DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 8, 2015

W. Wijngaards
NLnet Labs
G. Wiley
VeriSign, Inc.

September 4, 2014

Confidential DNS
draft-wijngaards-dnsop-confidentialdns-02

Abstract

This document offers opportunistic encryption to provide privacy for DNS queries and responses.

Requirements Language

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 RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The privacy of the Question, Answer, Authority and Additional sections in DNS queries and responses is protected by the confidential DNS protocol by encrypting the contents of each section. The goal of this change to the DNS protocol is to make large scale monitoring more expensive, see [draft-bortzmeyer-dnsop-dns-privacy] and [draft-koch-perpass-dns-confidentiality]. Authenticity and integrity may be provided by DNSSEC, this protocol does not change DNSSEC and does not offer the means to authenticate responses.

Confidential communication between any pair of DNS servers is supported, both between iterative resolvers and authoritative servers and between stub resolvers and recursive resolvers.

The confidential DNS protocol has minimal impact on the number of packets involved in a typical DNS query/response exchange by leveraging a cacheable ENCRYPT Resource Record and an optionally cacheable shared secret. The protocol supports selectable cryptographic suites and parameters (such as key sizes).

The client fetches an ENCRYPT RR from the server that it wants to contact. The public key retrieved in the ENCRYPT RR is used to encrypt a shared secret or public key that the client uses to encrypt the sections in the DNS query and which the name server uses to encrypt the DNS response. Note that an ENCRYPT RR must be fetched for each name server in order for the entire session to be confidential.

As this is opportunistic encryption, the key is (re-)fetched when the exchange fails. If the key fetch fails or the encrypted query fails, communication in the clear may be performed.

The server advertises which crypto suites and key lengths may be used in the ENCRYPT RR, the client then chooses a crypto suite from this list and includes that selection in subsequent DNS queries.

The key from the server can be cached by the client, using the TTL specified in the ENCRYPT RR, the IP address of the server distinguishes keys in the cache. The server may also cache shared secrets and keys from clients.

2. ENCRYPT RR Type

The RR type for confidential DNS is ENCRYPT TBD (decimal). The presentation format is:

```
. ENCRYPT [flags] [algo] [id] [data]
```

The flags, algo and id are unsigned numbers in decimal and the data is in base-64. The wireformat is: one octet flags, one octet algo, one octet id and the remainder of the rdata is for the data. The type is class independent and it is a hop-by-hop transaction RR type. The domain name of the ENCRYPT record is '.' (the root label) for hop-by-hop exchanges.

In the flags the least two bits are used as usage value. The other flag bits MUST be ignored by receivers and sent as zeroes.

- PAD (value=0): the ENCRYPT contains padding material. Algo and id are set to 0. Its data length is random (say 1-63 octets), and has some random values. It is a resource record that may be appended to resource records that are encrypted so that identical queries encrypt to different encrypted data of different lengths.

- KEY (value=1): the ENCRYPT contains a public or symmetric key. The algo field gives the algorithm. The id identifies the key, this id is copied to ENCRYPT type RRS to identify which key to use to decrypt the data. The data contains the key bits.

- RRS (value=2): encrypted data. The data contains encrypted resource records. The data is encrypted with the selected algorithm and key id. The data contains resource records in DNS wireformat [RFC1034], with a domain name, type, class, ttl, rdatalength and rdata.

- SYM (value=3): the ENCRYPT contains an encrypted symmetric key. The contained, encrypted data is rdata of an ENCRYPT of type KEY and has the symmetric key. The data is encrypted with the algorithm and id indicated. The encrypted data encompasses the flags, algo, id, data for the symmetric key.

The ENCRYPT RR type can contain keys. It uses the same format as the DNSKEY record [RFC4034] for public keys. algo=0 is reserved for future expansion of the algorithm number above 255. algo=1 is RSA, the rdata determines the key size, such as 512 and 768 bits. algo=2 is AES, aes-cbc, size of the rdata determines the size of the key, such as 128 and 192 bits.

3. Server and Client Algorithm

If a clients wants to fetch the keys for the server from the server, it performs a query with query type ENCRYPT and query name '.' (root label). The reply contains the ENCRYPT (or multiple if a choice is offered) in the answer section. These ENCRYPTs have the KEY flag set.

If a client wants to perform an encrypted query, it sends an unencrypted outer packet, with query type ENCRYPT and query name '.' (root label). In the additional section it includes an ENCRYPT record of type RRS. This encrypts a number of records, the first is a query-section style query record, and then zero or more ENCRYPTs of type KEY that the server uses to encrypt the reply. If the client wants to use a symmetric key, there are no ENCRYPTs of type KEY inside the encrypted ENCRYPT data, instead an ENCRYPT of type SYM is positioned in the outer packet, before the ENCRYPT of type RRS and the ENCRYPT of type RRS is encrypted with the symmetric key.

If a server wants to encrypt a reply, it also uses the ENCRYPT type. The reply looks like a normal DNS packet, i.e. it has a normal unencrypted outer DNS packet. Because the query name and query type have been encrypted, the outer packet has a query name of '.' and query type of ENCRYPT and the reply has an ENCRYPT record in the answer section with flag RRS. The reply RRs have been encrypted into the data of the ENCRYPT record. The RR counts for every section are stored in the outer (unencrypted) header. Thus, the combination of the original header and the decrypted data from this record results in the decrypted packet.

The client may lookup keys whenever it wants to. It may cache the keys for the server, using the TTL of those ENCRYPT records. It should also cache failures to lookup the ENCRYPT record for some time (eg. the negative TTL if the reply contained one). Errors and also timeouts should also be taken as an indication that the ENCRYPT cannot be looked up, and the client MUST fall back to unencrypted communication (this is the opportunistic encryption case). The result of an encrypted query may also be timeouts, errors or replies with mangled contents, in that case the client MUST fall back to unencrypted communication (this is the opportunistic encryption case). Note that if some middlebox removes the ENCRYPT from the additional section of an encrypted query, likely a reply with ENCRYPTs of type KEY is returned instead of the encrypted reply with an ENCRYPT of type RRS, and again the client does the unencrypted fallback. If the server has changed its keys and does not recognize the keys in an encrypted query, it should return FORMERR, and include its current ENCRYPTs of type KEY in that FORMERR reply. A server may decide it does not (any longer) have the resources for encryption and reply with SERVFAIL to encrypted queries, forcing unencrypted fallback. Keys for unknown algorithms should be ignored by the client, if no usable keys remain, fallback to insecure.

The client may cache the ENCRYPT of type SYM for a server together with the symmetric secret, this is better for performance, as public-key operations can be avoided for repeated queries. The server may also cache the ENCRYPTs of type SYM with the decoded secret,

associating a lookup for the rdata of the SYM record with the decoded secret, avoiding public-key operations for repeated queries.

Key rollover is possible, use different key ids and support the old key for its TTL, while advertising the new key, for the servers. For clients, generate a new public or symmetric key and use it.

4. Authenticated Operation

The previous documented the opportunistic operation, where deployment is easier, but security is weaker. This documents options for authenticated operation. The client selects if encryption is authenticated, opportunistic, or disabled in its local policy (configuration).

The authentication happens with a DNSSEC signed DS record that carries the key for confidential DNS. This removes a full roundtrip from the connection setup cost. The DS has hash type TBDhashtype, that is specific for confidential DNS. The DS record carries a flag byte and the public key (in DNSKEY's wireformat) in its rdata data blob. This means that normally the confidential DNS keys are acquired with a referral to a zone and can be authenticated with DNSSEC.

Because the key itself is carried, the probe sequence can be avoided and an encrypted query can be sent straight away. This makes the protocol about equally fast as normal DNS (from a packet latency point of view) for DNSSEC signed zones. The servers for that zone have to share a public and private keypair between them that is used to authenticate the encrypted queries and answers. That would be a part of the nameserver configuration.

Because RFC4034 validators do not know this new hash type, they will ignore them and continue to DNSSEC validate the zone. Software that understands the new DS hash type, know that the DS record really carries confidential DNS keys and not DNSKEY hashes. These should continue to validate the zone with DNSSEC as normal with the remaining DS records. If there are no other DS records with another hash type, then the zone is not signed with DNSSEC, and the validator should not require DNSSEC for the zone.

This changes the opportunistic encryption to authenticated encryption. The fallback to insecure is still possible and this may make deployment easier. The one byte at the start of the base64 data, in its least significant bit, signals if fallback to insecure is allowed (value 0x01). That gives the zone owner the option to enable fallback to insecure or if it should be disabled. The remainder of the DS base64 data contains a public key in the same

wireformat as a DNSKEY has public keys. The type of the key is in the key type field of this DS record. With fallback to insecure disabled and the keys authenticated the confidential DNS query and response should be fully secure (i.e. not 'Opportunistically' secure). [[Note: we could also put the flags in the keytag field]].

With fallback to insecure disabled, queries fail instead of falling back to insecure. This means no answer is acquired, and DNS lookups for that zone fail because the security failed.

Servers with both downstream DNS clients and upstream DNS lookups MUST NOT give an answer to their downstream client when fallback to insecure is disabled and security fails, they MAY reply with SERVFAIL. (this wording leaves wiggle room for ratelimiting by such servers).

The DS method works for authority servers. For recursors, authentication can be performed by getting the keys (or hashes) either from DHCP or configuration, the same path as the IP address for the recursor was configured. However, these methods are usually very insecure. An improvement is for the client to cache the keys for the resolver on stable storage. This is performed by the server sending an RR together with the ENCRYPT keys in the key lookup, this RR has a fallback-to-insecure flag, and a TTL value for caching the key entry (suggested at 30 days). The cache TTL is set back at the full value every time the confidential DNS keys are probed, according to their normal TTL, which keeps the cache fresh. If the TTL time has elapsed since the last time the keys were probed, the client performs a new leap of faith, an insecure probe again. Keys can be signed with RRSIGs made with (cached) keys, so key rollovers can be performed. For relatively frequent queriers, this would make often visited resolvers safe. The resolver is identified by its IP address (perhaps coupled with the client idea of its 'network location', like wifi ssid), and the keys are stored for that IP address.

The RR is ENCRYPT with flags 0x04 (STORE, please store keys on disk for a a while, leap of faith style security), and flags 0x08 (fallback to insecure is allowed, or disabled). The id and algo are sent zero, ignore on receipt. The data portion contains 4 bytes with an unsigned 32 bit network byteorder number with a TTL in seconds for the disk cache. The TTL expires that number of seconds after the most recent ENCRYPT probe. More bytes in data portion, do not send, ignore on receipt. So, ENCRYPT flag value 12 to store and allow fallback to insecure could be sent along, or value 4 to disallow fallback to insecure.

For authenticated operation, fallback to insecure should not be performed. However, this will significantly harm deployment as

unclean lookup paths result in lookup failure. Keys with unsupported crypto algorithms MUST still be ignored and if no keys are left, fallback to insecure MUST still be performed, also for authenticated operation.

The key for recursive resolvers can be configured into the stub machines, or a domain name can be configured where the keys are looked up and they are signed with DNSSEC.

5. Comparison with TLS and DTLS

An alternative method of accomplishing confidential DNS would be to leverage one of the existing means for establishing a secure transport layer. For example a secure TCP session could be established to the name server over which DNS queries could be sent with no changes to the DNS protocol. The most significant down side to this approach is the burden that it places on high volume name servers. Very large scale DNS operators expect to answer hundreds of thousands of queries per second (possibly even more than a million qps) for each host in their name server footprint. The use of technologies such as IPSec or TLS may have such a severe impact on the largest name server operators as to impede adoption of confidential DNS.

DTLS (RFC 6347) offers a more interesting approach to securing the connection to a name server that may be implemented in a way that is less abusive to the large scale name servers. It looks as though the overhead imposed by DTLS would probably be significantly higher than the protocol described in this draft, however if the session established via DTLS is used over a large number of queries then the cost of the handshake could be amortized over the total number of queries.

6. IANA Considerations

An RR type registration for type ENCRYPT with number TBD and it references this document [[to be done when this becomes RFC]].

A DS record hash type is registered TBDhashtype that references this document. It is for the confidential DNS public key, acronym CONFKEY.

7. Security Considerations

Opportunistic encryption can be configured. Opportunistic encryption has many drawbacks, against active intrusion, but works against passives. The pervasive passive surveillance problem statement and also its security considerations are applicable to this document.

Hence the suggested short key sizes and opportunistic encryption.

With authentication the key, if security works, is authenticated. With fallback to insecure disabled, security is full featured. The keys are then signed by DNSSEC for authority servers, and a leap-of-faith store after first contact security for resolvers that want that.

This technique does not protect against timing, traffic analysis (what IP address is contacted), and the packet size, RR count, header flags and header RCODE can be observed. These could provide almost all the information that was encrypted. Such as: query to IP address for example.com nameservers, size of the packet is similar to a www.example.com lookup and is followed by http packets to www.example.com's IP address.

8. Acknowledgments

Roy Arends

9. Normative References

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

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

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

Authors' Addresses

Wouter Wijngaards
NLnet Labs
Science Park 140
Amsterdam 1098 XH
The Netherlands

EMail: wouter@nlnetlabs.nl

Glen Wiley
VeriSign, Inc.
Reston, VA
USA

EMail: gwiley@verisign.com