

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 8, 2014

Y. Sheffer
Porticor
P. Hoffman
VPNC
December 5, 2013

A DHCP Extension To Provide Initial Random Material
draft-sheffer-dhc-initial-random-00

Abstract

Some network devices get little or no entropy from their underlying operating systems when they are first started. As a result, cryptographic applications started before there is sufficient entropy in the operating system's pool can be initialized into a state that can be exploited by an attacker. This document defines a DHCP extension that can provide the operating system of a network device with some initial randomness that can only be known by an attacker who is on the same network segment as the device and its DHCP server. The operating system can mix this random input into its random pool early in the boot procedure and thus have more entropy available when cryptographic applications start.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 8, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

Internet-Draft

Initial Randomness

December 2013

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Conventions used in this document	3
2.	The Initial Randomness Extension	3
2.1.	Extension Format	4
2.1.1.	DHCPv4	4
2.1.2.	DHCPv6	4
2.2.	Client and Server Behavior	5
3.	Alternatives and Applicability	5
3.1.	Alternatives to This Proposal	5
4.	Security Considerations	6
4.1.	RNG Properties	6
4.2.	Resistance Against Network Attacks	6
4.3.	Denial of Service	6
4.4.	Saving RNG State	6
4.5.	Protocol Signaling	7
5.	IANA Considerations	7
6.	Acknowledgements	7
7.	References	7
7.1.	Normative References	7
7.2.	Informative References	8
	Authors' Addresses	8

Internet-Draft

Initial Randomness

December 2013

1. Introduction

Security protocols require random bits for secure use. Ideally, such randomness should be provided to the operating system (OS) by a physical source of entropy, a "True Random Number Generator" (TRNG). Unfortunately, such sources of entropy are not generally available. All common OSs use random bits from other sources such as clock variations, event timing, network traffic, and so on, to create a pool of random bits available to applications. Further, some OSs do not retain randomness pools across reboots, leading to the need for fresh random bits each time a system is started.

Recently it was discovered [[Mining](#)] that a relatively large number of TLS and SSH public keys in the wild share a factor. The reason for that is presumed to be lack of much initial randomness when devices are first started and the keys are created. Security protocols, in particular SSH [[RFC4253](#)], that generate secret parameters before there is sufficient randomness end up with long-lived private keys that are vulnerable to guessing.

This document proposes that network devices use DHCP to request additional random material to be mixed in with the recipient's operating system's own random pool early in the boot process. The DHCP server can provide such material from its own "Pseudo-Random Number Generator" (PRNG). The requesting network device's OS can then mix the obtained material into its own PRNG.

The proposal does not mandate any particular use by the client, but [Section 3](#) describes various ways of applying it in different settings.

1.1. Conventions used in this document

All the DHCP related terms used in this document are to be interpreted as defined in the Dynamic Host Configuration Protocol v4 (DHCPv4) [[RFC2131](#)] and Dynamic Host Configuration Protocol v6

(DHCPv6) [\[RFC3315\]](#) specifications. DHCP refers to both DHCPv4 and DHCPv6 messages and entities throughout this document.

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

[2.](#) The Initial Randomness Extension

[2.1.](#) Extension Format

The extension consists of a variable-length Random Material field of arbitrary (random) octets. The field's length MUST be either 0, or between 16 and 64 octets. If the length is 0, it means that the sender is requesting randomness from the server and does not include any random material of its own.

The design of the extension allows the DHCP client to request the server to send random material, and for both client and server to each offer its own random material to the other party. If the receiving party cryptographically mixes the offered random material into its random pool, even if that material is completely predictable (such as all zeros), it will not have a negative effect on the pool.

[2.1.1.](#) DHCPv4

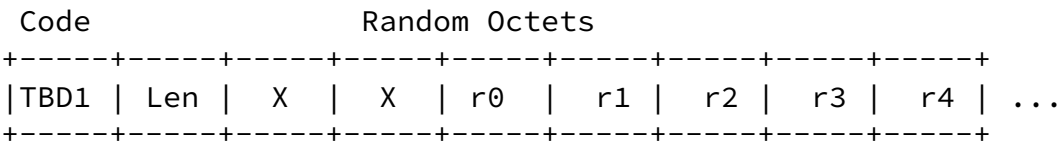


Figure 1: DHCPv4 Extension

The Code field is TBD by IANA.

The Len field is the length of the random octets, and excludes the Code field and the Length field itself.

The X octets are used for padding, they MUST be sent as zero and MUST be ignored by the receiver.

2.1.2. DHCPv6

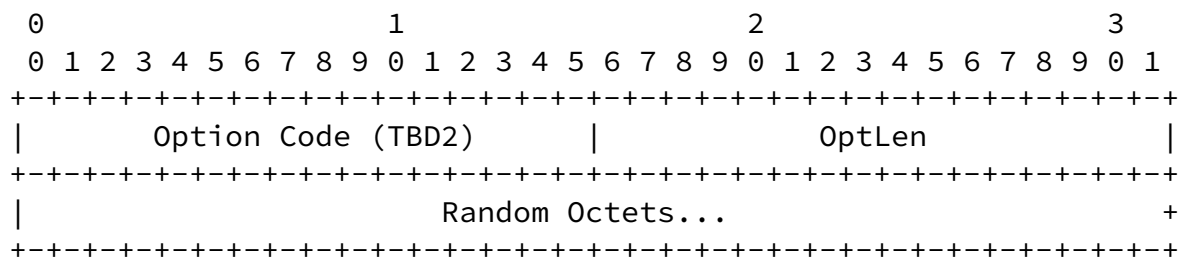


Figure 2: DHCPv6 Extension

The Option Code is TBD by IANA.

The OptLen field is the length of the random octets, and excludes the Option Code field and the OptLen field itself.

2.2. Client and Server Behavior

A client MAY use this extension in a DHCPREQUEST or a DHCPINFORM message. The client SHOULD include random material in its request if it believes it has seen more than n bits of physical entropy, where n is defined by local policy. If the client does not include any random material, it MAY still use this extension, with an empty Random Octets field, to signal that it requests random material from the server.

A server MAY use this extension in a DHCPACK message. A server using this extension that is unable to provide random material SHOULD include an empty Random Octets field.

If a server is also a DHCP client to some other DHCP server on a different interface, it is RECOMMENDED that the server obtain random material from its upstream server before it serves randomness to its clients.

3. Alternatives and Applicability

This section lists several viable alternatives to the current proposal. Then we discuss several use cases, and for each one, whether and how this extension can be used.

[3.1.](#) Alternatives to This Proposal

In general, there are three good alternatives to the current solution:

1. Use of an on-board hardware based TRNG, such as the Intel RdRand instruction [[IntelArch](#)].
2. Pre-provisioning, with the manufacturer of the device creating a unique and secret value on each new device, and seeding the RNG with this value.
3. In virtual systems, use of randomness obtained from the host.

It is noted that option #2 can be implemented by the current extension, when used in a secure network.

While all alternatives are technically feasible today, there are still many platforms that do not use any of them.

As discussed in [Section 4.1](#), a good random number generator must be able to mix randomness from multiple sources, in a manner than accumulates entropy even if some of the sources are highly non-random and/or observable to an attacker. So as a best practice,

implementors that have multiple sources available to them should mix them together, to avoid known and unknown issues with any of the sources.

[4.](#) Security Considerations

In addition to those listed here, please refer to [[RFC4086](#)] for further considerations.

[4.1.](#) RNG Properties

This document assumes a modern crypto-grade RNG on both client and server, which should have at least the following properties:

- o The output stream cannot be distinguished from a truly random one without knowing the secret state.
- o The RNG can be fed by any external material, even material known by an attacker, at any time. Such material will normally increase, and will never decrease the generator's entropy.
- o The generator's internal state can never be revealed to an attacker, even if the attacker can feed any amount of known data into the RNG.

[4.2.](#) Resistance Against Network Attacks

The current extension is not resistant to either passive or active attackers. Specifically, if an attacker knows the complete state of the RNG, and is able to listen in on the local network, they will be able to compute the state of the RNG after it has been fed with the random material.

[4.3.](#) Denial of Service

An active attacker can use this extension to overload the server's CPU and possibly to exhaust the server's entropy pool. To avoid such attacks, the server SHOULD rate-limit responses to this particular extension.

[4.4.](#) Saving RNG State

The current extension is targeted at the initial bootstrap of a host or device. Different devices choose whether or not to save random state across reboots based on their particular design considerations. In short, saving state causes the booting machine to have some random material already; saving state also allows someone who can view the quiescent state (such as reading from the drive) to know the initial state of the OS's random pool.

[4.5.](#) Protocol Signaling

This protocol does not allow the client or the server to signal what type of random material is being requested or offered. Specifically, they are unable to signal whether randomness is being backed by measured physical entropy. This was done to simplify the protocol, and also because the protocol is vulnerable to active attackers that may be present on the network and could alter any such signals.

5. IANA Considerations

This document defines the DHCP Initial Randomness option which requires assignment of DHCPv4 option code TBD1 assigned from the "Bootp and DHCP options" registry (<<http://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xml>>), as specified in [RFC2939].

This document defines the DHCP Initial Randomness option which requires assignment of DHCPv6 option code TBD2 assigned from the "DHCP option Codes" registry (<<http://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xml>>).

6. Acknowledgements

This proposal was inspired by a discussion on the Cryptography mailing list, and we would like to thank John Gilmore for triggering the idea of "BOOTP for RNG".

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", [RFC 2131](#), March 1997.
- [RFC2939] Droms, R., "Procedures and IANA Guidelines for Definition of New DHCP Options and Message Types", [BCP 43](#), [RFC 2939](#), September 2000.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", [RFC 3315](#), July 2003.

7.2. Informative References

- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", [BCP 106](#), [RFC 4086](#), June 2005.
- [RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", [RFC 4253](#), January 2006.
- [Mining] Heninger, N., Durumeric, Z., Wustrow, E., and J. Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices", Proceedings of the 21st USENIX Security Symposium , August 2012, <<https://factorable.net/>>.
- [IntelArch] "Intel 64 and IA-32 Architectures Software Developer's Manual Combined Volumes 2A, 2B, and 2C: Instruction Set Reference, A-Z", December 2012.

Authors' Addresses

Yaron Sheffer
Porticor
29 HaHarash St.
Hod HaSharon 4501303
Israel

Email: yarolf.ietf@gmail.com

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060
US

Email: paul.hoffman@vpnc.org