Network File System Version 4
Internet-Draft
Updates: 5531 (if approved)
Intended status: Standards Track
Expires: August 15, 2019

T. Myklebust
Hammerspace
C. Lever, Ed.
Oracle

February 11, 2019

Remote Procedure Call Encryption By Default
draft-cel-nfsv4-rpc-tls-02

Abstract

This document describes a mechanism that enables encryption of in-transit Remote Procedure Call (RPC) transactions with minimal administrative overhead and full interoperation with RPC implementations that do not support this mechanism. This document updates RFC 5531.

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 https://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 August 15, 2019.

Copyright Notice

Table of Contents

1. Introduction

In 2014 the IETF published [RFC7258] which recognized that unauthorized observation of network traffic had become widespread and was a subversive threat to all who make use of the Internet at large. It strongly recommended that newly defined Internet protocols make a real effort to mitigate monitoring attacks. Typically this mitigation is done by encrypting data in transit.

The Remote Procedure Call version 2 protocol has been a Proposed Standard for three decades (see [RFC5531] and its antecedants).

Eisler et al. first introduced an in-transit encryption mechanism for RPC with RPCSEC GSS over twenty years ago [RFC2203]. However, experience has shown that RPCSEC GSS is difficult to deploy:

- Per-client deployment and administrative costs are not scalable. Keying material must be provided for each RPC client, including transient clients.

- Parts of the RPC header remain in clear-text, and can constitute a significant security exposure.

- On-host cryptographic manipulation of data payloads can exact a significant CPU cost on both clients and the server.

- Host identity management must be carried out in a security realm that is separate from user identity management. In certain environments, for example, different authorities might be responsible for provisioning client systems versus provisioning new users.

However strong a privacy service is, it can not provide any security if the difficulties of deploying and using it result in it not being used at all.

An alternative approach is to employ a transport layer security mechanism that can protect the privacy of each RPC connection transparently to RPC and Upper Layer protocols. The Transport Layer Security protocol [RFC8446] (TLS) is a well-established Internet building block that protects many common Internet protocols such as the Hypertext Transport Protocol (http) [RFC2818].

Encrypting at the RPC transport layer enables several significant benefits.

Encryption By Default
: In-transit encryption can be enabled immediately after installation without additional administrative actions such as identifying the host system to a trust authority, generating additional key material, or provisioning a secure network tunnel.

Protection of Existing Protocols
: The imposition of encryption at the transport layer protects any Upper Layer protocol that employs RPC without alteration of that protocol. RPC transport layer encryption can protect recent versions of NFS such as NFS version 4.2 [RFC7862] and indeed legacy NFS versions such as NFS version 3 [RFC1813] and NFS side-band protocols such as the MNT protocol [RFC1813].

Decoupled User and Host Identities
: RPCSEC GSS provides a framework for cryptographically protecting user and host identities but assumes that both are managed by the same security authority.

Encryption Offload
: The use of a well-established transport encryption mechanism that is also employed by other very common network protocols makes it possible to use hardware encryption implementations so that the host CPU is not burdened with the work of encrypting and decrypting large RPC arguments and results.

2. Requirements Language

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

3. Terminology

This document adopts the terminology introduced in Section 3 of [RFC6973] and assumes a working knowledge of the Remote Procedure Call (RPC) version 2 protocol [RFC5531] and the Transport Layer Security (TLS) version 1.3 protocol [RFC8446].

Note also that the NFS community uses the term "privacy" where other Internet communities might use "confidentiality". In this document the two terms are synonymous.

4. RPC-Over-TLS in Operation

In this section we cleave to the convention that a "client" is the peer host that actively initiates a connection, and a "server" is the peer host that passively accepts a connection request.

4.1. Discovering Server-side TLS Support

The mechanism described in this document interoperates fully with implementations that do not support it. The use of TLS is automatically disabled in these cases. To achieve this, we introduce a new authentication flavor called AUTH_TLS. This new flavor is used to signal that the client wants to initiate TLS negotiation if the server supports it.

```
<CODE BEGINS>

enum auth_flavor {
        AUTH_NONE       = 0,
        AUTH_SYS        = 1,
        AUTH_SHORT      = 2,
        AUTH_DH         = 3,
        AUTH_KERB       = 4,
        AUTH_RSA        = 5,
        RPCSEC_GSS      = 6,
        AUTH_TLS        = 7,

        /* and more to be defined */
};

<CODE ENDS>
```

The length of the opaque data constituting the credential sent in the call message MUST be zero. The verifier accompanying the credential MUST be an AUTH_NONE verifier of length zero.

The flavor value of the verifier received in the reply message from the server MUST be AUTH_NONE. The bytes of the verifier's string encode the fixed ASCII characters "STARTTLS".

When an RPC client is ready to initiate a TLS handshake, it sends a NULL RPC request with an auth_flavor of AUTH_TLS. The NULL request is made to the same port as if TLS were not in use.

The RPC server can respond in one of three ways:

- If the RPC server does not recognise the AUTH_TLS authentication flavor, it responds with a reject_stat of AUTH_ERROR. The RPC client then knows that this server does not support TLS.

- If the RPC server accepts the NULL RPC procedure, but fails to return an AUTH_NONE verifier containing the string "STARTTLS", the RPC client knows that this server does not support TLS.

- If the RPC server accepts the NULL RPC procedure, and returns an AUTH_NONE verifier containing the string "STARTTLS", the RPC client MAY proceed with TLS negotiation.

If an RPC client attempts to use AUTH_TLS for anything other than the NULL RPC procedure, the RPC server responds with a reject_stat of AUTH_ERROR.

Once the TLS handshake is complete, the RPC client and server will have established a secure channel for communicating and can proceed to use standard security flavors within that channel, presumably after negotiating down the irrelevant RPCSEC_GSS privacy and integrity services and applying channel binding [RFC7861].

If TLS negotiation fails for any reason -- say, the RPC server rejects the certificate presented by the RPC client, or the RPC client fails to authenticate the RPC server -- the RPC client reports this failure to the calling application the same way it would report an AUTH_ERROR rejection from the RPC server.

4.2. Streams and Datagrams

RPC operates on several different types of transports. RPC on a stream transport is protected by using TLS [RFC8446]; on a datagram transport, RPC must use DTLS [RFC6347].

RPC-over-RDMA can make use of Transport Layer Security below the RDMA transport layer [RFC8166]. The exact mechanism is not within the scope of this document.

4.3. Authentication

Both RPC and TLS have their own variants of authentication, and there is some overlap in capability. The goal of interoperability with implementations that do not support TLS requires that we limit the combinations that are allowed and precisely specify the role that each layer plays. We also want to handle TLS such that an RPC implementation can make the use of TLS invisible to existing RPC consumer applications.

Toward these ends, there are two main deployment modes.

4.3.1. No Client Authentication

In a basic deployment, a server possesses a certificate that is self-signed or signed by a well-known trust anchor, while its clients might not possess a certificate. In this situation, the client MAY authenticate the server host, but the server cannot authenticate connecting clients. Here, encryption of the transport connection is established and the RPC requests in transit carry user and group identities according to the conventions of the ONC RPC protocol.

4.3.2. Client Authentication

In this type of deployment, both the server and its clients possess valid certificates. As part of the TLS handshake, both peers MAY authenticate. Should authentication of either peer fail, or should authorization based on those identities block access to the server, the connection can be rejected. However, once encryption of the transport connection is established, the server MUST NOT utilize TLS identity for the purpose of authorizing RPC requests.

In some cases, a client might choose to present a certificate that represents a user rather than one that is bound to the client host. As above, the server MUST NOT utilize this identity for the purpose of authorizing RPC requests.

4.3.3. Advanced Forms of RPC Authentication

RPCSEC GSS can provide integrity or privacy (also known as confidentiality) services. When operating over an encrypted TLS session, these services become redundant. Each RPC implementation is responsible for using channel binding for detecting when GSS integrity or privacy is unnecessary and can therefore be disabled See Section 2.5 of [RFC7861] for details.

Note that a GSS service principal is still required on the server, and mutual authentication of server and client still occurs after the TLS session is established.

4.3.4. Other Forms of TLS Authentication

Versions of TLS subsequent to TLS 1.2 feature a token binding mechanism which is nominally more secure than using certificates. This is discussed in further detail in [RFC8471]. When such versions of TLS are used to encrypted RPC traffic, token binding may replace the use of certificates, but the restrictions specified earlier in this section still apply.

5. Security Considerations

One purpose of the mechanism described in this document is to protect RPC-based applications against threats to the privacy of RPC transactions and RPC user identities. A taxonomy of these threats appears in Section 5 of [RFC6973]. In addition, Section 6 of [RFC7525] contains a detailed discussion of technologies used in conjunction with TLS. Implementers should familiarize themselves with these materials.

The NFS version 4 protocol permits more than one user to use an NFS client at the same time [RFC7862]. Typically that NFS client will conserve connection resources by routing RPC transactions from all of its users over a few or a single connection. In circumstances where the users on that NFS client belong to multiple distinct security domains, it may be necessary to establish separate TLS-protected connections that do not share the same encryption parameters.

5.1. Implications for AUTH_SYS

Ever since the IETF NFSV4 Working Group took over the maintenance of the NFSv4 family of protocols (currently specified in [RFC7530], [RFC5661], and [RFC7863], among others), it has encouraged the use of RPCSEC GSS over AUTH_SYS. For various reasons, unfortunately AUTH_SYS continues to be the primary authentication mechanism deployed by NFS administrators. As a result, NFS security remains in an unsatisfactory state.

A deeper purpose of this document is to attempt to address some of the shortcomings of AUTH_SYS so that, where it has been impractical to deploy RPCSEC GSS, better NFSv4 security can nevertheless be achieved.

When AUTH_SYS is used with TLS and no client certificate is available, the RPC server is still acting on RPC requests for which there is no trustworthy authentication. In-transit traffic is protected, but the client itself can still misrepresent user identity without detection. This is an improvement from AUTH_SYS without encryption, but it leaves a critical security exposure.

Therefore, the RECOMMENDED deployment mode is that both servers and clients have certificate material available so that servers can have a degree of trust that clients are acting responsibly.

6. IANA Considerations

In accordance with Section 6 of [RFC7301], the authors request that IANA allocate the following value in the "Application-Layer Protocol Negotiation (ALPN) Protocol IDs" registry. The "sunrpc" string identifies SunRPC when used over TLS.

Protocol:
   SunRPC

Identification Sequence:
   0x73 0x75 0x6e 0x72 0x70 0x63 ("sunrpc")

Reference:

7. References

7.1. Normative References

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

[RFC5531] Thurlow, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 5531, DOI 10.17487/RFC5531, May 2009, <https://www.rfc-editor.org/info/rfc5531>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC7861] Adamson, A. and N. Williams, "Remote Procedure Call (RPC) Security Version 3", RFC 7861, DOI 10.17487/RFC7861, November 2016, <https://www.rfc-editor.org/info/rfc7861>.

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

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

7.2. Informative References

[LJNL] Fisher, C., "Encrypting NFSv4 with Stunnel TLS", August 2018, <https://www.linuxjournal.com/content/encrypting-nfsv4-stunnel-tls>.

[RFC1813] Callaghan, B., Pawlowski, B., and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, DOI 10.17487/RFC1813, June 1995, <https://www.rfc-editor.org/info/rfc1813>.

[RFC2203] Eisler, M., Chiu, A., and L. Ling, "RPCSEC_GSS Protocol Specification", RFC 2203, DOI 10.17487/RFC2203, September 1997, <https://www.rfc-editor.org/info/rfc2203>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC5661] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, DOI 10.17487/RFC5661, January 2010, <https://www.rfc-editor.org/info/rfc5661>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[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, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC7530] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, <https://www.rfc-editor.org/info/rfc7530>.

[RFC7862] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 Protocol", RFC 7862, DOI 10.17487/RFC7862, November 2016, <https://www.rfc-editor.org/info/rfc7862>.

[RFC7863] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 External Data Representation Standard (XDR) Description", RFC 7863, DOI 10.17487/RFC7863, November 2016, <https://www.rfc-editor.org/info/rfc7863>.

[RFC8166] Lever, C., Ed., Simpson, W., and T. Talpey, "Remote Direct Memory Access Transport for Remote Procedure Call Version 1", RFC 8166, DOI 10.17487/RFC8166, June 2017, <https://www.rfc-editor.org/info/rfc8166>.

[RFC8471] Popov, A., Ed., Nystroem, M., Balfanz, D., and J. Hodges, "The Token Binding Protocol Version 1.0", RFC 8471, DOI 10.17487/RFC8471, October 2018, <https://www.rfc-editor.org/info/rfc8471>.

Acknowledgments

Special mention goes to Charles Fisher, author of "Encrypting NFSv4 with Stunnel TLS" [LJNL]. His article inspired the mechanism described in this document.

The authors are grateful to Bill Baker, David Black, Lars Eggert, Benjamin Kaduk Greg Marsden, Alex McDonald, David Noveck, Justin Mazzola Paluska, and Tom Talpey for their input and support of this work.

Special thanks go to Transport Area Director Spencer Dawkins, NFSV4 Working Group Chairs Spencer Shepler and Brian Pawlowski, and NFSV4 Working Group Secretary Thomas Haynes for their guidance and oversight.

Authors' Addresses

Trond Myklebust
Hammerspace Inc
4300 El Camino Real Ste 105
Los Altos, CA 94022
United States of America

Email: trond.myklebust@hammerspace.com

Charles Lever (editor)
Oracle Corporation
United States of America

Email: chuck.lever@oracle.com