システムソフトウェアと オペレーティング・システム 89－１１
システム評価 2－１１
（２００２．２．１５）

クライアントのユーザ認証情報を用いたサーバプロセスの権限変更機構

鈴木真一[†1] 光来健一[†3] 千葉滋[†2,†4,☆]
新城靖[†2] 板野肯三[†2]

要旨

この論文は，TCP/IP を利用したアプリケーションのために，クライアントプロセスの権限に従いサーバプロセスの権限を変更するための機構を提案している．本機構では，クライアントプロセスのユーザ認証情報 (Unix におけるユーザ ID およびグループ ID) が，IP オプションを利用して，サーバプロセスを実行するカーネルに送られる．新しいシステムコール `setremote` がサーバの権限を変更するために導入されている．このシステムコールは従来の UNIX とは異なり，`root` 特権がなくてもサーバの権限変更を可能にする．提案した機構は Linux カーネルにおいて実装されている．既存のサーバ（`inetd` および POP サーバ）において提案した機構を利用するようにしている．その結果，提案した機構の有効性を示している．

A mechanism to change authority of a server process using the user authentication information in a client

Shinichi Suzuki ,[†1] Kenichi Kourai ,[†3] Shigeru Chiba ,[†2,†4,☆]
Yasushi Shinjo [†2] and Kozo Itano [†2]

Abstract

This paper proposes a mechanism to change authority of a server process according to that of a client process for applications based on TCP/IP. In this mechanism, a credential (the user identifier and the group identifier in Unix) of a client process is sent to the kernel executing the server process by using the IP option. A new system call "setremote" is introduced to change authority of servers. This system call enables to change the authority of a server without the root privilege. The proposed mechanism has been implemented in the Linux kernel. Two existing servers (inetd and a POP server) have been adapted to use the the proposed mechanism, and the effectiveness of the proposed mechanism is shown.

1. はじめに

近年インターネットの普及により，TCP/IP[1)2)] を使うアプリケーションが多数開発されてきた．そのようなアプリケーションは，インターネット以外でも企業や学校などの LAN（Local Area Network）で利用されるようになった．このようなインターネットの技術を利用した LAN はイントラネットと呼ばれている．TCP/IP を利用するアプリケーションはクライアント・サーバ・モデルに基づいて構築されている．このようなネットワークでは，サーバは情報資源を集中的に管理し，クライアントはそれを利用する．

クライアント・サーバ・モデルに基づく通信では，接続を待っているサーバプロセス（以下サーバと略す）に対して，クライアントプロセス（以下クライアント

†1 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
Dept. of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba
†2 筑波大学電子・情報工学系
Institute of Information Science and Electronics, University of Tsukuba
†3 東京大学大学院理学系研究科情報科学専攻
Dept. of Information Science, Graduate School of Science, University of Tokyo
†4 科学技術振興事業団さきがけ研究 21
PREST, Japan Science Technology Corp.
☆ 現在，東京工業大学情報理工学研究科数理・計算科学専攻
Presently with Dept. of Mathematical and Computing-Sciences, Graduate School of Information Science and Engineering, Tokyo Institute of Technology

と略す）は接続要求を発行し，サーバが接続を受け付けることでコネクションが成立し，データの送受が行えるようになる．しかし，サーバは接続が成立した時点では，クライアントのユーザが何者であるかを知ることはできない．そのため，多くのサーバでは接続の成立後，クライアントのユーザを認証する．従来のオペレーティングシステム（OS）では，TCP/IP 経由でアクセスしているユーザを直接区別できないため，認証の仕組みは個々のサーバによって実装されている．

TCP/IP を利用したサーバの実現では，サーバが動作しているホストの UID（User IDentifier），GID（Group IDentifier）と遠隔のクライアントのユーザを対応させることが多い．例えば，ユーザ名とパスワードが正しければ，setuid() および，setgid() システムコールによって，そのプロセスの UID と GID を，対応したユーザの UID と GID に変更する．この UID と GID の変更を**権限変更**と呼ぶ．UNIX を中心とするイントラネットではユーザ名とパスワードを共有するために，NIS（Network Information Service）が用いられている．NIS を利用するイントラネットではサーバとクライアントの UID と GID は同じになることが自然である．

TCP/IP を利用するサーバでは，現在のところ認証にはユーザ名とパスワードが広く使われている．NIS を利用しているイントラネットでは，ユーザは手元のマシンにログインするとき，ユーザ名とパスワードを入力する．そのマシンのログインプログラムは，NIS サーバからユーザ名とパスワードを取り寄せ，認証を行う．その結果，そのユーザに対応した UID と GID をもつシェルが作られる．すなわち，カーネルの中にはユーザ認証が行われた結果を示す情報として，UID と GID が保存されている．この情報を**ユーザ認証情報**（credential）と呼ぶことにする．

イントラネットであっても，TCP/IP を利用する多くのサーバは，クライアントのユーザに対して認証を行う．そのため，そのようなサーバを利用するユーザは，手元のマシンにログインするときに，すでに入力したものと同じユーザ名とパスワードを，そのサーバに対して再度入力しなくてはならない．そこで，クライアントのユーザ認証情報が，サーバでも利用できることが望まれる．しかし，現状の UNIX ではそれを利用することができない．

この論文で我々は，イントラネットにおける，TCP/IP を利用したサーバの新しい権限変更機構を提案する．この機構では，クライアントのユーザ認証情報（UID，GID）をサーバに送る．サーバは，送られてきたユーザ認証情報を利用して，自分自身の権限をクライアントのユーザの権限に変更できるようにする．

この論文では，クライアントのユーザ認証情報を送るために IP オプションを用いる方法について述べる．IP オプションにより送られた，クライアントのユーザ認証情報はまずサーバの動作しているカーネル空間に保存される．サーバは，新しいシステムコール setremote() を利用して，カーネルに保存されているユーザ認証情報を基に，自分自身の権限を変更することができる．伝統的な UNIX では権限の変更を行うには root 権限（特権ユーザの権限）が必要であったが，このシステムコールは，root 権限をもたないプロセスでも利用可能にする．

この機構を利用することで，次のような利点が得られる．

(1) 新しいシステムコールの導入によりサーバを root 権限で動作する必要がなくなる．この結果，バッファオーバフロー攻撃などで root 権限が奪われる危険性がなくなる．
(2) TCP/IP を利用すること前提に設計されたサーバでも，イントラネットで実行される場合，個々にユーザ認証の仕組みを実装する必要がなくなる．
(3) ログイン時とサーバ接続時の，重複したユーザ認証がなくなる．ログイン時にユーザ認証を行うだけでよい．

2. 関 連 研 究

2.1 Identification Protocol (Ident)

Ident プロトコル[4] はサーバが，TCP コネクションの接続先にあるクライアントのユーザを調べる手段を提供するプロトコルである．Ident プロトコルを利用するためには，クライアントを動作させるホストにおいてデーモン（identd）を実行する．Ident プロトコルをユーザ認証やアクセス制御の目的に使うことはセキュリティーの低下を引き起こし好ましくないとされている[4]．しかし実際は，IP アドレスによるアクセスコントロールを強化する目的で，いろいろなサーバで利用されている．

サーバはクライアント側で動作している identd に対してポート番号を送ることで，そのポートを利用しているクライアントのユーザ情報を受け取ることができる．以下に応答の例を示す．

```
1022 , 22 : USERID : UNIX :shinichi
```

応答に含まれる情報は，ポート番号の組, USERID という文字列, OS の名前, およびユーザ情報である．ユー

ザ情報は最大で 512bites の文字列であり，一般にはクライアントの UID を/etc/passwd ファイルを使って逆引きしたユーザ名が使われる．現在 Ident プロトコルは，finger コマンド，WWW サーバ Apache，sendmail，TCP Wrapper（tcpd，tcpserver）などで利用されている．Apatch と tcpd ではその情報をロギングに利用しており，その情報は不正行為を行ったユーザの特定に有用である．sendmail や tcpserver☆では，IP アドレスによるアクセス制御に対して，付加的なアクセス制御を実現するために利用している．

Ident プロトコルを利用するには，すべてのクライアントが動作しているホストで identd を起動しなければならない．そしてサーバは，クライアントとの本来の情報の送受に使うコネクションとは別に，identd に対するコネクションを用意しなければならない．本論文で提案する機構では，Ident プロトコルとは異なり，本来のコネクション以外のコネクションを，別に用意する必要はない．

2.2 Network File System（NFS）

NFS[3] は Sun Microsystems 社が開発したネットワークファイルシステムである．NFS はカーネルレベルで実装され，カーネル内の NFS クライアントと NFS サーバで通信を行う．

NFS では，ネットワーク通信に Sun RPC（Remote Procedure Call）[5] を用いる．Sun RPC では，認証情報（クレデンシャル）と検証情報（ベリファイア）という概念により，NFS クライアントと NFS サーバの相互の認証を行っている．認証情報は，ホスト名，ファイルをアクセスしているプロセスの UID，GID，および groups（ユーザが所属するグループの配列）を含む．また検証情報は，認証情報が正しい事を確認するための情報である．例えば DES 認証では，暗号化されたタイムスタンプなどが含まれている．NFS クライアントは RPC 要求の中に認証情報と検証情報を入れて NFS サーバに送信し，NFS サーバは RPC 応答の中に検証情報のみを入れて NFS クライアントに返信する．RPC には表 1 のような認証の種類がある[7]．

UNIX で標準的に使われている NFS では，RPC の認証に UNIX 認証（AUTH_SYS）が用いられている．UNIX 認証では，検証情報は空である．つまり，NFS サーバには送られてきた認証情報が正しいものであるかどうかを検証することはできない．したがって，認証情報を偽造することで，正規のユーザになりすますことができてしまう．また，クライアントホストの特権ユーザは，任意のユーザにパスワードなしで変わることができるので，任意のユーザのファイルへアクセスすることが可能である．

このような NFS の問題はあるが，クライアントホストの IP アドレスを厳密に設定し，各クライアントホストで特権ユーザを守ることで実用的なレベルでその問題を解決することが可能である．したっがって NFS は，UNIX を中心とする LAN で広く利用されている．

NFS では，ホスト単位とプロセス単位で，アクセス制御を行う．ホスト単位のアクセス制御では，クライアントのホストの IP アドレスが使われる．プロセス単位のアクセス制御では，RPC メッセージのヘッダに組み込まれたクライアントの認証情報（UID，GID，groups）が使われる．

すべての NFS クライアントホストから UID が root（UID=0）によるアクセスを許可するとセキュリティ上重大な問題が発生することがある．この危険を少なくするために，一部のクライアントホストからのアクセスを除き，root（UID=0）は nobody（UID=-2）によるアクセスとして扱われる．

イントラネットで利用されている一般的なサーバと NFS を比較したとき，NFS の大きな特徴は次の 2 点である．

(1) NFS はカーネルレベルで実現されている．
(2) NFS は UNIX 認証では，認証情報（UID，GID，groups）は暗号化されずにそのまま送られている．

3. 対象とするサーバ構造

この論文で提案する権限変更機構は，次のようなホストから構成された環境で利用することを想定している．

(1) 各ユーザがログインするホスト（クライアントホスト）
(2) サーバを実行するホスト（サーバホスト）

すべてのホストでは，NIS などを利用することで，UID と GID が統一されているものとする．

ユーザがクライアントホストにログインするとき，

表 1 RPC の認証の種類
Table 1 Types of RPC authentication

認証の種類	認証技術
AUTH_SYS	認証情報として UID，GID およびユーザの所属するグループの配列を用いる
AUTH_DES	Diffie-Hellman 鍵配送方式，および DES を用いる認証技術
AUTH_KERB	Kerberos を用いる認証技術

☆ http://cr.yp.to/ucspi-tcp/tcpserver.html

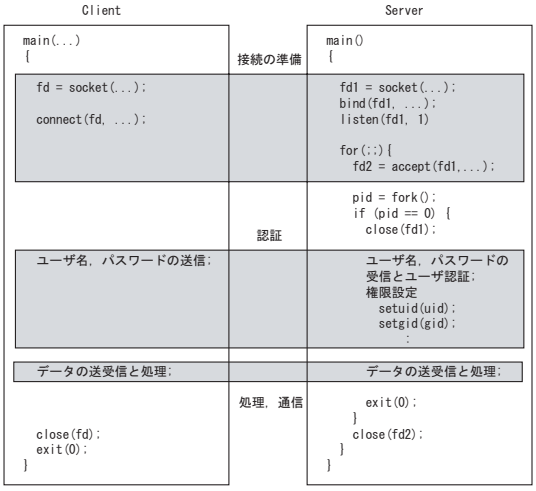

図 1 対象とするアプリケーションの構造
Fig. 1 The atructure of a target application

ユーザはユーザ名とパスワードを入力する．クライアントホストのシステムは有効であるユーザ名が，正しいユーザに使われていることをパスワードによって検証する．検証の結果システムが，正しいユーザであると認めれば，システムは最初のプロセスであるシェルを実行する．シェル，およびそれから派生したプロセスは属性として UID，GID，および groups を持つ．これらの属性はログイン時に，そのプロセスを起動したユーザが認証されたことを示すものである．この論文では，そのような属性を**ユーザ認証情報**（credential）と呼ぶことにする．カーネルは，プロセスをこの認証情報によって管理し，アクセス制御を行っている．

この論文で提案する権限変更機構が対象とする，クライアント・サーバ・モデルに基づいたプログラムは図 1 のような構造を持っている．クライアントは socket() システムコールでソケットを生成し，connect() システムコールによってサーバに対して接続要求を出す．サーバは，socket() システムコールでソケットを生成し，bind() システムコールでソケットに名前を割り当てる．次に，サーバは listen() システムコールによって接続の準備を行い，accept() システムコールによって接続要求のあったクライアントを受け入れ，クライアントと接続したソケットのファイルディスクリプタを得る．そして，サーバはこの接続されたソケットによって通信を行う．

コネクションが確立したクライアントとサーバでは，まず，クライアントの認証が行われる．典型的なアプリケーションでは，クライアントはユーザ名，パスワードをサーバに送る．サーバはユーザ名が有効であるかどうかを確かめ，パスワードによって，クライアントが本当にユーザ名によって示されるユーザであるかを確かめる．この手続きを経てサーバは，setuid() および setgid() システムコールを実行して権限の変更を行う．すなわち，そのプロセスの UID や GID といった属性を，クライアントのユーザに対応したものに変更する．

UNIX には，root 権限と呼ばれる考え方がある．root 権限，または特権ユーザの権限とは，UID が 0 のプロセスが持つ，すべてのことが実行できる権限である．

UNIX では，権限の変更（UID の変更）には，次の 2 つの方法がある．

(1) root 権限をもったプロセスが setuid() システムコールを実行する．
(2) set-uid ビットの立ったファイルを実行する．

TCP/IP を利用したサーバアプリケーションでは，普通 (1) の方法が用いられる．

root 権限から権限を変更するサーバはセキュリティホールによって root 権限を奪われる危険がある．UNIX における root 権限とは，すべてができるということであり，もっとも保護されるべきものである．root 権限を奪われる危険を減らし，システムを安全にするため，役割に基づくアクセス制御（Role-Based Access Control）に基づくシステム[10] や，新しい OS である Plan9[11] では root 権限のような特権を排除している．本論文では root 権限を使わずに権限を変更する機構を提案する．

4. ユーザ認証情報の伝播と権限変更

クライアントのユーザ認証情報を，そのままサーバで利用できるような機構を提案する（図 2）．クライアントが動作しているシステムのカーネルは connect() システムコールが発行されたとき，サーバにユーザ認証情報を送る．サーバは accept() システムコールを発行することで，クライアントのユーザ認証情報をソケットに記録する．この時点ではまだサーバの権限変更は行われていない．サーバは，必要に応じてあたらしく追加した setremote() システムコールを accept() したソケットに対し発行することで，保存されているクライアントのユーザ認証情報を基に，自分自身の権限を変更する．すなわち，そのプロセスの UID と GID 属性をクライアントのプロセスのものに変更する．ただし安全のため，クライアントが root 権限を持っていても，サーバの権限を root 権限へ変更することは禁止する．

setremote() システムコールは，root 権限がない

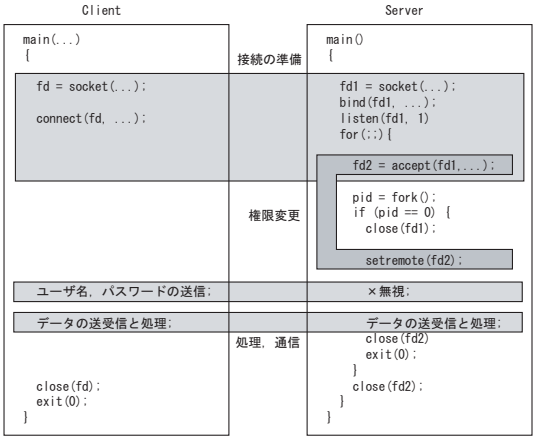

図 2 setremote() の利用法
Fig. 2 A server using setremote()

プロセスにも実行を許す．すなわち，UID が 0 ではないプロセスでも UID と GID 属性を，リモートのクライアントプロセスの UID と GID に設定することを許す．この機構によって，root 権限を用いずにクライアントに応じたサーバの権限の変更を行うことができる．UID と GID をプロセスの属性に設定しているので，サーバ側では OS によって提供されている，一般のアクセス制御を行うことができる．したがって，サーバプログラムごとにアクセス制御の仕組みを実装する必要がなくなる．

本機構を利用するためにクライアントのプログラムは変更する必要はない．ただしプロトコル上パスワードを送る必要がある場合，パスワードとしてはダミーの文字列を設定しておく．

4.1 IP オプションによるユーザ認証情報の伝播

ユーザ認証情報を伝播させるため IP オプションを利用する．IP オプション[1] とは，IP データグラムのヘッダに含めることができる少量のデータであり，制御やデバックの目的で用いられる．通常は IP オプションは設定されていない．ユーザ認証情報を伝播させる手段は TCP オプション[2] も考えられるが，今後 UDP への拡張を考慮して IP オプションを用いることにした．

ユーザ認証情報として今回は UID と GID を用いることにした．UNIX では，1 人のユーザが複数のグループに属すことが可能である．その場合，所属するグループは，グループの配列 groups で管理されている．groups は IP オプションに含めるには大きすぎるので，今回は扱わないことにした．Linux では，ユーザは最大 32 のグループに属することができる．GID は 16bits なので，groups の大きさは最大 64bytes になる．IP オプションの大きさは最大 44bytes である．

0			8	16	24	31
OPTION			LENGTH	DATA(32bits)		
0	0	10	6	UID		
GID				PADDING		

図 3 IP オプション USERINFO の構造
Fig. 3 Structure of IP option USERINFO

本権限変更機構を実現するために，新たに定義した IP オプション USERINFO の構造を図 3 に示す．図 3 の OPTION は，コピーフラグ，オプションクラス，オプションナンバーから構成される．コピーフラグは，フラグメンテーション時，このオプションをすべてのフラグメントにコピーするかどうかを決める．0 はコピーしないこと表している．オプションクラスは IP オプションを分類し，0 は制御用であることを表している．オプションナンバーは IP オプションを区別する番号である．今回は USERINFO として，使われていないオプションナンバーである 10 を使うことにする．

connect() システムコールが実行されると，TCP/IP ではコネクションを確立するために，クライアント（カーネル）とサーバ（カーネル）間で次のようなパケットのやり取りがされる．

(1) クライアントは，接続したいサーバのポート番号とクライアントの初期シーケンス番号 (普通クロックから作成される 32 ビットの整数) を指定した SYN パケットを送る．
(2) サーバは，サーバの初期シーケンス番号，クライアントのシーケンス番号+1 を含んだ SYN|ACK パケットを返す．
(3) クライアントはサーバから送られてきた SYN|ACK パケットに対して，サーバのシーケンス番号+1 を含んだ ACK パケットを返す．

本権限変更機構では，クライアント（カーネル）は SYN パケットの IP オプションにユーザ認証情報を入れ，サーバに送る（図 4）．サーバ（カーネル）は，SYN パケットを受け取ると，バッファに入っている IP オプションからユーザ情報をソケットに対応したデータ構造に保存する．クライアントは SYN|ACK パケットを受け取るとユーザ認証情報を再度送らないように IP オプションを設定しなおす．サーバ（カーネル）は ACK パケットを受け取ると accept() システムコールを完了する．

4.2 setremote システムコール

サーバ（プロセス）は，accept() システムコール時に生成されたソケットを指定して，setremote() システムコールを発行することで，自分自身の権限をク

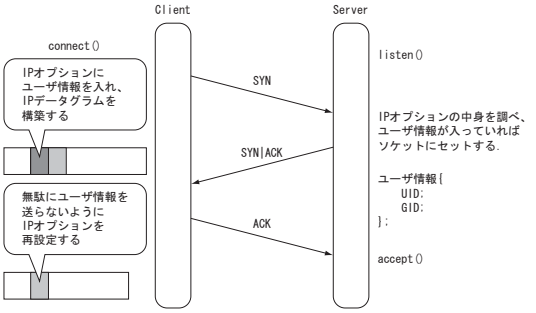

図 4 ユーザ認証情報の伝播
Fig. 4 Transmission of credential

ライアント（プロセス）の権限に変更することができる．このシステムコールが発行されると，カーネルは指定されたソケットに対応するデータ構造に保存されている UID と GID を取り出し，そのシステムコールを発行したプロセスの UID 属性と，GID 属性を変更する．ただし，ソケットに対応したデータ構造に保存されている UID と GID が 0 であった場合は，権限の変更を行わない．UID と GID が 0 であるとは，ユーザ情報がクライアントから送られてきていないか，もしくはクライアントのプロセスが root 権限を持っているかのいずれかを意味する．

5. 本機構の実装

4 章で述べた機構を次の環境で実装した．

- Red Hat Linux 6.2
- kernel 2.2.19

このカーネルのいくつかの既存の関数に変更を加え，いくつかの新しい関数と構造体を追加した（図 5）．はじめに，4 章で述べた IP オプション USERINFO を C 言語の構造体として実装した．次に，送られてきた UID と GID を元に，プロセスの UID と GID を変更する setremote() システムコールを実装した．

5.1 IP オプション USERINFO

IP オプション USERINFO の構造を C 言語の構造体として定義し，IP パケットに IP オプションとして含まれて送られるようにした．Linux のカーネルでは，IP オプション全体は構造体であり，個々のオプションは文字型の配列として定義されている．TCP の状態遷移[2] と，IP オプション USERINFO をあつかう関数の関係は，図 6 のようになっている．

Linux カーネルにおいては，connect() システムコールが発行されると，CLOSE 状態から，SYN_SENT 状態に遷移する．このとき，関数 tcp_v4_connect() が呼ばれる．この関数を変更し，新たな関数 set_ipopt_userinfo() を呼ぶようにした．この関数は IP オプションの構造体の OPTION（コピーフラグ，オプションクラス，オプションナンバー）と長さ，UID，GID を IP オプションの構造体に設定する．このとき，他オプションが先に設定されていれば，マージする．

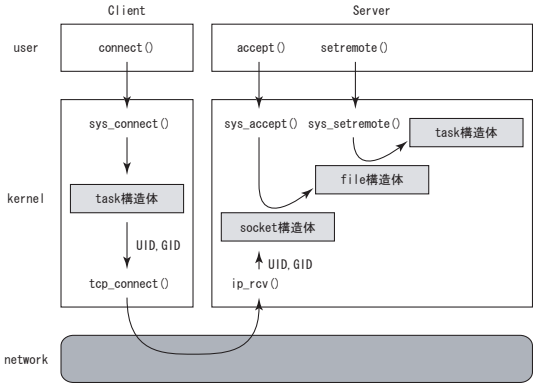

図 5 システムの概要
Fig. 5 Ovaer view of this system

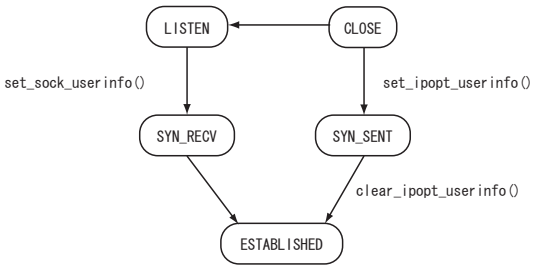

図 6 TCP 状態遷移と USERINFO の操作
Fig. 6 TCP state transition and manipulation USERINFO

Linux カーネルはネットワークからパケットを受け取ると，TCP/IP では最終的には関数 tcp_rcv_state_process() が呼ばる．この関数で，IP オプション USERINFO を扱うため，ソケットとパケットの状態に応じて次のような動作をするように変更した．

- ソケットが LISTEN 状態のとき，SYN パケットを受け取ると，set_sock_userinfo() が呼ばれるように変更した．この関数で IP オプション USERINFO からユーザ認証情報（UID，GID）を取り出し，ソケット構造体に保存する．ソケット構造体にはユーザ認証情報を保存できるように，2 つのフィールド userinfo_uid と userinfo_gid を追加した．
- ソケットが SYN_SENT 状態のとき，SYN|ACK パケットを受け取ると，clear_ipopt_userinfo() が呼ばれるように変更した．この関数では IP オプションの構造体を調べ，USERINFO が設定されていれ

ばそのデータを削除する．また，他のオプションがあれば，そのオプションのデータをマージする．

このようにコネクションをはるときのみ，ユーザ認証情報を IP オプションに入れて送るようにした．通常の read()，write() ではユーザ認証情報を送らない．このことはネットワークを流れるパケットのデータ量は，本権限変更機構を用いてもほとんど増加しないことを意味する．

サーバ側のカーネルは accept() システムコールが発行されると，ソケット構造体内のユーザ情報をファイル構造体にコピーする．これは setremote() システムコールで参照される．

5.2 setremote() システムコールの追加

setremote() システムコールは，accept() システムコールによって返されたファイルディスクリプタを指定することで，そのファイル構造体に保存されているユーザ認証情報（UID，GID）をタスク構造体の euid，ruid，fsuid，suid，egid，rgid，sgid，fsgid にコピーする．これは，setuid()，setgid() システムコールに習っている．

5.3 getcuid()，getcgid() システムコール追加

Ident プロトコルと同様に，権限の変更を伴わずクライアントのユーザ認証情報を得るだけでもロギングに便利である．そこで，クライアントのユーザ認証情報（UID，GID）を返すシステムコール getcuid() および getcgid() を実装した．

これらのシステムコールは accept() システムコールによって返された，ファイルディスクリプタを引数にとる．これらのシステムコールは，それぞれ結果としてそのファイル構造体に保存されている UID または GID を返す．

6. 既存のサーバでの利用例

本権限変更機構がどのように使われるかを示すために，実際に既存のサーバに変更を加えた．今回は，既存のサーバとして inetd と inetd から起動される POP サーバである qpopper を取り上げた．inetd から実行されるサーバをクライアントのユーザの権限で実行できれば，実行されるサーバ側での処理がより少なくなる．その結果，本機構がイントラネットにおいて有効に利用できることを示す．

6.1 inetd の変更

inetd から実行されるプロセスを，クライアントのユーザ権限で実行しできるように inetd の機能を拡張した（図 7）．inetd では実行するプロセスの UID を inetd.conf で次のように指定することができる．

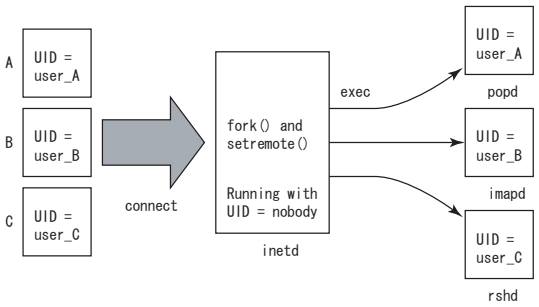

図 7 変更を加えた inetd の利用法
Fig. 7 A way to use modified inetd

```
pop3   stream  tcp     nowait  root    \
/sbin/popper   popper
```

これは pop3 という名前で表されるポート番号に対して TCP/IP（stream，tcp）により要求を受け付けたとき，プロセスを fork() して，プログラム/sbin/popper を実行することを意味している．nowait は，このプロセスの終了を待たない，すなわち，複数の popper プロセスを同時に実行することを意味する．root はこのサーバを実行するときに用いる UID である．

今回，inetd.conf の記述方法を拡張し，次のような指定を可能にした．

```
pop3   stream  tcp     nowait  remoteuser  \
/sbin/popper   popper
```

このように UID として，root の代わりに remoteuser と指定すると，クライアントのユーザ権限で指定したサーバを実行できるようにした．すなわち，クライアントのプロセスの UID と GID を持つプロセスが作られ，プログラム popper が実行される．

このような機能を実現するため必要だった，inetd のプログラムの変更は約 3000 行のコードのうち 8 行であった．プログラムの中で指定されたユーザ名の UID に setuid() に変更するところで，ユーザ名が remoteuser になっていれば setremote() を実行するというように変更した．

従来の inetd は実行するサーバの UID を指定すると setuid() システムコールが実行される．この setuid() システムコールを実行するためにはプロセスが root 権限で動作している必要がある．UID の指定を remoteuser のみにすると，本システムの setremote() のみを利用し，root 権限を使わずに済む．そのため，inetd プロセスを実行するのに root 権限が必要なくなる．その結果，万一 inetd が攻撃されプロセスを乗っ取られたとしても，root 権限が奪われる危険性はなくなる．

6.2 qpopper の変更

POP サーバである qpopper☆に変更を加えて，認証時にユーザ名のみを利用し，パスワードの確認を必要としないようにした．ただし，既存のメールクライアント（メーラ）に変更を加えなくても済むように，パスワードを受け取る部分などはそのまま残し，受け取ったパスワードは無視するようにした．また，実行時にすでにクライアントのユーザ権限に権限は変わっているので，認証後に `setuid()` と `setgid()` システムコールを発行しないようにした．

この機能を実現するために必要なプログラムの修正は，全体で約 20000 行のうち 1 行であった．

7. 今後の課題

クライアントのログをとるため `identd` を利用している sendmail，Apatch，TCP Wrapper といったサーバを本機構を利用するように変更する．また，IMAP，rsh といった，ユーザ名とパスワードを要求するほかのサーバでも，本機構を利用できるように変更する．これらの変更は，今回の `qpopper` 同様に行えると思われる．

現在の実装では，NFS と同様にクライアントからサーバに送られるユーザ情報を保護していない．すなわち，ネットワークの盗聴と IP オプションの偽装の対策を行っていない．本機構では NFS と同様に，イントラネットを構成しているクライアント側のホストが，きちんと管理されていることを想定している．

NFS におけるセキュリティの強化を参考に，本機構もより強化した認証の仕組みを導入する事を検討している．現在利用している IPv4 の IP オプションでは長さの制限があり，認証の強化には限界がある．

この問題を解決するために今後 IPv6 に対応させることを考えている．IPv6 では IPSec[9] が標準で利用される．IPSec は IP パケットを暗号化し，保護する機能がある．また，パケットの偽装を防ぎ，通信相手のホストの認証を行うことができる．IPv6 には IP オプションは存在しないが，IP ヘッダを独自に拡張することが許されている．そこでクライアントのユーザ認証情報を送るような IP ヘッダを追加したいと考えている．IP オプションと独自のヘッダにより，ネットワークの盗聴と IP オプションの偽装を防ぐことができる．

☆ http://www.eudora.com/qpopper/

8. ま と め

本稿では，TCP/IP を利用するアプリケーションを対象として，クライアントプロセスのユーザ認証情報（`UID`，`GID`）をサーバ側のカーネルに伝播し，その情報を利用してサーバプロセスの権限を変更する機構を提案した．そして，IP オプションを用いてユーザ認証情報を送る方法，および `root` 権限を用いずに権限の変更を行う `setremote()` システムコールについて述べた．提案した機構を Linux のカーネルにおいて実装した．また，既存のサーバとして `inetd` と POP サーバを取り上げ，本機構を利用するように変更した．その結果，本機構がイントラネットにおいて有効に利用できることを示した．今後の課題は，IPv6 において本機構を実装し，セキュリティを強化することである．

参 考 文 献

1) Jon Postel: Internet Protocol, *RFC 791*, September 1981.
2) Jon Postel: Transmission Control Protocol, *RFC 793*, September 1981.
3) Sun Microsystems, Inc.: NFS: Network File System Protocol Specification, *RFC 1094*, March 1989.
4) Michael C. St. Johns: Identification Protocol, *RFC 1413*, February 1993.
5) R. Srinivasan: RPC: Remote Procedure Call Protocol Specification Version 2, *RPC 1831*, August 1995.
6) Deering, S., and B. Hinden: Internet Protocol version 6 (IPv6) Specification, *RFC 1883*, December 1995.
7) M. Eisler: NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GSS and Kerberos V5, *RFC 2623*, June 1999.
8) Daniel P. Bovet, Marco Cesati: Understanding the Linux Kernel, *O' REILLY*, October 2000.
9) 萩野純一郎: IPSec, bit 別冊 情報セキュリティ, pp. 142-149, 共立出版, January 2000.
10) G.Faden: RBAC in UNIX administration, *In Proc. of ACM Workshop on Role Based Access Control*, pp. 95-101, October 1999.
11) Rob Pike, Dave Presotto, Sean Dorward, Bob Flandrena, Ken Thompson, Howard Trickey, and Phil Winterbottom: Plan 9 From Bell Labs, *Computing Systems, Vol 8 # 3*, pp. 221-254, Summer 1995.