Preventing Bad Prefixes Attack in IPv6 Stateless Address Auto-configuration Protocol

Rohan D. Doshi
Department of Computer Science and Engg.
National Institute of Technology Karnataka
Surathkal, India
doshirohan1@gmail.com

B. R. Chandavarkar
Department of Computer Science and Engg.
National Institute of Technology Karnataka
Surathkal, India
brcnitk@gmail.com

Abstract— **Stateless address auto-configuration protocol in IPv6 is found to be vulnerable to bad prefixes attack wherein malicious node in the network can misconfigure hosts in the network so as to cause disruption in the communication between nodes in the same subnet or in different subnets. This paper proposes solution to prevent this type of attack. Initially this paper demonstrates existence of such vulnerability in Linux operating system. At the end of this paper implementation of prevention technique and results are presented.**

Keywords- IPv6, Security Issues, Bad Prefixes Attack, Prevention Techniques

I. INTRODUCTION

The growth of internet has created need for more addresses than that are possible with IPv4. To deal with IPv4 address exhaustion, IPv6 was published initially in the form of RFC 1883 in Dec. 1995[1] and then revised in the form of RFC 2460 in Dec. 1998[2]. Along with the expansion of address space, few changes have also been introduced in the protocol specification to improve its performance.

IPv6 allows network layer addresses to be configured in two different ways, stateful address auto-configuration and stateless address auto-configuration. Further each interface connected to network can have multiple IP addresses simultaneously. Stateful address configuration is achieved using Dynamic Host Configuration Protocol for IPv6 [3] (DHCPv6), where central server is responsible for allocation of IP addresses to each host. Stateless address auto-configuration is defined in RFC 4862 [4] "IPv6 Stateless Address Auto-configuration Protocol" and allows hosts to configure their addresses on their own, without need of any central entity.

In stateless address auto-configuration (SLAAC) host generates IP address for himself by generating "interface identifier", which is unique on the link. This interface identifier is then combined with network prefix advertised by the routers, to form globally addressable unique IPv6 address. Many security issues have been identified in this mechanism. This paper addresses flaw in stateless address auto-configuration allowing bad prefixes attack to take place.

II. BACKGROUND

A. Stateless Address Auto-configuration

IPv6 Stateless Address Auto-configuration protocol described in RFC 4862 can be summarized as follows.

1) Host generates "host-identifier" to be used as suffix in the IPv6 address. Generation of this suffix is implementation specific. Few implementations (EUI-64) use MAC address of the interface to generate host ID.

2) Link local prefix is attached to this identifier to generate link local address. This address, if unique, can be used to communicate with other hosts in the subnet.

3) Duplicate Address Detection (DAD) is done in order to confirm uniqueness of the address generated. Detection of duplicate address on the link forbids host from acquiring this IP address, and host needs to generate another host-identifier in order to continue.

4) Subnet prefix is then applied to host-identifier in order to form globally routable address. This address, if unique, is then acquired and applied to the interface.

5) Subnet prefix can be discovered through periodic Router Advertisements (RA) sent by routers in the network. Hosts can also request for RA by sending all-router broadcast Router Solicitation packet.

6) Receipt of multiple RA advertising subnet prefix allows hosts to assign multiple IPv6 addresses to the interface.

Different functionalities used in the above process are Neighbor Solicitation, Neighbor Advertisement, Duplicate Address Detection, Router Solicitation, and Router Advertisement. Any of these functionalities can be mimicked by malicious node in the network without being detected.

B. Literature Survey

Stateless address auto-configuration in IPv6 raises issues related to security of network. Any node connected to network is allowed to acquire link-local address without any approval and thus to send messages over network. Also with the help of router solicitation and neighbor discovery it can construct globally routable address and use it to perform different kinds of attacks.

978-1-4673-2605-6/12/$31.00 © 2012 IEEE

1) Malicious Router: Node can act as malicious router by sending router advertisements and by responding to router solicitations. Unsuspecting node can accept malicious node as its on-link router and thus sending all traffic to malicious node causing man in middle attack.

2) Attack on Legitimate Router: Malicious node can attack on legitimate router by issuing router advertisements with zero router life-time and thus making legitimate router unavailable.

3) Bad Prefixes: Malicious node can advertise bad prefixes on network which does not exist on the link. Hosts which configure themselves based on these advertisements won't be able to communicate with other hosts having address with this prefix, which are external to the network, as victims will consider these addresses as on-link address and will not send packets to router. Instead they will try to perform address resolution by sending neighbor solicitation messages and will consider addresses to be unreachable.

4) Failure of DAD: Malicious node can respond to all DAD (Duplicate Address Detection) requests with corresponding neighbor advertisements and thus making these addresses unavailable to the newly joining nodes.

The above attacks have been discussed in literature [6] [7] and affect working of SLAAC. In the following sections we will discuss bad prefixes attack and its prevention.

III. Attack Analysis

In normal scenario when a host wants to communicate with another system, series of actions taken by host depends on destination IP address. If host discovers that destination IP address is on the same network as that of host, it will try to communicate directly with destination system. Host will look up for destinations MAC address in neighbor table. If neighbor entry exists, it will be used to communicate with neighbor; otherwise neighbor discovery is initiated with neighbor table entry for destination set to INCOMPLETE. Once neighbor discovery is finished, depending on whether or not reply is received from destination system, corresponding neighbor table entry will be set to REACHABLE or FAILED. In earlier case corresponding neighbor table entry will then be used for communicating with the neighbor.

On the other hand if host discovers that destination IP address is not on the same network, it will communicate to destination system through default gateway of the subnet. Host will forward all packets destined to destination system to default gateway and rely on gateway to take care of how these packets will reach the ultimate destination.

In case of attack, suppose malicious node has advertised prefix L11. All hosts in the subnet will then configure to use address with prefix L11 and assume that all IPv6 addresses starting with this prefix lie in the same subnet. When any host in the subnet tries to communicate with destination from another subnet with subnet prefix L11 instead of sending packets to default gateway, host will try to reach the system directly within the subnet. It will initiate neighbor discovery, with corresponding neighbor table entry set to INCOMPLETE. After certain time it will discover that it is not able to reach particular destination and will mark corresponding neighbor table entry as FAILED.

Thus in attack scenario host will incorrectly assume that destination system is unreachable and therefore won't be able to communicate with the system.

IV. Preventing Bad prefixes attack

Attack analysis in the previous section shows that reason behind success of bad prefixes attack is hosts' inability to discriminate between valid router advertisements and those sent by malicious node. RA sent by malicious node makes hosts in the subnet to visualize network differently than it actually is, resulting in successful attack.

One of the mechanisms to prevent these types of attacks is to initialize hosts with information required to identify routers/hosts allowed to send RA. This way hosts will never configure themselves with prefixes advertised by malicious node thus preventing the attack. However main goal behind stateless address auto-configuration is that any system connected to IPv6 enabled network should be able to configure itself, without need of any centralized server or any manual configuration of hosts. This essentially is to lessen burden on system administrators in order to manage large subnets in IPv6 network. Thus this solution cannot be applied to stateless address auto-configuration protocol.

The solution proposed in this paper is compatible with stateless address auto-configuration protocol and can address subnet unreachability due to mis-configured IP addresses.

Whenever neighbor state in neighbor table changes to FAILED, we know that neighbor in unreachable. This state can be achieved in two ways – either destination host is down or network is under attack and neighbor is unreachable due to bad prefixes attack. In either case we can send packets to default gateway and try to find alternate route to the destination system. If unreachability is because neighbor system is down, gateway will return ICMP neighbor unreachable error message. In other case, if alternate route to destination does exist, gateway will send packets to the destination system and thus communication will be restored. Thus all the packets to destination will now go through the default gateway as it was supposed to be.

V. Implementation

Figure 1 shows network setup for implementation of attack as well as prevention mechanism. All machines M1 to M5 are configured to use Ubuntu with Linux kernel 3.1.10. Machine M5 has two network interface cards and acts as router. Routing is enabled on M5 by enabling packet forwarding. To send router advertisements, router advertisement daemon (RADVD) is installed on M5 and configured to advertise 64 bit prefixes 2000:db8:0:1::/64 and 2001:db8:0:1::/64 on subnet1 and subnet2 respectively.

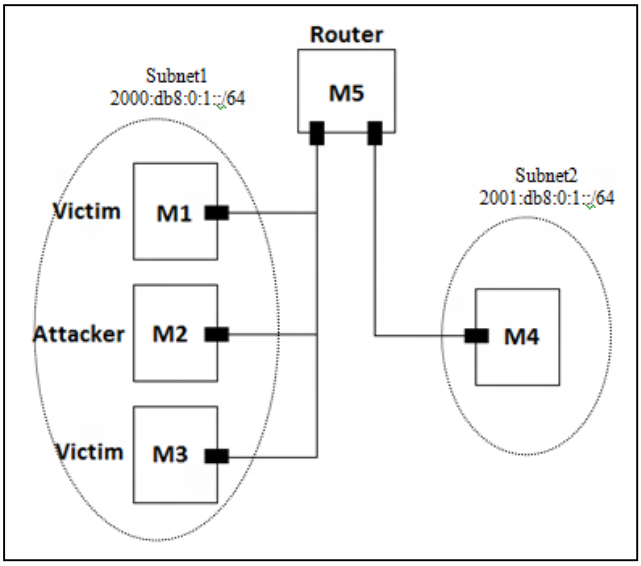

Figure 1. Network Setup

A. *Implementation of Bad Prefixes Attack*

Implementation of attack is done using RADVD daemon. RADVD is used to advertise router advertisement in Linux. It gives option to configure prefix to be advertised, prefix length, RA interval, router lifetime etc. RADVD is configured on machine M2 to advertise prefix 2001:db8:0:1::/64 on subnet1. As a result of this, machines M1 and M3 acquired second IPv6 address starting with advertised prefix and thus treat all the machines with this prefix to be within the same subnet. After the attack, machines M1 and M3 are unable to communicate with machine M4 having subnet prefix 2001:db8:0:1::/64, as they expect it to be present on same network as them. Neighbor solicitations (NS) for machine M4 can be seen on the subnet1 when M1 or M3 tries to communicate with M4. NS on subnet1 tells us that M1 (and M3) treats M4 as immediate neighbor and tries to communicate with it directly, instead of communicating through the default gateway.

B. *Implementation of Solution*

As per solution described in section 4, here we have to monitor reachability state of neighbor entry while sending packets to the given destination. If reachability state is FAILED, instead of sending packets directly to the destination, we send it through default gateway of the network.

Linux stores neighbor related information, such as timer information, hardware address, reachability state etc. in *struct neighbour*. Neighbor entry status can be NUD_INCOMPLETE, NUD_REACHABLE or NUD_FAILED depending on reachability state of neighbor under consideration. Reachability state of neighbor is maintained in variable *nud_state*. Whenever user requests neighbor lookup for particular destination, a new *struct dst_entry* instance is created. *dst_entry* maintains pointer to *neighbour* entry, which is next hop for communicating with given destination, along with other destination related information such as output device, path, reference count etc.

In Linux method *ip6_dst_lookup_tail()* is used for next hop resolution. It calls *ip6_route_output()* function to get *dst_entry* corresponding to given destination address. *dst_get_neighbour()* accesses *dst_entry* structure and returns neighbor entry linked to given *dst_entry*.

Implementation checks neighbor entry status before using it for outgoing packets. Neighbor entry state NUD_FAILED states that no neighbor advertisement has been received for neighbor solicitations sent and thus neighbor is unreachable. Under attack scenario, this situation can be because host is trying to communicate to the destination directly instead of through the default gateway. Thus we redirect such packets to route through the default gateway.

To send packets through the default gateway, we have to link neighbor entry of default gateway with *dst_entry* of given destination. Thus instead of returning *dst_entry* for requested destination, we can simply return *dst_entry* of default gateway from *ip6_dst_lookup_tail()* method, thus sending packet through the default gateway. All such packets destined to host which are not reachable directly, will be sent through default gateway. If destination is reachable through default gateway, we will receive reply from the host shortly. If default gateway doesn't know route to destination, it will send an ICMP error message destination host unreachable to originating host.

To get *dst_entry* of default gateway, we call *ip6_route_output()* for destination address 0 (dst_addr_any). Return value of this call, which is *dst_entry* of default gateway, is then sent back to the caller.

Source address to be set in the outgoing packet also needs to be changed when we return neighbor entry for default gateway instead of the one with neighbor state NUD_FAILED. If we are sending packet to default gateway instead of sending it directly to the destination system, source address in outgoing packet needs to be set to globally routable address instead of link local address. Address with global scope, excluding the address having same prefix as that of destination system, is chosen from list of IP addresses assigned to the interface using source address selection algorithm.

ipv6_dev_get_saddr() is used in Linux to select source address for given destination address. Replica *ipv6_dev_get_saddr_exlude()* of above method is created to exclude source addresses having same prefix as that of destination address. Method *ipv6_dev_get_saddr_exclude()* is called for source address selection only when next hop for given destination is unreachable and our modified algorithm chooses to send packets through default gateway.

VI. RESULTS

As shown in Figure 1 we have two subnets with prefixes 2000:db8:0:1::/64 (subnet1) and 2001:db8:0:1::/64 (subnet2), connected by a router. Communication from a node1 in subnet1 to a node2 in subnet2 is demonstrated below with the help of ping protocol.

As shown in Figure 2 below, nodes in subnet1 can communicate with nodes in subnet2 in normal scenario. All echo requests sent by node1 are received by node2, and corresponding echo reply is sent by node2 to node1.

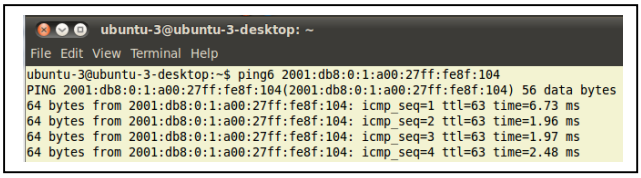

Figure 2. Reachability Before Attack

Figure 3 below shows scenario after attack is performed. Node1 acquires IPv6 address with prefix 2001:db8:0:1::/64 and thus tries to find node2 in the same network as node1 through neighbor discovery. Corresponding neighbor entry with reachability state FAILED is shown in Figure 4.

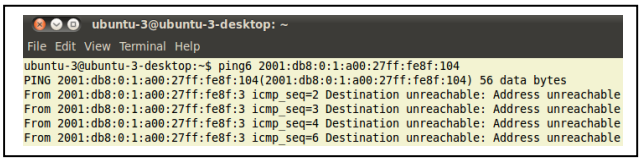

Figure 3. Reachability After Attack

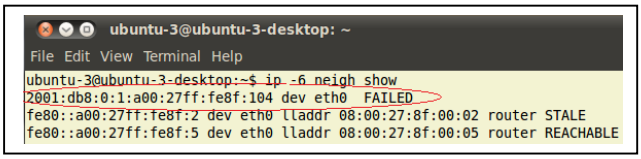

Figure 4. Neighbor Table After Attack

Figure 5 shows network scenario after preventive measures are applied. Initially, when host tries to communicate directly with the destination, we get "Address Unreachable" reply. Thus communication is broken. As a result neighbor status in neighbor table changes to FAILED as shown in Figure 6. This can be either because neighbor system is down or network is under attack. In either case, node1 sends successive packets for node2 via default gateway, thus communication is restored.

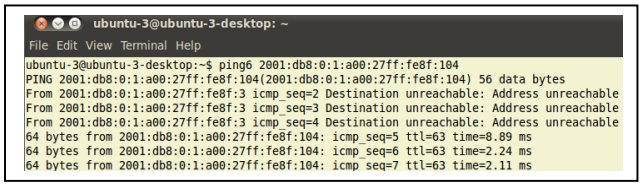

Figure 5. Reachability After Prevention

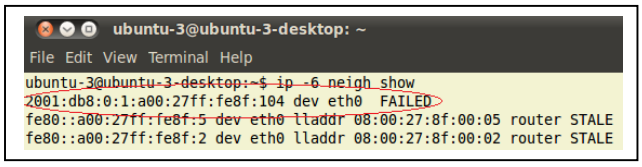

Figure 6. Neighbor Table After Prevention

VII. Conclusion and Future Work

The solution described in this paper prevents bad prefixes attack when implemented in all hosts of participating networks. Unfortunately existing implementations of IPv6 have no built-in prevention mechanism against bad prefixes attack and updating all hosts to use this solution will take some time. Any system in the network which has not implemented the solution described above is still susceptible to bad prefixes attack. This work can further be expanded to design solution where gateways, rather than hosts, play prominent role in preventing attacks within their own subnets.

References

[1] S. Deering, R. Hinden, "RFC 1883: Internet Protocol, Version 6 Specification", Dec 1995, Status: Obsoleted by RFC 2460

[2] S. Deering, R. Hinden, "RFC 2460: Internet Protocol, Version 6 Specification", Dec 1998, Status: Draft Standard

[3] J. Bound, B. Volz, T. Lemon, C. E. Perkins, and M. Carney, "RFC 3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," July 2003, Status: Proposed Standard.

[4] S. Thomson, T. Narten and T. Jinmei, "RFC 4862: IPv6 Stateless Address Auto-configuration", September 2007

[5] Xinyu Yang, Ting Ma and Yi Shi, "Typical DoS/DDoS Threats under IPv6", ICCGI '07, Proceedings of the International Multi-Conference on Computing in the Global Information Technology.

[6] Abdur Rahim Choudhary, "In-depth Analysis of IPv6 Security Posture", 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2009.