# セキュアな分散システム構築のための一検討

橋本正樹　金美羅　辻秀典　田中英彦

{mgs053504, mira, tsuji, tanaka}@iisec.ac.jp

情報セキュリティ大学院大学 情報セキュリティ研究科 情報セキュリティ専攻

〒 221-0835 神奈川県 横浜市 神奈川区 鶴屋町 2-14-1

分散システムは、単一システムと比較して経済性・速度・冗長性・拡張性・柔軟性といった面で優れているため、その実現に向けて従来より多くの研究が行われてきた。しかしながら、近年の分散環境における資源管理機構はミドルウェアやアプリケーションのような上位層で実装する方式が多く、システム全体に対する一定の信頼性担保を困難にしている。こういった背景のもと、本研究はシステムソフトウェアにおいて細粒度の資源管理を行うことにより、分散システム全体の信頼性向上を目指すものである。そのために、本稿では分散システムへのケイパビリティ方式の適用に関する考察を行う。

# A Study for the Security of Distributed System Architectures

MASAKI HASHIMOTO, MIRA KIM, HIDENORI TSUJI

and HIDEHIKO TANAKA

{mgs053504, mira, tsuji, tanaka}@iisec.ac.jp

Graduate School of Information Security, Institute of Information Security

2-14-1 Tsuruyamachi, Kanagawa-ku, Yokohama, 221-0835

This paper describes the use of operating system as a distributed secure computing infrastructure. In particular it reports the manner of resource management for distributed environment, addressing the fine-grained protection and the principle of least privilege. These manners are compared on the features they offer in the context of secure computing: Reference monitor concept, Secure Channel, Authorization, Naming. Finally, we suggest the prototype of our system and the future plan.

## 1 はじめに

分散システムは、単一システムと比較して経済性・速度・冗長性・拡張性・柔軟性といった面で優れている。そこで、その実現に向けた研究が1960年代から1970年代にかけて盛んに行われたが、最終的にはハードウェアやネットワークの性能に起因する原因、或は1980年代におけるUNIXの台頭により実質的にはプロジェクトを停止した。しかし現代においては、技術の進展によって当時想定していたコンピューティングモデルが現実的なものとなりつつあり、再び研究が再開されている。

近年の分散システムに関する研究は、資源管理をミドルウェアやアプリケーションのような上位層で実装する方式が多いが、これにはシステム全体に対する一定の信頼性担保を困難するという問題がある。また、上位層での実装はシステムソフトウェアが信頼できることを前提としているが、実際にそれを保証する仕組みはまだ一般的ではない。一方で、単一システムではシステムソフトウェアにおいて細粒度の強制的な資源管理を実施することにより信頼性を高める研究も行われているが、分散環境への拡張に関しては現在研究が進められている最中である。

こういった背景のもと、本研究はシステムソフトウェアにおいて分散環境向けの資源管理機構を提供することにより、分散システムの信頼性向上を目指すものである。そのために、本稿では分散システムへのケイパビリティ方式の適用に関する考察を行う。

## 2 研究の目的

### 2.1 背景

分散システムは、複数の独立した計算機が連携して単一の計算機のように機能するシステムの総称である。これはネットワークを介して接続される形態が代表的であるが、広義にはバスを介して接続される複数演算装置を備えたシステムを指す場合もあり、また一次記憶装置の共有形態によりさらに細かく分類される場合もある。

分散システムには、単一システムやメインフレームと比較して以下のような利点がある。

- 演算装置に関して良好な価格／性能比を提供する。（経済性）
- 単一計算機より高速な処理能力を持つことが可能。（速度）
- 1台の計算機が破壊されても全体としては稼働し続ける。（冗長性）
- 分散システムは計算能力を段階的に追加・削減可能。（拡張性）
- 効果的に利用できる計算機に負荷を分散できる。（柔軟性）

そのため、1960年代から1970年代にかけては分散システムの実現に向けた研究が盛んに行われ、一定の成果をあげた。Multics[1]・Hydra[2]・STAROS[3]・Amoeba[4]といったプロジェクトはその代表的なものであるが、ハードウェアやネットワークの性能に起因する原因、或は1980年代におけるUNIXの台頭により実質的にはプロジェクトを停止している。

しかし近年、２つの技術の進展によって高速ネットワークで接続された多数の演算装置からなるコンピューティングモデルが現実的なものとなりつつある。演算装置の価格／性能比は、かつての1秒あたり1命令／10億ドルといったスケールから、近年では1秒あたり10億命令／1000ドルを凌ぐものとなっているし、LANの転送速度は今や1秒あたり1Gbitに達している。

一方で、そのようなコンピューティングモデルは元来のメインフレームやパーソナルコンピュータ向けのものとは根本的に異なったソフトウェアを必要とするが、これに関しての研究は未だ発展途上にある。Jonathan等によるSecurity-Enhanced Linux[5]（以降SELinux）の強制アクセス制御をネットワーク上へ拡張する研究[6]、あるいはKenneth Thompson等によるPlan9[7]といった研究はそのような新たな要求に応えるべく比較的最近始まったプロジェクトである。

分散システムの実現に関わる研究課題は既にある程度の体系化がされており、代表的なテーマは資源参照・同期・一貫性・誤り制御・セキュリティに関わるものである。特にセキュリティに関してはリファレンスモニタ[8]の実現が一つの課題であり、本研究はこれに対して焦点を当てたものである。

## 2.2 関連研究

SELinuxは、今日の代表的なセキュアOSである。これはFlaskセキュリティアーキテクチャ[9]のLinuxカーネルに対する実装であり、単一計算機における資源管理機構として以下の特徴がある。第一は機構と方策の分離を実現していること、第二は細粒度の強制アクセス制御を実現していること、第三は管理対象資源のグループ化を柔軟に行うことができる仕組みを提供していることである。SELinuxはこれらの特徴により、従来のシステムソフトウェアより信頼性の高い資源管理を実現しているが分散環境への適用に関しては現在進行中である。

分散環境向けのシステムソフトウェアに関する研究としてはAmoebaがあるが、これは分散配置された単一システムの集合を一つの統合されたシステムとして扱うことができる汎用分散オペレーティングシステムである。しかし、Amoebaは単一システム向けのセキュアOSと比較すると資源管理における機構と方策の分離・権限管理の粒度・拡張性といった点において問題がある。

分散環境でのミドルウェア・アプリケーション層における資源管理に関する研究としては、The STRONGMAN Architecture[10]があるが、これはグローバルポリシーとローカルポリシーの整合性を図ることにより、ネットワークの拡張性に対して新しいアプローチを行っているものである。The STRONGMAN Architectureは、ネットワーク上のドメイン間で矛盾のない強制アクセス制御が可能であるが、計算機毎のアクセス制御が基本となるためAmoeba同様、資源管理の粒度が粗い。

## 2.3 本研究のねらい

分散環境においてはリファレンスモニタの実装に伴う資源管理を粗粒度に行う方式が現在主流であるが、これにはセキュリティ上の問題がある。

図１にWebDAVの例を示す。WebDAVはhttpを利用してネットワーク上の資源にアクセスする手法であるが、権限管理に関してはアプリケーション内で独自に提供される機構を利用している。この方式ではあらゆるWebDAV経由のアクセスがプロセス$S_B$の権限をもつことになり、仮にこの状況でプロセス$S_B$が乗っ取られるとWebDAV経由でアクセス可能であったあらゆる資源が危険に晒される。また同様に他計算機のデータベースにCGIプログラム等を経由してアクセスする際もCGIプログラムの提供する機構で認証することによりアプリケーション上では権限管理が行われているが、実際のシステム内における権限は一律に経由したCGIプログラムの権限となる。

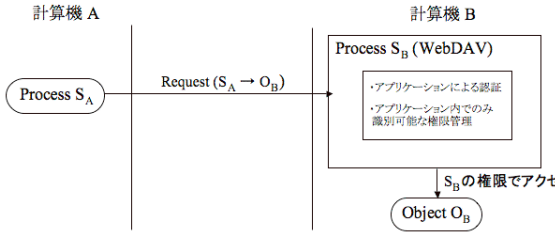


図１：WebDAVアクセスにおける権限管理

このような粗粒度による資源管理の問題点は、最小権限の原則[11]として知られるセキュアなシステムを構成するための指針に関するものであり、粒度が粗いために資源にアクセスする動作主体（以降サブジェクト）がその操作に必要のない権限まで保持することに由来する。最小権限の原則は細粒度の資源管理機構を必要とするが、これはSELinuxに代表されるセキュアOSにおいて実現されている一方で、分散システムへの適用に関してはまだ実現されていない。

そこで本研究では、細粒度の資源管理が可能な分散システムの実現を目指し、システムソフトウェアにおける資源管理機構の実装を目指すものである。そのために、本稿では分散環境におけるケイパビリティ方式を利用した資源管理に関する考察を行う。

## 3 指針と設計

### 3.1 分散環境向け細粒度資源管理

分散環境における資源管理はミドルウェアやアプリケーション層で行う方式が主流である。この方式には手続きと権限管理の簡略化やシステムソフトウェア・ハードウェアの差異を吸収可能等の利点があるが、一方で最小権限の原則からは乖離したものである。またミドルウェア・アプリケーション層、システムソフトウェア層、ハードウェア層の全てを信頼する必要があるが、実際にはシステムソフトウェア層以下の信頼性をミドルウェア・アプリケーション層が保証することはできない。

そこで本研究では、分散環境向け資源管理機構のOS層における実現を目指す。これは、単一システム同様な細粒度の資源管理を分散環境においても可能とするものであり、これにより最小権限の原則を分散環境に適用可能となる。また資源管理に関する仕組みをOS層にまとめることにより、トラステッドコンピューティングベース[8]の縮小も可能となる。

また、OS層における資源管理機構に関して本研究においてはケイパビリティ[12]方式を基本方式とする。ACL[13]方式はUNIX系OSで採用されたことにより近年では一般的な方式であり、保護情報の完全性を容易に保つことができるという利点があるが、アクセス対象となる資源（以後オブジェクト）に対する非特権的な参照を一度認めた後にOSがサブジェクトと保護情報を審査する方式なので、特に分散環境には不向きである。一方でケイパビリティ方式は保護情報に加えてオブジェクトへの参照情報も同時に管理できることが最大の利点であり、ケイパビリティの残存期間の扱い方が困難である等の課題はあるが分散環境向きである。ACL方式とケイパビリティ方式はそれぞれに利点・欠点が存在するが、本研究においてはケイパビリティ方式を分散環境における資源管理に適用するものである。

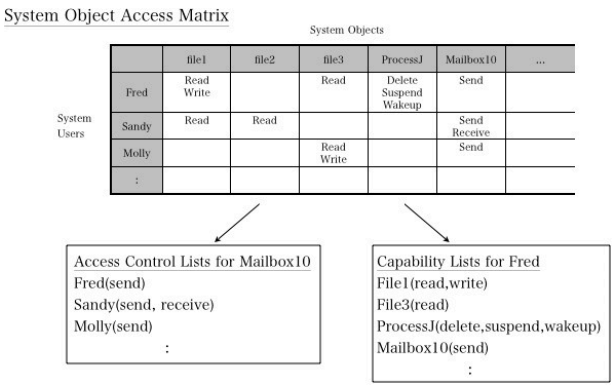

図２：ケイパビリティ方式とACL方式

### 3.2 期待できる効果と要求事項

以上の理論的背景により、本研究においてはセキュアな分散システム構築に関して以下の効果が期待できる。

- 細粒度のアクセス制御が可能となり、セキュリティ上のリスクを軽減することができる。
- OS層で資源管理を行うことによりミドルウェア・アプリケーション層の実装によらない一定の信頼性をシステムに保証することができる。
- OS層で資源管理を行うことにより計算機同士を密に結合することが可能であり、結果として計算機同士の連携による新しいセキュリティモデルの構築が可能となる。
- ケイパビリティ方式の適用は分散環境における参照情報と保護情報の管理問題を容易にする。

また、そのために本研究においては以下の要求事項を考慮する必要がある。

- 細粒度の資源管理を行うため、ミドルウェア・アプリケーション層で行う粗粒度のものに比べて複雑なメカニズムを必要がなる。
- 確実な情報伝播のためにセキュアチャネル確保が必要となる。
- サブジェクトとオブジェクトの詐称を防止する認証機構が必要となる。
- ケイパビリティそのものの完全性を保証する保護メカニズムが必要となる。

### 3.3 プロトタイプ設計

本研究で想定する分散環境のプロトタイプを図３に示す。計算機Aと計算機Bは、ネゴシエータ（以降NB）経由で安全な通信が可能なものとし、計算機Aのサブジェクト（以降SA）は計算機Bのオブジェクト（以降OB）にこの経路を通じてアクセスするものする。本プロトタイプにおいて、ネゴシエータはリファレンスモニタとしての役割を担うものであり、他計算機のサブジェクトの認証・アクセス制御の強制・確実なデータ転送・機密性保持を保証する。このような特性上、ネゴシエータは各計算機の資源管理に関わる外部とのあらゆる通信を制御するものとして想定する。

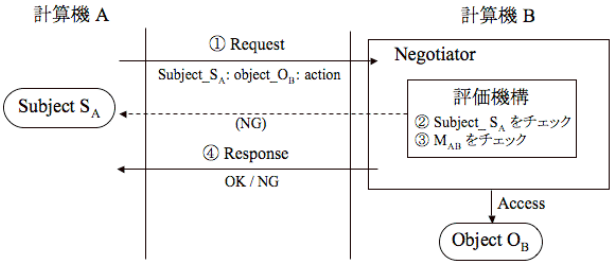


図３：分散環境のプロトタイプ

ある計算機A内のサブジェクトが別の計算機B内のオブジェクトを操作する際は以下の手順を経るものとする。

① SAは計算機B内のネゴシエータにOBへのアクセスを要求する。
② NBはSAを認証し、SAとの間に安全な通信路を確保する。以後、SA・NB間通信は全てこの通信路経由で行われるものとする。
③ NBはサブジェクトマッピングリスト（以降MAB）を参照し、アクセス可否決定を行う。具体的には、SAの要求するアクセスが可能で、且つMABにおいて、SAの代理アクセスを担当するサブジェクト（以降SB）を計算機B内で探す。該当サブジェクトがなければアクセス不可となる。
④ SBがOBに対する代理アクセスを行い、結果をNB経由でSAへ返す。

## 4 おわりに

本稿では、セキュアな分散システム構築によるシステム全体としてのセキュリティ向上を目指し、特に分散環境における細粒度の資源管理機構に関する考察を行った。それに関連して、OS層で資源管理機構を提供する利点と欠点を示し、またケイパビリティシステムの手法を取り入れる利点と欠点を示した。また、OS層におけるケイパビリティ方式を採用した分散環境向け資源管理機構のプロトタイプを示した。

今後は特に本稿で示したプロトタイプに関して以下を課題とする。

- 各手順に関するプロトコルの詳細化
- ネゴシエータ動作メカニズムの検討
- 未知の資源への対応
- 上位概念としてのセキュリティポリシーから自動的にケイパビリティを決定するメカニズムの検討
- 動的なケイパビリティ変更を可能にするメカニズムの検討
- ケイパビリティの保護に関する検討

## 参考文献


[1] M. D. Schroeder, D. D. Clark, J. H. Saltzer, D. H. Wells. “Final Report of the Multics Kernel Design Project,” 1977.

[2] W. Wulf, E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson, F. Pollack, “HYDRA: The kernel of a multiprocessor operating system,” Comm. ACM, 1974.

[3] K. Jones, Robert J. Chansler, Jr., Ivor Durham, Karsten Schwans, and Seven Vegdahl, “StarOS, A Multiprocessor Operating System for the Support of Task Forces,” In Proceedings of the 7th Symposium on Operating Systems Principles, December 1979.

[4] Andrew S. Tanenbaum, “The Amoeba distributed operating system,” Technical report, Vrije Universiteit, 1990.

[5] “Security-Enhanced Linux,” NSA, http://www.nsa.gov/selinux/.

[6] Jonathan M. McCune, Stefan Berger, Ramon Caceres, Trent Jaeger, Reiner Sailer, “Bridging Mandatory Access Control Across Machines,” November 4, 2005.

[7] R. Pike, D. Presotto, K. Thompson, and H. Trickey, “Plan 9 from Bell Labs,” UKUUG Proceedings of the Summer 1990 Conference, London, England, 1990.

[8] U.S.Dept. of Defence. “Department of Defence trusted computer system evaluation criteria,” Dept. of Defence, December 1985.

[9] Ray Spencer, Stephen Smalley, Peter Loscocco, Mike Hibler, David Andersen, Jay Lepreau, “The Flask Security Architecture: System Support for Diverse Security Policies,” USENIX, August 1999.

[10] A Keromytis, S Ioannidis, M Greenwald, and J Smith, “The strongman architecture,” In Third DARPA Information Survivability Conference and Exposition (DISCEX III), April 2003.

[11] P. J. Denning, “Fault tolerant operating systems,” Computing Surveys (USA), December 1976.

[12] J. B. Dennis and E. C. Van Horn, “Programming Semantics for

Multiprogrammed Computations," Comm. ACM, 1966.

[13] Jerome H. Saltzer, "Protection and the control of information sharing in Multics," Comm. ACM, 1974.