N/A	T. Keiser	
Internet-Draft	Sine Nomine	
Intended status: BCP	April 17, 2010	
Expires: October 19, 2010		

TOC

Rx Security Object Providing Cleartext Peer Identity Assertions
draft-tkeiser-rxrpc-sec-clear-02

Abstract

RxRPC was originally designed as the remote procedure call layer for AFS-3. Today there are a number of anonymous RxRPC applications which require identity assertions in order to ensure that the desired peer receives and processes a procedure call. This memo defines a replacement for the rxnull security class which provides a means for mutually agreeing upon who is communicating, without incurring cryptographic overhead. It should be noted that, much like rxnull, this security object is not suitable for use in a distributed environment due to its inability to provide integrity protection.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”
This Internet-Draft will expire on October 19, 2010.

Copyright Notice

Table of Contents

1. Introduction

TOC

RxRPC [draft-zeldovich-rx-spec] (Zeldovich, N. and M. Meffie, “Rx protocol (work in progress),” November 2009.) is a remote procedure call (RPC) protocol that evolved from earlier prototypes developed as part of the Andrew Project at Carnegie Mellon University [VICE1] (Satyanarayanan, M., Howard, J., Nichols, D., Sidebotham, R., Spector, A., and M. West, “The ITC Distributed File System: Principles and Design,” December 1985.) [CMU-ITC-85-003] (Satyanarayanan, M., “A Large-Parameter Remote Procedure Call Mechanism in Unix,” 1985.) [CMU-ITC-84-011] (Satyanarayanan, M., “RPC User Manual,” January 1985.) [CMU-ITC-85-038] (Satyanarayanan, M., “RPC2 User Manual,” 1985.). Its primary, although notably not its only, usage is by the AFS-3 distributed file system [AFS1] (Howard, J., “An Overview of the Andrew File System",” February 1988.) [AFS2] (Howard, J., Kazar, M., Menees, S., Nichols, D., Satyanarayanan, M., Sidebotham, R., and M. West, “Scale and Performance in a Distributed File System,” February 1988.). Rx provides remote procedure call services over top of multiplexed stateful virtual circuits called "connections". Individual call sessions within the multiplexed circuits are called "channels". Flow control, delivery guarantees, and security are provided at the connection level. Stream ordering is performed at the channel level. Security in Rx is handled at the connection level. Thus, all calls within a given multiplexed connection must be associated with the same security object, which in all current use cases means the same security context. Security mechanisms in Rx are pluggable -- the Rx packet header contains a single octet field which defines the security mechanism to be used. Rx packet payload encoding is under the control of the mutually agreed upon Rx security mechanism.

1.1. Existing Security Mechanisms

TOC

At present, there are two Rx security mechanisms in wide deployment: rxnull, and rxkad. Additionally, there was a security mechanism called rxvab, which was used by early VICE prototypes, never widely deployed, and considered to be entirely deprecated.

TOC

1.1.1. rxnull

As the name implies, rxnull provides no-op security services for anonymous services. Rxnull does not modify the packet payload in any manner. Absolutely no cryptography is used with rxnull; header fields are asserted to be correct.

1.1.2. rxkad

TOC

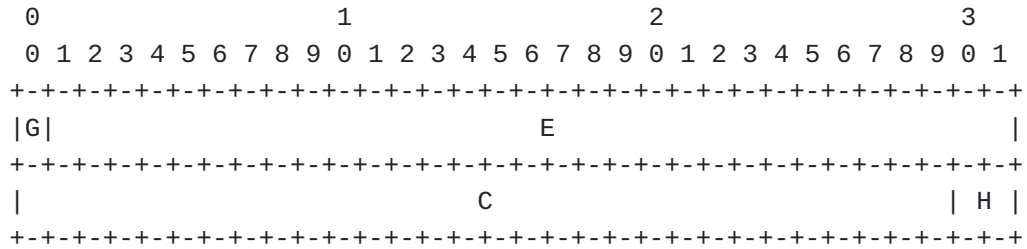
Rxkad was originally developed as a kerberos 4-based security mechanism implementing three security profiles: header integrity protection, payload integrity protection, and payload encryption. With the advent of Kerberos 5 [RFC4120] (Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)," July 2005.), the rxkad mechanism was extended to support DES Kerberos 5 tickets. Rxkad utilizes a spare 16-bit Rx header field to store a 16-bit cryptographic checksum of a bit-string called the packet pseudoheader. The pseudoheader contains most Rx header fields, as well as certain other ancillary pieces of data [AFS-RX] (Zayas, E., "AFS-3 Programmer's Reference: Specification for the Rx Remote Procedure Call Facility," August 1991.).

1.2. Multi-homed Host Support

TOC

During the 1990s Transarc extended the Rx protocol to support multihomed hosts. The Transarc design involved redefining the most significant bit of the packet header connection identifier field. Under the new design, the connection ID field was split as follows:

Rx Header Epoch and Connection ID Fields



G bit: 1 bit

When asserted, the (G)lobal bit indicates that the C field is globally unique. When not asserted, the tuple (IPv4 address, UDP port, E, C) is used to identify an Rx connection. However, when multi-homed hosts are involved in a connection, the

Global bit causes the C field to become globally unique, and thus IP address and port number matching is not performed as part of the virtual circuit identification process.

E bits: 31 bits (unsigned integer)
The (E)poch field is used to detect peer Rx state resets. Whenever an Rx protocol stack is initialized, an effort should be made to assigned it a different value. Typically, this is done by assigning the current Unix epoch time.

C bits: 30 bits (unsigned integer)
The (C)onnection bits are part of the virtual circuit identifier. As discussed above, the G bit controls what other date is used as part of the virtual circuit identifier.

H bits: 2 bits (unsigned integer)
The c(H)annel bits are used to multiplex four RPC call channels over a single Rx connection. Each packet is thus associated with a specific channel.

1.3. Motivations

TOC

IPv4 address renumbering is a frequent occurence in many environments. Due to the stateless nature of the Rx packet multiplexor, it is possible for race conditions to occur whereby an RPC call payload is delivered to the wrong peer. With the existing Rx security classes, the receiving peer will automatically create a new Rx connection, optionally go through a challenge/response phase, and then proceed to process the call arguments. Obviously, mis-delivery of an RPC call can result in incorrect behavior. For example, in the case of AFS-3, mis-delivery can lead to data corruption, loss of cache coherence, and other problematic situations.

1.4. Goals

TOC

Many high-performance applications based upon Rx RPC cannot tolerate cryptographic overhead. In order to ensure correctness in the face of transport-layer address renumbering, some form of context needs to be established between client and server to permit upper-layer applications to reject processing of remote procedure calls that were misdirected. This memo aims to replace rxnull with a minimally-

intrusive security object that provides a stateful means of detecting address renumbering events without introducing cryptographic overhead. Obviously, similar race conditions can occur with the rxkad security object. Solving that problem is considered outside the scope of this memo.

---

2. Conventions

TOC

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 (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.) [RFC2119].

---

3. Overview of Rx RPC

TOC

Rx RPC is a remote procedure call mechanism built on top of UDP. In order to establish a stateful call context on top of a stateless datagram protocol, Rx relies upon a number of client-asserted header fields to establish a flow-controlled communications channel between peers. To eliminate the need for context-establishment round-trips, Rx relies upon client assertions to establish a stateful context.

3.1. Packet Mux

TOC

Rx connection objects are identified by a tuple of packet header fields. The most important control field is the most-significant bit of the epoch header field. When this bit is asserted, the connection object is operating in multi-homing mode, as specified in [draft-zeldovich-rx-spec] (Zeldovich, N. and M. Meffie, "Rx protocol (work in progress)," November 2009.). In the normal Rx operating mode (with the multi-homing bit set to zero), Rx connections are identified by the following tuple: (host, port, epoch, cid), where these elemenets are defined as:

host: IPv4 address of peer

port: UDP port of peer

epoch: Rx header epoch field

cid: Rx header cid field (channel ID bits masked to zero)

However, when the multi-homing bit is asserted, the connection identifier tuple becomes: (epoch, cid). Thus, multi-homed Rx connection objects have a shared (epoch, cid) namespace, independent of peer address.

---

4. Presenting Problems

TOC

The design of this Rx security class is motivated by server and client renumbering incidents at large AFS-3 deployments. When a file server is renumbered, there is a several hour window until the next VL_GetAddrsU RPC is performed to refresh the file server UUID to IPv4 address mappings in the client. Due to the TTL-based invalidation of stale cached mappings, there is a substantial time interval during which RPCs can be delivered to the wrong file server, potentially leading to incorrect behavior.

Similarly, client renumbering can lead to incorrect behavior due to a loss of cache coherence. The AFS-3 callback mechanism relies upon correct knowledge of client UUID to IPv4 address mappings in order to deliver cache invalidation messages to clients. When these mappings become stale due to intervening address renumbering events, advertisement of incorrect addresses, NATs, etc. these "call back" remote procedure calls may be delivered to the wrong client node. In some circumstances this can lead to false state of success on the file server because an unintended client received, processed, and sent a response of success to the file server. Due to the success return code, the file server will no longer attempt to deliver the invalidation, and the client to which the call back was supposed to be delivered will continue to operate on stale cached data because it never received the cache invalidation message.

4.1. Node Renumbering

TOC

When servers are renumbered, one potential outcome is that two or more machines running the same service will swap addresses. In this case, there is a possibility for the wrong machine to correctly interpret, and attempt to execute, a procedure call.

In some cases, execution of an RPC by the wrong endpoint will still result in correct behavior. However, this is not generally true, where execution by an unintended target could result in undefined, or even dangerous, behavior. For example, in AFS-3, the existence of shadow clones could result in a situation where an RW shadow clone is updated instead of the canonical RW site registered in the VLDB.

4.2. Epoch ID Multi-Homing Bit

TOC

When the multi-homing bit is asserted, (connection,epoch) tuples become globally unique. This mode of operation permits clients to contact the server on multiple addresses, thus allowing client operating systems to route datagrams as desired. Current implementations of Rx bind the connection to the first peer address on which a datagram was received. Since all reply datagrams are sent to the bound peer, connection hijacking becomes impossible. Unfortunately, this comes at the expense of handling client renumbering events: when the bound peer address becomes unreachable, or is reassigned, the Rx connection enters a simplex state and consequently all call channels block until the connection times out.

4.3. Processing of Non-Idempotent Calls

TOC

Another problematic symptom of Rx connections entering a simplex state involves non-idempotent RPCs. The core problem is that by the time the lack of a reply channel is observed, the non-idempotent procedure call has already been executed. Rx RPCs are not generally transactional, and thus there is typically no means of rolling back the state-changing behavior. Obviously, this problem is not unique to multihomed hosts, but it is another indication of how Rx is lacking compared to alternative multi-home aware protocols, such as SCTP [RFC4960] (Stewart, R., “Stream Control Transmission Protocol,” September 2007.).

---

5. Rx Clear Security Class

TOC

In order to overcome the dangers inherent in assuming stability of transport addresses, the Rx Clear security class embeds a security header in all data packets. This security header contains application-specific endpoint identifier assertions for both the source and destination.
When a datagram is received by the wrong peer, an Rx abort packet will be dispatched notifying the peer of the need to re-bind transport addresses for this connection object. When such an abort packet is received by a client connection, the error will be immediately propagated back to the caller so that application-specific logic may be invoked to refresh transport-layer address mappings for the intended destination endpoint. In the server case, this memo standardizes new multi-homing Rx connection peer binding semantics which allow for graceful handling of client renumbering events.

5.1. Constants

TOC

The Rx Clear security class makes use of several newly defined constants, which are defined below:

RX_SEC_ID_CLEAR:
An Rx security index will be allocated by the Grand Central Registrar. As with all Rx security indices, this 8 bit integer will uniquely identify the security class bound to a given Rx datagram.

RXCL_HDR_VERS_1:
Rx Clear security header version 1 will be allocated by the Grand Central Registrar. This version number will correspond to the XDR [RFC4506] (Eisler, M., "XDR: External Data Representation Standard," May 2006.) encoded data structure called rxClear_Header, as specified in Section 5.2 (Security Header).

RXCL_ERR_UNKNOWN_VERS:
An Rx error code will be allocated which communicates that this version of the Rx Clear security header is unsupported by the peer. This error code will be sent as the user payload of an Rx abort packet.

RXCL_ERR_UNKNOWN_ID_TYPE:
An Rx error code will be allocated which communicates that this endpoint identifier type is not supported by the peer. This error code will be sent as the user payload of an Rx abort packet.

RXCL_ERR_WRONG_PEER:
An Rx error code will be allocated which communicates mis-delivery of an Rx Clear-protected datagram to the wrong peer. This error code will be sent as the user payload of an Rx abort packet.

RXCL_ERR_XCID_UNSUPP:
An Rx error code will be allocated which communicates to the peer that this node is incapable of supporting the extended connection id field. This error code will be sent as the user payload of an Rx abort packet upon receipt of an RxClear header containing a non-zero clh_xcid field by a node which cannot support extended connection identifiers.

RXCL_EI_TYPE_NULL:
An endpoint identifier type which provides fallback to rxnull-like semantics. In other words, the contents

> of the source and destination endpoint identifiers have no meaning (and SHOULD thus be zero octets in length). In this mode of operation, detection of address renumbering is impossible.

5.2. Security Header

TOC

In order to communicate expectations to the peer, all data packets travelling over an RxClear-protected connection will include an XDR-encoded security header which carries identity assertions. The RxClear mechanism uses a header rather than a challenge- response mechanism because the additional round-trips required by the Rx challenge-response mechanism were deemed too costly for the typical unauthenticated Rx call workload.
The proposed security header is an XDR-encoded structure defined as follows:

```
struct rxClear_Header {
    u_char clh_version;          /* authenticator version number */
    u_char clh_id_type;          /* how to interpret opaque peer
                                    identifier payloads */
    u_char clh_data_off;         /* data payload offset */
    u_char clh_spares1;          /* MUST be set to zero */
    afs_uint32 clh_data_len;     /* data payload length */
    afs_uint32 clh_trl_off;      /* security trailer offset */
    afs_uint32 clh_flags;        /* miscellaneous control flags */
    afs_uint32 clh_spares2;      /* MUST be set to zero */
    afs_uint32 clh_spares3;      /* MUST be set to zero */
    opaque clh_src_id;           /* assertion of client identity */
    opaque clh_dst_id;           /* assertion of server identity */
};
```

Rx Clear Security Header

Figure 1

This security header will be an XDR-encoded data structure, which will occupy the first octets of the data offset in an Rx packet -- it will start at the offset directly following the Rx packet header. The normal packet data will begin at the data offset specified in the clh_data_off field of the security header.

> **clh_version: 8-bits (unsigned integer)**

This contains the version of the Rx Clear security object header. If this version is unknown by the peer, then the connection must be aborted.

clh_id_type: 8-bits (unsigned integer)

An 8-bit unsigned integer which identifies the encoding of the XDR opaque fields src_id and dst_id. Values within this 8-bit namespace are allocated by the AFS Assigned Numbers Registrar.

clh_data_off: 8-bits (unsigned integer)

This field specifies the beginning of the data payload, in units of octets from the beginning of the Rx packet payload. This field is used by receivers to determine where to begin reading the encapsulated data payload.

clh_trl_off: 32-bits (unsigned integer)

This value specifies the offset in octets of the clear security class packet trailer. A value of zero indicates the absence of a security trailer.

clh_flags: 32-bits (unsigned integer)

This is a bitfield whose bits are used as protocol control flags. All flag bits whose semantics are not yet standardized MUST be sent as zeroes.

clh_src_id: XDR opaque

This field contains an application-specific source endpoint identifier. For example, in the case of AFS-3, this will likely be an XDR-encoded node UUID.

clh_dst_id: XDR opaque

This field contains an application-specific destination endpoint identifier. For example, in the case of AFS-3, this will likely be a XDR-encoded node UUID.

clh_spares1: 8-bits (unsigned integer)

This field is reserved for future use, and MUST be set to zero. Future memos MAY define a standardized use for this field, and thus implementors MUST NOT make private use of this field.

clh_spares2: 32-bits (unsigned integer)

This field is reserved for future use, and MUST be set to zero. Future memos MAY define a standardized use for this field, and thus implementors MUST NOT make private use of this field.

clh_spares3: 32-bits (unsigned integer)

> This field is reserved for future use, and MUST be set to zero. Future memos MAY define a standardized use for this field, and thus implementors MUST NOT make private use of this field.

5.3. Data Packet Validation

TOC

Upon receipt of a data packet with the security index set to RX_SEC_ID_CLEAR, the node will XDR decode the security header, and subsequently validate the security header. Following XDR decode, the node shall first verify that the clh_version field contains a supported version number. In the event that the node does not support this RxClear version, the node will send an Rx abort packet to the peer with error code RXCL_ERR_UNKNOWN_VERS.

The second step in validation involves the extended connection identifier field, clh_xcid. If this node does not support extended cid, and the clh_xcid field is non-zero, then an abort packet with user payload RXCL_ERR_XCID_UNSUPP should be sent to the peer, and the connection should transition to an error state.

Next, the application-specific endpoint identifier type specified in clh_id_type field is validated to ensure that the application layer can handle this identifier type. If this endpoint identifier type is not supported by the application layer, then the node will send an Rx abort packet with user payload of RXCL_ERR_UNKNOWN_ID_TYPE, and the connection should transition to an error state.

The application layer will then be asked to validate the clh_dst_id field. If there is a mismatch, an abort packet will be sent to the peer with user payload RXCL_ERR_WRONG_PEER, and the Rx connection will then transition into an error state.

5.4. Abort Packet Handling

TOC

Processing of received Rx Abort packets must be updated to handle the new RXCL_ERR_ error codes. If such an error code is received on a connection with security index other than RX_SEC_ID_CLEAR, then behavior is undefined.

TOC

5.4.1. RXCL_ERR_UNKNOWN_VERS

This error code indicates that the peer is unable to support the version of the RxClear security header sent in a packet. The connection is transitioned into an error state.

5.4.2. RXCL_ERR_UNKNOWN_ID_TYPE

TOC

This error code indicates that the peer is unable to support this application-specific endpoint identifier type. The connection is transitioned into an error state.

5.4.3. RXCL_ERR_WRONG_PEER

TOC

This error code indicates that the packet was delivered to the wrong peer. Behavior in this situation depends on several factors. First, for connections where the epoch multi-homing bit is zero, the connection must be transitioned to an error state. For multi-homed connections, behavior further depends upon whether this is a client connection, or a server connection. For client connections, the easiest course of action is to set the connection to an error state, and allow the client to re-resolve the application-specific endpoint-identifier to transport identifier mapping, allocate a new Rx connection, and re-try the call. In the case of a multi-homed server connection, the implementation SHOULD make a best-effort try to deliver the call reply data to the correct destination, as this may be a non-idempotent procedure call. This memo outlines in detail new peer binding semantics for multi-homed Rx connections in another section. Hence, whenever it is possible, the server will not transition a server connection into an error state upon receipt of this message. Instead, it SHOULD invalidate the peer currently bound to the connection so that future replies go to a different, hopefully correct, transport address.

5.4.4. RXCL_ERR_XCID_UNUSPP

TOC

This error code indicates that the peer is unable to support the extended connection identifier field in the RxClear security header. The connection is transitioned to error state, and the implementation SHOULD mark the peer as being incapable of supporting extended connection identifiers so that connections allocated to this peer in the future contain a clh_xcid field with value zero.

6. Multi-Home Behavior

TOC

Rx supports multi-homed clients through the assertion of the most-significant bit in the Rx header epoch field. When this bit is asserted, a server will accept datagrams into a connection regardless of the source host address and port. However, reply packets are always sent to the first peer address which contacted the server on any given (epoch, cid) tuple. This behavior prevents connection hijacking, at the expense of robust multi-homing support.

In order to properly support multi-homing this memo specifies relaxation of the peer binding policies. Most importantly, upon receipt of an RXCL_ERR_WRONG_PEER abort packet, an Rx server should not transition a server-mode connection to an error state. Rather, it SHOULD mark the peer currently bound to the Rx connection as being incorrect so that responses may be sent to a different peer, as determined upon receipt of the next ping packet. Although this does open up room for connection hijacking, it does so only for anonymous connections, which are otherwise exposed to denial of service attacks. To address the issue of lack of response, new Rx server implementations SHOULD permit re-binding of the peer on server-mode connections. To this end, servers should track liveness of peer addresses on a server connection in order to remove a dead peer from a connection. If an Rx ping comes from an address other than the currently bound peer transport address, the Rx implementation MAY try to re-send unacknowledged packets to this other address. If these re-transmits are correctly aknowledged, the connection may be re-bound to the new peer.

7. Acknowledgements

TOC

I would like to thank all of the participants at the 2009 Edinburgh AFS hackathon for their input into the design of this security mechanism. Specifically, I would like to thank Jeffrey Altman for suggesting that it would be architecturally cleaner to place peer identity assertions into a security header, rather than modifying AFS-3 RPCs to explicitly include application-layer identity assertions as IN parameters. Special thanks to Randall Atkinson for providing useful feedback with regard to the introduction and security considerations sections.

TOC

8. IANA Considerations

This memo includes no request to IANA.

---

9. AFS Assigned Numbers Registrar Considerations

TOC

This memo includes several assigned numbers requests which must be considered by the AFS Assigned Numbers Registrar.

9.1. Definition of new registries

TOC

This memo requests that the AFS Assigned Numbers Registrar allocate new registries for three fields in the Rx Clear security header, as described in Section 5.2 (Security Header):

- *clh_version
- *clh_id_type
- *clh_flags

Allocations for the following registries are to be processed by the AFS Assigned Numbers Regisrar pursuant to the policies dictated in sections 2.3.2 and 2.3.3 of [AFS3-STDS-CHARTER] (Wilkinson, S., “Options for AFS Standardisation,” September 2008.).

9.1.1. clh_version

TOC

The Rx Clear security class includes a version number in its packet header. This memo requests that the AFS Assigned Numbers Registrar allocate a new registry for tracking assigned values for this protocol field. The unsigned 8-bit namespace for this registry shall be divided into three regions as follows:

Range	Policy
0-239	standards
240-254	private use
255	reserved

Standards-track allocation requests for this registry MUST contain the following pieces of information:

*A reference to an RFC section documenting the XDR definition of the header

In addition, a standards-track allocation request MAY contain the following optional elements:

*A reference to an RFC section documenting security considerations for this header type

*A requested version number

9.1.2. clh_id_type

TOC

The Rx Clear security class provides a means of sending opaque application data, which is intended to provide a means of transmitting application-specific transport-independent endpoint identifiers. The 8-bit unsigned namespace for this registry shall be divided into three policy regions as follows:

Range	Policy
0-239	standards
240-254	private use
255	reserved

Standards-track allocation requests for this registry MUST contain the following pieces of information:

*A reference to an RFC section documenting the means of encoding the payload, such as an XDR type definition

*A reference to an RFC section documenting the semantics for this encoding type

In addition, a standards-track allocation request MAY contain the following optional elements:

*A reference to an RFC section documenting security considerations for this encoding type

*A requested type identification number

TOC

9.1.3. clh_flags

The Rx Clear security class header contains a 32-bit flags bit vector. Bits within this vector shall be allocated by the AFS Assigned Numbers Registrar. This 32-value namespace shall be subdivided into two policy regions as follows:

Range	Policy
2^0 to 2^27	standards
2^28 to 2^31	private use

Standards-track allocation requests for this registry MUST contain the following pieces of information:

- A reference to an RFC section documenting the semantics for this flag

In addition, a standards-track allocation request MAY contain the following optional elements:

- A reference to an RFC section documenting security considerations for this flag
- A requested bit position

9.2. Allocation of new values

TOC

This memo also makes several allocation requests to the AFS Assigned Numbers Registrar.

9.2.1. RxClear security index

TOC

A new Rx protocol security index must be allocated. It is anticipated that given the small size of the security index namepsace, the allocation will only be satisfied after rough consensus is established on the afs3-standardization@openafs.org mailing list.

TOC

9.2.2. Rx error codes

The Rx Clear security class allocates several new Rx error codes for use in Rx abort packet payloads. Given that there are multiple Rx implementations, it is assumed that the AFS Assigned Numbers Registrar will be responsible for allocating new error table values. Specifically, the following new Rx error codes need to be allocated:

*RXCL_ERR_UNKNOWN_VERS

*RXCL_ERR_UNKNOWN_ID_TYPE

*RXCL_ERR_WRONG_PEER

*RXCL_ERR_XCID_UNSUPP

Please see Section 5.1 (Constants) for further details regarding these constants.

9.2.3. Rx Clear Security Header Version

TOC

This memo requests allocation of version 1 within this new namespace for the protocol header described in Section 5.2 (Security Header).

9.2.4. Endpoint Identifier Type

TOC

One endpoint type identifier is requested at this time: RXCL_EI_TYPE_NULL. The null endpoint identifier type shall have encoding and semantics as defined in Section 5.1 (Constants)

---

10. Security Considerations

TOC

This protocol explicitly provides neither the means for encrypting nor integrity checking the contents of Rx headers or payloads. Its use, except in physically secured and isolated high-performance computing environments where cryptographic overhead is deemed to be unacceptable, is NOT RECOMMENDED. Where use on the internet is necessary, other means of protecting the Rx protocol from attack, such as IPsec [RFC4301] (Kent, S. and K. Seo, “Security Architecture for the Internet Protocol,” December 2005.) [RFC4302] (Kent, S., “IP Authentication Header,” December 2005.) [RFC4303] (Kent, S., “IP Encapsulating Security Payload (ESP),” December 2005.) are RECOMMENDED. It should be

noted that, due to its use of UDP as a transport, Rx is not a candidate for encapsulation within TLS [RFC5246] (Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," August 2008.).
Rx without packet header integrity, at a minimum, is open to a call injection attack. The following section outlines describes this known attack vector, and how the introduction of the Rx Clear security class changes the susceptibility of Rx to this attack.

10.1. call injection

TOC

With rxnull, simplex injection attacks have always been possible. For connections with the G bit (as described in Section 1.2 (Multi-homed Host Support)) asserted, this means that the attacker must correctly spoof the epoch and connection ID. For injection of data packets into a channel, one further piece of information must be available: the range of packet sequence numbers currently within the valid receive window. Much of this information is obtainable by probing the victim with Rx protocol debugging packets.
The Rx Clear security class changes the nature of this attack. If an attacker, in addition to the information above, also posesses the correct endpoint identifiers for the two peers, it may hijack the Rx connection. The key difference here is that the attack is full-duplex; all replies will now flow to the attacker instead of the original recipient.
Given that both rxnull and Rx Clear are unauthenticated security objects, this is unlikely to result in anything more severe than a denial of service. Furthermore, given the new Rx Clear abort codes, the peer will detect this situation one round trip after transmission of its next call, rather than ending up in a state where the two peers disagree on call channel window position, which is complex to detect and resolve.

---

11. References

TOC

11.1. Normative References

TOC

[AFS3-STDS-CHARTER]	Wilkinson, S., "Options for AFS Standardisation," September 2008.
[RFC2119]	

	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).

11.2. Informative References

TOC

[AFS-RX]	Zayas, E., “AFS-3 Programmer's Reference: Specification for the Rx Remote Procedure Call Facility,” Transarc Corp. Tech. Rep. FS-00-D164, August 1991.
[AFS1]	Howard, J., “An Overview of the Andrew File System",” Proc. 1988 USENIX Winter Tech. Conf. pp. 23-26, CMU ITC Tech. Rep. CMU-ITC-88-062, February 1988.
[AFS2]	Howard, J., Kazar, M., Menees, S., Nichols, D., Satyanarayanan, M., Sidebotham, R., and M. West, “Scale and Performance in a Distributed File System,” ACM Trans. Comp. Sys. Vol. 6, No. 1, pp. 51-81, CMU ITC Tech. Rep. CMU-ITC-87-068, February 1988.
[CMU-ITC-84-011]	Satyanarayanan, M., “RPC User Manual,” CMU ITC Tech. Rep. CMU-ITC-84-011, January 1985.
[CMU-ITC-85-003]	Satyanarayanan, M., “A Large-Parameter Remote Procedure Call Mechanism in Unix,” CMU ITC Tech. Rep. CMU-ITC-85-003, 1985.
[CMU-ITC-85-038]	Satyanarayanan, M., “RPC2 User Manual,” CMU ITC Tech. Rep. CMU-ITC-85-038, 1985.
[RFC4120]	Neuman, C., Yu, T., Hartman, S., and K. Raeburn, “The Kerberos Network Authentication Service (V5),” RFC 4120, July 2005 (TXT).
[RFC4301]	Kent, S. and K. Seo, “Security Architecture for the Internet Protocol,” RFC 4301, December 2005 (TXT).
[RFC4302]	Kent, S., “IP Authentication Header,” RFC 4302, December 2005 (TXT).
[RFC4303]	Kent, S., “IP Encapsulating Security Payload (ESP),” RFC 4303, December 2005 (TXT).
[RFC4506]	Eisler, M., “XDR: External Data Representation Standard,” STD 67, RFC 4506, May 2006 (TXT).
[RFC4960]	Stewart, R., “Stream Control Transmission Protocol,” RFC 4960, September 2007 (TXT).
[RFC5246]	Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246, August 2008 (TXT).
[VICE1]	Satyanarayanan, M., Howard, J., Nichols, D., Sidebotham, R., Spector, A., and M. West, “The ITC Distributed File System: Principles and Design,” Proc. 10th ACM Symp. Operating Sys. Princ. Vol. 19, No. 5, CMU ITC Tech. Rep. CMU-ITC-85-039, December 1985.
[draft-zeldovich-rx-spec]	Zeldovich, N. and M. Meffie, “Rx protocol (work in progress),” November 2009.

Author's Address

TOC

	Thomas Keiser
	Sine Nomine Associates
	43596 Blacksmith Square
	Ashburn, VA 20147
	USA
Phone:	+1 703 723 6673
Email:	tkeiser@sinenomine.net