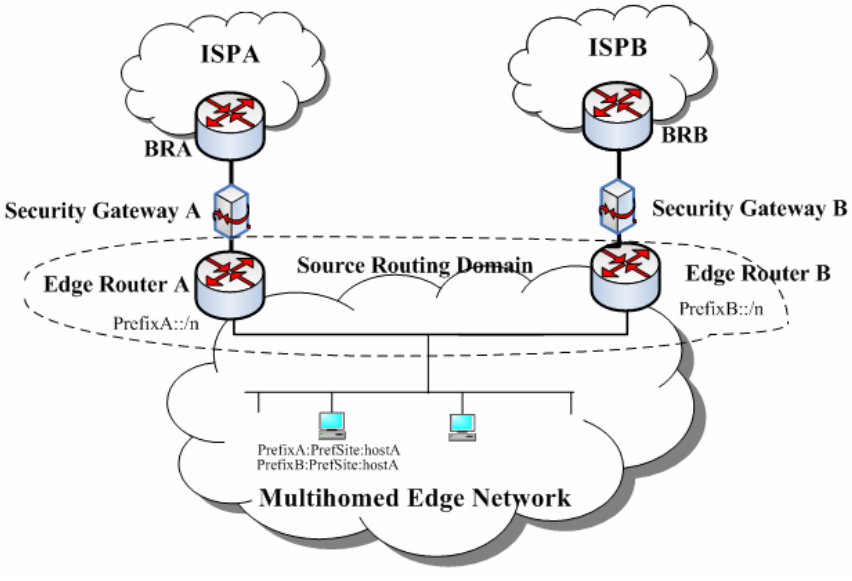


Fig. 2. The source address spoofing prevention solution for multihoming scenario

Initially, 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 802.1x[8], NAC[9], NAP[10] and TNC[11], etc.

If the access authentication succeeds, the host generates a session key and sends it to the security gateway via some key exchange mechanisms such as IKE and IKE2. The session key is a random number that is at least 12-byte-long. The security gateway binds the session key and the host's IP address.

When the host sends packets to somewhere outside the multihomed site, it needs to certify its ownership of a certain IP address via showing the security gateway its secret session key which is shared with the security gateway by generating one signature for each packet using the hash digest algorithm (MD5, SHA-1, etc). The approach mainly includes the following steps. 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. 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.

4.2.2 The Anti-replay Algorithm

After validating the signature, the security gateway also identifies the replay packets by checking whether the timestamp of the packet is expired and the sequence number of the packet is increasing. 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 T_a, which represents the sending time of the packet M. Once the security gateway receives the packet M, it reads its local time T_b. If $|T_b-T_a|>\triangle T$, where $\triangle T$ is the admission time window, the security gateway can conclude that the packet M is a replay one then drops it. However, it's hard to synchronize the clocks of the host and the gateway exactly. Moreover, the transmitting time of the packet in the network is also uncertain. Therefore, the admission time window $\triangle T$ is always larger than the real transmitting time of the packet. This feature makes the timestamp method unfaithful for anti-replay. When $|T_b-T_a|<\triangle T$, the packet should be a non-replay one. But afterwards, if the replay packet is received in the margin time ($\triangle T-|T_b-T_a|$), the security gateway will regard it as a normal packet incorrectly.

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 one. This method requires the packets to come in order. If the packets come out of order, one can employ a sequence number window to deal with it like IPSec. That the packets come out of order is mainly due to the load balance policy or queuing in the router or other 3-layer devices. This paper 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. 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 n^{th} cycle and replays it in the $(n+1)^{th}$ cycle, the security gateway can't 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 $\triangle T$; 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 $\triangle T$.

4.2.3 IPv6 Source Address Validation Header

In our approach, we design 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". We develop a new IPv6 extension header rather than use the IPv6 AH extension header to complete the authentication between host and security gateway, because the AH extension header just can be carried once in the IPv6 packet according to the IPv6 specification. If we use it here, the host can not use the AH extension header with its communication correspondent. That's not the result we want.

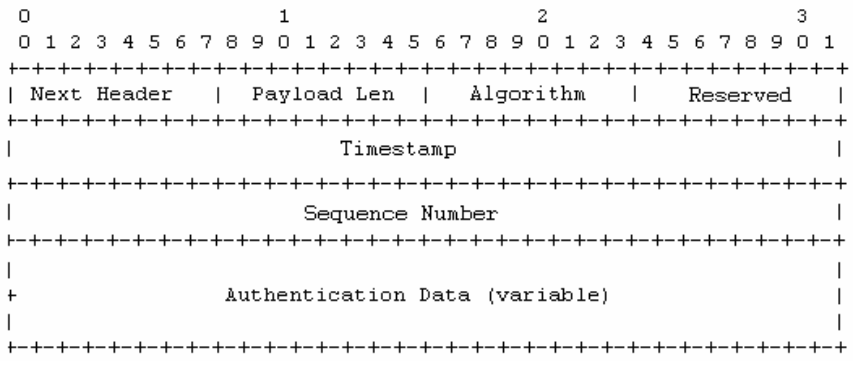


Fig. 3. The format of the source address validation header

The format of the extension header is shown in Fig. 3:

- 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 to anti-replay as described above.
- Sequence Number: 32-bit. It is used to 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 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 multihomed site may drop the packet due to misunderstanding of the new extension header. This step is unnecessary if our approach has been globally deployed.

4.3 Experiment Simulation

4 3.1 The Performance Evaluation of Source Address Validation Algorithm

Because every packet needs to be authenticated when they are sent to somewhere outside the edge network, the primary design requirement of source address validation

algorithm is the high performance. In our approach, this requirement means whether the performance of the hash digest algorithm is high enough. In addition, the security gateway needs to record the (IPv6 address, session key) pair for every host in the edge network, and each authentication procedure needs to look up the session key according to the IPv6 address. We use the hash table to organize the (IPv6 address, session key) pair, and its average performance for looking up is $O(1)$. We conduct some experiments to validate the source address validation algorithm. Table 1 shows our experimental results, which are evaluated in the platform of Intel P4 2.0G CPU and 512M memory.

Table 1. The performance comparison of two main hash digest algorithms

HASH Digest algorithm	The capacity per second (MB/S)
MD5	205
SHA-1	66

The results show that the performance of MD5 is about 1.64 Gbps This performance can meet the requirements in the most cases. We should note that this result is derived from the MD5 algorithm implemented in the software. If we implement the MD5 algorithm using hardware, we will get a higher performance. Therefore, the source address validation algorithm in our approach is completely feasible.

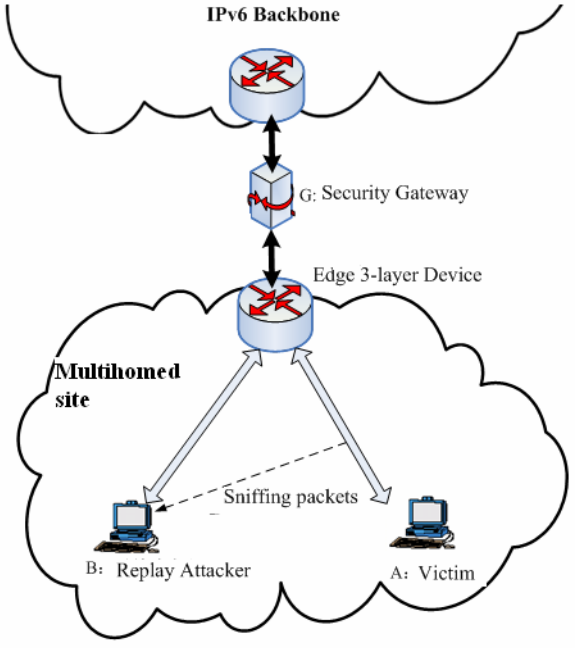


Fig. 4. The topology of anti-replay algorithm simulation experiments

4.3.2 The Effectiveness Evaluation of Anti-replay Algorithm

In order to validate the effectiveness of our anti-replay algorithm which integrates the timestamp and the sequence number methods, we conduct several experiments to compare the effects of the timestamp, sequence number and our anti-replay algorithm. The experimental topology is shown as Fig 4. In Fig 4, host A is the victim, host B sniffs the packets sent from A and replays them.

In the simulation experiment to validate effectiveness of the timestamp method, host A and the security gateway G synchronize the clocks exactly. The packet transmitting time from A to G is about 8~10ms. The host B takes 2ms±50% to sniff a packet and replay it. We set the admission time window $\triangle T$ in the security gateway is 10ms, 11ms and 12ms respectively to see the effectiveness of the timestamp method. Fig 5 shows the result when the attacker B replays 10,000 packets.

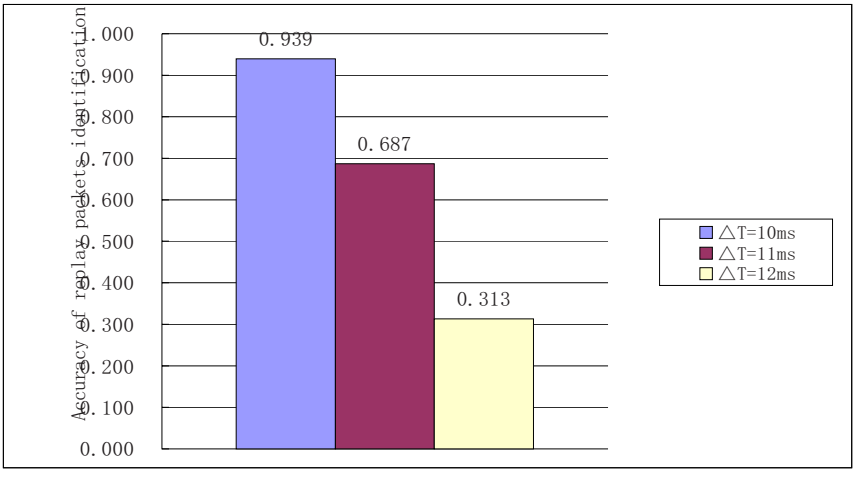


Fig.5. The simulation result of the timestamp method

From Fig 5 we can see, the effectiveness of timestamp method seriously relies on the size of admission time window $\triangle T$. When $\triangle T$=11ms, that is, $\triangle T$ has a growth of 10% from the upper bound of packet transmitting time, the gateway can identify 68.7% of replay packets. When $\triangle T$=12ms, that is, $\triangle T$ has a growth of 20% from the upper bound of packet transmitting time, the gateway can identify only 31.3% of replay packets. In the simulation experiment, we make host A and the security gateway G synchronize the clocks exactly. But, in the real network environment, it is hard to synchronize clocks exactly. What's worse, the packet transmitting time in real network may be more fluctuant than that we set in the experiment. So, the admission time window $\triangle T$ may have larger margin and the effectiveness may be much worse in the real network.

In the simulation experiment to validate effectiveness of the sequence number method, the sequence number is 16-bit long. After recording the packets sent from A in the n^{th} sequence number increasing cycle, attacker B replays these packets persistently in the $(n+1)^{th}$ sequence number increasing cycle. Fig 6 shows the accuracy of replay

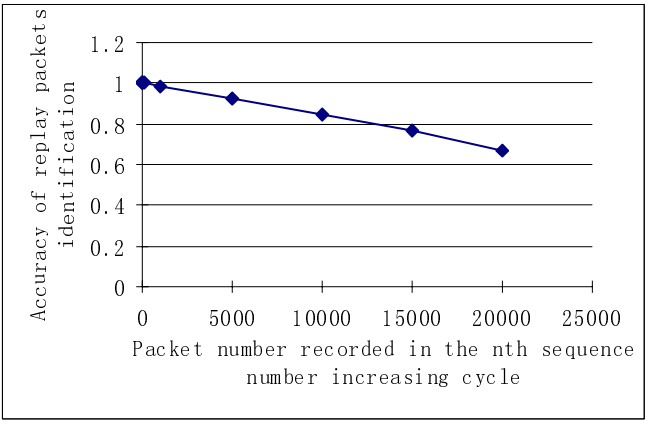
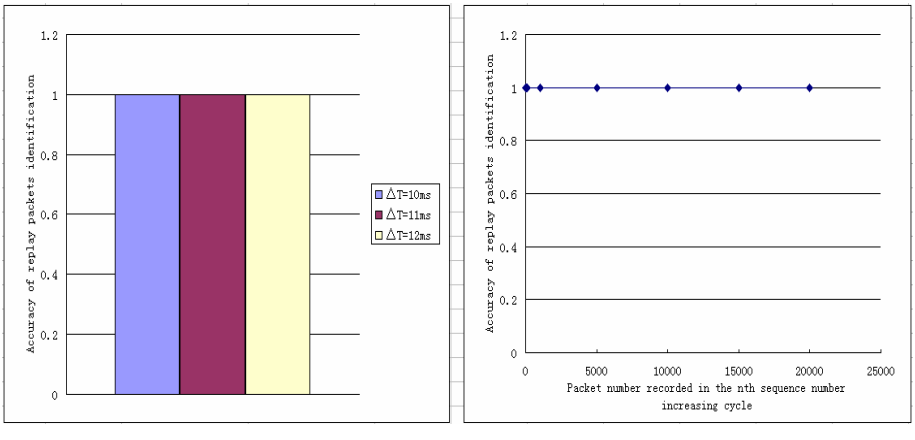


Fig.6. The simulation result of the sequence number method

packets identification in the $(n+1)^{th}$ sequence number increasing cycle when attacker B respectively records 10, 20, 50, 100, 1000, 5000, 10000, 15000 and 20000 packets sent from A in the n^{th} sequence number increasing cycle.

From Fig 6 we can see, under condition of persistent replay, the accuracy of replay packets identification in the $(n+1)^{th}$ sequence number increasing cycle is linear with the packet number recorded in the n^{th} sequence number increasing cycle. The experiment result shows that almost all packets recorded can be replayed successfully under condition of persistent replay.

We redo the above two simulation experiments to validate the effectiveness of our anti-replay algorithm which integrates the timestamp and the sequence number methods. The result is shown as Fig 7.



a. Comparison with timestamp b. Comparison with sequence number

Fig. 7. The results of our anti-replay algorithm integrating timestamp and sequence number

In Fig 7a, even though the admission time window $\triangle T$ has a large margin, with the help of sequence number method, all replay packets are identified successfully. In Fig 7b, in the $(n+1)^{th}$ sequence number increasing cycle, when attacker B replays the packets recorded in the n^{th} 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 $\triangle T$.

From the above experiments we can conclude that the anti-replay algorithm this paper proposed can overcome the shortcomings of both timestamp and sequence number methods, and form a more effective, fine-grain anti-replay mechanism.

5 Summary

This paper studies shim6 from several perspectives, including shim6 protocol implementation, shim6 mechanism optimization and security enhancement. In order to provide a shim6 research platform, we implement shim6 protocol on the Linux 2.6 platform, which is one of the first reference implementations in the world. Based on this research platform, we introduce HEARTBEAT message to refine the shim6 address switching mechanism, which reduces shim6 address switching time greatly. In addition, we propose an enhanced shim6 security mechanism to defeat reflection-type DoS/DDoS attacks launched from the multihomed site, by preventing source address spoofing in the multihomed site.

References

1. Nordmark, E.: Level 3 multihoming shim protocol, draft-ietf-shim6-proto-08.txt (2007)
2. http://www.netfilter.org
3. Salim, J., Khosravi, H., Kleen, A., Kuznetsov, A.: Linux Netlink as an IP Services Protocol. RFC 3549 (July 2003)
4. Arkko, J., Beijnum, I.: Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming. draft-ietf-shim6-failure-detection-07.txt (2006)
5. Stewart, R., et al.: Stream Control Transmission Protocol. IETF RFC 2960 (2000)
6. Ferguson, P., Senie, D.: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC2827 (2000)
7. Baker, F. and Savola, P.: Ingress Filtering for Multihomed Networks. RFC3704 (2004)
8. American National Standards Institute: IEEE-SA Standards Board: IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control (2001)
9. Cisco Systems: Network Admission Control
10. Microsoft: Network Access Protection
11. TNC: TCG Trusted Network Connect TNC Architecture for Interoperability (2005)