Back to the Future: Revisiting IPv6 Privacy Extensions*

David Barrera, Glenn Wurster and P. C. van Oorschot
Carleton University
School of Computer Science
Ottawa, ON, Canada

ABSTRACT

We identify issues in current IPv6 privacy extensions and propose improvements that significantly enhance both the flexibility and functionality, to protect a client from being tracked as it moves between different IPv6 networks. This is achieved by generating a new interface identifier for each visited network. We discuss a preliminary Linux implementation of the proposal.

1. INTRODUCTION

While the IPv6 protocol was introduced in 1998 [7], its importance has only recently achieved mainstream attention as IPv4 addresses run out. The address space in IPv4 has shown high resiliency to exhaustion, but it is now reaching critically low levels (less than 8% of the IPv4 addresses were available for allocation as of April 2010[1]). The projected exhaustion date (sometime in mid-2012) has added urgency to the long-standing movement of ISPs, businesses, operating system vendors and end users to begin upgrading to IPv6. Major service providers such as Google and Comcast are testing IPv6 through services made available over both IPv4 and IPv6. Google is currently capable of offering all services over IPv6 for clients who resolve their domains through an approved Domain Name System (DNS) server [9]. Although a large percentage of end users are estimated to have IPv6 capable devices, only 1% of Internet users have Internet-wide network connectivity over IPv6. Over 23% of inter-ISP peering points, however, support IPv6 [19]. It is anticipated that as IPv4 addresses become more scarce, the adoption of IPv6 will necessarily grow substantially.

With more people adopting IPv6, some features of the protocol are slowly being explored by a small user-base. Security issues related to IP packet fragmentation and malicious route headers[2] have been identified and new RFCs addressing those issues have been published [1, 13]. The iterative process of identifying flaws and creating fixes led IPv4 to becoming a stable and mature protocol. Since IPv6 is much newer and just now being broadly deployed, many of its features have not enjoyed broad testing or security analysis. In this paper we concentrate on one such feature: IPv6 privacy extensions.

IPv6 makes available the option for clients to assign themselves an IPv6 address based on a 64-bit prefix periodically broadcasted (by a local server), and a 64-bit value derived from the network interface identifier (usually the MAC address of the network card). Having a globally routable IP address which includes (and therefore reveals to remote servers) the MAC address of a client was regarded as a potential privacy issue, leading to the development of IPv6 privacy extensions [17, 18]. Through the use of these extensions, a host can generate and assign itself a partially randomized (but still valid) IP address at fixed intervals, allowing connectivity without revealing its MAC address. The existing IPv6 privacy extensions are not only important for personal privacy, but also for hiding information which can otherwise allow wide scale targeted malware attacks (cf. Internet-scale *hitlists* [20]).

1.1 Problem

The problem we address is that of tracking users based solely on IP address information. More specifically, for an adversary who has recorded multiple IPv6 addresses, it should be hard for them to identify two or more of those addresses as corresponding to the same client. In practice, the adversary could be a corporation wishing to provide targeted services only to users that fit a specific profile (e.g., users who have visited more than 3 coffee shops in the past week or have been at 5 airports in the past month). Other adversaries may include governments or eavesdroppers who wish to follow users as they roam through multiple locations. We note that tracking users in IPv6 has been partially addressed by the IPv6 Privacy Extensions [18]. The specification, however, has a number of design issues (see Section 4) which can lead to the desired goal (keeping the client-to-IP-address mapping private across locations) not being achieved.

Without privacy extensions, tracking is possible because the last 64 bits of a client's IPv6 address are constant: if a client has IPv6 address `1::2345:6789` while on network 1, the client may have IPv6 address `2::2345:6789` when using network 2. This provides the means to track users as

*Version: September 14, 2010. Contact author: `dbarrera@ccsl.carleton.ca`. A version of this technical report is to appear in ;login: (Vol. 35, No. 6)

[1] `http://www.potaroo.net/tools/ipv4/index.html`

[2] `http://www.securityfocus.com/bid/23615`

they move between IPv6-capable networks. Existing privacy extensions essentially randomize the last 64 bits.

1.2 Contributions

Our main contribution is a new technique and prototype implementation for generating private IPv6 addresses. Our proposal differs from current IPv6 privacy extensions [17, 18] in that it can be configured for different privacy requirements, and is capable of providing private addresses even if an administrator has configured the network to deter their use. Our proposal provides as much privacy as IPv4 and has minimal overhead. We also describe the implementation of a Linux kernel prototype of our proposal.

In this paper, we identify issues with the current state of IPv6 privacy extensions that could lead to a downgrade attack, enabling eavesdroppers to track IPv6 users as they move through networks. We also identify issues with currently deployed implementations of IPv6 privacy extensions in modern operating systems, and propose a more flexible and robust algorithm for generating private IPv6 addresses.

Through this paper, we solicit feedback from the community on the seriousness of current privacy leaks at the network level; indeed, upper layer protocols are only as secure as their underlying counterparts. Our proposal presents an implementation with several options for generating private IP addresses, providing different degrees of privacy. We encourage discussion on the preferred configuration options for individuals with varying privacy requirements.

The sequel is structured as follows. Section 2 defines terminology and relevant IPv6 background. Section 3 explains our threat model and new proposal for protecting against tracking IPv6 users. A prototype for Linux is also discussed. Section 4 discusses related work. We conclude in Section 5.

2. BACKGROUND

This section reviews relevant terminology and background on how clients are assigned IPv6 addresses, and on the originally proposed privacy extensions.

2.1 Terminology

We use the following generally-accepted terminology [24].

- **Prefix:** the first (most significant) 64 bits of an IPv6 address. A prefix can be learned through periodic router advertisements, DHCPv6, or self-assigned for loopback and link local addresses.
- **Interface identifier:** the least significant 64 bits of an IPv6 address. The prefix and interface identifier together fully specify an IPv6 address.
- **Preferred lifetime:** a lifetime associated with a particular IPv6 address during which the address should be used to initiate connections. Once the lifetime expires, the IPv6 address is *deprecated*, but still active for the remaining open connections. The address remains deprecated until the valid lifetime expires.
- **Valid lifetime:** a lifetime associated with a particular IPv6 address. When it expires, the IPv6 address is removed from the network interface by the kernel and no longer used.

2.2 IPv6 Addresses

Clients can obtain IPv6 addresses through one of three methods. (1) The user manually assigns a valid IPv6 address to an interface; (2) a periodically advertised prefix is appended to the self-generated interface identifier; or (3) a DHCP server is queried and the received response used. This section reviews methods (2) and (3).

2.2.1 Stateless address auto-configuration

IPv6 provides a method for clients to automatically assign themselves a valid IPv6 address based on periodically broadcasted router advertisements. In a typical setup, a router broadcasts the IPv6 prefix that all clients should prepend to their auto-configured interface identifier. Early RFCs [23] suggested that clients use their network MAC address in the generation of the interface identifier. The rationale was that this provided sufficient uniqueness, and would require no persistent storage. A later RFC [24] removed the MAC address suggestion, allowing hosts to choose their own method for generating interface identifiers. The interface identifier is always appended to the network prefix, after which the Duplicate Address Detection (DAD) algorithm is run by the client to ensure that the address is unique to the network segment, and therefore globally unique (as prefixes are also unique).

In cases where a MAC address is used as the interface identifier (still currently the default behaviour of Linux and Mac OS), the IPv6 address reveals information which can be used to identify the client hardware. This ability to determine the hardware configuration of a machine may lead to additional information about the client being revealed on the network (e.g., Mac OS runs on a specific underlying hardware platform, allowing the identification of Apple users based on MAC addresses [11]). This ability to determine the characteristics of a client through the MAC address can be used in a targeted attack on a user or organization (e.g., sending a malicious PDF that only exploits Mac OS). Bellovin et al. [6] argue that the MAC address could also be used by IPv6 worms to target specific hosts.

2.2.2 Inadvertent IPv6 Users

We define *inadvertent IPv6 users* as users who unknowingly use IPv6 to connect to remote servers. While the vast majority of Internet users currently use IPv4, modern OSs attempt to use IPv6 by default when resolving hosts. In a typical network, connecting (and therefore revealing the source IP address of the connection) to a remote server over IPv4 will not typically allow the server to track the individual or identify the network hardware (see Section 2.2.1). Connecting to the same server over IPv6 may reveal sufficient information to track the individual and identify hardware. Because stateless address auto-configuration does not depend on additional client software (other than an updated kernel), it is likely to cause inadvertent IPv6 use. This increases the importance of IPv6 privacy extensions that truly provide protection and information hiding.

2.2.3 DHCPv6

With the addition of stateless address auto-configuration for IPv6, hosts can obtain network information and learn how to route packets without installing additional software. While this may seem ideal from a network management standpoint (e.g., set up a route prefix advertisement daemon and IPv6 just works), there may be other configuration parameters needed by hosts in order to actually communicate with external hosts. Parameters will vary from network

to network, but some include WINS, NTP, NETBIOS and DNS.[3]

There are cases where administrators may choose to replace stateless auto-configuration with DHCPv6, or use both simultaneously. When using DHCPv6 for address assignment, the server keeps track of assigned addresses and the hosts using them (similar to DHCP in IPv4). When using both, a host obtains its IPv6 address through stateless auto-configuration and other information through a server on the local network. The issue of tracking clients using IPv6 is specific to those who obtain an address through stateless address auto-configuration.

2.3 Original Privacy Extensions

IPv6 privacy extensions for stateless address auto-configuration [18] were proposed specifically to address privacy concerns with having a static and globally unique interface identifier. Concerns that a well-placed sniffer (or prolific ad network) might track users as they roam through different networks are partially mitigated by privacy extensions through using periodically changing random interface identifiers. RFC 4941 specifies the algorithm used to generate a random identifier as well as when to update it. As shown in Figure 1, a hash function (MD5 is suggested in the RFC [18]) is used to generate the interface identifier. The first 64 bits of output are used as the interface identifier, while the last 64 are stored for the next iteration of the algorithm, which takes place every x seconds (or when a duplicate address is detected by the client). The first iteration of the function uses a random value as the history value.

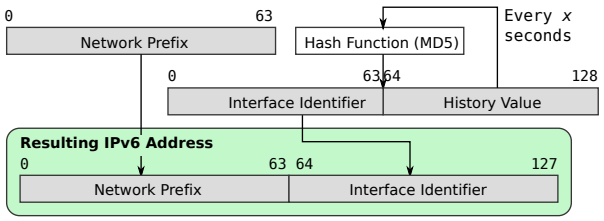

Figure 1: Original privacy extension address generation

The current specification has two important limitations.

1. The only configurable parameter is the interval at which new random interface identifiers are generated. The default interval is to generate a new identifier every 24 hours. This still allows a user moving between two or more IPv6 networks in a 24 hour window to be tracked by an adversary (since the client's interface identifier will not change during this time, even if the network prefix does). The (expert) user can configure the regeneration interval to be smaller, at the expense of no longer maintaining long-lasting connections (e.g., SSH or movie downloads).

2. The intervals are dependent on the configuration of the network. If a user has configured the interval for regeneration of addresses to be small, but the network advertises smaller intervals, the smallest takes precedence. This means that if the network is configured to advertise prefixes with valid lifetimes of 60 seconds, a user with privacy extensions enabled will generate a new and different IPv6 address roughly every 60 seconds. This will severely impact user experience: no connection made will last more than 60 seconds.

The latest RFC for privacy extensions [18] also specifies that system implementers should add an option for the user to enable or disable random interface identifiers on a per-prefix basis. This is similar to our proposal in that a new full IPv6 address is generated when the prefix changes (the user changes networks), but differs in that they rely on the client to maintain a list of networks for which privacy extensions should be enabled (or disabled), and do not use the prefix directly in the generation of random interface identifiers.

3. NEW PROPOSAL

3.1 Assumptions and Threat Model

In this paper, we focus on protecting clients who configure their IPv6 address through router advertisements from being tracked as they move between IPv6 enabled networks. We do not address clients configured through DHCPv6, or clients with static IPv6 addresses. We assume that each IPv6 network visited by a client is associated with a distinct prefix (routing problems result if two networks share the same IPv6 prefix).

We assume that the attacker does not have access to the LAN segment, and hence can not associate IPv6 addresses with MAC addresses, but we do not assume that the network administrator is totally benign. We assume the network administrator is capable of modifying router advertisements, forcing users to renew their IPv6 address often. We assume an attacker attempting to track the client can see traffic generated with each IPv6 address the client uses. We do not attempt to protect against tracking clients using higher level protocols such as HTTP [14, 8].

3.2 New Privacy Extensions Proposal

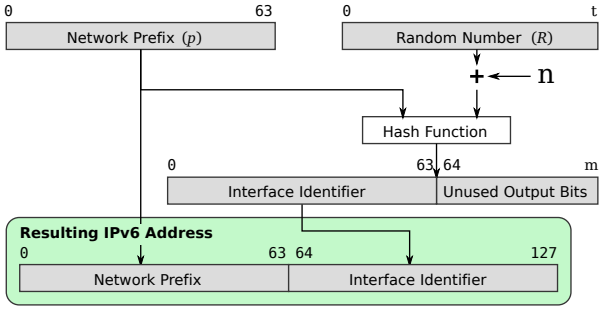

Figure 2: Generation of new IPv6 addresses

We now describe the way in which IPv6 addresses are generated, and the different configuration options for our prototype implementation.

The new proposal is for clients to generate IPv6 address interface identifiers (I) through hashing the IPv6 prefix advertised by the router advertisement daemon (p) with a t-bit random number (R) incremented by n (in order to resolve

[3]Current implementations also support sending a list of IPv6 DNS servers, but a separate client side application is needed to read the received list and apply the changes to the system.

duplicate addresses). R is generated locally and does not leave the client. The generated interface address is composed of the first 64 bits of the resulting m-bit hash value, as illustrated in Figure 2.

$$I = H(p|(R+n))$$

We require a pre-image resistant hash function H [15] so that given the prefix and interface identifier, an attacker can not determine R. Keeping R hidden prevents an attacker from determining that two distinct IPv6 addresses correspond to the same client. There are no compatibility or interoperability concerns should two clients choose to use different hash functions in generating the interface identifier.

To ensure that a new I is generated for every new network prefix (which is not possible in the current privacy extensions), p is included in the hash. There are several options regarding when to change R (see Section 3.2.1), allowing the client control over when to start using a new interface identifier. To prevent known attacks [22, 15] against guessing R, its length (t) should be sufficient (e.g., $t = 256$ bits should certainly be enough), and it should be set from a cryptographically secure random number generator.[4]

Should a client discover (through duplicate address detection, as explained in Section 2.2.1) that it is attempting to use the same generated IPv6 address as another client on the local network (an unlikely scenario), the client should generate a new I (and hence IPv6 address) by incrementing n and recomputing the hash. The client should reset n to 0 on reboot and whenever p or R changes.

3.2.1 Generation of new random numbers

Our proposal includes several options on when to (re)generate R, resulting in a changed IPv6 address. The different options provide different levels of privacy protection, which we now discuss.

1. **Generate R on OS install.** If R is generated during system install and then never changed, I will change when the network prefix advertised by the router changes. As long as p remains constant, so will I. This option is useful for laptops in enterprise environments. As long as the laptop is on the corporate network, the IPv6 address will be fixed. When the laptop is removed from the network (e.g., the employee goes to HotSec), the interface identifier I will change, preventing the employee from being tracked as they roam between networks. When they rejoin the enterprise environment, they will re-obtain the original interface identifier.
2. **Generate R on OS reboot.** This option results in a new IPv6 address every time the computer is rebooted, even if the client receives the same IPv6 prefix from the broadcast daemon.
3. **Generate R on network interface change.** This option results in a new interface identifier I being generated whenever the client computer brings up the network interface. Since interfaces are brought up on boot and when connecting to a wireless network, a client will use a different I each time it joins a network broadcasting the same IPv6 prefix.
4. **Generate R when the user chooses.** This option results in a new interface identifier I being generated based on user involvement (e.g., the user regenerates R when transitioning between tasks). While we include this option for completeness, we do not suggest implementing this option.
5. **Generate R every x seconds.** In this option, the client generates a new IPv6 address every x seconds. This approach allows the current IPv6 privacy scheme as an option. Unlike the approaches discussed above, a new IPv6 address may be generated while network connections are open, causing these connections to be dropped. To reduce the number of dropped connections, the kernel can avoid deleting old IPv6 temporary addresses associated with active network connections. As long as the active network prefix is the same as that contained in the old temporary address, the temporary address can continue to be used. While we note that x does not need to stay fixed (i.e., a new x can also be chosen when the random identifier is updated), we currently see no additional benefit in randomly changing x. A default x of one day mirrors the current default with IPv6 privacy extensions.

We suggest the generation of a new random number (and hence interface identifier) whenever the network interface is brought up (option 3). This method generates IPv6 addresses as frequently as possible without interrupting open connections (since connections are terminated when the interface goes down).

3.2.2 Security Comments

The proposal is designed so that if an attacker has two distinct IPv6 addresses, it should be hard to answer the question "Did the same client use both IPv6 addresses?" To answer this question, the attacker must be able to determine that the same R value was used in the generation of both addresses (since two clients sharing the same R value is extremely unlikely).

In answering this question, we assume that the attacker has access to the generated interface identifier as well as the prefix. The security, therefore, depends on the difficulty of determining the random number provided to the pre-image resistant hash function – which is assumed to be hard for a sufficiently large R. An attacker attempting to track a client would need to keep trying random values R until finding one which generates multiple distinct and observed interface identifiers; therefore a birthday attack [5, 15] does not seem to help.

Because the interface identifier changes whenever the prefix changes, a client connecting through two networks with different prefixes will also connect with different interface identifiers, leaking no information in the IPv6 address.

One potential attack against our proposal involves a network administrator (as attacker) broadcasting target prefixes in an effort to detect what interface identifier would be used by the client on that network (e.g., the administrator broadcasts prefix `1:2:3::/64` to determine what IPv6 address the client would attempt to use on that network). The attack would be successful if R was not updated by the client before visiting the target network. One way to defend against this attack is to configure the client to generate a new R whenever enabling the network interface (option 3 in Section 3.2.1).

3.3 Limitations

[4]The formal cryptographic requirements of R and H are not the focus of the present paper.

The proposal for generating interface identifiers relies on an appropriate random number generator. If R can be guessed, an attacker can determine whether a client using the same R value generated multiple distinct IPv6 addresses using distinct prefixes. The proposal also depends on a pre-image resistant hash function (SHA-2 should suffice).

An attacker identifying two addresses used by a client through correlating between when one IPv6 address stops being used an the other starts being used is not protected against by our approach. We expect that as IPv6 privacy extensions are deployed, the volume of IPv6 address churn will make correlations more difficult.

3.4 Extensions to the core approach

As an extension to the core approach, it may be possible to use multiple IPv6 temporary addresses concurrently on a host. As an example, a new IPv6 address could be used for every application running on the client (e.g., web browsing would use one IPv6 address, while DNS queries would use another; an active SSH connection may use a third). While not directly privacy related, a server may also choose to use multiple IPv6 addresses (e.g., as a method for distributing firewalls [25]).

3.5 Implementation

For our prototype implementation, we used version 2.6.34 of the Linux kernel. We modified the currently implemented IPv6 privacy extensions. The modified kernel provides several sys-controls which can be read and written to by user-space programs, controlling the operation of IPv6 privacy extensions. These sys-controls are as follows:

Value	Meaning
0	Privacy extension disabled
1	Original privacy extension enabled but not used by default for new outgoing connections
2	Original privacy extension enabled and used by default for new outgoing connections
5	Proposed privacy extension enabled but not used by default for new outgoing connections
6	Proposed privacy extension enabled and used by default for new outgoing connections

Table 1: Possible values and their meaning for the `use_tempaddr` sys-control in the modified kernel.

1. **use_tempaddr:** controls whether or not to enable privacy extensions. Possible values are listed in Table 1.
2. **temp_valid_lft:** the maximum amount of time a temporary address is valid for. In the original approach, a new distinct temporary address would be created. In this proposal, the lifetime of an already-existing temporary address will be extended when router advertisements are received if both p and R are unchanged.
3. **temp_prefered_lft:** the maximum amount of time a temporary address will be the preferred address for the interface. Like with *temp_valid_lft*, the lifetime will be extended if both p and R are unchanged when a router advertisement is received.
4. **temp_random (new):** a 32 byte (256 bit) random value R used as input to the hash function. This sys-control is specific to our proposal.

We tested our prototype by switching between several IPv6 networks and verifying that the generated IPv6 addresses were different at each network. We did not notice any impact on activities such as web browsing and SSH. During the course of implementing the proposed IPv6 privacy extensions in Linux, we found several bugs which cause IPv6 privacy extensions to be disabled and/or all temporary address deleted. We are preparing to submit a patch to the Linux kernel which addresses these implementation deficiencies.

4. OTHER RELATED WORK

4.1 Cryptographically Generated Addresses

Cryptographically Generated Addresses [4] (CGA) were proposed to prevent stealing and/or spoofing of IPv6 addresses. CGAs define a method for securely associating a public key to an IPv6 address. The interface identifier of the IPv6 address is a cryptographic hash of the public key, which can later be verified by the recipient of the packet or message. Because CGAs tie a public key to an IPv6 address, even as hosts switch networks, they are uniquely identifiable through use of the public key. CGAs, like our proposal, use a cryptographic hash to generate the interface identifier, but the purpose of CGAs are contrary to ours and our proposal does not involve public keys.

4.2 Mobility Extensions

Mobility extensions [12] define ways in which a mobile host can move between networks while maintaining open connections, even if the networks use different link layer technologies (WiMAX, LTE, WiFi). This is accomplished by establishing a tunnel (usually with IPSec) to the *home network*. The mobile host is then reachable through the proxy home network. Route optimization [3] allows the correspondent node (server) to communicate directly with the mobile host, even though the mobile host's IPv6 address may continue to change. IPv6 Mobile extensions with route optimization are illustrated in Figure 3.

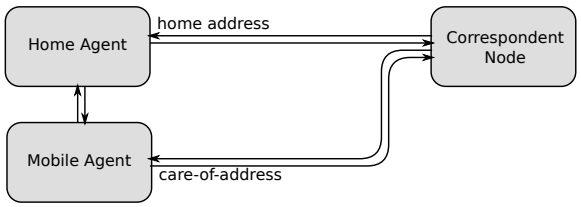

Figure 3: Communication with Mobile IPv6 agents

For mobile agents, implementing route optimization is mandatory [12]. The correspondent node receives both the home address and care-of-address, and can track the location of the mobile agent as it moves between IPv6 networks. Because the mobile agent is also tied to the fixed home address, enabling privacy extensions at the mobile agent does not prevent the correspondent node from tracking the mobile agent. To prevent tracking the mobile agent, both the home address and care-of-address must be changed at the same time. Otherwise, the correspondent node can tie the old address to the new address, and continue to track the mobile node.

Mobility extensions do not aim to address privacy concerns, but rather address connection persistence problems. The latter issue has also been solved independently in the cell phone industry at the link layer, where cell towers hand off connections to prevent active phone calls from being dropped when a user device switches towers.

4.3 Tracking at Other Layers

Guha et al. [10] demonstrate how to track users through DNS. Their analysis shows that dynamic DNS updates combined with geo-location [16] can provide a passive attacker with all the approximate locations visited by a victim. Geolocation in IPv6 may also reveal more accurate results compared to IPv4 due to address allocation recommendations [2].

Users with private IPv6 addresses may be tracked at the application layer through cookies [14] or browser characteristics [8]. Protecting privacy at all layers is clearly a difficult problem and beyond our scope, but we argue that in order to be have privacy at higher level protocols, underlying protocols must also be private.

While we do not address the problem of tracking users on the LAN specifically in this paper, we note that tracking users at the Ethernet level is possible due to clients broadcasting their MAC addresses [21]. Because MAC addresses in the Ethernet header are overwritten on a hop-by-hop basis, attackers outside the LAN do not obtain the MAC address of a client. This paper focuses on the network layer, where tracking can be performed across the Internet.

5. CONCLUSIONS AND FUTURE WORK

We have proposed a new way of generating the interface identifier used in temporary IPv6 addresses. The use of temporary addresses prevents tracking of clients as they move between IPv6 networks. Our approach does not use MAC addresses, which can be used to identify client hardware.

Our proposal has several benefits over the current IPv6 privacy extension scheme, including: 1) the ability to maintain a consistent IPv6 address over an extended period regardless of the lifetime specified by a router advertisement (as long as the prefixed being advertised does not change); 2) the ability to always use the same interface identifier while connected to a network broadcasting an unchanging prefix; 3) the ability to configure when a new interface identifier should be created (e.g., whenever the network interface is brought up); and 4) not being able to track a client through the use of a common interface identifier across networks broadcasting different IPv6 prefixes. We have implemented and tested the approach in Linux, and found that it generates new interface identifiers as designed while not impacting Internet activities.

6. REFERENCES

[1] J. Abley, P. Savola, and G. Neville-Neil. Deprecation of Type 0 Routing Headers in IPv6. RFC 5095 (Proposed Standard), Dec. 2007.

[2] APNIC. IPv6 address allocation and assignment policy. (viewed 11 Apr 2010). `http://www.apnic.net/policy/ipv6-address-policy`.

[3] J. Arkko, C. Vogt, and W. Haddad. Enhanced Route Optimization for Mobile IPv6. RFC 4866 (Proposed Standard), May 2007.

[4] T. Aura. Cryptographically generated addresses (CGA). In *Proc. 6th International Conference on Information Security*, pages 29–43.

[5] M. Bellare, O. Goldreich, and H. Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. In *Proc. 19th International Cryptology Conference on Advances in Cryptology*, pages 270–287, 1999.

[6] S. Bellovin, B. Cheswick, and A. Keromytis. Worm propagation strategies in an IPv6 internet. *;login - The USENIX Magazine*, 31(1):70–76, 2006.

[7] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Dec. 1998. Updated by RFCs 5095, 5722.

[8] P. Eckersley. A primer on information theory and privacy. Web Page, Jan 2010. `https://www.eff.org/deeplinks/2010/01/primer-information-theory-and-privacy`.

[9] Google. Google over IPv6. Web Page (viewed 26 Apr 2010). `http://www.google.com/intl/en/ipv6/`.

[10] S. Guha and P. Francis. Identity Trail: Covert Surveillance Using DNS. In *Proc. 2007 Privacy Enhancing Technologies*, pages 153–166.

[11] IEEE Standards Association. IEEE OUI and company_id assignments. Public Listing (viewed 28 Apr 2010). `http://standards.ieee.org/regauth/oui/index.shtml`.

[12] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[13] S. Krishnan. Handling of Overlapping IPv6 Fragments. RFC 5722 (Proposed Standard), Dec. 2009.

[14] D. Kristol and L. Montulli. HTTP State Management Mechanism. RFC 2965 (Proposed Standard), Oct. 2000.

[15] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[16] J. A. Muir and P. C. V. Oorschot. Internet geolocation: Evasion and counterevasion. *ACM Computing Surveys (CSUR)*, 42(1):1–23, 2009.

[17] T. Narten and R. Draves. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 3041 (Proposed Standard), Jan. 2001. Obsoleted by RFC 4941.

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

[19] Organisation for Economic Co-operation and Development (OECD). Internet Addressing: Measuring Deployment of IPv6. (viewed 10 Apr 2010). `http://bit.ly/aGLvAg`.

[20] V. Paxson, S. Staniford, and N. Weaver. How to 0wn the internet in your spare time. In *Proc. 11th Usenix Security Symposium*, pages 404–413, 2002.

[21] L. Peterson and B. Davie. *Computer networks: a systems approach*. Morgan Kaufmann, 2007.

[22] B. Preneel and P. C. van Oorschot. On the security of iterated message authentication codes. *IEEE Transactions on Information Theory*, 45(1):188–199, Jan 1999.

[23] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 2462 (Draft Standard), Dec. 1998. Obsoleted by RFC 4862.

[24] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), Sept. 2007.

[25] H. Zhao, C.-K. Chau, and S. M. Bellovin. ROFL: Routing as the firewall layer. In *Proc. 2008 New Security Paradigms Workshop*, pages 23–31.