Network Working Group
Request for Comments: 2845
Category: Standards Track
Updates: 1035

P. Vixie
ISC
O. Gudmundsson
NAI Labs
D. Eastlake 3rd
Motorola
B. Wellington
Nominum

May 2000

Secret Key Transaction Authentication for DNS (TSIG)

Status of this Memo

Copyright Notice

Abstract

This protocol allows for transaction level authentication using shared secrets and one way hashing. It can be used to authenticate dynamic updates as coming from an approved client, or to authenticate responses as coming from an approved recursive name server.

No provision has been made here for distributing the shared secrets; it is expected that a network administrator will statically configure name servers and clients using some out of band mechanism such as sneaker-net until a secure automated mechanism for key distribution is available.

1 - Introduction

1.1. The Domain Name System (DNS) [RFC1034, RFC1035] is a replicated hierarchical distributed database system that provides information fundamental to Internet operations, such as name <=> address translation and mail handling information. DNS has recently been extended [RFC2535] to provide for data origin authentication, and public key distribution, all based on public key cryptography and public key based digital signatures. To be practical, this form of

security generally requires extensive local caching of keys and tracing of authentication through multiple keys and signatures to a pre-trusted locally configured key.

1.2. One difficulty with the [RFC2535] scheme is that common DNS implementations include simple "stub" resolvers which do not have caches. Such resolvers typically rely on a caching DNS server on another host. It is impractical for these stub resolvers to perform general [RFC2535] authentication and they would naturally depend on their caching DNS server to perform such services for them. To do so securely requires secure communication of queries and responses. [RFC2535] provides public key transaction signatures to support this, but such signatures are very expensive computationally to generate. In general, these require the same complex public key logic that is impractical for stubs. This document specifies use of a message authentication code (MAC), specifically HMAC-MD5 (a keyed hash function), to provide an efficient means of point-to-point authentication and integrity checking for transactions.

1.3. A second area where use of straight [RFC2535] public key based mechanisms may be impractical is authenticating dynamic update [RFC2136] requests. [RFC2535] provides for request signatures but with [RFC2535] they, like transaction signatures, require computationally expensive public key cryptography and complex authentication logic. Secure Domain Name System Dynamic Update ([RFC2137]) describes how different keys are used in dynamically updated zones. This document’s secret key based MACs can be used to authenticate DNS update requests as well as transaction responses, providing a lightweight alternative to the protocol described by [RFC2137].

1.4. A further use of this mechanism is to protect zone transfers. In this case the data covered would be the whole zone transfer including any glue records sent. The protocol described by [RFC2535] does not protect glue records and unsigned records unless SIG(0) (transaction signature) is used.

1.5. The authentication mechanism proposed in this document uses shared secret keys to establish a trust relationship between two entities. Such keys must be protected in a fashion similar to private keys, lest a third party masquerade as one of the intended parties (forge MACs). There is an urgent need to provide simple and efficient authentication between clients and local servers and this proposal addresses that need. This proposal is unsuitable for general server to server authentication for servers which speak with many other servers, since key management would become unwieldy with

the number of shared keys going up quadratically. But it is suitable for many resolvers on hosts that only talk to a few recursive servers.

1.6. A server acting as an indirect caching resolver -- a "forwarder" in common usage -- might use transaction-based authentication when communicating with its small number of preconfigured "upstream" servers. Other uses of DNS secret key authentication and possible systems for automatic secret key distribution may be proposed in separate future documents.

1.7. New Assigned Numbers

```
RRTYPE = TSIG (250)
ERROR = 0..15 (a DNS RCODE)
ERROR = 16 (BADSIG)
ERROR = 17 (BADKEY)
ERROR = 18 (BADTIME)
```

1.8. The key words "MUST", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in [RFC 2119].

2 - TSIG RR Format

2.1 TSIG RR Type

To provide secret key authentication, we use a new RR type whose mnemonic is TSIG and whose type code is 250. TSIG is a meta-RR and MUST not be cached. TSIG RRs are used for authentication between DNS entities that have established a shared secret key. TSIG RRs are dynamically computed to cover a particular DNS transaction and are not DNS RRs in the usual sense.

2.2 TSIG Calculation

As the TSIG RRs are related to one DNS request/response, there is no value in storing or retransmitting them, thus the TSIG RR is discarded once it has been used to authenticate a DNS message. The only message digest algorithm specified in this document is "HMAC-MD5" (see [RFC1321], [RFC2104]). The "HMAC-MD5" algorithm is mandatory to implement for interoperability. Other algorithms can be specified at a later date. Names and definitions of new algorithms MUST be registered with IANA. All multi-octet integers in the TSIG record are sent in network byte order (see [RFC1035 2.3.2]).

2.3. Record Format

NAME The name of the key used in domain name syntax. The name should reflect the names of the hosts and uniquely identify the key among a set of keys these two hosts may share at any given time. If hosts A.site.example and B.example.net share a key, possibilities for the key name include <id>.A.site.example, <id>.B.example.net, and <id>.A.site.example.B.example.net. It should be possible for more than one key to be in simultaneous use among a set of interacting hosts. The name only needs to be meaningful to the communicating hosts but a meaningful mnemonic name as above is strongly recommended.

The name may be used as a local index to the key involved and it is recommended that it be globally unique. Where a key is just shared between two hosts, its name actually only need only be meaningful to them but it is recommended that the key name be mnemonic and incorporate the resolver and server host names in that order.

TYPE TSIG (250: Transaction SIGnature)

CLASS ANY

TTL 0

RdLen (variable)

RDATA

Field Name	Data Type	Notes
Algorithm Name	domain-name	Name of the algorithm in domain name syntax.
Time Signed	u_int48_t	seconds since 1-Jan-70 UTC.
Fudge	u_int16_t	seconds of error permitted in Time Signed.
MAC Size	u_int16_t	number of octets in MAC.
MAC	octet stream	defined by Algorithm Name.
Original ID	u_int16_t	original message ID
Error	u_int16_t	expanded RCODE covering TSIG processing.
Other Len	u_int16_t	length, in octets, of Other Data.
Other Data	octet stream	empty unless Error == BADTIME

2.4. Example

NAME HOST.EXAMPLE.

TYPE TSIG

CLASS ANY

TTL 0

RdLen as appropriate

RDATA

Field Name	Contents
Algorithm Name	SAMPLE-ALG.EXAMPLE.
Time Signed	853804800
Fudge	300
MAC Size	as appropriate
MAC	as appropriate
Original ID	as appropriate
Error	0 (NOERROR)
Other Len	0
Other Data	empty

3 - Protocol Operation

3.1. Effects of adding TSIG to outgoing message

Once the outgoing message has been constructed, the keyed message digest operation can be performed. The resulting message digest will then be stored in a TSIG which is appended to the additional data section (the ARCOUNT is incremented to reflect this). If the TSIG record cannot be added without causing the message to be truncated, the server MUST alter the response so that a TSIG can be included. This response consists of only the question and a TSIG record, and has the TC bit set and RCODE 0 (NOERROR). The client SHOULD at this point retry the request using TCP (per [RFC1035 4.2.2]).

3.2. TSIG processing on incoming messages

If an incoming message contains a TSIG record, it MUST be the last record in the additional section. Multiple TSIG records are not allowed. If a TSIG record is present in any other position, the packet is dropped and a response with RCODE 1 (FORMERR) MUST be returned. Upon receipt of a message with a correctly placed TSIG RR, the TSIG RR is copied to a safe location, removed from the DNS

Message, and decremented out of the DNS message header's ARCOUNT. At this point the keyed message digest operation is performed. If the algorithm name or key name is unknown to the recipient, or if the message digests do not match, the whole DNS message MUST be discarded. If the message is a query, a response with RCODE 9 (NOTAUTH) MUST be sent back to the originator with TSIG ERROR 17 (BADKEY) or TSIG ERROR 16 (BADSIG). If no key is available to sign this message it MUST be sent unsigned (MAC size == 0 and empty MAC). A message to the system operations log SHOULD be generated, to warn the operations staff of a possible security incident in progress. Care should be taken to ensure that logging of this type of event does not open the system to a denial of service attack.

3.3. Time values used in TSIG calculations

The data digested includes the two timer values in the TSIG header in order to defend against replay attacks. If this were not done, an attacker could replay old messages but update the "Time Signed" and "Fudge" fields to make the message look new. This data is named "TSIG Timers", and for the purpose of digest calculation they are invoked in their "on the wire" format, in the following order: first Time Signed, then Fudge. For example:

Field Name	Value	Wire Format	Meaning
Time Signed	853804800	00 00 32 e4 07 00	Tue Jan 21 00:00:00 1997
Fudge	300	01 2C	5 minutes

3.4. TSIG Variables and Coverage

When generating or verifying the contents of a TSIG record, the following data are digested, in network byte order or wire format, as appropriate:

3.4.1. DNS Message

A whole and complete DNS message in wire format, before the TSIG RR has been added to the additional data section and before the DNS Message Header's ARCOUNT field has been incremented to contain the TSIG RR. If the message ID differs from the original message ID, the original message ID is substituted for the message ID. This could happen when forwarding a dynamic update request, for example.

3.4.2. TSIG Variables

Source	Field Name	Notes
TSIG RR	NAME	Key name, in canonical wire format
TSIG RR	CLASS	(Always ANY in the current specification)
TSIG RR	TTL	(Always 0 in the current specification)
TSIG RDATA	Algorithm Name	in canonical wire format
TSIG RDATA	Time Signed	in network byte order
TSIG RDATA	Fudge	in network byte order
TSIG RDATA	Error	in network byte order
TSIG RDATA	Other Len	in network byte order
TSIG RDATA	Other Data	exactly as transmitted

The RR RDLEN and RDATA MAC Length are not included in the hash since they are not guaranteed to be knowable before the MAC is generated.

The Original ID field is not included in this section, as it has already been substituted for the message ID in the DNS header and hashed.

For each label type, there must be a defined "Canonical wire format" that specifies how to express a label in an unambiguous way. For label type 00, this is defined in [RFC2535], for label type 01, this is defined in [RFC2673]. The use of label types other than 00 and 01 is not defined for this specification.

3.4.3. Request MAC

When generating the MAC to be included in a response, the request MAC must be included in the digest. The request’s MAC is digested in wire format, including the following fields:

Field	Type	Description
MAC Length	u_int16_t	in network byte order
MAC Data	octet stream	exactly as transmitted

3.5. Padding

Digested components are fed into the hashing function as a continuous octet stream with no interfield padding.

4 - Protocol Details

4.1. TSIG generation on requests

Client performs the message digest operation and appends a TSIG record to the additional data section and transmits the request to the server. The client MUST store the message digest from the request while awaiting an answer. The digest components for a request are:

```
DNS Message (request)
TSIG Variables (request)
```

Note that some older name servers will not accept requests with a nonempty additional data section. Clients SHOULD only attempt signed transactions with servers who are known to support TSIG and share some secret key with the client -- so, this is not a problem in practice.

4.2. TSIG on Answers

When a server has generated a response to a signed request, it signs the response using the same algorithm and key. The server MUST not generate a signed response to an unsigned request. The digest components are:

```
Request MAC
DNS Message (response)
TSIG Variables (response)
```

4.3. TSIG on TSIG Error returns

When a server detects an error relating to the key or MAC, the server SHOULD send back an unsigned error message (MAC size == 0 and empty MAC). If an error is detected relating to the TSIG validity period, the server SHOULD send back a signed error message. The digest components are:

```
Request MAC (if the request MAC validated)
DNS Message (response)
TSIG Variables (response)
```

The reason that the request is not included in this digest in some cases is to make it possible for the client to verify the error. If the error is not a TSIG error the response MUST be generated as specified in [4.2].

4.4. TSIG on TCP connection

A DNS TCP session can include multiple DNS envelopes. This is, for example, commonly used by zone transfer. Using TSIG on such a connection can protect the connection from hijacking and provide data integrity. The TSIG MUST be included on the first and last DNS envelopes. It can be optionally placed on any intermediary envelopes. It is expensive to include it on every envelopes, but it MUST be placed on at least every 100'th envelope. The first envelope is processed as a standard answer, and subsequent messages have the following digest components:

```
Prior Digest (running)
DNS Messages (any unsigned messages since the last TSIG)
TSIG Timers (current message)
```

This allows the client to rapidly detect when the session has been altered; at which point it can close the connection and retry. If a client TSIG verification fails, the client MUST close the connection. If the client does not receive TSIG records frequently enough (as specified above) it SHOULD assume the connection has been hijacked and it SHOULD close the connection. The client SHOULD treat this the same way as they would any other interrupted transfer (although the exact behavior is not specified).

4.5. Server TSIG checks

Upon receipt of a message, server will check if there is a TSIG RR. If one exists, the server is REQUIRED to return a TSIG RR in the response. The server MUST perform the following checks in the following order, check KEY, check TIME values, check MAC.

4.5.1. KEY check and error handling

If a non-forwarding server does not recognize the key used by the client, the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 17 (BADKEY). This response MUST be unsigned as specified in [4.3]. The server SHOULD log the error.

4.5.2. TIME check and error handling

If the server time is outside the time interval specified by the request (which is: Time Signed, plus/minus Fudge), the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 18 (BADTIME). The server SHOULD also cache the most recent time signed value in a message generated by a key, and SHOULD return BADTIME if a message received later has an earlier time signed value. A response indicating a BADTIME error MUST be signed by the same key as the

request. It MUST include the client's current time in the time signed field, the server's current time (a u_int48_t) in the other data field, and 6 in the other data length field. This is done so that the client can verify a message with a BADTIME error without the verification failing due to another BADTIME error. The data signed is specified in [4.3]. The server SHOULD log the error.

4.5.3. MAC check and error handling

If a TSIG fails to verify, the server MUST generate an error response as specified in [4.3] with RCODE 9 (NOTAUTH) and TSIG ERROR 16 (BADSIG). This response MUST be unsigned as specified in [4.3]. The server SHOULD log the error.

4.6. Client processing of answer

When a client receives a response from a server and expects to see a TSIG, it first checks if the TSIG RR is present in the response. Otherwise, the response is treated as having a format error and discarded. The client then extracts the TSIG, adjusts the ARCOUNT, and calculates the keyed digest in the same way as the server. If the TSIG does not validate, that response MUST be discarded, unless the RCODE is 9 (NOTAUTH), in which case the client SHOULD attempt to verify the response as if it were a TSIG Error response, as specified in [4.3]. A message containing an unsigned TSIG record or a TSIG record which fails verification SHOULD not be considered an acceptable response; the client SHOULD log an error and continue to wait for a signed response until the request times out.

4.6.1. Key error handling

If an RCODE on a response is 9 (NOTAUTH), and the response TSIG validates, and the TSIG key is different from the key used on the request, then this is a KEY error. The client MAY retry the request using the key specified by the server. This should never occur, as a server MUST NOT sign a response with a different key than signed the request.

4.6.2. Time error handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 18 (BADTIME), or the current time does not fall in the range specified in the TSIG record, then this is a TIME error. This is an indication that the client and server clocks are not synchronized. In this case the client SHOULD log the event. DNS resolvers MUST NOT adjust any clocks in the client based on BADTIME errors, but the server's time in the other data field SHOULD be logged.

4.6.3. MAC error handling

If the response RCODE is 9 (NOTAUTH) and TSIG ERROR is 16 (BADSIG), this is a MAC error, and client MAY retry the request with a new request ID but it would be better to try a different shared key if one is available. Client SHOULD keep track of how many MAC errors are associated with each key. Clients SHOULD log this event.

4.7. Special considerations for forwarding servers

A server acting as a forwarding server of a DNS message SHOULD check for the existence of a TSIG record. If the name on the TSIG is not of a secret that the server shares with the originator the server MUST forward the message unchanged including the TSIG. If the name of the TSIG is of a key this server shares with the originator, it MUST process the TSIG. If the TSIG passes all checks, the forwarding server MUST, if possible, include a TSIG of his own, to the destination or the next forwarder. If no transaction security is available to the destination and the response has the AD flag (see [RFC2535]), the forwarder MUST unset the AD flag before adding the TSIG to the answer.

5 - Shared Secrets

5.1. Secret keys are very sensitive information and all available steps should be taken to protect them on every host on which they are stored. Generally such hosts need to be physically protected. If they are multi-user machines, great care should be taken that unprivileged users have no access to keying material. Resolvers often run unprivileged, which means all users of a host would be able to see whatever configuration data is used by the resolver.

5.2. A name server usually runs privileged, which means its configuration data need not be visible to all users of the host. For this reason, a host that implements transaction-based authentication should probably be configured with a "stub resolver" and a local caching and forwarding name server. This presents a special problem for [RFC2136] which otherwise depends on clients to communicate only with a zone’s authoritative name servers.

5.3. Use of strong random shared secrets is essential to the security of TSIG. See [RFC1750] for a discussion of this issue. The secret should be at least as long as the keyed message digest, i.e. 16 bytes for HMAC-MD5 or 20 bytes for HMAC-SHA1.

6 - Security Considerations

6.1. The approach specified here is computationally much less expensive than the signatures specified in [RFC2535]. As long as the shared secret key is not compromised, strong authentication is provided for the last hop from a local name server to the user resolver.

6.2. Secret keys should be changed periodically. If the client host has been compromised, the server should suspend the use of all secrets known to that client. If possible, secrets should be stored in encrypted form. Secrets should never be transmitted in the clear over any network. This document does not address the issue on how to distribute secrets. Secrets should never be shared by more than two entities.

6.3. This mechanism does not authenticate source data, only its transmission between two parties who share some secret. The original source data can come from a compromised zone master or can be corrupted during transit from an authentic zone master to some "caching forwarder." However, if the server is faithfully performing the full [RFC2535] security checks, then only security checked data will be available to the client.

6.4. A fudge value that is too large may leave the server open to replay attacks. A fudge value that is too small may cause failures if machines are not time synchronized or there are unexpected network delays. The recommended value in most situation is 300 seconds.

7 - IANA Considerations

IANA is expected to create and maintain a registry of algorithm names to be used as "Algorithm Names" as defined in Section 2.3. The initial value should be "HMAC-MD5.SIG-ALG.REG.INT". Algorithm names are text strings encoded using the syntax of a domain name. There is no structure required other than names for different algorithms must be unique when compared as DNS names, i.e., comparison is case insensitive. Note that the initial value mentioned above is not a domain name, and therefore need not be a registered name within the DNS. New algorithms are assigned using the IETF Consensus policy defined in RFC 2434. The algorithm name HMAC-MD5.SIG-ALG.REG.INT looks like a FQDN for historical reasons; future algorithm names are expected to be simple (i.e., single-component) names.

IANA is expected to create and maintain a registry of "TSIG Error values" to be used for "Error" values as defined in section 2.3. Initial values should be those defined in section 1.7. New TSIG error codes for the TSIG error field are assigned using the IETF Consensus policy defined in RFC 2434.

8 - 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 1034, November 1987.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC1750] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1995.

[RFC2104] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC-MD5: Keyed-MD5 for Message Authentication", RFC 2104, February 1997.

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

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y. and J. Bound "Dynamic Updates in the Domain Name System", RFC 2136, April 1997.

[RFC2137] Eastlake 3rd, D., "Secure Domain Name System Dynamic Update", RFC 2137, April 1997.

[RFC2535] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[RFC2673] Crawford, M., "Binary Labels in the Domain Name System", RFC 2673, August 1999.

9 - Authors' Addresses

Paul Vixie
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063

Phone: +1 650 779 7001
EMail: vixie@isc.org

Olafur Gudmundsson
NAI Labs
3060 Washington Road, Route 97
Glenwood, MD 21738

Phone: +1 443 259 2389
EMail: ogud@tislabs.com

Donald E. Eastlake 3rd
Motorola
140 Forest Avenue
Hudson, MA 01749 USA

Phone: +1 508 261 5434
EMail: dee3@torque.pothole.com

Brian Wellington
Nominum, Inc.
950 Charter Street
Redwood City, CA 94063

Phone: +1 650 779 6022
EMail: Brian.Wellington@nominum.com

10 Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.