社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ＣＳＥＣ－38（56）
2007／7／20

アプリケーション・プラットフォームとしてのセキュアOSに関する初期的検討

辻 秀典[†‡] 橋本 正樹[†] 金 美羅[†] 田中 英彦[‡]

†情報セキュリティ大学院大学 〒221-0835 神奈川県横浜市神奈川区鶴屋町2-14-1
‡株式会社情報技研 〒103-0024 東京都中央区日本橋小舟町3-1
E-mail: †{tsuji, dgs074105, mira, tanaka}@iisec.ac.jp, ‡hide@iit.jp

あらまし 現在、情報システムにおけるセキュリティ確保は必須となっている。しかしながら、現在の情報システムの標準的な形態であるネットワーク分散システムにおいて、セキュリティの確保は個々のアプリケーションの実装方式もしくは一部のミドルウェア機能に依存しており、共通のセキュアプラットフォームが確立されているとは言いがたい。本来であれば、セキュリティ確保は共通化されたプラットフォームで行うべきであり、アプリケーションレベルではなく、OS レベルでの確保が必要不可欠と考える。そこで、アプリケーションのセキュリティ確保のために必要となる要件を整理し、アプリケーションの視点からプラットフォームとしてのセキュア OS に必要となる機能の初期的検討を行う。

キーワード セキュアOS、セキュリティ確保、セキュアプラットフォーム、分散システム、アプリケーション

Preliminary Studies of Secure OS as Application Platform

Hidenori TSUJI[†‡] Masaki HASHIMOTO[†] Mira KIM[†] and Hidehiko TANAKA[†]

†Institute of Information Security 2-14-1 Tsuruya-Cho, Kanagawa-ku, Yokohama, 221-0835, Japan
‡Institute of Information Technology, Inc. 3-1 Nihonbashi-Kobunacho, Chuo-ku, Tokyo, 103-0024, Japan
E-mail: †{tsuji, dgs074105, mira, tanaka}@iisec.ac.jp, ‡hide@iit.jp

Abstract Today, ensuring security for information systems is very important. However, ensuring security depends on the individual application or some of middleware functions. Therefore, it is hard to say common secure platform has been established for network distributed system, today's standard information system. Essentially, ensuring security should be on the standardized platform; hence, it is necessary to ensure security not on the application level but on the OS level. Consequently, we organize the requirements of ensuring security in the application and discuss the preliminarily study of secure OS as a platform from the application standpoint.

Keyword secure OS, ensuring security, secure platform, distributed system, application

1. 背景

現在、情報システムにおけるセキュリティ確保は必須である。特に、近年の情報システムは、ネットワークを介して複数の処理要素が連携処理を行う分散システムが基本である。しかしながら、セキュリティの確保は個々のアプリケーションの実装方式、もしくは一部のミドルウェア機能に依存しており、共通のセキュアプラットフォームが確立されているとは言いがたい。

そのため、アプリケーション開発者に対してセキュリティ確保のための負担を強いるとともに、情報システムのセキュリティレベルが個々のシステムによって異なる要因となる。さらに、アプリケーションやミドルウェアで確保できるセキュリティには限界がある。

本来であれば、セキュリティ確保は共通化されたプラットフォームで行うべきであり、個々のアプリケーションレベルではなく、OS レベルで行うことで、情報システム一般のセキュリティレベルを標準化することが、必要不可欠と考える。

セキュアプラットフォームの代表としてはセキュア OS があげられるが、現在のセキュア OS は、分散システムに対するセキュリティ確保の必要十分な機能を

有しているとはいえない。そこで本稿では、現在の標準的な情報システムに対してセキュアプラットフォームとなりうるセキュアOSの要件について議論する。

まず、2節において、セキュアOSの現状について述べる。3節において、現在の情報システムにおけるセキュリティ確保について整理した上で、4節において現状のセキュアOSの限界について述べる。5節では、現在の情報システムに適合したセキュアなアプリケーション・プラットフォームの要件について検討を行い、6章で全体をまとめる。

2. セキュアOSの現状

近年の情報システムに対するセキュリティ意識は非常に高くなっているが、情報システムのセキュリティに関する議論そのものは、古くより行われてきた。

例えば1983年に定められ、1985年に改定された米国のTCSEC[3]（通称、オレンジブック）は、さかのぼること1967年に米国のDSB (Defense Science Board/国防科学委員会）によって議論が開始されている。TCSECは後に欧州で策定されたITSEC（Information Technology Security Evaluation Criteria）などとともに、CC（Common Criteria）プロジェクトによって1999年に策定されたISO15408の基礎となっている。

セキュリティ要件の標準化において、プラットフォームに位置づけられるOSのセキュリティに関しても議論されている。また、1976年の「最小特権のセキュリティ原則（PLP: Principle of Least Privilege）」[4]に関する考察に代表されるように、OSそのもののセキュリティに関する議論も古くより行われている。

2.1. セキュアOSの一般的定義

従来のOSにおけるセキュリティ管理は、ACL（Access Control List/アクセス制御リスト）[5]に基づいて行われてきた。ACLに基づきOS資源へのアクセスを管理することでセキュリティを確保する。しかし、ACLによるセキュリティ管理は、アクセス制御の粒度が粗いこと、そして制御の確実性が担保されていないことが問題とされてきた。

それに対してセキュアOSでは、より精度の高いセキュリティ管理を行うために、MAC（Mandatory Access Control/強制アクセス制御）[3]機能を実現している。そして、これらを厳密に適用するために、リファレンスモニタによりOSの資源管理が監視される。リファレンスモニタは、資源へのアクセスが迂回不可能、資源が改ざん不可能、資源が安全であることの証明を行わなければならない。

一般的には、これらの要件を基本的に満たすOSがセキュアOSと呼ばれている。

2.2. セキュアOSの実装例

現在、代表的なセキュアOSとしては、Security-Enhanced Linux[6]（以降SELinux）があげられる。SELinuxは米国のNSA（National Security Agency/国家安全保障局）が主導となり開発される、Linuxカーネルのセキュリティ拡張モジュールである。

SELinuxにおいては、リファレンスモニタを実現するために、Flaskセキュリティアーキテクチャ[7]が採用されている。この、Flaskセキュリティアーキテクチャに対して、Type Enforcement(TE)およびRole-Based Access Control（RBAC）というセキュリティポリシーを適用することで、MACが実現されるとともに、Multi-Level Security[8]についても実現されている。

この他、数多くのLinuxベースのセキュアOSだけでなく、FreeBSDをベースとしてセキュリティ機能を拡張したTrusted BSD[9]など、多くのセキュアOSが存在している。

これらに共通していることは、さまざまなアーキテクチャにより、先に述べたMACを実現していることが基本となっていることである。

3. 情報システムのセキュリティ確保

情報システムのセキュリティを確保するためには、情報システムを構成するすべての要素のセキュリティを確保しなければならない。ネットワークが標準的となった現在の情報システムにおいては、アプリケーションもしくは、それをサポートするミドルウェアやフレームワークによって、セキュリティ確保が行われることが一般的となっている。

3.1. 情報システムのセキュリティ構成

現在の情報システムは、単一の計算機のみによって構成されることは少ない。計算機をひとつの構成ノードとしたとき、ネットワークによって接続された複数のノードによって構成される分散システムが一般的である。ネットワーク機器などの専用機器も、特定目的のハードウェアとソフトウェアの組み合わせであり、計算機であることには変わりはない。

このような分散システムにおいて、セキュリティ構成は図1のようにとらえることができる。

単一の構成ノードに注目すると、複数の階層構造を持っており、最下層にはハードウェア、その上には複数の階層からなるソフトウェアが存在する。最もシンプルな構成であっても、ソフトウェアはプラットフォームであるOSと、アプリケーションという階層から構成される。ノードによっては、アプリケーションとOSの間にミドルウェア等の中間レイヤが配置される

こともある。単一ノード内のセキュリティを考える時、

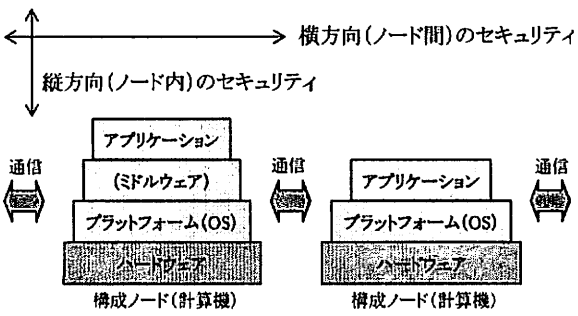

図 1: 分散システムのセキュリティ構成

ノード内の各階層でのセキュリティを考慮する必要がある。これを縦方向のセキュリティと表現する。

ネットワークによって結合された分散システム全体においては、それぞれの構成ノード内だけでなく、全構成ノード間のセキュリティについても考慮する必要がある。これを横方向のセキュリティと表現する。

分散システム全体のセキュリティを考慮する際、このような縦方向と横方向のセキュリティレベルを統一化する必要がある。

3.2. 現在の OS のセキュリティ的な位置づけ

OS は計算機において、アプリケーションのために資源管理を行うプラットフォームである。下位層にあるハードウェアを管理し、上位層のアプリケーションに対して、資源を利用するための機能であるシステムコールを提供する。ネットワークが重要な基盤となった現在においては、ローカルの資源管理だけでなく、通信の管理も重要な機能である。

現在の情報システムにおいて、OS に対して要求されていることは、アプリケーション・プラットフォームとしての安定性である。TCP/IP ネットワークの普及により、むしろプラットフォームの差異をアプリケーションが通信を介することによって吸収している側面もあり、プラットフォームである OS が分散システムの構成をサポートしているとはいえない。

そのため、分散システム全体のセキュリティ確保は、プラットフォームより上位層のソフトウェア階層で行われることが一般的となっている。

3.3. アプリケーション依存のセキュリティ確保

分散システムの典型的な例といえば、ユーザに対してネットワークを介してサービス提供するサーバ、クライアント型のシステムである。このとき、クライアントも含めた一連のシステムが分散システム全体と位置づけることができる。例えば、サーバ側も複数のノードが連携し、DB により情報を管理する EC（電子商取引）サイトは、システム構成的には典型的な分散システムの形態といえる。

ここで、EC サイトの情報に対するアクセス権について考察する。図 2 では EC サイトのシステムをモデル化して示した。ノード p、q、r がサービスを提供の中心となるサーバ群であり、サイト X、Y、Z はこれらのサーバが連携する外部サイトのシステム、A、B がサイトを利用するユーザである。ここでは、ノード p が Web サーバ、ノード q がアプリケーションサーバ、ノード r を DB サーバとする。

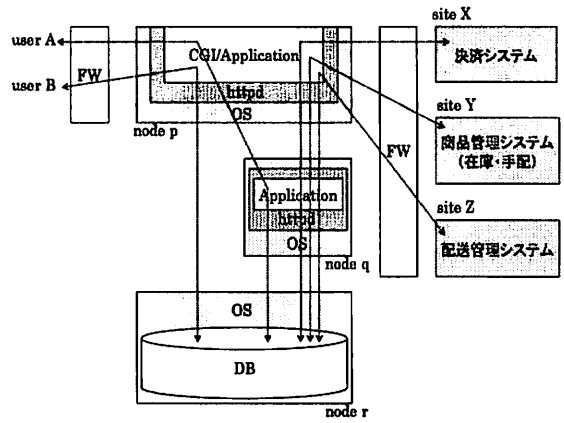

図 2: EC サイトシステムのモデル化

このような分散システムにおいては、なんらかのアクションには通信を伴う。通信は OS のスタックで処理され、Web サーバプロセスである httpd に処理を渡す。httpd は通信内容の処理を行うために、CGI 経由でアプリケーションプログラムを起動する。プログラムは、処理内容に応じて、DB サーバもしくはアプリケーションサーバに処理を渡す。その際、再び httpd および OS を経由して別のノードに処理が受け渡されることになる。受け渡されたノードにおいても、再び OS と httpd を経由してプログラムに処理が渡される。

ここで、図 3 に具体的な処理の受け渡しフローの例を図示した。

ユーザ A もしくは B からのアクセスは、OS の通信スタックを経由して、ノード p の httpd でリクエストを受け付け、CGI を経由して httpd プロセスの権限でアプリケーションプログラムを起動する。このとき、OS の通信スタックも、httpd そして CGI プログラムのプロセスも、受け付けたリクエストが A もしくは B のどちらのユーザのものであるか区別はできない。ユーザを区別する情報は、TCP/IP のペイロードの中に、ア

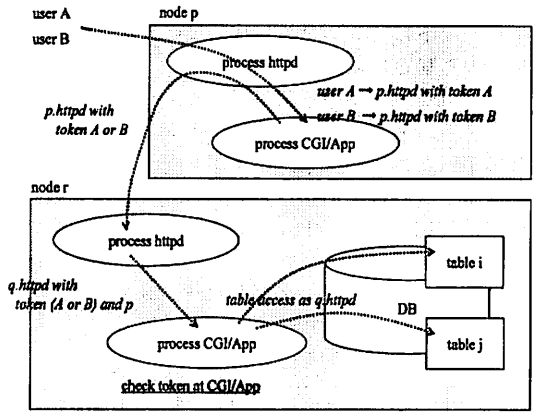

図 3: アクセスフローと権限遷移

プリケーション階層の情報（トークン等）として埋め込まれているからである。そのため、アプリケーションより下の階層において、ユーザに応じた処理の権限制御をおこなうことはできず、受け取ったリクエストが正規のものかどうかの判断は、すべてアプリケーションに委ねるしかない。このような、分散システムにおけるノード間の権限管理は、あくまでアプリケーション階層で付加される情報をもとに識別される。

ノードpのCGIプログラムがDBへのアクセスを必要とするとき、httpdとOSを経由してDBサーバであるノードrに処理を受け渡す。このとき、受け側のノードrは処理のリクエストをノードpでの処理と同様に、httpdにてリクエストを受け付け、CGIを経由してhttpdプロセスの権限でアプリケーションプログラムを起動し、最終的にDBにアクセスする。このとき、リクエストが正規のサーバかつユーザのものであるかどうか、ノードrのOSもhttpdプロセスも知ることはできない。あくまで、アプリケーション階層で付加された情報に基づいて、CGIプログラムがチェックを行い、その内容に応じてDBにアクセスして処理を行う。つまり、アクセス制御はアプリケーション階層まで処理されて初めて行われる。

ここで問題となるのが、アプリケーション階層で処理が行われるということは、OS階層で処理の権限管理はできない。httpdもCGIプログラムもDBアクセスも特殊な実装を行わない限り、常に同じOS階層の権限で処理されることになる。

ここでは、ユーザAまたはBの処理例のみで述べたが、さまざまな情報を扱うECサイトの処理において、プラットフォームレベルでは権限管理が非常に低い粒度で行われていることがうかがえる。逆にいえば、権限管理の信頼性はアプリケーションの実装に完全に依存している。情報システムにおける要が情報であるととらえるならば、管理中枢たるDBに対してのアクセス制御が、そのような形で処理されている現状は、セキュリティ的観点で問題である。

3.4. フレームワークによるセキュリティ確保

アプリケーションに依存するセキュリティ確保を改善し、アプリケーション開発負担の軽減と、システム全体のセキュリティレベル統一のために、フレームワークもしくはミドルウェアといった、アプリケーションより低い中間階層でセキュリティを確保する方法も提案されている。その例として、WS-Security（Web Service-Security）[10]について述べる。

WS-Securityは簡単に言えばEnd-to-Endのセキュリティを確保する方法である。

これに対する最も一般的な方法がSSLによるPoint-to-Pointのセキュリティ確保があげられる。これは、先に述べたECサイトの例でもそうであるが、情報を経由するすべてのノード間において、アプリケーション階層でそれぞれのノード間のセキュリティを確保する。また、各ノードはアプリケーション階層までデータを処理しなければ、内容の妥当性もわからず、さらに、すべての情報通過ノードにおいてアプリケーション階層において情報を参照することが可能となる。

それに対してEnd-to-Endのセキュリティにおいては、情報の発信ノードから、到達ノードまでのセキュリティを確保するもので、到達ノードに指定されていなければ、中間ノードで情報を参照することはできない。これを具体的に示したのが図4である。

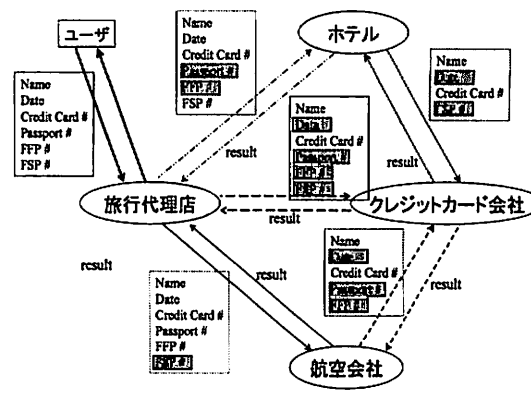

図 4: WS-Securityにおける情報の流れ

ユーザが旅行代理店経由で旅行を手配する際、必要な情報を旅行代理店に提示するが、その情報のうちホ

テル、航空会社、クレジットカード会社それぞれに必要な情報しか開示されない。WS-Securityでは、やりとりされる情報は暗号化されるが、必要な情報に対してのセキュリティトークンを持っていなければ情報を参照することができない。これにより、中間ノードでの情報参照が不可能となる。

実際には、WS-Securityで定義されるセキュリティトークン等のコンテキスト情報は、SOAP メッセージとして定義されている。つまり、あくまでアプリケーションレベルの実装を標準化する規定でしかない。そのため、アプリケーションで参照が許可された情報を持つノードにおいて、下位のソフトウェア階層まで安全性を保証するものではない。

3.5. 現在のプラットフォームの限界

プラットフォームにおける粗粒度アクセス制御の限界は、Confused Deputy Problem[11]として、以前より指摘されている。それでも、Confused Deputy Problemについては、あくまでローカルシステムにおける問題ととらえることができ、細粒度アクセス制御が可能なセキュアOSで解決することは不可能ではない。

しかしながら、大規模にネットワーク化された現在の分散システムにおいては、プラットフォームにおける粗粒度アクセス制御は、システム全体のセキュリティの確保という観点で、より深刻な問題として顕在化しているといえる。

4. 現状のセキュアOSの限界

プラットフォームにおける粗粒度アクセスが、分散システムのセキュリティ確保において問題であるとするならば、MACにより細粒度アクセス制御が可能なセキュアOSを利用することで、問題が解決するという思考は一見妥当と考えられる。

しかしそれは、3つの理由により現在のセキュアOSでは単純に対応できないという限界がある。

4.1. 単一システム志向のセキュアOS

まず、セキュアOSにおける細粒度アクセスの適用可能範囲を考慮する必要がある。MACによる再粒度アクセス制御は、あくまでセキュアOSが稼働しているシステム内にしか及ばない。これまで問題にしてきたことは、OSが稼働しているノードを越えてやり取りされる情報において、細粒度アクセス制御が不可能な点である。

現状のセキュアOSの動向としては、いかにローカルシステムの資源を、現実的かつ効率的に細粒度アクセスを行うかが主流であり、分散システムの大域的な情報に対する細粒度アクセスについては、積極的に検討されているとはいえない。

4.2. 細粒度すぎるセキュアOSのアクセス制御

実際には、SELinuxのラベルの通信を行うことで、ローカルシステムのMACを、ネットワークを介した複数のOSに拡張しようという研究[12]も存在する。これは先に指摘したセキュアOSの単一システム志向を解決する一手法と考えることができる。

しかしながら、ここで問題となるのはSELinuxにおけるMACがあまりにも細粒度すぎる点である。MACを分散システムに拡張できたとしても、実際に利用できるかどうかという問題がある。

例えば、Red Hat Enterpisei Linux 4をデフォルトインストールした場合、ハードウェア構成により異なるものの、カーネルオブジェクト数だけでも130,000を超える。SELinuxが細粒度アクセス制御を可能にするにもかかわらず、一般の情報システムにおいて積極的に利用されない背景には、細粒度すぎるがゆえの、実用上の難易度である。これを、単純に分散システムに拡張した場合、複雑性を増すだけで現実的ではない。

4.3. ローカル資源のアクセス制御志向

セキュアOSのMACが分散システムに適用でき、その複雑性を緩和できたとして、最後の問題は分散システムとセキュアOSのアクセス制御の親和性の問題があると考える。少なくとも、一般的な分散システムのアプリケーションで処理が必要とされるアクセス制御を、現在のセキュアOSのアクセス制御に移行することは難しい。

先に指摘した粒度の問題に限らず、セキュアOSのアクセス制御の志向が、ローカル資源管理に主眼が置かれているため、分散システムにおける処理フローの管理には現状適しているとはいえない。

5. アプリケーション・プラットフォームとしてセキュアOSに要求されること

アプリケーション・プラットフォームとしてのセキュアOSの要求を考えるとき、分散システムに対する適合性を考える必要がある。

現在のプラットフォームでは粒度の低すぎるアクセス制御であるが、セキュアOSのアクセス制御志向では適合性が低い。であるならば、分散システムの特性を考慮する必要がある。

分散システムにおける情報処理は、複数のノード間で連携した情報のやりとりがある点である。その情報に対して、分散処理全体として共有したアクセス制御が行えることが理想であると考える。また、End-to-End

のセキュリティの確保も必要である。これらを、アプリケーションによる実装ではなく、プラットフォームが実現し、アプリケーションに対して提供する必要がある。基本的な考えは、分散システム内でやり取りされる一連のネットワークセッションを、各ノードのOSレベルで共通にハンドリングできるとともに、透過的なアクセス制御を実現することが必要となる。そのために必要な要件を列挙すると、次のようになる。

- ローカルノードおいても、ネットワーク経由の別ノードにおいても、OS がアクセス主体（Subject）を透過的に扱える必要がある。透過的扱いとは、分散システム内の Subject の共有化もしくは、各ノードにおける Subject 同士の権限の継承をさす。
- Subject は分散システム内において、許可された Object にしかアクセスできないこと。
- Subject がアクセスを許可される Object は分散システム内で共有化されていること。
- OS が識別できる Subject および Object に対するアクセス制御情報を、アプリケーションに対しても提供できること。

これにより、アプリケーション階層で、ネットワークセッションのハンドリング、およびアクセス制御を行う必要がなくなる。アプリケーションは、プラットフォームが提供する情報を利用するだけでよい。

それとともに、各ノードにおける情報管理のソフトウェア階層が下がり情報管理がよりセキュアになる。例えば、DB のアクセス権限をプロセスレベルから、上記で定義した分散システム内の Subject レベルにまで粒度を高めることができる。

さらに、アプリケーションはプラットフォームが確保する End-to-End のセキュリティの結果を利用することとなり、アプリケーションレベルで情報を不正利用できなるだけでなく、意図しない情報漏洩を防ぐこともできる。

6. むすび

本稿は現在のセキュア OS そのものを否定するものではない。あくまで、現在の情報システムのセキュリティ確保という観点で、適合したプラットフォームが必要であるということを検討したものである。

現在の情報システムの標準といえる分散システムのセキュリティを確保するには、現在のプラットフォームではアクセス制御の粒度が粗く、セキュア OS ではアクセス制御の粒度が細かすぎる上に、志向が異なっている。そのため、分散システムの性質を議論するとともに、分散システムに適したプラットフォームとして必要な基本要件を整理した。

本稿の議論は非常に初期的な議論にとどまっており、必要となる要件の大きな方向性を示したにすぎない。そのため、検討した要件をより具体的に掘り下げて整理することが今後の課題である。

最終的には、分散システムに適したプラットフォームの要件を詳細化することで標準を示すことで、具体的なプラットフォームの提案の指針としたいと考えている。

文　献

[1] 橋本 正樹, 藤澤 一樹, 宮本 久仁男, 金 美羅, 辻 秀典, 田中 英彦, “分散システムにおける Capability を用いた資源アクセス制御”, 第 38 回コンピュータセキュリティ研究会(CSEC), July, 2007.

[2] 橋本 正樹, 金 美羅, 辻 秀典, 田中 英彦, "セキュアな分散システム構築のための一検討", Computer Security Symposium 2006 (CSS2006), October, 2006.

[3] “Trusted Computer System Evaluation Criteria (TCSEC).”, DOD 5200.28-STD, National Computer Security Center, December 1985.

[4] Peter J. Denning, “Fault Tolerant Operating Systems.”, ACM Computing Surveys (CSUR). 8(4), pp.359-389, 1976.

[5] Jerome H. Saltzer, “Protection and the control of information sharing in Multics.”, Comm. ACM, vol.17, pp.388-402, July 1974.

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

[7] Ray Spencer, Peter Loscocco, Stephen Smalley, Mike Hibler, David Anderson, and Jay Lepreau, "The Flask Security Architecture: System Support for Diverse Security Policies", Proc. 8th USENIX Security Symposium, August 1999.

[8] D.E. Bell and L.J. LaPadula, “Secure Computer Systems: Mathematical Foundations and Model”, M74-244, The MITRE Corporation, May, 1973.

[9] TrustedBSD Project, http://www.TrustedBSD.org/

[10] “Web Service Security: SOAP Message Security 1.1 （WS-Security 2004）.”, http://www.oasis-open.org/committees/download.php/16790/

[11] N.Hardy, "The Confused Deputy: (or why capabilities might have been invented)", ACM SIGOPS Operating Systems Review 22(4), October 1988, pp.36-38.

[12] Trent Jaeger, Kevin Butler, David H. King, Serge Hallyn, JoyLatten, Xiaolan Zhang, "Leveraging IPsec for Mandatory Access Control across Systems.", Proc. 2nd IEEE International Conference on Security and Privacy for Communication Networks 2006.