CHAPTER 4

Detector モジュールの設定

この章では、Cisco Traffic Anomaly Detector モジュール（Detector モジュール）のサービスの設定方法について説明します。

この章には、Detector モジュールの関連製品である Cisco Guard（Guard）についての記述があります。Guard は、DDoS 攻撃（分散型サービス拒絶攻撃）を検出して軽減するデバイスです。Guard は、ゾーン トラフィックが通過する際に攻撃トラフィックをドロップして正常なトラフィックをネットワークに戻し、ゾーン トラフィックをクリーンにします。Detector モジュールは、ゾーンが攻撃を受けていると判断したときに、Guard の攻撃軽減サービスをアクティブにできます。また、Detector モジュールは Guard とゾーン設定を同期させることができます。Guard の詳細については、『*Cisco Anomaly Guard Module Configuration Guide*』または『*Cisco Guard Configuration Guide*』を参照してください。

この章は、次の項で構成されています。

- Detector モジュールのサービスのアクティブ化
- AAA を使用したアクセス コントロールの設定
- Guard との通信の確立
- SSH 鍵の管理
- SFTP 接続および SCP 接続用の鍵の設定
- ホスト名の変更
- SNMP トラップのイネーブル化
- SNMP コミュニティ ストリングの設定
- ログイン バナーの設定
- Web-Based Manager ロゴの設定
- セッション タイムアウトの設定

Detector モジュールのサービスのアクティブ化

Detector モジュールには、いくつかのサービス オプションがあります。そのサービス オプションをアクティブにするには、サービスをイネーブルにしてから、サービスへのアクセスを許可される IP アドレスを定義します。この項では、常にアクティブであるセキュア シェル サービス以外のサービスをアクティブにする方法について説明します。

Detector モジュールのサービスは次のとおりです。

- ノード間通信サービス：Detector モジュールは、Cisco Anomaly Guard モジュールとの通信チャネルを確立するときにこのサービスを使用します。詳細については、P.4-18 の「Guard との通信の確立」を参照してください。
- 簡易ネットワーク管理プロトコル（SNMP）サーバ サービス：SNMP を使用して Detector モジュールにアクセスすることにより、次の MIB で定義された情報を取得できます。
  - Riverhead の専用 MIB。
  - MIB2（RFC1213-MIB）：EGP グループおよび伝送 MIB グループ以外のすべての MIB グループ。
  - UCDAVIS（UCD-SNMP-MIB）：memory、latable、systemStats、version、および snmperrs の MIB グループのみ。

  MIB 定義の詳細については、このソフトウェア バージョンでリリースされた MIB ファイルを参照してください。

（注）　Riverhead MIB には、64 ビットのカウンタが含まれています。MIB を読み取るには、SNMP バージョン 2 をサポートするブラウザを使用する必要があります。

- SNMP トラップ サービス：snmp-trap サービスをアクティブにすると、Detector モジュールは SNMP トラップを生成します。詳細については、P.4-29 の「SNMP トラップのイネーブル化」を参照してください。
- セキュア シェル（SSH）サービス：SSH サービスは常にアクティブです。詳細については、P.3-15 の「SSH を使用した Detector モジュールへのアクセス」および P.4-25 の「SSH 鍵の管理」を参照してください。
- Web-Based Manager（WBM）サービス：Web ブラウザを使用して Web から Detector モジュールを制御できます。詳細については、P.3-13 の「Web-Based Manager を使用した Detector モジュールの管理」を参照してください。
- MultiDevice Manager（MDM）サービス：Web ブラウザを使用して、Detector モジュールなどの Guard および Detector デバイスを MDM から監視および制御できます。詳細については、P.3-14 の「Cisco DDoS MultiDevice Manager による Detector モジュールの管理」を参照してください。

Detector モジュールのサービスをアクティブにするには、次の手順を実行します。

ステップ 1　設定モードで次のコマンドを入力して、Detector モジュールのサービスをイネーブルにします。

```
service {internode-comm | mdm | snmp-server | snmp-trap | wbm}
```

表 4-1 に、**service** コマンドのキーワードを示します。

表 4-1　service コマンドのキーワード

サービス	説明
internode-comm	ノード間通信サービスを指定します。
mdm	MDM サービスを指定します。
snmp-server	SNMP サーバ サービスを指定します。
snmp-trap	SNMP トラップ サービスを指定します。
wbm	WBM サービスを指定します。

ステップ 2　次のいずれかのコマンドを入力して、Detector モジュールのサービスへのアクセスを許可します。

- MDM サービスの場合は、設定モードで次のコマンドを入力して、MDM から Detector モジュールのサービスへのアクセスを許可します。

  mdm server *ip-addr*

  ip-addr 引数には、MDM サーバの IP アドレスを定義します。IP アドレスをドット区切り 10 進表記で入力します。
- その他のサービスの場合はすべて、設定モードで次のコマンドを入力して、Detector モジュールのサービスへのアクセスを許可し、接続をイネーブルにします。

  permit {**internode-comm** {*ip-address-general* [*ip-mask*] | *} | **snmp-server** {*ip-address-general* [*ip-mask*] | *} | **ssh** {*ip-address-general* [*ip-mask*] | *} [*if-service*] | **wbm** {*ip-address-general* [*ip-mask*] | *} [*if-service*]}

表 4-2 に、**permit** コマンドの引数とキーワードを示します。

表 4-2　permit コマンドの引数とキーワード

パラメータ	説明
internode-comm	ノード間通信サービスを指定します。
ip-address-general	アクセスを許可する IP アドレス。IP アドレスをドット区切り 10 進表記で入力します（たとえば 192.168.10.2）。
ip-mask	（オプション）IP サブネット マスク。サブネット マスクをドット区切り 10 進表記で入力します（たとえば 255.255.255.0）。デフォルトのサブネット マスクは、255.255.255.255 です。
*	ワイルドカード文字としてのアスタリスク（*）。アスタリスク（*）を使用すると、すべてのリモート マネージャの IP アドレスからのアクセスが許可されます。 ⚠ **注意**　セキュリティ上の理由から、すべての IP アドレスからのサービスへのアクセスを許可しないことをお勧めします。
snmp-server	SNMP サーバ サービスを指定します。
ssh	SSH サービスを指定します。
wbm	WBM サービスを指定します。
if-service	（オプション：1 Gbps 動作のみ）ユーザ アクセスを管理インターフェイスだけに制限する管理ポート デジグネータ。デフォルトはすべてのインターフェイスです。mng を入力します。 デフォルト設定に戻すには、このコマンドの **no** バージョンを使用します。

次の例は、WBM サービスをアクティブにする方法を示しています。

```
user@DETECTOR-conf# service wbm
user@DETECTOR-conf# permit wbm 192.168.10.35
```

AAA を使用したアクセス コントロールの設定

認証、認可、アカウンティング（AAA）とは、Detector モジュールおよび Detector モジュールのサービスへのユーザ アクセスを制御する方式のことです。AAA には次の機能があります。

- 認証 : ユーザに対しシステムおよびシステム サービスへのアクセスを許可する前に、そのユーザを識別します。
- 認可 : ユーザがシステムへのアクセス権を取得した後で、実行が許可される内容を決定します。このプロセスは、ユーザ認証後に発生します。
- アカウンティング : ユーザが実行中または実行済みの内容を記録します。アカウンティングにより、ユーザがアクセスしているサービスを追跡することができます。

Detector モジュールには、次のシステム ユーザ アカウントがあらかじめ設定されています。

- admin : admin ユーザ アカウントには、Detector モジュールの CLI およびすべての機能にアクセスできる管理アクセス権が設定されています。Detector モジュールの CLI に初めて接続すると、このアカウントに対するパスワードを設定するように要求されます。新しいユーザ アカウントを設定するには admin ユーザ アカウントを使用します。
- riverhead : riverhead ユーザ アカウントには、ダイナミック（dynamic）のアクセス権が設定されています。Detector モジュールはこのユーザ アカウントを使用して、Guard との最初の通信チャネルを確立します。Detector モジュール CLI に初めて接続すると、このアカウントに対するパスワードを設定するように要求されます。

システム ユーザ アカウントは削除できません。

Detector モジュールのユーザ コミュニティをドメインに分割し、安全な管理アクセスのためにパスワードを割り当てることができます。初期設定が完了した後は、ユーザのアクションを監視できるように新しいアカウントを作成し、システム ユーザ アカウントは使用しないことをお勧めします。

この項では、次のトピックについて取り上げます。

- 認証の設定
- 認可の設定
- アカウンティングの設定
- TACACS+ サーバ アトリビュートの設定

認証の設定

ユーザが Detector モジュールにログインしようとしたとき、または（**enable** コマンドを使用して）上位の特権レベルを要求したときに、Detector モジュールで使用する認証方式を設定することができます。Detector モジュールは、次の認証オプションを提供します。

- ローカル認証 : ローカルに設定されたログイン名およびイネーブル パスワードを認証に使用する認証方式。この認証方式がデフォルトです。詳細については、P.4-6 の「ローカル認証の設定」を参照してください。
- Terminal Access Controller Access Control System Plus（TACACS+）認証 : 1 つ以上の TACACS+ サーバを使用するリモート ユーザ認証方式。

（注）　ユーザ認可を設定するには、まず TACACS+ サーバでユーザの認証を設定しておく必要があります。設定していない場合は、Detector モジュールにアクセスできない可能性があります（P.4-9 の「認可の設定」を参照）。

一方または両方のユーザ認証方式を使用するように Detector モジュールを設定できます。TACACS+ 認証を使用するときに、複数の TACACS+ サーバを定義できます。複数の認証方式を定義すると、通信エラーによって最初の認証方式が失敗した場合にバックアップが提供されます。

Detector モジュールは、定義された各方式を、Detector モジュールで定義した順序で使用してユーザを認証します。定義済みの認証方式のリストを表示するには、**show running config** コマンドを入力します。Detector モジュールは、リストの最初に表示されている方式を使用してユーザの認証を試みます。最初の認証方式が応答しない場合、Detector モジュールは、応答のある認証方式が見つかるまで、リストにある次の認証方式を順に選択していきます。

tacacs-server first-hit コマンドを使用して、Detector モジュールが最初の TACACS+ サーバからの応答を受信する際に実行するアクションを設定できます。first-hit オプション（デフォルト設定）をディセーブルにしていて、最初のサーバが認証を拒否した場合、Detector モジュールは、他の TACACS+ サーバを順にスキャンして、その認証を受け入れるサーバを検索します。定義されていない TACACS+ サーバが認証を受け入れた場合、または Guard がどのサーバとも通信できなかった場合は、ユーザ認証は失敗します。first-hit オプションをイネーブルにしている場合、Detector モジュールは、最初の TACACS+ サーバの認証応答（拒否または受け入れ）を最終決定として受け入れます。デフォルトでは、first-hit オプションはディセーブルになっています。**tacacs-server first-hit** コマンドの詳細については、P.4-16 の「TACACS+ 検索方式の設定」を参照してください。

（注）　Detector モジュールが TACACS+ サーバと通信できない場合に、ローカル データベースをユーザ認証のフォールバックとして使用するように Detector モジュールを設定できます（「認証方式の設定」の項を参照）。

この項では、次のトピックについて取り上げます。

- 認証方式の設定
- ローカル認証の設定

認証方式の設定

Detector モジュールで使用する認証方式を設定するには、次の手順を実行します。

ステップ 1　TACACS+ 認証が必要な場合は、TACACS+ サーバ接続を設定します。詳細については、P.4-14 の「TACACS+ サーバ アトリビュートの設定」を参照してください。

ステップ 2　設定モードで次のコマンドを入力し、認証方式を定義します。

```
aaa authentication {enable | login} {local | tacacs+}
[tacacs+ | local]
```

表 4-3 に、**aaa authentication** コマンドのキーワードを示します。

表 4-3　aaa authentication コマンドのキーワード

パラメータ	説明
enable	ユーザが上位の特権レベルに入るときに Detector モジュールが認証できるようにします。
login	ユーザがログインするときに Detector モジュールが認証できるようにします。
local	Detector モジュールがユーザを認証するために使用するローカル データベースを指定します。
tacacs+	TACACS+ サーバがユーザを認証できるようにします。
tacacs+ | local	（オプション）設定された認証方式が失敗した場合の代替の認証方式を指定します。

認証方式を変更するには、このコマンドを再入力します。

次の例は、上位の特権レベルに入る際に認証を行うように設定する方法を示しています。最初の認証方式は TACACS+ に設定され、2 番目の認証方式はローカル ユーザ データベースに設定されています。

```
user@DETECTOR-conf# aaa authentication enable tacacs+ local
```

ローカル認証の設定

Detector モジュールには、管理者特権を持つユーザ名（ユーザ定義と呼ばれる）があらかじめ設定されています。このユーザ名を使用して新しいユーザを作成できます。ユーザ定義を使用すると、Detector モジュールのユーザ コミュニティをドメインに分割し、安全な管理アクセスのためにパスワードを割り当てることができます。

TACACS+ サーバを使用した CLI ユーザの認証をイネーブルにするには、P.4-4 の「認証の設定」を参照してください。

この項では、次のトピックについて取り上げます。

- ユーザの追加
- 自分のパスワードの変更
- 他のユーザのパスワードの変更
- ローカル ユーザ データベースからのユーザの削除

ユーザの追加

Detector モジュールのローカル データベースにユーザを追加するには、設定モードで次のコマンドを使用します。

username *username* {**admin** | **config** | **dynamic** | **show**} [*password*]

表 4-4 に、**username** コマンドの引数とキーワードを示します。

表 4-4　username コマンドの引数とキーワード

パラメータ	説明
username	ユーザの名前。1 ～ 63 文字の英数字の文字列を入力します。大文字と小文字が区別され、先頭は英字である必要があります。この文字列にはスペースを含めることはできませんが、アンダースコアを含めることはできます。
admin	すべての操作にアクセスできます。
config	ユーザの定義、削除、および修正に関連する操作を除いて、すべての操作にアクセスできます。
dynamic	監視と診断、検出、およびラーニングに関する操作にアクセスできます。dynamic 特権を持つユーザは、フレックスコンテンツ フィルタおよび動的フィルタを設定することもできます。
show	監視操作と診断操作にアクセスできます。
password	（オプション）ユーザ パスワード。6 ～ 24 文字の英数字の文字列を入力します。スペースは使用できず、大文字と小文字が区別されます。パスワードを入力しない場合、入力するよう要求されます。

次の例は、新しいユーザを設定し、パスワードを設定する方法を示しています。

```
user@DETECTOR-conf# username Robbin config 123456
```

ユーザはクリア テキストでパスワードを入力しますが、Detector モジュールの設定ファイルでは、パスワードが暗号化されて表示されます。次の例は、Detector　モジュールの設定ファイル（running-config）を表示します。

```
username Richard config encrypted 840xdMk3
```

上の例の **encrypted** キーワードは、パスワードが暗号化されていることを示しています。

Detector モジュール上に設定されているユーザのリストを表示するには、**show running-config** コマンドまたは **show detector** コマンドを使用します。

現在 CLI にログインしているユーザのリストを表示するには、**show users** コマンドを使用します。

自分のパスワードの変更

ユーザは、自分自身のパスワードを変更することができます。管理者は、自分自身のパスワードと、他のユーザのパスワードを変更できます（P.4-8 の「他のユーザのパスワードの変更」を参照）。

自分自身のパスワードを変更するには、次の手順を実行します。

ステップ 1　グローバル モードで次のコマンドを入力します。

```
password
```

ステップ 2　現在のパスワードを入力します。新しいパスワードの入力を求めるプロンプトが表示されます。

ステップ 3　新しいパスワードを入力します。パスワードは、スペースを含まない、6 ～ 24 文字の英数字の文字列である必要があります。パスワードでは大文字と小文字が区別されます。確認のため新しいパスワードを再入力するよう求めるプロンプトが表示されます。

次の例は、自分のパスワードを変更する方法を示しています。

```
user@DETECTOR# password
Old Password: <old-password>
New Password: <new-password>
Retype New Password: <new-password>
```

他のユーザのパスワードの変更

他のユーザのパスワードを変更するには、管理ユーザ特権を持っている必要があります。

他のユーザのパスワードを変更するには、次の手順を実行します。

ステップ 1　グローバル モードで次のコマンドを入力します。

password *username-password*

username-password 引数は、変更対象のパスワードの持ち主のユーザです。

ステップ 2　新しいパスワードを入力します。

パスワードは、スペースを含まない、6 ～ 24 文字の英数字の文字列である必要があります。パスワードでは大文字と小文字が区別されます。確認のため新しいパスワードを再入力するよう求めるプロンプトが表示されます。

次の例では、管理者がユーザ Jose のパスワードを変更しています。

```
user@DETECTOR# password Jose
New Password: <new-password>
Retype New Password: <new-password>
```

ローカル ユーザ データベースからのユーザの削除

ローカル ユーザ データベースからユーザを削除すると、そのローカル ユーザ データベースのみを使用して認証を行っている場合、関連付けられているユーザが Detector モジュールにアクセスできなくなります。

Detector モジュールのローカル ユーザ データベースからユーザを削除するには、**no username** *username* コマンドを使用します。

次の例は、ローカル ユーザ データベースからユーザを削除する方法を示しています。

```
user@DETECTOR-conf# no username Robbin
```

認可の設定

システム管理者は、ユーザが使用できるサービスを制限することができます。認可をイネーブルにすると、Detector モジュールはユーザ プロファイルを確認します。ユーザ プロファイルは、ローカル ユーザ データベース内または TACACS+ セキュリティ サーバ上にあります。ユーザは、そのユーザのプロファイル内の情報で許可されている場合にのみ、要求したサービスへのアクセスを許可されます。

ユーザがコマンドを実行しようとするときに Detector モジュールで使用する認可方式を設定することができます。Detector モジュールでは、次の認可オプションが提供されています。

- TACACS+ 認可：1 つ以上の TACACS+ サーバを使用するリモート ユーザ認可方式。

  次の 2 種類の TACACS+ 認可がサポートされています。

  - EXEC 認可：ユーザが Detector モジュールにログインして認証されたときに、そのユーザの特権レベルを決定します。
  - コマンド認可：ユーザがコマンドを入力すると、そのコマンドの許可を取得するために、TACACS+ サーバを調べます。

  TACACS+ 認可では、コマンドごとにアクセス権を指定できます。

注意　認可は **copy running-config** コマンドに制限することをお勧めします。これは、**copy running-config** コマンドを使用すると、設定ファイル内でユーザがすべてのコマンドに対して認可されているかどうかに関係なく、ユーザがすべての設定コマンドを実行できるようになるためです。

- ローカル認可：コマンド グループのアクセス コントロールにローカルで設定されたユーザ プロファイルを使用する認可方式。認可は、指定された特権レベルのすべてのコマンドに対して定義されます。この認可方式がデフォルトです。

Detector モジュールは、TACACS+ サーバへの通信に失敗した場合、ローカル認可を使用できます。

ユーザ認証方式を定義する順次認証リストを設定できます。順次認証リストには認証に使用する方式を 1 つ以上指定でき、最初の認証方式への通信が失敗した場合は、バックアップが提供されます。

Detector モジュールは、最初に一覧表示した方式を使用してユーザを認可します。その方式が応答しない場合、Detector モジュールは 2 番目の認可方式を選択します。認可方式が両方とも成功しなかった場合にのみ、認可は失敗します。

Detector モジュールが認証拒否を最終的なものと見なし、それ以上他の TACACS+ サーバやローカル ユーザ データベースを検索しないように設定するために、TACACS+ サーバ アトリビュートを設定できます。詳細については、P.4-14 の「TACACS+ サーバ アトリビュートの設定」を参照してください。

この項では、次のトピックについて取り上げます。

- ローカル認可の設定
- 認可方式の設定
- TACACS+ サーバの設定例
- ゾーン名のタブ補完のディセーブル化

ローカル認可の設定

Detector の操作にアクセスできるかどうかは、ユーザの特権レベルによって決まります。システム管理者は、ユーザが使用できる操作を制限することができます。Detector モジュールは、ユーザのプロファイルをチェックして、ユーザのアクセス権を確認します。認可されると、ユーザは、そのユーザのプロファイル内の情報で許可されている場合にのみ、要求した操作へのアクセス権を付与されます。ユーザの特権レベルについては、表 3-1 を参照してください。

この項では、次のトピックについて取り上げます。

- パスワードを使用した特権レベルの割り当て
- ユーザ特権レベル間の移動

パスワードを使用した特権レベルの割り当て

管理者は、ユーザの特権レベルへのアクセスを制限するパスワードを設定できます。特権レベルおよびパスワードを指定したら、この特権レベルにアクセスする必要のあるユーザにそのパスワードを付与することができます。ユーザは、特権レベルのパスワードを知らないと、そのパスワードで保護されたレベルに移動することはできません。

ローカル パスワードを設定して特権レベルへのアクセスを制御するには、設定モードで次のコマンドを使用します。

enable password [**level** *level*] [*password*]

表 4-5 に、**enable password** コマンドの引数とキーワードを示します。

表 4-5　enable password コマンドの引数とキーワード

パラメータ	説明
level *level*	（オプション）ユーザの特権レベル。特権レベルは次のいずれかになります。 • **admin**：すべての操作にアクセスできる。 • **config**：ユーザの定義、削除、および修正に関連する操作を除いて、すべての操作にアクセスできる。 • **dynamic**：監視と診断、検出、およびラーニングに関する操作にアクセスできる。dynamic 特権を持つユーザは、フレックスコンテンツ フィルタおよび動的フィルタを設定することもできます。 • **show**：監視操作と診断操作にアクセスできる。 デフォルトのレベルは **admin** です。
password	（オプション）特権レベルのパスワード。パスワードは、スペースを含まない、6 ～ 24 文字の英数字の文字列である必要があります。パスワードでは大文字と小文字が区別されます。パスワードを入力しない場合、入力するよう要求されます。

次の例は、ユーザの特権レベル **admin** にパスワードを割り当てる方法を示しています。

```
user@DETECTOR-conf# enable password level admin <password>
```

ユーザ特権レベル間の移動

認可されたユーザは、ユーザ特権レベル間を移動することができます。

ユーザ特権レベル間を移動するには、次の手順を実行します。

ステップ 1　グローバル モードで次のコマンドを入力します。

```
enable [level]
```

level 引数には、ユーザの特権レベルを指定します。このレベルは、次のいずれかです。

- **admin**：すべての操作にアクセスできる。
- **config**：ユーザの定義、削除、および修正に関連する操作を除いて、すべての操作にアクセスできる。
- **dynamic**：監視と診断、検出、およびラーニングに関する操作にアクセスできる。dynamic 特権を持つユーザは、フレックスコンテンツ フィルタおよび動的フィルタを設定することもできます。
- **show**：監視操作と診断操作にアクセスできる。

デフォルトのレベルは **admin** です。

ステップ 2　特権レベルのパスワードを入力します。

次の例は、ユーザの特権レベル **admin** に切り替える方法を示しています。

```
user@DETECTOR> enable admin
Enter enable admin Password: <password>
```

show 特権レベル（表 4-5 の説明を参照）に戻る場合は、**disable** コマンドを使用します。

認可方式の設定

認可方式を設定するには、次の手順を実行します。

ステップ 1　TACACS+ 認可が必要な場合は、TACACS+ サーバ接続を設定します。詳細については、P.4-14 の「TACACS+ サーバ アトリビュートの設定」を参照してください。

ステップ 2　設定モードで次のいずれかのコマンドを入力して、認可方式を定義します。

- **aaa authorization exec tacacs+**
- **aaa authorization commands** *level* **tacacs+**

認可方式のシーケンシャルなリストを設定できます。各方式について、**aaa authorization** コマンドを入力します。認証方式を削除するには、このコマンドの **no** 形式を使用します。

表 4-6 に、**aaa authorization** コマンドの引数とキーワードを示します。

表 4-6　aaa authorization コマンドの引数とキーワード

パラメータ	説明
exec	ユーザが EXEC シェルの実行を許可されているかどうかを判別するために認可が実行されます。Detector モジュールは、TACACS+ サーバに確認して、認証されたユーザの特権レベルを判別します。 **注意**　認可を設定する前に、TACACS+ サーバにそのユーザを設定しておく必要があります。設定しておかないと、Detector モジュールにアクセスできない可能性があります。
commands	指定された特権レベルのすべてのコマンドに対して認可が実行されます。複数の特権レベルの認可を設定するには、認可が必要な特権レベルごとにこのコマンドを使用します。
level	次のいずれかの特権レベルに対する認可です。 • **admin**：すべての操作にアクセスできる。 • **config**：ユーザの定義、削除、および修正に関連する操作を除いて、すべての操作にアクセスできる。 • **dynamic**：監視と診断、検出、およびラーニングに関する操作にアクセスできます。dynamic 特権を持つユーザは、フレックスコンテンツ フィルタおよび動的フィルタを設定することもできます。
tacacs+	TACACS+ サーバでユーザのアクセス権を確認します。

Detector モジュールのパフォーマンスに影響する可能性があるため、**show** 特権レベルコマンドに対する認可は設定しないことをお勧めします。

（注）　コンソール セッションから入力したコマンドには、TACACS+ 認可は実行されません。

次の例は、config 特権レベルを必要とするコマンドの認可を設定する方法を示しています。最初の認可方式は TACACS+ に設定され、2 番目の認可方式はローカル ユーザ データベースに設定されています。

```
user@DETECTOR-conf# aaa authorization commands config tacacs+ local
```

注意　設定コマンド モードにアクセスできるようにするには、dynamic ユーザ特権レベルに対するアクセス権を付与するか、**configure** コマンドへのアクセス権を指定する必要があります。

TACACS+ サーバの設定例

TACACS+ サーバのデータベースで、各コマンドの認可を指定することができます。

次の例は、ユーザ Zoe に対して、TACACS+ サーバ上で認可を設定する方法を示しています。

```
user=Zoe {
    cmd = detect {
        permit .*
    }
    cmd = "no detect" {
        permit .*
    }
    cmd = learning {
        deny policy*
    }
    cmd = "no learning" {
        deny .*
    }
    cmd = dynamic-filter {
        permit .*
    }
    cmd = "no dynamic-filter" {
        permit .*
    }
}
```

ゾーン名のタブ補完のディセーブル化

ゾーン名を入力するときのタブ補完機能をディセーブルにして、ゾーン設定へのアクセスを認可されたユーザのみに制限できます。この設定は、ゾーン名を指定するすべてのコマンドに適用されます。

グローバル モードまたは設定モードで **zone** コマンド、**no zone** コマンド、**show zone** コマンド、および **deactivate** コマンドなどのコマンドを入力しても、Detector モジュールはゾーン名の表示や補完を行わなくなります。ゾーン名を完全に入力する、ゾーン操作モードを変更する、またはゾーン統計情報を表示する必要があります。

ゾーン名のタブ補完をディセーブルにすると、Detector モジュールは **tab-complete zone-list** コマンドを TACACS+ サーバに送信します。認可されたユーザに対してゾーン名のタブ補完をイネーブルにするには、**tab-complete zone-list** コマンドに対する認可を TACACS+ サーバ上で設定します。

次の例は、すべての **zone** コマンドでゾーン名のタブ補完をディセーブルにする方法を示しています。

```
user@DETECTOR-conf# aaa authorization commands zone-completion tacacs+
```

ゾーン名のタブ補完をイネーブルにするには、このコマンドの **no** 形式を使用します。

アカウンティングの設定

アカウンティング管理により、ユーザがアクセスしているサービスを追跡し、TACACS+ サーバに関するアカウンティング情報を保存することができます。課金、レポート、またはセキュリティ目的で、要求されたサービスのアカウンティングをイネーブルにできます。デフォルトでは、Detector モジュールでアカウンティング管理はディセーブルに設定されています。

アカウンティングを設定するには、次の手順を実行します。

ステップ 1　TACACS+ サーバ接続を設定します。詳細については、P.4-14 の「TACACS+ サーバ アトリビュートの設定」を参照してください。

ステップ 2　設定モードで次のコマンドを入力して、アカウンティングを設定します。

```
aaa accounting commands {show | dynamic | config | admin} stop-only {local | tacacs+}
```

表 4-7 に、**aaa accounting commands** コマンドのキーワードを示します。

表 4-7　aaa accounting commands コマンドのキーワード

パラメータ	説明
show | dynamic | config | admin	指定された特権レベルのアカウンティングを定義します（ユーザの特権レベルについては、表 3-1 を参照）。
stop-only	コマンドの実行が終了したときにアクションを記録します。
tacacs+	アカウンティング情報の記録に TACACS+ サーバのデータベースを使用します。
local	アカウンティング情報を保存しません。

複数の特権レベルのアカウンティングを設定するには、必要に応じて、特権レベルごとに **aaa accounting** コマンドを入力します。

アカウンティング管理は、config ユーザ特権レベルに対してのみイネーブルにすることをお勧めします。アカウンティング データの追跡および保存を行うと、Detector モジュールのパフォーマンスに影響を及ぼす可能性があります。

特権レベルのアカウンティング管理を削除するには、このコマンドの **no** 形式を使用します。

次の例は、TACACS+ サーバ上で config 特権レベルを必要とするコマンドのアカウンティングを設定する方法を示しています。

```
user@DETECTOR-conf# aaa accounting commands config stop-only tacacs+
```

TACACS+ サーバ アトリビュートの設定

TACACS+ サーバで認証、認可、またはアカウンティングをイネーブルにするには、TACACS+ サーバ アトリビュートを設定する必要があります。

注意　TACACS+ 認証方式を適用する前に、TACACS+ サーバ アトリビュートを設定しておく必要があります。設定していない場合は、Detector モジュールにアクセスできないことがあります。

TACACS+ サーバのアトリビュートを設定するには、次の手順を実行します。

ステップ 1　**tacacs-server host** *ip-address* **port** *port_number* コマンドを入力して、TACACS+ サーバの IP アドレスを設定します。

詳細については、P.4-15 の「TACACS+ サーバの IP アドレスの設定」を参照してください。

ステップ 2　**tacacs-server key** *tacacs-key* コマンドを入力して、Detector モジュールが TACACS+ サーバへのアクセスに使用する暗号鍵を設定します。

詳細については、P.4-16 の「TACACS+ サーバの暗号鍵の設定」を参照してください。

ステップ 3　（オプション）**tacacs-server first-hit** コマンドを入力して、Detector モジュールが認証に使用する検索方式を設定します。

詳細については、P.4-16 の「TACACS+ 検索方式の設定」を参照してください。

ステップ 4　（オプション）**tacacs-server timeout** *timeout* コマンドを入力して、TACACS+ サーバ接続のタイムアウトを設定します。

詳細については、P.4-17 の「TACACS+ サーバの接続タイムアウトの設定」を参照してください。

ステップ 5　**show tacacs statistics** コマンドを入力して、TACACS+ サーバ接続の統計情報を表示します。

詳細については、P.4-17 の「TACACS+ サーバの統計情報の表示」を参照してください。

Detector モジュールのユーザ特権レベルは、TACACS+ の特権番号に次のように対応しています。

- **admin** = 15
- **config** = 10
- **dynamic** = 5
- **show** = 0

この項では、次のトピックについて取り上げます。

- TACACS+ サーバの IP アドレスの設定
- TACACS+ サーバの暗号鍵の設定
- TACACS+ 検索方式の設定
- TACACS+ サーバの接続タイムアウトの設定
- TACACS+ サーバの統計情報の表示

TACACS+ サーバの IP アドレスの設定

Detector モジュールが TACACS+ サーバの一連のリストを認証、認可、およびアカウンティングに使用するように設定できます。Detector モジュールは、TACACS+ サーバ リストを使用してユーザを認証、認可、またはアカウンティング イベントを送信します。そのサーバが応答しない場合、Detector モジュールは 2 番目のサーバを選択します。リスト上のすべてのサーバが応答しなかった場合にのみ、認証または認可は失敗します。

または、Detector モジュールがリストの最初の TACACS+ サーバのみを使用してユーザを認証するように設定することもできます（詳細については、P.4-16 の「TACACS+ 検索方式の設定」を参照）。

リストには、各 TACACS+ サーバの IP アドレスを定義する必要があります。最大 9 つの TACACS+ サーバを定義できます。

リストに TACACS+ サーバを追加し、IP アドレスを割り当てるには、設定モードで次のコマンドを使用します。

tacacs-server host *ip-address* [**port** *port_number*]

表 4-8 に、**tacacs-server host** コマンドの引数とキーワードを示します。

表 4-8　tacacs-server host コマンドの引数とキーワード

パラメータ	説明
ip-address	TACACS+ サーバの IP アドレス。IP アドレスをドット区切り 10 進表記で入力します（たとえば 192.168.100.1）。
port *port_number*	（オプション）使用するポート番号を指定します。ポート番号を指定しない場合、Detector モジュールはポート 49（デフォルト）を使用します。

TACACS+ サーバは、入力した順序でリストに追加されます。リストには、最大 9 つのサーバを追加できます。

次の例は、TACACS+ サーバ リストにサーバを追加する方法を示しています。

```
user@DETECTOR-conf# tacacs-server host 192.168.33.45 port 60
```

TACACS+ サーバの暗号鍵の設定

TACACS+ サーバにアクセスするには、暗号鍵を設定する必要があります。暗号鍵は、TACACS+ サーバ上の暗号鍵と一致する必要があります。暗号鍵にスペースを含めることはできません。

サーバの暗号アクセス鍵を設定するには、設定モードで次のコマンドを使用します。

tacacs-server key *tacacs-key*

引数 *tacacs-key* は、最大 100 文字の英数字の文字列です。

（注）　定義できる暗号鍵は 1 つだけです。複数の TACACS+ サーバを使用する場合、Detector モジュールは、同じ鍵を使用して、すべての TACACS+ サーバとの通信を暗号化します。

暗号化機能をディセーブルにするには、次のコマンドを使用します。

no tacacs-server key

次の例は、TACACS+ サーバの暗号鍵を MyKey に設定する方法を示しています。

```
user@DETECTOR-conf# tacacs-server key MyKey
```

TACACS+ 検索方式の設定

設定モードで **tacacs-server first-hit** コマンドを使用すると、認証拒否を最終的なものと見なし、他の TACACS+ サーバをそれ以上検索しないように、Detector モジュールを設定できます。Detector モジュールは、サーバ リストの最初の TACACS+ サーバだけを使用してユーザ認証を実行します。最初の TACACS+ サーバが応答しない場合、Detector モジュールはリストにある次のサーバを選択します。Detector モジュールは、ユーザ認証に対して最初に受け取る承認または拒否を最終決定と見なし、他の TACACS+ サーバを使用したそのユーザ認証の試行を停止します。

定義されている TACACS+ サーバの順次検索を継続して、ユーザ認証を受け入れるサーバを見つけるように Detector モジュールを設定するには、設定モードで **no tacacs-server first-hit** コマンドを使用します。この方式が、first-hit 動作のデフォルト設定です。すべての定義済み TACACS+ サーバがユーザ認証を拒否した場合や、Detector モジュールがどのサーバとも通信できない場合は、ユーザ認証が失敗します。

次の例は、Detector モジュールがリスト内の最初の TACACS+ サーバのみを使用してユーザ認証をするように、TACACS+ 検索方式を設定する方法を示しています。

```
user@DETECTOR-conf# tacacs-server first-hit
```

TACACS+ サーバの接続タイムアウトの設定

Detector モジュールが TACACS+ サーバからの応答を待つ時間を設定できます。タイムアウトが終了すると、Detector モジュールは次の TACACS+ サーバ（そのようなサーバが設定されている場合）との接続を確立しようとするか、ローカルの AAA にフォールバックします（フォールバックが設定されている場合）。フォールバックの方式が設定されていない場合、認証と認可は失敗します。

（注）　すべての TACACS+ サーバとの通信に同じサーバ タイムアウトが使用されます。

TACACS+ サーバの接続タイムアウトを設定するには、設定モードで次のコマンドを使用します。

tacacs-server timeout *timeout*

timeout 引数には、Detector モジュールが TACACS+ サーバの応答を待つ時間を秒単位で指定します。デフォルトのタイムアウトは 0 です。

次の例は、TACACS+ サーバの接続タイムアウトを 600 秒に設定する方法を示しています。

```
user@DETECTOR-conf# tacacs-server timeout 600
```

ヒント　ネットワークに問題がある場合や、TACACS+ サーバの応答が遅いためにタイムアウトが繰り返し発生する場合は、タイムアウトの値を大きくすることができます。

TACACS+ サーバの統計情報の表示

設定モードで **show tacacs statistics** コマンドを使用して、定義した TACACS+ サーバの統計情報を表示できます。

TACACS+ の統計情報をクリアするには、設定モードで **clear tacacs statistics** コマンドを使用します。

表 4-9 に、**show tacacs statistics** コマンド出力のフィールドを示します。

表 4-9　show tacacs statistics コマンド出力のフィールドの説明

フィールド	説明
PASS	Detector モジュールが TACACS+ サーバに正常にアクセスし、アクセス権を付与された回数。
FAIL	Detector モジュールが TACACS+ サーバに正常にアクセスし、アクセス権を拒否された回数。
ERROR	Detector モジュールが TACACS+ サーバにアクセスできなかった回数。

Guard との通信の確立

Detector モジュールと、Detector モジュールのリモート Guard リストに定義した Guard の間に安全な通信チャネルを確立できます。安全な通信チャネルにより、Detector モジュールは次のタスクを実行できます。

- Guard のアクティブ化：Detector モジュールは、ゾーン トラフィックの異常を検出すると、通信チャネルを使用して、ゾーン保護を行う Guard をアクティブにし、ゾーン保護中に Guard をポーリングします。
- ゾーン設定の同期化：Detector モジュールは、通信チャネルを使用して、ゾーン設定情報を Guard と交換します。

ユーザは、Detector モジュールと Guard の両方で通信チャネル パラメータを設定した後、Detector モジュールから Guard との接続を開始します。この接続により、Detector モジュールは、安全な通信チャネルの確立に必要な鍵と証明書を Guard と交換します。その後 Detector モジュールは接続を閉じ、Guard をアクティブにする、ゾーン設定を同期させる、または Guard にポーリングする必要がある場合に、通信チャネルを確立します。

Detector モジュールおよび Guard は、次の 2 つのタイプの通信チャネルをサポートしています。

- セキュア シェル（SSH）バージョン 2：Detector モジュールが Guard をアクティブにできるようにします。
- Secure Sockets Layer（SSL）：Detector モジュールが Guard のアクティブ化、Guard のポーリング、およびゾーン設定の同期化を行うことができるようにします。

Detector モジュール上のゾーン リモート Guard リストおよびデフォルト リモート Guard リストを使用して、Detector モジュールがゾーン保護と同期化のために通信する Guard を指定します。リモート Guard リストに Guard を指定する場合は、Detector モジュールがその Guard と確立する通信チャネルのタイプ（SSH または SSL）を選択します。SSH または SSL 通信チャネルを確立するには、両方のデバイスに SSH サービスが必要です。デフォルトでは、両方のデバイスで SSH サービスが常にイネーブルになっています。SSL 通信チャネルを確立する場合、Detector モジュールは Guard との初期接続に限り SSH 通信チャネルを使用します。このときに、デバイスが鍵と証明書を交換します。

（注）　Guard との通信チャネルを確立する前に、Detector モジュールのリモート Guard リストに Guard を追加しておく必要があります。詳細については、P.9-6 の「ゾーンを保護するためのリモート Guard のアクティブ化」を参照してください。

この項では、次のトピックについて取り上げます。

- SSL 通信チャネル パラメータの設定
- SSH 通信チャネル パラメータの設定
- 通信チャネルの確立

SSL 通信チャネル パラメータの設定

次のように Detector モジュールが Guard と相互に通信する必要がある場合は、Detector モジュールと Guard の間に SSL 通信チャネルを設定します。

- Detector モジュールがトラフィック異常を検出したときに Guard をアクティブにする。
- Guard とゾーン設定を同期化する。

- Guard をポーリングして、ゾーンに対する攻撃が終了したことを確認する。Detector モジュールで検出およびラーニング プロセスをイネーブルにした場合、Detector モジュールは、ゾーンに対する攻撃を検出すると、ラーニング プロセス（しきい値調整）を停止します。Detector モジュールは、攻撃を軽減するためにアクティブにした Guard をポーリングし、攻撃がいつ終了したかを確認します。その時点で、Detector モジュールは自動的にラーニング プロセスを再開します。
- Guard との通信を監視し、リモート処理（Guard によるゾーン保護のアクティブ化など）が失敗した場合に通知する。

SSL 通信チャネルとは、認証とデータの暗号化を組み合せることにより安全な接続を提供するもので、デジタル証明書、秘密鍵と公開鍵の交換ペア、および Diffie-Hellman 鍵合意パラメータによって高度なセキュリティを実現します。SSL は、指定された受信者のみがデータを解読できるようにデータを暗号化します。

Guard および Detector モジュールはそれぞれ、デジタル証明書を使用して、通信チャネル経由で通信を試みるデバイスを認証します。SSL 証明書にある Guard と Detector モジュールの ID は、デバイスの IP アドレスに関連付けられます。安全な接続を確保するために、Detector モジュールは秘密鍵と公開鍵のペアを生成し、公開鍵をリモート Guard リストに定義されている Guard に配布します。

Guard と Detector モジュールの両方で SSL 通信パラメータを設定した後、2 つのデバイス間で通信チャネルを確立する必要があります。これは、Detector モジュールから行います。Guard への初期接続中、Detector モジュールは Guard 上のユーザ riverhead との SSH 通信チャネルを確立し、その後これらのデバイスが通信チャネルのセキュリティ保護に必要な鍵と証明書を交換します。初期接続後、Detector モジュールは、Guard のアクティブ化、Guard のポーリング、またはゾーン設定の同期化が必要な場合に SSL 通信チャネルを確立します。

SSL 通信チャネルの片側でいずれか一方のデバイスを交換した場合、またはいずれか一方のデバイスの IP アドレスを変更した場合は、2 つのデバイスが相互の認証に成功するように、両方のデバイスで SSL 証明書を再生成する必要があります。

この項では、次のトピックについて取り上げます。

- SSL 通信チャネルのイネーブル化
- SSL 証明書の再生成

SSL 通信チャネルのイネーブル化

SSL 通信チャネルをイネーブルにするには、次の接続タイプを許可するように Detector モジュールおよび Guard モジュールを設定する必要があります。

- SSH：Detector モジュールは、SSH 通信チャネルを使用して Guard モジュールとの初期接続を確立し、鍵と証明書を交換します。
- SSL：初期接続後、Detector モジュールは SSL 通信チャネルを使用して、Guard モジュールとのすべての接続を確立します。

注意　Guard が TACACS+ 認証を使用してユーザを認証している場合、Detector モジュールが Guard との初期接続中に SSH 通信チャネルを確立できるようにするには、TACACS+ サーバに riverhead ユーザを定義する必要があります。

SSL 通信チャネルをイネーブルにするには、Detector モジュールおよび Guard モジュールの両方で次の手順を実行します。

ステップ 1　設定モードで **permit ssh** *ip-address-general* [*ip-mask*] を入力して、コンパニオン デバイスの IP アドレスによる SSH サービスへのアクセスを許可します。

ip-address-general 引数および *ip-mask* 引数には、コンパニオン デバイスの IP アドレスを指定します。

ステップ 2　設定モードで **service internode-comm** コマンドを入力して、通信チャネル サービスをイネーブルにします。

ステップ 3　設定モードで **permit internode-comm** *ip-address-general* [*ip-mask*] コマンドを入力して、コンパニオン デバイスの IP アドレスによる通信チャネル サービスへのアクセスを許可します。

ip-address-general 引数および *ip-mask* 引数には、コンパニオン デバイスの IP アドレスを指定します。

SSL 通信チャネルをイネーブルにするように Detector モジュールおよび Guard モジュールを設定した後、Guard と Detector の間で通信チャネルを確立できます。通信チャネルの確立については、P.4-22 の「通信チャネルの確立」を参照してください。

SSL 証明書の再生成

SSL 証明書で Guard と Detector モジュールを識別する鍵は、デバイスの IP アドレスに関連付けられます。次の変更を行う場合、通信チャネルの両側で Guard と Detector モジュールの新しい SSL 証明書を再生成する必要があります。

- いずれか一方のデバイスの IP アドレスを変更する。
- いずれか一方のデバイスを交換する。

新しい SSL 証明書を再生成するプロセスには、両方のデバイスから現在の証明書を削除する作業が含まれます。

現在使用している SSL 証明書を表示するには、**show internode-comm certs** コマンドを使用します。

SSL 証明書を再生成するには、次の手順を実行します。

ステップ 1　設定モードで次のコマンドを使用して、Guard の SSL 証明書を Detector モジュールから削除します。

```
cert remove cert-host-ip
```

cert-host-ip 引数には、Guard の IP アドレスを指定します。リモート Guard リストで定義するすべての Guard の SSL 証明書を削除するには、アスタリスク（*）を入力します。

次の例は、SSL 証明書を削除する方法を示しています。

```
user@DETECTOR-conf# cert remove 10.56.36.4
```

ステップ 2　設定モードで次のコマンドを使用して、Detector モジュールの SSL 証明書を Guard から削除します。

```
cert remove cert-host-ip
```

cert-host-ip 引数には、Detector モジュールの IP アドレスを指定します。Guard との通信チャネルを確立しているすべての Detector モジュールの SSL 証明書を削除するには、アスタリスク（*）を入力します。

ステップ 3　Guard を交換する場合は、Detector モジュールから SSH ホスト鍵も削除する必要があります。設定モードで次のコマンドを使用して、Guard の SSH ホスト鍵を Detector モジュールから削除します。

```
no host-keys ip-address-general
```

ip-address-general 引数には、リモート デバイスの IP アドレスを指定します。

次の例は、Detector モジュールからホスト鍵を削除する方法を示しています。

```
user@DETECTOR-conf# no host-keys 10.56.36.4
```

ステップ 4　Detector モジュールから、Guard と Detector モジュールとの間に新しい SSL 通信チャネルを確立することにより、新しい SSL 証明書を再生成します。通信チャネルの確立については、P.4-22 の「通信チャネルの確立」を参照してください。

SSH 通信チャネル パラメータの設定

トラフィック異常を検出したときに Detector モジュールが Guard をアクティブにすること以外に Detector モジュールと Guard が相互に通信する必要がない場合は、Detector モジュールと Guard の間の SSH 通信チャネルを設定します。SSH 通信チャネルでは、Detector モジュールが Guard に対して次のタスクを実行できません。

- Guard とゾーン設定を同期化する。
- Guard をポーリングして、ゾーンに対する攻撃が終了したことを確認する。Detector モジュールで検出およびラーニング プロセスをイネーブルにした場合、Detector モジュールは、ゾーンに対する攻撃を検出すると、ラーニング プロセス（しきい値調整）を停止します。Detector モジュールは、Guard をポーリングできず、攻撃がいつ終了したかを確認できないため、攻撃が終了しても自動的にラーニング プロセスを再開できません。
- Guard との通信を監視し、リモート処理（Guard によるゾーン保護のアクティブ化など）が失敗した場合に通知する。

Detector モジュールがこれらのタスクを実行できるようにするには、SSL 通信チャネルを設定する必要があります（P.4-18 の「SSL 通信チャネル パラメータの設定」を参照）。

安全な SSH 通信チャネルを確保するために、Detector モジュールは秘密 SSH 鍵と公開 SSH 鍵のペアを生成し、公開 SSH 鍵をリモート Guard リストにある Guard に配布します。

SSH 通信チャネルをイネーブルにした後、Detector モジュールと Guard の間で通信チャネルを確立する必要があります。これは、Detector モジュールから行います。

SSH 通信チャネルの片側で Guard を新しいものと交換した場合は、Detector モジュールが新しい Guard での認証に成功するように、Detector モジュールで SSH 秘密（ホスト）鍵および SSH 公開鍵を再生成する必要があります。

この項では、次のトピックについて取り上げます。

- SSH 通信チャネルのイネーブル化
- SSH 通信チャネル鍵の再生成

SSH 通信チャネルのイネーブル化

Guard と Detector モジュールの間の SSH 通信チャネルをイネーブルにするには、Guard から設定モードで **permit ssh** コマンドを入力して、Detector モジュールの IP アドレスで SSH サービスへのアクセスを許可します。

Guard と Detector モジュールの間の SSL 通信チャネルをイネーブルにした後、Guard と Detector モジュールの間で通信チャネルを確立できます。通信チャネルの確立については、P.4-22 の「通信チャネルの確立」を参照してください。

SSH 通信チャネル鍵の再生成

Detector モジュールが SSH 通信チャネル経由で通信する Guard を新しいものと交換した場合は、次の手順を実行して、SSH 通信チャネル鍵を再生成する必要があります。

ステップ 1　Detector モジュール上で **no host-keys** *ip-address-general* 設定モード コマンドを入力して、SSH ホスト鍵を Detector モジュールから削除します。

ip-address-general 引数には、リモート デバイスの IP アドレスを指定します。

Detector モジュールにリストされているホスト鍵を表示するには、**show host-keys** コマンドを使用します。

ステップ 2　次のいずれかの処理を実行して、リモート Guard で SSH 鍵を設定します。

- 新しい SSH 通信チャネルを Detector モジュールから確立します（P.4-22 の「通信チャネルの確立」を参照）。
- Detector モジュールの公開鍵をリモート Guard に手動で追加します。Detector モジュールの公開 SSH 鍵をコピーし、Guard が保持している SSH 鍵のリストにペーストすることができます。

  Detector モジュールの公開 SSH 鍵を表示するには、Detector モジュール上で **show public-key** コマンドを使用します。

  Detector モジュールの公開 SSH 鍵を、Guard が保持している SSH 鍵のリストに追加するには、Guard 上で **key add** コマンドを使用します。

通信チャネルの確立

Detector モジュールが Guard と直接通信できるようにするため、Detector モジュールと Guard との間に SSH または SSL 通信チャネルを確立します。

（注）　Detector モジュールと Guard の両方で通信チャネルをイネーブルにしてから、通信チャネルを確立する必要があります。通信チャネルのイネーブル化については、P.4-19 の「SSL 通信チャネルのイネーブル化」または P.4-22 の「SSH 通信チャネルのイネーブル化」を参照してください。

これら 2 つのデバイス間での初期接続中、Detector モジュールは通信チャネルの確保に必要な SSH 鍵と SSL 証明書を交換します。その後 Detector モジュールは接続を閉じ、Guard をアクティブにする、Guard とゾーン設定を同期させる（SSL 通信チャネルのみ）、または Guard にポーリングする（SSL 通信チャネルのみ）必要がある場合に、通信チャネルを確立します。

注意　Guard で TACACS+ 認証を使用してユーザを認証している場合、**key publish** コマンドを機能させるためには、TACACS+ サーバにユーザ riverhead を定義する必要があります。

Detector モジュールから Guard への通信チャネルを確立するには、Detector モジュール上で次の手順を実行します。

ステップ 1　設定モードで次のコマンドを入力して、SSH 秘密および公開鍵ペアを生成します。

```
key generate
```

SSH の鍵ペアがすでに存在している場合は、次のメッセージが表示されます。

```
/root/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

次のオプションのいずれかを入力して必要なオプションを選択します。

- **y**：Detector モジュールは新しい SSH 鍵ペアを生成します。
- **n**：Detector モジュールは新しい SSH 鍵ペアを生成しません。

ステップ 2　SSH 通信チャネルに必要な SSH 公開鍵だけをパブリッシュするか、または SSH 公開鍵をパブリッシュしてから、SSL 通信チャネルに必要な SSL 証明書を交換します。設定モードで、次のコマンドのいずれかを使用します。

- **key publish** *remote-guard-address* {**ssh** | **ssl**}
- **key publish ***

表 4-10 に、**key publish** コマンドの引数とキーワードを示します。

表 4-10　key publish コマンドの引数とキーワード

パラメータ	説明
remote-guard-address	リモート Guard の IP アドレス。
ssh	SSH 公開鍵を *remote-guard-address* 引数で定義されているリモート Guard にパブリッシュします。
ssl	SSH 公開鍵をパブリッシュし、SSL 証明書を生成して *remote-guard-address* 引数で定義されているリモート Guard と交換します。
*	SSH 鍵をパブリッシュして SSL 証明書を生成し、リモート Guard リストに設定されているすべての Guard と SSL 証明書を交換します。 Detector モジュールは、リモート Guard リスト内の各 Guard と SSH 通信チャネルを確立します。各リモート Guard について、ステップ 3 とステップ 4 を繰り返します。

ステップ 3　man-in-the-middle 攻撃（2 者間で交わされるメッセージを攻撃者が傍受して変更することが可能な攻撃）を防ぐために、SSH2 ではホスト鍵を使用してリモート ホスト、つまり Guard の確実性が確認されます。Guard に対して初めて SSH 通信チャネルを確立したときは、Guard はその公開鍵を Detector モジュールに送信します。これが Detector モジュールから Guard に対して開始される最初の接続である場合は、次のメッセージが表示されます。

```
The authenticity of host '<remote-hostname> (<remote-host IP
address>)' can't be established.
RSA key fingerprint is <RSA key fingerprint>
Are you sure you want to continue connecting (yes/no)?
```

yes と入力します。

次のプロンプト行が表示されます。

```
riverhead@remote-Guard-IP-address's password:
```

ステップ 4　Guard でユーザ riverhead 用に設定されたパスワードを入力します。

次の例で、SSH 秘密および公開鍵ペアを生成する方法と、Detector モジュールと IP アドレスが 192.168.100.33 のリモート Guard との間で SSH 通信チャネルを確立する方法について説明します。

```
user@DETECTOR-conf# key generate
user@DETECTOR-conf# key publish 192.168.100.33 ssh
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
The authenticity of host '192.168.100.33 (192.168.100.33)' can't be
established.
RSA key fingerprint is
de:cb:ac:36:9a:fe:33:f9:6a:d8:7b:98:a9:51:75:54.
Are you sure you want to continue connecting (yes/no)? yes
riverhead@192.168.100.33's password: <password>
user@DETECTOR-conf#
```

次の例では、リモート Guard リストに記載されている Guard モジュールとの通信チャネルを確立する方法について説明します。

```
user@DETECTOR-conf# key publish *
```

Detector モジュール SSH 公開鍵を表示するには、**show public-key** コマンドを使用します。

Detector モジュールにリストされているホスト鍵を表示するには、**show host-keys** コマンドを使用します。

SSH 鍵の管理

Detector モジュールは、安全なリモート ログインのために SSH をサポートしています。SSH 鍵のリストを追加すると、ログインとパスワードを入力しなくても、リモート デバイスから Detector モジュールに安全な通信ができます。

次の各項では、Detector モジュールの SSH 鍵リストの管理方法について説明します。

- SSH 鍵の追加
- SSH 鍵の削除

SSH 鍵の追加

ログイン名とパスワードを入力しない SSH 接続をイネーブルにするには、Detector モジュールの SSH 鍵リストにリモート接続の SSH 公開鍵を追加します。

設定モードで次のコマンドを入力します。

key add [*user-name*] {**ssh-dsa** | **ssh-rsa**} *key-string comment*

表 4-11 に、**key add** コマンドの引数とキーワードを示します。

表 4-11　key add コマンドの引数とキーワード

パラメータ	説明
user-name	（オプション）SSL 鍵の追加先ユーザの名前。他のユーザの SSH 鍵を追加できるのは管理者だけです。 デフォルトは、現行ユーザの SSH 鍵の追加です。
ssh-dsa	SSH2-DSA 鍵のタイプを指定します。
ssh-rsa	SSH2-RSA 鍵のタイプを指定します。
key-string	Guard またはリモート端末で作成された公開 SSH 鍵。鍵ストリングは、8,192 ビットまでに制限されています。 鍵タイプの識別（ssh-rsa または ssh-dsa）を除いた完全な鍵をコピーする必要があります。
comment	デバイスの説明。コメントの形式は、通常、鍵の生成に使用されるユーザとマシンを表す user@hostname になります。たとえば、Guard で生成される SSH 公開鍵に使用されるデフォルトのコメントは、root@GUARD です。

次の例は、SSH RSA 鍵を追加する方法を示しています。

```
user@DETECTOR-conf# key add ssh-rsa 14513797528175730. . .user@Detector module.com
```

SSH 鍵の削除

リストから SSH 鍵を削除できます。SSH 鍵を削除すると、次に Detector モジュールと SSH セッションを確立するときには認証を受ける必要があります。

Detector モジュールから SSH 鍵を削除するには、設定モードで次のコマンドを使用します。

key remove [*user-name*] *key-string*

表 4-12 に、**key remove** コマンドの引数を示します。

表 4-12　key remove コマンドの引数

パラメータ	説明
user-name	（オプション）SSH 鍵が削除されるユーザ名。 他のユーザの SSH 鍵を削除できるのは管理者だけです。デフォルトは、現行ユーザの SSH 鍵の削除です。
key-string	削除する公開 SSH 鍵。 プロンプトに SSH 公開鍵をペーストします。識別フィールド（ssh-rsa または ssh-dsa）は除き、鍵だけをペーストしてください。

次の例は、**key remove** コマンドにカット アンド ペーストを行えるように、ユーザ鍵を表示する方法を示しています。

```
user@DETECTOR-conf# show keys Lilac
ssh-rsa 2352345234523456... user@Detector module.com
user@DETECTOR-conf# key remove Lilac 2352345234523456...
```

SFTP 接続および SCP 接続用の鍵の設定

SSH2 の最上層にある Secure File Transfer Protocol（SFTP; セキュア ファイル転送プロトコル）、および SSH に依存する Secure Copy Protocol（SCP）は、ファイルのコピー方式を提供します。このコピー方式は、安全で信頼できます。SFTP および SCP では、公開鍵による認証と強力なデータ暗号化を使用しています。したがって、ログイン、データ、およびセッションの情報が送信中に傍受されたり変更されたりすることを防止できます。

SFTP 接続および SCP 接続用の鍵を設定するには、次の手順を実行します。

ステップ 1　設定モードで **show public-key** コマンドを入力して、Detector モジュール上で Detector モジュールの公開鍵を表示します。

鍵が存在する場合は、ステップ 2 を省略してステップ 3 に進みます。

鍵が存在しない場合は、ステップ 2 に進みます。

ステップ 2　設定モードで **key generate** コマンドを入力して、Detector モジュール上で秘密および公開鍵ペアを生成します。

注意　秘密および公開鍵ペアがすでに存在する場合は、再生成しないことをお勧めします。不必要に鍵ペアを再生成すると、現在オンラインになっていないリモート Guard と後で通信するときに問題が発生する可能性があります。秘密および公開鍵ペアを再生成する場合は、**key publish** コマンドを使用して、Detector モジュールのデフォルトのリモート Guard リストとゾーンのリモート Guard リストに設定されているすべてのリモート Guard にこの新しい公開鍵をパブリッシュする必要があります。

SSH の鍵ペアがすでに存在している場合は、次のメッセージが表示されます。

```
/root/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

y を入力して鍵を生成します。

Detector モジュールが公開および秘密鍵ペアを作成します。Detector モジュールの公開鍵を表示するには、設定モードで **show public-key** コマンドを使用します。

ステップ 3　公開鍵を Detector モジュールからコピーし、ネットワーク サーバ上の鍵ファイル内にペーストします。

たとえば、Linux オペレーティング システムにインストールされている ネットワーク サーバに username というユーザ アカウントで接続している場合は、Detector モジュールの公開鍵を /home/username/.ssh/authorized_keys2 ファイルに追加します。

鍵は 1 行に収まるようにコピーしてください。鍵が 2 行としてコピーされた場合は、1 行目の末尾にある改行文字を削除します。

（注）　公開鍵をコピーしてネットワーク　サーバ上の鍵ファイルにペーストしないと、自動エクスポート機能（**export reports** コマンドなど）が使用できず、手動でネットワーク サーバに接続するたびに、パスワードを入力する必要があります。

ホスト名の変更

Detector モジュールのホスト名を変更できます。この変更はすぐに反映され、新しいホスト名は自動的に CLI プロンプト ストリングに組み込まれます。

Detector モジュールのホスト名を変更するには、設定モードで次のコマンドを使用します。

hostname *name*

name 引数には、新しいホスト名を指定します。

次の例は、Detector モジュールのホスト名を変更する方法を示しています。

```
user@DETECTOR-conf# hostname CiscoDetector
admin@CiscoDetector-conf#
```

SNMP トラップのイネーブル化

Detector モジュールが SNMP トラップを送信し、Detector モジュールで発生する重大なイベントを管理者に通知するように設定することができます。また、Detector モジュールの SNMP トラップ ジェネレータのパラメータを設定し、Detector モジュールが報告する SNMP トラップ情報の範囲を定義することもできます。

トラップのログは、Detector モジュールのイベント ログに記録され、トラップ条件が発生すると、SNMP エージェントがトラップを送信するかどうかに関係なく、イベント モニタに表示されます。

Detector モジュールが SNMP トラップを送信するように設定するには、次の手順を実行します。

ステップ 1　設定モードで次のコマンドを入力して、SNMP トラップ ジェネレータ サービスをイネーブルにします。

```
service snmp-trap
```

ステップ 2　次のコマンドを入力して、SNMP トラップ ジェネレータのパラメータ（トラップの宛先 IP アドレスとトラップ情報の範囲）を設定します。

```
snmp trap-dest ip-address [community-string [min-severity]]
```

表 4-13 に、**snmp trap-dest** コマンドの引数を示します。

表 4-13　snmp trap-dest コマンドの引数

パラメータ	説明
ip-address	宛先ホストの IP アドレス。
community-string	（オプション）トラップとともに送信されるコミュニティ ストリング。このストリングは、宛先ホスト用に定義されたコミュニティ ストリングと一致する必要があります。デフォルトのコミュニティ ストリングは、*public* です。1 ～ 15 文字の英数字文字列を入力します。この文字列にスペースを含めることはできません。
min-severity	（オプション）トラップ情報のスコープ。重大度レベルの範囲の下限を指定してスコープを定義します。この定義により、トラップは指定された重大度レベル以上のすべてのイベントを表示します。たとえば、Warnings を指定すると、トラップは Warnings から Emergencies までのすべての重大度レベルのイベントを表示します。重大度レベルのオプションを次に示します。 • Emergencies：システムは使用不能（重大度 = 0）。 • Alerts：即座に処置が必要（重大度 = 1）。 • Critical：深刻な状態（重大度 = 2）。 • Errors：エラー状態（重大度 = 3）。 • Warnings：警告状態（重大度 = 4）。 • Notifications：正常ではあるが、重要な状態（重大度 = 5）。 • Informational：情報通知のメッセージ（重大度 = 6）。 • Debugging：デバッグ メッセージ（重大度 = 7）。 デフォルトでは、レポートにはすべての重大度レベルのイベントが表示されます。

SNMP トラップ ジェネレータ パラメータを削除するには、**no snmp trap-dest** コマンドを使用します。すべての SNMP トラップ宛先パラメータを削除するには、アスタリスク（*）を入力します。

次の例は、errors 以上の重大度レベルのトラップが、SNMP コミュニティ ストリング tempo とともに宛先 IP アドレス 192.168.100.52 に送信される例を示しています。

```
user@DETECTOR-conf# snmp trap-dest 192.168.100.52 tempo errors
```

表 4-14 に、Detector モジュールが生成する SNMP トラップを示します。

表 4-14　SNMP トラップ

SNMP トラップ	重大度	説明
rhExcessiveUtilizationTrap	EMERGENCY	すべての Detector モジュール ゾーン上で 150,000 個以上の動的フィルタが同時にアクティブになっているため、Detector モジュールは新しい動的フィルタを追加できない。
rhExcessiveUtilizationTrap	EMERGENCY	異常検出エンジン メモリが制限値に達した（90 % を超えた）。
rhGeneralTrap	EMERGENCY	ソフトウェア ライセンスがインストールされていないか、正しくない。このデバイスは動作しません。
rhGeneralTrap	EMERGENCY	インストールされているソフトウェア ライセンスを確認できない。このデバイスは動作しません。
rhGeneralTrap	EMERGENCY	インストールされているソフトウェア ライセンスの期限が明日（<dd-mmm-yyyy>）で切れる。
rhGeneralTrap	EMERGENCY	インストールされているソフトウェア ライセンスの期限が今日で切れる。このデバイスは動作しなくなります。
rhGeneralTrap	EMERGENCY	インストールされているソフトウェア ライセンスが無効：< 失敗メッセージ >。
rhGeneralTrap	ALERT	ディスク スペースが 80% である。
rhGeneralTrap	ALERT	インストールされているソフトウェア ライセンスの期限が 3 週間または 2 週間後の <dd-mmm-yyyy> に切れる。
rhGeneralTrap	ALERT	インストールされているソフトウェア ライセンスの期限が 1 週間後の <dd-mmm-yyyy> に切れる。
rhGeneralTrap	ALERT	インストールされているソフトウェア ライセンスの期限が 30、6、5、4、3、または 2 日後の <dd-mmm-yyyy> に切れる。
rhExcessiveUtilizationTrap	CRITICAL	ギガビット インターフェイス リンクの使用率（bps[1] 単位）が 85% を超えている。
rhExcessiveUtilizationTrap	CRITICAL	メモリ使用率が 85% を超えている。
rhExcessiveUtilizationTrap	CRITICAL	アクセラレータ カード CPU 使用率が 85% を超えている。
rhDynamicFilterTrap	ERROR	保留中の動的フィルタ数が 1,000 のため、新しい保留動的フィルタは廃棄される。

表 4-14　SNMP トラップ（続き）

SNMP トラップ	重大度	説明
rhProtectionTrap	ERROR	Detector モジュールは SSL 通信チャネルを使用してゾーンを保護するためのリモート Guard のアクティブ化に失敗した。
rhZoneGenericTrap	ERROR	Detector モジュールはゾーン設定の同期化に失敗した。
rhGeneralTrap	ERROR	Detector モジュールが次のようなゾーン異常検出のアクティブ化に失敗した。 • 検出またはラーニングから検出とラーニングまで。 • 検出とラーニングから検出またはラーニングへ。 Detector モジュールはゾーン異常検出とラーニング プロセスを非アクティブ化した。
rhDynamicFilterTrap	WARNING	Detector モジュールは動的フィルタの追加に失敗した。 このエラーは次のいずれかの場合に発生する可能性があります。 • アクティブな動的フィルタが多すぎる。 • 動的フィルタのアクションはリモートのアクティブ化であり、Detector モジュールはリモート Guard リストに記載されているリモート Guard と接続できなかった。
rhExcessiveUtilizationTrap	WARNING	Detector モジュールに、すべてのゾーン上で同時にアクティブになっている動的フィルタが 135,000 個以上ある。アクティブな動的フィルタの数が 15,000 に到達すると、Detector モジュールは新しい動的フィルタを追加できなくなります。
rhGeneralTrap	WARNING	ディスク スペースが 75% である。
rhPolicyConstructionTrap	WARNING	ラーニング プロセスのポリシー構築フェーズが失敗した。
rhDetectionTrap	WARNING	Detector モジュールはゾーン異常検出の開始に失敗した。
rhReloadTrap	WARNING	Detector モジュールは再起動した。トラップには、MIB2 ウォーム スタート トラップまたはコールド スタート トラップと、Detector モジュールが再始動した原因に関する情報が含まれています。
rhReloadTrap	WARNING	Detector モジュールがシャットダウンした。トラップには、MIB2 ウォーム スタート トラップまたはコールド スタート トラップと、Detector モジュールがシャットダウンした原因に関する情報が含まれています。
rhThresholdTuningTrap	WARNING	ラーニング プロセスのしきい値調整フェーズが失敗した。

表 4-14　SNMP トラップ（続き）

SNMP トラップ	重大度	説明
rhAttackTrap	NOTIFICATIONS	攻撃が開始した。
rhAttackTrap	NOTIFICATIONS	攻撃が終了した。
rhPolicyConstructionTrap	NOTIFICATIONS	ラーニング プロセスのポリシー構築フェーズが開始された。
rhPolicyConstructionTrap	NOTIFICATIONS	ラーニング プロセスのポリシー構築フェーズが受け入れられた。
rhPolicyConstructionTrap	NOTIFICATIONS	ラーニング プロセスのポリシー構築フェーズが停止された。
rhDetectionTrap	NOTIFICATIONS	ゾーン異常検出が開始した。
rhDetectionTrap	NOTIFICATIONS	ゾーン異常検出が終了した。
rhProtectionTrap	NOTIFICATIONS	Detector モジュールは SSL 通信チャネルを使用してゾーンを保護するためのリモート Guard のアクティブ化に成功している。
rhThresholdTuningTrap	NOTIFICATIONS	ラーニング プロセスのしきい値調整フェーズが開始された。
rhThresholdTuningTrap	NOTIFICATIONS	ラーニング プロセスのしきい値調整フェーズが受け入れられた。
rhThresholdTuningTrap	NOTIFICATIONS	ラーニング プロセスのしきい値調整フェーズが停止された。
rhZoneGenericTrap	NOTIFICATIONS	Detector モジュールはゾーン設定の同期化を開始した。
rhZoneTrap	NOTIFICATIONS	新しいゾーンが作成された。
rhZoneTrap	NOTIFICATIONS	ゾーンが削除された。
rhDynamicFilterControlTrap	INFO	Detector モジュールが特定のポリシーに送信しなかった攻撃検出イベントの数。
rhDynamicFilterControlTrap	INFO	Detector モジュールは、アクティブな動的フィルタが 1,000 個以上あるため、削除する動的フィルタのトラップを送信しない。
rhDynamicFilterTrap	INFO	動的フィルタが追加された。
rhDynamicFilterTrap	INFO	動的フィルタが削除された。
rhDynamicFilterTrap	INFO	保留動的フィルタが追加された。

1. bps = bits per second（ビット / 秒）

SNMP コミュニティ ストリングの設定

Detector モジュールの SNMP サーバにアクセスすることにより、管理情報ベース 2（MIB2）および Cisco Riverhead 専用 MIB で定義された情報を取得することができます。コミュニティ ストリングはパスワードのように機能し、Detector モジュールの SNMP エージェントからの読み取りアクセスを許可します。Detector モジュールの SNMP コミュニティ ストリングを設定して、異なる組織のクライアントがそれぞれ異なるコミュニティ ストリングを使用して SNMP エージェントにアクセスできるようにすることができます。

SNMP コミュニティ ストリングを追加するには、設定モードで次のコマンドを使用します。

snmp community *community-string*

community-string 引数には、目的の Detector モジュールのコミュニティ ストリングを指定します。1 ～ 15 文字の英数字文字列を入力します。この文字列にスペースを含めることはできません。Detector モジュールのデフォルトのコミュニティ ストリングは riverhead です。コミュニティ名はいくつでも指定できます。コミュニティ ストリングを削除するには、**no community string** コマンドを使用します。すべての SNMP コミュニティ ストリングを削除するには、アスタリスク (*****) を入力します。

次の例は、SNMP コミュニティ ストリングを設定する方法を示しています。

```
user@DETECTOR-conf# snmp community tempo
```

ログイン バナーの設定

ログイン バナーとは、SSH セッション、コンソール ポート接続、または Detector モジュールへの WBM セッションを開いたときに、ユーザ認証の前の画面に表示されるテキストのことです。

認可されていないアクセスに対してユーザに警告したり、適切と見なされるシステムの使用法を説明したり、不適切な使用法や不正な活動を検出するためにシステムが監視されていることをユーザに警告したりするように、ログイン バナーを設定できます。

Detector モジュールは、次の場所にログイン バナーを表示します。

- CLI : パスワード ログイン プロンプトの前、またはポップアップ ウィンドウとして（使用している SSH クライアントによって異なる）。
- WBM : Detector モジュールのログイン ウィンドウの右側。

この項では、次のトピックについて取り上げます。

- CLI からのログイン バナーの設定
- ログイン バナーのインポート
- ログイン バナーの削除

CLI からのログイン バナーの設定

login-banner コマンドを使用すると、単一または複数のメッセージ バナーを作成できます。複数のログイン バナーを入力した場合、新しいログイン バナーは、既存のログイン バナーの最後に新しい行として追加されます。

ログイン バナーを設定するには、設定モードで次のコマンドを使用します。

login-banner *banner-str*

banner-str 引数には、バナーのテキストを指定します。文字列の長さは最大 999 文字です。式にスペースを使用する場合は、式を引用符（" "）で囲みます。

ログイン バナーを表示するには、**show login-banner** コマンドを使用します。

次の例は、ログイン バナーを設定して表示する方法を示しています。

```
user@DETECTOR-conf# login-banner "Welcome to the Cisco Traffic Anomaly Detector
Module"
user@DETECTOR-conf# login-banner "Unauthorized access is prohibited."
user@DETECTOR-conf# login-banner "Contact sysadmin@corp.com for access."
user@DETECTOR-conf# show login banner
Welcome to the Cisco Traffic Anomaly Detector Module
Unauthorized access is prohibited.
Contact sysadmin@corp.com for access.
```

ログイン バナーのインポート

グローバル モードまたは設定モードで次のいずれかのコマンドを入力して、ネットワーク サーバからテキスト ファイルをインポートし、既存のログイン バナーを差し替えることができます。

- **copy ftp login-banner** *server full-file-name* [*login* [*password*]]
- **copy** {**sftp** | **scp**} **login-banner** *server full-file-name login*

インポートするファイル内の各行の最大長は、999 文字です。

SFTP および SCP はセキュアな通信を SSH に依存しているので、**sftp** または **scp** オプションを使用して **copy** コマンドを入力する前に、Detector モジュールで使用する鍵を設定していない場合、Detector モジュールによってパスワードの入力が求められます。Detector モジュールがセキュアな

通信のために使用する鍵を設定する方法の詳細については、P.4-27 の「SFTP 接続および SCP 接続用の鍵の設定」を参照してください。

表 4-15 に、**copy login-banner** コマンドの引数とキーワードを示します。

表 4-15　copy login-banner コマンドの引数とキーワード

パラメータ	説明
ftp	FTP を指定します。
sftp	SFTP を指定します。
scp	SCP を指定します。
server	ネットワーク サーバの IP アドレス。IP アドレスをドット区切り 10 進表記で入力します（たとえば 192.168.10.2）。
full-file-name	ファイルの完全な名前。パスを指定しない場合、サーバはユーザのホーム ディレクトリからファイルをコピーします。
login	サーバのログイン名。 *login* 引数は、FTP サーバを定義するときは省略可能です。ログイン名を入力しない場合、FTP サーバは匿名ログインであると想定し、パスワードを要求しません。
password	（オプション）リモート FTP サーバのパスワード。パスワードを入力しない場合、Detector モジュールによってパスワードを要求されます。

次の例は、FTP サーバからログイン バナーをインポートする方法を示しています。

```
user@DETECTOR-conf# copy ftp login-banner 10.0.0.191 /root/login-banner <user>
<password>
```

ログイン バナーの削除

ユーザ認証の前にメッセージを表示する必要がなくなった場合、ログイン バナーを削除できます。

ログイン バナーを削除するには、設定モードで **no login-banner** コマンドを使用します。

次の例は、ログイン バナーを削除する方法を示しています。

```
user@DETECTOR-conf# no login-banner
```

Web-Based Manager ロゴの設定

エンドユーザ インターフェイスをカスタマイズするために、企業のロゴまたはカスタマイズされたロゴを Web-Based Manager（WBM）Web ページに追加することができます。

新しいロゴは、次の場所に表示されます。

- Detector モジュールのログイン ページで、Cisco Systems ロゴの下。
- すべての WBM ページ（Detector モジュールのログイン ページは除く）で、Cisco Systems ロゴの右側。

新しいロゴは GIF 形式である必要があります。新しいロゴのサイズは、幅 = 87 ピクセル、高さ = 41 ピクセルにすることをお勧めします。

この項では、次のトピックについて取り上げます。

- WBM ロゴのインポート
- WBM ロゴの削除

WBM ロゴのインポート

WBM で使用する新しいロゴをネットワーク サーバからインポートするには、グローバル モードまたは設定モードで次のコマンドを入力します。

- **copy ftp wbm-logo** *server full-file-name* [*login* [*password*]]
- **copy** {**sftp** | **scp**} **wbm-logo** *server full-file-name login*

SFTP および SCP はセキュアな通信では SSH を使用するため、Detector モジュールは **sftp** オプションまたは **scp** オプションを使用して **copy** コマンドを入力する前に Detector モジュールが使用する鍵を設定しない場合、パスワードの入力を求めます。Detector モジュールがセキュアな通信のために使用する鍵を設定する方法の詳細については、P.4-27 の「SFTP 接続および SCP 接続用の鍵の設定」を参照してください。

表 4-16 に、**copy wbm-logo** コマンドの引数とキーワードを示します。

表 4-16　copy wbm-logo コマンドの引数とキーワード

パラメータ	説明
ftp	FTP を指定します。
sftp	SFTP を指定します。
scp	SCP を指定します。
server	ネットワーク サーバの IP アドレス。IP アドレスをドット区切り 10 進表記で入力します（たとえば 192.168.10.2）。
full-file-name	GIF ファイル拡張子を含む、完全なファイル名。パスを指定しない場合、サーバはユーザのホーム ディレクトリからファイルをコピーします。
login	（オプション）サーバのログイン名。*login* 引数は、FTP サーバを定義するときは省略可能です。ログイン名を入力しない場合、FTP サーバは匿名ログインであると想定し、パスワードを要求しません。
password	（オプション）リモート FTP サーバのパスワード。パスワードを入力しない場合、Detector モジュールによってパスワードを要求されます。

次の例は、FTP サーバから WBM ロゴ ファイルをインポートする方法を示しています。

```
user@DETECTOR-conf# copy ftp wbm-logo 10.0.0.191 /root/WBMlogo.gif <user> <password>
```

WBM ロゴの削除

WBM ロゴを削除するには、設定モードで **no wbm-logo** コマンドを使用します。

次の例は、WBM ロゴを削除する方法を示しています。

```
user@DETECTOR-conf# no wbm-logo
```

セッション タイムアウトの設定

セッション タイムアウトとは、アクティビティが何もない状態でセッションがアクティブでいられる時間のことです。設定された時間内に何もアクティビティがなかった場合、タイムアウトが発生し、再びログインする必要があります。セッション タイムアウトは、デフォルトではディセーブルになっています。

セッション タイムアウトは CLI にのみ適用され、WBM には適用されません。

設定モードで次のコマンドを入力して、Detector モジュールがアイドル セッションを自動的に切断するまでの分数を設定できます。

session-timeout *timeout-val*

timeout-val 引数には、Detector モジュールが自動的にアイドル セッションを切断するまでの分数を指定します。有効な値は、1 ～ 1,440 分（1 日）です。

次の例は、Detector モジュールが 10 分後にアイドル セッションを切断するように設定する方法を示しています。

```
user@DETECTOR-conf# session-timeout 10
```

Detector モジュールが自動的にアイドル セッションを切断しないようにするには、**no session-timeout** コマンドを使用します。

セッション タイムアウトの値を表示するには、**show session-timeout** コマンドを使用します。