システム アーキテクチャ

この付録では、IPS 5.1 のシステム アーキテクチャについて説明します。この章は、次の項で構成されています。

システムの概要

Cisco IPS ソフトウェアは、アプライアンスおよびモジュールという 2 つのプラットフォームにインストールできます（現在のアプライアンスとモジュールのリストについては、『*Installing Cisco Intrusion Prevention System Appliances and Modules 5.1*』の「Supported Sensors」を参照してください）。

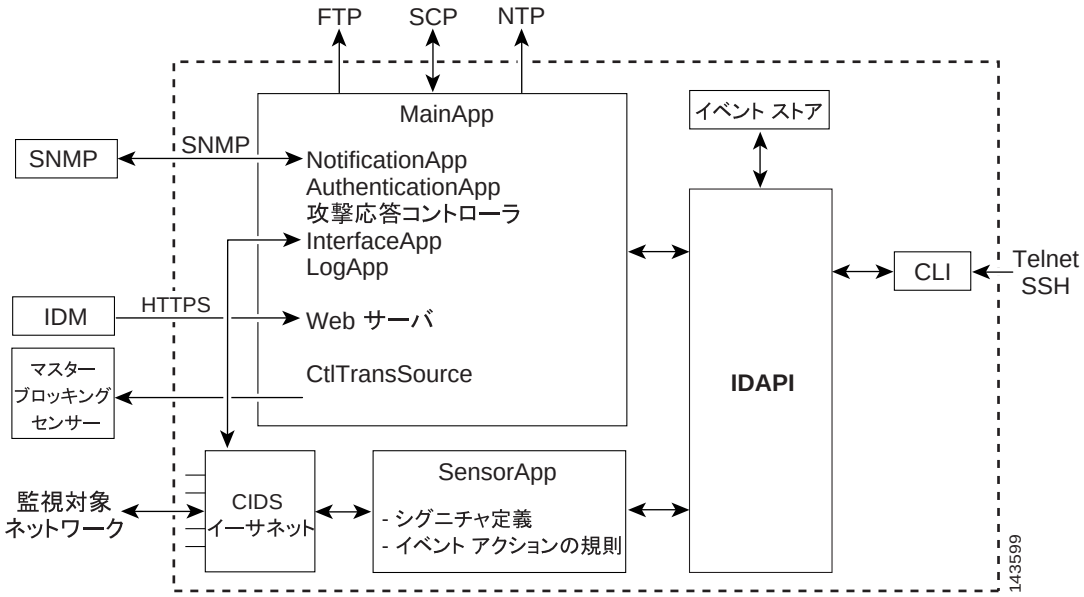
この項では、次のトピックについて説明します。

システムの設計

IPS ソフトウェアは、Linux オペレーティング システムで動作します。これまで、不必要なパッケージを OS から削除し、使用しないサービスをディセーブルにし、ネットワーク アクセスを制限し、シェルへのアクセスを削除することによって、Linux OS を強化してきました。

図 A-1 に、システムの設計を示します。

図 A-1　システムの設計



IPS ソフトウェアには次のアプリケーションが含まれています。

（注）　アプリケーションには XML 形式でそれぞれ独自のコンフィギュレーション ファイルがあります。

- MainApp：システムを初期化し、その他のアプリケーションを開始および停止し、OS を設定して、アップグレードを実行します。これには、次のコンポーネントが含まれています。
  - ctlTransSource (Control Transaction server)：センサーによる制御トランザクションの送信を許可します。これは、Attack Response Controller（以前は Network Access Controller と呼ばれていました）のマスター ブロッキング センサー機能をイネーブルにするために使用されます。
  - Event Store：CLI、IDM、ASDM、または RDEP でアクセスできる IPS イベントの格納に使用される、固定サイズのインデックス付きストア（エラー、ステータス、およびアラート システム メッセージ）。
  - InterfaceApp：バイパス設定と物理設定を処理し、対になったインターフェイスを定義します。物理的な設定は、速度、二重、および管理状態です。
  - LogApp：すべてのアプリケーションのログ メッセージをログ ファイルに書き込み、アプリケーションのエラー メッセージをイベント ストアに書き込みます。
  - Attack Response Controller（以前は Network Access Controller と呼ばれていました）：リモート ネットワーク デバイス（ファイアウォール、ルータ、およびスイッチ）を管理して、アラート イベントの発生時にブロッキング機能を提供します。ARC は、制御されたネットワーク デバイスで ACL を作成し適用するか、または **shun** コマンドを使用します（ファイアウォール）。
  - NotificationApp：アラート、ステータス、およびエラー イベントによってトリガーされる SNMP トラップを送信します。NotificationApp はパブリック ドメイン SNMP エージェントを使用します。SNMP GET は、センサーの全般的な健全性に関する情報を提供します。
  - Web サーバ（HTTP RDEP2 サーバ）：いくつかのサーブレットを使用して IPS サービスを提供することにより、Web インターフェイスおよび通信に RDEP2 を介したその他の IPS デバイスを提供します。
  - AuthenticationApp：CLI、IDM、ASDM、または RDEP アクションを実行する許可がユーザにあることを確認します。
- SensorApp (Analysis Engine)：パケットの取り込みと分析を実行します。
- CLI：Telnet または SSH でセンサーに正常にログインしたときに動作しているインターフェイス。CLI で作成されたすべてのアカウントがシェルとして CLI を使用します（ただし、サービス アカウントを除きます。許可されるサービス アカウントは 1 つだけです）。許可される CLI コマンドは、ユーザの特権によって異なります。

すべての IPS アプリケーションが、IDAPI と呼ばれる共通 API で互いに通信します。リモート アプリケーション（その他のセンサー、管理アプリケーション、サードパーティ ソフトウェア）は、RDEP2 プロトコルと SDEE プロトコルでセンサーと通信します。

センサーには次のパーティションがあります。

- アプリケーション パーティション：完全な IPS システム イメージ。
- メンテナンス パーティション：IDSM-2 のアプリケーション パーティションのイメージを再作成するために使用される、特別な目的の IPS イメージ。メンテナンス パーティションのイメージを再作成すると、すべてのコンフィギュレーション設定が失われます。
- リカバリ パーティション：センサーのリカバリに使用される、特別な目的のイメージ。ブートしてリカバリ パーティションに入ると、アプリケーション パーティションのイメージを完全に再作成することができます。ネットワーク設定は保存されますが、その他のコンフィギュレーションはすべて失われます。

IPS 5.1 の新機能

Cisco IPS 5.1 には、次の新機能があります。

- Incident Control System（ICS）のサポート

  ICS サービスは、侵入する脅威に対してより迅速に的を絞って対応できるようにすることによって、シスコの現在の IPS シグニチャ サービスを強化します。

- インライン VLAN（on a stick）

  センサーでは、1 つのセンサー インターフェイス上で 1 つまたは複数の VLAN ペア間のインライン センシングを実行できます。スイッチのバックプレーンに直接接続された Cisco Catalyst ラインカード、およびスイッチの外部ポートに接続されたアプライアンスで、この機能を使用することができます。

- レート制限

  レート制限は、ネットワーク デバイスのインターフェイス上で、指定されたタイプのトラフィックの量を帯域幅の最大容量の一定の率までに制限します。この率を超えるトラフィックは、ネットワーク デバイスによってドロップされます。レート制限では、指定されたターゲット ホストへのトラフィック、または設定されたインターフェイスや方向を通過するすべてのトラフィックを制限することができます。レートは、恒常的に、または指定した期間だけ制限することができます。レート制限は、プロトコル、オプションの宛先アドレス、およびオプションのデータ値によって指定されます。

- 新しいイベント アクション

  リリース 5.1 では、2 つの新しい拒否攻撃者イベント アクション、Deny Attacker Service Pair Inline と Deny Attacker Victim Pair Inline が追加されています。レート制限をサポートするために、新しい Request Rate Limit イベント アクションも追加されています。このイベント アクションには、拒否された攻撃者からのトラフィックの率を指定できるパラメータが用意されています。

- GRE/IPV4-in-IPV4 トンネリング

  IPS 5.1 センサーで、GRE および IPV4-in-IPV4 カプセル化トラフィックの監視が可能になりました。

ユーザによる対話操作

ユーザは、次の場合に IPS 5.1 を対話的に操作します。

- デバイスのパラメータ設定

  ユーザは、システムとその機能の初期設定を行います。これは頻繁に実行する作業ではなく、通常は一度だけ実行します。システムには妥当なデフォルト値が設定され、ユーザができるだけ変更を加えずに済むようになっています。IPS 5.1 は、CLI、IDM、IDS MC、ASDM を使用して設定できるほか、別のアプリケーションで RDEP2 および IDCONF を使用して設定することができます。

- 調整

  ユーザは、主として分析エンジンに対して、わずかなコンフィギュレーションの変更を行います。分析エンジンは、アプリケーションでネットワークのトラフィックを監視する部分です。ネットワークに最初にシステムをインストールした後で、システムが効率的に動作し、有用な情報のみが生成されるようになるまで、何度もシステムを調整することができます。カスタム シグニチャの作成、機能のイネーブル化、サービス パックの適用、およびシグニチャのアップデートが可能です。IPS 5.1 は、CLI、IDM、IDS MC、ASDM を使用して調整できるほか、別のアプリケーションで RDEP2 および IDCONF を使用して調整することができます。

- アップデート

  自動アップデートをスケジュールしたり、アプリケーションやシグニチャのデータ ファイルにすぐにアップデートを適用したりすることができます。IPS 5.1 は、CLI、IDM、IDS MC、ASDM を使用してアップデートできるほか、別のアプリケーションで RDEP2 および IDCONF を使用してアップデートすることができます。

- 情報の取得

  CLI、IDM、IDS MC、ASDMY を使用して、または、別のアプリケーションで RDEP または RDEP2 を使用して、システムからデータ（ステータス メッセージ、エラー、アラーム）を取得できます。

セキュリティ機能

IPS 5.1 には次のセキュリティ機能があります。

- ネットワーク アクセスは、アクセスを許可された特定のホストに限定されます。
- Web サーバ、SSH と SCP、または Telnet を使用して接続を試みるリモート ホストには、すべて認証が実行されます。
- デフォルトでは、Telnet アクセスはディセーブルになっています。Telnet をイネーブルにするように選択することができます。
- デフォルトでは、SSH アクセスはイネーブルです。
- FTP サーバはセンサー上では実行されません。SCP を使用するとファイルをリモートでコピーできます。
- デフォルトでは、Web サーバでは TLS または SSL が使用されます。TLS と SSL をディセーブルにするように選択することができます。
- 不要なサービスはディセーブルになります。
- CISCO-CIDS-MIB 内では、Cisco MIB Police で必要とされる SNMP セットのみが許可されます。パブリック ドメイン SNMP エージェントによって実装される OID は、MIB によって指定される場合は書き込み可能です。

MainApp

SensorApp と CLI を除くすべての IPS コンポーネントは、MainApp に含まれるようになりました。

この項では、MainApp について説明します。取り上げる事項は次のとおりです。

MainApp の機能

MainApp には、次の機能があります。

- シスコでサポートされるハードウェア プラットフォームの検証
- ソフトウェアのバージョンおよび PEP 情報の報告
- IPS コンポーネントのバージョンの起動、停止、報告
- ホスト システムの設定
- システム クロックの管理
- イベント ストアの管理
- ソフトウェア アップグレードのインストールとアンインストール
- オペレーティング システムのシャットダウンとリブート

MainApp は、**show version** コマンドに対して、次の情報を表示して応答します。

- センサーのビルドのバージョン
- MainApp のバージョン
- 実行中の各アプリケーションのバージョン
- インストールされている各アップグレードのバージョンとタイムスタンプ
- インストールされている各アップグレードの次のダウングレード バージョン
- プラットフォームのバージョン（IDS-4240 や WS-SVC-IDSM2 など）
- 別のパーティション上に構築されたセンサーのバージョン

また、MainApp はホストの統計情報も収集します。

MainApp の一部となった、イベント ストア、NotificationApp、CtlTransSource、ARC（以前の Network Access Controller）、LogApp の各アプリケーションは、イベントの保存、管理、アクション、および通信を担当します。

これらのアプリケーションには、次の新機能が含まれます。

- NotificationApp を使用した SNMP サポート

  SNMP のサポートは、このシステムの管理インターフェイスにとって最も重要な変更です。SNMP の使用により、システムの健全性と稼働状況に関する標準的な情報を入手することができます。シグニチャには、シグニチャが発生すると SNMP トラップが送信されるという新しい SNMP 通知のアクションが加わりました。

  サポートされる SNMP のバージョンは、SNMP バージョン 2 のみです。

- イベントの保存と取得

  新しいエントリ用の空き領域がなくなると、イベント ストアで最も古いエントリの有効期限が切れます。RDEP では、監査データまたは IPS アラート データのみを取得する、各種のクエリーを実行できます。すべての RDEP および RDEP2 SDEE クエリーがサポートされます。すべてのイベントは SDEE CIDEE 形式で保存されます。

- 新しい「健全性」制御トランザクション

  IDCONF の仕様では、健全性と稼働環境に関する新しいタイプの制御トランザクションが定義されています。この制御トランザクションは、システムの状態と稼働状況を報告します。

イベント ストア

この項では、イベント ストアについて説明します。取り上げる事項は次のとおりです。

イベント ストアについて

各 IPS イベントは、タイムスタンプおよび一意の単調な昇順 ID とともに、イベント ストアに保存されます。このタイムスタンプは、固定サイズのインデックス付きイベント ストア内でイベントにインデックスを付ける際の主キーとして使用されます。循環式のイベント ストアが設定されたサイズに達すると、最も古い 1 つまたは複数のイベントが新しく保存されるイベントで上書きされます。SensorApp は、イベント ストアにアラート イベントを書き込む唯一のアプリケーションです。すべてのアプリケーションが、ログ、ステータス、およびエラー イベントをイベント ストアに書き込みます。

固定サイズのインデックス付きイベント ストアでは、時刻、タイプ、優先順位、および限られた数のユーザ定義アトリビュートに基づいて、単純なイベント クエリーを実行できます。各イベントに、low、medium、または high の優先順位が割り当てられている場合は、1 つのイベント クエリーで、目的のイベント タイプ、侵入イベントの優先順位、および時間範囲を指定できます。

表 A-1 に、いくつか例を示します。

表 A-1　IPS イベントの例

IPS イベント タイプ	侵入イベントの優先順位	開始タイムスタンプの値	終了タイムスタンプの値	意味
status	—	0	最大値	保存されているすべてのステータス イベントを取得します。
error status	—	0	65743	時刻 65743 より前に保存されたすべてのエラー イベントおよびステータス イベントを取得します。
status	—	65743	最大値	時刻 65743 以降に保存されたステータス イベントを取得します。
intrusion attack response	low	0	最大値	保存されている、優先順位が low のすべての侵入イベントおよび攻撃対応イベントを取得します。

表 A-1　IPS イベントの例（続き）

IPS イベント タイプ	侵入イベントの優先順位	開始タイムスタンプの値	終了タイムスタンプの値	意味
attack response error status intrusion	medium high	4123000000	4123987256	4123000000 と 4123987256 の間の時刻に保存された、優先順位が medium または high の攻撃対応、エラー、ステータス、侵入の各イベントを取得します。

イベント ストアのサイズは、センサーが IPS イベント コンシューマに接続されていない場合にも十分なバッファリングが可能な大きさです。十分なバッファリングは、要件および使用されるノードの性能によって異なります。循環バッファ内の最も古いイベントが最新のイベントで置き換えられます。

イベントのデータ構造

各種の機能ユニットは、次の 7 種類のデータを交換します。

- 侵入イベント：SensorApp によって生成されます。センサーは侵入イベントを検出します。
- エラー イベント：ハードウェアまたはソフトウェアの誤作動によって発生します。
- ステータス イベント：コンフィギュレーションが更新された場合など、アプリケーションのステータスの変化を報告します。
- 制御トランザクション ログ イベント：センサーは制御トランザクションの結果をログに記録します。
- 攻撃対応イベント：ARC 用のアクション。ブロック要求などがあります。
- デバッグ イベント：デバッグに使用される、アプリケーションのステータス変化の詳細な報告です。
- 制御トランザクション　データ：制御トランザクションに関連付けられたデータです。アプリケーションの診断データ、セッション ログ、アプリケーションとの間でやりとりされるコンフィギュレーション データなどがあります。

この全 7 種類のデータは、集合的に *IPS データ*と呼ばれます。侵入、エラー、ステータス、制御トランザクション ログ、ネットワーク アクセス、デバッグの 6 つのイベント タイプは、類似した特性を持ち、集合的に *IPS イベント*と呼ばれます。IPS イベントは、IPS を構成する複数の異なるアプリケーションによって生成され､他の IPS アプリケーションはこれらのイベントに登録しています。IPS イベントには、次の特性があります。

- イベントを発生させるように設定されたアプリケーション　インスタンスから自然発生的に生成されます。別のアプリケーション インスタンスから特定のイベントを生成するように要求があるわけではありません。
- 特定の宛先を持ちません。イベントは保存され、その後 1 つまたは複数のアプリケーション インスタンスによって取得されます。

制御トランザクションには、次のタイプの要求が関係します。

- アプリケーション インスタンスのコンフィギュレーション データの更新要求
- アプリケーション インスタンスの診断データの要求
- アプリケーション インスタンスの診断データのリセット要求
- アプリケーション インスタンスの再起動要求
- ブロック要求など、ARC に対する要求

制御トランザクションには、次の特性があります。

- 常に要求とそれに続く応答で構成されます。

  要求と応答には、任意の量のデータを関連付けることができます。応答には、必ず少なくとも肯定または否定の確認応答が含まれます。
- ポイントツーポイントのトランザクションです。

  制御トランザクションは、1 つのアプリケーション インスタンス（発信側）から別のアプリケーション インスタンス（応答側）に送信されます。

IPS データは、XML 文書として、XML 形式で表現されます。ユーザ設定可能なパラメータは、複数の XML ファイルに保存されます。

IPS イベント

IPS アプリケーションは、IPS イベントを生成していくつかの stimulus メッセージの発生を報告します。イベントは、SensorApp によって生成されるアラートや、任意のアプリケーションによって生成されるエラーなどのデータです。イベントは、イベント ストアと呼ばれるローカル データベースに保存されます。

イベントには、次の 5 つのタイプがあります。

- <evAlert>: ネットワーク アクティビティによってシグニチャがトリガーされると報告されるアラート イベント メッセージ。
- <evStatus> : IPS アプリケーションのステータスとアクションを報告するステータス イベント メッセージ。
- <evError> : 応答アクションの試行中に発生したエラーを報告するエラー イベント メッセージ。
- <evLogTransaction> : 各センサー アプリケーションによって処理された制御トランザクションを報告するログ トランザクション メッセージ。
- <evShunRqst> : ARC がブロック要求を発行すると報告されるブロック要求メッセージ。

ステータス メッセージとエラー メッセージは、CLI、IDM、および ASDM を使用して表示することができます。

SensorApp と ARC は、応答アクション（TCP リセット、IP ロギングの開始と終了、ブロッキングの開始と終了、トリガー パケット）をステータス メッセージとしてログに記録します。

NotificationApp

NotificationApp により、センサーは、アラートおよびシステム エラー メッセージを SNMP トラップとして送信することができます。NotificationApp はイベント ストアのイベントに登録し、それらのイベントを SNMP MIB に変換して、パブリックドメイン SNMP エージェントを使用して宛先に送信します。NotificationApp は、set および get の送信をサポートします。SNMP GET では、センサーの健全性に関する基本情報を取得できます。

希薄モードでは、NotificationApp は <evAlert> イベントから次の情報を送信します。

- 発信者情報
- イベント ID
- イベントの重大度
- 時刻（UTC および現地時間）
- シグニチャ名
- シグニチャ ID
- サブシグニチャ ID

- 参加者情報
- アラーム特性

詳細モードでは、NotificationApp は <evAlert> イベントから次の情報を送信します。

- 発信者情報
- イベント ID
- イベントの重大度
- 時刻（UTC および現地時間）
- シグニチャ名
- シグニチャ ID
- サブシグニチャ ID
- バージョン
- 要約
- インターフェイス グループ
- VLAN
- 参加者情報
- アクション
- アラーム特性
- シグニチャ
- IP ログ ID

NotificationApp は、ユーザが定義したフィルタに従って、どの <evError> をトラップとして送信するかを決定します。エラーの重大度（error、fatal、warning）に基づいてフィルタリングすることができます。NotificationApp は、<evError> イベントから次の情報を送信します。

- 発信者情報
- イベント ID
- イベントの重大度
- 時刻（UTC および現地時間）
- エラー メッセージ

NotificationApp は、センサーからの次の一般的な健全性情報およびシステム情報の GET をサポートします。

- パケット損失
- パケット拒否数
- 生成されたアラーム数
- FRP 内のフラグメント数
- FRP 内のデータグラム数
- 初期状態の TCP ストリーム数
- 確立状態の TCP ストリーム数
- 終了状態の TCP ストリーム数
- システム内の TCP ストリーム数
- 再構成用にキューイングされた TCP パケット数
- アクティブなノードの合計数
- 両方の IP アドレスと両方のポートでキー付けされた TCP ノード数
- 両方の IP アドレスと両方のポートでキー付けされた UDP ノード数
- 両方の IP アドレスでキー付けされた IP ノード数

- センサー メモリのクリティカルな段階
- インターフェイスのステータス
- コマンド / コントロール パケットの統計情報
- フェールオーバーの状態
- システムの動作時間
- CPU 使用状況
- システムのメモリ使用状況
- PEP

（注）　すべての IDS および IPS プラットフォームで PEP がサポートされるわけではありません。

NotificationApp は、次の統計情報を提供します。

- エラー トラップ数
- イベント アクション トラップ数
- SNMP GET 要求数
- SNMP SET 要求数

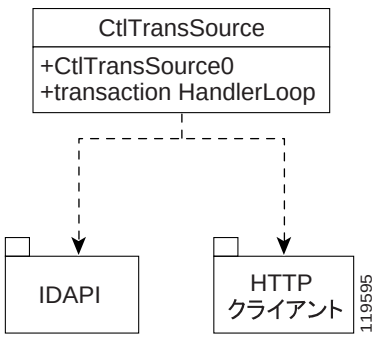
CtlTransSource

CtlTransSource は、ローカルに開始されたリモート制御トランザクションを RDEP および HTTP プロトコルを使用してリモートの宛先に転送するアプリケーションです。CtlTransSource は、TLS または非 TLS 接続を開始し、このような接続を介してリモート制御トランザクションが HTTP サーバと通信するようにします。

CtlTransSource は、リモート制御トランザクションを実行するために、リモート HTTP サーバに対して十分なクレデンシャルを確立する必要があります。CtlTransSource は、リモート ノード上の HTTP サーバに対してユーザ名とパスワードの形式（基本認証）で ID を提示することにより、クレデンシャルを確立します。認証が成功すると、要求者には、その接続の各要求で提示する必要のあるユーザ認証を格納した Cookie が割り当てられます。

CtlTransSource の transactionHandlerLoop メソッドは、リモート制御トランザクションのプロキシの役割を果たします。ローカル アプリケーションがリモート制御トランザクションを開始すると、IDAPI は最初にそのトランザクションを CtlTransSource に送ります。transactionHandlerLoop メソッドは、CtlTransSource に送られたリモート制御トランザクションを待つループです。図 A-2 は、CtlTransSource の transactionHandlerLoop メソッドを示しています。

図 A-2　CtlTransSource



transactionHandlerLoop は、リモートに対するトランザクションを受け取ると、リモート制御トランザクションをリモートの宛先に転送しようとします。transactionHandlerLoop は、トランザクションを RDEP 制御トランザクション メッセージの形式にします。transactionHandlerLoop は、HttpClient クラスを使用して、リモート ノード上の HTTP サーバに対して RDEP 制御トランザクション要求を発行します。リモート HTTP サーバは、リモート制御トランザクションを処理し、HTTP 応答で適切な RDEP 応答メッセージを返します。リモート HTTP サーバが IPS Web サーバの場合、Web サーバは CtlTransSource サーブレットを使用してリモート制御トランザクションを処理します。

transactionHandlerLoop は、制御トランザクションの応答として、RDEP 応答か障害応答のいずれかをリモート制御トランザクションの発信側に返します。HTTP サーバが未認可のステータス応答（HTTP クライアントの HTTP サーバに対するクレデンシャルが不十分であることを示す）を返すと、transactionHandlerLoop は要求者の ID を認証するために、CtlTransSource の指定されたユーザ名とパスワードを使用してトランザクション要求を再発行します。transactionHandlerLoop は、終了を指示する制御トランザクションを受け取るか、終了イベントのシグナルが発生するまでループを続けます。

ARC

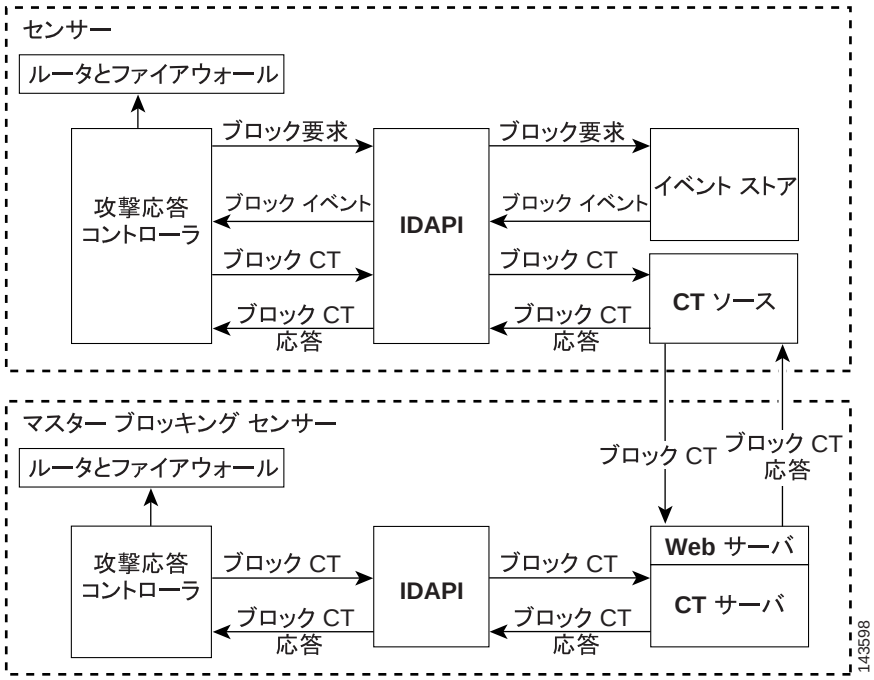
この項では、ARC について説明します。ARC は、ルータ、スイッチ、およびファイアウォールでブロッキングを開始および終了する IPS アプリケーションです。*ブロック*は、デバイスのコンフィギュレーションまたは ACL 内で、特定のホスト IP アドレスまたはネットワーク アドレスの着信および発信トラフィックをブロックするエントリです。また、ARC は Cisco IOS 12.3 を実行するルータでレート制限の制御も実行します。

この項では、次のトピックについて説明します。

ARC について

ARC の主な機能はイベントをブロックすることです。ブロックに応答する場合、ARC は、管理しているデバイスと直接対話してブロックをイネーブルにするか、制御トランザクション サーバを使用してマスター ブロッキング センサーにブロック要求を送信します。マスター ブロッキング センサー上の Web サーバは、制御トランザクションを受け取り、制御トランザクション サーバに渡します。その後、トランザクションは ARC に渡されます。マスター ブロッキング センサーの ARC は、その後管理しているデバイスと対話して、ブロックをイネーブルにします。図 A-3 は、ARC を示しています。

図 A-3　　ARC



（注）　1 つの ARC インスタンスは、0 台以上のネットワーク デバイスを制御することができます。ARC は、他の ARC アプリケーション、IPS 管理ソフトウェア、その他のネットワーク管理ソフトウェア、またはシステム管理者とネットワーク デバイスの制御を分担することはありません。特定のセンサー上では、1 つの ARC インスタンスのみの実行が許可されます。

ARC は次のいずれかに応答してブロックを開始します。

- ブロック アクションとともに設定されているシグニチャから生成されたアラート イベント
- CLI、IDM、または ASDM を使用して手動で設定されたブロック
- ホストまたはネットワーク アドレスに対して永続的に設定されたブロック

デバイスをブロックするように ARC を設定すると、ARC はデバイスと Telnet 接続または SSH 接続を開始します。ARC は、各デバイスとの接続を維持します。ブロックが開始されると、ARC は新しいコンフィギュレーションまたは ACL のセット（インターフェイスの方向ごとに 1 つずつ）を制御対象の各デバイスに配信します。ブロックが完了すると、すべてのコンフィギュレーションまたは ACL がブロックを削除するようにアップデートされます。

ARC の機能

ARC には、次の機能があります。

- 3DES（デフォルト）または DES 暗号化に対応した Telnet および SSH 1.5 を使用した通信

  該当デバイスの ARC コンフィギュレーションで指定されたプロトコルのみが試行されます。何らかの理由で接続に失敗すると、ARC は接続を再び確立しようとします。

- ルータ上の既存 ACL およびスイッチ上の既存 VACL

  ARC によって制御されるルータのインターフェイスまたは方向に既存の ACL がある場合は、その ACL を ARC によって生成されたコンフィギュレーションにマージするように指定することができます。マージは、preblock ACL を指定してすべてのブロックの前で実行するか、postblock ACL を指定してすべてのブロックの後で実行することができます。Catalyst 6000 VACL デバイス タイプでは、ARC が制御するインターフェイスごとに preblock VACL および postblock VACL を指定することができます。ファイアウォール デバイス タイプでは、別の API を使用してブロックが実行されるため、ARC はファイアウォール上の既存の ACL にはまったく影響しません。

  （注） Catalyst 5000 RSM および Catalyst 6000 MSFC2 の各ネットワーク デバイスは、Cisco ルータと同様にサポートされます。

  詳細については、P.A-16 の「ACL と VACL」を参照してください。

- リストに記載されたリモート センサーへのブロックの転送

  ARC は、リストに記載された複数のリモート センサーにブロックを転送して、事実上それらが共同で 1 つのネットワーク デバイスを制御できるようにすることができます。このようなリモート センサーをマスター ブロッキング センサーと呼びます。マスター ブロッキング センサーの詳細については、P.10-35 の「センサーをマスター ブロッキング センサーとして使用するための設定」を参照してください。

- ネットワーク デバイス上でのブロッキング インターフェイスの指定

  ルータの ARC コンフィギュレーションでは、ブロッキングを実行するインターフェイスと方向を指定できます。VACL のコンフィギュレーションでは、ブロッキングを実行するインターフェイスを指定できます。

  （注） Cisco ファイアウォールでは、インターフェイスや方向に基づいたブロッキングは行われないため、このコンフィギュレーションが指定されることはありません。

  ARC は、最高 250 のインターフェイスを同時に制御できます。

- ホストまたはネットワークの指定された期間のブロッキング

  ARC は、ホストまたはネットワークを指定された期間（分単位）または無限にブロックすることができます。ARC は、ブロックの有効期限が切れるとそれを確認し、その時点でホストまたはネットワークのブロックを解除します。

- 重要なイベントのロギング

  ARC は、ブロックまたはブロック解除のアクションが正常に完了するか、エラーが発生すると、確認イベントを書き込みます。また、ARC は、ネットワーク デバイスの通信セッションの切断とリカバリ、コンフィギュレーション エラー、ネットワーク デバイスから報告されたエラーなどの重要なイベントもログに記録します。

- ARC の再起動前後のブロッキング状態の維持

  ARC は、シャットダウンまたは再起動が発生した時点で有効期限が切れていないブロックを再適用します。ARC は、シャットダウン中に有効期限が切れたブロックを削除します。

  （注） ARC は、アプリケーションのシャットダウン中にシステム時間が変更されていない場合にのみ、ブロッキング状態を維持します。

  詳細については、P.A-16 の「再起動前後の状態の維持」を参照してください。

- ネットワーク デバイスの再起動前後のブロッキング状態の維持

  ARC は、ネットワーク デバイスがシャットダウンおよび再起動されると、必要に応じてブロックを再適用し、有効期限が切れたブロックは削除します。ARC は ARC の同時または重複したシャットダウンと再起動の影響を受けません。

- 認証と認可

  ARC は、リモート TACACS+ サーバを使用の使用も含めて、AAA 認証および認可を使用するネットワーク デバイスとの間に通信セッションを確立することができます。

- 2 タイプのブロッキング

  ARC は、ホスト ブロックとネットワーク ブロックをサポートします。ホスト ブロックは、接続ベースまたは無条件です。ネットワーク ブロックは常に無条件です。

  詳細については、P.A-17 の「接続ベースおよび無条件のブロッキング」を参照してください。

- NAT アドレッシング

  ARC は、センサーに NAT アドレスを使用するネットワーク デバイスを制御することができます。ネットワーク デバイスの設定時に NAT アドレスを指定すると、そのデバイスのブロックからセンサーのアドレスがフィルタリングされるときに、ローカル IP アドレスではなく、その NAT アドレスが使用されます。

- シングル ポイント制御

  ARC は、管理者やその他のソフトウェアとネットワーク デバイスの制御を分担することはありません。コンフィギュレーションを更新する必要がある場合は、変更が完了するまで ARC をシャットダウンしてください。ARC は、CLI または任意の IPS マネージャを使用してイネーブルまたはディセーブルにできます。ARC は、再度イネーブルになると、制御対象の各ネットワーク デバイスの最新コンフィギュレーションの再読み取りも含めて、ARC 自体を完全に再度初期化します。

（注） ファイアウォールなどのネットワーク デバイスを設定するときには、ARC のブロッキングをディセーブルにすることをお勧めします。

- 任意の時点で最高 250 のアクティブなブロックを維持

  ARC は、同時に最高 250 のアクティブなブロックを維持できます。ARC は最高 65535 のブロックをサポートできますが、同時に許可する数は 250 までにすることをお勧めします。

（注） ブロックの数は、インターフェイスおよび方向の数と同じではありません。

サポートされているブロッキング デバイス

ARC は、次のデバイスを制御できます。

- Cisco IOS 11.2 以降を実行する Cisco ルータ

（注） レート制限を実行するには、ルータで Cisco IOS 12.3 以降を実行している必要があります。

- スーパーバイザ エンジンでスーパーバイザ エンジン ソフトウェア 5.3(1) 以降を実行し、RSM で IOS 11.2(9)P 以降を実行する Catalyst 5000 シリーズ スイッチ

（注） ブロッキングは RSM で実行されるため、RSM が必要です。

- Catalyst ソフトウェア 5.3 以降を実行する、PFC がインストールされた Catalyst 6000 シリーズ スイッチ
- Catalyst ソフトウェア 5.4(3) 以降がインストールされ、MSFC2 に Cisco IOS 12.1(2)E 以降がインストールされた Catalyst 6000 MSFC2
- Cisco ASA 500 シリーズの、ASA 5510、ASA 5520、ASA 5540 の各モデル
- FWSM

（注）　FWSM はマルチモードの管理コンテキストではブロックを実行できません。

ACL と VACL

ARC で制御されるインターフェイスまたは方向でパケットをフィルタリングする場合は、すべてのブロックの前に ACL を適用したり（preblock ACL）、すべてのブロックの後で ACL を適用したりする（postblock ACL）ように ARC を設定することができます。これらの ACL は、ネットワーク デバイス上で非アクティブな ACL として設定されます。preblock ACL および postblock ACL は、インターフェイスおよび方向ごとに定義できます。ARC は、このリストを取得してキャッシュし、ネットワーク デバイス上のアクティブな ACL をアップデートするときに、ブロッキングを行う ACE とその内容をマージします。ほとんどの場合は、既存の ACL を postblock ACL として指定して、その ACL がブロックの有効な動作を妨げないようにします。ACL は、パケットを最初に検索された ACE と照合することによって動作します。この最初の ACE でパケットが許可されると、それより後ろの deny 文は検索されません。

インターフェイスと方向ごとに異なる preblock ACL や postblock ACL を指定したり、同じ ACL を複数のインターフェイスや方向に再利用したりすることができます。preblock リストを維持しない場合は、ブロック対象外オプションを使用し、既存の設定文を使用して常にホストやネットワークをブロックすることができます。永久ブロックは、タイムアウト値が -1 の通常のブロックです。

ARC は、その ARC が所有している ACL のみを変更します。ARC はユーザ定義の ACL は変更しません。ARC によって管理される ACL では特殊な形式が使用されますが、これはユーザ定義の ACL では使用できません。命名規則は、**IPS_**<interface_name>**_[in | out]_[0 | 1]** です。<interface_name> は、ARC のコンフィギュレーションで指定されている、ブロックするインターフェイスの名前です。

Catalyst スイッチの場合は、ブロックするインターフェイスの VLAN 番号になります。これらの名前を preblock ACL や postblock ACL に使用しないでください。

Catalyst 6000 の VACL では、preblock および postblock VACL を指定でき、インターフェイスのみの指定が可能です（VLAN では方向は使用されません）。

ファイアウォールの場合は、ブロッキングに異なる API が使用されるため、preblock ACL と postblock ACL は使用できません。代わりに、ファイアウォールに直接 ACL を作成する必要があります。詳細については、P.A-18 の「Cisco ファイアウォールでのブロッキング」を参照してください。

再起動前後の状態の維持

センサーがシャットダウンされると、ARC はすべてのブロックとレート制限を（開始タイムスタンプとともに）ARC によって管理されるローカル ファイル（nac.shun.txt）に書き込みます。ARC が起動すると、管理対象のネットワーク デバイスでブロックのアップデートが必要かどうかがこのファイルを使用して判断されます。ファイル内に有効期限が切れていないブロックがあれば、起動時にすべてネットワーク デバイスに適用されます。ARC がシャットダウンされるときには、未完了の有効なブロックがあっても ACL に対して特別なアクションは実行されません。nac.shun.txt ファイルが正確であるのは、ARC が実行されていない間にシステム時間が変更されなかった場合のみです。

注意　nac.shun.txt ファイルを手動で変更しないでください。

次のシナリオは、ARC が再起動の前後でどのように状態を維持するかを示しています。

シナリオ 1

ARC の停止時点で 2 つのブロックが有効であり、ARC の再起動前に 1 つの有効期限が切れます。ARC は、再起動すると、最初に nac.shun.txt ファイルを読み取ります。その後、ARC は preblock および postblock の ACL または VACL を読み取ります。アクティブな ACL または VACL が次の順序で作成されます。

1. **allow** *sensor_ ip_address* コマンド（**allow sensor shun** コマンドが設定されていない場合に限られます）
2. preblock ACL
3. コンフィギュレーションからの **always block** コマンド入力
4. nac.shun.txt からの期限が切れていないブロック
5. postblock ACL

ARC のコンフィギュレーションで、特定のホストがブロック対象外として指定されている場合、そのホストは ACL 内の permit 文に入れられません。代わりに、そのホストは ARC によってキャッシュされ、着信する addShunEvent イベントおよび addShunEntry 制御トランザクションのフィルタリングに使用されます。

シナリオ 2

preblock または postblock ACL は指定されていませんが、既存のアクティブな ACL が存在します。新しい ACL は、次の順序で作成されます。

1. **allow** *sensor_ ip_address* コマンド（**allow sensor shun** コマンドが設定されていない場合に限られます）
2. コンフィギュレーションからの **always block** コマンド入力
3. nac.shun.txt からの期限が切れていないブロック
4. **permit IP any any** コマンド

接続ベースおよび無条件のブロッキング

ARC は、ホストとネットワークで、それぞれ 2 種類と 1 種類のブロッキングをサポートします。ホスト ブロックは、接続ベースまたは無条件です。ネットワーク ブロックは常に無条件です。

ホスト ブロックを受信すると、ARC はホスト ブロックの connectionShun アトリビュートを確認します。connectionShun が true に設定されている場合、ARC は接続ブロッキングを実行します。ホスト ブロックでは、宛先 IP アドレス、送信元ポート、宛先ポート、プロトコルなどのオプション パラメータを指定できます。接続ブロックを実行するには、少なくとも送信元と宛先の IP アドレスが必要です。接続ブロックに送信元ポートが存在する場合、その送信元ポートは無視され、ブロックには含まれません。

次の条件に当てはまる場合、ARC はブロックを強制的に無条件にし、必要に応じてブロックを接続タイプから変換します。

- 指定された送信元 IP アドレスに関して任意のタイプのブロックがアクティブである
- その送信元 IP アドレスに関する任意のタイプの新しいブロックを受け取った
- 新しいブロックは、古いブロックとオプション パラメータ（送信元ポート以外）が異なっている

ブロックが更新される（たとえば、該当の送信元 IP アドレスまたはネットワークに関する既存のブロックがすでに有効であるときに新しいブロックを受け取る）と、既存のブロックの残り時間（分単位）が確認されます。新しいブロックの時間が残り時間（分単位）以下である場合は、アクションは発生しません。そうでない場合は、新しいブロックのタイムアウトで既存のブロックのタイムアウトが置き換えられます。

注意　Cisco ファイアウォールは、ホストの接続ブロッキングをサポートしません。接続ブロックが適用されると、ファイアウォールはそれを無条件のブロックのように扱います。また、Cisco ファイアウォールは、ネットワーク ブロッキングもサポートしません。ARC は、Cisco ファイアウォールにネットワーク ブロックを適用しません。

Cisco ファイアウォールでのブロッキング

ARC は、**shun** コマンドを使用してファイアウォールでのブロッキングを実行します。**shun** コマンドの形式は、次のようになっています。

- IP アドレスをブロックする場合
  shun srcip [*destination_ip_address source_port destination_port* [*port*]]
- IP アドレスのブロックを解除する場合
  no shun ip
- すべてのブロックをクリアする場合
  clear shun
- アクティブなブロックを表示したり、実際にブロックされたグローバル アドレスを表示したりする場合
  show shun [*ip_address*]

ARC は、**show shun** コマンドへの応答を使用して、ブロックが実行されたかどうかを確認します。

shun コマンドは、既存の ACL、コンジット、または outbound コマンドを置き換えないため、既存のファイアウォール コンフィギュレーションをキャッシュしたり、ブロックをファイアウォール コンフィギュレーションにマージしたりする必要はありません。

注意　ARC の実行中は、手動のブロックや、既存のファイアウォール コンフィギュレーションの変更を実行しないでください。

block コマンドで送信元 IP アドレスのみが指定されている場合、既存のアクティブな TCP 接続は切断されませんが、ブロックされたホストからのすべての着信パケットはドロップされます。

ARC が最初に起動すると、ファイアウォール内のアクティブなブロックが内部のブロッキング リストと比較されます。内部リストに対応するエントリのないブロックはすべて削除されます。

ARC は、ファイアウォール上でローカル ユーザ名または TACACS+ サーバを使用した認証をサポートします。TACACS+ サーバを使用せずに AAA で認証を実行するようにファイアウォールを設定すると、ARC は予約されたユーザ名 *pix* を使用してファイアウォールと通信します。

ファイアウォールで TACACS+ サーバを認証に使用する場合は、TACACS+ ユーザ名を使用します。AAA ログインを使用する一部のファイアウォール コンフィギュレーションでは、最初のファイアウォール パスワード、AAA パスワード、およびイネーブル パスワードの入力を求める 3 つのプロンプトが表示されます。ARC では、最初のファイアウォール パスワードと AAA パスワードが同じである必要があります。

ファイアウォールが NAT または PAT を使用するように設定され、センサーがファイアウォール外部のネットワークのパケットをチェックしている場合、ファイアウォール内部のネットワークから発信されたホスト攻撃が検出されると、センサーはファイアウォールによって提供される変換されたアドレスをブロックしようとします。ダイナミック NAT アドレッシングを使用している場合は、ブロックが無効になったり、正常なホストがブロックされたりする場合があります。PAT アドレッシングを使用している場合は、ファイアウォールで内部ネットワーク全体がブロックされる場合があります。このような状況を避けるためには、センサーを内部インターフェイス上に配置するか、センサーがブロックを実行しないように設定します。

Catalyst スイッチでのブロッキング

PFC を搭載した Catalyst スイッチは、VACL を使用してパケットをフィルタリングします。VACL は、VLAN 間、および VLAN 内でパケットをフィルタリングします。

MSFC ルータの ACL は、WAN カードがインストールされている場合にサポートされ、その場合はセンサーが MSFC2 を介してインターフェイスを制御します。

（注） VACL を使用したブロッキングの場合、MSFC2 カードは Catalyst スイッチのコンフィギュレーションに必須の構成要素ではありません。

注意 Catalyst スイッチ用に ARC を設定する場合は、制御対象のインターフェイスで方向を指定しないでください。インターフェイス名は VLAN 番号です。preblock および postblock のリストは VACL である必要があります。

Catalyst の VACL には、次のコマンドが適用されます。

- 既存の VACL を表示する場合
  show security acl info *acl_name*
- アドレスをブロックする場合（*address_spec* は、ルータの ACL で使用されるものと同じ）
  set security acl ip *acl_name* **deny** *address_spec*
- リストの作成後に VACL をアクティブにする場合
  commit security acl all
- 1 つの VACL をクリアする場合
  clear security acl map *acl_name*
- すべての VACL をクリアする場合
  clear security acl map all
- VACL を VLAN にマッピングする場合
  set sec acl *acl_name vlans*

LogApp

センサーは、すべてのイベント（アラート、エラー、ステータス、およびデバッグ メッセージ）のログを、永続的な循環バッファに記録します。また、センサーは IP ログも生成します。メッセージと IP ログには、CLI、IDM、ASDM、および RDEP クライアントを使用してアクセスできます。

IPS アプリケーションは、LogApp を使用してメッセージのログを記録します。LogApp は、debug、timing、warning、error、fatal の 5 つの重大度レベルのログ メッセージをすべて送信します。LogApp は、循環テキスト ファイルである /usr/cids/idsRoot/log/main.log にログ メッセージを書き込みます。ファイルが最大サイズに達すると、新しいメッセージが古いメッセージを上書きするため、最後に書き込まれたメッセージが main.log の最後に表示されるとは限りません。main.log に最後に書き込まれた行を探すには、文字列「= END OF FILE =」を検索します。

main.log は、**show tech-support** コマンドの出力に含まれます。メッセージが warning およびそれ以上（error または fatal）のレベルでログに記録されると、LogApp はそのメッセージを evError イベント（対応するエラー重大度の）に変換し、イベント ストアに挿入します。

（注）　テクニカル サポート情報を表示する手順については、P.13-22 の「テクニカル サポート情報の表示」を参照してください。イベントを表示する手順については、P.13-5 の「イベントの表示」を参照してください。

LogApp は、cron メッセージを除いて、informational およびそれ以上のレベルのすべての syslog メッセージを受け取り（*.info;cron.none）、それらをエラー重大度が Warning に設定された <evErrors> としてイベント ストアに挿入します。LogApp およびアプリケーションのロギングは、service logger コマンドで制御されます。

LogApp では、異なるロギング ゾーンのロギング重大度を制御することにより、各アプリケーションでどのようなログ メッセージを生成するかを制御することができます。ユーザは、TAC のエンジニアまたは開発者の要請と指示の下でのみ、logger service の個別ゾーン コントロールにアクセスします。TAC は、トラブルシューティングの目的で、ユーザにデバッグ ロギングを有効にするように要請する場合があります。詳細については、P.C-28 の「デバッグ ロギングのイネーブル化」を参照してください。

AuthenticationApp

この項では、AuthenticationApp について説明します。取り上げる事項は次のとおりです。

AuthenticationApp の機能

AuthenticationApp には、次の機能があります。

- ユーザの ID の認証
- ユーザのアカウント、特権、鍵、および証明書の管理
- AuthenticationApp およびセンサー上のその他のアクセス サービスで使用される認証方式の設定

ユーザの認証

ユーザのアクセスに関して適切なセキュリティを確立するために、センサーに認証を設定する必要があります。センサーをインストールすると、有効期限切れのパスワードを持つ初期の cisco アカウントが作成されます。センサーに対する管理アクセス権を持つユーザは、CLI、または IDM や ASDM などの IPS マネージャを使用して、このデフォルトの管理アカウント（cisco）を使用してログインすることによってセンサーにアクセスします。CLI では、管理者はパスワードを変更するように求められます。IPS マネージャでは、アカウントのパスワードを変更するために、setEnableAuthenticationTokenStatus 制御トランザクションが開始されます。

管理者は、CLI または IPS マネージャを使用して、使用する認証方式を設定します。たとえば、ユーザ名とパスワードや、SSH 許可鍵の使用です。管理者にサービスを提供するアプリケーションが setAuthenticationConfig 制御トランザクションを開始し、認証のコンフィギュレーションが実行されます。

認証のコンフィギュレーションには、アカウントのロックの処理方法を指定するために使用されるログイン試行回数の制限値なども含まれます。アカウントのロックは、特定のアカウントで連続してログインの試行に失敗した回数がログイン試行の制限値を超えると発生します。アカウントがロックされると、そのアカウントに対するそれ以降のログインの試行は拒否されます。アカウントは、setEnableAuthenticationTokenStatus 制御トランザクションを使用してアカウントの認証トークンをリセットすることによってロック解除されます。ログイン試行の制限値がゼロに設定されている場合、アカウントのロック機能はディセーブルになります。

管理者は、CLI または IPS マネージャを使用して、ユーザ アカウントを追加することができます。詳細については、P.A-29 の「ユーザ ロール」を参照してください。

センサーでの認証の設定

ユーザが Web サーバや CLI などのサービスを使用してセンサーにアクセスしようとする場合は、ユーザの ID が認証され、ユーザの特権が確定される必要があります。ユーザのアクセスを可能にするサービスにより、ユーザの ID を認証するように AuthenticationApp に求める execAuthenticateUser 制御トランザクション要求が開始されます。この制御トランザクション要求には、通常、ユーザ名とパスワードが含まれます。また、ユーザの ID は SSH 許可鍵によっても認証できます。

AuthenticationApp は、ユーザの ID の認証を試行して、execAuthenticateUser 制御トランザクション要求に応答します。AuthenticationApp は、ユーザの認証ステータスと特権を含んだ制御トランザクション応答を返します。ユーザの ID を認証できない場合、AuthenticationApp は、制御トランザクション応答で未認証のステータスと匿名ユーザ特権を返します。また、この制御トランザクション応答では、アカウントのパスワードの有効期限が切れているかどうかも示されます。execAuthenticateUser 制御トランザクションを開始してユーザを認証するユーザ インターフェイス アプリケーションによって、パスワードの変更を求めるプロンプトが表示されます。

AuthenticationApp は、基盤となるオペレーティング システムを使用してユーザの ID を確認します。すべての IPS アプリケーションは制御トランザクションを AuthenticationApp に送信し、AuthenticationApp がオペレーティング システムを使用して応答を作成します。

リモート シェル サービスの Telnet と SSH は、IPS アプリケーションではありません。これらは直接オペレーティング システムを呼び出します。ユーザが認証されると、IPS の CLI が起動されます。この場合、CLI は特別な形式の execAuthenticateUser 制御トランザクションを送信して、ログインしたユーザの特権レベルを確認します。その後、CLI は、この特権レベル基づいて、使用可能にするコマンドを調整します。

TLS と SSH の信頼関係の管理

IP ネットワーク上の暗号化された通信は、受動的攻撃者が交換されているパケットのみからパケット内のデータの復号化に必要な秘密鍵を発見できないようにすることで、データ プライバシーを実現します。

一方、同じように危険な攻撃として、接続のサーバ エンドを装う偽者に対しても注意を向ける必要があります。すべての暗号化プロトコルは、クライアントがこのような攻撃を防御するための手段を提供しています。IPS は、SSH と TLS の 2 つの暗号化プロトコルをサポートし、センサーが暗号化通信においてクライアントまたはサーバのいずれかとなる場合には、AuthenticationApp が信頼の管理を支援します。

IPS Web サーバと SSH サーバは、暗号化通信のサーバ エンドポイントです。これらは、自身の ID を秘密鍵で保護し、接続してくるクライアントに公開鍵を提供します。TLS では、この公開鍵は他の情報とともに X.509 証明書の中に含まれています。センサーに接続する各リモート システムは、接続の確立時に受け取る公開鍵が目的のものであるかどうかを確認する必要があります。

クライアントは、信頼された公開鍵のリストを維持し、man-in-the-middle 攻撃から自身を保護する必要があります。この信頼が確立される過程は、プロトコルとクライアント ソフトウェアによって異なります。一般に、クライアントは 16 または 20 バイトのフィンガープリントを表示します。クライアントが信頼を確立するように設定するオペレータは、信頼を確立しようとする前に、アウトオブバンド方式を使用して、サーバの鍵のフィンガープリントを知る必要があります。このフィンガープリントが一致する場合は、信頼関係が確立され、それ以降クライアントは自動的にそのサーバに接続し、リモート サーバが偽者でないことを確信することができます。

センサーの鍵のフィンガープリントを表示するには、**show ssh server-key** および **show tls fingerprint** を使用することができます。センサーのコンソールに直接接続しているときにこれらのコマンドの出力を記録しておくことにより、後から信頼関係を確立する際、この情報を使用して、ネットワーク上で確実にセンサーの ID を確認することができます。

たとえば、Microsoft Internet Explorer Web ブラウザを使用して初めてセンサーに接続すると、証明書が信頼できないことを示すセキュリティ警告ダイアログボックスが表示されます。Internet Explorer のユーザ インターフェイスを使用すると、証明書のサムプリントを検査できます。サムプリントは、**show tls fingerprint** コマンドによって表示される SHA1 のフィンガープリントと完全に一致する値です。これを確認した後で、この証明書をブラウザの信頼された CA のリストに追加して、永続的な信頼を確立します。

この信頼を確立するための手順は、各 TLS クライアントで異なります。センサー自体にも、他のセンサーに制御トランザクションを送信し、他の TLS Web サーバからアップグレードやコンフィギュレーション ファイルをダウンロードするために使用される TLS クライアントが含まれています。センサーが通信する TLS サーバの信頼を確立するには、**tls trusted-host** コマンドを使用します。

同様に、センサーには、管理対象ネットワーク デバイスとの通信、アップグレードのダウンロード、およびコンフィギュレーション ファイルやサポート ファイルのリモート ホストへのコピーに使用される SSH クライアントも含まれます。センサーが接続する SSH サーバとの信頼関係を確立するには、**ssh host-key** コマンドを使用します。

TLS の trusted certificate と SSH の既知のホストのリストは、**service trusted-certificates** コマンドと **service ssh-known-hosts** コマンドで管理することができます。

X.509 証明書には、信頼関係のセキュリティを強化するその他の情報も含まれていますが、このような情報が混乱につながる場合もあります。たとえば、X.509 証明書には、証明書を信頼できる有効期間が含まれます。一般に、この期間は、証明書が作成された瞬間を起点とした年数です。X.509 証明書が使用されている瞬間に、それが有効であることを確認するためには、クライアント システムが正確なクロックを維持している必要があります。

また、X.509 証明書は、特定のネットワーク アドレスに結び付けられています。センサーは、このフィールドにセンサーのコマンド / コントロール インターフェイスの IP アドレスを設定します。したがって、センサーのコマンド / コントロールの IP アドレスを変更すると、サーバの X.509 証明書は再生成されます。ネットワーク上でこの古い証明書を信頼していたすべてのクライアントが新しい IP アドレスでセンサーを探し、新しい証明書を信頼するように、再設定を行う必要があります。

AuthenticationApp で SSH の既知のホストと TLS の trusted certificate の各サービスを使用することにより、センサーをより高いセキュリティ レベルで運用することができます。

Web サーバ

Web サーバは、RDEP2 サポートを提供します。RDEP2 を使用すると、センサーでセキュリティ イベントの報告、IDIOM トランザクションの受信、およびサーバの IP ログの記録が可能になります。

Web サーバは、HTTP 1.0 および 1.1 をサポートします。Web サーバとの通信には、多くの場合、パスワードなど、攻撃者に盗聴された場合にシステムのセキュリティを大きく危険にさらすことになる機密情報が伴います。このような理由から、センサーは TLS がイネーブルな状態で出荷されます。TLS プロトコルは、SSL と互換性のある暗号化プロトコルです。

SensorApp

この項では、SensorApp について説明します。取り上げる事項は次のとおりです。

機能とコンポーネント

SensorApp は、パケットの取り込みと分析を実行します。SensorApp では、シグニチャを使用してポリシー違反が検出され、違反に関する情報がアラートの形式でイベント ストアに転送されます。

パケットは、プロセッサのパイプラインを通過します。このパイプラインは、設計者がセンサー上のネットワーク インターフェイスからパケットを収集するように設計します。

SensorApp は次のプロセッサをサポートします。

- Time Processor（TP）

  このプロセッサは、タイムスライス カレンダに格納されたイベントを処理します。主なタスクは、失効したデータベース エントリを廃棄し、時間に依存する統計を計算することです。
- Deny Filters Processor（DFP）

  このプロセッサは、拒否攻撃者機能を処理します。拒否された送信元 IP アドレスのリストを管理します。リスト内の各エントリは、グローバルな拒否タイマーに基づいて有効期限切れになります。拒否タイマーは、仮想センサーのコンフィギュレーション内で設定できます。
- Signature Event Action Processor（SEAP）

  このプロセッサは、イベント アクションを処理します。次のイベント アクションがサポートされます。

  — TCP フローのリセット
  — IP ログ
  — パケットの拒否
  — フローの拒否
  — 攻撃者の拒否
  — アラート
  — ホストのブロック
  — 接続のブロック
  — SNMP トラップの生成
  — トリガー パケットの取り込み

  イベント アクションは、イベントの RR しきい値と関連付けできます。アクションが実行されるには、このしきい値を超える必要があります。
- Statistics Processor（SP）

  このプロセッサは、パケットのカウントやパケットの到達率など、システム統計情報を継続して追跡します。
- Layer 2 Processor（L2P）

  このプロセッサは、レイヤ 2 関連イベントを処理します。また、異常形式のパケットを識別し、処理パスから削除します。アラート、取り込みパケット、拒否パケットなど、形式の正しくないパケットを検出するための実行可能なイベントを設定することができます。レイヤ 2 プロセッサは、ユーザが設定したポリシーのために拒否されたパケットに関する統計情報を更新します。

- Database Processor（DBP）

  このプロセッサは、シグニチャの状態とフロー データベースを管理します。

- Fragment Reassembly Processor（FRP）

  このプロセッサは、フラグメント化された IP データグラムを再構成します。センサーがインライン モードの場合、IP フラグメントの正規化も処理します。

- Stream Reassembly Processor（SRP）

  このプロセッサは、さまざまなストリームベース インスペクタでパケットが適切な順序で到着するよう、TCP ストリームを並べ替えます。また、TCP ストリームの正規化も行います。Normalizer エンジンを使用すると、アラートおよび拒否アクションを有効または無効にできます。

  TCP SRP の正規化エンジンはホールドダウン タイマーを備え、これによって再構成イベントの後でストリームの状態を再構築することができます。ユーザがタイマーを設定することはできません。ホールドダウン間隔の間、システムは、ストリームの状態を、システムを通過するストリーム内の最初のパケットと同期化します。ホールドダウンの有効期限が切れると、sensorApp は設定されたポリシーを実施します。このポリシーで、3 ウェイ ハンドシェイクを使用してオープンされていないストリームの拒否が要求される場合、ホールドダウン期間中に静止していた確立済みストリームは転送されず、タイムアウトが許可されます。ホールドダウン期間中に同期化されたストリームは、続行を許可されます。

- Signature Analysis Processor（SAP）

  このプロセッサは、ストリーム ベースでなく、処理中のパケット用に設定されているインスペクタにパケットを送信します。

- Slave Dispatch Processor（SDP）

  デュアル CPU システムにのみ存在するプロセスです。

一部のプロセッサは、シグニチャの分析を実行するためにインスペクタを呼び出します。すべてのインスペクタは、アラーム チャネルを呼び出して、必要に応じてアラートを生成することができます。

また、SensorApp は次のユニットもサポートします。

- 分析エンジン

  分析エンジンは、センサーのコンフィギュレーションを処理します。インターフェイスをマップし、またシグニチャおよびアラーム チャネル ポリシーを設定済みインターフェイスにマップします。

- アラーム チャネル

  アラーム チャネルは、インスペクタによって生成されたすべてのシグニチャ イベントを処理します。主な機能は、渡された各イベントのアラートを生成することです。

パケット フロー

パケットは、NIC によって受信され、IPS 共有ドライバにより、ユーザによってマッピングされたカーネル内のメモリ スペースに配置されます。パケットの前には、IPS ヘッダーが付加されます。また、各パケットには、パケットが SEAP に到達したときにそのパケットを通過させるか拒否するかを示すフィールドも含まれます。

ユーザによってマッピングされた共有カーネルのパケット バッファからパケットが取り出され、センサーのモデルに適したプロセッサを実装する処理機能を呼び出します。順序は、次のようになります。

- シングル プロセッサの実行

  TP --> L2P --> DFP --> FRP --> SP --> DBP --> SAP --> SRP --> EAP

- デュアル プロセッサの実行

  実行スレッド 1 TP --> L2P --> DFP --> FRP --> SP --> DBP --> SAP --> SDP --> | 実行スレッド 2 DBP --> SRP --> EAP

SEAP

SEAP は、アラーム チャネル内のシグニチャ イベントから、SEAO、SEAF の処理を経由して SEAH で処理されるまでのデータ フローを調整します。これは次のコンポーネントから構成されます。

- アラーム チャネル

  Sensor App 検査パスからシグニチャ イベント処理へ向かうシグニチャ イベントと通信するエリアを表す単位。
- シグニチャ イベント アクション オーバーライド（SEAO）

  RR 値に基づいて、アクションを追加します。SEAO は、設定済みの RR しきい値の範囲に該当するすべてのシグニチャに適用されます。各 SEAO は独立しており、各アクション タイプには別個の値が設定されています。詳細については、P.6-9 の「リスク評価の計算」を参照してください。
- シグニチャ イベント アクション フィルタ（SEAF）

  シグニチャ イベントのシグニチャ ID、アドレス、および RR に基づいてアクションを削除します。SEAF へ入力するのは、SEAO によって追加される可能性のあるアクションを持つシグニチャ イベントです。

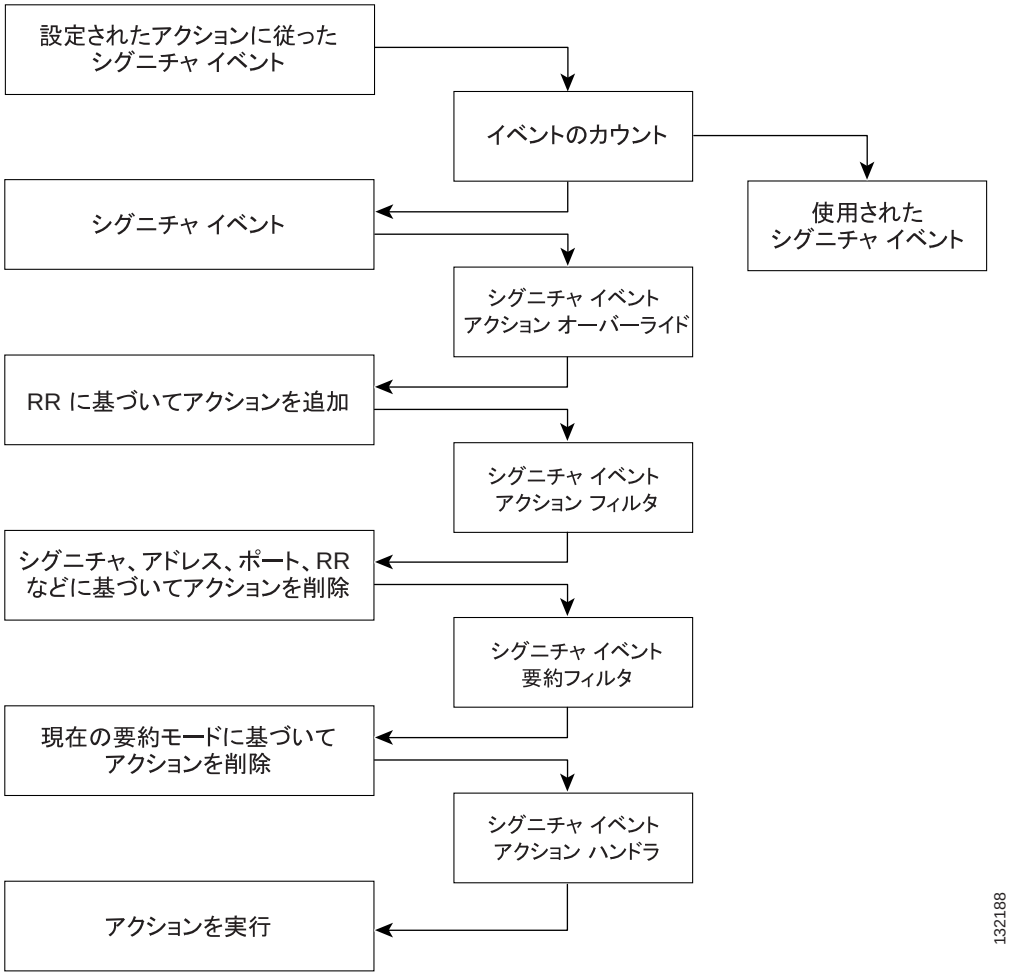
  （注） SEAF が実行できるのはアクションの削除だけであり、新規アクションの追加はできません。

  SEAF には、次のパラメータが適用されます。
  - シグニチャ ID
  - サブシグニチャ ID
  - 攻撃者のアドレス
  - 攻撃者のポート
  - 被害先のアドレス
  - 被害先のポート
  - RR しきい値の範囲
  - 削除するアクション
  - シーケンス識別子（オプション）
  - ストップ ビットまたは継続ビット
  - アクション フィルタ行をイネーブルにするビット
- シグニチャ イベント アクション ハンドラ（SEAH）

  要求されたアクションを実行します。SEAH から出力されるのは、実行中のアクションだけでなく、イベント ストアに書き込まれる <evIdsAlert> である可能性があります。

図 A-4 に、SEAP を使用したシグニチャ イベントの論理フローと、このイベントに関してアクションで実行される動作を示します。これは、アラーム チャネルで受信された設定済みアクションを持つシグニチャ イベントで開始され、シグニチャ イベントが SEAP の機能コンポーネントを通過するときに上から下へ流れます。

図 A-4　　SEAP を通過するシグニチャ イベント



新機能

SensorApp には、次の新機能が含まれます。

- パケットのインライン処理

  センサーがデータ パスのパケットを処理するときには、ポリシー コンフィギュレーションで明示的に拒否されていない限り、すべてのパケットは変更されずに転送されます。TCP の正規化により、一部のパケットが適切なカバレッジを確保するために遅延することがあります。ポリシー違反が発生すると、SensorApp はアクションの設定を許可します。パケットの拒否、フローの拒否、攻撃者の拒否など、追加のアクションがインライン モードで使用できるようになっています。

  IPS にとって不明であるか、対象でないすべてのパケットは、分析されずにペアのインターフェイスに転送されます。すべてのブリッジングおよびルーティング プロトコルは、ポリシー違反が原因で発生する可能性のある拒否の場合のみ、転送されます。インライン（または混合モード）のデータ処理に使用されるインターフェイスには、IP スタックは関連付けられません。混合モードにおける 802.1q パケットの現在のサポートが、インライン モードまで拡張されます。
- 拡張コンフィギュレーション
- インライン処理でのデータフローのバックアップ

- ホールドダウン タイマー

  SensorApp は、最初に起動したときに、その時点で存在するすべてのフローの状態情報を構築する必要があります。ホールドダウン タイマーは、SensorApp がこの状態情報を構築している間にパケットを拒否しないようにします。ホールドダウン タイマーの期間中も、SensorApp は十分な情報があればポリシーを実施します。

- IP の正規化

  IP データグラムのフラグメンテーションは、意図的であってもなくても不正利用の隠蔽につながり、その検知を困難または不可能にします。フラグメント化は、ファイアウォールやルータにあるようなアクセス コントロール ポリシーを回避するために使用することもできます。また、さまざまなオペレーティング システムがさまざまなメソッドを使用して、フラグメント化されたデータグラムをキューに入れたり送出したりします。エンド ホストがデータグラムを再構成する際に使用可能なすべての方法をセンサーでチェックする必要がある場合、センサーは DoS 攻撃（サービス拒絶攻撃）に対して脆弱になります。フラグメント化したすべてのデータグラムをインラインで再構成して、完成したデータグラムのみを転送し、必要に応じてデータグラムを再度フラグメント化することが、この問題の解決策です。IP フラグメント化の正規化の装置は、この機能を実行します。

- TCP の正規化

  意図的または意図しない TCP セッション セグメンテーションによって、いくつかの攻撃クラスが非表示になることもあります。ポリシーが false positive や false negative なしに実施されるようにするには、2 つの TCP エンドポイントの状態が追跡され、実際のホスト エンドポイントによって処理されたデータだけが渡される必要があります。TCP ストリームで重複が発生する可能性がありますが、TCP セグメントの再転送以外は、非常にまれです。TCP セッションでの上書きは発生しないはずです。上書きが発生する場合は、誰かがセキュリティ ポリシーを意図的に回避しようとしているか、TCP スタックの実装が破損しています。センサーが TCP プロキシとして動作していない限り、両方のエンドポイントの状態について完全な情報を保持することはできません。センサーが TCP プロキシとして動作する代わりに、セグメントが適切に並べられ、正規化エンジンによって回避や攻撃に関係する異常なパケットが検索されます。

- イベントの RR

  イベントの RR に、潜在的に悪意のあるアクションの検出だけでなく、次の追加情報が組み込まれています。

  - 攻撃が成功した場合の攻撃の重大度
  - シグニチャの忠実度
  - ターゲット ホストに関する潜在的な攻撃の関連性
  - ターゲット ホストの全体の価値

  イベントの RR はシステムからの false positive の減少に役立ち、アラーム発生原因のより適切な制御を可能にします。

- イベント アクション フィルタとイベント アクションの処理

  4.*x* のイベント フィルタは、すべてのアクションをフィルタリングしていました。5.*x* のイベント フィルタでは、イベントを個別に処理することができます。アラームの送信もアクションと見なされるようになり、その他のアクションと同様にフィルタリングおよび設定することができます。

- SensorApp と TCPdump による同時取得に対応したドライバ サポート

  データ インターフェイス用ドライバは、SensorApp と TCPdump またはその他の libpcap ベース リーダーによるインターフェイスの同時使用をサポートします。

CLI

CLI は、Telnet、SSH、シリアル インターフェイスなど、ノードへの直接アクセスすべてを提供するセンサーのユーザ インターフェイスです。センサーのアプリケーションは、CLI を使用して設定します。基盤となる OS には、サービス ロールを使用すると直接アクセスできます。

この項では、次のトピックについて説明します。

ユーザ ロール

IPS 5.1 の CLI では、複数ユーザの同時ログインが許可されます。ローカル センサーからユーザを作成および削除することができます。一度に変更できるユーザ アカウントは 1 つだけです。各ユーザは、ユーザが何を変更でき何を変更できないかを制御するロールに関連付けられます。

CLI は、管理者、オペレータ、ビューア、およびサービスの 4 つのユーザ ロールをサポートします。特権レベルはロールごとに異なるので、メニューや使用できるコマンドもロールによって異なります。

- **管理者**：このユーザ ロールは、最高レベルの特権を持ちます。管理者は、無制限の表示アクセス権を持ち、次の機能を実行できます。
  - ユーザの追加とパスワードの割り当て
  - 物理インターフェイスおよび仮想センサーの制御のイネーブル化とディセーブル化
  - 仮想センサーへの物理センシング インターフェイスの割り当て
  - 設定または表示エージェントとしてセンサーへ接続することを許可されたホストのリストの変更
  - センサー アドレス コンフィギュレーションの変更
  - シグニチャの調整
  - 仮想センサーへのコンフィギュレーションの割り当て
  - ルータの管理
- **オペレータ**：このユーザ ロールは、2 番目に高いレベルの特権を持ちます。オペレータは、無制限の表示アクセス権を持ち、次の機能を実行できます。
  - パスワードの変更
  - シグニチャの調整
  - ルータの管理
  - 仮想センサーへのコンフィギュレーションの割り当て
- **ビューア**：このユーザ　ロールは、最低レベルの特権を持ちます。ビューアは、コンフィギュレーションおよびイベント データの表示と、パスワードの変更を実行できます。

ヒント　アプリケーションのモニタリングでは、センサーに対するビューア アクセス権のみが必要とされます。CLI を使用してビューア特権を持つユーザ アカウントをセットアップし、このアカウントを使用してセンサーに接続するようイベント ビューアを設定することができます。

- **サービス**：このユーザ ロールは、CLI への直接アクセス権を持ちません。サービス アカウント ユーザは、バッシュ シェルに直接ログインします。このアカウントは、サポートとトラブルシューティングにのみ使用します。不正な変更はサポートされません。不正に変更が行われた場合、適切な操作を保証するためデバイスのイメージを再作成する必要があります。サービス ロールを持つユーザは 1 人しか作成できません。

  サービス アカウントにログインすると、次の警告が表示されます。

```
****************************** WARNING ****************************************
UNAUTHORIZED ACCESS TO THIS NETWORK DEVICE IS PROHIBITED.
This account is intended to be used for support and troubleshooting purposes only.
Unauthorized modifications are not supported and will require this device to be
re-imaged to guarantee proper operation.
*******************************************************************************
```

（注）　サービス ロールは、必要に応じて CLI をバイパスできる特殊なロールです。サービス アカウントを編集できるのは、管理者特権を持つユーザだけです。

サービス アカウント

サービス アカウントは、TAC が CLI シェルではなく、ネイティブ オペレーティング システムのシェルに直接ログインできるようにする、サポートおよびトラブルシューティング用ツールです。デフォルトでは、このアカウントはセンサー上に存在しません。TAC がセンサーのトラブルシューティングに使用できるように、このアカウントを作成する必要があります。サービス アカウントの作成手順については、P.4-16 の「サービス アカウントの作成」を参照してください。

サービス アカウントはセンサーごとに 1 つのみ許可されます。また、サービス ロールを許可されるアカウントは 1 つのみです。サービス アカウントのパスワードが設定またはリセットされると、root アカウントのパスワードが同じパスワードに設定されます。これにより、サービス アカウントのユーザは同じパスワードを使用して root に su することができます。サービス アカウントが削除されると、root アカウントのパスワードはロックされます。

サービス アカウントは、コンフィギュレーションに使用することを目的としたものではありません。サービス アカウントでは、TAC の指示の下でセンサーを変更する場合にのみ使用することができます。サービス アカウントを使用したオペテーティング システムへのその他のサービスの追加またはそのようなサービスの実行は、他の IPS サービスの適切なパフォーマンスと機能に影響を与えるため、シスコシステムズはこれをサポートしません。TAC は、その他のサービスが追加されたセンサーをサポートしません。

サービス アカウントへのログインは、ログ ファイル /var/log/.tac を確認することによって追跡できます。このファイルは、サービス アカウントのログインがあるとレコードが更新されます。

（注）　IPS 5.1 には、CLI または IDM から使用可能ないくつかのトラブルシューティング機能が組み込まれています。ほとんどのトラブルシューティングの場面でサービス アカウントは不要です。きわめてまれな問題では、TAC の指示の下にサービス アカウントを作成してトラブルシューティングを実行しなければならない場合があります。サービス アカウントを使用すると、CLI に組み込まれた保護をバイパスし、通常はディセーブルになっている root 特権でセンサーにアクセスすることができます。特別な理由で必要となる場合を除いては、サービス アカウントを作成しないことをお勧めします。サービス アカウントは、不要になったら削除してください。

CLI の動作

IPS CLI を使用する際は、これらのヒントに従ってください。

プロンプト

- CLI コマンドに対して表示されるプロンプトは変更できません。
- システムから質問が表示され、ユーザの入力待ちになる場合は、対話型のプロンプトになります。デフォルト入力はカッコ [ ] 内に表示されます。デフォルト入力を受け入れるには、**Enter** キーを押します。

ヘルプ

- コマンドのヘルプを表示するには、コマンドの後ろに **?** を入力します。

  次の例では、? の機能を示します。

```
sensor# configure ?
terminal     Configure from the terminal
sensor# configure
```

  （注）　プロンプトがヘルプの表示から戻ると、前に入力されたコマンドが ? なしで表示されます。

- 不完全なトークンの後に **?** を入力して、コマンドを完成させる有効なトークンを表示することもできます。トークンと ? の間にスペースがある場合は、コマンドがあいまいであることを示すエラーが表示されます。

```
sensor# show c ?
% Ambiguous command : "show c"
```

  スペースなしでトークンを入力すると、コマンドを完成させるために使用できるトークンの選択肢が表示されます（ヘルプの説明なしで）。

```
sensor# show c?
clock configuration
sensor# show c
```

- 現在のモードで使用できるコマンドだけがヘルプによって表示されます。

タブ補完

- 現在のモードで使用できるコマンドだけがタブ補完およびヘルプによって表示されます。
- コマンドの完全な構文がわからない場合は、コマンドの一部を入力して **Tab** キーを押し、コマンドを完成させることができます。
- タブ補完で複数のコマンドが一致する場合は、何も表示されません。

再呼び出し

- モードで入力されたコマンドを再呼び出しするには、↑ キーまたは ↓ キーを使用するか、あるいは **Ctrl+P** キーまたは **Ctrl+N** キーを押します。

  （注）　ヘルプおよびタブ補完の要求は、再呼び出しリストでは報告されません。

- ブランクのプロンプトは、再呼び出しリストの最後を示します。

大文字と小文字の区別

- CLI では、大文字と小文字は区別されませんが、テキストは入力された大文字と小文字の状態を保ってエコー バックされます。たとえば、次のように入力します。

  ```
  sensor# CONF
  ```

  Tab キーを押すと、次のように表示されます。

  ```
  sensor# CONFigure
  ```

表示オプション

- –More– は、ターミナル出力が割り当てられた表示スペースを超えていることを示しています。残りの出力を表示する場合は、**Space** キーを押して出力の次のページを表示するか、**Enter** キーを押して出力を 1 行ずつ表示します。
- 現在の行の内容をクリアしてブランクのコマンドラインに戻るには、**Ctrl+C** キーを押します。

通信

この項では、IPS 5.1 で使用される通信プロトコルについて説明します。取り上げる事項は次のとおりです。

IDAPI

IPS アプリケーションは、IDAPI というプロセス間通信 API を使用して内部通信を処理します。IDAPI はイベント データを読み書きし、制御トランザクションのメカニズムを提供します。IDAPI は、すべてのアプリケーションの通信に使用されるインターフェイスです。

SensorApp は、それ自体のインターフェイス上でネットワーク トラフィックを取得し、分析します。シグニチャが一致すると、SensorApp はアラートを生成し、アラートはイベント ストアに保存されます。ブロッキング応答アクションを実行するようにシグニチャが設定されている場合、SensorApp はブロック イベントを生成します。このイベントもイベント ストアに保存されます。

図 A-5 は、IDAPI インターフェイスを示しています。

図 A-5　IDAPI

各アプリケーションは、IDAPI に登録してイベントと制御トランザクションを送受信します。IDAPI は、次のサービスを提供します。

- 制御トランザクション
  - 制御トランザクションの開始
  - 着信制御トランザクションの待機
  - 制御トランザクションへの応答
- IPS イベント
  - リモート IPS イベント（受信時にイベント ストアに保存される）への登録
  - イベント ストアからの IPS イベントの読み取り
  - イベント ストアへの IPS イベントの書き込み

IDAPI は、アトミックなデータ アクセスを保証するために必要な同期化メカニズムを提供します。

RDEP2

外部通信では、RDEP2 が使用されます。RDEP2 は、IPS クライアントと IPS サーバ間で、IPS イベント、IP ログ、コンフィギュレーション、および制御メッセージの交換に使用される、アプリケーションレベルの通信プロトコルです。RDEP2 の通信は、要求メッセージと応答メッセージで構成されます。RDEP2 クライアントは、RDEP2 サーバに対して要求メッセージを開始します。RDEP2 サーバは、要求メッセージに対して応答メッセージで応答します。

RDEP2 では、イベント、IP ログ、トランザクションの 3 つのクラスの要求 / 応答メッセージが定義されています。イベント メッセージには、IPS アラート、ステータス、エラーの各メッセージがあります。クライアントは、IP ログ要求を使用して、サーバから IP ログ データを取得します。トランザクション メッセージは、IPS サーバの設定と制御に使用されます。

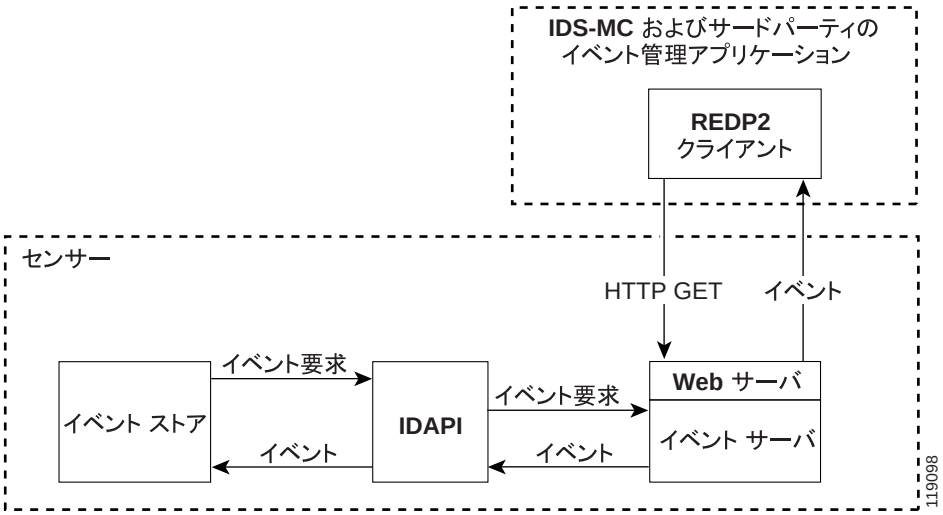
RDEP2 は、業界標準の HTTP、TLS および SSL、および XML を使用して、RDEP2 エージェント間に標準化されたインターフェイスを提供します。RDEP2 プロトコルは HTTP 1.1 プロトコルのサブセットです。すべての RDEP2 メッセージは、有効な HTTP 1.1 メッセージです。RDEP2 は、HTTP のメッセージ形式とメッセージ交換プロトコルを使用して、RDEP2 エージェント間でメッセージを交換します。

ネットワーク経由でセンサーへのアクセスが許可されるホストを指定するには、IPS マネージャを使用します。センサーは、1 ～ 10 の RDEP2 クライアントからの接続を同時に受け入れます。クライアントは、時間範囲、イベントのタイプ（アラート、エラー、またはステータス メッセージ）、およびイベントのレベル（アラートの場合、high、medium、low、informational、エラーの場合、high、medium、low）別に、データを選択して取得します。イベントは、クエリー（1 回のバルク取得）または登録（リアルタイムの永続的な接続）、またはその両方によって取得されます。通信の安全は、TLS または SSL で保護されます。

（注）　イベントの取得に関しては、RDEP2 が取得用の新しい標準ですが、センサーは下位互換で、RDEP にも対応します。IPS 5.1 のイベントの取得とコンフィギュレーション変更の送信には、RDEP2 を使用することをお勧めします。

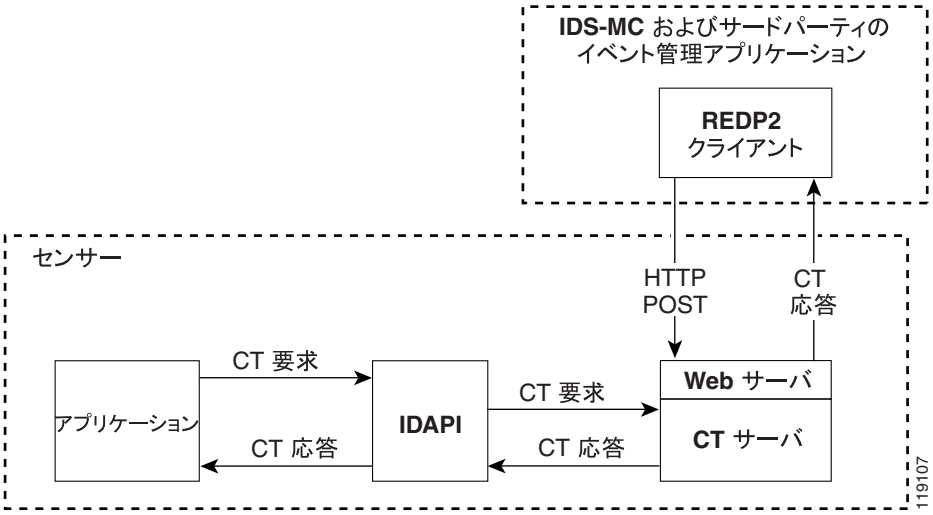
リモート アプリケーションは、RDEP2 を使用してセンサーからイベントを取得します。リモート クライアントが RDEP2 のイベント要求をセンサーの Web サーバに送信すると、Web サーバがそれをイベント サーバに渡します。イベント サーバは IDAPI を使用してイベント ストアをクエリーし、結果を戻します。図 A-6 に、RDEP2 を使用してセンサーからイベントを取得するリモート アプリケーションを示します。

図 A-6　RDEP2 を使用したイベントの取得



リモート アプリケーションは、RDEP2 を使用してセンサーにコマンドを送信します。リモート クライアントが RDEP2 の制御トランザクションをセンサーの Web サーバに送信すると、Web サーバがそれを制御トランザクション サーバに渡します。制御トランザクション サーバは IDAPI を経由して該当するアプリケーションに制御トランザクションを渡し、アプリケーションの応答を待ち、その後結果を返します。図 A-7 に、RDEP2 を使用してコマンドをセンサーに送信するリモート アプリケーションを示します。

図 A-7　RDEP2 を使用したコマンドの送信



IDIOM

IDIOM は、IPS によって報告されるイベント メッセージ、および侵入検知システムの設定と制御に使用される操作メッセージを定義するデータ形式の規格です。これらのメッセージは、IDIOM XML スキーマに準拠した XML 文書で構成されます。

IDIOM は、イベントと制御トランザクションの 2 つのタイプのインタラクションをサポートします。イベント インタラクションは、アラートなどの IPS イベントの交換に使用されます。IDIOM は、イベント インタラクションに、イベント メッセージとエラー メッセージの 2 つのタイプのメッセージを使用します。制御トランザクションは、1 つのホストが別のホストでアクションを開始したり、別のホストの状態の変更や読み取りを実行したりする手段となります。制御トランザクションでは、要求、応答、コンフィギュレーション、エラーの 4 つのタイプの IDIOM メッセージが使用されます。同じホスト内のアプリケーション間で通信されるイベントと制御トランザクションは、ローカル イベントおよびローカル制御トランザクションと呼ばれ、集合的にローカル IDIOM メッセージと呼ばれます。異なるホスト間で RDEP2 プロトコルを使用して通信されるイベントと制御トランザクションは、リモート イベントおよびリモート制御トランザクションと呼ばれ、集合的にリモート IDIOM メッセージと呼ばれます。

（注）　IDIOM の大部分は、IDCONF、SDEE、および CIDEE によって置き換えられています。

IDCONF

IPS 5.1 は、XML 文書を使用して自身のコンフィギュレーションを管理します。IDCONF は、IPS 5.1 の制御トランザクションも含めた XML スキーマを指定します。IDCONF スキーマは、コンフィギュレーション文書の内容を指定するのではなく、コンフィギュレーション文書作成の基になるフレームワークと構成単位を指定します。このスキーマは、特定の機能でサポートされるアトリビュートを使用することにより、それを設定することができないプラットフォームや機能の場合は IPS マネージャおよび CLI がその機能を無視できるメカニズムを提供します。

IDCONF メッセージは IDIOM の要求および応答メッセージ内にラップされ、RDEP2 によって交換されます。

IDCONF の例を次に示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<request xmlns="http://www.cisco.com/cids/idiom" schemaVersion="2.00">
  <editConfigDelta xmlns="http://www.cisco.com/cids/idconf">
    <component name="userAccount">
      <config typedefsVersion="2004-03-01" xmlns="http://www.cisco.com/cids/idconf">
        <struct>
           <map name="user-accounts‟ editOp=‟merge”>
             <mapEntry>
               <key>
                <var name="name">cisco</var>
              </key>
               <struct>
                <struct name="credentials">
                  <var name="role">administrator</var>
                </struct>
               </struct>
            </mapEntry>
          </map>
        </struct>
      </config>
    </component>
  </editDefaultConfig>
</request>
```

SDEE

IPS は、侵入アラートやステータス イベントなどのさまざまなタイプのイベントを生成します。IPS は、専用の RDEP2 を使用して、管理アプリケーションなどのクライアントにイベントを伝達します。また、シスコは、IPS 業界の主要プロトコルである SDEE も開発しました。SDEE は、セキュリティ デバイスのイベントを伝達するための製品に依存しない規格です。SDEE は、現行バージョンの RDEP2 を強化して、各種のセキュリティ デバイスによって生成されるイベントの伝達に必要な拡張機能を追加したものです。

SDEE を使用してクライアントにイベントを伝達するシステムは、SDEE プロバイダーと呼ばれます。SDEE では、イベントを、HTTP プロトコル、または SSL および TLS を介した HTTP プロトコルを使用して転送するように指定できます。HTTP または HTTPS を使用する場合は、SDEE クライアントが HTTP 要求の発信側となり、SDEE プロバイダーは HTTP サーバとして動作します。

IPS には、HTTP または HTTPS 要求を処理する Web サーバが含まれています。Web サーバは、実行時にロード可能なサーブレットを使用して、各種の HTTP 要求を処理します。各サーブレットは、そのサーブレットに関連付けられた URL に送られる HTTP 要求を処理します。SDEE サーバは、Web サーバ サーブレットとして実装されます。

SDEE サーバは、許可された要求のみを処理します。要求は、クライアントの ID を認証し、クライアントの特権レベルを確認するために Web サーバから発信されている場合に許可されます。

CIDEE

CIDEE は、Cisco IPS が使用する SDEE への拡張を指定します。CIDEE 規格は、IPS がサポートする可能性のあるすべての拡張を指定しています。特定のシステムでは、CIDEE の拡張のサブセットを実装する場合もあります。ただし、必須と指定された拡張は、すべてのシステムでサポートする必要があります。

CIDEE では、IPS 固有のセキュリティ デバイス イベント、および、SDEE の <evIdsAlert> 要素に対する IPS の拡張が指定されています。

CIDEE は、次のイベントをサポートします。

- <evError>：エラー イベント

  CIDEE プロバイダーがエラーまたは警告状態を検出すると生成されます。<evError> イベントには、エラー コードとテキストによるエラーの説明が含まれます。
- <evStatus>：ステータス メッセージ イベント

  ホストで潜在的に注意が必要な何らかの状態が発生したことを示すために、CIDEE プロバイダーによって生成されます。ステータス イベントでは、異なるタイプのステータス メッセージが報告される可能性があります。イベントごとに 1 つのメッセージが生成されます。各タイプのステータス メッセージには、そのステータス メッセージで説明されている発生タイプに固有のデータ要素のセットが含まれます。ステータス メッセージの多くに含まれる情報は、監査の目的で役立つ場合があります。エラーと警告はステータス情報とは見なされず、<evStatus> ではなく <evError> を使用して報告されます。
- <evShunRqst>：ブロック要求イベント

  ネットワークのブロッキングを処理するサービスがブロック アクションを開始することを示すために生成されます。

CDIEE の拡張イベントの例を次に示します。

```
<sd:events xmlns:cid="http://www.cisco.com/cids/2004/04/cidee"
xmlns:sd="http://example.org/2003/08/sdee">
 <sd:evIdsAlert eventId="1042648730045587005" vendor="Cisco" severity="medium">
  <sd:originator>
    <sd:hostId>Beta4Sensor1</sd:hostId>
    <cid:appName>sensorApp</cid:appName>
    <cid:appInstanceId>8971</cid:appInstanceId>
  </sd:originator>
  <sd:time offset="0" timeZone="UTC">1043238671706378000</sd:time>
  <sd:signature description="IOS Udp Bomb" id="4600" cid:version="S37">
    <cid:subsigId>0</cid:subsigId>
  </sd:signature>
  …
```

IPS 5.1 のファイル構造

IPS 5.1 のディレクトリ構造は次のようになっています。

- /usr/cids/idsRoot：メインのインストール ディレクトリ。
- /usr/cids/idsRoot/shared：システムのリカバリ中に使用されるファイルが保存されます。
- /usr/cids/idsRoot/var：センサーの実行中にダイナミックに作成されたファイルが保存されます。
- /usr/cids/idsRoot/var/updates：アップデート インストールのファイルとログが保存されます。
- /usr/cids/idsRoot/var/virtualSensor：SensorApp で正規表現の分析に使用されるファイルが保存されます。
- /usr/cids/idsRoot/var/eventStore：イベント ストア アプリケーションが格納されます。
- /usr/cids/idsRoot/var/core：システムがクラッシュしたときに作成されるコア ファイルが保存されます。
- /usr/cids/idsRoot/var/iplogs：iplog ファイルのデータが保存されます。
- /usr/cids/idsRoot/bin：バイナリ実行可能ファイルが格納されます。
- /usr/cids/idsRoot/bin/authentication：認証アプリケーションが格納されます。
- /usr/cids/idsRoot/bin/cidDump：テクニカル サポート用のデータを収集するスクリプトが格納されます。
- /usr/cids/idsRoot/bin/cidwebserver：Web サーバ アプリケーションが格納されます。
- /usr/cids/idsRoot/bin/cidcli：CLI アプリケーションが格納されます。
- /usr/cids/idsRoot/bin/nac：ARC アプリケーションが格納されます。
- /usr/cids/idsRoot/bin/logApp：ロガー アプリケーションが格納されます。
- /usr/cids/idsRoot/bin/mainApp：メイン アプリケーションが格納されます。
- /usr/cids/idsRoot/bin/sensorApp：センサー アプリケーションが格納されます。
- /usr/cids/idsRoot/bin/falcondump：IDS-4250-XL および IDSM-2 のセンシング ポートでパケット ダンプを取得するためのアプリケーションが格納されます。
- /usr/cids/idsRoot/etc：センサーのコンフィギュレーション ファイルが保存されます。
- /usr/cids/idsRoot/htdocs：Web サーバ用の IDM ファイルが格納されます。
- /usr/cids/idsRoot/lib：センサー アプリケーションのライブラリ ファイルが格納されます。
- /usr/cids/idsRoot/log：デバッグ用ログ ファイルが格納されます。
- /usr/cids/idsRoot/tmp：センサーの実行時に作成される一時ファイルが保存されます。

IPS 5.1 アプリケーションの要約

表 A-2 に、IPS を構成するアプリケーションを要約して示します。

表 A-2　アプリケーションの要約

アプリケーション	説明
AuthenticationApp	IP アドレス、パスワード、およびデジタル証明書に基づいて、ユーザを認可および認証します。
CLI	コマンドライン入力を受け入れ、IDAPI を使用してローカル コンフィギュレーションを変更します。
イベント サーバ[1]	リモート クライアントからイベントに対する RDEP2 要求を受け入れます。
MainApp	コンフィギュレーションを読み取ってアプリケーションを起動します。また、アプリケーションの起動と停止、ノードのリブート、およびソフトウェア アップグレードを処理します。
InterfaceApp	バイパスと物理的な設定を処理し、ペアのインターフェイスを定義します。物理的な設定は、速度、二重、および管理状態です。
LogApp	アプリケーションのすべてのログ メッセージをログ ファイルに書き込み、エラー メッセージをイベント ストアに書き込みます。
Attack Response Controller	ARC は各センサーで実行されます。各 ARC は、ローカル イベント ストアのネットワーク アクセス イベントに登録します。ARC のコンフィギュレーションには、各センサー、およびそのセンサーのローカル ARC で制御されるネットワーク アクセス デバイスのリストが含まれます。マスター ブロッキング センサーにネットワーク アクセス イベントを送信するように設定されている場合、ARC は該当デバイスを制御するリモート ARC に対してネットワーク アクセス制御トランザクションを開始します。また、このようなネットワーク アクセス アクションの制御トランザクションは、IPS マネージャでも随時発生するネットワーク アクセス アクションの発行に使用されます。
NotificationApp	アラート、ステータス、およびエラー イベントによって起動された場合に、SNMP トラップを送信します。NotificationApp はパブリック ドメイン SNMP エージェントを使用します。SNMP GET では、センサーの一般的な健全性情報を取得できます。
SensorApp	監視対象ネットワークのトラフィックを取り込んで分析し、侵入イベントやネットワーク アクセス イベントを生成します。IP ロギング制御トランザクションに応答します。IP ロギング制御トランザクションは、ロギングのオン / オフの切り替えや、IP ログ ファイルの送信と削除を実行します。
制御トランザクション サーバ[2]	リモート RDEP2 クライアントからの制御トランザクションを受入れてローカルの制御トランザクションを開始し、リモート クライアントに応答を返します。
制御トランザクション ソース[3]	リモート アプリケーションに送られる制御トランザクションを待ち、RDEP2 を使用してその制御トランザクションをリモート ノードに転送し、発信側に応答を返します。

表 A-2　アプリケーションの要約（続き）

アプリケーション	説明
IDM	HTML の IPS 管理インターフェイスを提供する Java アプレットです。
Web サーバ	リモート HTTP クライアント要求を待ち、適切なサーブレットアプリケーションを呼び出します。

1. これは Web サーバ サーブレットです。
2. これは Web サーバ サーブレットです。
3. これは、リモート制御トランザクションのプロキシです。