IPv6: Now You See Me, Now You Don't

Matthew Dunlop*† Stephen Groat*† Randy Marchany† Joseph Tront*
*Bradley Department of Electrical and Computer Engineering
†Virginia Tech Information Technology Security Office
Virginia Polytechnic Institute and State University, Blacksburg, VA 24060, USA
Email: {dunlop,sgroat,marchany,jgtront}@vt.edu

***Abstract*—Current implementations of the Internet Protocol version 6 (IPv6) use stateless address auto configuration (SLAAC) to assign network addresses to hosts. This technique produces a static value determined from the Media Access Control (MAC) address as the host portion, or interface identifier (IID), of the IPv6 address. Some implementations create the IID using the MAC unobscured, while others compute a onetime hash value involving the MAC. As a result, the IID of the address remains the same, regardless of the network the node accesses. This IID assignment provides third parties (whether malicious or not) with the ability to track a node's physical location by using simple tools such as ping and traceroute. Additionally, the static IID provides a means to correlate network traffic with a specific user through simple traffic analysis. We examine the techniques used to create autoconfigured addresses. We also discuss how these techniques violate a user's privacy. The serious breaches in privacy caused by SLAAC need to be addressed before deployment of IPv6 becomes widespread. To that end, we provide a detailed taxonomy of different methods for obscuring IPv6 autoconfigured IIDs.**

***Index Terms*—IPv6 addressing, privacy protection**

I. Introduction

The next generation of Internet protocol, the Internet Protocol version 6 (IPv6), implements new features based on the existing Internet Protocol version 4 (IPv4). One major change, and the driving force behind IPv6, is the address architecture. The address space in IPv4 is limited to 32 bits. Unallocated addresses in IPv4 are quickly being depleted and will be exhausted by early to mid 2011 [6], [13]. To combat the shortage of addresses in IPv4, IPv6 employs 128-bit addresses. With the current number of Internet-ready devices on the network, the immense address space provided by IPv6 is sparsely populated. However, as new classes of devices become interconnected and networked, manually managing subnets becomes complex and time consuming.

One solution being used to solve the problem of subnet management in IPv6 is stateless address auto configuration (SLAAC). SLAAC allows an administrator to configure the network and subnet portion of the address, while each device automatically configures the host portion, or interface identifier (IID), of the address. The IID is often formed by extending the 48-bit Media Access Control (MAC) address to 64 bits, spanning half of the IPv6 address.

Using a node's MAC address in the IID has serious unintended consequences to a user's privacy. While the observation that this addressing scheme could allow an attacker to analyze payload, packet size, and packet timing was made in RFC 4941 [11], the privacy implications that arise from stateless address generation have not been addressed. The issue is not only that the MAC address is used as the IID, but also that the IID remains static. As a result, no matter what network the node accesses, the IID remains the same. Consequently, simple network tools such as ping and traceroute permit tracking a node's geographic location from anywhere in the world. All the cyber-stalker needs to know is the location of the subnet.

Such "cyber-stalking" is not possible in IPv4. In IPv4, a node's MAC address is restricted to the local subnet. Additionally, a node's location is often obscured through the use of the Dynamic Host Configuration Protocol (DHCP), which leases host addresses based upon availability. Furthermore, the deployment of carrier-grade Network Address Translation (NAT) in IPv4 has the unintentional benefit of protecting a host's identity by placing it within a private address space, which is not globally addressable.

With IPv6, a user's privacy can also be violated through the monitoring of network traffic. Traffic analysis can be used to deduce an identity by correlating traffic captures from a specific IID. This analysis is possible in IPv4, but only for short periods of time, since DHCP addresses change. In contrast, a static IID permits correlation of a specific user's data over multiple sessions. The deterministic IPv6 addresses that globally tie users to each of their packets make this correlation possible. Once an attacker is able to deduce a user's identity and location, the attacker can then target the user for identity theft or other related crimes. Using static IIDs to monitor traffic for identity theft is one of many potential privacy exploits of deterministic stateless IPv6 addressing.

We will show why deterministic addresses have serious privacy implications and why this issue should be addressed before IPv6 is deployed globally. First, we provide background on IPv6 in Section II. Section III describes how the deterministic IID is formed. We discuss the privacy implications in Section IV. In Section V, we provide our taxonomy of methods for hiding users' IIDs. In Section VI, we discuss future work. We conclude in Section VII.

II. Background

The tremendous growth of the Internet has created the need for a new version of the Internet Protocol. Despite the advent of technologies such as NAT, IPv4 will soon be unable to support the addresses needed. Increasing the address space was the main motivation for developing IPv6. Researchers saw the

 ISBN:978-1-61208-113-7

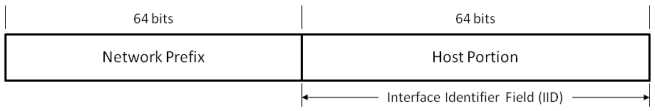

Fig. 1. IPv6 128-bit address format

need for more address space, however, as an opportunity to improve upon an already successful IPv4.

A. Benefits of IPv6

As previously discussed in Section I, IPv6 was primarily developed to support more address space in the Internet. An IPv4 address, consisting of 32 bits, provides approximately 4.2 billion possible address combinations. This address space is not sufficient to support the emerging myriad of Internet-capable devices. Therefore, the IPv6 address was expanded to 128 bits. This new address size allows for 2^{128} possible addresses, approximately $5 \cdot 10^{28}$ addresses for every one of the 6.8 billion people [17] in the world.

In addition to the larger address space, IPv6 was designed with five other main improvements. The first is simplifying the header format to 40 bytes. The second improvement makes the number of IP options extensible by moving the options out of the header and into the payload of the packet. Third, the protocol was designed to be extendible, allowing for the future definition of additional options. Flow labeling, the fourth improvement added to IPv6, allows for classification of packets belonging to particular flows. With flow labels, each router can determine which flow a packet belongs to and prioritize the packets appropriately. The fifth and final major improvement in IPv6 is the integration of authentication and encryption into the protocol stack. In IPv4, Internet Protocol security (IPsec) [9] was developed as an add-on so IPv4 did not have to be redefined. As a result, there are inefficiencies with its implementation. IPv6 solves this by integrating IPsec.

B. Stateless IPv6 Addressing

The large size of the IPv6 address space requires a new network address configuration architecture to simplify network administration. For this reason, IPv6 combines a Neighbor Discovery Protocol (NDP) [12] with SLAAC to allow for nodes to self-determine their IP addresses. Designed as a replacement for the Address Resolution Protocol (ARP), NDP facilitates nodes within a particular subnet learning of other nodes on the link using Internet Control Message Protocol version 6 (ICMPv6) messages. Once an NDP message is received, the node uses the network portion of the address to configure the first 64 bits of its IPv6 address. For the last 64 bits, the node automatically configures an address, designated as the IID of the address. The final step combines the 64-bit network address with the 64-bit host address to form a complete 128-bit IPv6 address (See Fig. 1).

NDP and SLAAC eliminate the need for DHCP addressing services currently implemented on the majority of IPv4 networks. DHCP implements a client-server architecture in which

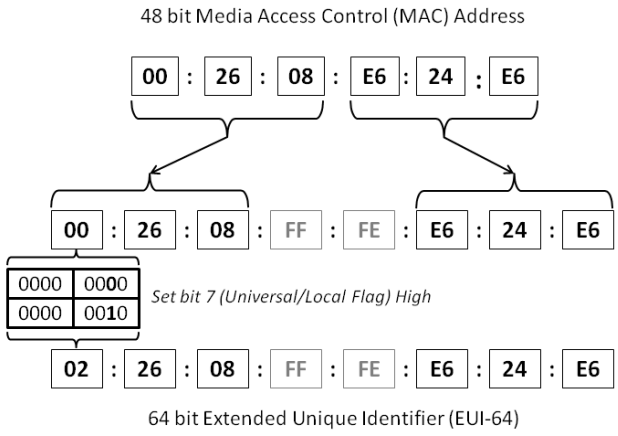

Fig. 2. 64-bit Extended Unique Identifier (EUI-64) format

a DHCP server assigns addresses to clients and keeps state of which addresses have been assigned to particular clients. DHCP also exists for IPv6 in the form of DHCPv6. The sparse address space and the ease of address autoconfiguration, however, make DHCPv6 addressing an unnecessary service. The extra expense and network complexity involved in DHCP addressing have been removed with NDP and SLAAC.

III. Deterministic IID

Due to the current accepted definition of SLAAC on most operating systems, the IID of a node's IPv6 address is deterministic across networks. For the last 64 bits, the node automatically configures an address based upon the MAC address of its network interface. By extending the 48-bit MAC address to 64 bits through the EUI-64 format [7], the IID of an IPv6 address is created. The EUI-64 format splits the 48-bit MAC address into two 24-bit halves. The 16-bit hex value `0xFFFE` is inserted between the two halves to form a 64-bit address. Also, the universal/local flag, located at bit seven of the 64-bit host portion, is set to universal. Fig. 2 illustrates this process.

While different operating systems configure IPv6 addresses differently, no current operating system implementations of IPv6 stateless addressing dynamically obscure the IID of all IPv6 addresses on the system. OS X and common Linux distributions, such as CentOS and Ubuntu, follow the EUI-64 format. The MAC address appears virtually unaltered in the IPv6 address. The Windows operating system obscures the host portion of an IPv6 address according to RFC 4941 and sets a temporary address [11], [16]. Windows operating systems, however, also carry another IPv6 address used for neighbor solicitation. This other IPv6 address contains an IID that is obscured but never changes, regardless of the subnet the node connects to. Not dynamically obscuring a user's host portion for all of the IPv6 addresses associated with a system threatens a user's privacy. The static IID currently implemented in major operating systems can be linked to a particular node, even as the node changes networks.

 ISBN:978-1-61208-113-7

Many mobile devices, such as Android and iPhone, support IPv6 in WiFi. Their implementations follow the EUI-64 format, providing these mobile devices with static IIDs that are easily tracked on their WiFi connections. Since most users frequently carry their mobile devices and leave them on and connected, the ability to track a user is increased dramatically. While the need to address the privacy concerns in Mobile IPv6 has been identified [3], [10], [14], it does little good until the privacy concerns due to IID tracking are addressed. Since Mobile IPv6 would only be applied to the cellular connections and the majority of these wireless devices also deploy WiFi, users can still be tracked through their wireless devices as they move between different WiFi networks. Therefore, address privacy must be dealt with for all connections of a mobile device to assure complete privacy.

IV. Privacy Implications

The static IID created by the EUI-64 format and the Windows operating systems compromises a user's privacy. Creating a static IID from a MAC address allows nodes to be logically and geographically tracked as they travel to different networks. Since the EUI-64 format results in a deterministic IID, users can be tracked on a network by scanning different subnets and searching for the MAC-generated IIDs. Using simple commands such as ping and traceroute, the location of a user can be determined with reasonable geographic accuracy. Even the Windows obscuration of the IID within the IPv6 address does not protect a user. By locally capturing a user's traffic once, a specific user can be paired with the deterministically obscured IID and tracked with the same technique of searching subnets as used for unobscured host addresses. Since the obscuration occurs independent of the network, a Windows host carries the same obscured IID between networks.

By monitoring the traffic on a network over an extended period of time, a single user's traffic can be identified and analyzed. Armed with this data, a third party (whether malicious or not) can potentially tie a device to its actual user. As the user crosses different subnets, traffic can be collected and correlated by examining the static IID. This vulnerability to tracking does not typically apply when using IPv4. Most medium to large IPv4 networks implement DHCP, which changes user addresses randomly. As a result, DHCP logs are needed to tie traffic sniffed from a network with a particular user. Due the to the deterministic IID in IPv6 SLAAC, simple filters could be created to filter the traffic of a single user on any subnet. This would allow an interested party to identify and monitor a user's on-line activity through traffic analysis. In a dual-stack implementation where a node uses a mixture of IPv4 and IPv6, special ICMPv6 Neighbor Solicitation messages can provide an interested party with the IPv4 address linked to an IPv6 address. This correlation allows for traffic collection to extend to IPv4 for a single session.

Tracking users or monitoring their on-line activity is not the only concern. If it was known that location or traffic monitoring was occurring, a malicious host could spoof the IID of an innocent node. The malicious node could then masquerade as the innocent node and create false traffic or locations using the innocent node's IID.

V. Possible Solutions

There are three primary classifications for protecting a user's IID from tracking and monitoring. The most straightforward classification is dynamic obscuration of the IID. The second classification is third party address assignment. The final classification is address tunneling.

A. IID obscuration

There are two methods proposed for obscuring the IID when using stateless generation of IPv6 addresses. The first method uses cryptographically generated addresses (CGAs) [2]. The second method uses privacy extensions for static IIDs [11].

1) Cryptographically Generated Addresses: CGAs are IPv6 IIDs that are generated by hashing a sender's public key and other parameters [2]. Although CGAs can be used with multiple applications, they were originally designed to work with the SEcure Neighbor Discovery (SEND) protocol [1] to prevent denial of service attacks. Since CGAs obscure a sender's IID, they could prevent tracking a user through their autoconfigured IPv6 address. There are, however, a number of disadvantages to using CGAs for dynamically obscuring an IPv6 IID.

The main disadvantage to using CGAs is the cryptographic cost of CGA generation. In order to generate a CGA, the sender must apply the SHA-1 hash algorithm to the CGA parameters a minimum of two times. The CGA parameters consist of the concatenation of a pseudo-random number, the sender's public key, nine zero octets, and optional fields. The first hash calculation generates Hash2. The ease of achieving an acceptable Hash2 value depends on the strength of the security parameter (Sec). The Sec is a three-bit value used to determine how many leading zeros Hash2 must contain. Hash2 must have $16 \cdot Sec$ leading zeros. If it does not, a pseudo-random number is incremented and Hash2 is recalculated. Hash2 continues to be recalculated until this condition is met. This can result in a large number of hash calculations. In fact, RFC 3972 states that on average it takes O($2^{16 \cdot Sec}$) iterations to generate a CGA [2].

After Hash2 is successfully calculated, Hash1 is calculated using the CGA parameters with the final pseudo-random number used to generate Hash2. The left-most 64 bits of the SHA-1 output are used as the IID with the exception of bits 0-2, 6 and 7 used for other purposes. Duplicate address detection is conducted as required [1]. If a collision occurs, a collision count field is incremented and Hash1 is recalculated. This is repeated at most three times after which, CGA calculation stops [2]. It is possible that after all of these hash calculations no CGA will result and the process will need to start anew.

These repeated hash calculations require minimal overhead for the average personal computer, but are likely infeasible for most handheld devices due to limited power and computational capability. To help mitigate this cost, RFC 3972 discusses that CGAs can be precomputed or offloaded to more powerful

 ISBN:978-1-61208-113-7

computers [2]. This solution, however, does not help a user connecting to a network for the first time, which is not uncommon when roaming with a handheld device. Additionally, the CGA generation cost makes it prohibitive for users to generate new CGAs. Thus, CGAs for subnets will be deterministic.

There are two possible implementation alternatives to reduce the cost of CGA generation. The first involves setting $Sec = 0$. By setting $Sec = 0$, Hash2 is not required to have any leading zeros. This means that the calculation of Hash2 is not required [2]. The problem with this alternative is that the CGA becomes susceptible to a brute-force attack. If the CGA is used only to prevent IPv6 address tracking, this may not be a concern. However, if the intent is to also protect the IID from denial of service or address spoofing, this alternative is not acceptable.

The second implementation alternative is to omit the subnet prefix from CGA generation [2]. By omitting the subnet prefix, senders need only calculate the CGA once. The same CGA can be used for multiple subnets without having to go through the expensive CGA generation for every subnet. The downside is that the IID is again deterministic across multiple subnets. This alternative maintains the benefit of protection against brute-force attack through the security parameter, but leaves senders vulnerable to address tracking.

Even if users do change their CGA every time they connect to a subnet, they can still be tracked. Senders using CGAs send their CGA parameters along with their IPv6 address for verification purposes. The CGA parameters contain the users' public key. Unless senders also generate new public keys for every connection, they can be tracked through their CGA parameters.

An additional disadvantage of using CGAs is that an attacker can easily impersonate or slander a user by forming a CGA claiming to be the target. Asymmetric key pairs are self-generated to eliminate the key management infrastructure. As a result, anyone can generate a key pair and claim to be someone else. RFC 3972 claims that CGAs protect against address spoofing [2]. This is only true in the case where an attacker attempts to hijack an existing session between two nodes. An attacker is free to initiate new sessions claiming to be someone else. By doing this, an attacker can make it appear as if a target is behaving in a certain way to others who are monitoring the target on the network. Spoofing is common in IPv4, but CGAs in IPv6 make a spoofed address appear more authentic.

Requiring the use of public keys to form CGAs is another drawback. One of the strengths of stateless autoconfiguration of IPv6 addresses is that it is transparent to users. Requiring users to generate asynchronous key pairs removes this transparency. Many users do not have asynchronous key pairs or even understand the concept. As a result, most users will not implement CGAs to obscure their addresses.

The remaining vulnerabilities do not affect tracking of users, but how an attacker can take advantage of flaws in CGAs to achieve malicious results. For instance, a man-in-the-middle can cause all data checks to fail by modifying any of the CGA parameters sent with the IPv6 address. This can be accomplished by changing the collision count to a number greater than two, because the collision count is only valid for values of 0-2. An attacker also can hijack a connection by finding a CGA hash collision. Only 59 bits (64 bits minus bits 0-2, 6, and 7) of the SHA-1 hash are used for the CGA. Using the birthday attack [15], a collision can be found in $2^{59/2}$ guesses on average.

Despite the numerous disadvantages of CGAs, there are several advantages. The primary advantage is, of course, that CGAs obscure a sender's IID. Even if senders do not go through the costly procedure of computing a new CGA every time they connect to a network, they will at least have a different deterministic CGA for every subnet. This will make an attacker's attempt at tracking a target much more difficult. In order for an attacker to track a target, the attacker will need to know the sender's CGA on every subnet. Another benefit of using CGAs is that they require no key management infrastructure [2]. Since CGAs can use user-generated asymmetric key pairs, there is no need for a certification authority, making CGAs scalable.

2) Privacy Extensions: The privacy extensions proposed in RFC 4941 provide another way to obscure the IPv6 IID. Where the goal of CGAs is primarily to provide security of IPv6 addresses for the SEND protocol as discussed in Section V-A1, the goal of implementing privacy extensions is to prevent an attacker from correlating network traffic with a particular user. Privacy extensions produce a random IID by hashing the concatenation of a user's IID and a 64-bit "history value." This "history value" is initially generated from the rightmost 64 bits of a hashed random number. Subsequently, the "history value" becomes the previously calculated 64-bit hash result. The random IID is formed using the leftmost 64 bits of the resultant hash calculation with the "u" bit (bit 6) set to zero for local scope. Duplicate address detection is then performed on the random IID to detect duplicate IIDs on the local network. If a duplicate IID is detected, a new "history value" is formed and the process is repeated [11].

Using privacy extensions to obscure a user's IID is much more appealing than using CGAs for several reasons. First, the cryptographic cost is much lower. Assuming no duplicate address is detected, privacy extensions require only one hash calculation by the sender at IID generation and none by the receiver. With CGAs, the sender requires on average $O(2^{16 \cdot Sec})$ hash calculations to generate a CGA, and the receiver is required to complete two hashes to verify the CGA. The cryptographic cost of using privacy extensions is more feasible, allowing a node to change their IID often.

The use of privacy extensions does not require the use of a public key [11]. As a result, implementation of privacy extensions are more transparent to the user. Additionally, there are no required accompanying parameters that can be used to link a sender to the random IID.

Privacy extensions are more effective at obscuring IIDs than CGAs because of the lifetime of a generated IID. Having users generate new IIDs when they connect to new subnets

 ISBN:978-1-61208-113-7

effectively masks users' activities on the Internet, but it does nothing for users that never migrate from a particular subnet. A CGA user that never moves between subnets (e.g., a desktop computer) would not necessarily ever generate a new random IID. The privacy extensions specification uses a TEMP_VALID_LIFETIME parameter to set the maximum amount of time a random IID can be used before needing to be regenerated [11]. This feature improves on DHCP addresses that may last months or longer.

Implementing random IIDs through privacy extensions does have disadvantages. Just as with CGAs, IID collisions are possible [11]. Since only 64 bits of the hash calculation are used, the chance of hash collisions increases. Similar to CGAs, the generation process terminates after a set number of IID collisions.

As mentioned previously, privacy extensions specify several different parameters to limit the time a random IID is valid. The default values of these parameters are set too long. The TEMP_PREFERRED_LIFETIME parameter is set to one day and specifies the preferred life of a random IID. The default value of TEMP_VALID_LIFETIME is one week. An application can force an address to use TEMP_VALID_LIFETIME rather than TEMP_PREFERRED_LIFETIME. Additionally, RFC 4941 states that a new random IID *should* be computed when connecting to a new subnet [11]. It does not state that this *must* be done. Users choosing not to change IIDs and accepting the default expiration times could be monitored for up to a week at a time before an attacker needs to reestablish an identity pairing. Fortunately, RFC 4941 allows users to modify these defaults.

An unfortunate side-effect of privacy extensions is that IIDs are obscured from everyone, including network administrators. Combined with frequently changing and obscured IIDs, privacy extensions make fault isolation and debugging difficult [11]. With CGAs, public keys are tied to addresses. When DHCP is used, system administrators can track changes in addresses, while still protecting a user's identity.

CGAs are more robust than privacy extensions in that they can protect a sender's address from spoofing once a session has been initiated. Since there is no verification process implemented by privacy extensions, an attacker can easily inject new traffic claiming to be the sender. However, preventing this type of malicious activity is not one of the design goals of privacy extensions. This type of attack, however, is mitigated by frequently changing a user's IID.

B. Third Party Address Assignment

DHCPv6 provides third party address assignment for IPv6 [5]. Instead of allowing a client to configure his/her own address, a DHCP addressed network requires a DHCP server in order for a client to get an address on a network. DHCP addresses are leased to clients when they connect to the network. If the network is not overloaded with clients, a client may receive the same DHCP address each time he/she connects. For heavily populated networks, clients may receive different addresses each time they connect.

There are several advantages to using DHCPv6. First, DHCPv6 provides IIDs that are not necessarily tied to a user. A user may receive a different DHCPv6 address each time he/she connects. Whereas collisions were possible with the address obscuration techniques discussed in Section V-A, a properly configured DHCPv6 server should not issue duplicate addresses. Although DHCPv6 can operate without any cryptographic cost for address generation or verification, it is worth noting that DHCPv6 can be configured to use optional authentication [5]. DHCPv6 authentication uses shared keys which, although more cryptographically efficient, do not scale well.

Despite these advantages, DHCPv6 does not necessarily protect a user's identity well because the IPv6 address space is sparsely populated. As a result, there will likely be little competition for addresses when connecting to even the most populated subnets. Therefore, users should expect to get the same address each time they connect to the network. This may change over time as more devices connect to the network, and the address space becomes more densely populated. However, the DHCPv6 specification promotes static addresses. Unless specifically requested by the client using the Identity Association for Temporary Addresses (IA_TA) option, the client will be issued a non-temporary address [5]. A static address, whether generated by the client itself or issued by a DHCP server, exposes the user to monitoring. If IA_TA is implemented, it would be provided through the use of privacy extensions discussed in Section V-A2.

Even if temporary addresses are requested, an attacker may still be able to monitor a user. For example, the server controls how often a client's address changes, not the client. The client does have the option to specify a preferred-lifetime and a valid-lifetime, but the timing of an address change is ultimately determined by the server. Additionally, communications between client and server use what is called a DHCP Unique Identifier (DUID). The DUID is a globally unique value that should never change [5]. The DUID is used in many of the DHCP exchange messages and could be used by an attacker to track a target's presence on a network. Once the attacker locates the user, the attacker can monitor the DHCP exchanges to harvest the IPv6 address. The scope of this attack is limited to the subnet of the user, the DHCP server, or any relays.

A final downside to using DHCPv6 versus stateless address autoconfiguration is that it must be managed. This could add a tremendous burden on network administrators as more and more devices connect to the network. DHCPv6 also increases the chances of incorrect configuration, which may lead to other vulnerabilities.

C. Address Tunneling

Address Tunneling can be achieved through the use of Internet Protocol Security (IPsec). IPsec is not a very comprehensive method to use for masking IPv6 addresses, but it can provide excellent obscuration from external attackers. IPsec provides authentication and/or encryption to network layer packets. For the purposes of obscuring IPv6 addresses, we re-

 ISBN:978-1-61208-113-7

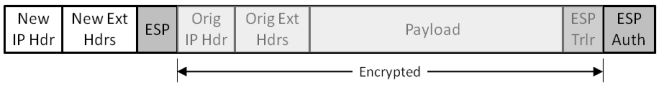

Fig. 3. IPv6 packet encrypted using IPsec in tunnel mode

fer only to the encryption aspect provided by the encapsulating security payload (ESP) as illustrated in Figure 3. Specifically, ESP used in tunnel mode provides address obscuration. When used in tunnel mode, ESP encrypts the entire IPv6 packet and provides a new IPv6 header, complete with new source and destination addresses [8]. The new source and destination addresses are those of the endpoints of the tunnel. It should not be possible for devices external to the tunnel to learn the true identity of the sender or receiver. The tunnel endpoints are typically network gateway devices. Although it is possible for the sender and receiver to act as tunnel endpoints in IPv6, this technique would gain very little privacy. A host acting as its own tunnel endpoint would be easy to link as the actual target host. The ability to link to the target host can be prevented by using one of the obscuration techniques described in Section V-A after applying ESP. Using IPsec for this purpose, however, would then be pointless.

There are three main benefits to using IPsec in tunnel mode to obscure IPv6 addresses. As mentioned previously, the sender's address is hidden from those external to the tunnel. Second, the cryptographic burden of encryption and decryption is offloaded to the gateway devices. This makes address obscuration feasible for devices limited by battery or computational power, such as handheld devices and sensors. Also, since the address used is that of the gateway devices, there are no address collisions as there were in the address obscuration techniques discussed in Section V-A.

Unfortunately, IPsec in tunnel mode does not provide any address protection from an attacker inside the sender's or receiver's subnets. Since address obscuration does not occur until the packet reaches the gateway, an attacker monitoring the subnet will have no trouble monitoring users through their IIDs. This does, however, limit the scope of the attack to the two subnets mentioned.

Perhaps an even bigger issue is the requirement for a key management infrastructure. Wide-scale deployment of IPsec requires a global trust model in place as well as a management infrastructure [4]. In today's infrastructure, only those networks with security requirements utilize IPsec. It is not reasonable to assume that networks would implement IPsec for address obscuration purposes.

VI. Future Work

The next phase of our research is to design and implement an address obscuration technique. Each of the techniques described in Section V has associated shortcomings. Those that obscure the IPv6 IID, are not feasible for resource constrained devices. Third-party address assignment and address tunneling have scope limitations. Our technique is designed to minimize computational complexity while avoiding scope limitations, thus being feasible for power-constrained devices. We also plan to test the validity and overhead of our design using our campus-wide IPv6 production network.

VII. Conclusion

IPv6 is a definite improvement over IPv4, allowing more devices to connect to the Internet using globally unique addresses. SLAAC, however, violates a user's privacy and needs to be addressed before IPv6 is deployed. A number of different methods can be used to obscure a user's IID from monitoring. Each method comes with its associated benefits and shortcomings. Currently, the privacy extensions outlined in RFC 4941 [11] appear to provide the best balance of both. Privacy extensions provide an unmanaged solution to address obscuration with low cryptographic cost. However, the current lack of computational power of many handheld devices combined with possible address collisions are likely obstacles to implementing this algorithm. Regardless of which solution is implemented, some method of obscuring IIDs should be deployed as part of operating systems and embedded devices to protect the privacy of users.

References

[1] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). RFC 3971 (Proposed Standard), Mar. 2005.

[2] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972 (Proposed Standard), Mar. 2005. Updated by RFCs 4581, 4982.

[3] C. Castelluccia, F. Dupont, and G. Montenegro. A simple privacy extension for mobile IPv6. In *Mobile and Wireless Communication Networks, IFIP TC6 / WG6.8 Conference on Mobile and Wireless Communication Networks (MWCN 2004)*, pages 239–249, Oct. 2004.

[4] A. Choudhary. In-depth analysis of IPv6 security posture. In *The 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2009)*, pages 1 –7, Nov. 2009.

[5] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315 (Proposed Standard), July 2003. Updated by RFCs 4361, 5494.

[6] A. Gonsalves. IP addresses predicted to be exhausted in 2011. *InformationWeek*, July 2010.

[7] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), Feb. 2006.

[8] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard), Dec. 2005.

[9] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), Dec. 2005.

[10] R. Koodli. IP Address Location Privacy and Mobile IPv6: Problem Statement. RFC 4882 (Informational), May 2007.

[11] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), Sept. 2007.

[12] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), Sept. 2007.

[13] Remaining IPv4 address space drops below 5%. Available at: http://www.nro.net/media/remaining-ipv4-address-below-5.html/, Oct. 2010.

[14] Y. Qiu, J. Zhou, F. Bao, and R. Deng. Protocol for hiding movement of mobile nodes in Mobile IPv6. In *62nd IEEE Vehicular Technology Conference*, volume 2, pages 812 – 815, Sept. 2005.

[15] W. Stallings. *Cryptography and network security (2nd ed.): principles and practice*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[16] Introduction to IP version 6. Available at: http://download.microsoft.com/download/e/9/b/e9bd20d3-cc8d-4162-aa60-3aa3abc2b2e9/ipv6.doc accessed on 24 May 2010.

[17] U.S. & World population clocks. Available at: http://www.census.gov/population/www/popclockus.html/ accessed on 4 Mar 2010.

 ISBN:978-1-61208-113-7