第Ⅰ編　概要調査編

目　次

1. 概要

1.1. 調査概要

本報告書は、１２のシステムについて以下の観点から調査をまとめたものである。調査には、標準、製品情報およびインターネット上で入手可能な公開情報を基にしている。

- 提供元
- 現在の開発・リリース
- セキュリティ機能
- おおよそのセキュリティレベル

1.2. 調査期間

本報告書の調査期間は、平成１３年１１月から平成１３年１２月である。

1.3. 調査対象

表 1.3-1に挙げた 12 のシステムを対象として以下の調査・分類を行う。

表 1.3-1 概要調査システム一覧

#	調査対象システム 【URL】
1	Security-Enhanced Linux (SELinux) 【http://www.nsa.gov/selinux/】
2	TrustedBSD 【http://www.trustedbsd.org/】
3	OpenBSD 【http://www.openbsd.org/】
4	PitBull 【http://www.argus-systems.com/product】
5	hp virtualvault 【http://www.hp.com/security/products/virtualvault/】

#	調査対象システム 【URL】
6	hp secure OS software for Linux 【http://www.hp.com/security/products/linux/】
7	Openwall 【http://www.openwall.com/】
8	Trusted Solaris 【http://www.sun.com/trustedsolaris/】
9	CBOSS/LOMAC 【http://opensource.nailabs.com/initiatives/cboss/】
10	Linux Intrusion Detection System (LIDS) 【http://www.clublinux.org/lids/】
11	Medusa DS9 【http://medusa.fornax.sk/】
12	Rule Set Based Access Control(RSBAC) 【http://www.rsbac.de/】

1.4. 調査方法

調査は、標準、製品情報およびインターネット上で入手可能な公開情報による文献調査を行った。また、おおよそのセキュリティレベルには TCSEC(オレンジブック)を基にレベル分けを行っている。

オレンジブックでは、４つのセキュリティレベルと、レベル内に１つもしくは複数のクラスが定義されており、それらをレベルの低いほうから並べると次のようになる。

D　　最低限のセキュリティ
C１　任意保護
C２　アクセス制御による保護
B１　ラベル式保護
B２　構造化保護
B３　セキュリティドメイン
A　　検証された設計

表 1.4-1は、オレンジブックの高信頼コンピュータシステム評価標準一覧で

ある。

表 1.4-1 高信頼コンピュータシステム評価標準一覧

		C1	C2	B1	B2	B3	A1
セキュリティポリシー	任意アクセス制御(DISCRETIONARY ACCESS CONTROL)	■	■	■	■	■	■
	オブジェクトの再使用 (OBJECT REUSE)		■	■	■	■	■
	ラベル (LABELS)			■	■	■	■
	ラベルの完全性 (LABEL INTEGRITY)			■	■	■	■
	ラベルつき情報のエクスポート (EXPORTATION OF LABELED INFORMATION)			■	■	■	■
	マルチレベル装置へのエクスポート (EXPORTATION TO MULTILEVEL DEVICES)			■	■	■	■
	単一レベル装置へのエクスポート (EXPORTATION TO SINGLE-LEVEL DEVICES)			■	■	■	■
	人間の判読可能な出力へのラベル (LABELING HUMAN-READABLE OUTPUT)			■	■	■	■
	強制アクセス制御 (MANDATORY ACCESS CONTROL)			■	■	■	■
	サブジェクト感度ラベル (SUBJECT SENSITIVITY LABELS)				■	■	■
	装置ラベル (DEVICE LABELS)				■	■	■
追跡性	識別と認証 (IDENTIFICATION AND AUTHENTICATION)	■	■	■	■	■	■
	監査 (AUDIT)		■	■	■	■	■
	高信頼経路 (TRUSTED PATH)				■	■	■
保証	システムアーキテクチャ (SYSTEM ARCHITECTURE)	■	■	■	■	■	■
	システムの完全性 (SYSTEM INTEGRITY)	■	■	■	■	■	■
	セキュリティテスト (SECURITY TESTING)	■	■	■	■	■	■
	設計仕様と検証 (DESIGN SPECIFICATION AND VERIFICATION)			■	■	■	■
	機密チャネルの分析(COVERT CHANNEL ANALYSIS)				■	■	■
	高信頼運用管理(TURSTED FACILITY MANAGEMENT)				■	■	■
	構成管理 (CONFIGURATION MANAGEMENT)				■	■	■
	高信頼リカバリ (TRUSTED RECOVERY)					■	■
	高信頼配布 (TRUSTED DISTRIBUTION)						■
文書	セキュリティ機能ユーザガイド(SECURITY FEATUERS USER'S GUIDE)	■	■	■	■	■	■
	高信頼運用マニュアル(TRUSTED FACILITY MANUAL)	■	■	■	■	■	■
	テストドキュメント (TEST DOCUMENTATION)	■	■	■	■	■	■
	設計ドキュメント (DESIGN DOCUMENTATION)	■	■	■	■	■	■

□ 要件なし
■ 新規または拡張要件
■ 追加要件なし

また、各国で運用されてきたITセキュリティ評価・認証制度におけるレベルと、CCに基づくレベルの大まかな対応を表 1.4-2に示す。

表 1.4-2 評価保証クラスの対応表

ISO15408 Common Criteria	US TCSEC	European ITSEC
-	D	E0
EAL1	-	-
EAL2	C1	E1
EAL3	C2	E2
EAL4	B1	E3
EAL5	B2	E4
EAL6	B3	E5
EAL7	A1	E6

2. 調査内容

2.1. Security-Enhanced Linux (SELinux)

2.1.1. 提供元

SELinux は米国家安全保障局（NSA[1]）によって開発されている。

2.1.2. 状況

SELinux の核を成すセキュリティアーキテクチャは Flask（Flux Advanced Security Kernel）と呼ばれ、NSA と Secure Computing Corporation[2]との共同研究で開発されたものをベースに、さらにユタ大学の The Flux Research Group[3]を含めた 3 者で機能拡張を施したものである。現在は、アンチウィルスベンダである Network Associates Technology, inc.[4]の 1 事業部門である PGP Security[5]が NSA やそのパートナー企業と協力して、SELinux の開発を進めている。PGP Security では、その研究部門である NAI Labs が NSA と 2 年の契約を結び[6]、セキュリティ強化を目的とした研究開発を担当しており、NAI Labs が開発した Linux カーネル用のセキュリティ管理技術とカーネル管理技術用のセキュリティポリシー設定技術を SELinux に統合することで SELinux の機能拡張を行う予定である。

2.1.2.1. リリース状況

SELinux は当初、カーネルに対する独自のパッチをリリースする形式で開発さ

---

[1] National Security Agency の略
[2] http://www.securecomputing.com/
[3] http://www.cs.utah.edu/flux/
[4] http://www.nai.com/
[5] http://www.pgp.com/
[6] 本契約は 2001 年 4 月 9 日に PGP Security によって発表された

れていたが、その後 Linux Security Module(LSM)[7]をベースとしたバージョンが開発されており、この LSM ベースである SELinux の 1st パブリックリリースが行われた時点で、従来のバージョンの開発は打ち切られている。現在までのリリース状況は以下のとおりである。

- 2000 年 12 月 22 日 1st パブリックリリース
  これは、カーネルバージョン 2.2.12 および Red Hat バージョン 6.1 に含まれるユーティリティをベースとしたバージョンである。
- 2001 年 3 月 16 日 アップデート版のリリース
  SELinux がアーカイブ化され、カーネルバージョン 2.2.18 および 2.4.2 対応のカーネルパッチがリリースされる。
- 2001 年 4 月 12 日 アップデート版のリリース
  カーネルバージョン 2.2.19 および 2.4.3 対応のカーネルパッチがリリースされる。
- 2001 年 8 月 23 日 LSM ベース SELinux プロトタイプの 1st リリース
  LSM カーネルパッチを利用した新たな SELinux のプロトタイプが公開された。このプロトタイプでは、カーネルバージョン 2.4.9 対応の LSM（2001 年 8 月 16 日版）をベースにしたカーネルパッチをベースとしている。
- 2001 年 9 月 26 日 LSM ベース SELinux の 2nd パブリックリリース
  カーネルバージョン 2.4.10 対応の LSM（2001 年 9 月 23 日版）をベースとしている。

[7] Linux システムにおいて、合理的にセキュリティ拡張パッケージをサポートするためのフレームワークとなるローダブルカーネルモジュール。http://lsm.immunix.org/

- 2001 年 10 月 16 日 LSM ベース SELinux の 3rd パブリックリリース
  カーネルバージョン 2.4.12 対応の LSM（2001 年 10 月 11 日版）をベースとしている。
- 2001 年 11 月 19 日 LSM ベース SELinux の 4th パブリックリリース
  カーネルバージョン 2.4.14 対応の LSM（2001 年 11 月 5 日版）をベースとしている。
- 2001 年 12 月 10 日 アップデート版のリリース
  4th パブリックリリースに対するアップデート版のリリース。
  カーネルバージョン 2.4.16 対応の LSM（2001 年 12 月 10 日版）をベースとしている。
- 2002 年 1 月 16 日 アップデート版のリリース
  4th パブリックリリースに対するアップデート版のリリース。
  このアップデートでは 2 つのアップデート版がリリースされており、1 つはカーネルバージョン 2.4.17 対応の LSM（2001 年 12 月 23 日版）をベースとしたもの、もう 1 つはカーネルバージョン 2.5.2 対応の LSM（2002 年 1 月 15 日版）をベースとしたものである。

2.1.3. セキュリティ機能

2.1.3.1. フレキシブルな MAC

SELinux では、プロセスやプロセスがアクセスするファイル等に付与されたセキュリティラベルに基づいてアクセス制御が行われる。管理者権限を持つユーザであっても、このセキュリティラベルに基づいたアクセス制御を逃れることはできない。

(1) セキュリティコンテキスト

SELinux において、プロセスはサブジェクト、プロセスがアクセスする対象はオブジェクトと呼ばれる。SELinux では、このサブジェクトおよびオブジェクトに対してセキュリティラベルが付与されており、サブジェクトからオブジェクトへの、あるいは、サブジェクト間のアクセスは、いかなるアクセスであってもこのセキュリティラベルに基づいたセキュリティポリシーによって許可されなければならない。このセキュリティラベルはセキュリティコンテキストと呼ばれる。

セキュリティコンテキストは、実際には以下にあげる複数の属性から構成されるものである。

- user 属性
- role 属性
- type 属性（あるいは domain[8]属性）
- MLS のレベルあるいは範囲[9]

サブジェクトあるいはオブジェクトに対してセキュリティコンテキストを付与することは、上記の属性を付与することに他ならない。

セキュリティコンテキストにおける user 属性は、Linux システムにおけるユーザの通常の識別属性には依存しないものであり、システム既存のユーザ識別属性と切り分けられたこの user 属性を利用することによって、

[8] サブジェクトに付与されるセキュリティコンテキストの場合、type 属性は domain 属性と呼ばれている。
[9] MLS の実装は不完全であり、オプション扱いとなっている。

既存のアプリケーションとの互換性に影響を及ぼすことなく、そしてシステム既存のユーザ識別属性における変更にも関係なく、正確な制御を施行できるようになっている。

(2) TE（Type Enforcement）

SELinuxでは、セキュリティコンテキストによってサブジェクトに付与されたdomain属性と、オブジェクトに付与されたtype属性との間にアクセスベクタと呼ばれるパーミッションの集合を定義してアクセス制御が行われる。オブジェクトには、ディレクトリ、ファイル、デバイスファイル等の複数種類が存在し、アクセスベクタはオブジェクトの種類ごとに異なるパーミッションの集合として定義される。あるプロセスが、あるオブジェクトへアクセスするためには、アクセス対象のオブジェクトに定義されたアクセスベクタに含まれるパーミッションの集合の中で、同プロセスが試みているアクセス形態に関連したパーミッションが、このサブジェクトとオブジェクトとの間の関係において許可されている必要がある。

SELinuxではサブジェクトおよびオブジェクトに付与されたラベルに基づいてアクセス制御が行われるため、たとえ管理者権限を持つユーザであってもこのアクセス制御によって許可されなければ、何もできないことになる。

(3) セキュリティラベルの遷移

あるプロセスがその処理の中でプログラムを起動した際、新たに起動するプロセスに付与されるdomain属性は、デフォルトでは起動元プロセスに付与されているdomain属性である。しかしあらかじめ設定しておくことによって、新たに起動されるプロセスのdomain属性を変化させることが可能

である。例えば、ユーザがシェルのコマンドプロンプトよりプログラムを起動した場合、シェルに付与されている domain 属性が起動されるプログラムにデフォルトでは付与されてしまう。しかしあらかじめ設定により、同プログラム専用の domain 属性を定義した上で、この domain 属性を起動時に付与するように設定しておくことにより、シェルの domain 属性とは異なる domain 属性にてプログラムを起動させることが可能となっている。起動されるプログラムの domain 属性を起動元プログラムの domain 属性から変化(遷移)させることで、起動されたプログラムが持つ権限を、そのプログラムにとって必要最低限なものに制限できる。

2.1.3.2. RBAC (Role-Based Access Contorol)

RBAC はユーザごとに権限を分割する機能である。role 属性は、ユーザをその役割に応じて分類するためのセキュリティラベルであり、ユーザはシステムへのログイン時に何かしらの role 属性を選択しなければならない。この role 属性は、セキュリティコンテキストの付与によりプロセスに対しても付与されている。SELinux では、この role 属性に基づいてプロセスが遷移できる domain 属性を制限することが可能となっている。つまりあるプログラムが、その処理をまともに実行するためには、必要な権限が与えられる domain 属性への遷移を認められた role 属性のユーザによって起動される必要がある。

2.1.3.3. セキュリティポリシーの動的変更

セキュリティポリシーは、基本的にはシステムの起動時にポリシーファイルから読み込まれるものである。しかし SELinux では、このセキュリティポリシー

を稼動中に読み込み、動的に変更する機能を提供している。

2.1.3.4. セキュリティアーキテクチャ

SELinux は以下にあげる大きく 3 つのコンポーネントから構成されている。

- セキュリティサーバ
- オブジェクトマネージャ
- アクセスベクタキャッシュ

(1) セキュリティサーバ

セキュリティサーバはその起動時に、セキュリティポリシーを読み込んでメモリ上に展開し、あらかじめ設定ファイルにて定義されたセキュリティポリシーを保持・管理するコンポーネントである。セキュリティサーバは、アクセス制御のための外部からのパーミッションチェック要求に対して、自身の保持するセキュリティポリシーに基づいたパーミッションチェックを行い、結果をアクセスベクタとして返却する。またセキュリティサーバは、起動時のセキュリティポリシーの読み込みだけではなく、稼動中の動的なセキュリティポリシーの変更もサポートしている。セキュリティポリシーを稼動中に読み込むという行為は、パーミッションチェックのタイミングとの兼ね合いで、不正なチェック結果を外部に出力しかねないという問題がある。しかし SELinux では、この動的なセキュリティポリシーの変更と、パーミッションチェックとの間の時間的な前後関係を正しく管理できる機構を備えており、動的にセキュリティポリシーが変更された場合でも、不正なパーミッションチェックが行われることが決してない

という特徴を持つ。

(2) オブジェクトマネージャ

セキュリティポリシーによるアクセス制御を実際に実施するコンポーネントがオブジェクトマネージャである。オブジェクトマネージャは、そのアクセス制御のために必要なパーミッションのチェックをセキュリティサーバに対して要求する。オブジェクトマネージャとは、各種のカーネルのサブシステム（例えば、プロセス管理、ファイルシステム、ソケットによるプロセス間通信、System V IPC）に該当する。

(3) アクセスベクタキャッシュ

オブジェクトマネージャがセキュリティサーバからアクセスベクタを取得する際には、厳密には直接セキュリティサーバへ打診しているのではなく、アクセスベクタキャッシュと呼ばれるコンポーネントに対して打診をするようになっている。このアクセスベクタキャッシュの目的は、セキュリティサーバによるパーミッションチェックの結果として返信されるアクセスベクタを、オブジェクトマネージャによって引き続き利用される場合に備えて保持しておくことである。アクセスベクタキャッシュが用意されていることでオブジェクトマネージャは、自身がいったん参照したパーミッション情報（アクセスベクタ）をアクセスベクタキャッシュの中に適切なエントリで保存しておくことができ、パーミッションのチェックに掛かるコストを大幅に減らすことができるようになっている。

またアクセスベクタキャッシュには、動的なセキュリティポリシー変更に対応するための機構も備えられている。前述のように、セキュリティサーバは稼動中における動的なセキュリティポリシーの変更をサポートしてい

る。セキュリティサーバによってセキュリティポリシーが動的に読み込まれると、アクセスベクタキャッシュ内で保持しているアクセスベクタは不正なものとなってしまう。そこでアクセスベクタキャッシュには、セキュリティサーバと連携して、キャッシュされているアクセスベクタの整合性を保つための機構が備えられている。

2.1.4. セキュリティレベル

Security-Enhanced Linux のセキュリティレベルは、B1 クラスに相当する。

2.2. TrustedBSD

2.2.1. 提供元

TrustedBSD の提供元は、TrustedBSD Project である。

TrustedBSD Project は、FreeBSD オペレーティングシステムに対するセキュリティの拡張機能を提供するプロジェクトで、2000 年 4 月 9 日にその発表を行った。本プロジェクトは、CC(コモンクライテリア)を目標[10]に定めている。

2.2.2. 状況

(1) 拡張属性（EA, Extended Attributes)

拡張属性とは、カーネルおよびユーザプロセスに任意の名前データを、また、ファイルやディレクトリにタグ付けできるようにした機能で、ACL・Capability・MAC ラベルを含むさまざまな TrustedBSD セキュリティ拡張機能に必要なセキュリティデータを格納する。

UFS/FFS ファイルシステムのための拡張属性は、2000 年 4 月 20 日に FreeBSD 5.0-CURRENT に統合された。

(2) アクセス制御リスト

ACL は、ファイルやディレクトリに対するよりきめ細かな任意アクセス制御を可能にする機能で、現在 UFS/FSS ファイルシステム上の拡張属性を利用している。

2000 年 10 月 1 日にバージョン 0.4 のパッチが公開され、2001 年 3 月 19 日にバージョン 0.6.1 が公開され、その後 2001 年 3 月 26 日に FreeBSD

5.0-CURRENT のソースツリーに統合された。

(3) 強制アクセス制御

2000 年 4 月 8 日に強制アクセス制御をサポートしたバージョン 0.3 の差分パッチが公開された。2001 年 9 月 26 日には FreeBSD 5.0-CURRENT の post-KSE/SMPng バージョンに対する差分パッチ(バージョン 0.5)がリリースされた。このパッチは、プロセスおよびファイルシステムの Biba/MLS(多層セキュリティ機能)/区画 MAC ポリシーを処理するパッチである。

(4) Capability (能力、資格)

2000 年 4 月 3 日にバージョン 0.1 の差分パッチが提供され、2000 年 9 月 19 日時点でバージョン 0.5.4 のリリースノート、差分パッチ、テストツールが提供されている。

2.2.3. セキュリティ機能

TrustedBSD プロジェクトでは、以下の機能が開発中である。

- サードパーティの認証モジュールを統合するための、拡張可能な認証フレームワーク
- 最小特権および侵害時の危険性軽減を実現するためのきめ細かな権限の設定
- 強制アクセス制御(MAC)
- きめ細かで扱いやすい任意アクセス制御を可能にするファイルシステムおよび他のカーネルリソースのためのアクセス制御リスト
- セキュリティを侵害するイベント監視のサポート

---

[10] TrustedBSD の発表時は、TCSEC(レンジブック)の B1 レベルを目標と記載されていた。

TrustedBSD 1.0 では以下に示す拡張機能を FreeBSD オペレーティングシステムに実装する予定である。

2.2.3.1. セキュリティ管理ツール

従来の UNIX システムは、固定のセキュリティポリシーと管理能力が期待された状態で出荷されていた。システム管理者は購入すると直ちにセキュリティの危険性のある機能を削除したり、または無効にしてオペレーティングシステムの環境を強化する必要があった。

セキュリティ管理ツールは、ポリシー管理のための集中化し一貫性のある構造を提供するばかりでなく、システムポリシーの操作や自動強化機能も可能にしている。

2.2.3.2. Capability 機能

従来の UNIX は、特定のユーザにシステムの全特権を持たせており、重大な危険を招くような環境であった。Capability 機能は、root 権限のないプロセスが特権が必要なシステムリソースへのアクセスを認可するためのきめ細かな設定が可能である。Capability により、スーパーユーザが必要としていた条件や侵害される危険性を軽減することができる。

2.2.3.3. アクセス制御リスト

従来の UNIX 環境下でのファイルのパーミッション設定は、「所有者(単一ユーザ)」、「システム管理が設定したグループ」、「その他のユーザ」の 3 種類に限られており、設定方法が非常に限られていた。アクセス制御リストは、ファイルやその他のシステムオブジェクトに関連した任意の権限をきめ細かに設定することを可能にしている。

2.2.3.4. セキュリティイベント監査機能

従来の UNIX 環境では、認証のイベントやエラー条件のログは非常に少なく、システムイベントのログ程度を記録していた。高信頼環境下では、異常な状態に関連したセキュリティイベントの監査ばかりか、管理者に必要なログ、侵入検知のログが要求される。

2.2.3.5. 強制アクセス制御

従来の UNIX 環境では、ユーザの対話環境下で広範囲の任意の制限が許可されたオープンなアクセスが可能であった。強制アクセスポリシーは、システム管理者がシステムのデータやユーザのプライベートなデータを保護するためのポリシー定義を可能にする。

2.2.3.6. TrustedBSD Feature Set

TrustedBSD の機能セットは、いくつかのコミュニティからの多くの要求を取り込ませた。これは、オレンジブックやコモンクライテリアに従った高信頼オペレーティングシステムコミュニティやセキュリティ機能の向上を目指す FreeBSD コミュニティの広範囲な要求も含まれている。この機能は、既存コードの改善、開発工程の改善、新規機能の改良を行っている。

(1) 一貫性と正確性の向上

組み込み予定の新セキュリティモデルおよび広範囲に修正するセキュリティサブシステムの重要部分は、正確に実装されることを確認しながら行われている。FreeBSD 開発者のコミュニティは、セキュリティ機能をカバーするテストスイートを持っていなかったため、新たに脆弱性が組み込まれていないかを確認するテストプログラムが開発された。

同等のセキュリティチェック機能が複数の方法で実装されるだろうとい

うのが開発当初の意見であった。例えば、あるアクセス制御チェックはプロセスのデバッグ用に開発され、もう 1 つはプロセスファイルシステムの実装に開発された。特に、プロセス間の認証やファイルシステムの権限評価の分野ではソースコードを開発者間で共有せず、またインクリメンタルな開発スタイルはシステム内での機能の矛盾や文書化されていない機能を作り込む原因になった。

「正確性」は、セキュリティプロジェクトでは明らかに重要な部分である。しかし、その正確さを検証する適切なツールがない限り、それを成し遂げるには至難の業である。アクセス制御の「一貫性」は、目的を達成するためには注意深くドキュメント化し、広範囲で厳密なテストが必要である。

(2) 抽出化とモジュール化の向上

アクセス制御チェックのソースコードの実装を共通化して組み込むことは、システムの一貫性を改善するばかりか、ポリシー論理の入れ替えに新しいモデルも容易に組み込むことができる。同様に、サブジェクトのラベル(プロセスの証明書)やシステムオブジェクト(ファイル、ネットワークのパケットとインタフェースおよびカーネル管理下のサービス等)に関連した部分を抽出し改善することは、MAC ポリシーのような広範囲のセキュリティ機能の一貫性を高めることができる。

(3) セキュリティ要件をサポートする一般サービス

TrustedBSD の多くのコンポーネントは、単にセキュリティ機能に関する処理を行うばかりでなく有用な機能を持たせている。1 つの例がファイルシ

ステムの拡張属性機能である。セキュリティ機能には、ファイルシステムオブジェクトに関連付けられたセキュリティラベルという付加情報を格納しなければならないが、拡張属性機能ではこのラベル情報以外のメタデータのバージョン、簡易情報、ファイルのアイコンの退避にも使用される。

(4) きめ細かな任意アクセス制御

UNIX のパーミッションは作成したオブジェクトの任意のプロテクションをユーザが設定できる。しかし、このメカニズムは複雑なセキュリティ設定を要求するような大規模な環境下では設定の自由度と表現方法が乏しすぎる。

POSIX.1e のアクセス制御リスト(ACL)は、オブジェクトの所有者にファイルシステム上のオブジェクトにきめ細かな保護設定を指定できる。ACL の実装は、従来のパーミッションモデルとの高い互換性も提供することができる。ACL は、比較的簡単な実装作業であるばかりか、多くのオペレーティングシステム使用者に期待されている機能でもある。

(5) きめ細かな特権モデル

従来の UNIX のセキュリティモデルは、システムの特権がひとりの root ユーザに集中していることが問題であった。これは、多くのアプリケーションに不必要な高レベルの特権を集中させたり、最小特権の方針に背き、攻撃されやすいアプリケーションを放置することになる。

POSIX.1e の Capability は、root ユーザに与えられていた特権を複数の論理的な構成に分割し権限をプロセスのユーザ ID から切り離している。プロセスは、Capability の可用性・継承性・有効性を管理し、侵害時の被害範囲をできるだけ小さくする。この実装は、プログラム(バイナリファイル)

に Capability を持たせるために拡張属性を利用している。

(6) 強制アクセス制御（MAC)

強制アクセス制御は、システムのサブジェクトとオブジェクトの関連を強制的なセキュリティポリシーとして定義することをセキュリティ管理者に許可する機能である。従来の MAC ポリシーは、Biba 保全モデル、区分モデル、TE(Type Enforcement)や DTE(Domain and Type Enforcement)のような一般的なポリシー機構のモデルと同様な多層セキュリティ(MLS)を含んでいた。

一般的に非商用の UNIX 系システムでは、従来の高信頼オペレーティングシステムにあるようなセキュリティ機能が不足していた。FreeBSD では、既存のセキュリティモデル(Jail モデルを含む)に準じた MAC の処理が組み込まれるようになってきたが、拡張属性にラベル情報を恒久的に格納するようなラベル機能とアクセス制御機能の改善が要求されている。

(7) Plugability

アーキテクチャ上の長期的な目標は、機能抽出とモジュール化の推進を続けることにより、新しいセキュリティ機能を迅速に導入することである。この最も重要な要素は、既存のセキュリティモデルを紐解いて、アクセス制御決定を生成するために構成された独立な構造のコンポーネントに導入することである。新しいモデルの組み込みや統合が簡単になるということは、開発工程を簡素化しより新しいセキュリティ機能の開発を促進するということである。この目標は、設計と実装の両方に本質的な要求を提示し、要求性能と利便性を維持することである。

(8) ドキュメンテーションと教育

TrustedBSD には、システム開発者とユーザの両方に関連した膨大な数の新しいインタフェースとサービスが組み込まれている。持続的で実のある努力は開発者用ドキュメントの更新や保守に精力を注ぎ込み、FreeBSD にその機能をフィードバックすることである。また、ユーザ用のドキュメントを最新にすることも忘れてはいけない。ドキュメントが不十分だったため、従来の高信頼オペレーティングシステムの機能は、ほとんどのシステムプログラマやユーザにはあまり知られていなかった。詳細なドキュメントは、実装した機能が正確に分かるように維持したり、オペレーティングシステム使用者に最新の機能を紹介するのにきわめて重要である。

2.2.4. セキュリティレベル

TrustedBSD のセキュリティレベルは、B1 クラスに相当する。

2.3. OpenBSD

2.3.1. 提供元

OpenBSD の提供元は、OpenBSD project である。

現在 OpenBSD は、表 2.3-1に示すプラットフォームを公式にサポートしている。

表 2.3-1 OpenBSD のサポートプラットフォーム

プラットフォーム	説　明
alpha	DEC Alpha プロセッサベースのシステム
amiga	Amiga m68k ベースのシステム（MMU 必須）
hp300	Hewlett-Packard HP300/HP400
i386	Intel i386 アーキテクチャ準拠のシステム
mac68k	Apple 社のシステム（MC680x0 プロセッサベース）
macppc	Apple 社のシステム（PowerPC ベース）
mvme68k	Motorola MVME147/16x/17x68K VME カード
sparc	Sun Microsystems 社の Sun4、Sun4c、Sun4m モデル
sparc64	Sun Microsystems 社の SUN UltraSPARC ベースのシステム
sun3	Sun Microsystems 社の 68020 ベースの Sun3 モデル
vax	DEC の VAX コンピュータ

2.3.2. 状況

OpenBSD はボランティアによって開発されている。現在、プロジェクトの開発資金は、OpenBSD を収録した CD や OpenBSD のロゴをあしらった T シャツの売上と寄付によって賄われており、その結果として著作権フリーであることが保証されている。現在の最新バージョンは 2001 年 12 月 1 日にリリースされた OpenBSD3.0 である。

OpenBSD では「監査プロセス」と呼ばれるセキュリティチェックが常に行われている。これは、6～12 人程度のメンバーで構成される監査チームが、OpenBSD におけるセキュリティ問題の有無を調査し、セキュリティホールが見つかった

場合には迅速に対処するというものである。この際の調査では、重要なソフトウェアコンポーネントに対して、構成するファイルの１つ１つを総合的に分析していくという形態が採られている。この分析過程で発見された問題は、その性質に関らず積極的に修正される。つまり、ソフトウェアコンポーネントで発見された不具合は、その不具合が悪用される（つまり、セキュリティホールになり得る）可能性があるかどうかに関係なく、単純にソフトウェアのバグという観点で積極的に修正が施されることになる。この監査プロセスは、OpenBSD バージョン 2.0 リリースの直前と、バージョン 2.0 から 2.1 への移行の際に重点的に行われており、この段階で何千ものセキュリティ問題が修正されているとのことである。この結果として現在では、発見されるセキュリティ問題の数は少なくなり、かつよりマイナーで複雑なものであるとのことである。それでもなお、OpenBSD プロジェクトではこの監査プロセスに対して拘りを持っており、現在も継続して行われている。また、発見されたセキュリティ問題は完全に公開されるべきであるというポリシーから、OpenBSD の公式サイトではセキュリティ警告情報や、システムに対して施された変更ログが公開されている。

2.3.3. セキュリティ機能

OpenBSD におけるセキュリティ面の最大の特徴は、オペレーティングシステムの随所に暗号化機能が盛り込まれている点である。OpenBSD プロジェクトの本拠地であるカナダは、暗号ソフトの輸出に関して非常に寛容で、自由に入手できる暗号ソフトの輸出は自由に行って良いとしている。このため OpenBSD では、適切な輸出ライセンスの元で配布されていて自由に入手可能であり、かつそのライセンスも適切な暗号ソフトのいくつかをシステムに適用している。また、OpenBSD のスナップショットやリリース用バイナリの作成も、その輸出に規制を受けることのない国で行われており、ライセンス面で不適切な部分がないよう

に注意が払われている。

2.3.3.1. IPsec のサポート

OpenBSD では、初期バージョンである OpenBSD バージョン 2.1 のリリース以来、そのネットワークプロトコルスタックに、VPN を実現する最も有力なプロトコルである IPsec をサポートするプロトコルスタックを実装している。

2.3.3.2. OpenSSH(Open Secure Shell)

OpenBSD ではバージョン 2.6 以降で OpenSSH を含んでいる。この OpenSSH には rlogin や telnet を置き換える ssh、rcp や、ftp を置き換える scp といったプログラムが含まれており、これらを用いることでリモートホストとやり取りする情報を暗号化することが可能となる。OpenSSH は、商業目的以外に関して著作権フリーであった「ssh[11] 1.2.12」を母体として、特許やライセンス的な問題を含んでいるコンポーネントを直接ソースコードから取り除き、これらのコンポーネントの提供する機能を OpenSSL[12]等の外部ライブラリから取り込むことによって、母体である ssh 1.2.12 の厳しいライセンス制限を取り除いたものである。古いバージョンの ssh を母体としているが、バグフィックスおよび機能拡張が施されているため、現在では ssh プロトコル 2.0 もサポートしており、OpenSSH は主要な ssh プロトコルである ssh プロトコル 1.3、同プロトコル 1.5、同プロトコル 2.0 をサポートしていることになる。

OpenSSH には以下の機能がある。

---

[11] 現在、Datafellows 社（http://www.datafellows.com）で販売されている「F-Secure SSH」の初期バージョンにあたる。
[12] Open Secure Sockets Layer の略。Secure Sockets Layer および世界標準の暗号プロトコル Transport Layer Security を実装したオープンソースのツールキット。

- 暗号化アルゴリズム 3DES および Blowfish のサポート
- X11 転送
- ポート転送
- 認証機能
- エージェント転送
- 相互運用
- SFTP クライアントおよびサーバのサポート
- データ圧縮

(1) 暗号化アルゴリズム 3DES および Blowfish のサポート

OpenSSH では暗号化アルゴリズムとして、3DES（トリプルデス）および Blowfish[13]を採用している。OpenSSH では、以下のようなプログラムを提供している。

- ssh
- sftp
- scp

ssh は従来の rlogin、rsh あるいは telnet と同様な機能を提供するリモートログインプログラム、sftp は従来の ftp と同様な機能を提供するファイ

[13] Bruce Schneier 氏によって考案された 448bit 可変長キーのフリー暗号アルゴリズム。

ル転送プログラム、scp は従来の rcp プログラムと同様な機能を提供するファイルコピープログラムである。これらのプログラムでは、リモートホストとやり取りされる情報が 3DES あるいは Blowfish によって暗号化される。また、認証方法として RSA を利用する場合にはリモートログイン時にパスワードをネットワーク上へ送出する必要がないため、セキュリティはより向上する。

(2) X11 の転送機能

xhost や xauth コマンドを利用するよりも安全に、ネットワーク経由で X ウィンドウシステムを利用する機能である。通常、X サーバへ接続するプロセスを制限するコマンドとして xhost や xauth といったコマンドが用意されている。xhost コマンドは、あるホストで稼動している X サーバへ接続可能なホストの制限を設定するコマンドであり、xauth コマンドは X サーバへ接続可能なクライアントをパスワードで制限するコマンドである。xauth コマンドの場合、パスワードを知られてしまうと他人がその X サーバに接続可能であり、そもそも xhost によって許可されているクライアントからの接続の場合には xauth による保護が効かないという問題が存在する。OpenSSH の提供する ssh プログラムによってリモートログインした場合、ログイン先で X ウィンドウアプリケーションを実行すると、ログイン先で稼動している OpenSSH のサーバ（デーモン）プログラムによる認証が実行され、起動した X ウィンドウアプリケーションがローカルの端末画面に表示される。この際、ログイン先のホストにおける環境変数 DISPLAY は OpenSSH のサーバプログラムによって自動的に設定されるため、ユーザが環境変数

---

http://www.counterpane.com/

DISPLAY の設定を改めて行う必要はない。

(3) ポート転送

ポート転送は、暗号化された経路を通じて TCP/IP の接続をリモートホストへ転送する機能である。標準的なインターネットアプリケーションはこの機能によってその安全性が向上する。

(4) 認証機能

次にあげる認証方法をサポートしている。

- RSA ベースのホスト間認証を含めた rhosts 認証
- 純粋な RSA 認証
- S/Key[14]を用いたワンタイムパスワード
- Kerberos による認証

(5) エージェント転送

認証エージェントとは、ユーザが利用する PC やローカルのホスト上で稼動し、ユーザの RSA 認証のための鍵を保持するプログラムである。OpenSSH では、認証要求の接続を自動的に認証エージェントへ転送するので、RSA 認証のための鍵をユーザが利用しているローカルマシン以外に用意しておく必要がない。認証鍵は認証エージェントのみが保持するため、OpenSSH での認証プロトコルではネットワークに対して認証鍵を掲示してしまうことは決してない。認証鍵は、ユーザのローカルホストで稼動している認証エージェントが正しい鍵を保持していることを確認する際にのみ利用される。

(6) 相互運用

OpenSSH のバージョン 2.0 では、SSH リモートプロトコルのバージョン 1.3、1.5 および 2.0 をサポートしている。これにより殆どの UNIX や Windows およびその他の商用 ssh との通信が可能となっている。

(7) SFTP クライアントおよびサーバのサポート

OpenSSH のバージョン 2.5.0 では、SFTP(Secure File Transfer Protocol)が完全サポートされている。

(8) データ圧縮

OpenSSH では暗号化の前にデータの圧縮を行う。この機能により、キャパシティの低いネットワーク接続で利用した際の性能の低下を低く抑えることが可能となっている。

2.3.3.3. 暗号化ハッシュ関数

OpenBSD では、MD5、SHA1、RIPEMD-160 が暗号化ハッシュ関数として利用されており、以下のような場面で利用されている。

[14] http://www.telcordia.com/

- S/Key によるワンタイムパスワード生成時
- IPsec において IPsec パケット内の IP ヘッダ情報の一貫性を証明するための認証ヘッダ生成時
- MD5 パスワード
- TCP SYN クッキーサポート時
- libssl におけるメッセージのデジタル署名

2.3.3.4. 暗号変換

暗号変換は、データを暗号化する際、および、複合化する際に利用されるもので、通常暗号化の際には暗号化キー、複合化の際には複合化キーを利用する。OpenBSD では、DES、3DES、Blowfish、Cast といった暗号変換を、カーネルやユーザランドプログラムのために提供している。

2.3.3.5. 暗号化ハードウェアのサポート

バージョン 2.7 以降の OpenBSD では、乱数発生装置など、一部の暗号化ハードウェアをサポートしている。

(1) IPsec における暗号化処理の分離

OpenBSD のサポートする IPsec プロトコルスタックでは、IPsec パケット生成時の一連の処理シーケンスから暗号ペイロード部分を外部に抽出した実装としている。多くのソフトウェアによる IPsec プロトコルスタックでは、パケットを生成する度に暗号化処理を稼動させなければならない。これは言わば反復処理である。IPsec パケット生成の際に暗号化ハードウェア

を利用した場合に、ハードウェアを適切に、かつ高速に利用するために、OpenBSD では暗号ペイロード部分とそれ以外の部分を分離している。分離したことにより、暗号化ハードウェア利用時だけではなく、ソフトウェアにより処理する場合でもパフォーマンスの向上を実現しているとのことである。

(2) Hifn 社「Hifn 7751」プロセッサのサポート

Hifn 社[15]の Hifn 7751 プロセッサは、IPsec パケットにおけるデータ圧縮、暗号化および認証をハードウェアでサポートする暗号化プロセッサである。OpenBSD では本プロセッサを搭載した暗号化カード[16]をサポートしており、OpenBSD プロジェクトによる独自ドライバが提供されている。本プロセッサを搭載した暗号化カードを IPsec パケットの送受信双方のホストで稼動させた場合の処理パフォーマンスは、送受信双方のホストで Intel 社[17]プロセッサ PentiumIII-550MHz クラスでのソフトウェア処理と比較して、600%近い性能向上が期待できるとのことである。

(3) Hifn 社 「Hifn 6500」プロセッサのサポート

Hifn 社の Hifn 6500 プロセッサは RSA、DSA、DH のアルゴリズムの処理をハードウェアでサポートし、かつ乱数ジェネレータを有したプロセッサである。OpenBSD プロジェクトでは、現在このプロセッサの詳細情報を入手済

[15] http://www.hifn.com/
[16] 例えば、PowerCrypt 社（http://www.powercrypt.com/）の PowerCrypt
[17] http://www.intel.com/

みであり、将来サポートされる予定である。

(4) Broadcom[18]社「BCM5805」セキュリティプロセッサのサポート

Broadcom 社の BCM5805 プロセッサは、IPsec の完全なハードウェアアクセラレーションを提供するシングルチップのプロセッサである。OpenBSD バージョン 2.7 直後に試験的サポートを実現した後、OpenBSD バージョン 2.8 では本プロセッサのもつ乱数ジェネレータも利用できるようになっている。生成された乱数はカーネル内のエントロピープールで利用可能である。本プロセッサの大まかな性能は、Hifn 社の Hifn7751 の 4 倍程度高速であるとのことである。

(5) Securelink 社[19]「PCC-ISES」チップのサポート

Securelink 社の PCC-ISES チップは暗号化アクセラレーションチップである。このチップのドライバは現在開発中である。現在は、このチップの乱数ジェネレーション機能を利用でき、生成された乱数はカーネルのエントロピープールで利用可能である。

(6) SafeNet 社[20]「SafeXcel 2141」チップのサポート

SafeNet 社の SafeXcel 2141 チップは、暗号化アクセラレーションチップである。本チップのドライバは現在開発中である。

---

[18] http://www.broadcom.com/
[19] http://www.securelink.com/
[20] http://www.safenet-inc.om/

(7) 3com 社[21]「3c990」チップのサポート

3com 社の 3c990 チップは暗号化アクセラレーションをサポートするイーサネットチップである。本チップをサポートするドライバは既に開発されており、マイクロコードに関するフリーのライセンスが得られた段階でリリースできるとしている。

(8) Intel 社 IPsec カード

現在開発されているドライバは、Intel 社からの正式なドキュメント提供を受けずに開発されたものであり、その信頼性は低い。

(9) Intel 社「82802AB/82802AC Firmware Hub」乱数ジェネレータ

Intel 社の 82802 FWH チップ(Intel 社製 CPU 向けチップセット i810、i820、i840、i850、i860 に含まれるチップ）ではハードウェアによる乱数生成をサポートしており、OpenBSD では既にこのチップをサポートしている。しかし、このチップの乱数生成能力は低く、高性能な IPsec ではもっと高い性能の乱数生成能力を必要とするとのことである。

(10) OpenSSL の機能拡張

OpenBSD プロジェクトでは、RSA や DSA をハードウェアで処理するプロセッサを積極的にサポートしていく姿勢を見せており、これらのプロセッサを搭載した各メーカーの暗号処理カードの持つ機能を外部ライブラリである OpenSSL にエクスポートしていくとしている。これは、ユーザ空間で

[21] http://www.3com.com/

稼動し、暗号処理を必要とするプログラムからこれらの暗号処理カードの機能を容易に利用できるようにすることを目的としたものである。

2.3.4. セキュリティレベル

OpenBSD のセキュリティレベルは、C1 クラスに相当する。

2.4. PitBull

2.4.1. 提供元

PitBull は、米国 Argus 社によって開発、販売されている。

商用を目的に開発されており、ベースとなる OS(SUN Solaris、IBM AIX、Linux)と 100%の互換性を実現している。また、Solaris、AIX、Linux アプリケーションとの 100%の互換性も実現している。PitBull を導入するときに、OS 自体を入れ替える必要がないのが特徴である。

PitBull には、PitBull Foundation、PitBull .comPack と PitBull LX の３製品がある。

PitBull Foundation は、Sun Solaris と IBM AIX に TrustedOS の機能を付加する製品である。

PitBull .comPack は、PitBull Foundation をベースにした Web サービス向けの製品で、PitBull Foundation にアプリケーション連帯用ゲートウェイ機能などを追加するものである。

PitBullLX は、PitBull .comPack 相当のセキュリティ機能を実現する製品である。PitBull .comPack とは独立しており、PitBull Foundation が対応していない Linux にも対応している。PitBull LX はカーネルモジュールの形でインストールされる。

2.4.2. 状況

現在のリリース状況は、以下のようになっている。

(A) PitBull Foundation and .comPack

- PitBull Foundation4.0 and .comPack4.0　Solaris8 版
- PitBull Foundation3.1 and .comPack2.2　IBM AIX4.3.3 版

(B) PitBull LX

- PitBull LX1.1.0 Linux 版、Linux カーネル 2.2.19 と 2.4.5 に対応
- PitBull LX1.0 Solaris 版、Solaris 8 に対応
- PitBullLX1.1.8.0 IBM AIX 版、AIX4.3.3 に対応

2.4.3. セキュリティ機能

PitBull では MLS による MAC、最小特権、役割分割(role)を実装している。

2.4.3.1. 機能概要

(1) アクセス制御機能

PitBull では、MLS による MAC を実装し、カーネルよって強制的に実施される。MAC は、すべてのサブジェクトおよびすべてオブジェクトに付与される機密ラベルを使って行なう。機密ラベルは、機密分類とコンパートメントからなる。機密種別は階層的に分類されたものである。コンパートメントは、階層的に分類するのではなく分野を表す。例えば、アプリケーションを区別するために使われ、異なるコンパートメント間のアプリケーション同士のアクセスは拒否される。

機密ラベルによるアクセス制御では、サブジェクトとオブジェクト間の機密ラベルを比較し、どちらの機密ラベルが優位であるかによって許可するか、拒否するかが決まる。機密ラベルが優位であるとは、以下の条件を満たした場合のみである。

- 機密ラベルの機密種別が、もう一方の機密種別と同等か、それよりも上の階層であるとき。
- 機密ラベルのコンパートメントのすべてが、もう一方のコンパートメントに含まれているとき。

ファイルなどのオブジェクトの内容を読み込むには、サブジェクトの機密ラベルがオブジェクトの機密ラベルより優位である必要がある。逆に、書き込みを行なう場合は、オブジェクトの機密ラベルが優位でなければならない。

ユーザに対しては、クリアランスと呼ばれるユーザがアクセスできる下限と上限の機密ラベルと、ユーザがログインしたときのデフォルトの機密ラベルを付与する。これによって、ユーザがアクセスできるファイルなどの範囲が決定する。また、ディレクトリに対しては、下限と上限の機密ラベルを付与する。このディレクトリは MLD（マルチレベルディレクトリ）と呼ばれる。これによって、ディレクトリに異なる機密ラベルを持つファイルを作成することができる。また、ユーザが MLD にアクセスすると、ユーザの機密ラベルと同じ機密ラベルのファイルおよびサブディレクトリしか表示されない。

(A) 実装

ファイルシステムオブジェクトとプロセスへの機密ラベル付けは、ファイルシステムオブジェクトに関しては inode で、プロセスはプロセス構造体で行っている。

すべてのファイルシステムオブジェクト（ファイル、ディレクトリ、

リンク、名前付きパイプなど）は inode 内に、セキュリティ情報へのリファレンスがある。プロセスは、プロセス構造体に各々のプロセスのセキュリティ情報へのポインタがある。

メッセージと IPC（セマフォ、メッセージキュー、共有メモリ）に対してもセキュリティ情報が付加される。

メッセージは、メッセージのそのものにセキュリティ情報が付属している。例えば、ネットワークでのパケットは、ホストに出入りするすべてのパケットの中にセキュリティ情報が付加される。セキュリティ情報を持たないパケットを受信した場合、ホストに届いた時点でラベル付けを行う。

IPC のそれぞれのオブジェクトもセキュリティ情報を持つので、カーネルは要求が許可されているかどうか決定できる。

(2) 特権機能

(A) 最小特権

PitBull は、root が持つ特権を分割して、機能ごとにグループ化および階層化している。例えば、DAC パーミッションを無視できる権限は、DAC 特権としてグループ化されており、PV_DAC_O（所有者）、PV_DAC_R（読み取り）、PV_DAC_W（書き込み）、PV_DAC_X（実行）などの特権に分割される。そして、分割された各特権は、DAC による制限をすべて迂回できる PV_DAC 特権の下位特権になる。このように階層化による特権分割で、必要最低限の特権のみをプロセスに与えることが可能となる。

(B) 特権継承

プロセスと実行ファイルは、それぞれ異なる特権セットがある。これら

の特権セットによって子プロセスに継承する特権が決定する。

プロセスが持つ特権セット

- 有効権限セット

  プロセスがその時点で使用できる特権セット

- 上限権限セット

  プロセスが自身の有効権限セットに置かれる可能性がある権限セット

- 制限権限セット

  プロセス自身または、子プロセスに継承を許可する権限セット

実行ファイルが持つ特権セット

- Innate 特権セット

  この特権セットと親プロセスの制限特権セットに特権があれば、子プロセスは、上限特権セットに特権を持つ。もし、実行ファイルが FSF_EPS フラグを持っていれば、子プロセスは、有効権限セットに innate 特権セットを持つ。

- プロキシ特権セット

  この特権セットと親プロセスの上限特権セットに特権があれば、子プロセスは、上限特権セットに特権を持つ。もし、実行ファイルが FSF_EPS フラグを持っていれば、子プロセスは、有効特権セットにプロキシ特権セットを持つ。もし、プロキシ特権セットが空ならば、子プロセスは、特権を継承しない。

- 認可特権セット

  認証を持つユーザがオーナのプロセスだけを利用できる特権であ

る。許可特権セットは、ファイルの許可特権セットにある認証を持つユーザだけに与えられる。

(3) 役　割　分　割　（　Role compartmentalization)

特権と許可の組み合わせによって、役割区分を強制する。特権がプロセスや実行可能ファイルに与えられるのに対して、許可とはユーザに対して付与される。ユーザは特権が許可されていなければ、特権のあるファイルの実行許可があってもユーザに対して特権が許可されていないので実行できない。

デフォルトでは、システム管理者の責務は、情報システムセキュリティオフィサー（ISSO）許可、システム管理者（SA）許可、システムオペレータ（SO）許可を割り当てられたユーザに分けられる。ISSO は、PitBull セキュリティ機能を管理し、SA は、元々の UNIX 機能を管理し、そして SO はデバイスとハードウェアを管理する。ユーザやソフトウェアの追加作業は、これら３つの役割が必要となるので、UNIX 環境での root のような絶対的なユーザが存在しなくなる。

(4) 監査機能

監査可能なイベントはあらかじめ監査イベントとして定義されており、新たに追加することもできる。また、いくつかの関連する監査イベントをグループ化して、監査クラスとして定義されている。例えば、監査クラスとして lo（ログインに関するクラス）が定義されており、監査イベントとして、コンソールログイン、telnet ログイン、リモートログイン、ログア

ウトなどが含まれる。PitBull は、すべてのユーザに何らかの監査クラスを割り当て、監査を行う。

(5) コマンドセンタ（Solaris 版のみの機能）

コマンドセンタは、PitBull の設定を行うためのブラウザベース GUI ツールである。このツールにより、サブジェクトとオブジェクトへのラベル付けや、承認などの設定を行える。コマンドセンタは、Netscape Entriprise Server に付いてくる ORB を使う。ただし、ORB が SSL に対応していないため、システムコンソールからしか使うことができない。

2.4.3.2. PitBull .comPack

PitBull .comPack は、PitBull Fondation 上で、Web サービスに必要となるモジュールを提供するものである。PitBull .comPack は以下に示す７つのモジュールから構成される。

(1) SCE(UDE)

SCE(UDE とも呼ばれる)は、特定のポートで、ネットワークからのすべての情報をいったん受け取り、指定されたサーバやデーモンに転送する。SCE は、ルールにしたがって付けた機密ラベルと、リクエスト URL を元に転送先を決める。SCE は、OS レベルで動作し、外部から入ってきたパケットを隔離されたコンパートメントに送信できる唯一のメカニズムである。

(2) セキュアゲート

セキュアゲート（以降 SG）は、異なるコンパートメント間のアプリケーション間通信をあらかじめ決められた制限の元で許可し、Web サーバなどの

フロントエンドサーバが SG を越えて、アクセス許可がないコンパートメントにアクセスすることを防ぐ。SG は、TCP、UDP と UNIX ドメインプロトコルをサポートしている。

(3) セキュア認証モジュール

セキュア認証モジュール（以降、SAM）は、ユーザ認証で必要となる機密情報へのアクセスを制限するために、SAM のコンパートメント内で Web サーバの認証を行なう。

(4) セキュア CGI モジュール

セキュア CGI モジュールは、Web サーバから CGI 機能を分離する。CGI はこのモジュールによって異なった機密ラベルとサーバとは違うコンパートメントで実行される。よって、CGI を使った攻撃を受けても、CGI を実行しているプロセスがコンパートメントに隔離されているので、HTML の改ざんなどを防ぐことができる。

(5) セキュアプログラムラウンチャ

SPL は、特定のユーザに対して、ホストのセキュリティ設定を操作するのに必要な特権の承認を受けないで、セキュリティ操作をするプログラムの実行を許可するものである。

(6) 信頼された SSH

システム管理は、SSH を使って遠隔保守ができる。また、SSH によるログイン時、システム管理者に機密ラベルを付与することを許可する。

(7) IPSec

PitBull サーバ間の末端同士の完全なセキュリティを実現するために、

IPSec を実装することでネットワークトラフィックの暗号化を提供する。許可と機密性のエリアに付加的なセキュリティを提供する。

2.4.3.3. PitBull LX

PitBull LX は、インストール先のオペレーティングシステムのカーネルモジュールとして提供される。PitBull LX は、システムの必要な部分にだけ、例えば、Web サービスのところだけといったように、セキュリティ拡張を施すことが可能である。それ以外の部分は、通常の UNIX 環境と同様に動作する。

PitBull LX は、アプリケーションごとに、プロセス、ユーザ、ファイルそしてネットワークインタフェースを他の区分から完全に隔離することで、あるアプリケーションを隔離された安全区画に閉じ込めることができる。PitBull LX では、これらの区画をドメインと呼ぶ。アプリケーションをドメインごとに閉じ込めることで、アクセスできるファイルなどを制限できる。

(1) アクセス制御

(A) DBAC

PitBull LX は DBAC によって、ドメインがアクセスできる資源を制限し、アプリケーションの隔離を行っている。DBAC は、２つのドメイン、オブジェクトのアクセス制御に用いるファイルドメインと、ネットワークデータのアクセス制御に用いるネットワークドメインからなる。以下にこれらのドメインについて説明する。

- ファイルドメイン

  ファイルドメインは、ドメイン名(パーミッション)と表現される。パーミッションには、UNIX でのファイルパーミッションと同じ r(Read)、w(Write)、x(eXecute)を使用する。ファイルアクセスドメ

インは、ファイルオブジェクトとプロセスに付与される。プロセスに付与されるとプロセスがアクセスできるドメインとパーミッションを意味し、ファイルオブジェクトに付与されるとそのファイルにアクセスできるドメインとパーミッションを意味する。

- ネットワークドメイン

  ネットワークドメインは、ドメイン名（net）と表現される。net はネットワークオブジェクトを利用することを意味する。ネットワークドメインはネットワークオブジェクトとプロセスに付与される。

(B) セキュリティフラグ

セキュリティフラグは、サブジェクトとオブジェクトの振る舞いを制御するものである。また、セキュリティフラグは、スーパユーザ特権を制限する要素の一つでもある。セキュリティフラグには以下の４つのカテゴリがある。

- プロセスセキュリティフラグ

  プロセスセキュリティフラグは、プロセスの振る舞いを制限するものである。このカテゴリには、PitBull LX によってプロテクトされたプロセスを示すフラグ、PitBull LX によってプロテクトされたファイルにしかアクセスできないことを示すフラグ、PitBUll LX によってプロテクトされたネットワークデータにだけにしか反応しないことを示すフラグなどがある。

- ファイルセキュリティフラグ

  このカテゴリには、PitBull LX によってプロテクトされたファイルであることを示すフラグ、実行可能なファイルを実行するプロセスが、実行ファイルの PitBull LX 属性を変更するかもしれないことを

示すフラグなどがある。

- ネットワークセキュリティフラグ

  このカテゴリには、PitBull LX によってプロテクトされたネットワークデータであることを示すフラグ、PitBull LX によってプロテクトされたプロセスによって書かれたデータを示すフラグがある。

- IPC セキュリティフラグ

  このカテゴリには、PitBull LX によってプロテクトされた IPC オブジェクトであることを示すフラグがある。

(2) ユーザ設定

PitBull LX によって拡張された PAM を使って、ユーザにアクセスできるファイルドメインとネットワークドメインを付与する。そして、ユーザプロセスに付与するセキュリティフラグを設定する。また、コンソールログインか、リモートログインかによって、付与する各ドメインとセキュリティフラグが変えられる。

もし、ユーザに各ドメインを与えなければ、PitBull LX によってアクセス制御されているファイルにはアクセスできない。

2.4.4. セキュリティレベル

PitBull Foundation のセキュリティレベルは、B1 クラスに相当（ITSEC E3 認定)する。また、PitBull LX のセキュリティレベルは、C2 クラスに相当する。

2.5. hp virtualvault

2.5.1. 提供元

hp virtualvault の開発元は、米国 Hewlett-Packard Company である。

hp virtualvault は、インターネットアプリケーションのためのセキュアなプラットフォームを提供するウェブサーバプラットフォームである。HP-UX をベースとして、セキュリティをさらに向上させたオペレーティングシステムである「The VirtualVault trusted operating system (VVOS)」をその OS として採用し、「iPlanet Web Server – Enterprise Edition 4.1」(virtualvault 4.1) あるいは「Apache Web Server version 1.3.14」(virtualvault 4.5) をセキュアに統合したウェブサーバプラットフォームである。

2.5.2. 状況

現在の最新バージョンは 4.5 である。

ここ最近のリリース状況は以下のとおりである。

- 2001 年 4 月 9 日　　hp virtualvault 4.5 発表
- 2001 年 2 月 13 日　　hp virtualvault 4.0 発表

また、2002 年の初頭には Linux 版が出荷される予定となっている。

2.5.3. セキュリティ機能

2.5.3.1. 特権機能

(1) 最小特権メカニズム

hp virtualvault が採用している VVOS は最小特権メカニズムを実装している。これによって、ルートユーザの持つ特権は 50 に及ぶ、別個の特権に細分化されており、あらゆる権限を有するルートユーザは無効化されてい

る。VVOS 上で稼動するアプリケーションは、この最小特権メカニズムによって、正常稼動に必要最低限な特権のみが許可されるため、たとえ root ユーザのユーザ ID を利用して実行されたプロセスであっても一切の特権が与えられることはない。

(2) 特権継承の制限

VVOS でサポートするファイルシステム上のファイルには、従来のファイル属性に加えて新たな属性が追加されており、特権情報を保持することが可能となっている。この新たなファイル属性に対応しているアプリケーションであれば、そのコードの中で特権を変更したり、この新たなファイル属性をサポートしていない既存のアプリケーションの実行時に自動的に特権を割り当てることが可能となっている。あるプロセスが新たなプログラムを、そのプログラムの特権の属性に基づいて実行する際には、その都度特権の再設定が行われるため、特権の継承が禁止される。

2.5.3.2. 強制アクセス制御

VVOS には、データやアプリケーションのコンポーネントが常駐する区域（パーティション）を定義することを目的とした、リソース属性が用意されており、あらゆるプロセス、ファイル、プロセス間通信により利用されるリソース、ネットワークインタフェースやその他のデバイスには、アプリケーションレベルでは制御することのできない、このリソース属性が割り当てられる。あるパーティションに割り当てられているプロセスは、他のパーティションに割り当てられているリソースに対して、許可なくしてアクセスは許されない。この機構により、データやアプリケーションのコンポーネントは VVOS 以外からの要求に対して保護されており、効果的に分離されている。hp virtualvault ではこの機構の

ことを「データパーティショニング」と呼んでいる。

hp virtualvault におけるデータパーティショニングでは、「SYSTEM」および「SYSTEM_HI」と呼ばれる２つのシステムパーティションと、「INSIDE」および「OUTSIDE」と呼ばれる２つのアプリケーションパーティションが定義されている。

(1) SYSTEM パーティション

SYSTEM パーティションには VVOS の関連ファイル、つまりオペレーティングシステムファイルが含まれる。また、アプリケーションが稼動する際に参照する静的なアプリケーションファイル、例えば設定ファイル、HTML ファイル、テンプレートあるいは CGI スクリプトのような実行可能ファイルといったものをこのパーティションに割り当てることも可能となっている。SYSTEM パーティションに割り当てられたこれらのリソースは、後述する INSIDE パーティションあるいは OUTSIDE パーティションに割り当てられたアプリケーションによる読み込みあるいは実行は可能であるが、変更は許可されないため、アプリケーションファイルの完全性を確実に保護することが可能となり、システム上のウェブページ、テンプレート、プログラムあるいは設定ファイルの不正な書き換えを阻止することが可能となっている。

(2) SYSTEM_HI パーティション

SYSTEM_HI パーティションは、システムの監査による証跡を保持するために利用されるアクセス不可能なパーティションであり、システムへの侵入者がその侵入の証跡を隠蔽することを妨げる。

(3) INSIDE および OUTSIDE パーティ

ション

INSIDE パーティションおよび OUTSIDE パーティションは、VVOS 上で稼動するアプリケーションのコンポーネントを、イントラネット側への直接のアクセスを許可する内部接続可能領域（INSIDE パーティション）と、インターネット側からのアクセスを受け付けるものの、イントラネット側への直接のアクセスが許可されていない外部接続可能領域（OUTSIDE パーティション）に分離する働きをする。INSIDE パーティションと OUTSIDE パーティションの間は「エアギャップ型ホスト内ファイアウォール」と呼ばれる論理エアギャップが設けられており、INSIDE パーティションと OUTSIDE パーティション間の通信は両パーティション間に設けられた「高信頼性アプリケーションゲートウェイ(TGA:Trusted Gateway Agent)」によって行われる。この TGA には、INSIDE パーティションと OUTSIDE パーティションの境界を越えるための特権が与えられており、この TGA を利用せずに両パーティション間で通信を行うことは不可能となっている。アプリケーションのコンポーネントをその性質を考慮して INSIDE パーティションおよび OUTSIDE パーティションに分離して割り当てることで、この TGA により安全性が保障されたアプリケーションコンポーネント間の通信のみが有効となり、インターネット側から侵入した攻撃プログラムのイントラネット側への侵入を防ぐことが可能となる。なお上記の論理エアギャップおよび TGA は、米軍技術 CMW[22]を満たすものである。

また、INSIDE パーティションおよび OUTSIDE パーティションに割り当て

[22] **US Defense Intelligence Agency (DIA), "Compartmented Mode Workstation Specification", Technical Report DDS-2600-6243-87.**

られたアプリケーションには、パーティションの変更や、SYSTEM パーティションに割り当てられたリソースに対して変更を施す特権は与えられない。たとえ、ファイル属性としてオーナーやグループのアクセス許可が与えられているアプリケーションであっても、システムがファイルへの書き込みを禁止する。

2.5.3.3. 暗号化

hp virtualvault は、インターネットフロントエンドのためのプラットフォームであり、インターネットクライアントに対して自身の身元を証明するための認証手段を提供しなければならない。hp virtualvault はその内部に、iPlanet Web Server – Enterprise Edition (旧名 Netscape Enterprise Server)を統合しており、この iPlanet Web Server – Enterprise Edition がサポートする認証機能を使用することによって、シークレットキー40 ビット、128 ビットの SSL やパスワードあるいはデジタル認証をサポートしている。

2.5.3.4. 監査機能

VVOS は、すべてのシステムコールとイベントログを SYSTEM_HI パーティションに保存する。この監査機能に対応するアプリケーションには、監査結果の記録の際に VVOS が提供している API を利用するように設定することで、SYSTEM_HI パーティション内の監査証跡に監査記録を行うためのシステムコール実行の特権を付与することができる。SYSTEM_HI パーティション内の監査証跡に対する監査結果の記録は、VVOS の一部である専用のユーティリティが行う。監査証跡に対して直接アクセスする特権を付与されているのはこのユーティリティのみであり、監査機能を利用するアプリケーションはこのユーティリティに対して監査依頼を発行する形となる。これにより、外部からの侵入者が自らの侵入証跡

を隠蔽することが不可能となっている。

VVOS では、リアルタイムでのモニタ機能も備えている。個々のイベントや、指定した時間内の一定数のイベントに対してのシステムの応答や、イベントのしきい値を越えた際のシステムの応答を設定することが可能である。この際のシステムの応答としては、電子メールの送信、システムコンソールへのメッセージの送信、外部プログラムの実行といったものをサポートしている。

2.5.3.5. その他

(1) 運用管理 GUI

hp virtualvault では非常にシンプルな運用管理 GUI を装備している。この運用管理 GUI は、高度な専門知識や操作スキルを必要とせず、また、ウェブサーバの管理に特化したものとなっている。この運用管理 GUI により運用管理担当者は、ウェブサーバの運用に必要最低限な操作だけが許可される形となり、運用管理担当者による操作ミスや不正操作を抑止している。

(2) セキュリティレベルの維持管理

近年のインターネットサービスに対する攻撃手法の多くは、オペレーティングシステムあるいはアプリケーションのセキュリティホールを利用したものが圧倒的多数である。結果として運用管理担当者は、オペレーティングシステムやアプリケーションのセキュリティホールに関する情報を日々確認し、最新の修正パッチをシステムに適用していかなければならない。しかし、オペレーティングシステムやアプリケーションに対する修正パッチというものは、システムの環境に応じてその有効性あるいは必要性の面でさまざまであり、この点を調査する作業もまた運用管理担当者に課せられることとなる。オペレーティングシステムとアプリケーションの双

方のセキュリティ管理と整合性管理は非常に複雑な作業であり、最悪の場合、自らの人為的なミスから不本意な障害を発生させてしまうことにもなり兼ねない。hp virtualvault は VVOS と呼ばれるオペレーティングシステムを採用し、このオペレーティングシステムのシステムレベルにて外部からの不正なアクセスを阻止する論理エアギャップを実装しているため、VVOS の単一管理だけのシンプルな運用管理が実現されている。

2.5.4. セキュリティレベル

hp virtualvault のセキュリティレベルは、B3 に相当する。

2.6. hp secure OS software for Linux

2.6.1. 提供元

米 Hewlett‐Packard 社(米 hp 社)が開発、販売している。

hp Secure OS Software for Linux（以降、HP-LX と呼ぶ。）は、インターネットで利用されるサービスのセキュアな実行環境を提供するため、Linux カーネルやアプリケーションレイヤをはじめとする重要なサーバコンポーネントを保護する機能が実装されている。おもなセキュリティ特徴は、プロセスがアクセスできるファイルの制限と通信路の厳密な制限による抑制（containment）や、OS とアプリケーションに対する攻撃の検知機能などがある。

HP-LX は、米 hp 社が Linux カーネルのセキュリティ強化を、米 hp 社と他ベンダによってアプリケーションの変更を行っている。販売されているパッケージには、Web サーバ Apache や、セキュリティ監視アプリケーションの Tripwire などが含まれる。米 hp 社が開発した Linux カーネル 2.4 に対するセキュリティ拡張の差分は、米 hp 社のホームページで公開されている。

2.6.2. 状況

2001 年８月に米国で、Linux カーネル 2.4 と Red Hat Linux7.1 をベースとした HP-LX1.0 をリリースした。日本 hp 社ではまだアナウンスされていない。

2.6.3. セキュリティ機能

HP-LX の主たる目的は、インターネットサービスにおいて、攻撃者に対して堅牢でセキュアな実行環境の提供である。HP-LX のセキュリティの特徴として、抑制(封じ込め)、ファイルシステム保全、システム構成のロックダウン、監査、セキュアな管理モデルなどが挙げられる。

HP-LX のセキュリティ機能の大部分は Linux カーネル 2.4 にセキュリティフッ

クを加えることで実現している。セキュリティフックは動的にロードされるカーネルモジュール（DLKMs）と呼ばれる。

(1) アプリケーションとデータの抑制によるアクセス制御

アプリケーションとデータの抑制は、アプリケーションやサービスが攻撃を受けた時に、攻撃を受けたプロセスから他のアプリケーションやサービスへの攻撃を防止することである。

HP-LX は、アプリケーションとデータの抑制を実現するため、コンパートメント(分類)を表すラベルをすべてのプロセスに付加している。２つのプロセスが同じコンパートメント属性を持つとき、同じコンパートメントにいると言う。また、プロセスに付与されるラベルは、親プロセスのラベルが子プロセスに継承される。

プロセスがファイルにアクセスしたり、他のプロセスまたはネットワークリソースと通信するたびに、カーネルがルールテーブルを参照して、ラベルを強制的にチェックする。同様に、プロセスが他のプロセスやネットワークから、メッセージを受け取るときもチェックされる。システム内でのラベル付けでは、Bell-LaPadula モデルのように、ラベルに階層的な関連づけはされていない。

ルールテーブルは、コミュニケーションアクセス制御とファイルアクセス制御からなる。

(A) コミュニケーションアクセス制御

異なるコンパートメント間でのプロセス間通信を制限するために、コミュニケーション制御テーブル（CCT）を定義する。また、CCT はコンパー

トメントが使用できるネットワークインタフェース指定し、外部との通信も制限できる。同じコンパートメントでのプロセス間通信は、CCT によって制限されない。

以下に CCT の例を示す。

```
HOST * -> COMPARTMENT web PORT 80
            METHOD tcp NETDEV lan_eth0
```

この例では、すべてのホストがネットワークインタフェース lan_eth0 を介し、TCP 接続で 80 番ポートを通してのみ通信できることを定義している。このルールでは、外部からの接続だけを許可し、コンパートメント Web から外部へ接続は許可しない。これにより、Web サーバがたとえ攻撃者の支配下になったとしても、攻撃者はその他のすべてのサービスに接続できないので踏み台に使われることもない。また、ワームに感染したとしても、他のマシンに増殖することもない。

METHOD の指定では、tcp 以外にも共有メモリ、セマフォやメッセージキューなどが使用できる。

(B) ファイルアクセス制御

HP-LX は、ファイルパーミッションに加えて、ファイル制御テーブル（FCT）によってコンパートメント属性がアクセスできるファイルを定義している。FCT の内容は、コンパートメント名、パス、アクセスモードとなっている。アクセスモードには、read、write、append そして none がある。カーネルは FCT に従って強制アクセス制御を行う。

以下に FCT の例を示す。

```
Web /var/www/http-doc read,write
```

上記例では、コンパートメント Web が http-doc ファイルに対して、読み書きが許可される。

(C) Linux chroot

HP-LX は、FCT によるファイルアクセス制御だけでなく、chroot とコンパートメントの封じ込めを組み合わせることで、さらにセキュリティを高めることができる。

chroot とは、任意のディレクトリをルートディレクトリに設定して、サーバプログラムを実行できる機能である。これにより、サーバプログラムは設定されたルートディレクトリの外にあるディレクトリにアクセスすることができなくなる。コンパートメントの封じ込めとは、封じ込められたコンパートメントで UID が root であるプロセスの実行を許可しないことである。また、SUID プログラムの実行も許可しないので、mknod によって chroot を回避することができなくなる。

chroot と FCT を組み合わせることでファイルシステムが重なることなしに、コンパートメントを完全に別々に隔離することが可能となる。chroot により、隔離したファイルシステムのように見せかけることができ、FCT により隔離されたファイルシステム内での細かいアクセス制御を行うことが可能なる。

(D) CGI プログラムの隔離

HP-LX では、Web サーバと CGI を別々のコンパートメントで実行させる

ことでセキュリティを強化している。Web サーバと CGI の通信は、MCGA（Multi-Compartment Gateway Agent）によって、Web サーバと CGI のコンパートメント間での実行要求と応答の通信を許可する。

MCGA は、クライアントーサーバアーキテクチャである。MCGA クライアントが Web サーバのコンパートメントで動き、MCGA サーバが動いているコンパートメントで CGI プログラムが実行される。

(2) ファイルシステムの完全性

HP-LX は、Tripwire を組み込むことで、ファイルシステムの完全性を保証する。Tripwire は、ファイルシステムを監視し、無許可や予期しない変更からファイルの保護と保守を行う。また、監視対象のファイルやディレクトリに変更があった場合は、電子メールで管理者に通知する。

(3) システム構成のロックダウン

新規にインストールされたオペレーティングシステムでは、セキュリティホールとなり得る不用なサービスとデーモン、そしてファイルパーミッションなどの設定が有効になっているので攻撃者に対して脆弱である。

HP-LX は、デフォルトでは脆弱な設定を、最低限のサービス（SSH のみ）のインストールとロックダウンのスクリプトによってセキュリティを強化している。

ロックダウンスクリプトの機能は以下のとおりである。

- コンソールログインの制限

  ルートとシステム管理者以外のユーザのコンソールログインを許可しない。

- リソースの制限

  特定のユーザやグループが利用できるリソース（CPU とディスク容量）の制限を定義する。

- SUID/SGID ビットのセット

  サーバなどの実行ファイルの SUID/GUID ビットを外す。

- ログ

  週単位でログファイルを交代し、古いログファイルを圧縮する。

- パスワードの期限設定

  パスワードの有効期限を９０日に設定する。

- システムタスクの制限

  cron ジョブの使用を制限する。

- 実行可能プログラムの制限

  さまざまな実行可能プログラムの許可を制限する。

(4) 監査機能

HP-LX が収集する監査データは、実行されたシステムコールと、アプリケーションが監査 API を使って生成するアプリケーションイベントデータである。監査機能は、カーネルモジュールとして提供される。

監査システムのアーキテクチャは図 2.6-１のようになっている。

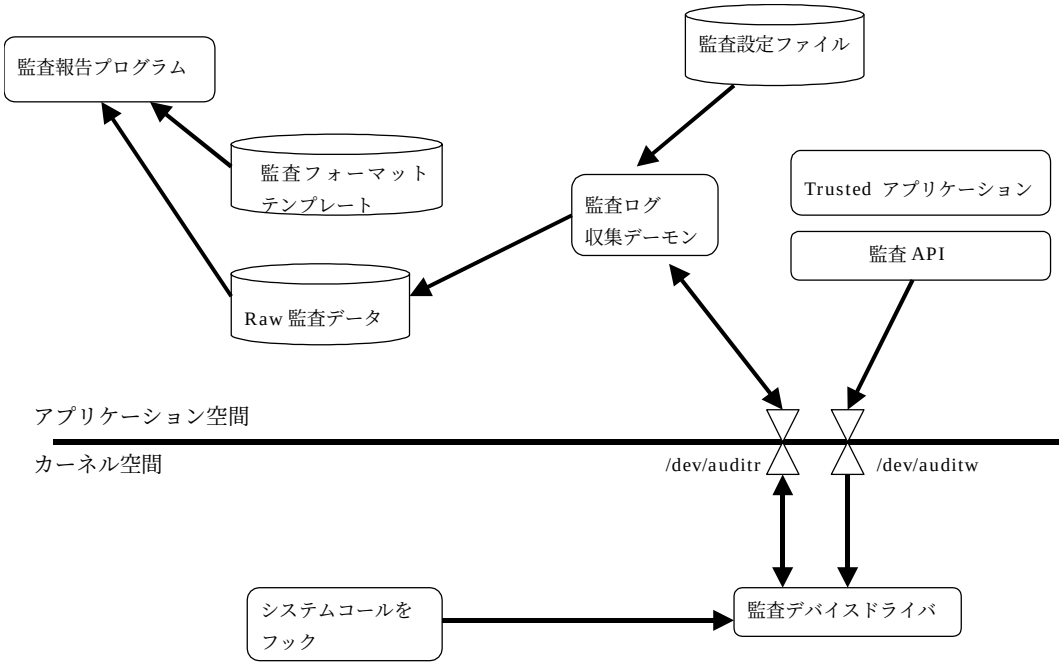

図 2.6-1 監査サブシステムアーキテクチャ

すべての監視データは、いったんカーネルによって収集される。これは、DLKM として実装されている。収集されたデータは、ユーザ空間のデーモンによってディスクなどのストレージに書き込まれる。

監視システムは、Linux Capabiliy とコンパートメントによる隔離でログデータの改ざんを保護している。Linux Capability によって、ログデータに追加するには、AUDIT_WRITE が必要となる。

カーネルが取得する監査データを生成する監査 API を提供している。これは、ユーザプロセスに監査データの書き込み権限を許可する。この API により、信頼されたアプリケーションがカーネルモジュールによって、各種ログデータを出力することが可能となる。

また、HP-LX では、以下の２つの機能によって、攻撃者の行動に対するトレースを容易化する。

● フィルタリング機能

カーネルが取得する監査データを減らすためにフィルタリング機能がある。フィルタリングの設定ファイルは、XML 形式で書かれている。この機能により、どのシステムコールのログを採るかといった設定ができる。

● プレゼンテーションテンプレート

カーネルが取得した監査情報を調査、分析するためにプレゼンテーションテンプレートを提供する。このテンプレートは、XML 形式や、プレーンテキスト形式などでレポート生成を支援する。管理者が、テンプレートを作成できるので、自由にデータ表示を行うことができる。

(5) 特権機能

HP-LX は、３つの特権、root、hplx_admin と set_comp がある。hplx_admin と set_comp はセキュリティ管理者用の特権である。

- root

  UNIX 環境での root 特権と同じであるが hplx_admin と set_comp の特権を持たない。

- hplx_admin

  新たなコンパートメント、CCT や FCT などのセキュリティ設定を許可する。

- set_comp

  現在のコンパートメントを変更できる許可を与える。

プロセスが持つ特権は、子プロセスに継承される。ただし、init はすべての子プロセスに対して特権を継承させず、特定のプロセスに対してのみ継承させる。デフォルトの設定では、init はコンソールログインと SSH デーモンのみ起動させ、この２つにセキュリティ管理者用特権 hplx_admin と set_comp を継承させる。

特権は、ユーザ空間でセットする方法は提供されておらず、クリアまたセットされている特権のみを知ることできる。

(6) セキュア管理モデル

セキュアなリモート管理を行うため、OpenSSH によって提供される SSH（Secure Shell Deamon）による通信路の暗号化と、PAM（Pluggable Authentication Modules）を使ったユーザ認証を行う。

SSH によるリモートログインでは、SSH による認証の後に PAM によってユーザが管理者として承認されているかどうかチェックする。もし、ユーザが承認されていれば、セキュリティ管理者特権が付与される。これにより、PAM によって承認されているユーザだけがリモートでファイルのアクセス権などを変更するユーティリティを実行できる。

コンソールログインの場合は、ユーザ ID に関わらずセキュリティ管理者特権が付与される。

(7) その他の機能

(A) システムのバックアップ／リストア

システムのバックアップとリストアを行うために、オープンソースの AMANDA(Automated Network Disk Archive)を使う場合に、HP-LX では、カスタマイズされた AMANDA クライアントを提供している。AMANDA は、ネッ

トワークホストの大容量テープデバイス用（バックアップサーバ）に最適化されており、バックアップサーバのディスクに複数のファイルを並列に保存する。このディスクに保存されたファイルをテープに保存する。

AMANDA のクライアントは、特別な backup コンパートメントで実行され、AMANDA クライアントとテープサーバとのやり取りは、xinetd デーモンを仲介して行われる。

(B) Apache および Tomcat

HP－LX は、Web サーバ Apache1．3.19、Java Servlet と JSP の実行環境である Tomcat3.2.1 をバンドルしている。これらに対して、抑制とファイルシステム保全の機能を実施することで、セキュアな Web サービスが提供できるようにしている。

2.6.4．セキュリティレベル

hp secure OS software for Linux のセキュリティレベルは、C2 クラスに相当する。

2.7. Openwall

2.7.1. 提供元

提供元は、Openwall プロジェクトである。(http://www.openwall.com/)

Openwall プロジェクトが提供する“Owl(または、Openwall GNU/*/Linux)”は、他のメジャーなディストリビューションと互換性を保ちながらLinuxおよびGNUソフトウェア群のセキュリティ機能を強化するものである。

2.7.2. 状況

2001 年 5 月 11 日に“Owl 0.1-prerelease”がリリースされた。現在は、“Owl 0.1-stable”と“Owl-current”として開発されている。2001 年 11 月 3 日には、Linux 2.2.20 に対応した“Linux 2.2.20-ow1”が公開されている。

Owl は、Linux カーネルばかりでなく、表 2.7-1に示す GNU のパッケージに対してもセキュリティ対策を行い、その差分パッチを公開している。

表 2.7-1 Owl がセキュリティ対策をしているプログラム一覧

SimplePAMApps	SysVinit	acct	autoconf	automake	bash	bc
binutils	bison	bzip2	chkconfig	console-tools	cpio	dev86
dialog	diffstat	diffutils	e2fsprogs	ed	elftoaout	file
fileutils	findutils	flex	gawk	gcc	gdbm	gettext
glibc	gnupg	gpm	grep	groff	gzip	hdparm
iputils	ldconfig	less	lftp	libnet	libnids	libpcap
libtermcap	libtool	lilo	logrotate	ltrace	m4	mailx
make	man	mingetty	mktemp	modutils	mtree	ncurses
net-tools	newt	openssh	openssl	pam	pam_mktemp	pam_passwdqc
pam_userpass	patch	perl	popa3d	postfix	procmail	procps
psmisc	pwdb	readline	rpm	screen	sed	sh-utils
shadow-utils	silo	slang	sparc32	strace	sysklogd	tar
tcp_wrappers	tcsh	termcap	texinfo	textutils	time	traceroute
utempter	util-linux	vim	vixie-cron	which	xinetd	zlib

2.7.3. セキュリティ機能

Openwall プロジェクトから Linux カーネルに対するセキュリティの差分パッチが提供されている。以下にその内容を示す。

2.7.3.1. ユーザスタック内命令実行制限

バッファオーバフローの悪用とは、スタック上に設定された関数の戻りアドレスを同じくスタック上に意図的に書き込まれたコードのアドレスに書き換えることである。もし、スタック領域でプログラムの実行ができなければ、バッファオーバフローの脆弱性を悪用するのが困難になる。また、同じようにバッファオーバフローの悪用として、関数の戻りアドレスを libc ライブラリ内の関数(通常は、system 関数)に書き換える方法がある。

このセキュリティパッチは mmap システムコールで割り当てられた共用ライブラリのデフォルトのアドレスを変更し、各ライブラリ間に隙間ができないように配置する。これにより、関数のパラメタやリターンアドレスのコピー、文字列等のデータを設定することができなくなる。しかし、このパッチはバッファオーバフローの脆弱性を完全に解決する方法ではなく、単に１つのセキュリティ層を提供するだけである。多くのバッファオーバフローの脆弱性の問題は、このパッチで対処できないもっと複雑な処理が残っていると考えられる。例えば、サービス妨害攻撃(DoS Attack)でバッファオーバフローが使用されるが、このパッチでは対処できない。それでも、このパッチを使用する理由は、未知のバッファオーバフローの脆弱性に対する保護をするためである。

重要なのは、たとえこのパッチを使用していても、脆弱性の問題が公開されたらできるだけ早く問題となったプログラムの対策を施すことである。

2.7.3.2. /tmp ディレクトリへのリンク制

限

これは､スティッキービット(ファイルのプロテクションモードの１つ)のモードが設定されているディレクトリへのハードリンクを抑止する機能である。一般ユーザの権限で他ユーザ所有のファイルをハードリンクすることにより攻撃されることを防ぐためである。他ユーザ所有のファイルを“/tmp”のように一般ユーザに書き込み権限のあるディレクトリにハードリンクできるのはシステムの仕様上問題ないのだが、スティッキービットを持つディレクトリに作成されたファイルは所有者が異なるため削除できないという問題がある。

残念ながら、この機能は既存のアプリケーションに大きな影響がある。

2.7.3.3. FIFO ファイルへの書き込み制限

これは、ハードリンク抑止機能に加えデータ改ざん攻撃をしにくくするために、信頼されていない FIFO ファイル(名前付きパイプ)への書き込みを制限する機能である。このオプションを有効にすると、ディレクトリまたは FIFO ファイルのオーナが同じでない限り、スティッキービット付きディレクトリ下に作成された他ユーザ所有の FIFO ファイルへの書き込みが禁止される。

2.7.3.4. /proc へのアクセス制限

当初は、root 権限でしかアクセスできないように/proc 内の特定のディレクトリのパーミッションを変更するだけのパッチであった。このオプションは、一般ユーザが特別なグループに属さない限り、自分自身とアクティブなネットワーク接続のないプロセスのみを参照できるようにするために/proc ディレクトリ下のファイルのパーミッションを制限する機能である。特別なグループの ID は/proc をマウントするときの“gid=”オプションで指定され、デフォルトの ID は“0”である。また、一般ユーザにシステムの情報を見られないように

dmesg(8)コマンドの実行も制限している。

このパッチを使用したとしてもほとんどのプログラム(ps、top、who)は期待どおりに動作するが、root 権限や特別なグループまたは適切な権限がない限り、自分自身のプロセスしか見ることができない。

2.7.3.5. ファイルディスクリプタの特殊処理

ファイルディスクリプタ(fd)の０番(標準入力)、１番(標準出力)、２番(標準エラー出力)は C 言語ライブラリやその他多くのプログラムでは特別な意味を持っており、通常は番号(数値)で参照される。これらの fd がクローズされてもプログラムは正常に実行できるし、これらの fd がクローズされた状態で実行されたプログラムが他のファイルをオープンすると０、１、または２の fd がプログラムに与えられる。このとき、標準入力、標準出力、標準エラー出力に対して入出力処理を行うと、今オープンしたファイルへアクセスすることになる。もし、そのようなプログラムが SUID や SGID ビットを伴ってインストールされていた場合、セキュリティ上の問題に遭遇することになる。

この機能を有効にすると、SUID/SGID ビットが設定されたプログラムの実行時には常に０、１、２の fd がオープンされていることを保証する。もし、fd がクローズされていると“ /dev/null ”がオープンして割り当てられる(chroot のような環境下のようにファイルシステムに/dev がなくてもよい)。

2.7.3.6. execve(2)時のプロセス数制限

Linux は、RLIMIT_NPROC を引数とした setrlimit システムコールで一人のユーザが生成できるプロセスの数を制限することができる。残念ながらこの制限はプロセスを生成する fork システムコール時にのみチェックされる。もし、プロ

セスが UID(ユーザ ID)を変更した場合、この制限を越えてプロセスを生成することができる。UID を変更するには特権が必要なため、本件自身はセキュリティの問題ではないが、プロセスのリソース制限を設定してユーザ環境に切り替わる特権を持ったプログラムが存在し、新しい UID に切り替わる前に fork システムコールが発行されると、RLIMIT_NPROC に対するチェックがされないことになる。

このプロセス数制限機能を有効にすると、execve システムコールで RLIMIT_NPROC が実行されプロセス生成の上限がチェックされる。なぜ、setuid システムコールでこのチェックをしなかったかというと、root 権限で実行しているプログラムの中には、setuid システムコール自身が失敗すること自体を仮定した処理をしていないためである。

2.7.3.7. 未使用共用メモリセグメントの破棄

プロセスのメモリ消費量を setrlimit システムコールで制限することができるが、共用メモリはプロセスが終了しても残すことができ、しかもリソースの制限の対象とされない。

これは、共用メモリが使用していたプロセスの空間から切り離されたり、使用していたプロセスが終了することによって、共用メモリが誰からも参照されていない状態になったとき、そのメモリセグメントを自動的に破棄する機能である。また、共有メモリを作成したがプロセス空間に一度も張り付けられずにそのプロセスが終了しても当該セグメントを破棄する。

この機能により動作しないアプリケーションがあるかもしれない。Apache や PostgreSQL は動作することは確認しているが、特に、商用のデータベースは動

作しなくなるであろう。

なお、この機能は、システムのリソース制限(特に、RLIMIT_AS と RLIMIT_NPROC)を設定しないかぎり、有用ではない。

2.7.4. セキュリティレベル

Openwall のセキュリティレベルは、C1 クラスに相当する。

2.8. Trusted Solaris

2.8.1. 提供元

Trusted Solaris は、Sun Microsystems,Inc.が開発、販売を行っている。

Trusted Solaris は、Sun Microsystems,Inc.が開発したオペレーティングシステム Solaris に対して次の５つの分野、ファイルシステム、システム管理ツール、デスクトップ環境、ネットワークおよびプリントの分野でセキュリティの強化を行っている。

2.8.2. 状況

Trusted Solaris は、日本、米国で以下のようにリリースしている。

- 2000 年５月に、日本で「Trusted Solaris 7 日本語版」をリリース
- 2000 年 11 月に、米国で「Trusted Solaris 8 英語版」をリリース

調査時点では、「Trusted Solaris 8 日本版」のリリース案内は出ていない。しかし、Trusted Solaris 8 は日本語ロケールにローカライズされており利用可能であると米 Sun ホームページに記載されている。

2.8.3. セキュリティ機能

(1) アクセス制御機能

Trusted Solaris では従来からの Solaris に実装されている UNIX アクセス権ビットと ACL による DAC に加え、MLS による MAC を実装し、システムによって自動的に実施される。Trusted Solaris の MAC は、システム内のすべ

てのサブジェクト(能動的な存在であり、通常はプロセスを指す)とオブジェクト(受動的な存在であり、データファイルやディレクトリ、プリンタなどのデバイスを指す)に機密ラベルを付加し、各ユーザには許可上限(clearances)を割り当てることで実現している。機密ラベルとは、ユーザが作業を許可されている機密度や、作業用に選択する機密度を表す。機密レベルは、機密種別とコンパートメント部分からなる。機密種別とは、セキュリティの階層レベルを表すものである。コンパートメントは、共通の情報を必要とするユーザグループである。許可上限は、ユーザに委ねられたセキュリティの程度を示し、機密ラベルと同様に機密種別とコンパートメント部分からなる。ユーザがアクセスできる情報は、この機密ラベルと許可上限によって決まる。

Trusted Solaris では、サブジェクトとオブジェクトの機密ラベルを比較し、どちらの機密ラベルが優位であるかによって、処理の許可、拒否を決定する。ある機密ラベルは、次の 2 つの条件が満たされていれば、もう 1 つの機密ラベルよりも優位だとみなされる。

- 機密ラベルの機密種別が、もう 1 つの機密ラベルの機密種別と同等であるか、それよりも重要度が高いとき。
- ラベルのコンパートメント部分がもう1つのラベルのをすべて含んでいるとき。

オブジェクトの読み取り処理では、サブジェクトの機密ラベルのほうがオブジェクトの機密ラベルよりも優位でなければならない。オブジェクトの書き込み処理は、作成または変更されたオブジェクトの機密ラベルのほうがサブジェクトのラベルよりも優位になる必要がある。これによって、

サブジェクトがオブジェクトの機密ラベルを降格させるようなことはなくなる。

MAC は、ユーザサイトのセキュリティポリシーに応じて実施され、これを書き換えることができるのは、特別な承認または特権をもつユーザのみとなる。

また、Trusted Solaris は、管理用に２つの特別な管理ラベル、ADMIN_HIGH と ADMIN_LOW がある。このラベルは、システムのリソースを保護するために用いられる。通常は管理者しか使用せず、一般ユーザが使用することはできない。

- ADMIN_HIGH

  システム内の最上位ラベルである。管理データベースや監査トレールなどのシステムデータが読み取られないよう保護する。

- ADMIN_LOW

  システム内の最下位ラベルである。MAC により、ユーザは、サブジェクトの機密ラベルよりも低い機密ラベルのファイルにはデータを書き込みことができない。したがって、最下位機密ラベルである ADMIN_LOW をファイルに適用することで、一般ユーザは、そのファイルを読み取ることしかできなくなる。ADMIN_LOW は、通常、すべてのユーザが使用する実行可能ファイルと設定ファイルを保護する目的で使用される。

(2) シングルレベルディレクトリとマルチレベルディレクトリ

Trusted Solaris は、ラベルの異なるファイルを一緒にしてしまうことが

ないように、マルチレベルディレクトリとシングルレベルディレクトリという 2 種類の特別なディレクトリが用意されている。

- マルチレベルディレクトリ(MLD)

  異なる機密ラベルを持つファイルとサブディレクトリを格納できるディレクトリ

- シングルレベルディレクトリ(SLD)

  同じ機密ラベルを持つファイルとサブディレクトリを格納する、MLD 内の隠しサブディレクトリ

MLD により、同じディレクトリに異なるラベルで書き込みを行うアプリケーションを複数実行できる。例えば、/tmp やホームディレクトリは、多数のアプリケーションに利用されるので MLD となっている。

ユーザは、常に MLD において、現在の機密ラベルに相当するファイルだけを表示したりアクセスしたりできる。

(3) RBAC

RBAC によるアクセス制御機能も実装している。RBAC は、役割、承認、実行プロファイルの３つの要素で構成される。役割は、ユーザに特定のアプリケーションに対するユーザアクセス権と、その実行に必要な承認や特権を与えるための特別なユーザアカウントである。RBAC によって、スーパユーザを排除し、スーパユーザの機能を複数の役割に分割している。スーパユーザが一元的に持っていた権限を以下の４つに分割している。

- 制限付き root

  ソフトウェアのインストール

- セキュリティ管理者

  機密ラベルの割り当てとユーザの監視

- システム管理者

  ユーザアカウントのセキュリティ部分以外の設定

●システムオペレータ

システムのバックアップ、プリンタの管理、リムーバブルメディアのマウント

例えば、システム管理者は「Web 管理者」のような新しい役割を作成することもできる。

(4) 特権機能

(A) 特権の細分化（最小特権）

Trusted Solaris には 70 以上の特権が存在し、以下の特権分類がある。

- ファイルシステムセキュリティ

  ユーザ ID、グループ ID に対するファイルシステムの制約、アクセス権、ラベル付け、所有権、ファイルの特権セットを無効にする。

- System V IPC セキュリティ

  メッセージ待ち行列、セマフォセット、共用メモリ領域の制約を無効にする。

- ネットワークセキュリティ

  予約ポートの割り当てやマルチレベルポートへの関連付け、ブロードキャストメッセージの送信、セキュリティ属性（ラベル、メッセージの特権、終端のデフォルト値など）の指定などに関する制約を無効にする。

- プロセスセキュリティ

  監査、ラベル付け、隠しチャネル遅延、所有権、認可上限、ユー

ザ ID、グループ ID などに関する制約を無効にする。

- システムセキュリティ

  監査、ワークステーションのブート、ワークステーションの構成管理、コンソールの出力先変更、デバイス管理、ファイルシステム、複数ディレクトリへのハードリンクの作成、メッセージ待ち行列サイズの拡大、プロセス数の増大、ワークステーションのネットワーク構成、他者の読み込み可能なモジュール、ラベル変換などに関する制約を無効にする。

- ウインドウセキュリティ

  カラーマップ、ウィンドウに対する読み取りと書き込み、入力デバイス、ラベル付け、フォントパス、ウィンドウ間でのデータの移動、X サーバのリソース管理、ダイレクトグラフィックスアクセス (DGA)の X プロトコル拡張機能などに関する制約を無効にする。

(B) 特権獲得方法

プロセスが特権を獲得する方法は、特権を継承する方法と、実行可能ファイルの強制特権を介して獲得する方法の２つある。特権が継承可能であれば、プロセスの起動時に親プロセスから特権が継承される。強制特権は、プロセスの実行に無条件で必要となる特権である。許容特権により、特権の獲得を行えないようすることも可能である。

アプリケーションで特権を使用できるようにするには、次の 2 つ以上の特権セットをアプリケーションに割り当てる。

- 許容セット

  アプリケーションの実行可能ファイルに対応付けられる。許容さ

れた特権セットは、プロセスが特権を使うことができるかどうかを決める一般的な判定要因である。ここから特権を除外すると、どのユーザもアプリケーションでその特権を使うことができなくなる。

- 強制セット

  アプリケーションの実行可能ファイルに対応付けられる。アクセス権のあるユーザがアプリケーションを実行すると、無条件に有効になる。許容セットにない特権を強制することはできない。

- 継承可能セット

  アプリケーションのプロセスに対応付けられる。「継承可能な特権」は、実行プロファイルでアプリケーションに割り当てられた特権と、親プロセスから継承された特権の組み合わせで、プロセスが起動すると有効になる。ただし、アプリケーションの許容された特権セットに含まれていることが前提である。

(C) 特権検出モニタ

あるアプリケーションに必要な特権を判別するために用いられる。このモニタを有効にしてシステムを起動することで、アプリケーションに特権が与えられていないことにより、発生したエラーのみが記録される。実行は特権が与えられている場合と同様に終了する。よって、市販のSolarisアプリケーションや、フリーのアプリケーションをインストールした後に、特権検出モニタを有効にして、アプリケーションを実行することで必要な特権を判別することが可能となる。

(5) ユーザのアクセス制御

Trusted Solaris では、各ユーザに対して実行プロファイルを割り当てることで各ユーザの権限を定義できる。これにより、ユーザに必要最低限の権限のみを与えることが可能となる。

実行プロファイルは、アクション、コマンド、承認、セキュリティ属性をグループ化したものである。

アクションは、CDE(Common Desktop Environment)で起動されるアプリケーションのことである。各アクションには、機密ラベルの範囲（最上位機密ラベルと最下位機密レベルで定義されたレベル範囲)、有効なユーザ ID／グループ ID、指定された継承可能な特権が規定されている。

コマンドは、Trusted Solaris が提供する UNIX コマンドである。コマンドは、アクションと同様に機密ラベルの範囲、有効なユーザ ID／グループ ID、特権属性が規定されている。

承認は、ユーザや役割に与えられる個別の権利であり、プログラムに対する特権と同様に考えることができる。承認には、ユーザのリモートログインの許可や、ファイルの機密ラベルの昇格などがある。承認があれば、ホストのセキュリティポリシーの一部をバイパスする権限をユーザに与えることができる。

セキュリティ属性は、Trusted Solaris 環境にあるセキュリティ関連の要素(ファイル、ディレクトリ、プロセス、デバイス、ネットワークインタフェースなど)の属性である。セキュリティ属性は、適用先によって次のように分類される。

- ユーザアカウントセキュリティ属性

  ユーザアカウントに対して付加されるセキュリティ属性であり、

ユーザのラベル範囲やユーザ識別情報などの情報からなる。

- ファイルセキュリティ属性

  ファイル属性、特権、機密ラベルからなる。特権には、許容された特権と強制された特権からなる。

- プロセスセキュリティ属性

  承認、コマンド、アクションからなる。

実行プロファイルは、管理者が新たに定義することもできるが、いくつかの実行プロファイルがあらかじめ定義されている。例えば、監査ログの読み取り、ファイルのバックアップやファイルの所有権とアクセス権の変更などがある。

(6) 監査機能

Trusted Solaris は、Solaris の監査機能を拡張して実装している。監査可能なシステムアクションはすべて監査イベントとして定義される。監査イベントは、ファイル操作に関するイベント(ファイルオープンや読み込み、書き込みなど)やプロセス操作に関するイベント(子プロセス生成やプロセス終了)ごとにグループ化されており、監査クラスに属するものとして定義される。監査イベントが発生すると、そのイベントに定義された情報(監査トークン)の一部またはすべてが、監査レコードとして監査ログに書き込まれる。監査イベントは、Trusted Solaris に最初からあるもの以外に、Sun 以外のベンダーが提供する Trusted Solaris 用のアプリケーションで必要となる監査イベントを追加できる。また、管理者によって監査クラス、監査イベントを定義できる。

監査機能は、ユーザがログイン時に割り当てられるユニークな監査 ID

(audit ID)をユーザプロセスに結び付けることで行われる。監査 ID は、ユーザが起動するプロセスすべてに継承される。たとえユーザのロールが変わったとしても、監査 ID が継承されるので、あるユーザのすべてのアクションを監査することが可能である。

ネットワーク越しに他の Trusted Solaris の監査ログを取ることができるので、監視専用サーバを設置することも可能である。

(7) API の提供

Trusted Solaris では、プロセスが持つ特権セットなどのセキュリティ機構にアクセスするため、ラベル付きプロセス間通信、アプリケーション監査ログ作成などの API を提供している。これらの API を使うことで、トラステッドアプリケーション作成ができる。

(8) Trusted CDE (Common Desktop Environment)

Trusted Solaris は、共通デスクトップ環境（CDE）に次に示すセキュリティ拡張を加えている。

(A) トラステッドパス

トラステッドパスは、ユーザがキーボードの入力を捕捉するようなプログラムに騙されて、パスワードなどの情報を提供することがないようにするための機構である。画面の底部に沿ったトラステッドスクリーンストライプで、ユーザはトラステッドパスを使用して通信を行っているかを確認できる。

(B) X サーバセキュリティ

Xプロトコルを介してアクセスできるすべてのXリソースに対して細かいアクセス制御を行える。また、セッションを開始するには、Trusted Solarisの CDEでしかできない。

(C) ラベル付きウィンドウおよびワークスペース

すべてのウィンドウとワークスペースには機密ラベルが対応付けされている。機密ラベルをカラーコード化することで機密レベルを視覚的に認識できる。

(D) Trusted ドラッグ＆ドロップ

管理者によって承認されてユーザは、異なる機密ラベルのウィンドウ間でテキストデータやバイナリデータのドラッグ＆ドロップまたは、コピー＆ペイストを行うことができる。

(9) ネットワーク機能

Trusted Solarisは、通信先として許可されるホストおよびネットワークごとにセキュリティ属性を指定する。各ネットワークインタフェースの認可範囲は、最上位機密ラベルによって上限を、最下位機密ラベルによって下限を定義されている。パケットには、機密ラベルなどのセキュリティ属性が付加されている。インタフェースを介して送受信される情報の機密度は、この認可範囲によって制御される。

Trusted Solarisは、複数のトラステッドプロトコルをサポートすることで、Trusted Solaris同士、セキュリティ属性を認識しないOS(Solarisや Windows など)と他の TrustedOS との通信が可能である。また、Trusted Solarisマシンからの通信が、転送データの機密レベルと同じセキュリティ

レベルのゲートウェイを介してのみ配信されるように指定できる。また、許可範囲内のホストのみを通したデータ転送も可能である。

NFS もサポートしており、システムの強制アクセス制御と任意アクセス制御を完全に実施しながら、リモートファイルに透過的にアクセスすることが可能である。

(10) 運用管理ツール

Trusted Solaris は、実行プロファイルおよびユーザや役割アカウント情報を表示・編集するために、GUI ベースの管理ツールや Solaris マネージメントコンソール 2.0 を提供している。これは、Solaris で提供されているマネージメントコンソールに、Trusted Solaris 用の拡張を施したものである。

ネームサーバ上で Solaris マネージメントコンソールのサーバが動作していれば、Solaris マネージメントコンソールのクライアントを使って、どこからでも設定ができる。

(11) ネームサービス

Trusted Solaris は、Solaris 同様に NIS と NIS＋のネームサービスをサポートしている。NIS＋により、実行プロファイル、ネットワークインタフェース、デフォルトのセキュリティ属性などのセキュリティ情報を透過的に配布できる。

(12) デバイスのラベル範囲

Trusted Solaris は、デバイスに対しても機密ラベルの範囲を設定することで、デバイスを利用できるユーザを制限できる。例えば、印刷に関しては、任意のプリンタを特定の機密範囲を持つファイルの印刷だけに制限できる。

2.8.4. セキュリティレベル

Trusted Solaris のセキュリティレベルは、B1 クラスに相当する（EAL4 に認定)。

2.9. CBOSS/LOMAC

2.9.1. 提供元

Community-Based Open Source Security（以下、CBOSS）は、オープンソースにおける先駆者や、オペレーティングシステムのセキュリティにおける専門家らによる国際的な共同研究体である。CBOSS の第一の目的は、FreeBSD に焦点を合わせ、オープンソフトウェアのセキュリティを改善していくことである。

2.9.2. 状況

2001年7月2日にスタートしたCBOSSプロジェクトは､プロジェクトから18ヶ月という期限を設け、この期間内にオープンソースシステムの実際のセキュリティレベルをある程度まで改善することを目的とし、オープンソースの重要な要素とコンピュータセキュリティのコミュニティを纏め上げようと努めている。プロジェクト発足から 18 ヶ月以内とは、具体的には 2002 年 12 月までの期間を指す。CBOSS プロジェクトでは、オープンソースシステムにおける目先のセキュリティ向上と、抜本的なセキュリティ向上を目的とした長期間にわたる調査の双方をそのターゲットとしている。

2.9.3. セキュリティ機能

現在 CBOSS プロジェクトでは、主要ないくつかの問題を処理するべく以下にあげる 4 つのコミュニティベースのイニシアティブを定義している。

- セキュリティ情報の明確化
- セキュリティ技術の明確化
- Poligraph の開発
- PRIVMAN の開発

このように CBOSS プロジェクトでは、セキュリティ技術の開発とは直接関係のないものも含めてさまざまな取り組みが行われている。ここでは、CBOSS プロジェクトにて行われているセキュリティ技術開発だけではなく、それ以外の取り組み関しての情報も記載する。

2.9.3.1. セキュリティ情報の明確化

FreeBSD プロジェクトでは、オペレーティングシステムの主要なコンポーネントに関する大量のドキュメントが存在するが、同等に重要であるシステム全体のセキュリティアーキテクチャに関しては一切ドキュメント化されていない。そこでこのイニシアティブでは、FreeBSD における現状のセキュリティアーキテクチャのドキュメント化、および、マニュアルページに対するセキュリティ情報の記載をその目的としている。

(1) セキュリティアーキテクチャのドキュメント化

本イニシアティブの 1 番目の目的は、FreeBSD の開発者向けのセキュリティアーキテクチャに関するドキュメントを作成することである。このドキュメントの目的は、システムのセキュリティアーキテクチャに関する明確で詳細な資料を提供することであり、加えて、これらのセキュリティアー

キテクチャ情報をシステムの発展に追従してアップデートし続けるための効率的な手段を確立することにある。このドキュメントによってプログラマに基本的なセキュリティ情報を提供できるとしている。

(2) マニュアルページへのセキュリティ情報の記載

本イニシアティブの 2 番目の目的は、FreeBSD のマニュアルページ（man ページ）にセキュリティに関する情報を追加することである。

現在マニュアルページとして提供されている、FreeBSD の開発者向けドキュメントは、おもなユーティリティ、ライブラリ、システムコールおよびカーネルインタフェースといった非常に広範囲に渡ったものであるが、システムレベルでのセキュリティアーキテクチャに関する注意や、個々のサービスやインタフェースにおけるセキュリティ関連の詳細な情報といった、セキュリティ分野に関する記述は乏しい。そこで、セキュリティアーキテクチャに関する詳細な資料を作成するのと同様に、マニュアルページに対してもセキュリティ情報を反映し、頻繁に起こしやすいセキュリティ上の誤りや、セキュアなコーディングを行うためのテクニック、そして効果的にセキュリティレビューを行うためのプロセスに関するアドバイス等を記載することによって、プログラマがセキュリティ情報を入手しやすくし、その結果としてソフトウェア開発時およびメンテナンス時の作業負担を軽減することを目的としている。

2.9.3.2. セキュリティ技術の明確化

有益なセキュリティ技術が数多く存在しているにも関わらず、その多くがあまり知られていないために、主要なソフトウェアプロジェクトでそれらのセ

キュリティ技術が利用されていないという現状を踏まえて本イニシアティブでは、これらのセキュリティ技術を完成させて移植し、そしてオープンソースシステムでもっと利用されることを目標として広めていくことを目的としている。現在は以下にあげる技術をターゲットとしている。

- Enhanced File System Extend Attributes
- LOMAC/FreeBSD
- Network Stack Hardening
- PAM Authentication
- Cryptographic Protection of Swap File Systems
- SELinux/MLS

(1) Enhanced File System Extend Attributes

FreeBSD のアクセス制御メカニズムは任意アクセス制御を提供するものであり、「Multi-Level Security (MLS)」、「Type-Enforcement (TE)」あるいは「Biba Potection Models」といった強制アクセス制御は実装されていない。FreeBSD において現在利用可能なセキュリティポリシーの取り込み作業は進行中であるが、これらのポリシーの多くは、ファイルシステム上のオブジェクトに対して執拗なまでのセキュリティラベルのラベル付けを必要とする。FreeBSD 5.0-RELEASE では、TrustedBSD プロジェクトによる成果物をサポートする形で拡張属性が導入されたが、現在の実装ではパフォーマンスが悪く、手動でのコンフィギュレーションを必要とする。拡

張属性を広く利用してもらうためには、この拡張属性をファイルシステムにおける必須な部分として、再度実装しなおす必要がある。これにより、高いパフォーマンスを得られるようになる。新しい実装では、管理上の負担も軽減したものになる予定である。

(2) LOMAC/FreeBSD

CBOSS プロジェクトでは、FreeBSD の開発者と協力して NAI Labs のソフトウェアである LOMAC を FreeBSD に移植し、FreeBSD のディストリビューションへの統合を予定している。また、この移植作業という機会を利用して LOMAC 自体の見直しも行われる予定であり、これにより LOMAC アーキテクチャにおけるカーネル依存部分と非依存部分の分離を強化するとしている。カーネル依存部分と非依存部分の隔離性を強めることにより、NetBSD や OpenBSD をはじめ、Solaris のような商用 UNIX への移植性を高めることを狙っている。最終的には、後述する「Poligpraph」で LOMAC をサポートするための調査も行われる予定となっている。

LOMAC の各種プラットフォームへの移植のステイタスは以下のようになっている。

- LOMAC/FreeBSD

  FreeBSD 5.0 対応のバージョンが提供されている。

  試験的な利用においては充分安定している。

  統合作業および開発作業は進行中である。

- LOMAC/Linux 2.2 kernel

  バージョン 0.3 以降のリリースから含まれている。

- LOMAC/Linux 2.4 kernel

  2001 年 6 月に開始されている。

- LOMAC/Linux Security Modules(LSM)をサポートする Linux

  2001 年 6 月に開始されている。

- LOMAC/RSBAC をサポートする Linux

  2001 年 6 月に開始されている。

LOMAC の詳細に関しては後述する。

(3) ネットワークプロトコルスタックの強化

BSD に実装されているネットワークプロトコルスタックの開発以来、DoS アタックのような新しいタイプの攻撃が増加しており、このようなネットワーク攻撃は、比較的単純な、そしてスクリプトで作成できてしまうようなレベルの攻撃に対してすら、ネットワークの脆弱性を高めてしまうものである。このような状況の中で、FreeBSD でも時間をかけてそのネットワークプロトコルスタックのセキュリティ強化が行われてきたが、残念ながらすべてが万事上手く実装されてきたわけではないらしい。特にこの点に該当するのが、TCP SYN Cookies と TCP Endpoint Congestion Notification(ECN)の実装部分で、この部分が強化される予定である。また、FreeBSD では IP Firewall module(IPFW)の実装が遅れていたが、この点の実装も推し進めるとのこと。

(4) PAM（Pluggable Authentication Modules）サポートの徹底

本イニシアティブは、FreeBSD において認証を必要とするアプリケーションは PAM を利用するべきであるという考えを持っている。PAM を利用しないアプリケーションの存在は、より強力な認証機構の開発の妨げになるだけでなく、システムの一貫した管理を困難なものとし、新たなセキュリティ機構の実装の妨げにもなるというのがその理由である。残念ながら現状の FreeBSD では、認証を必要とするすべてのアプリケーションが PAM を利用していないため、これらのアプリケーションの PAM 対応作業を推し進めている。なおこの作業は、より強力な認証機構の開発を促し、結果として FreeBSD のセキュリティの改善を推し進めるものとして、非常に高い優先度が与えられている。

(5) スワップおよびファイルシステムの暗号化による保護

モバイルコンピュータの利用率の向上に伴い、物理的なストレージメディアが盗難に遭う可能性が高くなっている。一般的なシステムの場合、仮想メモリスワップサービスにより仮想メモリ内のデータとしてストレージメディアに書き込まれるデータと、実際にファイルシステムを経由してストレージメディアに書き込まれるデータの 2 つが存在し、これらのデータが暗号鍵等の非常に重要なデータを含んでいる可能性は大いに有り得ることであるとして、データの盗難に対応できる機構の実装が試みられている。こういった機構の実装形態として既存の技術は、高い有用性が有りながらもその開発は非常に手間の掛かるものであり、早急に開発が行える別

の手段として、デバイス間でのデータ転送層にて保護機構を組み込むことで、デバイス層レベルで保護する方法が試みられているようである。

(6) SELinux/MLS

本イニシアティブでは、SELinux の拡張作業を予定している。SELinux はその構造上、BLP ポリシーモデル[23]や Biba ポリシーモデル[24]をサポートすることが可能であるが、現状でこれらはサポートされていない。BLP ポリシーモデルは実装されてはいるが利用できるようには構成されておらず、Biba ポリシーモデルにいたっては全く実装されていないというのが現状である。結果として、TCSEC-style MAC[25]、LSPP-style MAC[26]、POSIX.1e-style MAC[27]といった既存のセキュリティ技術と互換性のあるポリシーあるいはアプリケーションを提供しておらず、SELinux を受け入れ難いものとして位置付けてしまう可能性がある。この状況を改善すべく、以下の作業が予定されている。

---

[23] D. E. Bell and L. J. La Padula, "Secure Computer Systems: Mathematical Foundations and Model", The MITRE Corporation, report # M74-244, may, 1973.
[24] K. J. Biba, "Integrity Considerations for Secure Computer Systems," Electronic Systems Division, Hanscom Air Force Base, Bedford, MA, April 1977.
[25] US Department of Defense, "Trusted Computer Systems Evaluation Criteria," Standard, DOD 5200.28-STD, December, 1985.
[26] National Security Agency, "Labeled Security Protection Profile Version 1.b", Protection Profile, http://www.radium.ncsc.mil/tpep/library/protection_profiles, October 1999.
[27] Institute of Electrical and Electronics Engineers, "Draft Standard for Information Technology --- Portable Operating System Interface (POSIX) --- Part 1: System Application Program Interface (API): Protection, Audit, and Control Interfaces [C Language]", PSSG/D17, October, 1997.

- SELinux のセキュリティサーバに対する、BLP ポリシーモデルと Biba ポリシーモデルの完全な実装
- SELinux 上のアプリケーションのための POSIX.1e-style MAC ライブラリの実装

2.9.3.3. Poligraph

Poligraph とは、モジュール形式で FreeBSD のカーネルを拡張し、新たなアクセスコントロールを導入するための仕組みであり、アクセスコントロール導入の際の一貫したメカニズムを提供するものである。Poligraph は、さまざまなアクセスコントロールモデル間での競合や、サブジェクトあるいはオブジェクトにラベリングする際の依存関係といった点で、セキュリティモジュール間の相互関係を管理しようというものである。

2.9.3.4. PRIVMAN

本イニシアティブの開発する予定である PRIVMAN は、特権管理のためのツールキットである。このツールキットの目的は、ユーザ空間にて稼動するソフトウェアの開発において、むやみにルートの権限を利用して稼動するソフトウェアが作成されてしまうのを防ぐことにある。特にネットワークサーバにおいて特権の濫用というものは、深刻な被害をもたらすような攻撃の対象となるものである。主要な目標事項として、以下の 3 つが挙げられている。

- 短期間での実装

  14 ヶ月以内（具体的には 2002 年 9 月迄）に実用可能なレベルの特権管理

ツールキットの実装が予定されている。この際、最大限の効果をもたらすことができるように、過去にその脆弱性を攻撃されたことのあるアプリケーション、つまりWebサーバ、FTPサーバ等に焦点をおいて作業が進められる予定である。

- オープンソース開発における慣習に従った特権管理技術を実装

  PRIVMANはオープンソースとして公開される予定となっており、オープンソースのコミュニティでお馴染みな、既存のUNIXファシリティにおいて利用しやすいものとされる予定である。

- 互換性およびパフォーマンスを維持

  オープンソースコミュニティにて最大限受け入れられることを念頭に、特権管理ツールキットを利用して再構築したソフトウェアのオープンソースカーネルに対する互換性は100%確保するとしている。このツールキットはカーネルのローダブルモジュールの形式で実装される予定であり、選択的に利用できる形を採るとしている。また、PRIVMANにて実装される技術は、本質的にはパフォーマンスの低下をもたらさないものであるとしている。

また、アプリケーションを再構成し直し、ルート権限を利用することなく稼動できるようにするとしている。ただしこのアプリケーションの再構成は、既存の機構を利用して実現できるものではなく、以下の 3 つを開発する予定であるとしている。

- アプリケーションに代わって、そのアクセスを特定の操作へ取り次ぐ特権サーバ
- アプリケーションの再構成を容易にするサポートライブラリ
- 特権管理ツールキットをサポートするアプリケーションの構築あるいは既存アプリケーションの再構築の際の構造的な基本方針

なおこの特権管理ツールキットの有効性を証明するために、手始めに「Washington University FTP (WU-FTP)」に対してこの技術を適応することが予定されている。この WU-FTP は、ファイル転送プログラムとして非常に多くのサイトで利用されており、それ故に、このプログラムのほんの些細なエラーですら非常に深刻な被害をもたらす可能性を秘めているというのがその理由である。実際、多くのアタッカーがこの WU-FTP を利用してルートアクセスを得ようとしているという。

2.9.3.5. LOMAC

ここでは、CBOSS において FreeBSD に移植される予定の LOMAC に関して記載する。

(1) 概要

LOMAC はフリーUNIX のカーネルをターゲットとして開発されたローダブルカーネルモジュールで、「Low Water-Mark Access Control[28]」と呼ばれるアクセス制御モデルをベースに実装されており、プロセスとデータの integrity を保護する機能を提供する機能拡張モジュールである。LOMAC と

[28] K. J. Biba, "Integrity Considerations for Secure Computer Systems," Electronic

いう名称は、採用されているアクセス制御モデルの名称である「Low Water-Mark」および「Access Control」の頭文字から名付けられており、開発元は Networks Associates, inc.[29]のセキュリティ調査部門 NAI Labs である。

LOMAC は、「より単純な強制アクセス制御機構」を念頭に開発されている。現在に至るまでさまざまな強制アクセス制御技術が開発されており、かつそれなりに成功を収めているにも関わらず、メインストリームの UNIX カーネルに取り込まれることがなかった理由はその使い難さにあるとの考えから、LOMAC は以下の点を重視して開発されているものである。

- 標準のカーネルに適用可能
- 既存のアプリケーションと互換性がある
- サイトごとに設定を必要とすることがない
- ユーザになるべく意識させない

LOMAC はカーネルの持つ既存の保護メカニズムと機能的に重複する部分は存在しない。既存のメカニズムに加えて LOAMC によるパーミッションチェックが行われるのであり、リソースへのアクセス許可を得るためには、既存のメカニズムと LOMAC の双方でアクセスが許可されなくてはならない。LOMAC によるチェックはユーザの身分を基準とした判定はではないため、この部分の判定は既存のメカニズムを利用する必要がある。

Systems Division, Hanscom Air Force Base, Bedford, MA, April 1977.
[29] http://www.nai.com/

(2) Low Water-Mark Access Control

プロセスやデータの integrity 保護機能を提供するアクセス制御モデルは大きく以下の 2 つのタイプに分類される。

- 特権が変更されることのないタイプ
- 特権が変更されることのあるタイプ

「特権が変更されることのないタイプ」とは、サブジェクトに対して割り当てられた特権の集合が、1 度割り当てられると変更できないタイプのアクセス制御モデルの分類を指す。「特権が変更されることのあるタイプ」とは、サブジェクトが何かしらのオペレーションを実行する際に新たな特権集合を割り当てることが可能なアクセス制御モデルを指す。特権が変更されることのあるタイプに該当するアクセス制御モデルは、さらに以下の 3 つのタイプに分類が可能である。これは、サブジェクトの基本的な 3 つのオペレーションに従った分類である。

- 実行時に特権が変更されるタイプ
- 参照時に特権が変更されるタイプ
- 更新時に特権が変更されるタイプ

既存のシステム環境に対して、integrity の保護機能を提供することが可能なアクセス制御モデルは数多く存在するが、既存のシステム環境に適応させる際に最適なアクセス制御モデルは「更新時に特権が変更されるタイプ」であると結論付けている。LOMAC で採用されている Low Water-Mark

Access Control は、このタイプに分類されるアクセス制御モデルである。

(3) 強制アクセス制御機能

(A) integrity-level

LOMAC では、システムを 2 つの integrity-level に分割することにより完全性を保証する機能を実装している。この 2 つのレベルは各々「high レベル」と「low レベル」と呼ばれ、high レベルには init プロセス、カーネルデーモン、システムバイナリ、ライブラリやコンフィギュレーションファイルといったシステムコンポーネントが割り当てられる。対して、low レベルには high レベルに属さない残りのコンポーネントが割り当てられる。これらは、ネットワークにアクセスするクライアント、サーバプロセス、ローカルのユーザプロセスやそれらが利用するファイルが該当する。LOMAC がディレクトリやファイルに対して integrity-level をいったん割り当てると、その後そのディレクトリあるいはファイルに割り当てられた integrity-level を変更することはない。ただしプロセスの場合は例外で、high レベルに割り当てられたプロセスを low レベルへ降格させる場合がある。しかし、プロセスの場合であっても integrity-level を昇格させることは決してない。

(B) 完全性保護のメカニズム

LOMAC では 2 つのメカニズムによって完全性を保証する機能を提供している。

high レベルの保護

LOMAC では、low レベルに割り当てられたプロセスによる high レベル

に割り当てられているファイルに対する修正や、high レベルに割り当てられたプロセスに対するシグナルの送信を許可していない。

integrity-level の再割り当て

LOMAC では、low レベルに割り当てられたファイルから、high レベルに割り当てられたファイルへデータが移動しないことも保証している。プロセスは、low レベルに割り当てられたファイルから内容を読み取り、high レベルに割り当てられたファイルへ書き込むといった操作によってデータを移動することが可能である。しかし、LOMAC ではこのようなデータの移動も許可しない。これは、high レベルに割り当てられたプロセスが low レベルに割り当てられたファイルを読み込もうとした時点で、そのプロセスを low レベルに降格させることによって実現している。いったん low レベルに降格したプロセスは、決して high レベルへ昇格されることはないので high レベルへ割り当てられたファイルの修正ができなくなる。

(c) PLM

ディレクトリやファイルに対する integrity-level の割り当ては、「Path Level Map(PLM)」と呼ばれるモジュールで保持されている。この PLM はディレクトリあるいはファイルの絶対パスと integrity-level が対となったマッピング情報を保持するテーブルであり、カーネルがあるファイルを開く際には、同ファイルに割り当てられた integrity-level を PLM より検索し、メモリ中のデータ構造にラベル付けする。

PLM は、integrity-level、フラグ、絶対パス等の 25 個の項目から構成されるレコードの集合であり、各レコードはその絶対パスの長い順に保

持されている。PLM の簡単なイメージを表 2.9-1に示す。

表 2.9-1 3 つの PLM 規則

一貫性レベル	フラグ	絶対パス
high		/home/httpd
low	child-of	/home
high		/

表 2.9-1のような PLM が定義されている場合、”/home/httpd/html”の integrity-level は high となる。フラグに”child-of”が指定されている場合、そのレコードに設定されている絶対パスが示すディレクトリ以下のすべてのディレクトリおよびファイルに対して、そのレコードで設定されている integrity-level がデフォルトで適用される。表 2.9-1の場合、”/home”ディレクトリには low レベルが割り当てられており、かつそのフラグに”child-of”が指定されているので、”/home”に含まれるすべてのディレクトリおよびファイルの integrity-level は low となる。ただし、フラグに”child-of”が指定されている場合、そのレコードで設定されているディレクトリあるいはファイルに対して、そのレコードで設定されている integrity-level は適応されない。PLM のレコードリストの探索時に、ターゲットであるパスと正確に一致したパスを有するレコードが見つかると、まずはフラグがチェックされ、”child-of”フラグが指定されている場合には、このレコードは無視され、レコードリストの探索が継続される。従って表 2.9-1では、”/home/httpd”の integrity-level は high であり、”/home/tfraser”は low、”/home”は high であることになる。

PLM には 2 つの欠点が存在する。1 つは、ディレクトリおよびファイルの絶対パスを必要とする点であり、探索等に時間が掛かってしまうとい

う問題がある。2 つめは、PLM 探索時に LOMAC が矛盾した integrity-level を取得してしまう可能性があるという点である。これは、マルチハードリンクされて名前付けされたファイルに関して integrity-level 探索が行われた場合である。マルチハードリンクによる異なる名前に対して異なる integrity-level が割り当てられている場合、誤った integrity-level を結果として返す可能性が生じる。そのため LOMAC は、その稼動中に上記のようなハードリンクが作成されないように監視している。LOMAC を正確に稼動させるためには、システム起動時等の LOMAC が稼動を開始する前にハードリンクが作成されることがないように管理者が管理しなくてはならないという欠点でもある。この問題は、シンボリックリンクには当てはまらない問題である。

2.9.4. セキュリティレベル

LOMAC のセキュリティレベルは、C2 クラスに相当する。

2.10. Linux Intrusion Detection System (LIDS)

2.10.1. 提供元

提供元は、LIDS プロジェクトである。(http://www.lids.org)

LIDS プロジェクトは、Xie Huagang と Phillippe Biondi が中心となって開発を行い、1999 年 10 月に最初の公開が行われた。

LIDS は、Linux カーネルにセキュリティ強化を施すものであり、Linux カーネルにはないセキュリティ機能を実装している。実装されているセキュリティ機能には、ファイル／プロセス保護機能の追加、ACL による MAC の実装、Linux Capability および独自 Capability の実装、ポートスキャン検知、カーネルからのセキュリティ警告などがある。

LIDS は、カーネルパッチとシステムツールからなり、GPL で配布されている。

2.10.2. 状況

(1) 最新のリリース状況

LIDS は、LIDS 0.X を Linux カーネル 2.2.X 向け、LIDS 1.X を Linux カーネル 2.4.X 向け、そして LIDS2.X を Linux カーネル 2.5.X 向けとしてリリースしている。最新版はそれぞれ以下のとおりである。

(A) Linux カーネル 2.2 系

- 安定版　カーネル 2.2.20 に対応している LIDS 0.11.0
- 開発版　カーネル 2.2.20 に対応している 0.11.0pre5

(B) Linux カーネル 2.4 系

- 安定版　まだリリースされていない
- 開発版　カーネル 2.4.16 に対応している LIDS 1.1.1pre4

(C) Linux カーネル 2.5 系

- 安定版　まだリリースされていない
- 開発版　カーネル 2.5.2 に対応している 2.0pre1

また、Openwall プロジェクトから提供されているパッチを組み合わせた LIDS も提供されている。

(2) セキュリティホールに関する情報

2002年1月9日に、発見されたLIDSのセキュリティホールがアナウンスされ、修正したバージョンがリリースされた。発見されたセキュリティホールは以下である。

- 攻撃者が環境変数 LD_PRELOAD を使用して、プログラムに与えられた特権を本来は与えられることがないプロセスに与えること（leak）が可能となる。LIDS によって、特別な特権を与えらたすべてのプログラムは、環境変数 LD_*が使用できないように変更された。
- カーネル封印前に、ロードし実行されたプログラムが全部の Capability をカーネル封印後も持ち続けてしまう。プロセスが持つ必要のない Capability を削除するように変更された。
- ACL を継承するプログラムが環境変数 PATH や ALIAS を使うことで悪意あるプログラムにリダイレクトされてしまう。また、悪意あるプログラムに ACL が継承されてしまう。保護されたプログラムにしか ACL が継承されないように変更された。

2002 年 1 月 9 日以降にリリースされた LIDS では、上記のセキュリティホールが修正されている。

2.10.3. セキュリティ機能

(1) アクセス制御

(A) アクセス制御リスト

LIDS では、従来のファイルパーミッションによる DAC に加えて、ファイルアクセス制御リストと Capability アクセス制御リストの２種類のアクセス制御リストによる MAC を実装している。この２つのアクセス制御によって、ファイル保護とプロセス保護を行っている。

- ファイルアクセス制御リスト

  ファイルアクセス制御リストでは、ファイルやディレクトリなどのオブジェクトに対して、読み込みしかできないファイルまたは追加しかできないファイルといったパーミッションを設定する。また、実行プログラムなどのサブジェクトがアクセスできるオブジェクトへのパーミッションを設定する。

  LIDS のファイルアクセス制御リストで、設定できるパーミッションには以下の５種類がある。

READONLY　　アクセス権を読み込み専用に設定

APPEND　　アクセス権を追加専用(読み込みも含む)に設定

WRITE　　アクセス権を読み書き可能に設定

DENY　　指定したオブジェクトへのアクセスを拒否

IGNORE　　オブジェクトに設定されている Capability を無視

以下にファイルアクセス制御リストの設定例を示す。

```
lidsconf –A –o /etc –j READONLY
lidsconf –A – o /etc/shadow -DENY
lidsconf –A – s /usr/sbin/sshd –o /etc/shadow –j READONLY
```

１番目の例では、/etc ディレクトリ以下は読み込みしか許可しないように設定している。２、３番目の例では、shadow ファイルへのアクセスをすべて許可しないように設定してから、サブジェクト sshd にだけ読み込み権限を許可している。

オブジェクトがディレクトリの場合は、再帰的にその下の同じファイルシステム上にあるものに対しても、指定されたパーミッションが適用される。ただし、指定したディレクトリの中に、そのディレクトリのファイルシステムとは異なるファイルシステムがマウントされている場合、異なるファイルシステムのオブジェクトに対しては適用されない。

パーミッションを DENY にすると、ファイルを表示する ls コマンドなどのサブジェクトに READONLY などの権限が与えられていない限り、

たとえ root であってもファイルの存在を確認することができない。また、サブジェクトがアクセスできるオブジェクトを実行ドメインとして、ファイルアクセス制御リストに設定すると、明示的に許可を与えたオブジェクト以外のアクセスはすべて拒否される。例えば以下の例のように設定すると、httpd は、/etc/httpd、/home/httpd 以外のファイルアクセス許可がなくなる。

```
lidsadm - A - s /usr/sbin/httpd - d /etc/httpd - j READONLY
lidsadm - A - s /usr/sbin/httpd - d /home/httpd - j READONLY
```

- Capability アクセス制御リスト

  Capability アクセス制御リストによって、実行プログラムや実行スクリプトなどのサブジェクトに必要となる Capability を与える。これによって、プロセスに必要最小限の権限だけ与えることができる。LIDS には、31 の Capability が存在する。以下にそのうちのいくつかをあげる。

  CAP_SYS_RAWIO

  /dev/hda などのデバイスに raw アクセスを許可する。

  LIDS では、raw デバイスファイルにアクセス可能ならば、ファイル ACL をすり抜けて、ファイルの変更ができてしまう。CAP_SYS_RAWIO を不許可にすることで raw デバイス経由でのファイルアクセスを防ぐことができる。

  CAP_BIND_NET_SERVICE

  特権ポートにバインドすることを許可する。

特権ポートを使うプログラムは、ポートにバインドするためにこの特権を必要とする。この特権をシステム全体で無効し、プログラムがバインドできるポート番号を個別に与えることで、不必要な特権ポートを利用不可にできる。

CAP_HIDDEN

ps コマンドなどのプロセス情報を表示するプログラムから、プロセスを隠すために使う。

CAP_HIDDEN は LIDS 独自の権限である。これを許可すると、ps コマンドや/proc ディレクトリにプロセス情報が表示されなくなり、プロセスを隠すことができる。

CAP_INIT_KILL

init が立ち上げたプロセスにシグナルを送るのを制限する。

CAP_INIT_KILL も LIDS 独自の権限である。これを不許可にすると、init が立ち上げたプロセスは誰であっても kill できなくなる。これによって、重要なプロセスが攻撃者などによって、終了させられることを防ぐことができる。

サブジェクトに Capability を与えるには、GRANT 属性を使う。以下に例を示す。

```
lidsconf -A  /usr/sbin/httpd - o CAP_INIT_KILL - j GRANT
lidsconf -A /usr/sbin/httpd - o CAP_BIND_NET_SERVICE 80-80 - j GRANT
```

２番目の例では、Web サーバが 80 番ポートにだけバインドできる許

可を与えている。

- ACL の時間制限

  LIDS は、ACL の記載をある特定の時間だけ有効にすることができる。例えば以下のようにすると、ユーザのログインを午前８時から、午後５時まで許可するといったことができる。

```
lidconf - A - s /bin/login - o /etc/shadow - t 0800-1700 - j READONLY
```

ファイルアクセス制御リストは、ブート時から有効になるのに対して、Capability アクセス制御リストはカーネル封印後から有効になる。

(B) Capability の設定

LIDS は、２つの異なった方法で Capability を無効／有効にできる。１つは先に説明した、Capability 制御リストによって個々のサブジェクトに与える方法である。もう１つは、Linux カーネルに対して、システム全体で無効／有効な Capability を与える方法がある。これは、uid/euid が０（root）であるプログラムに対して、Capability を制限するものとなる。すなわち、Linux カーネルに対して無効になっている Capability は、Capability 制御リストで有効になっていない限り、root であっても絶対に使用できない。

(C) アクセス制御リストの継承

デフォルトでは、子プロセスは親プロセスのアクセス制御リストを継承しない。親プロセスが子プロセスに対して、ファイル、Capability アクセス制御リストを継承させるには、個々のサブジェクトに対して継承

レベルを指定しなければならない。継承レベルとは、アクセス制限リストが継承される世代数である。例えば、継承レベルが１に指定されていれば、すべての子プロセスはアクセス制御リストを継承する。しかし、孫プロセスには継承されない。

(D) アクセス制御の実装

LIDS のファイル、ディレクトリ保護機能は、VFS レイヤの最上部で動作している。VFS の inode でファイルを識別しているので、Linux がサポートするあらゆるファイルシステムで動作することができる。各ファイルは、パス名ではなく、ファイルシステムと inode を使って、ファイルと ACL を対応させている。もし、ファイルの inode が変わってしまうと、そのファイルと ACL の対応が間違ったものになってしまう。例えば、ユーザパスワードの変更で shadow ファイルを削除して再生成したときや、ソフトウェアのインストールのときなど、inode が変わるとファイルと ACL の対応を更新する必要がある。

また、LIDS ではシステムコールをインターセプトし、システム全体で有効な Capability であるか、そしてプロセスに許可された Capability かを判定して、システムコールの許可／不許可を決定する。

(2) カーネル封印

カーネル封印とは、Linux カーネルに動的にモジュールを追加、または削除をできなくすることである。カーネルを封印してしまうと、root であってもモジュールを追加、または削除ができなくなる。

通常の Linux カーネルは動的にカーネルモジュールを追加、削除することができる。また、カーネルモジュールはカーネル空間で実行されており、

カーネルモジュールが何を実行しているかは知ることができないので、カーネルモジュールがクラッカーに悪用される恐れがある。LIDS は、起動時に必要なカーネルモジュールだけをカーネル空間に読み込んでからカーネル封印を行っている。

(3) LFS（LIDS Free Session）

LFS とは、LIDS のアクセス制御が及ばない端末のことである。これを使うことによって、システムのポリシー変更などのメンテナンスを容易にすることができる。LFS は、ローカルの端末から起動しなければならないので、リモートから LFS を奪取されることはない。

(4) ポートスキャン検知

LIDS は、ポートスキャン検知機能をカーネル内に最初から組み込んでいる。ポートスキャン検知機能は、ポートスキャンがあったときに syslogd を使って、攻撃元の IP アドレスを記録するだけである。

(5) 監査機能

LIDS は、あるユーザが権限を与えられてないファイルにアクセスしたり、また許可されていない権限を使ってアクセスすると、アクセス内容を LIDS によって保護されたログファイルに記録する。また、アクセス拒否されたユーザのユーザセッションを一度、閉じることもできる。

(6) 通報機能

LIDS は、システム内のアクセス違反の記録や、ポートスキャンを検知した情報をメールで送ることができる。ただし、記録の内容や、検知した内容に応じてメッセージを送るか、無視するかといった設定をすることはできない。

2.10.4. セキュリティレベル

LIDS のセキュリティレベルは、C2 クラスに相当する。

2.11. Medusa DS9

2.11.1. 提供元

提供元は、Medusa プロジェクトである。(http://medusa.fornax.sk/)

このプロジェクトは、スロバキア共和国の Slovak Linux User Group(SkLUG) のプロジェクトの 1 つで 1997 年の夏に誕生した。

“ medusa ”には“ jellyfish ”という意味もあり、Medusa はクラゲのようにその触手で敵を刺す、また、頭髪が蛇でその目を見た人は石に変わったといわれるギリシャ神話のゴルゴン姉妹のひとりメドゥサにその名前を由来している。“ DS9 ”の名称は、米国映画の“ StarTrek Deep Space Nine ”から借用しており、前バージョンの名称は“ Medusa NG ”で同じくスタートレックの Next Generation から借用している。

2.11.2. 状況

Medusa DS9 の最初のバージョンが公開されたのは 1998 年のことである。

最新バージョンは、2001 年 11 月 22 日にリリースされたバージョン 0.9.0 である。サポートしてる Linux カーネルのバージョンは、2.2.20 および 2.4.14 である。Linux カーネル 2.4 系のサポートがなされたのは、約 1 年ぶりに公開されたバージョン 0.8.1-alpha からで、2001 年 8 月 3 日のことである。

Linux2.4 系に対する品質はまだベータバージョンレベルである。

2.11.3. セキュリティ機能

Medusa は、下位互換を保ちながら Linux オペレーティングシステムのセキュリティ全般を向上させるパッケージである。

現在の Medusa は、Linux カーネルに対する少量のパッチと“ Constable ”と呼ばれるユーザ空間で動作する認可デーモンの 2 つの基本部分から構成されてい

る。ユーザ空間でのセキュリティ実装は、カーネルに対する変更作業を軽減でき、新しいバージョンへの対応がより迅速にそしてより柔軟に行えるという利点がある。また、認可サーバの機能改善でカーネルを変更する必要もない。

この理論は、Medusa の開発者である Marek Zelom と Milan Pikula のスロバキア工科大学での論文“ZP Security Framework”が基になっている。ZP Security Framework は二つの部分から構成されている。1 つは、“VS モデル(Virtual Spaces model)”、もう 1 つは、“セキュリティ決定センタ(Security Decision Center)”である。

2.11.3.1. ZP Security Framework

(1) VS モデル

Medusa では、実際のオペレーティングシステムに実装するには難しすぎたアクセス制御マトリックするに代わって VS(virtual spaces)モデルが採用されている。VS モデルは、Lampson[30]および Graham と Denning[31]のアクセス制御モデルがベースで、オブジェクトとサブジェクトがバーチャルスペースと呼ばれる領域に分離されている。各オブジェクトは同時に複数のバーチャルスペースのメンバになることができ、各サブジェクトにはアクセスタイプごとに権限としてのバーチャルスペースの集合が関連付けされる。与えられたアクセスタイプについて、オブジェクトとサブジェクトが共通のバーチャルスペースを持てば、そのタイプのアクセスが許可されることになる。

この VS モデルには、サブジェクトのセットとオブジェクトへの許可され

[30] Lampson, B.W. “Protection.” 5th Princeton Symposium on Information Science and Systems(1971).
[31] Graham, G. and Denning, P. “Protection – principles and practice” Spring Joint

たアクセスタイプを制御する「ACL(アクセス制御リスト)」、サブジェクトのアクセス権を表す「権限」、バーチャルスペースを使用した「セキュリティレベル[32]」、サブジェクトを分離するアクセス制御機能の「隔離機能」、Bell－LaPadula MAC モデルの「強制アクセス制御」が実装されている。

(2) セキュリティ決定センタ

セキュリティ決定センタ(SDC, Security Decision Center)のおもな役割は、システムのオブジェクトとサブジェクトのバーチャルスペースの組み合わせを変更することである。SDC は、オブジェクトやサブジェクトの生成、サブジェクトからオブジェクトへアクセスしたときのシステムイベントの検知、アクセスの有効化・無効化、また VS モデルの状態変更を行う。VS モデルの設計を含めて、SDC がどのように処理するかがシステムのセキュリティポリシーとなる。

Medusa DS9 では、SDC は“Constable”と呼ばれるユーザ空間で実行するデーモンプロセスとして実装されている。カーネルとの通信にはキャラクタ型の特殊デバイスファイルを使用し、カーネルからのメッセージを読み込み、決定内容およびバーチャルスペースの変更内容をカーネルに送り返す処理を行っている。

2.11.3.2. ファイルシステムアクセス制御

Medusa は、VFS(仮想ファイルシステム)内に全ファイル(inode)に対するバーチャルスペース(現在、32 個のバーチャルスペースをサポートしている)のビッ

Computer Conference AFIPS Press(1972).
[32] Denning, D.E. “A Lattice Model of Secure Information Flow.” Communications of the the ACM (1976).

トマップを保持している。ファイルがどのバーチャルスペースに属しているかをビットの位置で表している。すべてのファイルは、Medusa の検証に必要なオペレーションに関するビットマップと、セキュリティデーモンのための追加情報を保持している。セキュリティデーモンからの確認要求時のカーネルの性能低下は非常に小さいが、全体としての性能はセキュリティデーモンの性能に依存している。カーネル内で新しい VFS inode エントリが作成される時、Medusa は inode に関する必要な情報をセキュリティデーモンに問い合わせる。このように、セキュリティデーモンが性能向上の要因を握っており、セキュリティデーモンのコンフィグレーションに大きく依存している。Medusa は、access、create、delete、rename、link、execute のファイルオペレーションの検証をセキュリティデーモンに問い合わせる。

Medusa 独自のファイルアクセス制御として、特定のファイルへのアクセスを別のファイルに向ける「リダイレクション機能」がある。例えば、あるプログラムを実行しようとした時に、別のプログラムを実行することができる。

以下は、セキュリティデーモンの設定ファイルの例である。あるユーザが ifconfig コマンドを実行しようとしたとき、実効ユーザ ID (euid)がゼロでない(管理者権限がない)と date コマンドを実行する、という設定である。

```
for exec “/sbin/ifconfig” {
    if (euid != 0) {
        redirect “/bin/date”;
    }
}
```

システムの構成情報等を参照できるコマンドを一般ユーザに隠蔽する時に有

用な機能である。

2.11.3.3. プロセス動作制御

Medusa は、カーネル内で全プロセスに付加情報を割り当てている。この情報には、プロセスが属しているバーチャルスペースのビットマップ、そのプロセスがバーチャルスペースに存在する他のプロセスやファイルを参照できるか、Medusa によって検証されるプロセスとファイルシステムのオペレーションのビットマップおよびログイン ID が格納されている。ログイン ID は、通常ログイン時にシステムコールを通して一度だけ設定され、ユーザの身分証明、プロセスの実行者、プロセスが使用できる特権の独立化に有効である。Medusa は、fork、exec、シグナル送信、setuid プログラムの実行、setreuid/seteuid、能力チェック、ptrace の検証をセキュリティデーモンに問い合わせる。

2.11.3.4. 認可モデル

以下は、カーネル内での Medusa の認可の標準的な処理フェーズである。各項番は、フェーズ番号を表している。

(1) バーチャルスペースのチェック

このフェーズは、ターゲットとなるオブジェクトのバーチャルスペースに少なくとも 1 つがマッチするかどうかを、プロセスの要求オペレーションのタイプに関連したプロセスのバーチャルスペースのチェックを行う。例えば、ファイルへの書き込みの時、プロセスの「書き込み」のバーチャルスペースがファイルが属するバーチャルスペースに対してチェックされ、マッチすれば次のフェーズに移るが、マッチしなければそのオペレーションは拒否される。

(2) 認証状態のチェック

要求されたオペレーションタイプがMedusaによって認証されるかどうかチェックする。チェックの結果は、関連オブジェクトのカーネルレコードに依存する。Medusaがそのオペレーションを認証すればフェーズ３に進み、認証しなければフェーズ４に進む。セキュリティデーモンが実行していなければ、フェーズ３は常にスキップされる。

(3) セキュリティデーモンのチェック

カーネルは、オペレーション情報が記述されたリクエストをセキュリティデーモンに送り、その応答を待つ。セキュリティデーモンが返す応答には、以下の通信プロトコルが用意されている。

ERR　エラー。未定義のリクエスト

YES　オペレーション許可

NO　オペレーション禁止

SKIP　オペレーション禁止だが、正常リターン

OK　オペレーション許可だが、標準的にシステムパーミッションチェックに進む

セキュリティデーモンから“YES”が返ればフェーズ５に進み、“NO”が返ればそのオペレーションを否認する。“SKIP”が返ればフェーズ６に進み、“OK”の場合はフェース４に進む。“ERR”の場合は“OK”と同じようにフェーズ４に進むがカーネルのデータ構造への変更は適用されないし、Medusa が組み込まれてないかのように処理される。

(4) システムパーミッションの

チェック

このフェーズは、標準的な Linux のセキュリティモデルの基づいたカーネルのパーミッションチェックである。

(5) オペレーションの実行

プロセスからの要求されたオペレーションを実行する。

(6) リターン

制御がユーザプロセスに返る。

2.11.3.5. プラットフォームに特化したオプション

システムのアーキテクチャに依存した機能が 2 つある。1 つは、システムコールのトレース機能で、もう 1 つは任意のコード(プログラム)を強制的に実行する機能である。

以下は、任意のプログラムを強制的に実行するための設定例である。ここで、/tmp/delme ファイルを削除(unlink)しようとした時、/home/medusa/bin ディレクトリにある“exitme”というプログラムが実行される。

```
for unlink “/tmp/delme” {
    force “/home/medusa/bin/exitme”;
    answer = NO;
}
```

exitme プログラムは以下のように何もしないプログラムである。

```
#include “ fc_std.h ”
#include “ mlibc.h ”
int main(int argc, char **argv)
{
    return (0);
}
```

例えば、/tmp/delme ファイルの削除を rm コマンドで行った場合、ファイルの削除自体はセキュリティデーモンにより拒否されるが、rm コマンドはあたかも正常に終了したかのように見える。これは、rm コマンドの代わりに常に正常終了する exitme コマンドが実行されたからである。

これらの機能は、Linux カーネルのアーキテクチャ依存部分の変更が必要なため、現在は i386 アーキテクチャしかサポートしていない。

2.11.4. セキュリティレベル

Medusa DS9 のセキュリティレベルは、B1 クラスに相当する。

2.12. Rule Set Based Access Control (RSBAC)

2.12.1. 提供元

提供元は、RSBAC プロジェクトである。(http://www.rsbac.org/)

RSBAC プロジェクトは、1996 年 11 月の Amon Ott 氏のハンブルグ大学の修士論文がその始まりである。最初の公開は、1998 年 1 月 9 日に Linux カーネル 2.0.30 に対応したバージョン 0.9 である。

RSBAC は、Marshall Abrams と Leonard La Padula による Generalized Framework for Access Control (GFAC)がベースとなっており、Linux カーネルのセキュリティ機能を強化するもので、柔軟で、強力で、高速なアクセス制御を行うシステムを提供する。標準パッケージには、MAC (Mandatory Access Control)、RC (Role Compatibility)、ACL(Access Control Lists)などのモジュールが含まれている。

RSBAC の特徴は、「オープンソース(GPL)」、「周知の、そして新しいセキュリティモデル(MAC、ACL、RC)」、「モデルの組み合わせ」、「拡張が容易」、「現行カーネルのサポート」、「商用使用としての安定性」が挙げられる。

2.12.2. 状況

2001 年 8 月 27 日に公開された安定バージョン 1.1.2 は、Linux カーネル 2.2.20 および 2.4.17 に対応している。また、プレリリースバージョン 1.2.0 として、Linux カーネル 2.4.17 に対応した差分パッチも公開されている。

また、バグフィックス状況も Web で公開されており、バージョン 1.1.0、1.1.1、1.1.2 に対する不良対策用の差分パッチが Web から取得できる。

2.12.3. セキュリティ機能

2.12.3.1. 概要

RSBAC システムは、Marshall Abrams と Leonard La Padula による Generalized Framework for Access Control(GFAC)をベースに開発された。GFAC とは、データを変更可能にする権限や、執行、決断、アクセス制御データの分離、などの一般的なフレームワーク法である。

RSBAC は、執行(アクセス制御執行機能:Access Control Enforcement Facility, AEF)、決定(アクセス制御決定機能:Access Control Decision Facility, ADF)およびデータ(アクセス制御情報:Access Control Information, ACI)を分離したことにより柔軟な構造になっている。リクエスト抽出(例えば、システムコール)のため、AEF と ACI の一部はシステムに依存した構造となっている。ADF は、システム非依存なため、他の Unix 系のオペレーティングシステムで再コンパイルするだけで実行できる。

機能的にはほとんどのアクセス制御モデルのタイプをサポートしている。ADF 内でモデルを組み合わせるには、モデル間で発生した決定の矛盾を制限するメタポリシーが必要である。

実行時モジュール登録機能(Runtime Module Registration facility, REG)を使用して、決定モジュールを実行時に追加・削除することができる。

最重要機能として強力なロギングシステムがある。それは、リクエストタイプ、決定、ユーザ ID、プログラムの実行に依存したり、アクセスされたオブジェクトかどうかにかかわらずロギングされる。また、ロギングは擬似名称で処理されるのでユーザのプライバシーが守られる。

2.12.3.2. フレームワークの構造と実装

(1) サブジェクト、オブジェクト、リクエスト

RSBAC では、サブジェクトをプロセスの動作として定義している。表 2.12-1にオブジェクトタイプ(または、ターゲットタイプ)の定義を示す。

表 2.12-1 RSBAC のターゲットタイプ

ターゲット	説　明
FILE	デバイスファイルも含めたファイル
DIR	ディレクトリ
FIFO	名前付きパイプ
SYMLINK	シンボリックリンク
DEV	デバイスファイル
IPC	プロセス間通信(セマフォ、メッセージ、共用メモリ、ソケット)
SCD	システム制御データ(システム全体に影響を及ぼすオブジェクト。システムタイマ、時刻、ホスト名、ドメイン名等がある)
USER	ユーザ
PROCESS	プロセス
NONE	オブジェクトなし

アクセスモードは、抽象的なリクエストタイプにグループ化されている。サブジェクトがオブジェクトにアクセスする時は必ず、リクエストタイプ、サブジェクト、オブジェクトおよび属性データをパラメータに指定してリクエストコールが発行される。1 つのシステムコールでいくつかのリクエストコールが呼び出される。例えば、open システムコールでは SEARCH、CREATE、TRUNCATE およびすべての OPEN リクエストタイプが呼び出される。

(2) リクエストリストとターゲットタイプ

表 2.12-2にリクエストとそれに関連したターゲットタイプおよびシステムコール・関数を示す。システムコールや関数の中には条件(例えば、EXECUTE リクエストは、mmap システムコールの EXEC モードの時に発行される)によりリクエストで呼び出され、またカーネルの構成定義にも依存する。

表 2.12-2 リクエスト、ターゲット、システムコール一覧

リクエスト	説明	ターゲット	システムコール、関数
ADD_TO_KERNEL	カーネルモジュールの追加	NONE	create_module, init_module
ALTER	IPC 制御情報の変更	IPC	msgctl, shmctl
APPEND_OPEN	追加書きのオープン	FILE, DEV, IPC	open, msgsnd, sendto, sendmsg
CHANGE_GROUP	アクティブグループの変更	IPC, PROCESS, NONE	setgid, setregid, setresgid, setgroups, setfsgid, shmctl, msgctl
CHANGE_OWNER	オーナーの変更	FILE, DIR, FIFO, IPC, PROCESS, NONE	chown, lchown, fchown, setuid, setreuid, setresuid, setfsuid, shmctl, msgctl
CHDIR	作業ディレクトリの変更	DIR	chdir, fchdir, chroot
CLONE	プロセス生成	PROCESS	fork, vfork, clone
CLOSE	ファイルのクローズ	FILE, DIR, FIFO, DEV, IPC	close, shmdt, msgrcv, msgsnd, send, sendto, sendmsg, recv, recvfrom, recvmsg
CREATE	オブジェクトの生成	DIR, IPC	creat, open, mknod, mkdir, symlink, shmget, msgget, socket, accept
DELETE	オブジェクトの削除	FILE, DIR, FIFO, IPC	unlink, rmdir, msgctl, shmctl, shutdown, close
EXECUTE	プログラムの実行	FILE, NONE	exec, mmap (EXEC モードのみ), mprotect
GET_PERMISSIONS_DATA	パーミッション(モード)の取得	FILE, DIR, FIFO	access

リクエスト	説明	ターゲット	システムコール、関数
GET_STATUS_DATA	ファイル情報の取得	FILE, DIR, FIFO, IPC, SCD	open_port, open_kcore, stat, newstat, lstat, newlstat, fstat, newfstat, stat64, lstat64, fstat64, statfs, fstatfs, rsbac_stats, rsbac_check, rsbac_stats_pm, rsbac_stats_rc, rsbac_stats_acl, rsbac_log
LINK_HARD	ハードリンク	FILE, DIR, FIFO	link
MODIFY_ACCESS_DATA	ファイルのアクセス時間の変更	FILE, DIR, FIFO	utimes
MODIFY_ATTRIBUTE	RSBAC 属性の変更	すべて	--
MODIFY_PERMISSIONS_DATA	パーミッション(モード)の変更	FILE, DIR, FIFO, SCD	ioperm, iopl, chmod, fchmod
MODIFY_SYSTEM_DATA	システムデータの変更	SCD	stime, settimeofday, adjtimex, sethostname, setdomainname, setrlimit, syslog, sysctl, swapon, swapoff, rsbac_log
MOUNT	ファイルシステムのマウント	DIR, DEV	mount
READ	読み込み	DIR, FILE, FIFO, DEV, IPC	read, readv, pread, readdir, open
READ_ATTRIBUTE	RSBAC 属性値の読み込み	All target types	--
READ_OPEN	読み込み専用のオープン	FILE, FIFO, DEV, IPC	open, shmat, msgrcv, recv, recvfrom, recvmsg
READ_WRITE_OPEN	読み書き用オープン	FILE, FIFO, DEV, IPC	open, shmat, bind, connect, listen
REMOVE_FROM_KERNEL	カーネルモジュールの削除	NONE	delete_module
RENAME	リネーム	FILE, DIR, FIFO	rename
SEARCH	ディレクトリ・シンボリックリンクの検索	DIR, SYMLINK	内部関数：lookup_dentry, path_walk, lookup_hash, follow_symlink
SEND_SIGNAL	シグナル送信	PROCESS	kill
SHUTDOWN	シャットダウン、リブート	NONE	reboot

リクエスト	説明	ターゲット	システムコール、関数
SWITCH_LOG	RSBAC ログの設定変更	NONE	rsbac_adf_log_switch
SWITCH_MODULE	決定モジュールの切り替え	NONE	rsbac_switch
TERMINATE	プロセスの終了処理	PROCESS	exit
TRACE	プロセスのトレース	PROCESS	ptrace
TRUNCATE	ファイルの切り詰め	FILE	open, truncate, ftruncate, truncate64, ftruncate64
UMOUNT	ファイルシステムのアンマウント	DIR, DEV	umount
WRITE	書き込み	DIR, SCD , FILE, FIFO, DEV, IPC	write, writev, pwrite, rename, rsbac_write
WRITE_OPEN	書き込み専用オープン	FILE, FIFO, DEV, IPC	open

(3) 構造関連図

図 2.12-1 は、Linux カーネル内での RSBAC 実装の関連を表している。

通常のシステムコールにおいて、AEF は ADF への 2 種類の呼び出しを行う。1 つは決定のための要求で、もう 1 つはアクセスが許可されシステムコールの機能が正常に実行されたときの通知である。ACI データは、システムコールが他の何らかの理由で失敗する可能性があるので、通知呼び出しの延長でのみ更新される。

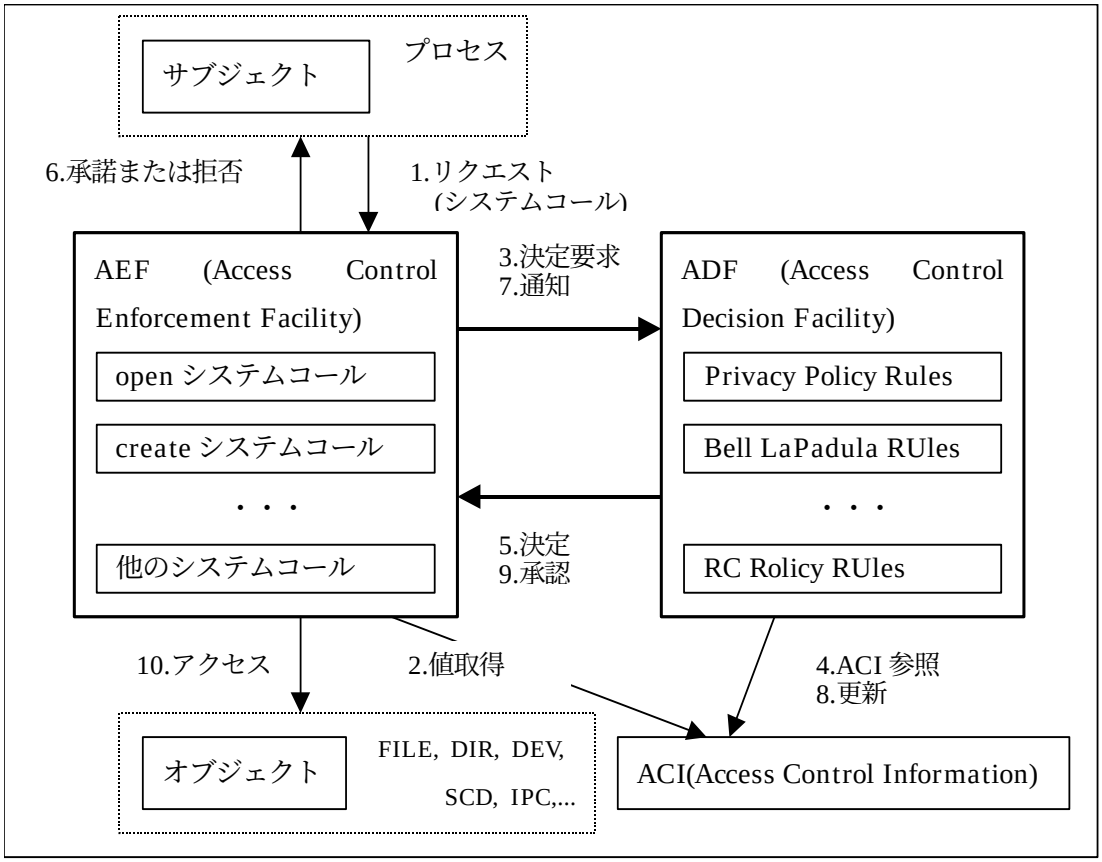

図 2.12-1 RSBAC の実装関連図

(4) モジュール登録

決定機能やシステムコールを、カーネルモジュールとして実装することにより、モジュール登録機能(REG)を通して実行時に追加することができる。つまり、必要に応じて、決定・通知機能、システムコール、属性リストを追加したり、削除したりできる。

2.12.3.3. 実装済みモデル

既にいくつかのモデルは RSBAC フレームワーク内に実装されており、カーネルのコンフィグレーションで選択することができる。

(1) MAC

強制アクセス制御(MAC)は、Bell – La Padula モデルに従って実装されており、253 に分類されたレベルと 64 個のカテゴリが利用できる。

ほとんどの Unix 系プログラムのように MAC を意識しないソフトのために、現在の read/write レベルの範囲内でのセキュリティレベルが必要に応じて自動的に調整される。

(2) FC

機能制御(Functional Control, FC)役割モデルは、「一般ユーザ(Nomal User)」、「セキュリティ管理者(Security Officer)」、「システム管理者(System Administrator)」の３種類の役割と、「一般(General)」、「セキュリティ(Security)」、「システム(System)」の 3 種類のタイプを定義している。

一般ユーザは一般データに、セキュリティ管理者は一般データとセキュリティデータに、システム管理者は一般データとシステムデータにアクセスすることができる。

(3) SIM

セキュリティ情報変更(Security Information Modification, SIM)モデルは、「セキュリティデータ」に指定されたオブジェクトへの書き込みアクセスを「セキュリティ管理者」のみに許可する機能である。

(4) PM

Simone Fischer-Hübner の複雑なプライバシーモデル(Privacy Model, PM)は、EU(ヨーロッパ連合)のプライバシー保護法に則った個人データの処理が制御できる。このモデルは、目的、タスク、アクセスなどを組にしたオ

ブジェクトクラスを定義している。

(5) MS

悪意ソフト検査機能(Malware Scanning, MS)は、アクセス制御モデルではなく、ファイルおよびソケットのリードアクセスを監視するカーネル内の検査機能である。まだ、プロトタイプのため DOS および Linux ウイルスの一部しか検知することができない。

(6) FF

ファイルフラグ(File Flags, FF)モデルは、FILE、DIR、FIFO および SYMLINK オブジェクト用に継承可能なフラグを提供する。フラグは、read_only、excecute_only、search_only、write_only、secure_delete、no_execute、no_delete_or_rename(継承不可)、append_only、add_inherited(これだけでは継承不可)が現在サポートされている。

(7) AUTH

認証モデル(AUTH)は、プロセスが CHANGE_OWNER リクエストを発行する権限を制限する。特定のユーザ ID を持つプロセスのみが AUTH 権限を許可されており、その他の setuid リクエストは拒否される。このように AUTH はユーザ ID で実行されるプログラムを制御でき、システムへのログイン経路を制限することができる。

AUTH 権限は、プログラムファイルに設定でき実行時に継承することもできる。また、特別な auth_may_set_cap フラグを持つプロセスから他のプロセスに直接設定することもできる。さらに、AUTH 権限チェックを停止させる auth_may_setuid というショートカットフラグがある。

権限の設定方法は認証デーモンによって行われ、認証デーモンのみが認

証に成功したプロセスにこれらの権限を設定することができる。

重要な基本モデルとして、AUTH は他のモデルに依存した唯一のモデルで、AUTH 権限設定に伴うすべての変更は ADF へのリクエストによって制御されている。

(8) RC

役割互換(Role Compatibility, RC)モデルは、64 の役割とターゲットごとの 64 のタイプが定義されている。容易に使用できるようにファイルシステムのターゲット(FILE, DIR, FIFO, SYMLINK)は同一の RC タイプを共有している。それぞれの役割とタイプの組に、リクエストタイプの互換ベクタが定義されている。サブジェクト-オブジェクト互換ベクタがリクエスト群のうち 1 つでも持っていれば、サブジェクトはオブジェクトにアクセスできる。

ターゲット NONE を持ったリクエストを制御できるようにするために、SCD ターゲット“other”に対してチェックされる。

すべてのユーザには、デフォルトの役割が割り当てられている。タイプに加えて、役割は他の役割と互換性を持つことができる。つまり、役割を持った実行中のプロセスは、すべての互換を持った役割に変更することができる。このように互換役割は、ある役割から目的の役割にたどり着けるように、連鎖・環状的に定義することができる。

役割はユーザばかりでなくプログラムファイルにも割り当てることができる。initial_role 属性を通して、または恒久的な force_role 属性で、プログラムの実行開始から最初の setuid 発行までの間に一時的に役割を割り当てることができる。また、force_role 属性は、役割割り当ての特殊ケースを制御できる。一般的に最初に割り当てられた役割は login プログラム

や、管理者ツールまたはデーモンプログラム用の強制的な役割に使用される。

(9) ACL

アクセス制御リスト(Access Control Lists, ACL)は、どのサブジェクトが何のリクエストタイプでどのオブジェクトにアクセスできるかを定義しており、常にオブジェクトにくくりつけられている。サブジェクトとは、RC 役割(RC モデルの拡張)、個々のユーザ、または ACL グループである。

すべてのユーザは、グローバルまたはプライベートなユーザグループを定義できる。グローバルグループは管理目的で他のユーザが使用できるが、プライベートグループは使用することはできない。オブジェクトに対するサブジェクトの ACL エントリがない場合は、親オブジェクトの ACL が使用され、オブジェクトツリーの先頭であれば、デフォルトの ACL が使用される。

管理用に 3 種類の特別なアクセス権限がある。1 つは、Access Control で標準権限を許可したり無効にしたりできる。2 つめの Forward は、自分自身が持つ標準権限を他者に渡すことができる。最後に、Supervisor はすべてが可能である。デフォルトのカーネル構成定義では、Supervisor 権限は除外されている。

ACL は、RC モデルの役割や抽出化タイプが必要なアクセス制御をカバーできない場合に推奨されているモデルであるが、ACL の設定は複雑なため RC と比べると難しい。

2.12.4. セキュリティレベル

RSBAC のセキュリティレベルは、B1 クラスに相当する。

3. まとめ

3.1. セキュリティ機能のまとめ

本調査の結果、各システムのセキュリティ機能について表 3.1-1にまとめる。

表 3.1-1セキュリティ機能のまとめ

#	システム	レベル	アクセス制御	特権機能	監査機能	暗号化機能	隔離機能	その他の機能
1	SELinux	B1	○					
2	TrustedBSD	B1	○	○	○		○	
3	OpenBSD	C1				○		
4	PitBull	B1	○	○	○	○	○	
5	hp virtual vault	B3	○	○	○	○		
6	hp linux	C2	○	○	○		○	
7	Openwall	C1						
8	Trusted Solaris	B1	○	○	○			
9	CBOSS/ LOMAC	C2	○			○		
10	LIDS	C2	○	○	○			通報機能
11	Medusa DS9	B1	○	○			○	ファイルのリダイレクション
12	RSBAC	B1	○	○				Malware Scan