DNSOP Working Group
Internet-Draft
Updates: 7873 (if approved)
Intended status: Standards Track
Expires: May 7, 2020

O. Sury
Internet Systems Consortium
W. Toorop
NLnet Labs
D. Eastlake 3rd
Futurewei Technologies
M. Andrews
Internet Systems Consortium

November 4, 2019

Interoperable Domain Name System (DNS) Server Cookies
draft-ietf-dnsop-server-cookies-01

Abstract

DNS cookies, as specified in RFC 7873, are a lightweight DNS transaction security mechanism that provides limited protection to DNS servers and clients against a variety of denial-of-service and amplification, forgery, or cache poisoning attacks by off-path attackers.

This document provides precise directions for creating Server Cookies so that an anycast server set including diverse implementations will interoperate with standard clients.

This document updates [RFC7873]

Status of This Memo

Table of Contents

1. Introduction

DNS cookies, as specified in [RFC7873], are a lightweight DNS transaction security mechanism that provides limited protection to DNS servers and clients against a variety of denial-of-service and amplification, forgery, or cache poisoning attacks by off-path attackers. This document specifies a means of producing

interoperable strong cookies so that an anycast server set including diverse implementations can be easily configured to interoperate with standard clients.

The threats considered for DNS Cookies and the properties of the DNS Security features other than DNS Cookies are discussed in [RFC7873].

In [RFC7873] in Section 6 it is "RECOMMENDED for simplicity that the same Server Secret be used by each DNS server in a set of anycast servers." However, how precisely a Server Cookie is calculated from this Server Secret, is left to the implementation.

This guidance has led to a gallimaufry of DNS Cookie implementations, calculating the Server Cookie in different ways. As a result, DNS Cookies are impractical to deploy on multi-vendor anycast networks, because even when all DNS Software share the same secret, as RECOMMENDED in Section 6 of [RFC7873], the Server Cookie constructed by one implementation cannot generally be validated by another.

There is no need for DNS client (resolver) Cookies to be interoperable across different implementations. Each client need only be able to recognize its own cookies. However, this document does contain recommendations for constructing Client Cookies in a Client protecting fashion.

1.1. Contents of this document

Section Section 2 summarises the changes to [RFC7873].

In Section Section 3 suggestions for constructing a Client Cookie are given.

In Section Section 4 instructions for constructing a Server Cookie are given.

In Section Section 5 instructions on updating Server Secrets are given.

In Section Section 6 the different hash functions usable for DNS Cookie construction are listed. [FNV] and HMAC-SHA-256-64 [RFC6234] are deprecated and [SipHash-2.4] is introduced as a REQUIRED hash function for server side DNS Cookie implementations.

IANA considerations are in Section 7.

Acknowledgements are in Appendix A.

Test vectors are in Appendix B.

1.2. Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "*NOT RECOMMENDED*", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

- o "IP Address" is used herein as a length independent term covering both IPv4 and IPv6 addresses.

2. Changes to [RFC7873]

In its Appendices A.1 and B.1, [RFC7873] provides example "simple" algorithms for computing Client and Server Cookies, respectively. These algorithms MUST NOT be used as the resulting cookies are too weak when evaluated against modern security standards.

In its Appendix B.2, [RFC7873] provides an example "more complex" server algorithm. This algorithm is replaced by the interoperable specification in Section 4 of this document, which MUST be used by Server Cookie implementations.

This document has suggestions on Client Cookie construction in Section 3. The previous example in Appendix A.2 of [RFC7873] is NOT RECOMMENDED.

3. Constructing a Client Cookie

The Client Cookie is a cryptographic nonce and should be treated as such. For simplicity, it can be calculated from Server IP Address, and a Client Secret known only to the Client that is changed whenever an IP address previously used by the Client is no longer available. The Client Cookie SHOULD have at least 64-bits of entropy.

Except for when the Client IP address changes, there is no need to change the Client Secret often if a secure pseudorandom function (like [SipHash-2.4]) is used. It is reasonable to change the Client secret then only if it has been compromised or after a relatively long period of time such as no longer than a year.

It is RECOMMENDED but not required that the following pseudorandom function be used to construct the Client Cookie:

```
Client-Cookie = MAC_Algorithm(
    Server IP Address, Client Secret )
```

Previously, the recommended algorithm to compute the Client Cookie included Client IP Address as an input to the MAC_Algorithm. However, when implementing the DNS Cookies, several DNS vendors found impractical to include the Client IP as the Client Cookie is typically computed before the Client IP address is known. Therefore, the requirement to put Client IP address as input was removed.

However, for privacy reasons, in order to prevent tracking of devices across links and to not circumvent IPv6 Privacy Extensions [RFC4941], Clients MUST NOT re-use a Client or Server Cookie after the Client IP address has changed.

The Client IP address is available on the UDP socket when it receives the Server Cookie and should be registered alongside the Server Cookie. In subsequent queries to the Server with that Server Cookie, the socket MUST be bound to the Client IP address that was also used (and registered) when it received the Server Cookie. Failure to bind must result in a new Client Cookie, which, for the method described in this section means a new Client Secret.

4. Constructing a Server Cookie

The Server Cookie is effectively a Message Authentication Code (MAC) and should be treated as such. The Server Cookie is calculated from the Client Cookie, a series of Sub-Fields specified below, the Client IP address, and a Server Secret known only to the servers responding on the same address in an anycast set.

Changing the Server Secret regularly is RECOMMENDED but, when a secure pseudorandom function is used, it need not be changed too frequent. For example once a month would be adequate. See Section 5 on operator and implementation guidelines for updating a Server Secret.

The 128-bit Server Cookie consists of Sub-Fields: a 1 octet Version Sub-Field, a 3 octet Reserved Sub-Field, a 4 octet Timestamp Sub-Field and an 8 octet Hash Sub-Field.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |                   Reserved                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Hash                              |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.1. The Version Sub-Field

The Version Sub-Field prescribes the structure and Hash calculation formula. This document defines Version 1 to be the structure and way to calculate the Hash Sub-Field as defined in this Section.

4.2. The Reserved Sub-Field

The value of the Reserved Sub-Field is reserved for future versions of Server Side Cookie construction. On construction it SHOULD be set to zero octets. On Server Cookie verification the server MUST NOT enforce those fields to be zero and the Hash should be computed with the received value as described in Section 4.4.

4.3. The Timestamp Sub-Field

The Timestamp value prevents Replay Attacks and MUST be checked by the server to be within a defined period of time. The DNS Server SHOULD allow Cookies within 1 hour period in the past and 5 minutes into the future to allow operation of low volume clients and some limited time skew between the DNS servers in the anycast.

The Timestamp value specifies a date and time in the form of a 32-bit unsigned number of seconds elapsed since 1 January 1970 00:00:00 UTC, ignoring leap seconds, in network byte order. All comparisons involving these fields MUST use "Serial number arithmetic", as defined in [RFC1982]

The DNS Server SHOULD generate a new Server Cookie at least if the received Server Cookie from the Client is more than half an hour old.

4.4. The Hash Sub-Field

It's important that all the DNS servers use the same algorithm for computing the Server Cookie. This document defines the Version 1 of the Server Side algorithm to be:

```
Hash = SipHash2.4(
    Client Cookie | Version | Reserved | Timestamp | Client-IP,
    Server Secret )
```

where "|" indicates concatenation.

Notice that Client-IP is used for hash generation even though it's not included in the cookie value itself. Client-IP can be either 4 bytes for IPv4 or 16 bytes for IPv6.

The Server Secret MUST be configurable to make sure that servers in an anycast network return consistent results.

5. Updating the Server Secret

All servers in an anycast group must be able to verify the Server Cookies constructed by all other servers in that anycast set at all times. Therefore it is vital that the Server Secret is shared among all servers before it us used to generate Server Cookies.

Also, to maximize maintaining established relationships between clients and servers, an old Server Secret should be valid for verification purposes for a specific period.

To facilitate this, deployment of a new Server Secret MUST be done in three stages:

Stage 1

The new Server Secret is deployed on all the servers in an anycast set by the operator.

Each server learns the new Server Secret, but keeps using the previous Server Secret to generate Server Cookies.

Server Cookies constructed with the both the new Server Secret and with the previous Server Secret are considered valid when verifying.

After stage 1 completed, all the servers in the anycast set have learned the new Server Secret, and can verify Server Cookies constructed with it, but keep generating Server Cookies with the old Server Secret.

Stage 2

This stage is initiated by the operator after the Server Cookie is present on all members in the anycast set.

When entering Stage 2, servers start generating Server Cookies with the new Server Secret. The previous Server Secret is not yet removed/forgotten about.

Server Cookies constructed with the both the new Server Secret and with the previous Server Secret are considered valid when verifying.

Stage 3

This stage is initiated by the operator when it can be assumed that most clients have learned the new Server Secret.

With this stage, the previous Server Secret can be removed and MUST NOT be used anymore for verifying.

We RECOMMEND the operator to wait at least a period to be the longest TTL in the zones served by the server plus half an hour after it initiated Stage 2, before initiating Stage 3.

The operator SHOULD wait at least longer than the period clients are allowed to use the same Server Cookie, which SHOULD be half an hour, see Section 4.3.

6. Cookie Algorithms

[SipHash-2.4] is a pseudorandom function suitable as Message Authentication Code. This document REQUIRES compliant DNS Server to use SipHash-2.4 as a mandatory and default algorithm for DNS Cookies to ensure interoperability between the DNS Implementations.

The construction method and pseudorandom function used in calculating and verifying the Server Cookies are determined by the initial version byte and by the length of the Server Cookie. Additional pseudorandom or construction algorithms for Server Cookies might be added in the future.

7. IANA Considerations

IANA is requested to create a registry on the "Domain Name System (DNS) Parameters" IANA web page as follows:

Registry Name: DNS Server Cookie Methods
Assignment Policy: Expert Review
Reference: [this document], [RFC7873]
Note: Server Cookie method (construction and pseudorandom algorithm) are determined by the Version in the first byte of the Cookie and by the Cookie size. Server Cookie size is limited to the inclusive range of 8 to 32 bytes.

Implementation recommendations for Cookie Algorithms [DNSCOOKIE-IANA]:

Version	Size	Method
0	8-32	reserved
1	8-15	unassiged
1	16	SipHash-2.4 [this document] Section 4
1	17-32	unassigned
2-239	8-32	unassigned
240-254	8-32	private use
255	8-32	reserved

8. References

8.1. Normative References

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7873] Eastlake 3rd, D. and M. Andrews, "Domain Name System (DNS) Cookies", RFC 7873, DOI 10.17487/RFC7873, May 2016, <https://www.rfc-editor.org/info/rfc7873>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[SipHash-2.4] Aumasson, J. and D. Bernstein, "SipHash: a fast short-input PRF", 2012, <https://131002.net/siphash/>.

8.2. Informative References

[FNV] Fowler, G., Noll, L., Vo, K., Eastlake, D., and T. Hansen, "The FNV Non-Cryptographic Hash Algorithm", <https://datatracker.ietf.org/doc/draft-eastlake-fnv>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

Appendix A. Acknowledgements

Thanks to Witold Krecicki and Pieter Lexis for valuable input, suggestions and text and above all for implementing a prototype of an interoperable DNS Cookie in Bind9, Knot and PowerDNS during the hackathon of IETF104 in Prague. Thanks for valuable input and suggestions go to Ralph Dolmans, Bob Harold, Daniel Salzman, Martin Hoffmann, Mukund Sivaraman, Petr Spacek, Loganaden Velvindron, Bob Harold and Philip Homburg

Appendix B. Test vectors

B.1. Learning a new Server Cookie

A resolver (client) sending from IPv4 address 198.51.100.100, sends a query for "example.com" to an authoritative server listening on 192.0.2.53 from which it has not yet learned the server cookie.

The DNS requests and replies shown in this Appendix, are in a "dig" like format. The content of the DNS COOKIE Option is shown in hexadecimal format after "; COOKIE:".

```
;; Sending:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57406
;; flags:; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2464c4abcf10c957
;; QUESTION SECTION:
;example.com.                IN      A

;; QUERY SIZE: 52
```

The authoritative nameserver (server) is configured with the following secret: e5e973e5a6b2a43f48e7dc849e37bfcf (as hex data).

It receives the query at Wed Jun 5 10:53:05 UTC 2019.

The content of the DNS COOKIE Option that the server will return is shown below in hexadecimal format after "; COOKIE:"

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57406
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2464c4abcf10c957010000005cf79f111f8130c3eee29480 (good)
;; QUESTION SECTION:
;example.com.                IN      A

;; ANSWER SECTION:
example.com.         86400   IN      A       192.0.2.34

;; Query time: 6 msec
;; SERVER: 192.0.2.53#53(192.0.2.53)
;; WHEN: Wed Jun  5 10:53:05 UTC 2019
;; MSD SIZE  rcvd: 84
```

B.2. The same client learning a renewed (fresh) Server Cookie

40 minutes later, the same resolver (client) queries the same server for for "example.org" :

```
;; Sending:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50939
;; flags:; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2464c4abcf10c957010000005cf79f111f8130c3eee29480
;; QUESTION SECTION:
;example.org.                IN      A

;; QUERY SIZE: 52
```

The authoritative nameserver (server) now generates a new Server Cookie. The server SHOULD do this because it can see the Server Cookie send by the client is older than half an hour Section 4.3, but it is also fine for a server to generate a new Server Cookie sooner, or even for every answer.

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50939
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 2464c4abcf10c957010000005cf7a871d4a564a1442aca77 (good)
;; QUESTION SECTION:
;example.org.                IN      A

;; ANSWER SECTION:
example.org.         86400   IN      A       192.0.2.34

;; Query time: 6 msec
;; SERVER: 192.0.2.53#53(192.0.2.53)
;; WHEN: Wed Jun  5 11:33:05 UTC 2019
;; MSD SIZE  rcvd: 84
```

B.3. Another client learning a renewed Server Cookie

Another resolver (client) with IPv4 address 203.0.113.203 sends a request to the same server with a valid Server Cookie that it learned before (at Wed Jun 5 09:46:25 UTC 2019). Note that the Server Cookie has Reserved bytes set, but is still valid with the configured secret; the Hash part is calculated taking along the Reserved bytes.

```
;; Sending:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34736
;; flags:; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: fc93fc62807ddb8601abcdef5cf78f71a314227b6679ebf5
;; QUESTION SECTION:
;example.com.                IN      A

;; QUERY SIZE: 52
```

The authoritative nameserver (server) replies with a freshly generated Server Cookie for this client conformant with this specification; so with the Reserved bits set to zero.

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34736
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: fc93fc62807ddb86010000005cf7a9acf73a7810aca2381e (good)
;; QUESTION SECTION:
;example.com.                IN      A

;; ANSWER SECTION:
example.com.         86400   IN      A       192.0.2.34

;; Query time: 6 msec
;; SERVER: 192.0.2.53#53(192.0.2.53)
;; WHEN: Wed Jun  5 11:38:20 UTC 2019
;; MSD SIZE  rcvd: 84
```

B.4. IPv6 query with rolled over secret

The query below is from a client with IPv6 address 2001:db8:220:1:59de:d0f4:8769:82b8 to a server with IPv6 address 2001:db8:8f::53. The client has learned a valid Server Cookie before when the Server had secret: dd3bdf9344b678b185a6f5cb60fca715. The server now uses a new secret, but it can still validate the Server Cookie provided by the client as the old secret has not expired yet.

```
;; Sending:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6774
;; flags:; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 22681ab97d52c298010000005cf7c57926556bd0934c72f8
;; QUESTION SECTION:
;example.net.                IN      A

;; QUERY SIZE: 52
```

The authoritative nameserver (server) replies with a freshly generated server cookie for this client with its new secret: 445536bcd2513298075a5d379663c962

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6774
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 22681ab97d52c298010000005cf7c609a6bb79d16625507a (good)
;; QUESTION SECTION:
;example.net.                IN      A

;; ANSWER SECTION:
example.net.         86400   IN      A       192.0.2.34

;; Query time: 6 msec
;; SERVER: 2001:db8:8f::53#53(2001:db8:8f::53)
;; WHEN: Wed Jun  5 13:36:57 UTC 2019
;; MSD SIZE  rcvd: 84
```

Authors' Addresses

Ondrej Sury
Internet Systems Consortium
CZ

Email: ondrej@isc.org

Willem Toorop
NLnet Labs
Science Park 400
Amsterdam 1098 XH
Netherlands

Email: willem@nlnetlabs.nl

Donald E. Eastlake 3rd
Futurewei Technologies
1424 Pro Shop Court
Davenport FL 33896
USA

Phone: +1-508-333-2270
Email: d3e3e3@gmail.com

Mark Andrews
Internet Systems Consortium
950 Charter Street
Redwood City CA 94063
USA

Email: marka@isc.org