IPsec 鍵交換アーキテクチャ racoon2 の開発

Title: IPsec 鍵交換アーキテクチャ racoon2 の開発
Authors(s):
神田充 (mk@wide.ad.jp)
鎌田健一 (kamda@nanohz.org)
坂根昌一 (sakane@kame.net)
福本淳 (fukumoto@isl.rdc.toshiba.co.jp)
宮澤和紀 (kazunori@miyazawa.org)

Date: 2004/01/27

1. はじめに

IPsec のための鍵交換プロトコルとしては標準の IKE があるが、IETF の IPsec WG では、IKE の改定版である Internet Key Exchange version 2 (IKEv2) の仕様策定が進んでいる。また、IPsec の鍵交換の認証に Kerberos を用いた Kerberized Internet Negotiation Keys (KINK) の策定が KINK WG で行われている。

WIDE IPsec WG では、昨年度に引き続き、これら複数の鍵交換プロトコルを、一つのプラットフォーム上で同時にサポートするためのアーキテクチャの研究を行い、その実装として racoon2 の開発を中心にした活動を行った。

WIDE 2004 年 12 月研究会では、その成果として KINK および IKEv2 の鍵交換のデモを行った。

2. racoon2

racoon2 は、複数の鍵交換プロトコルを BSD 系 OS と Linux で動作させるためのアーキテクチャである。サポートしているプロトコルは IKEv2、KINK であり、今後 Internet Key Exchange (IKEv1) もサポートする予定である。対象とするプラットフォームは、FreeBSD、NetBSD、Linux である。

2.1 アーキテクチャ

racoon2 は、以下の3つのデーモンから構成される。

- spmd　　policy management daemon
- iked　　IKE damon
- kinkd　　KINK daemon

これら3つのデーモンの中で特徴的なのは spmd である。spmd はセキュリティポリシを管理すると同時にリゾルバの役割を果たす。これは、FQDN から IP アドレスへの変換と、その結果をキャッシュし逆変換に用いるためである。この機能は KINK をサポートするために必要となる。

KINK は、認証に Kerberos を用い、そのためには Kerberos のプリンシパル名を必要する。それは以下のように FQDN から導出される。

kink/fqdn@REALM

一方、カーネルは FQDN に関する情報を持たず、IP アドレスによって鍵交換を始動するため、IP アドレスから FQDN のマッピングを行う必要があり、それは、アプリケーションが使用した FQDN であるべきである。このために IP アドレスと FQDN のマッピングを行う spmd を導入した。

racoon2 の実装は、libracoon を用いた以下のようなモジュールからなる。それぞれのデーモンに共通する機能は、libracoon に集約されている。

2.2 設定モデル

racoon2 の設定は以下の5つの情報をリンクしたものを基本とする。それらは

- selector_info
- policy_info
- ipsec_info
- sa_info
- remote_info

である。

selector_info は、Security Policy Database (SPD)の selector に相当する。selector としてアドレスを指定する部分には、FQDN または特定の IP アドレスを示す特殊な文字列を用いることができる。

policy_info は、selector_info からポイントされ selector_info にマッチしたパケットの処理に関する基本的な情報が格納される。ポリシが IPsec であった場合、policy_info は remote_info と ipsec_info をポイントする。

ipsec_info は、IPsec SA の有効期限やモード。トンネルモードの場合には IPsec SA のエンドポイントといった IPsec SA に関する基本的な情報を保持し、sa_info をポイントする。

sa_info は、IPsec SA 固有の情報を保持する。それらは、Authentication Header (AH)や IP Encapsulation Payload (ESP) といった IPsec protocol や暗号アルゴリズムである。静的な設定の場合には、SPI や鍵も含まれる。

remote_info は、鍵交換プロトコルに関する情報を保持する。
基本的な設定は図 2.1 のようになり、それらは各情報のインデックスを用いてリンクされている。

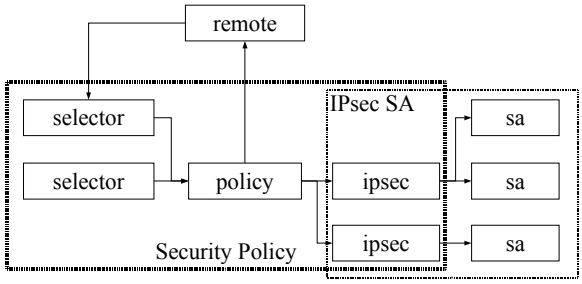

図 2.1 : racoon2 設定モデル

2.3 動作概要

racoon2 の動作は、設定が IP アドレスで行われている場合と FQDN で行われている場合で異なる。

IP アドレスで設定が行われている場合は、図 2.2 に示すように、racoon と同様に spmd によってあらかじめポリシが設定され、それにしたがってカーネルが SADB_ACQUIRE を用いて IPsec SA を要求するという動作になる。

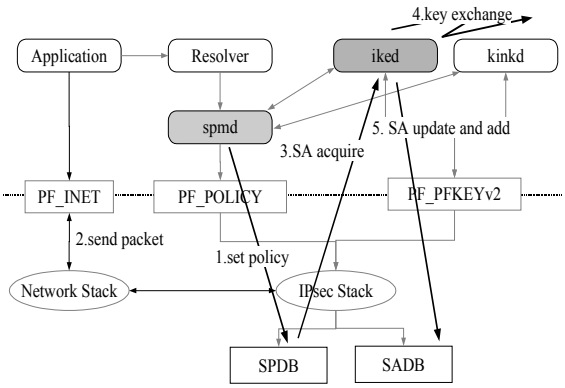

図 *2.2:IP* アドレス設定による *racoon2* の動作

一方、FQDN で設定が行われている場合、図 2.3 のようにアプリケーションが FQDN を解決するのをトリガとして spmd によってポリシが設定または更新される。spmd はアプリケーションからの要求に基づき FQDN を解決し、その結果を元にポリシを設定する。ポリシが設定された後にアプリケーションに結果を返す。アプリケーションは、解決したアドレスに対してパケットを送信し、そのパケットはポリシにマッチしカーネルは SADB_ACQUIRE によって IPsec SA を要求する。SADB_ACQUIRE メッセージには IP アドレスしかないため、FQDN を spmd に問い合わせる。

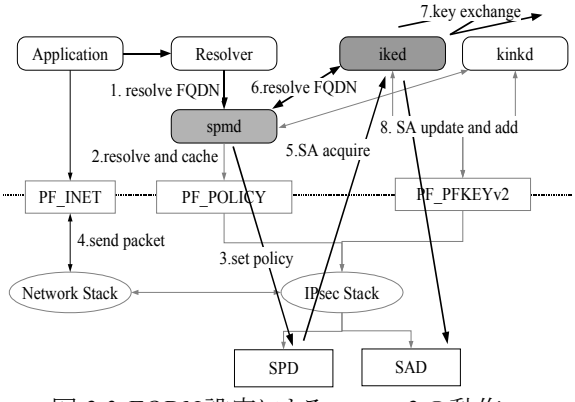

図 *2.3:FQDN* 設定による *racoon2* の動作

2.4 spmd

spmd(Security Policy Management Daemon)は前述の通り racoon2 において 2 つの役割を担う。

一つは racoon2 の設定ファイルに書かれた情報に基き libracoon の API を通して IPsec のセキュリティポリシーを設定する。もう一つは設定ファイル中に FQDN にて書かれたノードのアドレスを解決するための DNS proxy の機能である。(FQDN で記述する必要がなければ DNS proxy の機能はオフにすることもできる。)

2.5 iked

iked は racoon2 の中で IKE (Internet Key Exchange)プロトコルを処理するモジュールであり、単一の Unix デーモンプロセスとして動作する。

主に関連するプロトコルは以下の通り。

- IKE、ISAKMP、IPsec DOI
- IKEv2

そのほかに、libracoon を通じて PF_KEY（RFC2367、KAME/USAGI 拡張）を用いる。

IKEv2 は、IKE を改良したプロトコルで、IKE_SA_INIT,IKE_AUTH のリクエストとレスポンスからなる合計 4 つのメッセージの交換で IPsec SA を設定するプロトコルである。

IKEv2 は、まず IKE_SA_INIT では、2つのノード間の鍵交換を保護するための SA(IKE_SA)の情報と鍵を生成するための Diffie-Hellman のパラーメタを交換する。これによって IKE_SA を確立し、その IKE_SA 上において相互の認証と IPsec SA の確立を行う。

Initiator		***Responder***
IKE_SA_INIT(request)	→	
	←	IKE_SA_INIT(response)
IKE_AUTH(request)	→	
	←	IKE_AUTH(response)

2.6 kinkd

kinkd は racoon2 フレームワーク内で KINK プロトコルを提供する、鍵管理デーモンである。

KINK とははセキュリティープロトコルで用いるパラメータを管理するための、自動鍵管理プロトコルである。現在は IPsec の鍵管理に用いることが想定されているが、IKE と同様、DOI を定義することで IPsec 以外のセキュリティープロトコルにも適用可能となるよう設計されている。

KINK の主な特徴としては以下のものが上げられる。

- 認証機構として Kerberos を利用する。
- プロトコルは基本的に two-way であり、メッセージ交換中の 状態遷移を比較的簡単にできる。(responder が initator の提案に同意しなかった場合のみ、3-way のハンドシェイクが行われる)

これらによって、公開鍵暗号などのコストのかかる処理や、通信のレイテンシを最小限にとどめ、効率的かつ安全に鍵管理を行うことを目標としている。

KINK は、交換するパラメータを表現する部分では、ISAKMP で定義されているペイロードのフォーマットをそのまま利用している。

kinkd は KINK で定義されるメッセージタイプ、ペイロードタイプをすべてサポートし、ISAKMP で定義されいるペイロードタイプのうち、以下のものをサポートする予定である。

- Security Association Payload
- Proposal Payload

- Transform Payload
- Identification Payload
- Nonce Payload
- Notification Payload
- Delete Payload

3. 実装

3.1 spmd

後述する libracoon の API が決まる以前に UDP のみであるが DNS proxy としての機能は動作しており設定した FQDN と実 IP アドレスを内部にキャッシュする機能はできていたが今回 libracoon の API ができあがったことにより libracoon を利用するように改造を行なった。

改造を行なった点としては spmd 起動時に racoon2 設定ファイルを読みこみ初期 IPsec セキュリティポリシーとして値を設定する。

このとき設定ファイル中に FQDN が記述されていたら/etc/nsswitch.conf ファイル等 OS の名前解決設定ファイルを参照し/etc/hosts ファイルなどから静的な FQDN<->IP アドレス設定をキャッシュしかつ DNS の問い合わせパケット DNS サーバに投げ名前解決できた IPsec セキュリティポリシーを設定する。

ただし、現在 libracoon によって定義されているマクロ(MY_IP 等)への対応が不十分であるこれらをきちんと実装する必要がある。

また、現在主に kinkd のみが利用している UNIX ドメインソケットもしくは inet ソケットにより各鍵交換デーモンは spmd を接続し IP アドレスから FQDN 問い合わせ、IPsec セキュリティポリシーの設定依頼などを行なうことができる(spmif)。

3.2 iked

3.2.1 処理フロー

iked の内部処理フローの概略は以下の通り

1. コマンドラインオプション処理
2. コンフィギュレーションファイルを読む（libracoon / rcf_read()）
3. PF_KEY インタフェースライブラリ初期化（libracoon / rcpfk_init()）
4. ISAKMP の UDP ソケット作成
5. 無限ループで、作成したソケットに対して select (2)する

PF_KEY ソケットから SADB_ACQUIRE を受信
1. IKE_SA が存在しなければーとして作成し、
2. IKE_SA から CHILD_SA を initiator として作成する。その後の処理は状態遷移を参照

ISAKMP の UDP ソケットにパケットを受信
1. プロトコルバージョン番号により振り分ける（以下は IKEv2 に関して述べる）
2. IKE_SA が存在するか調べ、
3. 存在する場合は IKE_SA の状態処理ルーチンへ
4. 存在しない場合、パケットが新規リクエストであれば新しい IKE_SA を responder として作成する

IKE_SA と CHILD_SA の状態遷移の概略は以下の通り

IKE_SA initiator

IKEV2_STATE_IDLING
↓
・ー（通常のイニシエートの場合はここで）initiator の CHILD_SA を作成。
・← CHILD_SA が GETSPI_DONE に遷移するのを待つ
↓
・→ responder へ IKE_SA_INIT を送信
↓
IKEV2_STATE_INI_IKE_SA_INIT_SENT
↓
・← responder から IKE_SA_INIT を受信
↓
・→ responder へ IKE_AUTH パケットを送信
↓
IKEV2_STATE_INI_IKE_AUTH_SENT
↓
・← responder から IKE_AUTH を受信
↓
IKEV2_STATE_ESTABLISHED
↓
・← ライフタイム経過、またはエラー発生
↓
IKEV2_STATE_DYING
↓
・← すべての CHILD_SA が回収される
↓
（消滅）

IKE_SA responder

（IKE_SA が存在しない状態から）
↓
・← initiator から IKE_SA_INIT パケットを受信
↓
IKEV2_STATE_IDLING
↓
・→ initiator へ IKE_SA_INIT を送信
↓
IKEV2_STATE_RES_IKE_SA_INIT_SENT
↓
・← initiator から IKE_AUTH を受信
↓
・→ initiator へ IKE_AUTH を送信
↓
IKEV2_STATE_ESTABLISHED
↓
・← ライフタイム経過、またはエラー発生
↓
IKEV2_STATE_DYING
↓
・← すべての CHILD_SA が回収される
↓
（消滅）

CHILD_SA initiator

IKEV2_CHILD_STATE_IDLING
↓
・→ PF_KEY へ SADB_GETSPI リクエスト送出
↓
IKEV2_CHILD_STATE_GETSPI
↓
・← PF_KEY から SADB_GETSPI リプライ受信
↓
IKEV2_CHILD_STATE_GETSPI_DONE
↓
・→ IKE_SA が IDLING 状態ならば IKE_SA ネゴシエーションをスタート、
↓ ESTABLISHED 状態ならば CREATE_CHILD_SA ネゴシエーションをスタート
↓
IKEV2_CHILD_STATE_WAIT_RESPONSE
↓
・← ネゴシエーション終了
・→ PF_KEY へ SADB_ACQUIRE 送出
↓
IKEV2_CHILD_STATE_MATURE
↓
・← ライフタイム経過
↓
IKEV2_CHILD_STATE_EXPIRED
↓
（消滅）

CHILD_SA responder

IKEV2_CHILD_STATE_IDLING
↓
・→ PF_KEY へ SADB_GETSPI 送出
↓
IKEV2_CHILD_STATE_GETSPI
↓
・← PF_KEY から SADB_GETSPI 受信
↓
・→ IKE_SA が RES_IKE_SA_INIT_SENT 状態ならば IKE_SA ネゴシエーションの続き、
↓ ESTABLISHED 状態ならば CREATE_CHILD_SA ネゴシエーションの続き
↓
IKEV2_CHILD_STATE_MATURE
↓
・← ライフタイム経過
↓
IKEV2_CHILD_STATE_EXPIRED
↓
（消滅）

3.2.2 IKEv1 プロトコル実装状況

現在のところ実装されているのは IKEv2 のみであり、IKE (v1) プロトコルは対応していない。

3.2.3 IKEv2 プロトコル実装状況

IKEv2 仕様に対して以下のような制約・未実装部分がある。

- CERT、CERTREQ ペイロード取扱の未実装（受信すると無視またはエラー）
- certificate は x509_subject (ID_DER_ASN1_DN) でしか使えない
- HTTP_CERT_LOOKUP_SUPPORTED、ESP_TFC_PADDING_NOT_SUPPORTED、NON_FIRST_FRAGMENTS_ALSO 対応無し（受信すると無視）
- DSS (Digital Signature Standard)は無し
- ENCR_AES_CTR は無し
- プロトコルのオプショナルな部分でサポートしていない、または最低限サポートのもの
  - NAT traversal 無し
  - CONFIG ペイロード無し（受信するとエラー）
  - EAP 無し（同）
  - window は minimum（1）
  - rekey 無し
- informational exchange の扱いは minimum
- DELETE ペイロードの処理無し
- peer polling 無し
- 相手が NO_ADDITIONAL_SAS を返してきた時の処理が無い
- cookie の secret は固定
- トランスフォームネゴシエーション、ID=0 の対応が無し
- IPCOMP_SUPPORTED の取扱無し

3.2.4 racoon2 仕様に対する未実装部分

racoon2 仕様に対して以下のような制約・未実装部分がある。

- SA ライフタイムは iked が管理せずカーネルの SADB_EXPIRE メッセージで処理する
- コンフィギュレーション項目の proposal_check が未実装（指定にかかわらず obey）
- コンフィギュレーション項目の selector_check が ID の選択に反映されない。TS の選択は exact のみ（これは仕様どおり）

- コンフィギュレーションに指定する IP アドレスは数値形式でなければならない
- コンフィギュレーションにマクロが使用できない
- コンフィギュレーションに矛盾・不足があっても、使われるまで検出できない

- MobileIP 連携などはまだ無い

3.3 kinkd

2004 年 12 月末時点での kinkd の実装状況を示す。

3.3.1 メッセージ交換パターン

kinkd の行うメッセージ交換は、大きく 3 つに分けることができる。このうち、KINK プロトコルが規定し

ているものは、基本的に鍵管理の部分である。

- TGT 取得
  Kerberos の KDC (Key Distribution Center) から TGT (Ticket-Granting Ticket)を取得する。
- サービスチケット取得
  KDC から通信相手に提出するためのチケットを取得する。
  基本的に Kerberos の KRB_TGS_REQ メッセージのみで取得するが、user-to-user モードを用いる場合は KINK の GETTGT メッセージも用いる。
- 鍵管理
  通信相手と折衝を行い、SA を設定する。
  CREATE メッセージで SA を生成し、
  DELETE メッセージで SA を削除する。

以下、各機能の実装状況

表 *3.1:TGT* 取得 *(Kerberos)*

Message flow	
KRB_AS_REQ -- KRB_AS_REP	ok

表 *3.2:*サービスチケット取得 *(Kerberos (+ KINK))*

Message flow	
KRB_TGS_REQ -- KRB_TGS_REP	ok
GETTGT -- REPLY -- KRB_TGS_REQ -- KRB_TGS_REP	NG
CREATE -- REPLY (KRB_AP_ERR_USER_TO_USER_REQUIRED) -- GETTGT -- REPLY -- KRB_TGS_REQ -- KRB_TGS_REP	NG

表 *3.3:*鍵管理 *(KINK)*

Message flow	
CREATE -- REPLY	ok
CREATE -- REPLY-- ACK (i.e 3-way hand shake	ok
DELETE -- REPLY	ok
STATUS -- REPLY	ok

3.3.2 REPLY メッセージで送受信するエラー

「送」は responder がそのエラーを返信するかどうか、
「受」は initiator がそのエラーを受信して処理できるかどうかを示す。

表 *3.4:KINK_KRB_ERROR*

ERROR	送	受
KRB_AP_ERR_BAD_INTEGRITY	ok	ok
KRB_AP_ERR_TKT_EXPIRED	ok	ok
KRB_AP_ERR_SKEW	ok	ok
KRB_AP_ERR_NOKEY	ok	ok
KRB_AP_ERR_BADKEYVER	ok	ok

表 *3.5:KINK_ERROR*

ERROR	送	受
KINK_OK	never	ok
KINK_PROTOERR	ok	ok
KINK_INVDOI	ok	ok
KINK_INVMAJ	ok	ok
KINK_INVMIN	ok	ok
KINK_INTERR	ok	ok
KINK_BADQMVERS	ok	ok

KINK_KRB_ERROR では以下の処理が可能である。

- TKT_EXPIRED 時のトランザクションやり直し
- ERR_SKEW 時の処理[kink-06 5.1.4 (page 18), 7.1 (page 27)]
  - responder は ctime に時刻を入れて返す
  - 次回 initiator は ctime で受けとった時刻を元にオフセット する

3.3.3 SAD 管理

「送」は kinkd がカーネルにそのメッセージを送信するかどうか、
「受」は kinkd がカーネルからそのメッセージを受信して処理できるかどうかを示す。

表 *3.6:SAD* 管理

PF_KEYv2 message	送	受
SADB_GETSPI	ok	ok
SADB_UPDATE	ok	NG(無視)
SADB_ADD	ok	NG(無視)
SADB_DELETE	ok	ok
SADB_ACQUIRE	NG	ok
SADB_REGISTER	ok	NG(無視)
SADB_EXPIRE	never	ok

3.3.4 ISAKMP ペイロード

kinkd が各 ISAKMP ペイロードを送受信できるかどうかを示す。

[]で囲まれているものは、KINK で optional と規定されているものである。

表 *3.7:CREATE/REPLY-to-CREATE*

ISAKMP ペイロード	
SA	ok
Nonce(Ni)	ok
Nonce(Nr, 3-way の場合)	ok
[KE]	NG
[ID]	ok
[Notification]	NG

表 *3.8:DELETE/REPLY-to-DELETE*

ISAKMP ペイロード	
Delete	ok
[Notification]	inprogress

表 *3.9:STATUS/REPLY-to-STATUS*

ISAKMP ペイロード	
[Notification]	ok

3.3.5 各種パラメータ

表 *3.10:IPsec* プロトコル

IPsec protocol	
AH	ok
ESP	ok

表 *3.11: IPsec SA* のモード

Mode of SA	
Transport	ok
Tunnel	NG

表 *3.12:*認証アルゴリズム

Authentication Algorithm	
MD5	ok
SHA-1	ok

表 *3.13:*暗号アルゴリズム

Encryptoin Algorithm	
DES	ok
3DES	ok
AES-128	ok
AES-192	ok
AES-256	ok

3.3.6 libracoon 化

kinkd は最初 racoon1 をベースに書かれたので、racoon2 の共通ライブラリである libracoon を利用していない部分があった。
ここではそれらの部分の libracoon 化の状況を示す。

表 *3.14:libracoon* 対応

機能	
SADB インターフェース(rcpfk)	ok
設定ファイル操作(rcf)	inprogress
共通タイプ(RCT)	inprogress
spmd インターフェース(spmif)	ok
ログ出力(plog)	NG
バッファ(wmbuf, rbuf)	ok
ネットワークユーティリティ	inprogress

3.3.7 その他

表 *3.15:*その他機能

機能・仕様	
epoch change の検出・処理 (DPD) [kink-06 4.4.2 (page 10)]	ok
複数の KINK_ISAKMP ペイロード [kink-06 5.1.7 (page 21)]	NG
randomized rekeying time [kink-06 4.4.1 (page 9)]	NG
replay protection (by Kerberos)	ok
DELETE grace timer	ok

3.4 libracoon

本章は racoon2 の共通ライブラリ libracoon.a について記述する。

3.4.1 共通ライブラリ概要

共通ライブラリ libracoon は、鍵管理デーモンの実装を容易にするためにカーネルとのインターフェイスや設定ファイルの解読などの共通処理をライブラリ関数として提供する。
共通ライブラリは、主に以下のモジュールに分けられる。

- SAD/SPD 管理インターフェイス
- 設定ファイル操作
- spmd インターフェイス
- ログ出力
- 汎用バッファ rbuf に関するユーティリティ
- ネットワークに関するユーティリティ
- その他ユーティリティ

libracoon では、暗号アルゴリズムや各タイプの定数の実装による違いを吸収するために中間形式(RCT 形式)を定義している。ライブラリでの定数は全て RCT 形式で扱い、カーネルとの入出力や各鍵管理プロトコルに従って定数を変換する。
以下は RCT 形式のうち、特別な数字が予約されているリストである。

固定な値
RCT_BOOL_OFF = 0
RCT_BOOL_ON = 1
RCT_ADDR_INET = 0x1000
RCT_ADDR_FQDN = 0x2000
RCT_ADDR_MACRO = 0x4000
RCT_ADDR_FILE = 0x8000

以降は各モジュールについて説明する。詳細な記述や関数定義は配布予定のキットに含まれる libracoon.txt を参照されたい。

3.4.1 SAD/SPD 管理インターフェイス rcpfk

libracoon は、SAD や SPD を管理する API を提供している。これら関数はカーネルとのインターフェイスとして PF_KEYv2 を使っている。カーネルからの戻り値は、それぞれのメッセージに対応したコールバック関数を定義することで、アプリケーションがカーネルの戻り値を利用できるようにしている。
また、アプリケーションとパラメータの受渡しは構造体 rcpfk_cont を使い、パラメータの管理を容易にしている。現在の所、これらの API は非同期処理を想定していない。
この API の返り値として、sa_src, sa_dst, sp_src, sp_dst に値が 返される場合、これらのポインタは rcpfk_cont 内のバッファを指しているため、値を利用したい場合は呼出側でコピーしておくことが望ましい。

3.4.2 設定ファイル操作 rcf

racoon2 では設定の煩わしさを軽減させるために設定ファイルを共通化して各デーモンが同じファイルを読み込む。このために libracoon は、各デーモンが設定ファイルを操作し、設定を参照するための API を提供している。各設定はいくつかの構造体に納められ直接で参照できる。また個別の関数でも参照することができる。

3.4.3 spmd インターフェイス spmif

racoon2 では、各デーモンはセキュリティポリシーの設定や削除等を spmd に依頼する。libracoon は、この API を提供している。
コールバック関数を定義する事により、spmd からの戻り値をアプリケーションが利用できるようにしている。

3.4.4 ログ出力 plog

libracoon はログ出力のための共通関数群 plog を提供している。
plog は指定したメッセージはメモリダンプを、syslog()を使用して出力したり、ファイルや標準出力へ出力したりできる。またインラインで出力先を変更できる。

メッセージタグとして以下が用意されている。

RC_LOG_INFO
[INFO]と表示される。
一般的な情報

RC_LOG_PROTO_ERR
[PROTO_ERR]と表示される。
IKE や KINK 等のワイヤに流れるプロトコルのエラー
フォーマットがおかしい
チェックサムがおかしい
認証に失敗
RC_LOG_PROTO_WARN
[PROTO_WARN]と表示される。
ワイヤに流れたプロトコルのエラーで処理を中断しない時

RC_LOG_INTERNAL_ERR
[INTERNAL_ERR]と表示される。
PF_KEY や system call 等の内部処理に関するエラー
PF_KEY のエラー
malloc に失敗
DB にエントリーがない

RC_LOG_INTERNAL_WARN
[INTERNAL_WARN]と表示される。
内部処理に関するエラーで処理を中断しない時

RC_LOG_DEBUG
[DEBUG]と表示される。
デバッグ

RC_LOG_CRITICAL
[CRITICAL]と表示される。
exit(-1)する状態

3.4.5 汎用バッファ rbuf

libracoon は、固定で割り当てられたバッファ領域を順次使用する関数 rbuf を提供している。
rbuf は、2つの固定サイズバッファを1つの可変長バッファから構成され、初期化時にそのサイズと個数を設定する。
使用する時に必要なバッファサイズに応じた関数を呼び出し、使用後は free()する必要はなく順次再利用される。従って同時に使用される個数を予め見積もっておく必要がある。

3.4.6 ネットワークに関するユーティリティ

libracoon は、ネットワークに関する関数を提供している。racoon2 では、設定ファイルでインターフェイスについた IP アドレスを表現するために以下に示す特別な文字列を用意している。以下のプレフィックスに MY_が付く文字列は全て自インターフェースに対して作用する。

表 *3.16:*アドレス設定マクロ

文字列	アドレス
MY_IP	自インターフェース全てについている IP アドレス
MY_IPV6	自インターフェース全てについている IPv6 アドレス
MY_IPV6_GLOBAL	自インターフェース全てについているグローバルアドレス
MY_IPV6_LINKLOCAL	自インターフェース全てについているリンクローカルアドレス
MY_IPV4	自インターフェース全てについている IPv4 アドレス
IP_ANY	::と 0.0.0.0 が返る

IP_ANY を除く上記の文字列の直後に文字 '%' を付加し続けてインターフェイス名を指定することで、展開する IP アドレスの範囲を絞る事ができる。

libracoon では、この文字列や IP アドレス、FQDN を一元管理するために rc_addrlist 構造体を使っている。

以下は rc_addrlist 構造体である。

```
struct rc_addrlist {
        struct rc_addrlist *next;
        rc_type type;
        int port;
        int prefixlen;
        union {
                struct sockaddr *ipaddr;
                vchar_t *vstr;
        } a;
};
```

type
RCT_ADDR_INET
　IP アドレスまたはネットワークアドレス
RCT_ADDR_FQDN
　文字列
RCT_ADDR_MACRO
　IP アドレスを示す文字列(前述)
RCT_ADDR_FILE
　UNIX ドメインのファイル名
port
　type == RCT_ADDR_INET の時に a.ipaddr のポート番号がコピーされる

RCT_ADDR_INET の時は a.ipaddr で参照できる。ype がそれ以外の時は a.vstr で参照できる。

3.4.7 その他ユーティリティ

その他のユーティリティとして、バージョン番号を返す関数や
起動時に表示するメッセージを返す関数が用意されている。

4 まとめ

racoon2 は、IKEv2 と KINK を用いた基本的な鍵交換が可能である。今後の課題としては、まず IKE を含む未実装となっている機能の実装があげられる。また、既存の racoon の設定と racoon2 の設定が大きく異なるために、racoon2 を使いやすくするためには設定ファイルの変換ツールなどを用意する必要がある。

IPsec WG 全体では、アプリケーションから IPsec の状態を参照できる API に関する研究などが考えられるが、当面は引き続き racoon2 の開発を中心に活動を行う。

Appendix

Glossary

- Authentication Header (AH)

  RFC2402 で定義され、パケットの認証と完全性を保障するための情報を格納したヘッダである。

- Internet Security Association and Key Management Protocol (ISAKMP)

  RFC2408 で定義されるインターネット上で SA と鍵を交換するためのフレームワーク

- Internet Key Exchange version 2 (IKEv2)

  draft-ietf-ipsec-ikev2-xx.txt で定義される。IKE の鍵交換プロトコルの後継プロトコルで必要なメッセージ数などの改善がなされている。

- IP Encapsulation Security Payload (ESP)

  RFC2406 で定義され、ペイロードに対して機密性を提供する。認証と完全性を提供することもできる。

- IP Payload Compression Protocol (IPComp)

  IP レイヤにおいてペイロードを圧縮するためのプロトコル

- Kerberos

  RFC1510 に定義されている認証を行うプロトコル。三者認証を行うために Key

Destribution Center (KDC)を導入している。

- Kerberized Internet Negotiatoin of Keys (KINK)

  ノードの認証に Kerberos を用いた鍵交換プロトコルで、ペイロードに ISAKMP と IPsec DOI を利用している。draft-ietf-kink-kink-xx.txt に定義されている。

- PF_KEY Key management API, version2 (PF_KEYv2, PF_KEY)

  RFC2367 で定義され、socket を持つアーキテクチャにおいて SA を管理するインターフェースを定義している。インターフェースは拡張可能で BSD 系 OS や Linux では、SP も管理できるように拡張されている。

- Security Architecture for Internet Protocol(IPsec)

  RFC2401 で定義され、IP レイヤにおいて認証、機密性、完全性、否認防止などのセキュリティサービスを提供する。

- Security Association (SA)

  単方向の論理的なコネクションでセキュリティサービスを提供するための情報の単位

- Security Association Database (SAD)

  IPsec SA を格納したデータベースで、IP アドレスやセキュリティプロトコル(AH,ESP)から SA を検索する。

- Security Policy (SP)

  Security Policy は、IP アドレスやプロトコルなどに基づいてパケットをフィルタし、そのパケットに対してどのような処理を行うかを定義する。

- Security Policy Database (SPD)

  Security Policy を格納したデータベース

- The Internet IP Security Domain of Ineterpretation for ISAKMP（IPsec DOI)

  RFC2407 で定義され IPsec の SA と鍵を交換するために ISAKMP を使う際のパラーメータの値と意味を定義する。

- The Internet Key Exchange (IKE、IKEv1)

  RFC2409 で定義される RFC2407,RFC2408 を用い 2 つのノード間で IPsec SA を動的に設定するためのプロトコル。