DPRIVE
Internet-Draft
Intended status: Standards Track
Expires: November 7, 2015

T. Reddy
D. Wing
P. Patil
Cisco

May 6, 2015

DNS over DTLS (DNSoD)
draft-wing-dprive-dnsodtls-01

Abstract

DNS queries and responses are visible to network elements on the path between the DNS client and its server. These queries and responses can contain privacy-sensitive information which is valuable to protect. An active attacker can send bogus responses causing misdirection of the subsequent connection.

To counter passive listening and active attacks, this document proposes the use of Datagram Transport Layer Security (DTLS) for DNS, to protect against passive listeners and certain active attacks. As DNS needs to remain fast, this proposal also discusses mechanisms to reduce DTLS round trips and reduce DTLS handshake size. The proposed mechanism runs over the default DNS port and can also run over an alternate port.

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 November 7, 2015.

Table of Contents

1. Introduction

The Domain Name System is specified in [RFC1034] and [RFC1035]. DNS queries and responses are normally exchanged unencrypted and are thus vulnerable to eavesdropping. Such eavesdropping can result in an undesired entity learning domains that a host wishes to access, thus resulting in privacy leakage. DNS privacy problem is further discussed in [I-D.bortzmeyer-dnsop-dns-privacy].

Active attackers have long been successful at injecting bogus responses, causing cache poisoning and causing misdirection of the subsequent connection (if attacking A or AAAA records). A popular mitigation against that attack is to use ephemeral and random source ports for DNS queries.

This document defines DNS over DTLS (DNSoD, pronounced "dee-enn-sod") which provides confidential DNS communication for stub resolvers, recursive resolvers, iterative resolvers and authoritative servers.

The motivations for proposing DNSoD are that

- TCP suffers from network head-of-line blocking, where the loss of a packet causes all other TCP segments to not be delivered to the application until the lost packet is re-transmitted. DNSoD, because it uses UDP, does not suffer from network head-of-line blocking.

- DTLS session resumption consumes 1 round trip whereas TLS session resumption can start only after TCP handshake is complete. Although TCP Fast Open [RFC7413] can reduce that handshake, TCP Fast Open is not yet available in commercially-popular operating systems.

2. Relationship to TCP Queries and to DNSSEC

DNS queries can be sent over UDP or TCP. The scope of this document, however, is only UDP. DNS over TCP could be protected with TLS, as described in [I-D.hzhwm-start-tls-for-dns]. Alternatively, a shim protocol could be defined between DTLS and DNS, allowing large responses to be sent over DTLS itself, see Section 7.

DNS Security Extensions (DNSSEC [RFC4033]) provides object integrity of DNS resource records, allowing end-users (or their resolver) to verify legitimacy of responses. However, DNSSEC does not protect privacy of DNS requests or responses. DNSoD works in conjunction with DNSSEC, but DNSoD does not replace the need or value of DNSSEC.

3. Common problems with DNS Privacy

This section describes problems common to any DNS privacy solution. To achieve DNS privacy an encrypted and integrity-protected channel is needed between the client and server. This channel can be blocked, and the client needs to react to such blockages.

3.1. Firewall Blocking Ports or DNS Privacy Protocol

When sending DNS over an encrypted channel, there are two choices: send the encrypted traffic over the DNS ports (UDP 53, TCP 53) or send the encrypted traffic over a different port. The encrypted traffic is not normal DNS traffic, but rather is a cryptographic handshake followed by encrypted payloads. There can be firewalls, other security devices, or intercepting DNS proxies which block the non-DNS traffic or otherwise react negatively (e.g., quarantining the

host for suspicious behavior). Alternatively, if a different port is used for the encrypted traffic, a firewall or other security device might block that port or otherwise react negatively.

There is no panacea, and only experiments on the Internet will uncover which technique or combination of techniques will work best. The authors believe a combination of techniques will be necessary, as that has proven necessary with other protocols that desire to work on existing networks.

3.2. Authenticating the DNS Privacy Server

DNS privacy requires encrypting the query (and response) from passive attacks. Such encryption typically provides integrity protection as a side-effect, which means on-path attackers cannot simply inject bogus DNS responses. However, to provide stronger protection from active attackers pretending to be the server, the server itself needs to be authenticated.

To authenticate the server providing DNS privacy, the DNS client needs to be configured with the names or IP addresses of those DNS privacy servers. The server certificate MUST contain DNS-ID (subjectAltName) as described in Section 4.1 of [RFC6125]. DNS names and IP addresses can be contained in the subjectAltName entries. The client MUST use the rules and guidelines given in section 6 of [RFC6125] to validate the DNS server identity.

We imagine this could be implemented by adding the certificate name to the /etc/resolv.conf file, such as below:

```
nameserver 8.8.8.8
certificate google-public-dns.google.com
nameserver 208.67.220.220
certificate resolver.opendns.com
```

For DNS privacy servers that don't have a certificate trust chain (e.g., because they are on a home network or a corporate network), the configured list of DNS privacy servers can contain the certificate fingerprint of the DNS privacy server (i.e., a simple whitelist of name and certificate fingerprint).

We imagine this could be implemented by adding the certificate fingerprint to the /etc/resolv.conf file, such as below (line split for Internet Draft formatting):

```
nameserver 192.168.1.1
certificate-fingerprint
  01:56:D3:AC:CF:5B:3F:B8:8F:0F:B4:30:88:2D:F6:72:4E:8C:F2:EE
```

3.3. Downgrade attacks

Using DNS privacy with an authenticated server is most preferred, DNS privacy with an unauthenticated server is next preferred, and plain DNS is least preferred. An implementation will attempt to obtain DNS privacy by contacting DNS servers on the local network (provided by DHCP) and on the Internet, and will make those attempts in parallel to reduce user impact. If DNS privacy cannot be successfully negotiated for whatever reason, client can do three things:

1. refuse to send DNS queries on this network, which means the client can not make effective use of this network, as modern networks require DNS; or,

2. use DNS privacy with an un-authorized server, which means an attacker could be spoofing the handshake with the DNS privacy server; or,

3. send plain DNS queries on this network, which means no DNS privacy is provided.

Heuristics can improve this situation, but only to a degree (e.g., previous success of DNS privacy on this network may be reason to alert the user about failure to establish DNS privacy on this network now). Still, the client (in cooperation with the end user) has to decide to use the network without the protection of DNS privacy.

4. Terminology

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 [RFC2119].

5. Incremental Deployment

DNSoD can be deployed incrementally by the Internet Service Provider or as an Internet service.

If the ISP's DNS resolver supports DNSoD, then DNS queries are protected from passive listening and from many active attacks along that path.

DNSoD can be offered as an Internet service, and a stub resolver or DNS resolver can be configured to point to that DNSoD server (rather than to the ISP-provided DNS server).

6. Demultiplexing, Polling, Port Usage, and Discovery

[Note - This section requires further discussion]

Many modern operating systems already detect if a web proxy is interfering with Internet communications, using proprietary mechanisms that are out of scope of this document. After that mechanism has run (and detected Internet connectivity is working), the DNSoD procedure described in this document should commence. This timing avoids delays in joining the network (and displaying an icon indicating successful Internet connection), at the risk that those initial DNS queries will be sent without protection afforded by DNSoD.

DNSoD can run over standard UDP port 53 as defined in [RFC1035]. A DNS client or server that does not implement this specification will not respond to the incoming DTLS packets because they don't parse as DNS packets (the DNS Opcode would be 15, which is undefined). A DNS client or server that does implement this specification can demultiplex DNS and DTLS packets by examining the third octet. For TLS 1.2, which is what is defined by this specification, a DTLS packet will contain 253 in the third octet, whereas a DNS packet will never contain 253 in the third octet.

There has been some concern with sending DNSoD traffic over the same port as normal, un-encrypted DNS traffic. The intent of this section is to show that DNSoD could successfully be sent over port 53. Further analysis and testing on the Internet may be valuable to determine if multiplexing on port 53, using a separate port, or some fallback between a separate port and port 53 brings the most success.

After performing the above steps, the host should determine if the DNS server supports DNSoD by sending a DTLS ClientHello message. A DNS server that does not support DNSoD will not respond to ClientHello messages sent by the client, because they are not valid DNS requests (specifically, the DNS Opcode is invalid). The client MUST use timer values defined in Section 4.2.4.1 of [RFC6347] for retransmission of ClientHello message and if no response is received from the DNS server. After 15 seconds, it MUST cease attempts to re-transmit its ClientHello. Thereafter, the client MAY repeat that procedure in the event the DNS server has been upgraded to support DNSoD, but such probing SHOULD NOT be done more frequently than every 24 hours and MUST NOT be done more frequently than every 15 minutes. This mechanism requires no additional signaling between the client and server.

7. Performance Considerations

To reduce number of octets of the DTLS handshake, especially the size of the certificate in the ServerHello (which can be several kilobytes), we should consider using plain public keys [I-D.ietf-tls-oob-pubkey]. Considering that to authorize a certain DNS server the client already needs explicit configuration of the DNS servers it trusts, maybe the public key configuration problem is really no worse than the configuration problem of those whitelisted certificates?

Multiple DNS queries can be sent over a single DNSoD security association. The existing QueryID allows multiple requests and responses to be interleaved in whatever order they can be fulfilled by the DNS server. This means DNSoD reduces the consumption of UDP port numbers, and because DTLS protects the communication between the DNS client and its server, the resolver SHOULD NOT use random ephemeral source ports (Section 9.2 of [RFC5452]) because such source port use would incur additional, unnecessary DTLS load on the DNSoD server.

It is highly advantageous to avoid server-side DTLS state and reduce the number of new DTLS security associations on the server which can be done with [RFC5077]. This also eliminates a round-trip for subsequent DNSoD queries, because with [RFC5077] the DTLS security association does not need to be re-established. Note: with the shim (described below) perhaps we could send the query and the restore server-side state in the ClientHello packet.

Compared to normal DNS, DTLS adds at least 13 octets of header, plus cipher and authentication overhead to every query and every response. This reduces the size of the DNS payload that can be carried. Certain DNS responses are large (e.g., many AAAA records, TXT, SRV) and don't fit into a single UDP packet, causing a partial response with the truncation (TC) bit set. The client is then expected to repeat the query over TCP, which causes additional name resolution delay. We have considered two ideas, one that reduces the need to switch to TCP and another that eliminates the need to switch to TCP:

- Path MTU can be determined using Packetization Layer Path MTU Discovery [RFC4821] using DTLS heartbeat. [RFC4821] does not rely on ICMP or ICMPv6, and would not affect DNS state or responsiveness on the client or server. However, it would be additional chattiness.

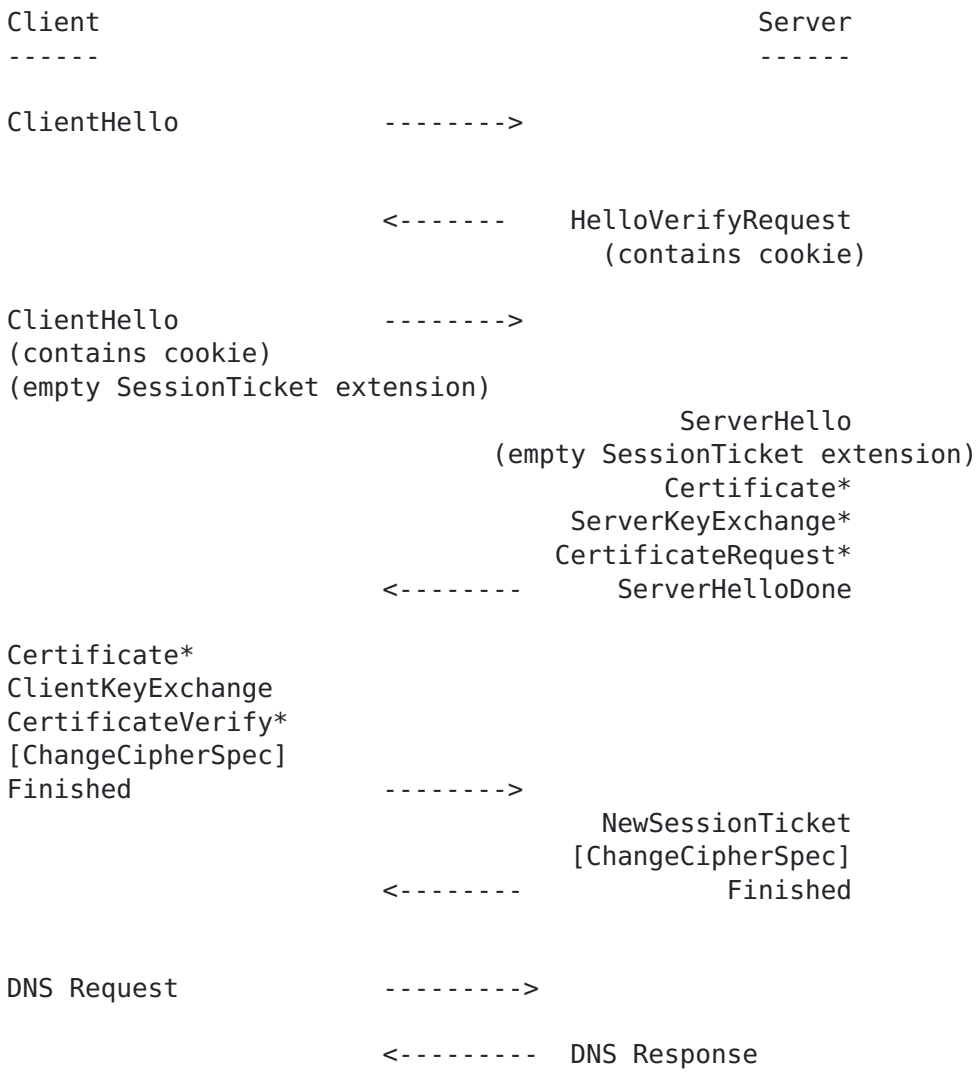
- To avoid IP fragmentation, DTLS handshake messages incorporate their own fragment offset and fragment length. We might utilize a similar mechanism in a shim layer between DTLS and DNS, so that

large DNS messages could be carried without causing IP fragmentation.

DNSoD puts an additional computational load on servers. The largest gain for privacy is to protect the communication between the DNS client (the end user's machine) and its caching resolver. Because of the load imposed, and because of the infrequency of queries to root servers means the DTLS overhead is unlikely to be amoritized over the DNS queries sent over that DTLS connection, implementing DNSoD on root servers is NOT RECOMMENDED.

8. Established sessions

In DTLS, all data is protected using the same record encoding and mechanisms. When the mechanism described in this document is in effect, DNS messages are encrypted using the standard DTLS record encoding. When a user of DTLS wishes to send an DNS message, it delivers it to the DTLS implementation as an ordinary application data write (e.g., SSL_write()). A single DTLS session can be used to receive multiple DNS requests and generate DNS multiple responses.



Message Flow for Full Handshake Issuing New Session Ticket

9. DTLS Features and Cipher Suites

To improve interoperability, the set of DTLS features and cipher suites is restricted. The DTLS implementation MUST disable compression. DTLS compression can lead to the exposure of information that would not otherwise be revealed [RFC3749]. Generic compression is unnecessary since DNS provides compression features itself. DNS over DTLS MUST only be used with cipher suites that have ephemeral key exchange, such as the ephemeral Diffie-Hellman (DHE) [RFC5246] or the elliptic curve variant (ECDHE) [RFC4492]. Ephemeral key exchange MUST have a minimum size of 2048 bits for DHE or security level of 128 bits for ECDHE. Authenticated Encryption with

Additional Data (AEAD) modes, such as the Galois Counter Model (GCM) mode for AES [RFC5288] are acceptable.

10. Anycast

DNS servers are often configured with anycast addresses. While the network is stable, packets transmitted from a particular source to an anycast address will reach the same server that has the cryptographic context from the DNS over DTLS handshake. But when the network configuration changes,a DNS over DTLS packet can be received by a server that does not have the necessary cryptographic context. To encourage the client to initiate a new DTLS handshake, DNS servers SHOULD generate a DTLS Alert message in response to receiving a DTLS packet for which the server does not have any cryptographic context.

11. IANA Considerations

If demultiplexing DTLS and DNS (using the third octet, Section 6) is useful, we should reserve DNS Opcode 15 to ensure DNS always has a 0 bit where DTLS always has a 1 bit.

12. Security Considerations

The interaction between the DNS client and the DNS server requires Datagram Transport Layer Security (DTLS) with a ciphersuite offering confidentiality protection and the guidance given in [RFC7525] must be followed to avoid attacks on DTLS. Once a DNSoD client has established a security association with a particular DNS server, and outstanding normal DNS queries with that server (if any) have been received, the DNSoD client MUST ignore any subsequent normal DNS responses from that server, as all subsequent responses should be inside DNSoD. This behavior mitigates all (?) attacks described in Measures for Making DNS More Resilient against Forged Answers [RFC5452].

Security considerations discussed in DTLS [RFC6347] also apply to this document.

13. Acknowledgements

Thanks to Phil Hedrick for his review comments on TCP and to Josh Littlefield for pointing out DNSoD load on busy servers (most notably root servers). The authors would like to thank Simon Josefsson for discussions and comments on the design of DNSoD.

14. References

14.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

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

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

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, August 2008.

[RFC5452] Hubert, A. and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", RFC 5452, January 2009.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, May 2015.

14.2. Informative References

[I-D.bortzmeyer-dnsop-dns-privacy]
Bortzmeyer, S., "DNS privacy considerations", draft-bortzmeyer-dnsop-dns-privacy-02 (work in progress), April 2014.

[I-D.hzhwm-start-tls-for-dns]
Zi, Z., Zhu, L., Heidemann, J., Mankin, A., and D. Wessels, "Starting TLS over DNS", draft-hzhwm-start-tls-for-dns-01 (work in progress), July 2014.

[I-D.ietf-tls-oob-pubkey]
Wouters, P., Tschofenig, H., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", draft-ietf-tls-oob-pubkey-11 (work in progress), January 2014.

[RFC3749] Hollenbeck, S., "Transport Layer Security Protocol Compression Methods", RFC 3749, May 2004.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, December 2014.

Authors' Addresses

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Prashanth Patil
Cisco Systems, Inc.
Bangalore
India

Email: praspati@cisco.com