コンピュータセキュリティ 11−12
（2000．9．8）

階層レベル構造を用いたUNIXセキュリティの強化法

井上清一† 下川徳之† 永瀬宏†

現在サーバを運営するためのオペレーティングシステムとしては，UNIXが主要な選択肢となっている．UNIXには，自分の所有するファイルに対して他者からの閲覧を制限する機能が標準的に備わっている．本稿はこの機能に，階層を考慮したアクセス規則を加えて運用するアクセス制御の形態を提案する．これにより，内部からのハッキングや誤操作などの防止に重点をおいたサーバの運用が可能となることを示す．

A method to strengthen UNIX security by introducing class level structure

KIYOKAZU INOUE,† NORIYUKI SHIMOKAWA† and HIROSHI NAGASE†

The UNIX is now the major choice as operating system to manage a server. The UNIX operating system has a function to prevent other people from reading one's file. This paper proposes how to manage a server by adding the access rule of the class structure to the usual access control function. It is shown that the control of the server which can prevent attack from the inside and mis-operation becomes possible by the method of this paper.

1. はじめに

UNIX系のオペレーティングシステム（以下，UNIXと述べる）では通常，アクセス制御リスト（ACL:Access Control List）を用いたアクセス制御機能を標準的に持っている[1]．これはファイルやディレクトリ等に対して個々にアクセス権限を設定する方式であり，設定は基本的に，サーバを設置して運営を行う管理者が，ユーザー間の関係を把握して手動で行う必要がある．一部のUNIXには，この設定を支援するソフトウェアが付加されている場合があるが，その支援内容はGUIに焦点を当てた操作性に関するものであり，管理者が手動で行っている部分を半自動化するシステムは，研究段階にある．ユーザー間の関係が複雑化した場合，管理者の考慮する内容は増加し，権限設定の時間の増加や設定ミス等の発生が考えられる．

以上のようなUNIXの現状から，セキュリティ設計を容易かつ正確に行うための"セキュリティ設計支援システム"が望まれている．現状のUNIXにおいても管理者の負担を軽減する工夫はいくつか存在しており，例えばUNIXが初めから持つファイルに対しては，UNIXの設計者が勧める標準的なアクセス権が自動的に設定されている．したがって，そのようなファイルに対しては，管理者が改めて権限を設定する必要は生じない．

しかしこの手法は，管理者が後に追加したファイルやプログラム群，及びユーザーが作成したファイル群に対しては適用できるものではなく，UNIXの設計者の意図が及ばない．特に前述の，UNIXが初めから持つファイルの権限を変更した場合は，その部分がシステムの弱点となる可能性を持つ．管理者はそのような変更に対して，プログラムとユーザー間の関連を調べ上げ，十分に検討を重ねなければならないという問題があった．アクセス制御リストでは，ファイル単位の細かなアクセス制御指定が可能であるという反面，その設定量はファイル数に比例して増加するため，この作業は容易ではない．

本稿ではこの問題に対して，SLA(Security Level Assignment)アルゴリズムを用いた階層レベルの自動設定手法[2]によって得られた情報を利用する，階層アクセス制御を導入することを提案する．これにより，主にサーバにアカウントを持つ人物による，内部からのハッキングや誤操作などの防止に重点をおいたサーバの運用が可能となることを示す．第2章では階層制御の概念となるBLP及び，UNIXにおける階層アク

† 金沢工業大学情報工学科，石川県
Department of Information and Computer Engneering, Kanazawa Institute of Technology, Ishikawa, 921 Japan

セス制御の位置づけについて述べ，第 3 章で UNIX に適用可能な実装方法について説明し，最後に階層アクセス制御の適用例と評価について述べる．

2. 階層アクセス制御

この章では階層アクセス制御が持つ基本的な性質と，現在の UNIX が持つ標準的なアクセス制御 (以下，ACL 制御と述べる) との位置付けについて述べる．

2.1 ACL 制御

ACL 制御によるアクセス制御では，すべてのファイルに細かなアクセス規則を設けることができる．しかし，アクセスを行うユーザーの区分としては 3 つしかなく，ファイルの所有者本人，所有者と同じグループに属するユーザー，所有者とは異なるグループに属するユーザーという区分でしか，分けることができない．したがって，複数のプロジェクトグループが単一のサーバを利用する運営を考えた場合，あるグループに対してはファイルのリードのみを許可したいが，別のグループに対してはライトも許可したいといった，グループによってアクセス権を変更する運営は，ACL 制御では実現することができない．

そこで，本稿では ACL 制御の持つ利点を生かした上で，後述する階層アクセス制御を取り入れることにより，実現可能な要求に幅を持たせる事を提案する．

2.2 Bell and LaPadula (BLP) モデル

Bell and LaPadula モデル (BLP モデル) は階層的アクセス制御を導入したセキュリティモデルであり，情報フローが一方向的である [3]．このため，レベルの上下関係で情報フローの有無が明確に判定できる利点がある．以下，本モデルの概要を述べる．

まず，コンピュータのアクセスに関わるユーザやファイル，プログラムなどをエンティティと総称する．また，プログラムやユーザなどのアクセスを行なうエンティティを主体 (Subject)，ファイルなどのアクセスされるエンティティを客体 (Object) とする．

全順序に限定した場合の BLP モデルは，いくつかのアクセス制御の規則に基づいて設計されている．具体的には，階層的なレベルが存在する中で，上位レベルに対しては書き込み，下位レベルに対しては読み出しのみ許可される．同レベル間においては，書き込み，読み出しともに許可される．例えば，主体 s が客体 o を読み出すときは，主体 s のレベルは客体 o のレベルよりも上位でなければならない．また，主体 s が客体 o に書き込むときは，主体 s のレベルは客体 o のレベルよりも下位でなければならない．

したがって，客体の読み出しは上位セキュリティレベルの主体に限定され，これは BLP の情報フローに準じたフローであるため，情報リークは発生しない．また，下位セキュリティレベルの主体は上位セキュリティレベルの客体を読み出せないため，下位セキュリティレベルの主体への情報リークは発生しない．この性質により，上位レベルのデータの機密性が確保される．この性質を図式化したものを図 1 に示す．

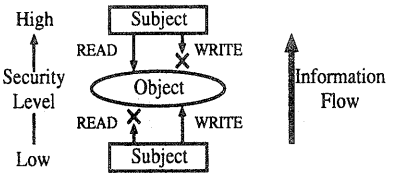

図 1 Bell and LaPadula モデル
Fig. 1 Bell and LaPadula model.

2.3 階層アクセス制御の位置付け

階層アクセス制御は，ACL 制御と共存させることができる．階層アクセス制御と ACL 制御はそれぞれ，次の順序で動作する．

(A1) ユーザーからの，ファイルへのアクセス要求が発生する．
(A2) ACL 制御が動作し，予め管理者が設定した ACL 情報に適合しないアクセスは，この時点で中断される．
(A3) A2 の制御をパスした場合は，階層アクセス制御が動作する．予め管理者が設定した階層レベル情報に適合しないアクセスは，この時点で中断される．
(A4) A2，A3 のどちらの制御にもパスした場合，アクセス処理は成功する．

したがって，これらのアクセス制御は AND 条件で動作し，片方のアクセス制御をパスしなかった場合は，そのアクセス要求は受け付けられない．

3. 実　　装

本章では，階層アクセス制御を UNIX に実装するにあたって用いた，具体的な手法について述べる．

3.1 タイムスタンプの監視によるアクセス検出

ファイルへのアクセスを検知する手法のひとつとして，タイムスタンプを利用した方法が考えられる．この制御構造を図 2 に示す．

まず，任意のタイミングでユーザからファイルへのアクセス要求が発生する (①)．ユーザーからの要求を受け取った UNIX はファイルへのアクセスを試み，ACL 制御でチェックされる (②)．この間にアクセスコントロールの制御プログラムは，各々のファイルが

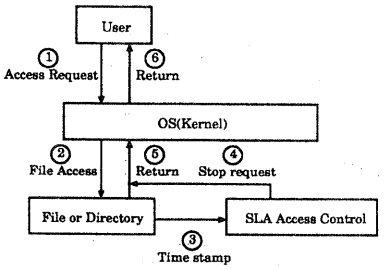

図 2 タイムスタンプを用いたアクセス監視
Fig. 2 The access monitor which uses time stamps.

持つタイムスタンプを常に監視し，その内容に変化が生じた場合，UNIX がアクセスを試みたと判断する．タイムスタンプはアクセスの種類がリード，ライトのどちらであっても書きかわり，アクセス検知に利用できる（③）．管理者の設定したレベル条件に適合しないアクセスと判断された場合は，この処理に割り込んで中断させる（④）．このチェックにもパスした場合は，ファイルに対する処理が継続される（⑤⑥）

タイムスタンプ情報の取得には i-node 情報を用いる[4]．この情報は UNIX では容易に取り出すことができ，last access の部分に示される値がアクセスによって変化を起こすという特徴がある．変化の一例を，図 3 に示す．

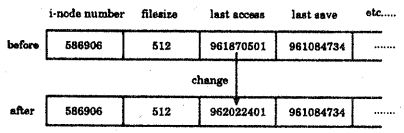

図 3 i-node 情報の変化例
Fig. 3 The change of the i-node information an example.

しかし，この監視手法は ACL 制御と並行に動く構造であるため，ファイルの監視に十分短い間隔を設定しない場合，アクセスの中断処理が間に合わない事態が考えられる．

3.2 クォータプログラムの拡張によるアクセス検出

ファイルサイズを監視してアクセス制御を行うクォータプログラムを，管理者の設定した階層レベル情報を監視するように拡張する．この制御構造を図 4 に示す．

まず，ユーザからファイルへのアクセス要求が発生する (①)．その処理はまず，ACL 制御でチェックされ，結果が得られる (②③)．ACL 制御をパスした場合，引き続いて拡張を行ったクォータプログラムによるチェックが行われ，管理者の設定した階層レベル情報に適合するかどうかを判定する（④）．このチェックにもパスした場合のみ，ファイルに対する処理が継続される（⑤⑥⑦⑧）．

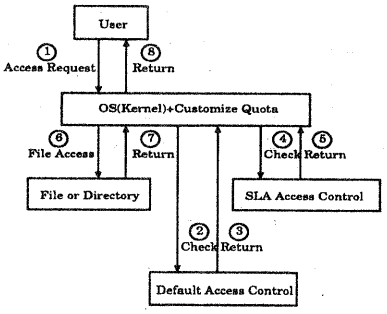

図 4 クォータプログラムを用いたアクセス監視
Fig. 4 The access monitor which uses quota program.

3.3 アクセス検知手法の比較

それぞれのアクセス検知手法は，UNIX の提供形態によって，適用が難しい場合がある．これらの手法の特徴を表 1 に示す．

表 1 実装手法の比較
Table 1 The comparison of the implementation methods.

	タイムスタンプ	拡張クォータ
ソースプログラム	必要なし	必要
監視時の負荷	多い	少ない
監視範囲	狭い	広い
システム依存	OS の設計に依存	クォータの設計に依存
動作の確実性	監視間隔に依存	一定（確実）

実装方式は大別して，独自のプログラムによる割り込み処理または，UNIX のオリジナルソースプログラムの拡張となる．現在提供されているいくつかの UNIX では，ソースプログラムも合わせて提供されているため，本稿では UNIX 自身の機能として無理なく実装が可能な拡張クォータ方式を優先して選択した．なお，ソースプログラムが提供されていない UNIX に関してはタイムスタンプ方式を用いるが，UNIX のアクセス制御と並行して監視を行うため，十分に短い間隔でのアクセス監視が必要と考えられるほか，動作の確実性に関しても拡張クォータ方式と比較して劣ると考えられる．

3.4 階層アクセス制御の適用範囲

階層アクセス制御プログラムは，UNIX の最大権限者である Root ユーザー (以下，Root と述べる) を除く範囲で動作する．Root はこのアクセス制御を開始及び停止させることのできる唯一のユーザーであり，階層アクセス制御の動作を開始した場合でも，その制御が Root に対して適用されることはない．また，Root

以外のユーザーは，このアクセス制御を独自に解除したり，中断することはできない．UNIX では原則として，Root によって開始されたサービスは，他のユーザーが中断することができないように設計されている．この構造と Root の位置づけを，図 5 に示す．

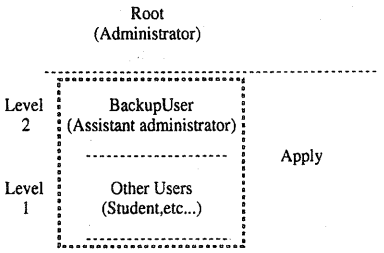

図 5 階層アクセス制御の適用範囲

Fig. 5 The target range of applying the class access control.

図 5 では，UNIX に存在するユーザーを Root とそれ以外の区分に分け，Root を除くすべてのユーザーに階層アクセス制御が適用されることを示している．なお，この図で使用されている BackupUser は，階層アクセス制御を利用した場合に作成が可能となる特殊なユーザーであり，4.2 に運用の一例として詳細を述べる．

4. 運 用 例

この章では，階層アクセス制御の適用例と，どのような問題が改善されるかについて述べる．

4.1 下位権限に位置する書類の改竄防止

UNIX では，権限の大きな人物は下位の人物の所有するファイルへの読み書き両方の権限を有する．しかしこの強い権限が悪用された場合，書類の改竄を行うことが可能となる．

このような状況における階層アクセス制御の効果は，権限の大きな人物による下位階層のファイルの書き換え防止である．階層アクセス制御においては情報フローの一方向性という特徴から，下位のユーザーから上位ユーザーに送られたファイルは操作できるが，下位のユーザーが所有するオリジナルファイルの操作は不可能である．大きな権限を持つ人物がファイルの改竄や破棄よって不正を隠蔽する構造が実社会に存在するが，本稿の技術はこれを防止する．この構造を図 6 に示す．

図 6 では，レベルを 1 に設定されたユーザーが，ファイルの作成や複写を行っている状況を想定している．そのとき，これらのファイルをレベル 2 のユーザーに送付することもできる．しかしレベル 2 のユーザーは，送られてきたファイルに対しては読み書き双方の権限を持つが，レベル 1 のユーザーが所持するオリジナルのファイルを編集することはできない．これは前述の原則である，情報の一方向性に沿っている．

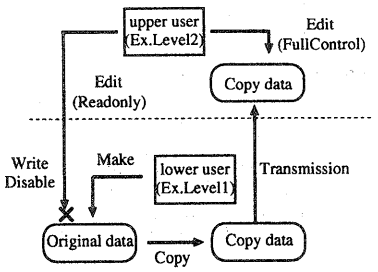

図 6 オリジナル書類の改竄防止構造

Fig. 6 Preventing illegal renewal of the original document.

4.2 準管理者権限の作成

UNIX で危険性が高いと考えられる作業の一つに，Root を用いる業務がある．しかし業務内容によっては，Root は過剰な権限となる場合があり，例えばファイルのバックアップ業務は単純な作業ではあるが，すべてのファイルを読みとる必要があるため，Root を必要とする．

しかし，Root ではどのような作業も可能であるという性質上，この状態での誤操作はシステムに致命的なダメージを与えることが多い．ある人物に依頼したい業務がバックアップだけであるとしても，現在の UNIX では Root を与えなければ不可能である．

このような状況における階層アクセス制御の効果は，準管理者の作成である．ここでは例として，大学においてサーバの運営に学生が関与する状況を考える．サーバの Root は，基本的にネットワークを熟知した教員によって管理されるが，管理業務にボランティアとして参加する学生にも，バックアップ業務の依頼のために Root が与えられている．この場合，参加する人物全員が Root の危険性を十分に理解している事が望ましいが，操作の経験が浅い者もおり，十分な教育を行わないまま Root を与えるには，危険が生じる．

しかし階層アクセス制御では，情報のフローが一方向である．この点を利用すると，最上位の階層としてレベルを設定されたユーザーは読みとりのみが可能であり，バックアップ業務に支障の無いユーザーとしての役割を持つことができる．下位レベルへの書き込みは階層アクセス制御では許可されないため，バックアップ中の誤操作によって，ファイルを破損する事が

なくなる．この構造を図7に示す．

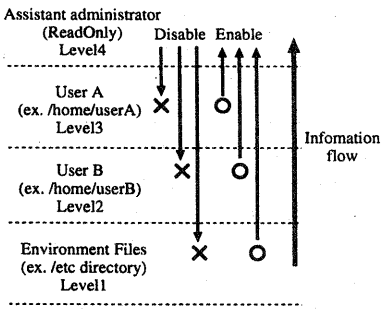

図7 準管理者の権限

Fig. 7 The authority of the assistant administrator.

4.3 ハッキング行為の検知

前述のような準管理者の作成を行った場合，Rootユーザーへの移行(Switch Userと呼ばれる)を準管理者に限定させることで，ハッキング行為の発見を助けることができる．侵入者は通常，Rootへ移行すると自分の侵入の痕跡を消去する．この対象となるのは，UNIXが自動的に作成するログファイルであり，Rootの場合はこのログファイルが編集可能となる．

しかし，準管理者を経由しなければRootへ移行できないという原則を設定することによって，侵入者は一旦，ファイルの変更を一切行うことのできない準管理者の権限を経由する．このことは，侵入者が準管理者への移行に成功したとしても，Rootに移行するまでの時間が余分に必要となることを意味する．この時間を増やすことは，正規の管理者が侵入者の存在を察知できる機会を増すことにつながる．この構造を図8に示す．

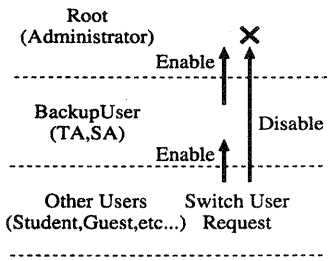

図8 SwitchUserコマンドの許可構造

Fig. 8 The permission structure of the SwitchUser command.

4.4 計算機を利用した授業への適用

大学における，計算機を利用した授業への適用が考えられる．UNIXの初期ACLでは，自分と同グループのユーザーであれば，それらのユーザーが所持するファイルは閲覧が許可されている．ユーザーがこの設定に修正を加えることもできるが，大部分のユーザーは初期設定のまま利用を続ける．

このような場合に考えられるのは，過去の演習内容が参照されてしまう危険性である．すでに演習を終えた人物のファイル（上位の学年等）が閲覧又は複写されることで，不正なレポートが作成され，正常な授業を行うことができない．

この問題の説明に適切な例として，実際に本大学の情報工学科で行われている演習を考える．この演習はUNIXを利用する講義として，3回生を対象として毎年冬に開講される．演習中に作成されたファイル群はユーザーが意図的に消去しない限り，3回生が卒業してアカウントが消滅する約1年後まで計算機内に存在し，2回生が次年度の講義に容易に利用できる可能性を発生させる．

このような状況における階層アクセス制御の効果は，学年を越えた閲覧権限の取り消しである．学年を単位としてレベルを設定することで，下位レベルに設定された学年は上位学年のファイルの閲覧ができなくなる．この構造を図9に示す．

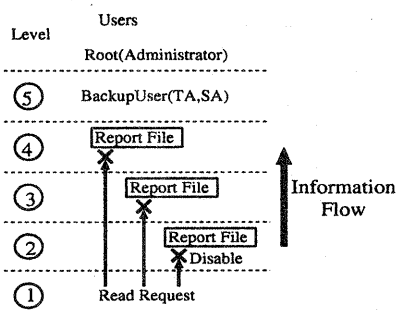

図9 レポート複写の防止構造

Fig. 9 Preventing illegal copy of the report.

5. 評　価

この章では，ACL制御と階層アクセス制御を比較し，階層アクセス制御を利用した場合の特徴について述べる．

5.1 アクセス制御方式の比較

ACL制御と階層アクセス制御の比較を，表2に示す．

前章等で述べたように，階層アクセス制御を適用することで，ACL制御では実現が不可能なアカウントの生成や，複数グループ間の関連を設定することが可

表 2 アクセス制御方式の比較
Table 2 The comparison of the access control methods.

	ACL	階層
準管理者アカウントの作成	不可能	可能
下位書類の改竄防止	不可能	可能
ログ改竄の防止	不可能	可能
一方向へのデータフロー	不可能	可能
複数グループ間の関連設定	不可能	可能
権限の機械的自動設定	無し	あり (SLA)
導入コスト	不要	必要
処理時間の遅延	無し	あり
トレーニング時間	不要	必要

能となる．しかし，階層アクセス制御は導入コストや処理時間の遅延が発生するほか，管理者のトレーニング時間も余分に必要となる．

5.2 処理時間の比較

階層アクセス制御は，ACL 制御と直列に動作するため，処理の遅延をもたらす．階層アクセス制御を組み込んだ場合との速度比較を，表 3 に示す．比較内容は一定数のファイル（個）を読み書きした場合の所要時間（秒）である．

表 3 アクセス制御時の遅延時間
Table 3 Delay time induced by various access controls.

ファイル	ACL	ACL +階層
1000	50	57
2000	100	115
3000	151	173
4000	201	232
5000	251	290
6000	301	348
7000	351	406
8000	402	465
9000	452	523
10000	502	581

この制御プログラムの実装は，FreeBSD4.0(ソースプログラムが提供されている UNIX) を用いて行い，現在も実装を継続中である．現段階では ACL 制御と比較して，約 15〜18%の遅延が確認されている．

6. むすび

エンティティー間に関係が与えられたとき，階層アクセス制御を行うためのレベルを割り付ける方法は，SLA アルゴリズムとして知られている．SLA は逆 SLA や拡張 SLA アルゴリズムなどと組み合わされて用いられ，管理者は機械的に設計されたいくつかのレベル設定情報の中から，最も適していると思われるものを選択する．

これに対して本稿では，SLA アルゴリズムで設定された階層レベル情報を用いて，UNIX に階層アクセス制御を実装する手法を提案している．各実装手法は UNIX の提供形態によって使い分けられる．ソースプログラムの提供状態によって実装手法を分け，提供形態に応じた実装を提案した．

階層アクセス構造を用いた制御の実施により，UNIX は ACL 制御を生かしつつ，複数グループ間の関係を設計できる構造等を持つ．この点に着目して，本稿の最後には UNIX が稼働している現場での，階層アクセス制御の適用部位と改善される問題を述べた．このような階層を使ったアクセス制御の実施により，ユーザー間の関係を設計する上で柔軟性が生まれる利点が得られることを示した．

謝　辞

本研究の一部は (財) 電気通信普及財団の研究助成金によって行われたものです．ここに深く感謝申し上げます．

参考文献

1) 佐々木良一, 宝木和夫, 櫻庭健年, 寺田真敏, 浜田成泰: インターネットセキュリティ基礎と対策技術, オーム社 (1996).
2) 永瀬宏, 井上清一, 四七秀貴: 階層的セキュリティレベルの自由度を用いたアクセス権設定法, 情報処理学会論文誌, Vol. 41, No. 8, 掲載予定 (2000).
3) T.Araki, T.Morizumi, H.Nagase, T.Takenaka and K.Yamashita: Security Level Assignment By Graph Analysis, *Issues on Cryptography and Information Security , IEICE Transactions*, Vol. 74, No. 8, pp. 2166–2175 (1991).
4) Maurice J.Bach 著: 坂本文, 多田好克, 村井純　訳 : UNIX カーネルの設計, 共立出版株式会社 (1991).