

Network File System Version 4
Internet-Draft
Updates: [5531](#) (if approved)
Intended status: Standards Track
Expires: August 6, 2020

T. Myklebust
Hammerspace
C. Lever, Ed.
Oracle
February 3, 2020

Towards Remote Procedure Call Encryption By Default
draft-ietf-nfsv4-rpc-tls-06

Abstract

This document describes a mechanism that, through the use of opportunistic Transport Layer Security (TLS), enables encryption of in-transit Remote Procedure Call (RPC) transactions while interoperating with ONC 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 6, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	3
2.	Requirements Language	5
3.	Terminology	5
4.	RPC-Over-TLS in Operation	5
4.1.	Discovering Server-side TLS Support	5
4.2.	Authentication	7
4.2.1.	Using TLS with RPCSEC GSS	8
5.	TLS Requirements	8
5.1.	Base Transport Considerations	8
5.1.1.	Operation on TCP	8
5.1.2.	Operation on UDP	9
5.1.3.	Operation on Other Transports	9
5.2.	TLS Peer Authentication	10
5.2.1.	X.509 Certificates Using PKIX trust	10
5.2.2.	X.509 Certificates Using Fingerprints	11
5.2.3.	Pre-Shared Keys	11
5.2.4.	Token Binding	11
6.	Implementation Status	12
6.1.	DESY NFS server	12
6.2.	Hammerspace NFS server	12
6.3.	Linux NFS server and client	13
6.4.	FreeBSD NFS server and client	13
7.	Security Considerations	13
7.1.	Limitations of an Opportunistic Approach	14
7.1.1.	STRIPTLS Attacks	14
7.2.	TLS Identity Management on Clients	14
7.3.	Security Considerations for AUTH_SYS on TLS	15
7.4.	Best Security Policy Practices	16
8.	IANA Considerations	16
9.	References	16
9.1.	Normative References	16

9.2. Informative References	18
Appendix A. Known Weaknesses of the AUTH_SYS Authentication Flavor	19
Acknowledgments	20
Authors' Addresses	20

[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 should make a genuine 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 antecedents). Over twenty years ago, Eisler et al. first introduced RPCSEC GSS as an in-transit encryption mechanism for RPC [[RFC2203](#)]. However, experience has shown that RPCSEC GSS with in-transit encryption can be challenging to use in practice:

- o Parts of each RPC header remain in clear-text, constituting a significant security exposure.
- o Offloading GSS privacy is not practical in large multi-user deployments since each message is encrypted using a key based on the issuing RPC user.

However strong a privacy service is, it cannot provide any security if the challenges of using it result in choosing not to deploy it at all.

Moreover, the use of AUTH_SYS remains common despite the adverse effects that acceptance of UIDs and GIDs from unauthenticated clients brings with it. Continued use is in part because:

- o Per-client deployment and administrative costs are not scalable. Administrators must provide keying material for each RPC client, including transient clients.
- o Host identity management and user identity management must be enforced in the same security realm. In certain environments, different authorities might be responsible for provisioning client systems versus provisioning new users.

The alternative described in the current document 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 standard Internet protocols such as the Hypertext Transport Protocol (HTTP) [[RFC2818](#)].

Encrypting at the RPC transport layer accords several significant benefits:

Encryption By Default: Transport encryption can be enabled without additional administrative tasks such as identifying client systems to a trust authority, generating additional keying material, or provisioning a secure network tunnel.

Encryption Offload: Hardware support for GSS privacy has not appeared in the marketplace. However, the use of a well-established transport encryption mechanism that is employed by other ubiquitous network protocols makes it more likely that encryption offload for RPC is practicable.

Securing AUTH_SYS: Most critically, transport encryption can significantly reduce several security issues inherent in the current widespread use of AUTH_SYS (i.e., acceptance of UIDs and GIDs generated by an unauthenticated client).

Decoupled User and Host Identities: TLS can be used to authenticate peer hosts while other security mechanisms can handle user authentication.

The current document specifies the implementation of RPC on an encrypted transport in a fashion that is transparent to upper-layer protocols based on RPC. The imposition of encryption at the transport layer protects any upper-layer protocol that employs RPC, without alteration of that protocol.

Further, the current document defines policies in line with [[RFC7435](#)] which enable RPC-on-TLS to be deployed opportunistically in environments with RPC implementations that do not support TLS. Specifications for RPC-based upper-layer protocols are free to require stricter policies to guarantee that encryption or host authentication is in use on every connection.

The protocol specification in the current document assumes that support for RPC, TLS, PKI, GSS-API, and DNSSEC is already available in an RPC implementation where TLS support is to be added.

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 use "confidentiality". In the current document the two terms are synonymous.

We adhere to the convention that a "client" is a network host that actively initiates an association, and a "server" is a network host that passively accepts an association request.

RPC documentation historically refers to the authentication of a connecting host as "machine authentication" or "host authentication". TLS documentation refers to the same as "peer authentication". In the current document there is little distinction between these terms.

The term "user authentication" in this document refers specifically to the RPC caller's credential, provided in the "cred" and "verf" fields in each RPC Call.

4. RPC-Over-TLS in Operation

4.1. Discovering Server-side TLS Support

The mechanism described in this document interoperates fully with RPC implementations that do not support TLS. The use of TLS is automatically disabled in these cases.

To achieve this, we introduce a new RPC authentication flavor called AUTH_TLS. This new flavor signals that the client wants to initiate TLS negotiation if the server supports it. Except for the modifications described in this section, the RPC protocol is unaware of security encapsulation.

<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 length of the verifier's body field is eight. The bytes of the verifier's body field encode the ASCII characters "STARTTLS" as a fixed-length opaque.

When an RPC client is ready to begin sending encrypted traffic to a server, it starts with 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:

- o If the RPC server does not recognize 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.
- o If the RPC server accepts the NULL RPC procedure but fails to return an AUTH_NONE verifier containing the "STARTTLS" token in the NULL Reply, the RPC client knows that this server does not support TLS.
- o If the RPC server accepts the NULL RPC procedure and returns an AUTH_NONE verifier containing the "STARTTLS" token in the NULL Reply, the RPC client immediately initiates a TLS session. This NULL Reply signals that the RPC server is prepared for the client to begin TLS session negotiation.

Once the TLS handshake is complete, the RPC client and server have established a secure channel for communicating. The client **MUST** switch to a security flavor other than AUTH_TLS within that channel, presumably after negotiating down redundant RPCSEC_GSS privacy and integrity services and applying channel binding [[RFC7861](#)].

If TLS negotiation fails for any reason, the RPC client reports this failure to the upper-layer application the same way it would report an AUTH_ERROR rejection from the RPC server.

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

[4.2.](#) Authentication

Both RPC and TLS have peer and user authentication, with some overlap in capability between RPC and TLS. The goal of interoperability with implementations that do not support TLS requires limiting the combinations that are allowed and precisely specifying 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.

Each RPC server that supports RPC-over-TLS **MUST** possess a unique global identity (e.g., a certificate that is signed by a well-known trust anchor). Such an RPC server **MUST** request a TLS peer identity from each client upon first contact. There are two different modes of client deployment:

Server-only Host Authentication

In this type of deployment, the client can authenticate the server host using the presented server peer TLS identity, but the server cannot authenticate the client. In this situation, RPC-over-TLS clients are anonymous. They present no globally unique identifier to the server peer.

Mutual Host Authentication

In this type of deployment, the client possesses a unique global identity (e.g., a certificate). As part of the TLS handshake, both peers authenticate using the presented TLS identities. If authentication of either peer fails, or if authorization based on those identities blocks access to the server, the peers **MUST** reject the association.

In either of these modes, RPC user authentication is not affected by the use of transport layer security. When a client presents a TLS peer identity to an RPC server, the protocol extension described in

the current document provides no way for the server to know whether that identity represents one RPC user on that client, or is shared amongst many RPC users. Therefore, a server implementation must not utilize the remote TLS peer identity for RPC user authentication.

4.2.1. Using TLS with RPCSEC GSS

RPCSEC GSS can provide per-request integrity or privacy (also known as confidentiality) services. When operating over a TLS session, the GSS services become redundant. A TLS-capable RPC implementation uses GSS channel binding to determine when GSS integrity or privacy is unnecessary. See [Section 2.5 of \[RFC7861\]](#) for details.

When using GSS on a TLS session, the RPC server is still required to possess a GSS service principal. GSS mutual authentication still occurs after a TLS session has been established.

5. TLS Requirements

When peers negotiate a TLS session that is to transport RPC, the following restrictions apply:

- o Implementations MUST NOT negotiate TLS versions prior to v1.3 [\[RFC8446\]](#). Support for mandatory-to-implement ciphersuites for the negotiated TLS version is REQUIRED.
- o Implementations MUST support certificate-based mutual authentication. Support for TLS-PSK mutual authentication [\[RFC4279\]](#) is OPTIONAL. See [Section 4.2](#) for further details.
- o Negotiation of a ciphersuite providing confidentiality as well as integrity protection is REQUIRED. Support for and negotiation of compression is OPTIONAL.

5.1. Base Transport Considerations

5.1.1. Operation on TCP

The use of TLS [\[RFC8446\]](#) protects RPC on TCP connections. Typically, once a client completes the TCP handshake and performs RPC service discovery via NULL RPC operations, it uses the mechanism described in [Section 4.1](#) to discover TLS support. It can then negotiate a TLS session on that connection.

After establishing a TLS session, an RPC server MUST reject with a `reject_stat` of `AUTH_ERROR` any subsequent RPC requests over a TLS-protected connection that are outside of a TLS session. Likewise, an

RPC client MUST silently discard any subsequent RPC replies over the connection that are outside of a TLS session.

This restriction includes reverse-direction RPC operations (i.e., RPC calls initiated on the server-end of the connection). An RPC client receiving a reverse-direction call on a connection outside of an existing TLS session MUST reject the request with a `reject_stat` of `AUTH_ERROR`.

An RPC peer terminates a TLS session by sending a TLS closure alert, or by closing the TLS-protected TCP connection.

5.1.2. Operation on UDP

RPC over UDP is protected using DTLS [[RFC6347](#)]. As soon as a client initializes a socket for use with an unfamiliar server, it uses the mechanism described in [Section 4.1](#) to discover DTLS support and then negotiate a DTLS session. Connected operation is RECOMMENDED.

Using a DTLS transport does not introduce reliable or in-order semantics to RPC on UDP. Also, DTLS does not support fragmentation of RPC messages. Each RPC message MUST fit in a single DTLS datagram. DTLS encapsulation has overhead, which reduces the effective Path MTU (PMTU) and thus the maximum RPC payload size.

DTLS does not detect STARTTLS replay. Sending a TLS closure alert terminates a DTLS session. Subsequent RPC messages passing between the client and server are no longer protected until a new TLS session is established.

5.1.3. Operation on Other Transports

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. Because there might not be other provisions to exchange client and server certificates, authentication material exchange would need to be provided by facilities within a future RPC-over-RDMA transport.

Transports that provide intrinsic TLS-level security (e.g., QUIC) would need to be addressed separately from the current document. In such cases, the use of TLS would not be opportunistic as it is for TCP or UDP.

[5.2.](#) TLS Peer Authentication

TLS can perform peer authentication using any of the following mechanisms:

[5.2.1.](#) X.509 Certificates Using PKIX trust

Implementations are REQUIRED to support this mechanism. In this mode, the tuple (serial number of the presented certificate; Issuer) uniquely identifies the RPC peer.

- o Implementations MUST allow the configuration of a list of trusted Certification Authorities for incoming connections.
- o Certificate validation MUST include the verification rules as per [[RFC5280](#)].
- o Implementations SHOULD indicate their trusted Certification Authorities (CAs).
- o Peer validation always includes a check on whether the locally configured expected DNS name or IP address of the server that is contacted matches its presented certificate. DNS names and IP addresses can be contained in the Common Name (CN) or subjectAltName entries. For verification, only one of these entries is to be considered. The following precedence applies: for DNS name validation, subjectAltName:DNS has precedence over CN; for IP address validation, subjectAltName:iPAddr has precedence over CN. Implementors of this specification are advised to read [Section 6 of \[RFC6125\]](#) for more details on DNS name validation.
- o Implementations MAY allow the configuration of a set of additional properties of the certificate to check for a peer's authorization to communicate (e.g., a set of allowed values in subjectAltName:URI or a set of allowed X509v3 Certificate Policies).
- o When the configured trust base changes (e.g., removal of a CA from the list of trusted CAs; issuance of a new CRL for a given CA), implementations MAY renegotiate the TLS session to reassess the connecting peer's continued authorization.

Authenticating a connecting entity does not mean the RPC server necessarily wants to communicate with that client. For example, if the Issuer is not in a trusted set of Issuers, the RPC server may decline to perform RPC transactions with this client.

Implementations that want to support a wide variety of trust models

should expose as many details of the presented certificate to the administrator as possible so that the administrator can implement the trust model. As a suggestion, at least the following parameters of the X.509 client certificate SHOULD be exposed:

- o Originating IP address
- o Certificate Fingerprint
- o Issuer
- o Subject
- o all X509v3 Extended Key Usage
- o all X509v3 Subject Alternative Name
- o all X509v3 Certificate Policies

5.2.2. X.509 Certificates Using Fingerprints

This mechanism is OPTIONAL to implement. In this mode, the fingerprint of the presented certificate uniquely identifies the RPC peer.

Implementations SHOULD allow the configuration of a list of trusted certificates, identified via fingerprint of the DER-encoded certificate octets. Implementations MUST support SHA-256 [[FIPS.180-4](#)] or stronger as the hash algorithm for the fingerprint.

5.2.3. Pre-Shared Keys

This mechanism is OPTIONAL to implement. In this mode, the RPC peer is uniquely identified by keying material that has been shared out-of-band or by a previous TLS-protected connection (see [Section 2.2 of \[RFC8446\]](#)). At least the following parameters of the TLS connection SHOULD be exposed:

- o Originating IP address
- o TLS Identifier

5.2.4. Token Binding

This mechanism is OPTIONAL to implement. In this mode, a token uniquely identifies the RPC peer.

Versions of TLS after TLS 1.2 contain a token binding mechanism that is more secure than using certificates. This mechanism is detailed in [[RFC8471](#)].

6. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [[RFC7942](#)]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs.

Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

6.1. DESY NFS server

Organization: DESY

URL: <https://desy.de>

Maturity: Implementation will be based on mature versions of the current document.

Coverage: The implementation is under way. The use of DTLS functionality is not implemented.

Licensing: LGPL

Implementation experience: The implementer has read and commented on the current document.

6.2. Hammerspace NFS server

Organization: Hammerspace

URL: <https://hammerspace.com>

Maturity: Prototype software based on early versions of this document.

Coverage: The bulk of this specification is implemented. The use of DTLS functionality is not implemented.

Licensing: Proprietary

Implementation experience: No comments from implementors.

6.3. Linux NFS server and client

Organization: The Linux Foundation

URL: <https://www.kernel.org>

Maturity: Prototype software based on early versions of this document.

Coverage: The bulk of this specification has yet to be implemented. The use of DTLS functionality is not planned.

Licensing: GPLv2

Implementation experience: No comments from the implementor.

6.4. FreeBSD NFS server and client

Organization: The FreeBSD Project

URL: <https://www.freebsd.org>

Maturity: Prototype software based on early versions of this document.

Coverage: The bulk of this specification is implemented. The use of DTLS functionality is not planned.

Licensing: BSD

Implementation experience: Implementers have read and commented on this document.

7. Security Considerations

One purpose of the mechanism described in the current 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 \[RFC6973\]](#). Also, [Section 6 of \[RFC7525\]](#) contains a detailed discussion of technologies used in conjunction

with TLS. Implementers should familiarize themselves with these materials.

7.1. Limitations of an Opportunistic Approach

The purpose of using an explicitly opportunistic approach is to enable interoperability with implementations that do not support RPC-over-TLS. A range of options is allowed by this approach, from "no peer authentication or encryption" to "server-only authentication with encryption" to "mutual authentication with encryption". The actual security level may indeed be selected based on policy and without user intervention.

In cases where interoperability is a priority, the security benefits of TLS are partially or entirely waived. Implementations of the mechanism described in the current document must take care to accurately represent to all RPC consumers the level of security that is actually in effect. Implementations are REQUIRED to provide an audit log of RPC-over-TLS security mode selection.

7.1.1. STRIPTLS Attacks

A classic form of attack on network protocols that initiate an association in plain-text to discover support for TLS is a man-in-the-middle that alters the plain-text handshake to make it appear as though TLS support is not available on one or both peers. Clients implementers can choose from the following to mitigate STRIPTLS attacks:

- o A TLSA record [[RFC6698](#)] can alert clients that TLS is expected to work, and provide a binding of hostname to x.509 identity. If TLS cannot be negotiated or authentication fails, the client disconnects and reports the problem.
- o Client security policy can require that a TLS session is established on every connection. If an attacker spoofs the handshake, the client disconnects and reports the problem. If TLSA records are not available, this approach is strongly encouraged.

7.2. TLS Identity Management on Clients

The goal of the RPC-on-TLS protocol extension is to hide the content of RPC requests while they are in transit. The RPC-on-TLS protocol by itself cannot protect against exposure of a user's RPC requests to other users on the same client.

Moreover, client implementations are free to transmit RPC requests for more than one RPC user using the same TLS session. Depending on the details of the client RPC implementation, this means that the client's TLS identity material is potentially visible to every RPC user that shares a TLS session. Privileged users may also be able to access this TLS identity.

As a result, client implementations need to carefully segregate TLS identity material so that local access to it is restricted to only the local users that are authorized to perform operations on the remote RPC server.

7.3. Security Considerations for AUTH_SYS on TLS

Using a TLS-protected transport when the AUTH_SYS authentication flavor is in use addresses several longstanding weaknesses (as detailed in [Appendix A](#)). TLS augments AUTH_SYS by providing both integrity protection and a privacy service that AUTH_SYS lacks. TLS protects data payloads, RPC headers, and user identities against monitoring and alteration while in transit. TLS guards against the insertion or deletion of messages, thus also ensuring the integrity of the message stream between RPC client and server. Lastly, transport layer encryption plus peer authentication protects receiving XDR decoders from deserializing untrusted data, a common coding vulnerability.

The use of TLS enables strong authentication of the communicating RPC peers, providing a degree of non-repudiation. When AUTH_SYS is used with TLS, but the RPC client is unauthenticated, the RPC server still acts on RPC requests for which there is no trustworthy authentication. In-transit traffic is protected, but the RPC client itself can still misrepresent user identity without server detection. TLS without authentication is an improvement from AUTH_SYS without encryption, but it leaves a critical security exposure.

In light of the above, it is RECOMMENDED that when AUTH_SYS is used, every RPC client should present host authentication material to RPC servers to prove that the client is a known one. The server can then determine whether the UIDs and GIDs in AUTH_SYS requests from that client can be accepted.

The use of TLS does not enable RPC clients to detect compromise that leads to the impersonation of RPC users. Also, there continues to be a requirement that the mapping of 32-bit user and group ID values to user identities is the same on both the RPC client and server.

7.4. Best Security Policy Practices

RPC-over-TLS implementations and deployments are strongly encouraged to adhere to the following policies to achieve the strongest possible security with RPC-over-TLS.

- o When using AUTH_NULL or AUTH_SYS, both peers are required to have DNS TLSA records and certificate material, and a policy that requires mutual peer authentication and rejection of a connection when host authentication fails.
- o When using RPCSEC_GSS, GSS/Kerberos provides adequate host authentication and a policy that requires GSS mutual authentication and rejection of a connection when host authentication fails. GSS integrity and privacy services, therefore, can be disabled in favor of TLS encryption with peer authentication.

8. IANA Considerations

Following [Section 6 of \[RFC7301\]](#), the authors request the allocation of 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:
RFC-TBD

9. References

9.1. Normative References

- [FIPS.180-4]
National Institute of Standards and Technology, "Secure Hash Standard, Federal Information Processing Standards Publication FIPS PUB 180-4", FIPS PUB 180-4, August 2015.
- [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>>.

- [RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", [RFC 4279](#), DOI 10.17487/RFC4279, December 2005, <<https://www.rfc-editor.org/info/rfc4279>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 5280](#), DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [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>>.
- [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](#), DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [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>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", [BCP 205](#), [RFC 7942](#), DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [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>>.

9.2. Informative References

- [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>>.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", [RFC 6698](#), DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [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>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", [RFC 7435](#), DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.
- [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>>.
- [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>>.

9.3. URIs

[1] <https://www.linuxjournal.com/content/encrypting-nfsv4-stunnel-tls>

Appendix A. Known Weaknesses of the AUTH_SYS Authentication Flavor

The ONC RPC protocol, as specified in [RFC5531], provides several modes of security, traditionally referred to as "authentication flavors". Some of these flavors provide much more than an authentication service. We refer to these as authentication flavors, security flavors, or simply, flavors. One of the earliest and most basic flavors is AUTH_SYS, also known as AUTH_UNIX. [Appendix A of \[RFC5531\]](#) specifies AUTH_SYS.

AUTH_SYS assumes that the RPC client and server both use POSIX-style user and group identifiers (each user and group can be distinctly represented as a 32-bit unsigned integer). It also assumes that the client and server both use the same mapping of user and group to an integer. One user ID, one primary group ID, and up to 16 supplemental group IDs are associated with each RPC request. The combination of these identifies the entity on the client that is making the request.

A string identifies peers (hosts) in each RPC request. [RFC5531] does not specify any requirements for this string other than that is no longer than 255 octets. It does not have to be the same from request to request. Also, it does not have to match the DNS hostname of the sending host. For these reasons, even though most implementations fill in their hostname in this field, receivers typically ignore its content.

[Appendix A of \[RFC5531\]](#) contains a brief explanation of security considerations:

It should be noted that use of this flavor of authentication does not guarantee any security for the users or providers of a service, in itself. The authentication provided by this scheme can be considered legitimate only when applications using this scheme and the network can be secured externally, and privileged transport addresses are used for the communicating end-points (an example of this is the use of privileged TCP/UDP ports in UNIX systems -- note that not all systems enforce privileged transport address mechanisms).

It should be clear, therefore, that AUTH_SYS by itself offers little to no communication security:

1. It does not protect the privacy or integrity of RPC requests, users, or payloads, relying instead on "external" security.
2. It does not provide authentication of RPC peer machines, other than inclusion of an unprotected domain name.
3. The use of 32-bit unsigned integers as user and group identifiers is problematic because these data types are not cryptographically signed or otherwise verified by any authority.
4. Because the user and group ID fields are not integrity-protected, AUTH_SYS does not provide non-repudiation.

Acknowledgments

Special mention goes to Charles Fisher, author of "Encrypting NFSv4 with Stunnel TLS" [[1](#)]. His article inspired the mechanism described in this document.

Many thanks to Tigran Mkrtchyan for his work on the DESY prototype and his feedback on the current document.

Thanks to Derrell Piper for numerous suggestions that improved both this simple mechanism and the current document's security-related discussion.

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

Lastly, special thanks to document shepherd David Noveck, Transport Area Director Magnus Westerlund, 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