INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: July 21, 2014

Donald Eastlake
Huawei

January 22, 2014

Domain Name System (DNS) Cookies

<draft-eastlake-dnsext-cookies-04.txt>

Abstract

DNS cookies are a lightweight DNS transaction security mechanism designed for incremental deployment. They provide limited protection to DNS servers and resolvers against a variety of increasingly common denial-of-service and amplification/forgery or cache poisoning attacks by off-path attackers. DNS Cookies are tolerant of NAT, NAT-PT, and anycast.

Status of This Document

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Distribution of this document is unlimited. Comments should be sent to the author or the DNSEXT mailing list <dnsext@ietf.org>.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html. The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Table of Contents

1. Introduction

As with many core Internet protocols, the Domain Name System (DNS) was originally designed at a time when the Internet had only a small pool of trusted users. As the Internet has grown exponentially to a global information utility, the DNS has increasingly been subject to abuse.

This document describes DNS cookies, a lightweight DNS transaction security mechanism specified as an OPT [RFC6891] option. This mechanism provides limited protection to DNS servers and resolvers against a variety of increasingly common abuses by off-path attackers.

The DNS cookies mechanism has a default mode that supports incremental deployment. If only one party to a DNS transaction supports the mechanism, it does not provide a benefit or significantly interfere, but, if both support it, the additional security provided is automatically available.

The DNS cookies mechanism is compatible with and can be used in conjunction with other DNS transaction forgery resistance measures such as those in [RFC5452].

The DNS cookies mechanism is designed to work in the presence of NAT and NAT-PT boxes and guidance is provided herein on supporting the DNS cookies mechanism in anycast servers.

1.1 Contents of This Document

In Section 2, we discuss the threats against which the DNS cookie mechanism provides some protection.

Section 3 describes existing DNS security mechanisms and why they are not adequate substitutes for DNS cookies.

Section 4 describes the COOKIE OPT option and Section 5 provides a protocol description including suggestions for calculating Resolver and Server Cookies.

Section 6 gives further details on the processing of COOKIE OPT options by resolvers and server and policies for such processing.

Section 7 discusses some NAT and anycast related DNS Cookies design considerations.

Section 8 discusses incremental deployment considerations.

Sections 9 and 10 describe IANA and Security Considerations.

1.2 Definitions

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].

An "off-path attacker", for a particular DNS resolver and server, is defined as an attacker who cannot observe the plain text of DNS requests and responses between that resolver and server.

"Soft state" indicates information learned or derived by a host which may be discarded when indicated by the policies of that host but can be later re-instantiated if needed. For example, it could be discarded after a period of time or when storage for caching such data becomes full. If operations requiring that soft state continue after it has been discarded, it will be automatically re-generated, albeit at some cost.

"Silently discarded" indicates that there are no DNS protocol message consequences; however, it is RECOMMENDED that appropriate network management facilities be included in implementations, such as a counter of the occurrences of each type of such events.

The term "IP address" is used herein in a length independent manner and refers to both IPv4 and IPv6.

2. Threats Considered

DNS cookies are intended to provide significant but limited protection against certain attacks by off-path attackers as described below. These attacks include denial-of-service, cache poisoning and answer forgery.

2.1 Denial-of-Service Attacks

The typical form of the denial-of-service attacks considered herein is to send DNS requests with forged source IP addresses to a server. The intent can be to attack that server or some other selected host as described below.

2.1.1 DNS Amplification Attacks

A request with a forged IP address generally causes a response to be sent to that forged IP address. Thus the forging of many such requests with a particular source IP address can result in enough traffic being sent to the forged IP address to interfere with service to the host at the IP address. Furthermore, it is generally easy in the DNS to create short requests that produce much longer responses, thus amplifying the attack.

The DNS Cookies mechanism can severely limit the traffic amplification obtained by attackers off path for the server and the attacked host. Enforced DNS cookies would make it hard for an off path attacker to cause any more than rate-limited short error responses to be sent to a forged IP address so the attack would be reduced rather than amplified. DNS cookies make it more effective to implement a rate limiting scheme for bad DNS cookie error responses from the server. Such a scheme would further restrict selected host denial-of-service traffic from that server.

2.1.2 DNS Server Denial-of-Service

DNS requests that are accepted cause work on the part of DNS servers. This is particularly true for recursive servers that may issue one or more requests and process the responses thereto, in order to determine their response to the initial request. And the situation can be even worse for recursive servers implementing DNSSEC ([RFC4033] [RFC4034] [RFC4035]) because they may be induced to perform burdensome public key cryptographic computations in attempts

to verify the authenticity of data they retrieve in trying to answer

the request.

The computational or communications burden caused by such requests may not dependent on a forged IP source address, but the use of such addresses makes

+ the source of the requests causing the denial-of-service attack harder to find and
+ restriction of the IP addresses from which such requests should be honored hard or impossible to specify or verify.

Use of DNS cookies should enables a server to reject forged queries from an off path attacker with relative ease and before any recursive queries or public key cryptographic operations are performed.

2.2 Cache Poisoning and Answer Forgery Attacks

The form of the cache poisoning attacks considered is to send forged replies to a resolver. Modern network speeds for well-connected hosts are such that, by forging replies from the IP addresses of heavily used DNS servers for popular names to a heavily used resolver, there can be an unacceptably high probability of randomly coming up with a reply that will be accepted and cause false DNS information to be cached by that resolver (the Dan Kaminsky attack). This can be used to facilitate phishing attacks and other diversion of legitimate traffic to a compromised or malicious host such as a web server.

With the use of DNS cookies, a resolver can generally reject such forged replies.

3. Comments on Existing DNS Security

Two forms of security have been added to DNS, data security and message/transaction security.

3.1 Existing DNS Data Security

DNS data security is one part of DNSSEC and is described in [RFC4033], [RFC4034], and [RFC4035] and updates thereto. It provides data origin authentication and authenticated denial of existence. DNSSEC is being deployed and can provide strong protection against forged data; however, it has the unintended effect of making some denial-of-service attacks worse because of the cryptographic computational load it can require and the increased size in DNS packets that it tends to produce.

3.2 DNS Message/Transaction Security

The second form of security that has been added to DNS provides "transaction" security through TSIG [RFC2845] or SIG(0) [RFC2931]. TSIG could provide strong protection against the attacks for which the DNS Cookies mechanism provide weak protection; however, TSIG is non-trivial to deploy in the general Internet because of the burden it imposes of pre-agreement and key distribution between resolver-server pairs, the burden of server side key state, and because it requires time synchronization between resolver and server.

TKEY [RFC2930] can solve the problem of key distribution for TSIG but some modes of TKEY impose a substantial cryptographic computation loads and can be dependent on the deployment of DNS data security (see Section 3.1).

SIG(0) [RFC2931] provides less denial of service protection than TSIG or, in one way, even DNS cookies, because it does not authenticate requests, only complete transactions. In any case, it also depends on the deployment of DNS data security and requires computationally burdensome public key cryptographic operations.

3.3 Conclusions on Existing DNS Security

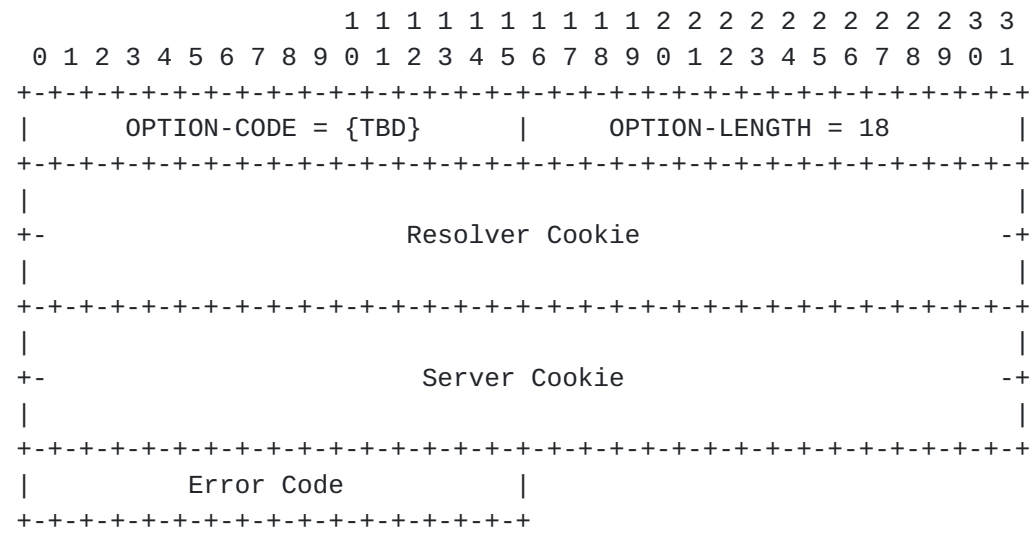
The existing DNS security mechanisms do not provide the services provided by the DNS Cookies mechanism: lightweight message authentication of DNS requests and responses with no requirement for

pre-configuration or server side state.

4. The COOKIE OPT Option

COOKIE is an OPT RR [RFC6891] option that can be included no more than once in the RDATA portion of an OPT RR in DNS requests and responses.

The option is encoded into 22 bytes as shown below.



The 64-bit Resolver and Server Cookies are stored in little endian order and are determined as described below.

4.1 Resolver Cookie

The Resolver Cookie SHOULD be a pseudo-random function of the server IP address and a secret quantity known only to the resolver. This resolver secret SHOULD have at least 64 bits of entropy [RFC4086bis] and be changed periodically (see Section 6.3). The selection of the pseudo-random function is a private matter to the resolver as only the resolver needs to recognize its own DNS cookies. An example method is the FNV-64 [FNV] of the server IP address and the resolver secret. That is

```
Resolver Cookie = FNV-64 ( Resolver Secret | Server IP Address )
```

where "|" indicates concatenation.

A resolver MUST NOT use the same Resolver Cookie value for queries to all servers.

4.2 Server Cookie

The Server Cookie SHOULD be a pseudo-random function of the request source IP address, the request Resolver Cookie, and a secret quantity known only to the server. (See Section 7 for a discussion of why the Resolver Cookie is used as input to the Server Cookie but the Server Cookie is not used as an input to the Resolver Cookie.) This server secret SHOULD have 64 bits of entropy [RFC4086bis] and be changed periodically (see Section 6.3). The selection of the pseudo-random function is a private matter to the server as only the server needs to recognize its own DNS cookies. An example method is the FNV-64 [FNV] of the request IP address, the Resolver Cookie, and the server secret. That is

```
Server Cookie =
   FNV-64 ( Server Secret | Request IP Address | Resolver Cookie )
```

where "|" represents concatenation.

A server MUST NOT use the same Server Cookie value for responses to all resolvers.

4.3 Error Code

The Error Code field MUST have one of the values listed below.

Requests have a COOKIE OPT Error Code equal to one of the following two values:

- Zero, if the resolver believes the Server Cookie field is correct, or
- CKPING (Cookie PING), if the resolver does not know the correct value for the Server Cookie field.

(In all cases, the RCODE in a DNS request header is zero.)

Replies have a COOKIE OPT with an Error Code equal to one of the following five values:

- Zero, if the request they respond to had one COOKIE OPT with a correct Server Cookie.
- NOCOOKIE, in which case the DNS reply header RCODE field is Refused.
- BADCOOKIE, in which case the DNS reply header RCODE field is

Refused.

MANYCOOKIE, in which case the DNS reply header RCODE field is FormErr.

CKPINGR (Cookie PING Response), which case the DNS reply RCODE field might be any value (see Section 5.2).

For more information on errors in replies see Section 6.

For further discussion of the Resolver Cookie field, see Section 5.1.
For further discussion of the Server Cookie field see Section 5.2.

5. DNS Cookies Protocol Description

The sections provide a general discussion of using DNS Cookies in the DNS Protocol. More details are provided in Section 6.

5.1 Originating Requests

A DNS resolver that implements DNS cookies includes a DNS Cookie option in every DNS request it sends unless DNS cookies are disabled in that resolver. The DNS Cookie in a request includes a Resolver Cookie as discussed in Section 4.1, a Server Cookie cached as soft state associated with that server IP address from a previous DNS response, and a zero Error Code field.

If the resolver has no cached Server Cookie for the server, then it sets the Server Cookie field to any value and sets the Error Code field to CKPING (Cookie Ping); this is the only case in which the Error Code field in a COOKIE OPT in a request is non-zero.

5.2 Responding to Requests

The Server Cookie, when it occurs in a COOKIE OPT option in a request, is intended to weakly assure the server that the request came from a resolver at the source IP address used because the Server Cookie value is the value that server would send to that resolver in a response.

A DNS server that implements DNS cookies and for which DNS cookies are not disabled always includes a DNS cookie in the response to a DNS request that includes such a cookie. If the request did not include a DNS cookie, inclusion of a DNS cookies in the response depends on the server mode for that resolved (see Section 6.2). In the DNS Cookie that the server includes, the Resolver Cookie field is copied from that field in the request. If there was no cookie in the request, it may be set to any value. The Server Cookie field is set as discussed in Section 4.2 and the Error Code field is set as specified in Section 6.

5.3 Processing Responses

The Resolver Cookie, when it occurs in a COOKIE OPT option in a DNS reply, is intended to weakly assure the resolver that the reply came from a server at the source IP address use in the response packet

because the Resolver Cookie value is the value that resolver would

send to that server in a request.

A DNS resolver that implements DNS cookies and for which DNS cookies are not disabled examines response for DNS cookies and will discard the response if it contains an incorrect Resolver Cookie or has multiple cookies. If the COOKIE OPT option Resolver Cookie is correct and the Error Code field is not NOCOOKIE, MANYCOOKIES, it caches the Server Cookie provided even if the response is an error response. The rest of the response is then processed normally.

6. DNS Cookie Policies and Implementation

Obviously, DNS resolvers that do not implement DNS cookies do not include them in requests and ignore them in replies and DNS servers that do not implement DNS cookies ignore them in requests and do not include the in replies.

DNS resolvers and servers that implement DNS cookies will adopt one of various policies regarding cookies. These policies SHOULD be logically settable on a per server IP address basis in resolvers and on a per resolver (IP address, Resolver Cookie) pair in servers. Thus a resolver can have different policies for different servers, based on the server IP address. And a server can have different policies for different resolvers, based on the resolver IP address and Resolver Cookie. Of course, the actual implementation of the configuration of these policies may be for blocks or classes of values or use sparse array techniques or the like.

The policy in each case is either "Disabled", "Enabled", or "Enforced" as described below.

6.1 Resolver Policies and Implementation

A resolver will logically have one of the following three modes of operation or "policies" for each DNS server as distinguished by server IP Address.

Disabled:
- Never include a COOKIE OPT option in requests.
- Ignore COOKIE OPT options in replies.

Enabled:
- Always include a COOKIE OPT option in requests. If a cached Server Cookie for the server is not available, the Server Cookie field can be set to any value and the COOKIE OPT Error Code field is set to CKPING (Cookie Ping); otherwise, the Error Code field is set to zero.
- Normally process replies without a COOKIE OPT option.
- Silently ignore replies with more than one COOKIE OPT option.
- Silently ignore replies with one COOKIE OPT option if it has an incorrect Resolver Cookie value.
- On receipt of a reply with one COOKIE OPT option carrying the correct Resolver Cookie value (even if it is a DNS error response), the DNS client performs normal response processing, including caching the received Server Cookie as soft state, and it MUST change to the Enforced policy for DNS requests to that DNS server IP address. This policy change to Enforced is

treated as soft state with the same retention strategy as the

Server Cookie value for that server. On discard of that state information, the policy for that DNS server IP address reverts to Enabled.

Enforced:
Always include a COOKIE OPT option in requests.
Silently ignore all replies that do not include exactly one COOKIE OPT option having the correct Resolver Cookie value.
On receipt of a reply with one COOKIE OPT option carrying the correct Resolver Cookie value (even if it is a DNS error response), the DNS client performs normal response processing, including caching the received Server Cookie. If a copy of the same Server Cookie value is already cached for that server, then its retention probability should be increased. For example, if a time out is being used for the discard to cached Server Cookies, that time out should be extended.

6.2 Server Policies and Implementation

A server will logically have one of the following three modes of operation or "policies" for each DNS resolver as discussed below.

Disabled:
Ignore COOKIE OPT options in requests.
Never include a COOKIE OPT option in replies.

Enabled:
Include a COOKIE OPT option in replies to requests that include a COOKIE OPT.
Normally process requests without a COOKIE OPT option except that it is RECOMMENDED that the processing of burdensome requests and requests producing replies substantially longer than the request be significantly rate limited.
Ignore, other than sending a FormErr/MANYCOOKIE error reply, any request with more than one COOKIE OPT option. Such replies MAY be rate limited and SHOULD be as short as practical.
Ignore, other than sending a BADCOOKIE error reply, any request with one COOKIE OPT option if it has an incorrect Server Cookie unless the request COOKIE has an Error Code of CKPING (Cookie Ping) in which case the response has an Error Code of CKPINGR (Cookie Ping Response). Such replies MAY be rate limited and SHOULD be as short as practical.
On receipt of a request with one COOKIE OPT option carrying the correct Server Cookie value and an Error Code of zero, the DNS server performs normal request processing and it SHOULD switch to the Enforced policy for DNS requests from that resolver IP

address with that Resolver Cookie in the request. This policy change to Enforced is treated as soft state. On discard of that

state information, the policy for that resolver IP and Resolver Cookie pair reverts to Enabled.

Enforced:
Always include a COOKIE OPT option in replies.
Ignore requests without a COOKIE OPT option or with more than one COOKIE OPT option, other than returning a NOCOOKIE or MANYCOOKIE DNS error respectively. Such replies MAY be rate limited to any particular IP address and SHOULD be as short as practical.
Ignore requests with one COOKIE OPT option if they have an incorrect Server Cookie, other than returning a BADCOOKIE error message, unless the request has an Error Code of CKPING in which case the response has an Error Code of CKPINGR. Such replies MAY be rate limited and SHOULD be as short as practical.
If a request has one COOKIE OPT option with a correct Server Cookie and an Error Code of zero, perform normal processing of the request.

6.3 Resolver and Server Secret Rollover

Resolvers and servers MUST NOT continue to use the same secret in new queries and responses, respectively, for more than 14 days and SHOULD NOT continue to do so for more than 1 day. It is RECOMMENDED that a resolver keep the Resolver Cookie it is expecting in a reply associated with the outstanding query to avoid rejection of replies due to a bad Resolver Cookie right after a change in the Resolver Secret. It is RECOMMENDED that a server retain its previous secret for a period of time not less than 1 second or more than 3 minutes, after a change in its secret, and consider queries with Server Cookies based on its previous secret to have a correct Server Cookie during that time.

Receiving a suddenly increased level of requests with bad Server Cookies or replies with bad Resolver Cookies would be a good reason to believe a server or resolver likely to be under attack and should consider more frequent rollover of its secret.

6.4 Implementation Requirement

DNS resolvers and servers SHOULD implement DNS cookies.

DNS resolvers SHOULD operate in and be shipped so as to default to the Enabled or Enforced mode for all servers.

DNS servers SHOULD operate in and be shipped so as to default to the Enabled or Enforced mode for all resolvers they are willing to service.

7. NAT Considerations and AnyCast Server Considerations

In the Classic Internet, DNS Cookies could simply be a pseudo-random function of the resolver IP address and a sever secret or the server IP address and a resolver secret. You would want to compute the Server Cookie that way, so a resolver could cache its Server Cookie for a particular server for an indefinitely amount of time and the server could easily regenerate and check it. You could consider the Resolver Cookie to be a weak resolver signature over the server IP address that the resolver checks in replies and you could extend this weak signature to cover the request ID, for example, or any other information that is returned unchanged in the reply.

But we have this reality called NAT [RFC3022], Network Address Translation (including, for the purposes of this document, NAT-PT, Network Address and Protocol Translation, which has been declared Historic [RFC4966]). There is no problem with DNS transactions between resolvers and servers behind a NAT box using local IP addresses. Nor is there a problem with NAT translation of internal addresses to external addresses or translations between IPv4 and IPv6 addresses, as long as the address mapping is relatively stable. Should the external IP address an internal resolver being mapped to change occasionally, the disruption is little more than when a resolver rolls-over its DNS COOKIE secret. And normally external access to a DNS server behind a NAT box is handled by a fixed mapping which forwards externally received DNS requests to a specific host.

However, NAT devices sometimes also map ports. This can cause multiple DNS requests and responses from multiple internal hosts to be mapped to a smaller number of external IP addresses, such as one address. Thus there could be many resolvers behind a NAT box that appear to come from the same source IP address to a server outside that NAT box. If one of these were an attacker (think Zombie or Botnet), that behind-NAT attacker could get the Server Cookie for some server for the outgoing IP address by just making some random request to that server. It could then include that Server Cookie in the COOKIE OPT of requests to the server with the forged local IP address of some other host and/or resolver behind the NAT box. (Attacker possession of this Server Cookie will not help in forging responses to cause cache poisoning as such responses are protected by the required Resolver Cookie.)

To fix this potential defect, it is necessary to distinguish different resolvers behind a NAT box from the point of view of the server. It is for this reason that the Server Cookie is specified as a pseudo-random function of both the request source IP address and the Resolver Cookie. From this inclusion of the Resolver Cookie in the calculation of the Server Cookie, it follows that a stable

Resolver Cookie, for any particular server, is needed. If, for example, the request ID was included in the calculation of the

Resolver Cookie, it would normally change with each request to a particular server. This would mean that each request would have to be sent twice: first to learn the new Server Cookie based on this new Resolver Cookie based on the new ID and then again using this new Resolver Cookie to actually get an answer. Thus the input to the Resolver Cookie computation must be limited to the server IP address and one or more things that change slowly such as the resolver secret.

In principle, there could be a similar problem for servers, not due to NAT but due to mechanisms like anycast which may cause queries to a DNS server at an IP address to be delivered to any one of several machines. (External queries to a DNS server behind a NAT box usually occur via port forwarding such that all such queries go to one host.) However, it is impossible to solve this the way the similar problem was solved for NATed resolvers; if the Server Cookie was included in the calculation of the Resolver Cookie the same way the Resolver Cookie is included in the Server Cookie, you would just get an almost infinite series of errors as a request was repeatedly retried.

For servers accessed via anycast to successfully support DNS COOKIES, the server clones must either all use the same server secret or the mechanism that distributes queries to them must cause the queries from a particular resolver to go to a particular server for a sufficiently long period of time that extra queries due to changes in Server Cookie resulting from accessing different server machines are not unduly burdensome. (When such anycast accessed servers act as recursive servers or otherwise act as resolvers they normally use a different unique address to source their queries to avoid confusion in the delivery of responses.)

For simplicity, it is RECOMMENDED that the same server secret be used by each DNS server in a set of anycast servers. If there is limited time skew in updating this secret in different anycast servers, this can be handled by a server accepting requests containing a Server Cookie based on either its old or new secret for the maximum likely time period of such time skew (see also Section 6.3).

8. Deployment

The DNS cookies mechanism is designed for incremental deployment and to complement the orthogonal techniques in [RFC5452]. Either or both techniques can be deployed independently at each DNS server and resolver.

In particular, a DNS server or resolver that implements the DNS COOKIE mechanism and is in the Enabled mode will interoperate successfully with a DNS resolver or server that does not implement this mechanism although, of course, in this case it will not get the benefit of the mechanism. When such a server or resolver interoperates with a resolver or server which also implements the DNS cookies mechanism and is in Enabled or Enforced mode, this is recognized and, for that transaction partner, it latches up into the Enforced mode and gets the full benefit of the DNS cookies mechanism until this soft state lapses and it reverts to Enabled mode.

9. IANA Considerations

IANA will assign the following code points:

The OPT option value for COOKIE is <TBD>.

Three new DNS error codes are assigned values in the range above 15 and below 3841 as listed below:

NOCOOKIE is assigned the value TBD (23 suggested).
BADCOOKIE is assigned the value TBD (24 suggested).
MANYCOOKIE is assigned the value TBD (25 suggested).

Two new DNS error codes are assigned in the range above 4095 and below 65535:

CKPING (Cookie PING) is assigned the value TBD (4096 suggested).

CKPINGR (Cookie PING Response) is assigned the value RBD (4097 suggested).

10. Security Considerations

DNS Cookies provide a weak form of authentication of DNS requests and responses. In particular, they provide no protection at all against "on-path" adversaries; that is, they provide no protection against any adversary that can observe the plain text DNS traffic, such as an on-path router, bridge, or any device on an on-path shared link (unless the DNS traffic in question on that path is encrypted).

For example, if a host is connected via an unsecured IEEE 802.11 link (Wi-Fi), any device in the vicinity that could receive and decode the 802.11 transmissions must be considered "on-path". On the other hand, in a similar situation but one where 802.11 Robust Security (WPAv2) is appropriately deployed on the Wi-Fi network nodes, only the Access Point via which the host is connecting is "on-path".

Despite these limitations, use of DNS Cookies on the global Internet is expected to provide a substantial reduction in the available launch points for the traffic amplification and denial of service forgery attacks described in Section 2 above.

Should stronger message/transaction security be desired, it is suggested that TSIG or SIG(0) security be used (see Section 3.2); however, it may be useful to use DNS Cookies in conjunction with these features. In particular, DNS Cookies could screen out many DNS messages before the cryptographic computations of TSIG or SIG(0) are required and, if SIG(0) is in use, DNS Cookies could usefully screen out many requests given that SIG(0) does not screen requests but only authenticates the response of complete transactions.

10.1 Cookie Algorithm Considerations

The cookie computation algorithm for use in DNS Cookies SHOULD be FNV-64 [FNV] or some stronger algorithm because an excessively weak or trivial algorithm could enable adversaries to guess cookies. However, in light of the weak plain-text token security provided by DNS Cookies, a strong cryptography hash algorithm is probably not warranted in many cases, and would cause an increased computational burden. Nevertheless there is nothing wrong with using something stronger, for example, HMAC-SHA256-64 [RFC6234], assuming a DNS processor has adequate computational resources available. DNS processors that feel the need for somewhat stronger security without a significant increase in computational load should consider more frequent changes in their resolver and/or server secret; however, this does require more frequent generation of a cryptographically strong random number [RFC4086bis].

Acknowledgements

The contributions of the following are gratefully acknowledged:

Tim Wicinski

Normative References

[FNV] - G. Fowler, L. C. Noll, K.-P. Vo, D. Eastlake, "The FNV Non-Cryptographic Hash Algorithm", draft-eastlake-fnv, work in progress.

[RFC2119] - Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6891] - Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, April 2013.

[RFC4086bis] - Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", draft-eastlake-randomness3, work in progress.

Informative References

[RFC2845] - Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

[RFC2930] - Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, September 2000.

[RFC2931] - Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, September 2000.

[RFC3022] - Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC4033] - Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] - Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] - Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4966] - Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", RFC 4966, July 2007.

[RFC5452] - Hubert, A. and R. van Mook, "Measures for Making DNS More

Resilient against Forged Answers", RFC 5452, January 2009.

[RFC6234] - Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

Author's Address

Donald E. Eastlake 3rd
Huawei Technologies
155 Beaver Street
Milford, MA 01757 USA

Telephone: +1-508-333-2270
EMail: d3e3e3@gmail.com

Copyright, Disclaimer, and Additional IPR Provisions