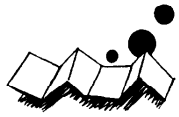
解　説

オペレーティングシステムのセキュリティ†



関野　陽††　久保秀士†††

1. はじめに

情報処理システムによる情報の処理・管理が一般化した結果，計算機内に管理される情報や計算機間で転送される情報に対するセキュリティの問題が大きくクローズアップされるようになった．情報は一般に計算機内に長期間蓄積される関係で，計算機内ではオペレーティングシステム (OS) のもとでさまざまなセキュリティ侵害要因から確実に保護することが必要になる．このようなことから，OS では，蓄積情報の保護を目的として各種のセキュリティ機構を用意している[1)~3)]．本稿は，このようなセキュリティ機構の考え方，実現法，具体例などを述べるものである．

一般に，計算機内の蓄積情報に対するセキュリティ侵害は，**表-1** に示すように，①機密情報の漏洩，②蓄積情報の不正化，③情報サービス停止という形で表れる．①は情報を見る資格のない者に情報を見せてしまうとか，情報を見る資格のある者が見た情報を資格のない者に横流しするなどといった原因で生じる問題である．②は情報の内容を変更する資格のない者に不正変更を許してしまうとか，本来ならば正当な情報変更がたまたま複数ユーザによって同時に行われたためにその内容に矛盾を来すなどといった問題である．③は本来の情報サービスが予期せぬシステムダウンやシステム破壊等によって利用不能になるといった問題である．このようなセキュリティ侵害は，計算機ユーザの偶然行為によって起こるばかりでなく，情報の盗用・改ざん・破壊等の悪意の意図的ユーザ行為，OS のバグ，計算機ハードウェアの誤動作などによって引き起こされる．OS のセキュリティ機構は，これらのセキュリティ侵害要因に対して計算機内部で各種の確実な制御を行わなければならない．

† Security in Operating Systems by Akira SEKINO (Computer Engineering Division, NEC Corp.) and Hidehito KUBO (C & C Systems Research Laboratories, NEC Corp.).

†† 日本電気(株)コンピュータ技術本部
††† 日本電気(株)C & C システム研究所

表-1　セキュリティ侵害の種別とその発生原因

侵害の種別	主要な発生原因		
機密情報の漏洩	無資格者による不当な情報アクセス，有資格者による不当な情報横流し	システムバグ	ハードウェア障害
蓄積情報の不正化	無資格者による不当な情報変更，情報の同時更新に起因する矛盾発生		
情報サービス停止	システムダウン，システム破壊		

一方，蓄積情報のセキュリティを計算機室運営のレベルで考える場合には，OS では対処不可能な問題も少なくない．物理的に，情報記憶媒体を盗み出すとか破壊するとか，磁気外乱等で計算機に誤動作を引き起こすといったものである．一般に，計算機室の施錠，計算機室の電磁遮蔽，操作員保守員の資格検査，保管情報記憶媒体の管理，各種災害対策等で対処しているが，これらの点については一応本解説の範囲外とする．

以下では，OS のセキュリティを述べるにあたり，まずセキュリティの考え方を明らかにするためにセキュリティのモデルやそれにまつわる諸問題を第**2**章で扱う．次に，OS におけるセキュリティの各種実現機構を第**3**章に述べ，引続き代表的な実際のシステム例を第**4**章に解説する．そして，最後に，OS のセキュリティの在り方などについて第**5**章で触れる．

2. セキュリティのモデル

実際のシステムを設計するには，「このようにすればこの種のセキュリティを必ず保証できる」といったセキュリティモデルが必要である．幸いこの10数年来の研究により，各種のモデルが提案されている．以下では，まずこれらのモデルを概観し，次に最も重要と思われるアクセス制御モデルを説明する．そして，セキュリティ制御上の問題とその解決方向を述べる．

2.1 セキュリティモデル

一般に，セキュリティのモデルと呼ばれるものは，

表-2　セキュリティモデルの分類

モデルの種類	各種モデルの名称
アクセス制御モデル	アクセス制御表モデル (権限リスト，ケイパビリティ) UCLA カーネルモデル 権限譲渡グラフモデル Bell-LaPadula モデル(原型)
フロー制御モデル	情報フローモデル Bell-LaPadula モデル(改良型)
推論制御モデル	フィルタモデル 強依存モデル 束縛条件モデル

①アクセス制御モデル，②フロー制御モデル，③推論制御モデルに大分類される．それぞれには，**表-2** に示すように，各種の詳細モデルが含まれる[4]．ここでは詳細は省略することとし，これらの3種のモデルの位置付けと狙いを示すと次のようになる．

① アクセス制御モデル　ユーザ（一般に主体と呼ぶ）ごとにどの情報記録物（一般に客体と呼ぶ）にはどのようなアクセス（参照，更新等）を許可するかを定め，この規則通りの情報共用とこの規則に反した不当アクセスの阻止を図ろうとするもの．

② フロー制御モデル　情報記録物に機密度のレベル（極秘，部外秘，一般等）を設定し，情報記録物間での情報転写の際に機密度のより低い情報記録物への転写を禁止することにより，不当な情報横流しを防止しようとするもの．

③ 推論制御モデル　データベースからの直接的な情報読出しが禁止されている場合，その情報に関連した統計的諸性質（総数，平均，最小等）に関する正当な問い合わせを重ねることによりその情報の値を推論するといった試みを阻止しようとするもの．

以上から知られるように，これらのモデルはそれぞれ別種のセキュリティの実現を狙うものであり，実際のシステムでは併用が望まれる．しかしながら，現段階では，一部の商用システムがアクセス制御モデルを採用し，米国の軍用システムがアクセス制御モデルにフロー制御モデルの併用を始めたといったところである[4]．推論制御モデルは未だ概して研究段階と思われる．このようなことから，以下ではアクセス制御モデルを中心に解説を進める．なお，実際の OS では，システムとしての全体的なセキュリティ向上のために，上述の制御に加えて更に下記の手法を使用している．

① 情報記録物（ファイル）上の情報の暗号化
② 情報同時更新時の矛盾排除のための排他制御
③ システム高信頼化技術（ハード，ソフト）
④ 破壊された情報の復元のためのファイル障害回復
⑤ 機密情報へのアクセス記録等に基づくシステム監査

2.2 アクセス制御モデル

アクセス制御モデルでは，主体から客体への正当なアクセスは**図-1** に一例を示すようなアクセス制御表として規則化される．一般には，後述のように，主体としてはユーザ，ジョブ，プロセス等が，客体としてはファイル，プログラム，セグメント，入出力装置等の種々のものが，アクセス制御の各場面に応じて設定される．図-1 の例はファイルシステムに対してアクセス制御を適用した場合であり，各ユーザが各ファイルに対してどのようなアクセス権（参照R，更新W，アクセス権種別の設定変更A，いずれの権限もなしは空欄）を持つかを規定している[2]．アクセス権種別の設定変更権はファイルの所有者に与えられるものであり，ファイル所有者はこの権限により各ユーザの当該ファイルへのアクセス権種別の設定やその変更を行うことができる．一般に，アクセス制御表で定められるアクセス規則を変更する権限は往々にしてセキュリティ侵害の原因となりやすいので，特別な保護コマンド（主体客体の追加，権限設定，権限変更，権限譲渡用など）を介して行うようにする．

	F_1	F_2	F_3	F_4	F_5	F_6	…
ユーザ 1	R,W,A		R		R	R	
ユーザ 2					R,W,A		
ユーザ 3	R		R	R,W		R,W,A	
ユーザ 4		R,W,A	R,W		R	R,W	
⋮							

F：ファイル，R：Read，W：Write，A：Alter

図-1　アクセス制御表モデル

このようなモデルを忠実に実現するには，主体が客体に対してアクセスを試みるごとに，そのアクセスがアクセス制御表上で許可されていることを間違いなく確認することが必要になる．この役割をするソフトウェアをアクセスモニタと呼び，OS 中核部に位置する後述のセキュリティカーネルで実現する．一般に，セキュリティカーネルは機能検証技術等で正しく動作することを確認することが必要である．なお，アクセス制御表の記憶の仕方には，①権限リスト方式，②ケイパビリティ方式の二方式がある[3]．図-1 で言えば，前者は制御表の内容を縦方向に見て客体ごとに許可された主体名とその権限をまとめて記憶する方式であり，後者は制御表の内容を横方向に見て主体ごとに許可された客体名とその権限をまとめて記憶する方式である．

本モデルは，主体と客体の間のアクセス関係の制御を統一的に扱う一般性の高いモデルであり，多くの理論的研究やシステム実現が進められている．その過程でいろいろなセキュリティ制御上の問題が明らかにされ，モデルとしての改善が進められている．次節にいくつかの点について述べる．

2.3 セキュリティ制御上の問題

以下では，セキュリティ制御で問題になる点の中から6つばかりを取り上げ，それぞれについて簡単に説明するとともにその解決方向を述べる．

（a） ユーザ身元確認　どのようなセキュリティ制御でも，情報へのアクセス主体としての計算機ユーザを誤りなく確認しうることが前提である．ユーザ本人の確認には，各種の個人識別技術が検討されている[3]．

（b） アクセス制御規則の変更権限　規則変更の柔軟性と濫用防止のトレードオフ，権限譲渡に伴う権限伝搬や拡大の危険性等の問題がある．アクセス権種別の設定変更権にユーザ間の階層性を導入する方法[3]や権限譲渡を制限する方法[5]等が考えられている．なお，アクセス制御規則の変更は監査用記録の採取対象となる．

（c） 軍用機密管理への応用　上述のような主体客体間のアクセス関係に加えて，客体となる情報に機密レベルや内容分類を考慮することが必要になる．このため，アクセス制御の精密化やフロー制御の概念を取り入れた Bell-LaPadula モデルと呼ぶものが軍用システムのセキュリティモデルとして開発されている[4]．

（d） トロイの木馬　外部購入のプログラムなどを自分のプログラム実行の一部として使用する場合に，外部購入プログラムが自分のアクセス権を利用して各種のセキュリティ侵害（情報破壊，情報横流し等）を行う可能性がある．プログラムごとにきめ細かく必要最小限のケイパビリティ（後述）を付与する方法[3),5]やフロー制御技術の適用[4]等が考えられている．

（e） データベース情報の推論制御　データベース管理システムがユーザに対して統計情報しか提供しないように制御を行ったとしても，実際には個別情報の推論が可能となる場合が少なくない．各種の推論制御モデルの研究が進められている[4]．

（f） システムのバグや障害　アクセスモニタのバグやハードウェア障害は重大なセキュリティ侵害の原因となる．前者に対してはアクセスモニタのカーネル化とその機能検証が，後者に対してはハードウェアのフォールトトレラント化が考えられている[3),4]．

3. セキュリティの実現機構

次に，以下では，実際のシステムの OS，アーキテクチャのレベルで実現されている具体的なセキュリティ機構について，モデルと関連づけつつ解説する．

3.1 情報へのアクセス過程とセキュリティ

計算機内の蓄積情報に対してユーザがアクセスを行う場合のシステムの処理過程を見ていき，途中でどのようなチェックが可能か概観してみよう（表-3 参照）．

システムを使用しようとするユーザはまず身元確認を受ける必要がある．身元確認のための技術にはいろいろなものがあるが，システム側では，パスワード/暗号化法を総当り的に試みてくることの阻止など，第4章第1節でも例を示すような OS 面での各種の実

表-3 情報へのアクセス過程と保護

アクセス過程	保護の種類	主体	客体	代表的な保護手段
システム使用開始	身元確認	ユーザ	システム	パスワード，プロファイル
ファイルへの最初のアクセス	アクセス制御	ユーザ，ジョブ	ファイル	権限リスト
プログラム実行	アクセス制御	プロセス・実行環境対	ファイルレコード，セグメント，主記憶領域	ケイパビリティリスト

際的配慮が必要である.

身元確認に合格したユーザはシステム内に受け入れられる. 汎用計算機の OS では, 一般にユーザプロファイルを管理している. ここには, 使用を許可されたユーザ全員について, 属するセキュリティ上のグループとその中での当人の位置づけ, 身元確認の方法, 実行可能な操作などが登録されている. ユーザは, この内容に基づいて権限を付与されたジョブを開始できる.

開始されたジョブは, プログラムの実行, データへのアクセスを要求する. 一般に, これらの実行対象はファイルとして存在しているが, これに対する最初のアクセス (オープン) 時に, ファイルごとにもつアクセス権限リストによるチェックを受け, 同時に, より詳細なアクセス制御情報が計算機内へ取り込まれる. 以後のプログラム実行においては, 計算機はこの情報に基づいて, 主記憶のアクセスのたびごとなどの細かいタイミングでアクセスの正当性のチェックを行う.

以上の過程においてアクセス違反が検出されると, 記録がとられ, そのアクセスを無効として続行するか, 処理を強制終了させるかなどの処置がとられる.

3.2 セキュリティ侵害の試み

セキュリティの実現機構については次節で述べるが, 一般の商用システムでは種々の試みにもかかわらずセキュリティ上の抜け道が存在することに触れておく.

積極的にセキュリティ侵害を試みた例として, IBM 社の OS/VS 2-2 の開発時の報告がある. これによると, それ以前のバージョンが内蔵していた 7 種類の弱点 (ユーザ領域内に置かれたシステムデータの問題等) が抽出され, いろいろな対策が施されている.

また, 後述のように仮想計算機システムでは一般の OS より高いセキュリティが実現可能と言われており, IBM 社の VM/370 に侵害を試み, 従来の OS では可能であった侵害の多くが阻止されることが確かめられている[6]. ただし, 入出力命令の実行の部分に一般の OS よりも弱いところが存在したとのことである.

3.3 セキュリティ機構

以上のような背景の下で, 高いセキュリティを実現するために, どのような機構が提案され, また採用されているかなどについて述べる (表-3 参照).

(1) 隔離と共用

計算機システムを 1 人または同じ目的を持つ一つのグループだけで専有して使うのであれば, セキュリティ侵害は通常起こらない. 実際, 高度の機密を扱う米国の軍関係では, 機密度のレベルに応じて時間帯を分け, 使用後は完全に初期状態へ戻してから次に渡す形がとられていた[9]. これに近い隔離性を実現し, かつ, 計算機の使用効率をあまり落さずにすむシステムとして仮想計算機が知られている. 仮想計算機とは, 一つの実システム上に複数の計算機システムを仮想的に実現し, 並列実行を可能にしたものである. 原則として, 仮想計算機間は完全に隔離されている.

一方, 情報の共用は計算機利用システムの本質的な狙いの一つであり, これを禁じることは角をためて牛を殺すことになり兼ねない. 一般的には, 共用を許しつつセキュリティも高める必要があり, 各主体は各時点で必要最小限の権限だけを持つべきであるという「最小権限の原理」が要請されてくる[3].

(2) 主体の選定

保護された客体へアクセスを行う主体としては, 第 3 章第 1 節で述べたように, システムの最も外側ではユーザ個人またはそのグループが意識される. 実行段階では, ジョブに付与された権限を引き継ぐ実体としてのプロセスが, 主体として考えられるべきである.

しかし, セキュリティの観点からは, プロセスが識別されるだけでは不十分で, 同一プロセスでも実行中に権限の拡大・縮小ができなければならない. たとえば,「トロイの木馬」の問題に見られるように, ユーザプロセス内で OS モジュールを実行し, 次には警戒を要する外部購入のプログラムを実行させるような場合などは, このような場合の一例である. すなわち, 主体はプロセス p とそれが実行中の実行環境 e の対 (p, e) で表現されるべきである.

実行環境として, 最も初期から広く採用されているのが実行モード (モニタモードとユーザモード) である. モニタモードではすべての種類の命令を実行でき, また広い範囲の主記憶領域へアクセスできる. ユーザモードでは特権命令は実行不可であり, 限られた範囲の記憶域へしかアクセスできない.

実行モードの概念をさらに拡張し, きめ細かくしたのがリング保護方式である. Multics において提案され, 現在では多くのシステムで 4～8 レベルのリング保護が, 採用されている. 主体の属すリングは実行中のプログラムモジュールの属性で定まる.

リングよりもダイナミックに, たとえばどのプロシージャから呼ばれて実行中かまで意識して「最小権

限の原理」を忠実に実現しようとするのがドメインである．ドメインは，アクセスしてもよい客体の集りを直接的に規定したものである．小型機の IBM システム/38，大型機の ACOS，後述のマイクロプロセッサ iAPX432 等の製品でも実現されている．

（3） 客体の選定

保護を受けるべきアクセス対象である客体についても，大まかな区分けからきめ細かい区分けまでさまざまな捉え方が可能である．ここでは，実/仮想記憶上の情報だけに対象を絞ることにする．

仮想記憶の採用されていない初期においては，プロセス単位にアクセス可能領域を設定するという初歩的な保護機構が使用された．IBM 360 のメモリキー方式，NEAC 2200 の境界レジスタ方式などが代表的である．特に，前者の場合には，2 KB のブロック単位での不当参照の制御も可能であった．仮想記憶の登場に伴って，このようなプロセス間の隔離（多重仮想空間方式）の他に，セグメンテーション方式の場合にはプログラムまたはデータの論理的なひとまとまり（セグメント）を単位として，個々に許されるアクセスの種別(参照，更新，実行)のチェックが可能となった．

更には，セグメントを一般化したオブジェクトという概念がある．オブジェクトとは，記憶システム上のすべての情報を統一的に扱うために提供される情報格納容器であり，内容に応じたタイプが割り当てられ，そのタイプで定義されたアクセス種別のみが受け入れられる．オブジェクトの概念は，ソフトウェア工学の抽象データタイプの概念とも結びついて，最近特に広く研究され，製品にも反映されてきている．

（4） プロテクション機構

主体客体間のアクセス関係を管理する方式としては，第2章で述べたように権限リスト方式とケイパビリティ方式がある．ファイルを客体とする場合には上述のように権限リスト方式がとられているが，実行制御の場面ではほとんどケイパビリティ方式が採用されている（表-3 参照）．ケイパビリティとは，アドレス空間における客体の存在位置，客体へのアクセス権種別等を記述する記述子で，これを所有する主体による客体へのアクセスを可能とする．たとえば，多重仮想空間の実現では，プロセスごとに用意されるセグメント記述子がケイパビリティに相当する．上述のオブジェクトを客体とし，ケイパビリティの集合でプロセスの実行環境としてのドメインを動的に制御する方式の例が後述の iAPX 432 である．この場合のアクセス制御表は OS で生成されるが，リング保護の場合におけるリング間の制御の移動では，アクセス制御表はハードウェアで実現されていると考えることができる．

概念的に優れた方式であっても，完璧に実現されないとセキュリティは損われる．OS は非常に複雑なシステムであり，完全なものを作るのは難しい．完全であることを検証するのはさらに難しい．少なくともセキュリティ実現の基本機構だけは完全である必要があり，これが保証されていれば安全は守られる．この部分だけをできるだけ小さくひとまとまりに作り，最も高い特権レベルを与えるという実現法が考えられる．この基本部分をセキュリティカーネルと呼び，Hydra[5] など多くのシステムで実現の試みがなされている[7),8)]．

4. 実際のシステム例

高度なセキュリティを狙ったシステムの具体例を簡単に紹介する．ここで述べるもの以外にも多数の例があるが，これらについては文献8)を参照されたい．

4.1 Multics

Multics は，情報共用の場を提供することを目指して MIT で研究開発されたシステムであり，共用と表裏一体をなすセキュリティの対策については，深い考慮が払われている[3)]．

ログイン時の身元確認はパスワードで行うが，パスワード登録ファイルは暗号化されており，また，端末から入力された時には印字をしない．ログイン時にはそのユーザの前回のログイン時刻と端末番号を出力して，他人の不正使用をチェック可能にしている．

Multics の保護機構では，アクセスの主体は基本的にはプロセスとリングの対であり，客体はセグメントである．プロセスの識別は，上述の身元確認に基づいてつけられた識別子によってなされる．すべてのセグメントはアクセスが開始されるまでは，ファイルとして木構造のディレクトリにより管理されている．ディレクトリにはファイル生成時に登録されたアクセス権限リストがあり，階層的に下位にあるディレクトリとファイルについて，アクセス可能なユーザ識別名と許されるアクセスの種別（参照，更新，実行）が規定されている．

アクセスが開始されると，ファイルはセグメントとして仮想記憶上に取り込まれる．許されるアクセスの種別はセグメント記述子に登録され，以後メモリ参照

のたびごとにハードウェアでチェックされる．前述のようにセグメントは情報の論理的な単位であり，特にMultics では多重仮想空間を実現しているばかりでなく，共用されるセグメントへのアクセス権種別を主体であるプロセスごとに変えることができる（セグメント記述子をプロセス別に持つ）．更に，各セグメント記述子は参照，更新，実行の各々に関するリング番号（0〜7 のいずれか）を持つ．リング番号は若いほど権限が強く，実行中プロセスのリング番号が，対象セグメントの今実行しようとしているアクセスについてのリング番号と等しいか小さければそのセグメントへのアクセスが許可される．リング間の制御の移動はコール命令で行われるが，ゲートとして登録された入口以外からは入れなくしてあり，呼ばれた側はコールの正当性チェックを確実に行いうる．これらの機構によって階層的なきめ細かい保護が実現されている．

Multics は基本的にセキュリティに向いたアーキテクチャと OS を持つため，これをベースに，より安全なシステムを構築しようとする試みがなされた．一つは米国空軍，他は MIT 自身のプロジェクトである．いずれもセキュリティカーネルのアプローチにより必要最小限の OS 機能だけをリング 0 に残すように再構成し，この部分の安全性を検証可能にしようとしたものである[8]．MIT の試算では 54 K 行のリング 0 コードを 26 K 行まで減らしうると見込まれていたが，両プロジェクト共実行には移されなかったようである．

4.2 仮想計算機と KVM/370

仮想計算機の隔離性はセキュリティに向いており，前述のようにこのことはある程度実証されている．

仮想計算機の代表的なものに IBM 社の VM/370 があり，多くのユーザで使用されている．VM/370 では，実マシン上で仮想マシンモニタ（CP）が走り，この制御の下にホストと論理的にはまったく同じ機能を持つ（構成は異なってもよい）複数の仮想計算機が作り出される．各仮想計算機上では異なる OS を走らせることができ，それらは各々ジョブを実行する．同一実マシン上でジョブ実行が進められるが，仮想計算機間の通信路は仮想ディスクの共有，仮想チャネル経由などの形でしか存在しないため，隔離性が高いと言える．

米国国防省では，前述の時間帯を分ける方式の無駄を避けるため，VM/370 をベースとした KVM/370 の開発を行った[9]．図-2 に構成を示す．KVM/370では，VM/370 の CP から真に実リソースの仮想化に必要な部分のみを抜き出してカーネル化し，その上で一種の仮想計算機として CP の残りの部分（NKCP：Non-Kernel CP）を機密度のレベル別に実現している．各 NKCP の下では，そのレベルの処理を行う仮想計算

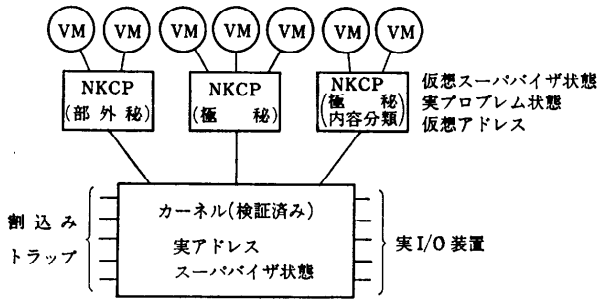


図-2 KVM/370 の構成

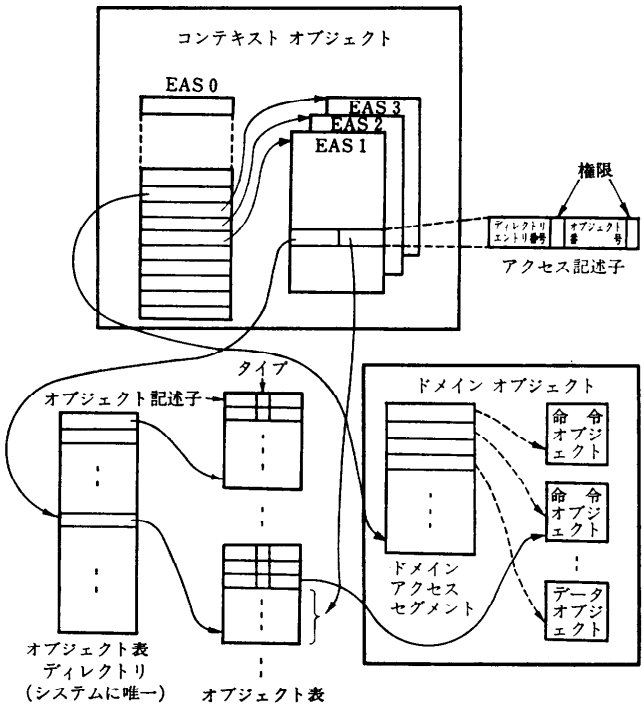


図-3 iAPX 432 のプロテクション機構

機が複数個動きうる．同一 NKCP 下の仮想計算機間の関係は VM/370 の仮想計算機間の関係と同じであるが，異なる NKCP 下の仮想計算機間の通信，資源共用についてはカーネルがチェックを行う．

仮想計算機ベースの高セキュリティシステムは，KVM/370 の他にもいくつかの試みがされている[8]．

4.3 iAPX 432

iAPX 432 は，インテル社が1981年に発表した3チップ構成の32ビットマイクロプロセッサである[10]．ケイパビリティ及びドメインによる保護をアーキテクチャのレベルで大胆に取り入れている．

図-3 に保護に関する部分の構成を示す．iAPX 432 ではすべてのデータ構造はオブジェクトとして統一的に管理される．OSの制御ブロックにはそれぞれ固有のタイプが割り当てられ，特有のアクセス制御を受ける．プログラムはドメインの集合として表現される．ドメインは Ada のパッケージに対応するプロシージャ，データセグメントの集りである．コール命令によってドメインからドメインへ制御が移るが，この時，プロセスから見えるアドレス空間，プロテクションは完全に切り換わる．

ドメインの実行中における動的な管理を行うのがコンテキストオブジェクトであり，ドメインのコール時に生成され，リターン時に壊される．主な構成要素として4個のエントリアクセスセグメント (EAS) があり，実行中のプロセスはここにアクセス記述子 (AD) の登録されているオブジェクトだけに直接アクセス可能である*．AD(ケイパビリティに相当) にはそのコンテキストで許される当該オブジェクトへのアクセス権限(参照，更新，消去)が記入されており，更にオブジェクト記述子にはタイプが記入されている．この結果，アクセスのたびに両面からのチェックを受ける．あるタイプのオブジェクトはある命令でしか参照できないなどがアーキテクチャで定められている．オブジェクトをユーザが定義し，アクセスをソフトウェアでチェックすることも可能になっている．また，ドメインへのアクセスを許可する時に，ドメインの一部分だけに制限することも可能である．

セキュリティの面から見て非常に望ましいアーキテクチャが提供されているが，サポートされる OS に依存するところが大であることと性能面のオーバヘッド等が問題点として指摘されている．

* 新しいドメインに制御が移った後に，ソフトウェアでドメインアクセスセグメントを EAS の一つとして指定することになると思われる．

5. むすび

以上，本稿では，計算機内の蓄積情報に対して OS が行っている各種のセキュリティ制御について，その考え方，実現法，システム例などを解説した．OS におけるセキュリティの研究は，1970年頃，米国で始められ，その後軍用応用等も含めて各方面で続けられてきた．セキュリティの理論研究としては未だ残された問題も見受けられるが，全体的には，実際の場面で実用上効果が大きいと思われる主要技術及びその実現機構がほぼ出揃ったという感が深い．

一般に，実用システムで完璧なセキュリティを目指すのは，実行オーバヘッドや動作効率の面から現実的と言えない．しかし，他方では，現実問題化しつつある各種の悪意のセキュリティ侵害の試みに実用上耐えうる程度のシステムでさえ数少ないのが現状である．今後，これらの点を飛躍的に改善していくには，計算機利用におけるセキュリティ意識が高まる中で，一般計算機のセキュリティの在るべき姿を明確化し，それを実際の計算機機能として着実に実現していくことが必要である．特に，システム全体の立場から総合的にセキュリティを捉えること(OSで対処できない問題も含めて)，アクセス制御を中心とするコスト効果性の高いセキュリティ制御の枠組みを商用システムの OS 機構として確立すること，システムのフォールトトレランス機能にセキュリティ上の考慮を十分に反映させることなどが重要である．

参考文献

1) 和田英一：情報保護の機構，情報処理，Vol. 16, No. 12, pp. 1092-1099(1975).
2) 関野 陽，後藤和夫：データ保護のための技法，情報処理，Vol. 23, No. 4, pp. 335-341(1982).
3) Saltzer, J. H. and Schroeder, M. D.: The Protection of Information in Computer Systems, Proc. IEEE, Vol. 63, No. 9, pp. 1278-1308 (1975).
4) Landwehr, C. E.: Formal Models for Computer Security, Comput. Surv., Vol. 13, No. 3, pp. 247-278(1981).
5) Jones, A. K.: Protection Mechanisms and the Enforcement of Security Policies, Lecture Notes in Computer Science 60, pp. 228-251, Springer-Verlag (1978).
6) Attanasio, C. R., Markstein, P. W. and Philips, R. J.: Penetrating an Operating System, IBM Syst. J., Vol. 15, No. 1, pp. 102-116(1976).
7) Popek, G. J., Kampe, M., Kline, C. S., Stoug-

hton, A., Urban, M. and Walton, E. J.: UCLA Secure Unix, Proc. AFIPS NCC, Vol. 48, pp. 355-364 (1979).
8) Landwehr, C. E.: The Best Available Technologies for Computer Security, Computer, Vol. 16, No. 7, pp. 86-100 (1983).
9) Gold, B. D., Linde, R. R., Peeler, R. J., Schaefer, M., Scheid, J. F. and Ward, P. D.: A Security Retrofit of VM/370, Proc. AFIPS NCC Vol. 48, pp. 335-344 (1979).
10) Intel Corp.: iAPX 432 General Data Processor Architecture Reference Manual.

(昭和59年4月9日受付)