オープンソースコミュニティへの貢献 ―Linux及びBSD用通信ソフトウェアの開発

Contribution to Open Source Community – Development of Communications Software for Linux and BSD Operating Systems

神田　充　　小堺　康之　　福本　淳
■ KANDA Mitsuru　■ KOZAKAI Yasuyuki　■ FUKUMOTO Atsushi

近年，オープンソースと呼ばれるソフトウェアと，その開発形態が普及してきた。オープンソースでは，誰でも自由に使用したり改良できるライセンスの下で，世界中の開発者が協力してソフトウェアを開発し，そのソースコードは無償で公開される。企業がこれらオープンソースソフトウェアを利用する場合，ただ単に利用するだけでなく，オープンソースソフトウェア開発への貢献が求められる。

東芝は，Linux(注1)のIPv6 (Internet Protocol version 6) IPsec (Security Architecture for the Internet Protocol) ネットワークスタック，Linux IPv6用パケットフィルタプログラム，及びLinuxとBSD (Berkeley Software Distribution) 系OS (基本ソフトウェア) のIPsec用暗号鍵交換プログラムの三つのコンピュータネットワークにかかわるオープンソースソフトウェア開発に参加し，これらのソフトウェアの普及に貢献した。

Open source software, which can be used and revised freely under publicly accessible licenses, has become increasingly popular in recent years. It is developed with the cooperation of many software engineers around the world, and the source code is open to anyone free of charge. When companies employ open source software in their products, they are required to not only make use of this open source software but also to contribute to the open source community.

Toshiba has been contributing to open source software activities through the development of computer networking software including Internet Protocol Version 6 (IPv6)/Security Architecture for Internet Protocol (IPsec) software for Linux, and IPsec keying software for Linux/Berkeley Software Distribution (BSD).

1　まえがき

ソフトウェアの設計図（ソースコード）を一般に公開するライセンスの下で，世界中の開発者の協力により開発されるオープンソースソフトウェア[1]は，インターネットの普及とともに広く利用されるようになってきた。企業も開発コストを抑えるなどの理由により，製品に利用することが多くなってきている。企業がオープンソースソフトウェアを製品に組み込む場合，ただ流用するだけでなく，自ら開発に参加し，オープンソースコミュニティに貢献することが求められる。これにより，企業は“ただ乗り”の批判を免れ，必要な開発に対する主導権を発揮しながら，オープンソースコミュニティからの協力を得ることができる。

ここでは，東芝のオープンソースコミュニティへの貢献について述べる。特に，Linux用IPv6 (Internet Protocol version 6) IPsec (Security Architecture for the Internet Protocol) スタックを開発しているUSAGIプロジェクト，Linuxのパケットフィルタ機能を開発しているNetfilterプロジェクト，及びIPsec暗号鍵交換プログラムを開発しているRacoon2プロジェクトの活動について述べる。

2　USAGIプロジェクト

インターネットの通信プロトコル（規約）であるIPは，現在使用されているIPv4の次世代規格としてIPv6が標準化されている。Linuxは，オープンソースライセンスであるGNU(注2) ジェネラルパブリックライセンス（GPL）の下で開発されているOSである。しかし，LinuxにはIPv6の機能が十分に実装されていなかった。そのためLinux用IPv6機能の開発，及びIPv6規格に標準で組み込まれているセキュリティ機能であるIPsecスタックを開発することが求められていた。これらの機能を実装するために，産学協同団体であるWIDEプロジェクト[2]を母体として国内企業及び大学が中心となり2000年にUSAGIプロジェクト[3]が結成され，当社も立上げ時から参加した。

2.1　IPv6 IPsecの開発

当社は，IPv6のIPsecスタックを中心に開発を担当した。開発にあたり留意した点は次のとおりである（**図1**）。

（注1）　Linuxは，Linus Torvalds氏の米国及びその他の国における登録商標。

（注2）　すべてフリーソフトウェアから成るUNIX互換の環境を実装することを目的としたプロジェクトの名称。

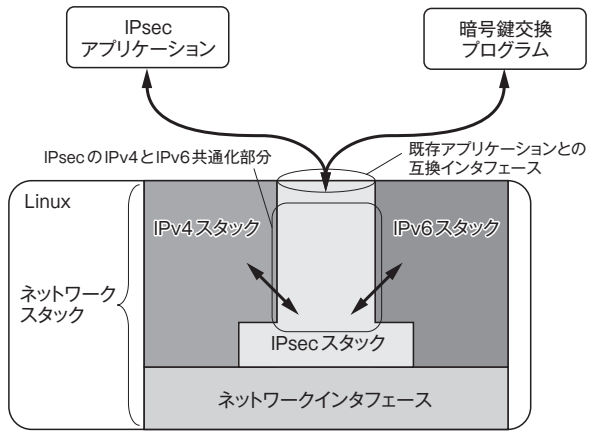

図1. Linuxネットワークスタックへの IPsec機能組込み — 既存アプリケーションとの互換性を保ち，IPv4とIPv6のIPsec処理を共通化している。

Embedded IPsec stack in Linux networking stack

(1) 既存のネットワークスタックに影響が少ない実装

既に動作しているLinuxのネットワークスタックの変更を少なくして，Linuxネットワークスタックのほかの部分に与える影響を少なくする必要があった。

(2) 既存のIPsec暗号鍵交換アプリケーションとの互換性

IPsecでは，通信を暗号化するために必要な暗号鍵を通信先と交換する機能が必要である。この機能を実現するソフトウェアは，既存のものを使用できるようにする必要があった。そのため暗号鍵などを設定するインタフェースについては，これらソフトウェアとの互換性を保つ必要があった。

(3) IPv4 IPsecとの共通化　　現行バージョンのIPv4用IPsecスタックと機能が共通化できる部分は，ソースコードの量を少なくするためにも共通化する必要があった。

2.2 Linuxカーネル本体への統合

開発した機能をLinuxのカーネル（OSの中心部分）本体へ統合して初めて，その機能を広く社会一般で使うことができるようになる。しかし，当初は開発したソースコードをLinux本体に取り込むことが困難であった。その理由は，次のとおりである。

(1) 知名度と信頼度の不足　　IPv6 IPsecの機能が現在社会に普及している技術ではないため，必要性そのものを開発コミュニティでは疑問視していた。また，USAGIプロジェクト自体がLinuxの開発コミュニティの中で無名であったため，作成したソースコードに対して信頼を得ることができないでいた。

(2) 膨大な量のソースコード　　開発した機能が多岐にわたったため，そのソースコードは膨大な量であった。そのため，開発コミュニティでの機能検証を困難にしていた。

Linux本体への統合を実現するために，次のような活動を行った。

(1) 広報活動　　国内外のLinux関連のカンファレンスなどに積極的に参加し，Linux開発コミュニティに対してUSAGIプロジェクトの活動内容やIPv6 IPsecの必要性の広報に努めた。

(2) 細かく分割したソースコード　　一度に膨大な量のソースコードをLinuxの開発責任者に送付するのではなく，機能ごとに細かく分割したソースコードを作成して，開発コミュニティが検証しやすい形にした。

これらの活動の結果，2003年にLinuxのメジャーバージョンアップである2.6のリリースに合わせて，Linux本体に統合されて配布されるようになった。また，初めてLinuxのソースコードに含まれるクレジットに当社の名前が掲載され，当社の貢献が広く認識されることになった。

3 Netfilterプロジェクト

Netfilterプロジェクト[4]は，Linuxのパケットフィルタ機能を開発し保守するために，1998年から活動しているオープンソースコミュニティである。パケットフィルタとは，不正な通信データを遮断する機能である。2003年ころ，IPv6が普及するにつれてLinuxのIPv6機能も充実しつつあった。しかし，IPv6のパケットフィルタの機能は不十分であった。

そこで当社は，Netfilterプロジェクトに参加し，IPv6に対応したパケットフィルタの機能を充実させた。特に熱望されていたStateful Packet Inspection（SPI）機能と，Netfilterプロジェクトをはじめとするコミュニティとのかかわり方について，以下に述べる。

3.1 IPv6対応SPIの開発

SPIは，ユーザーが複雑な設定を行わなくてもネットワーク内部の安全性を向上させる機能である。SPIを搭載したホームルータの例を**図2**に示す。

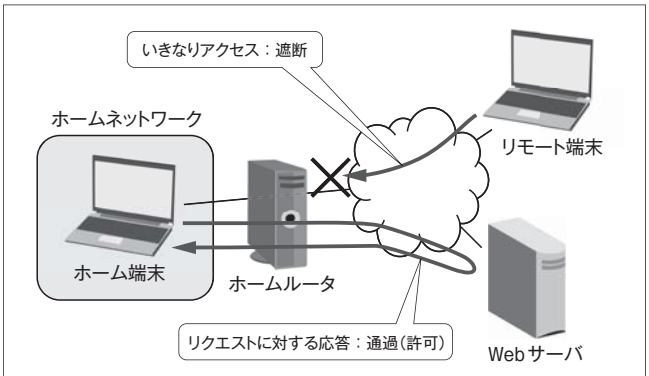

図2. SPIを搭載したホームルータの例 — SPIを備えたホームルータが外部からの進入を防いでいる。

Home router with stateful packet inspection (SPI)-based firewall

一般論文

SPIは，ホームネットワーク外部から到来したパケットと過去の通信との関連性を，通信規格に照らして検査する。関連性がなければ，ホームネットワーク外部から突然到来したパケットであると判断し，遮断する。一方，関連性があれば，そのパケットの通過を許可する。

SPIの設計に際し，LinuxのSPIが従来の通信規格であるIPv4のパケットだけでなくIPv6のパケットも遮断できるようにし，特に次の点を工夫した。

⑴ 保守しやすいプログラム構成　　IPv6に対応したSPIを容易に実現する一つの方法は，新たにIPv6専用のSPIを実装することである。しかし，従来のIPv4専用SPIにおける多くの処理をIPv6専用SPIに移植することになる。その結果，後に不具合が発見されると両方のSPIを修正する必要があり，プログラムの保守が煩雑になることが容易に想像された。そこで，SPIの構成をIPv4依存部，IPv6依存部，IPv4とIPv6の共通部に分けて設計した。実装の結果，IPv4とIPv6の共通部は全体の72％となり，大部分において重複した実装を回避できた。

⑵ 効率的なメモリの利用　　検査対象のパケットと過去の通信との関連性を調べるため，SPIはすべての通信について様々な情報とともにパケット中のIPv6アドレスを保持する。IPv6アドレスは128ビットであり，32ビットのIPv4アドレスよりも大きいため，1通信当たりのメモリ使用量が従来のIPv4専用SPIに比べ増大してしまう。そこで，代わりにIPv6アドレス以外の情報を収めるメモリ領域を削減した。SPIが保持する情報を分類すると，Network Address Translation（NAT）などIPv4の機能にしか利用されない情報があった。IPv6の通信に対してそれらを保持しないようにすることで，IPv6の1通信当たりのメモリ使用量を380バイトから244バイトへ，約36％削減した。

3.2 Linuxコミュニティとのかかわり方

初め社内で利用する目的でIPv6に対応したSPIを開発したが，その後，開発したSPIを公開し，Linuxコミュニティの協力を得ながら2005年11月Linux本体に統合した。オープンなLinuxコミュニティでの開発に切り替えた理由は，次のとおりである。

⑴ 多くの専門家との連携による相乗効果　　SPIをLinux本体に統合する際に多くの開発者が議論に加わることで，SPIを改善できる。実際，前述した工夫点もそういった議論のなかから生まれたものである。

⑵ 保守コストの分散　　継続的に保守するためには日々進化するLinuxに合わせてプログラムの修正が必要であり，多大な労力を伴う。SPIをLinux本体に統合すれば，世界中でLinuxのネットワーク機能を利用するユーザーや開発者が直接，又は間接的にSPIの改善と保守にかかわるようになり，保守コストを分散できる。また，オープンソースソフトウェアのコミュニティでは，開発物を保守する開発者が一定である必要がないため，必要に応じてLinuxコミュニティの中で興味のある開発者に引き継ぐことも可能である。

このように，Linuxユーザーが IPv6に対応したSPIのプログラムを得られる一方で，SPIをより改善できるとともに保守コストを下げることができた。

SPIはLinuxのネットワークに関係する機能であるため，ネットワーク機能開発者が集うLinuxコミュニティに参加した。当社が主にかかわったコミュニティは次の三つである。

⑴ USAGIプロジェクト　　当社が実装したIPv6対応のSPIを協力して検証した。LinuxやIPv6機能について開発者どうしが直接会い議論できる場であったため，開発スピードを上げることができた。

⑵ Netfilterプロジェクト　　既にLinuxコミュニティの信頼を得ていたUSAGIプロジェクトの実績をもとにこのプロジェクトに参加し，共同してSPIを開発した。Linuxパケットフィルタ機能に精通した開発者が集まっているため，深い議論を行うことができた。

⑶ Linuxネットワーク開発者コミュニティ　　Linuxカーネルのネットワーク機能開発者のメーリングリストに参加した。SPIはここでレビューを受けた後，最終的にLinuxカーネル本体に取り込まれた。

4 Racoon2プロジェクト

Racoon2プロジェクト[5]は，Linux及びBSD（Berkeley Software Distribution）系OSでIPsecプロトコルに用いる暗号鍵交換プロトコルを実装したソフトウェアRacoon2を開発している。

4.1 開発の経緯

KAMEプロジェクトは，BSD系OSのためのIPv6参照実装を開発するために，WIDEプロジェクトを母体として国内企業及び大学が中心となり設立された。当社も1998年のプロジェクト立上げから参加していた。IPv6プロトコルの参照実装を作成する一環として，KAMEプロジェクトではIPsecプロトコルのための暗号鍵交換プロトコルであるIKEv1（Internet Key Exchange version 1）プロトコルの実装をRacoonという名称で開発していた。

一方，インターネットのプロトコル標準仕様を策定する団体であるIETF（Internet Engineering Task Force）では，IKEv1を大幅改訂したIKEv2プロトコルの策定が進められていた（2005年に標準化）。IKEv2は，IKEv1に比べ相互接続性が向上しているとともにリモートアクセス機能とユーザー認証機能が充実している。また，IPv6の移動体向け拡張である

Mobile IPv6プロトコルなどでIKEv2が必須となることが見込まれていた。これらのプロトコルの検証のためにもIKEv2に対応した実装が必要であるとの認識がRacoon開発関係者にはあった。同時に，IKEプロトコルはプロトコルとして柔軟すぎて複雑であるため，より単純で処理負荷の小さい暗号鍵交換プロトコルであるKINK（Kerberized Internet Negotiation of Keys）の実装も同時に行いたいと考えていた。

これらのことから，新たに暗号鍵交換プログラムをRacoon2という名で作り直すことになった。

Racoon2は，暗号鍵交換プロトコルとしてIKEv1とIKEv2，KINKの三つを実装しており，spmd，iked，及びkinkdの三つのプログラムから構成される（**図3**）。

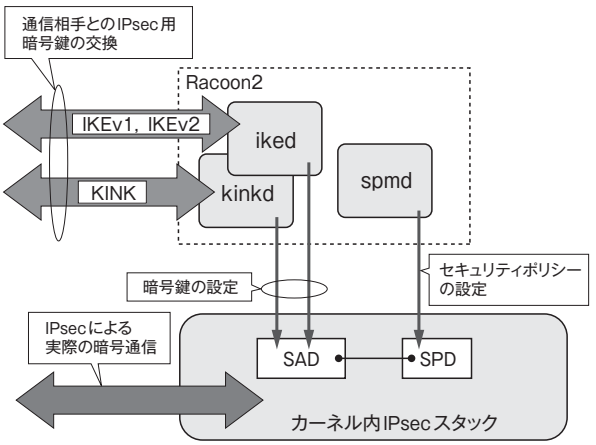

図3．Racoon2の構成 ― Racoon2は，セキュリティポリシーや暗号鍵の設定を行う三つのプログラムから構成されている。
Architecture of Racoon2 key exchange protocol

(1) spmd　　OSのIPsec適用方針を格納するセキュリティポリシーデータベース（SPD）への登録と管理を行う。

(2) iked　　IKEv1及びIKEv2プロトコルを処理し，暗号鍵をOSのIPsecパラメータを格納するセキュリティアソシエーションデータベース（SAD）に登録する。IKEv1とIKEv2は同一のUDP（User Datagram Protocol）ポートを使用している。両者の違いは，プロトコルで使用されるメッセージのヘッダ部分にあるIKEのバージョン番号によってだけ区別されるため，単一のプロセスで処理する構成となっている。

(3) kinkd　　KINKプロトコルの処理を担当し，暗号鍵をSADに登録する。

(4) 共通ライブラリ部　　設定ファイルの読込みやカーネルとの通信など，spmd，iked，kinkdそれぞれのプログラムで共通に使用される機能を共通ライブラリとして作成している。

4.2　開発体制

KAMEプロジェクトでは各種BSD系ＯＳのためのソフトウェアを開発していたため，それに合わせてオープンソースソフトウェアのライセンスの一つであるBSDライセンスを採用している。Racoonを継承したRacoon2でもBSDライセンスの下に公開されている。記述言語にはCを用いており，ソースコードは現在約10万行ほどである。

Racoon2での各プログラムは独立性が高いため，開発にあたってはプログラムそれぞれと共通ライブラリ部分に開発者1人ずつが担当し，当社はspmdとikedを主に担当した。

5　あとがき

ここで述べたオープンソースソフトウェアの開発を通して，当社のオープンソースへの貢献が広く認識されることになった。開発したソフトウェアはオープンソースコミュニティの協力によりメンテナンスされるため，コストを抑えながら様々な製品に利用することが可能になっている。また，今後ますます重要となるオープンソースソフトウェアの活用に関しても，コミュニティへの当社の貢献を通じて，率先してイニシアチブをとりながら必要な機能の開発に努めていく。

文　献

(1) Raymond, E. The Cathedral and the Bazaar. <http://www.catb.org/~esr/writings/cathedral-bazaar/>,（参照2009-09-08）.
(2) WIDEプロジェクト ホームページ. <http://www.wide.ad.jp/>,（参照2009-09-08）.
(3) USAGIプロジェクト ホームページ. <http://www.linux-ipv6.org/>,（参照2009-09-08）.
(4) Netfilterプロジェクト ホームページ. <http://www.netfilter.org/>,（参照2009-09-08）.
(5) Racoon2プロジェクト ホームページ. <http://www.racoon2.wide.ad.jp/>,（参照2009-09-08）.


神田　　充　KANDA Mitsuru
研究開発センター ネットワークシステムラボラトリー研究主務。ネットワークプロトコルの研究・開発に従事。
Network System Lab.


小堺　康之　KOZAKAI Yasuyuki
研究開発センター ネットワークシステムラボラトリー研究主務。ネットワーク家電の研究・開発に従事。
Network System Lab.


福本　　淳　FUKUMOTO Atsushi
研究開発センター ネットワークシステムラボラトリー研究主務。ネットワーク家電の研究・開発に従事。
Network System Lab.