ユーザ権限変更機構を利用した安全なイントラネットサーバの実現

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

この論文は，TCP/IP を利用したアプリケーションにおいて，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．そして，その機構を利用して安全なイントラネットサーバの実現方法について述べている．その機構では，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 など）が，IP オプションを利用して，サーバに送られる．送られたユーザ認証情報は，サーバプロセスの権限の変更に利用される．新しいシステムコール `getcuid`，`getcgid` および `getcgroups` がクライアントのユーザ認証情報を参照するために導入されている．従来の UNIX とは異なり，`root` 特権がなくてもシステムコール `setuid`，`setgid` および `setgroups` によってサーバの権限をクライアントの権限に変更することを可能にしている．これにより `root` 特権を利用せずにクライアントに応じた権限の変更を行うサーバの実現を可能にしている．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）および PAM（pluggable Authentication Modules）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，提案した機構の利便性を示している．

Realizing Secure Intranet Servers by Using a Mechanism Changing User Authority

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

This paper proposes a mechanism changing authority of a server process according to that of a client process for applications based on TCP/IP. Furthermore, this paper describes the realization of secure intranet servers by using this mechanism. In this mechanism, the kernel sends a credential, including a user identifier in Unix, of a client process to the remote kernel executing the server process by using the IP option. New system calls "`getcuid`", "`getcgid`" and "`getcgroups`" are introduced to refer the credential of a client. System calls "`setuid`", "`setgid`" and "`setgroups`" are adapted to change authority to the client without the root privilege. The proposed mechanism has been implemented in the Linux kernel. Two existing servers (`inetd` and a POP server) and PAM (Pluggable Authentication Modules) have been adapted to use the proposed mechanism, and the usefulness of the proposed mechanism is shown.

† 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
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba
†† 筑波大学電子・情報工学系
Institute of Information Science and Electronics, University of Tsukuba
††† 東京大学大学院理学系研究科情報科学専攻
Department of Information Science, Graduate School of Science, The University of Tokyo
☆ 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 Department of Mathematical and Computing Sciences, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
☆☆ 現在，NTT 未来ねっと研究所
Presently with NTT Network Innovation Laboratories

1. はじめに

近年インターネットの普及により，TCP/IP [12),13)] 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，クライアントと略す）は接続要求を発行し，サーバが接続を受付けることで接続が成立し，データの送受が行えるようになる．この際，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に対して認証を行う．ユーザの認証の結果は，カーネル内にプロセスの属性，`UID`（User IDentifer），`GID`（Group IDentifer）および `groups` という形で保存されている（以下，**ユーザ認証情報**と呼ぶ）．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，認証の仕組みは多くの場合，個々のサーバによって実装されている．

イントラネットではすべてのホストおよびサーバで，ユーザ名とパスワードを一元的に管理すると，パスワードの管理が楽である．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，そのようなイントラネット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ことがある．クライアントのユーザ認証情報が，サーバでも利用できれば，このような再度の入力が避けられ便利である．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 を利用したクライアントのユーザ認証情報の参照と，それを用いたサーバのユーザ権限変更機構[17]を提案し，さらにその機構を利用した安全なイントラネットサーバの実現について述べる．この機構では，クライアントのユーザ認証情報をサーバに送る．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 `getcuid()`（Get Client UID），`getcgid()`（Get Client GID）および `getcgroups`（Get Client groups）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得ることができる．それを基に，変更したシステムコール `setuid()`，`setgid()` および `setgroups()` を利用することで，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 `UID`，`GID` および `groups` 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，変更したシステムコールは，`root` 権限を持たないプロセスでも新しいシステムコール `getcuid()`，`getcgid()` および `getcgroups()` によって得られた `UID`，`GID` および `groups` に関しては変更可能にする．

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，およびシステムコールのセマンティクスの変更によりサーバを `root` 権限で動作させる必要がなくなる．この結果，この論文で提案する機構が利用可能なサーバでは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 を利用することを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，個々のサーバでユーザ認証の仕組みを実装する必要がなくなる．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．LAN 内のマシンにログインするとき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[8]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情報（ユーザ名など）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特別なサーバプロセス（`identd`）を実行する．

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
```

この応答に含まれる情報は，ポート番号の組（クライアント側が 1022，サーバ側が 22），`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 512 bytes 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．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の低下を引き起こし好ましくないとされている[8]．しかし実際は，IP アドレスによるアクセス制御を強化する目的で，いろいろなサーバで利用されている．

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．Apache 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，すべてのクライアントで `identd` を動作させる必要がなく，また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もない．

2.2 Unix の `root` 権限

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，すべてのことができる権限であり，最も保護されるべきものである．

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` 権限を持ったプロセスが `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 を利用したアプリケーションのサーバでは，普通 (1) の方法が用いられる．`setuid()` システムコールを用いて権限を変更するサーバにセキュリティ上の弱点があれば，`root` 権限を奪われる危険がある．

`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）モデル[7] や，`root` 権限のような特権を排除している新しいオペレーティングシステム[11] が開発された．本論文では UNIX において `root` 権限を使わずに権限を変更する機構を提案する．これによって `root` 権限が奪われる危険性を回避する．

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

2.3 Network File System（NFS）

NFS[16] 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 を利用することにより，ネットワークに接続されたコンピュータは，リモートホストのファイルシステムをローカルホスト上のファイルシステムと同様にアクセスすることが可能になる．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）[15] を用いている．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，認証情報が正しいことを確認するための情報である．RPC には**表 1** のような認証の種類がある[6]．たとえば DES 認証では，暗号化されたタイムスタンプなどが検証情報に含まれている．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 サーバは送られてきた認証情報が正しいものであるか検証する．

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 認証を行う NFS では，NIS などを用いてクライアントホストとサーバホストにおける `UID`，`GID` および `groups` を統一する必要がある．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，UNIX 認証が使われているところでは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．

NFS にはこのような問題はあるが，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

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

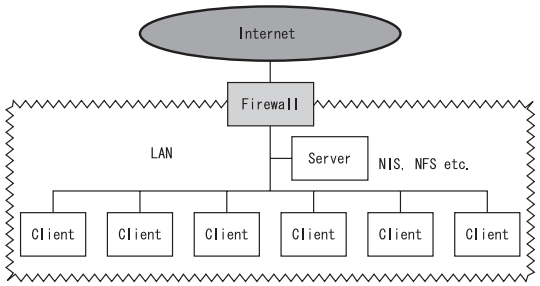


図 1 対象とするネットワーク環境
Fig. 1 The target networking environment.

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．したがって NFS は，UNIX を中心とする LAN で広く利用されている．

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，UNIX 認証を行う NFS の大きな特徴は次の 2 点である．

(1) カーネルレベルで実現されている．
(2) 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．

この論文で提案する機構はこのような NFS の特徴と同じ特徴を備えている．NFS との相違点は，提案する機構が一般的な TCP/IP に基づくサーバを対象としている点である．

3. 対象とするネットワーク，サーバおよび脅威

3.1 対象とするネットワーク

この論文で提案する権限変更機構は，**図 1** 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．図 1 のように，LAN はファイアウォールにおけるパケットフィルタによって外部から保護されており，特に，外部からのパケットに関しては IP オプション[12] を利用できないような設定になっているものとする．すべてのホストでは，NIS（Network Information Service）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．UNIX 認証を行う NFS が利用されているような環境と同様に，LAN は安全であり，盗聴，改ざん，偽装などはないものとする．さらに，クライアントホストの特権ユーザは保護されているものとする．すなわち，特権ユーザのパスワードは類推されにくいもので，厳重に管理されているものとする．しかし，一般ユーザは保護されていないものとする．たとえば，悪意のあるユーザにパスワードを類推され，またはパスワードの入力を盗み見られることで，正規ユーザに成りすまされる可能性がある．また，

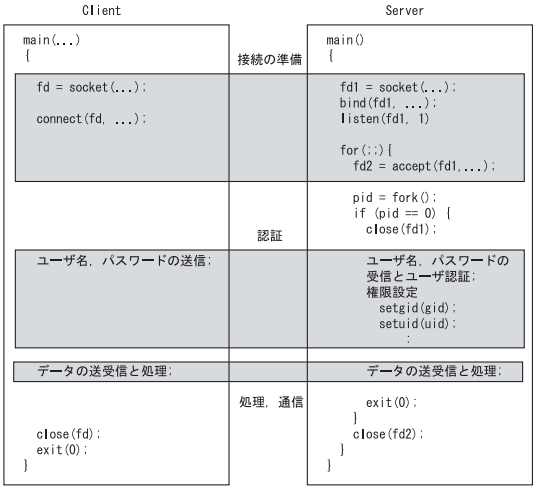


図 2 対象とするアプリケーションの構造
Fig. 2 The structure of a target application.

悪意を持った正規ユーザが存在することも考えられる．

この論文で提案する権限変更機構が対象とする，NIS など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 サーバなど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，GID および groups を持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，GID および groups が保存されている．この情報を**ユーザ認証情報**（User credential）と呼ぶことにする．

また提案する権限変更機構は，UNIX 認証を行う NFS と同様に UID と GID の一意性が保証されていることを必要とする．そのため，提案する機構では，UID と GID の一意性を保証できる規模の組織を対象としている．そのような NFS が利用されているような環境☆では，UID と GID の一意性は保証されているので，提案する機構は利用可能である．そのため，提案する機構が利用可能な環境のスケーラビリティは，UNIX 認証を行う NFS と同程度と考えられる．

3.2 対象とするサーバの構造

この論文で提案する権限変更機構が対象とする，クライアントとサーバの構造を**図 2** に示す．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

☆ 筑波大学情報学類では，クライアントマシン 120 台およびユーザ約 900 名の規模で NFS が利用されている．

備を行う．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 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るとともに，サーバは accept() システムコールによって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と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．このため，2.2 節で述べたように，サーバを root 権限で動作させる必要がある．

従来のシステムでは，このような環境では，各ユーザは手元のクライアントホストにログインするときに加えて，サーバに接続するたびにユーザ名とパスワードの組を入力することになる．同じユーザ名とパスワードを毎回入力するのでなく，はじめのクライアントのホストにログインするときの一度で済めばとても便利である．また，root 権限がなくても setuid() および setgid() システムコールによって権限の変更ができれば，より安全になる．

アプリケーションレベルで独自にユーザ名とパスワード管理しているサーバも存在するが，このようなサーバでは setuid() などを用いず，独自のアクセス制御を行っている．この場合，その独自に管理しているユーザ名とクライアントホストにログインするときに用いるユーザ名が異なることがある．提案する権限変更機構は，setuid() などを用いるサーバを対象としており，そのようなクライアントホストにログインするときとサーバに接続するときで異なるユーザ名を用いるようなサーバは対象としない．また，setuid() などを用いるサーバであっても，3.1 節で述べたようにクライアントホストとドメインの異なるところ（NIS などによって UID と GID の一意性が保証されないところ）に設置されたサーバホスト上のサーバは対象としない．

3.3 対象とする脅威（攻撃）

提案する権限変更機構では次のような脅威（攻撃）を対象としている．

(1) 外部の悪意を持ったユーザが，正規ユーザのユーザ名とパスワードを，入力を盗み見る，または類推して内部のホストにログインする．あるいは，内部の悪意を持った正規ユーザが内部のホストにログインする．

(2) 一般ユーザの権限でログインしたユーザは，root 権限で実行されているサーバプログラムに対して，セキュリティ上の弱点を利用した攻撃を行い root 権限を奪取する[1)~3)]．

(3) root 権限を奪取したユーザは，サーバ内の root 権限がなしではアクセスできないような重要なデータを読む，あるいは破壊する．

4. ユーザ認証情報の参照と権限変更

3 章で述べたように，クライアント側のカーネルには，ユーザ認証情報がある．そのクライアントのユーザ認証情報をサーバから参照し，サーバの権限変更に利用できるような機構を提案する（**図 3**）．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カーネル空間に記録する．この時点ではまだサーバの権限変更は行わない．サーバは，必要に応じて新しく追加した getcuid()，getcgid() および getcgroups() システムコールを accept() の結果として得られたファイルディスクリプタに対し発行することで，記録されているクライアントのユーザ認証情報を得ることができる．それを基に，セマンティクスを変更したシステ

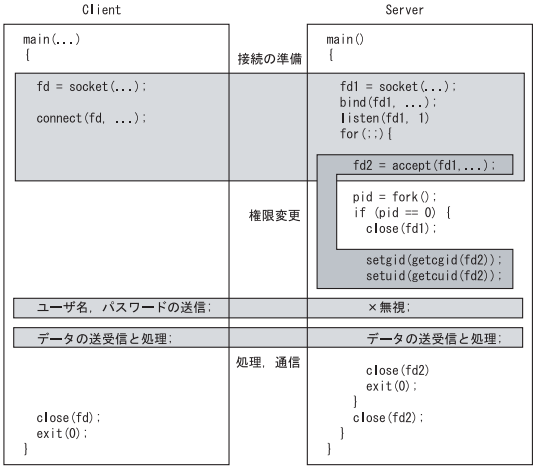


図 3 getcuid() と getcgid() の利用法
Fig. 3 Usage of getcuid() and getcgid().

ムコール setuid(), setgid() および setgroups() を利用して，自分自身の権限を変更する．すなわち，そのプロセスの UID, GID, groups 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．

2.2節で述べたように，従来の UNIX では setuid(), setgid() および setgroups() システムコールで UID, GID および groups を変更するには root 権限が必要であった．本機構ではこのセマンティクスを変更し，root 権限がないプロセスであっても，getcuid(), getcgid() および getgroups() システムコールによって得られた UID, GID および groups に関しては setuid(), setgid() および setgroups() システムコールによる UID, GID および groups の変更を許すようにした．すなわち，UID が 0 ではないプロセスでも UID, GID および groups 属性を，リモートのクライアントプロセスの UID, GID および groups 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 UID, GID および groups を変更するようなサーバを実行できるようになる．

図 3 において本機構を利用するためにクライアントのプログラムを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．

一般にクライアントアプリケーションにおいて，ユーザ名とパスワードを保存し，再度接続するときには入力の手間を省く手法もとられている．ところが，このような手法では，パスワードを変更するたびに再設定が必要になる．提案する権限変更機構がこのような手法より優れている点は，パスワードを変更しても個々のアプリケーションを再設定する必要がないところである．

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[12] 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（最長 40 bytes）であり，制御やデバックの目的で用いられる．IP オプションを有効に利用している例としては，たとえば，インターネットにおける反射撹乱型分散 DoS 攻撃を防ぐ仕組み[10] があげられる．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としては TCP オプション[13] 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．

実際の UNIX ではユーザ認証情報として，UID, GID および groups（GID のリスト）を利用している．今回用いる実行環境（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 16 bits なので，groups の大きさは最大 64 bytes になる．IP オプションの大きさは最大 40 bytes なので，すべてのユーザ認証情報を IP オプションに含めることはできない．そこで，今回は UID と GID は必ず送ることにし，groups は含められる範囲内（最大 17 個）で含めることにした．

表 2 IP オプション USERINFO の構造
Table 2 Structure of IP option USERINFO.

0			8	16	24	31
OPTION			LENGTH	DATA		
0	0	10	6 + 2n	UID		
GID				groups[0]		
groups[1]				…		
groups[n-1]				PADDING		

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**表 2** に示す．表 2 の OPTION は，コピーフラグ，オプションクラスおよびオプション番号から構成される．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．オプション番号は 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番号である 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 では一般に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とりが行われる[13]．

(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とクライアントから送られたシーケンス番号に 1 を加えたものを含んだ 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，サーバから送られたシーケンス番号に 1 を加えたものを含んだ 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

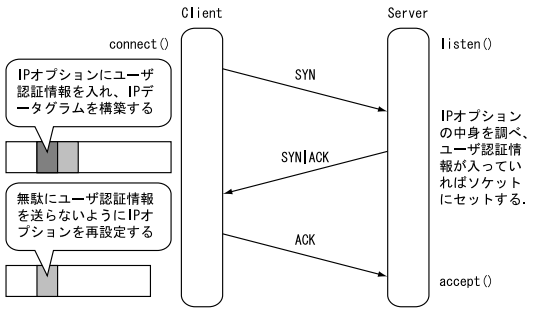


図 4 ユーザ認証情報の伝播
Fig. 4 Transmission of credential.

パケットを受け取ると，IP オプションからユーザ認証情報を取り出し，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．

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．

4.2 getcuid，getcgid および getcgroups 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，getcuid()（Get Client UID），getcgid()（Get Client GID）および getcgroups()（Get Client groups）システムコールを発行することで，それぞれクライアント（プロセス）の UID，GID および groups を得ることができる．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，GID および groups を取り出し，プロセスの属性として CUID（Client UID），CGID（Client GID）および CGROUPS（Client groups）を設定すると同時にリターンバリューとして返す．CUID，CGID および CGROUPS は今回新たに定義したプロセスの属性であり，クライアントの UID，GID および groups を示すものである．CUID，CGID および CGROUPS は通常は設定されていないことを意味する 0 である．

getcuid()，getcgid() および getcgroups() システムコールは，ソケットに対応したデータ構造に保存されている UID，GID および groups が 0 であった場合は，エラーを返す．UID，GID および groups が，0 であるとは，ユーザ認証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．

これらのシステムコールは，それぞれ主に以下で述べる setuid()，setgid() および setgroups() システムコールにより UID，GID および groups を変更するために用いられる．その他に，ident プロトコルと同様にロギングにも利用できる．

4.3 setuid，setgid および setgroups システムコールのセマンティクスの変更

通常の setuid() システムコールは，発行したプロレスが root 権限で実行されている（実効 UID が 0）か，もしくは root 権限で実行されていないプロセスが一度実効 UID を変更し，元の UID に戻る場合（実 UID が変更したい UID と同じとき）にしか使うことができない．

この setuid() システムコールのセマンティクスを変更した．変更したシステムコールは，プロセスの属性として設定されている CUID に関しては setuid() を許す．ただし，CUID が 0 であるときは CUID として UID が設定されていないと見なし，UID の変更は行わない．setgid() および setgroups() システムコールについても setuid() システムコールと同様に変更する．

サーバ（プロセス）は，getcuid()，getcgid() および getcgroups() システムコールによって得られた UID，GID および groups を指定して，setuid()，setgid() および setgroups() システムコールを発行することで，得られた UID，GID および groups をそのプロセスの属性に設定することができる．つまり，自分自身の権限をクライアント（プロセス）の権限に変更することができる．このように，クライアント側のプロセスのユーザ認証情報をサーバに送り，サーバ側のプロセスの権限をクライアント側のプロセスの権限に変更することは，実質的に NFS で行っていることと同じことである．したがって今回の setuid()，setgid() および setgroups() システムコールのセマンティクスの変更は，3 章で述べたような環境では問題ない．

4.4 対象とする脅威（攻撃）に対する本機構の有効性

本機構を利用すると，イントラネットサーバを root 権限を用いることなく（たとえば nobody 権限で）運用することができる．3.3 節で述べたような脅威はサーバが root 権限で実行されていることに起因する．したがって，はじめからサーバを root 権限で実行して

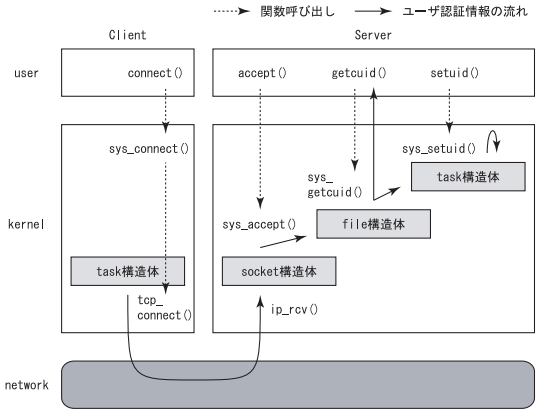


図 5 システムの概要
Fig. 5 Overview of the system.

いなければ，root 権限を奪われることはない．つまり，本機構は 3.3 節で述べた，一般ユーザの権限を持つ悪意を持った者が，セキュリティ上の弱点を持ったサーバを攻撃することで root 権限を奪う，という脅威に対し有効である．

5. Linux における実装

4 章で述べた機構を次の環境で実装した．

- Red Hat Linux 7.0
- kernel 2.2.20

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，送られてきたユーザ認証情報を得る getcuid()，getcgid() および getcgroups() システムコール，およびそれを元にプロセスの権限を変更する setuid()，setgid() および setgroups() 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 オプションとして設定され送られるようにした．

TCP の状態遷移[13] と，IP オプション USERINFO を扱う関数の関係は，**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，CLOSED 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，オプション番号）と長さ，および UID 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．

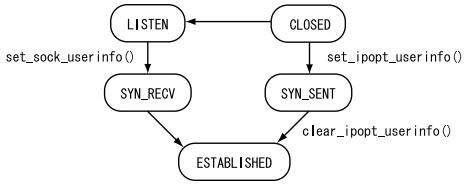


図 6 TCP 状態遷移と USERINFO の操作
Fig. 6 TCP state transition and manipulation USERINFO.

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() を呼ぶ．この関数で，IP オプション USERINFO を扱うため，TCP の状態とパケットの種類に応じて次のような動作をするように変更した．

- TCP の状態が LISTEN 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，新たなフィールドを追加した．
- TCP の状態が SYN_SENT 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．

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，ソケット構造体に含まれているユーザ認証情報をファイル構造体に追加した新たなフィールドにコピーし，ソケット構造体のユーザ認証情報は初期化する．

5.2 getcuid()，getcgid() および getcgroups() システムコールの追加

getcuid()，getcgid() および getcgroups() 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，それぞれそのファイル構造体に保存されている UID，GID および groups をタスク構造体に CUID，CGID および CGROUPS としてコピーし，リターンバリューとして返す．ファイル構造体に保存されている UID，GID および groups が 0 であるとき，もしくはすでにタスク構造体の CUID，CGID および CGROUPS に 0 以外の値が設定されているとき，エラーを返す．

5.3 setuid()，setgid() および setgroups() システムコールの変更

通常の Linux の setuid() システムコールは，発行したプロセスの UID 属性が 0 であれば，プロセスの UID 属性，EUID 属性（Effective UID），FSUID 属性（File System UID）☆1，および SUID 属性（Saved UID）にシステムコールの引数で指定された値を設定する．また，指定された値が UID 属性か SUID 属性と同じである場合，FSUID 属性と EUID 属性に指定された値を設定する．

これを変更し，指定された値が発行したプロセスの CUID 属性と同じであれば，setuid() システムコールを発行したプロセスの UID 属性が 0 でなくてもプロセスの UID 属性，EUID 属性，SUID 属性および FSUID 属性に指定された値を設定することができるようにした．

setgid() および setgroups() システムコールについても，setuid() システムコール同様に変更する．

6. 既存のサーバでの利用例

本権限変更機構がイントラネットにおいて容易に利用可能であることを示すために，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．また，既存のサーバの多くで PAM [14] が利用されていることから，既存のサーバから本機構を簡単に利用できるようにするため，PAM モジュールを本機構に対応させた．

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できるように inetd の機能を拡張した．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．

```
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 client_uid \
/sbin/popper popper
```

このように UID として，root の代わりに client_uid 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 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 3,000 行のコードのうち 8 行であった．プログラムの中で setuid() システムコールを用いて，指定されたユーザ名の UID に変更するところで，ユーザ名が client_uid になっていれば，次のようなコードを実行するように変更した☆2．

```
setuid(getcuid());
```

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．一方，本機構を実装したシステムでは，setuid() システムコールを変更したため，UID の指定を getcuid() システムコールによって行うことで，root 権限を使わずに済む☆3．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

広く使われている POP サーバの 1 つである qpopper☆4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．

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 20,000 行のうち 1 行であった．

6.3 PAM モジュールの変更

Linux における既存のサーバの多くは，種々のユーザ認証の方法に対応するために，PAM（Pluggable Authentication Modules）モジュールを利用することが可能になっている．そこで，本機構のユーザ認証の方

☆1 Linux 独自の属性．NFS アクセスのときにのみ使われている．

☆2 inetd では setgid() および setgroups() システムコールは使われていない．

☆3 この場合は，特権ポート以外のポート番号を割り当てる必要がある．

☆4 http://www.eudora.com/qpopper/

法に対応した PAM モジュールを用意し，既存のサーバのソースを変更することなく本機構を利用できるようにした．

PAM モジュールには多くの種類があり，それらの利用法は /etc/pam.d/ の下の設定ファイルに置かれる．今回は多くのサーバで利用されている，ユーザ名とパスワードを受け取ってユーザ認証を行う PAM モジュール（pam_pwdb）を対象とした．このモジュールは，PPP サーバ，FTP サーバ，Apache の拡張モジュールなどに使われている．

元の PAM モジュールは，ユーザ名とパスワードを基に認証を行い，成功すると PAM_SUCCESSED を返していた．これを変更し，受け取ったユーザ名とパスワードを無視するようにした，その代わり，getcuid()，getcgid() および getcgroups() システムコールを発行し，それらのシステムコールが成功したとき PAM_SUCCESSED を返すようにした．

変更したモジュールは約 3,200 行のコードであり，追加したのは 14 行であった．

7. 今後の課題

3 章で述べたように，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ことを検討している．

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．そこで，今後 IPv6 に対応させることを考えている．IPv6 では IPsec[9] が標準で利用できる．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．IPsec により，その独自のヘッダの偽装を防ぐことができる．

8. まとめ

この論文では，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 および groups）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と，新たなシステムコール getcuid()，getcgid() および getcgroups() について述べた．これらのシステムコールを利用することで，root 権限を用いずに権限の変更が可能になるので，安全なイントラネットサーバが実現できる．

提案した機構を Linux のカーネルにおいて実装した．また，inetd，POP サーバおよび PAM モジュールを取り上げ，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簡単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) CERT Advisory CA-1997-09: *Vulnerability in IMAP and POP* (Apr. 1997).
2) CERT Advisory CA-1998-08: *Buffer overflows in some POP servers* (July 1998).
3) CERT Advisory CA-2001-21: *Buffer Overflow in telnetd* (July 2001).
4) Bovet, D.P. and Cesati, M.: *Understanding the Linux Kernel*, O' REILLY (Oct. 2000).
5) Deering, S. and Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification, *RFC 2460* (Dec. 1998).
6) Eisler, M.: NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GSS and Kerberos V5, *RFC 2623* (June 1999).
7) Faden, G.: RBAC in UNIX administration, *Proc. ACM Workshop on Role Based Access Control*, pp.95–101 (Oct. 1999).
8) St. Johns, M.: Identification Protocol, *RFC 1413* (Feb. 1993).
9) Kent, S. and Atkinson, R.: Security Architecture for the Internet Protocol, *RFC 2401* (Nov. 1998).
10) 西尾信彦，原嶋章介，徳田英幸：反射撹乱型分散 DoS 攻撃に対する逆探知機構，情報処理学会研究報告システムソフトウェアとオペレーティング・システム，2002-OS-90, pp.65–72 (June 2002).
11) Pike, R., Presotto, D., Dorward, S., Flandrena, B., Thompson, K., Trickey, H. and Winterbottom, P.: Plan 9 From Bell Labs, *Computing Systems*, Vol.8, No.3, pp.221–254 (Summer 1995).
12) Postel, J.: Internet Protocol, *RFC 791* (Sep. 1981).
13) Postel, J.: Transmission Control Protocol, *RFC 793* (Sep. 1981).
14) Samar, V. and Schemers, R.: Unified Login with Pluggable Authentication Modules (PAM), *DCE-RFC 86.0* (Oct. 1995).
15) Srinivasan, R.: RPC: Remote Procedure Call

Protocol Specification Version 2, *RFC 1831* (Aug. 1995).
16) Sun Microsystems, Inc.: NFS: Network File System Protocol Specification, *RFC 1094* (Mar. 1989).
17) 鈴木真一，光来健一，千葉　滋，新城　靖，板野肯三：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，情報処理学会研究報告システムソフトウェアとオペレーティング・システム，2002-OS-89, pp.79–86 (Feb. 2002).

(平成 14 年 12 月 23 日受付)
(平成 15 年 4 月 1 日採録)

鈴木　真一（学生会員）

1979 年生．2001 年筑波大学第三学群情報学類卒業．現在，同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博士課程に在学中．オペレーティングシステム，情報セキュリティに興味を持つ．

新城　　靖（正会員）

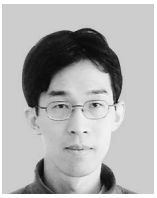
1965 年生．1993 年筑波大学博士課程工学研究科電子・情報工学専攻修了．同年琉球大学工学部情報工学科助手．1995 年筑波大学電子・情報工学系講師，2003 年同助教授．分散型オペレーティング・システム，並列処理，情報セキュリティに興味を持つ．1995 年情報処理学会山下記念研究賞受賞．博士（工学）．日本ソフトウェア科学会，ACM，IEEE CS 各会員．

光来　健一（正会員）

1975 年生．2002 年東京大学大学院理学系研究科情報科学専攻博士課程修了．同年日本電信電話株式会社入社，現在に至る．博士（理学）．オペレーティングシステム，ネットワークに興味を持つ．日本ソフトウェア科学会，ACM 各会員．

板野　肯三（正会員）

1948 年生．1977 年東京大学大学院理学系研究科物理学専門課程単位取得後退学．1993 年より筑波大学電子・情報工学系教授．計算機アーキテクチャ，分散システム，プログラミングシステム等に関する研究に従事．理学博士．日本ソフトウェア科学会，電子情報通信学会，ACM，IEEE CS 各会員．



千葉　　滋（正会員）

1968 年生．1991 年東京大学理学部情報科学科卒業．1993 年同大学院理学系研究科情報科学専攻修士課程修了．1996 年同専攻博士（理学）取得．同専攻助手，筑波大学電子・情報工学系講師，東京工業大学情報理工学研究科講師を経て，現在同助教授．言語処理系およびオペレーティングシステム等システムソフトウェアの研究に従事．日本ソフトウェア科学会，ACM 各会員．