セキュア シェル バージョン 2 サポート

セキュア シェル バージョン 2 サポート機能を使用して、Secure Shell（SSH; セキュア シェル）バージョン 2 を設定できます（SSH バージョン 1 のサポートは、以前の Cisco IOS ソフトウェア リリースで実装されていました）。SSH は信頼できる転送レイヤの上位で実行され、強化認証および暗号化機能を実現します。SSH では、信頼できる転送として定義されているのは TCP のみです。SSH で、ネットワーク上の他のコンピュータに安全にアクセスしたり、コマンドを安全に実行できます。SSH とともに提供される Secure Copy Protocol（SCP; セキュア コピー プロトコル）機能で、ファイルを安全に転送できます。

機能情報の確認

ご使用のソフトウェア リリースでは、このモジュールで説明されるすべての機能がサポートされているとは限りません。最新の機能情報と注意事項については、ご使用のプラットフォームとソフトウェア リリースに対応したリリース ノートを参照してください。この章に記載されている機能の詳細、および各機能がサポートされているリリースのリストについては、「セキュア シェル バージョン 2 サポートの機能情報」（P.26）を参照してください。

Cisco Feature Navigator を使用すると、プラットフォーム、および Cisco ソフトウェア イメージの各サポート情報を検索できます。Cisco Feature Navigator には、http://www.cisco.com/go/cfn からアクセスします。Cisco.com のアカウントは必要ありません。

この章の構成

- 「セキュア シェル バージョン 2 サポートの前提条件」（P.2）
- 「セキュア シェル バージョン 2 サポートの制約事項」（P.2）
- 「セキュア シェル バージョン 2 サポートに関する情報」（P.2）
- 「セキュア シェル バージョン 2 サポートの設定方法」（P.5）
- 「セキュア シェル バージョン 2 サポートの設定例」（P.19）
- 「関連情報」（P.24）
- 「その他の参考資料」（P.24）
- 「セキュア シェル バージョン 2 サポートの機能情報」（P.26）

セキュア シェル バージョン 2 サポートの前提条件

SSH の設定前に、次のタスクを行ってください。

- 必要なイメージをルータにダウンロードします。SSH サーバでは、Cisco IOS Release 12.3(4)T、12.2(25)S、または 12.3(7)JA から k9（Triple Data Encryption Standard（3DES））ソフトウェア イメージをルータにダウンロードする必要があります。

（注） SSH Version 2 サーバは Cisco IOS Release 12.3(4)T、12.3(2)XE、12.2(25)S、および 12.3(7)JA でサポートされます。SSH Version 2 クライアントは Cisco IOS Release 12.3(7)T からサポートされ、Cisco IOS Release 12.3(7)JA でサポートされています（SSH クライアントは SSH バージョン 1 プロトコルおよびバージョン 2 プロトコルの両方を実行し、Cisco IOS Release 12.3(4)T の k8 および k9 両方のイメージでサポートされます）。

ソフトウェア イメージのダウンロードの詳細については、『*Cisco IOS Configuration Fundamentals Configuration Guide*, Release 12.4T』および『*Cisco IOS Network Management Configuration Guide*, Release 15.0』を参照してください。

セキュア シェル バージョン 2 サポートの制約事項

- SSH サーバおよび SSH クライアントは、3DES ソフトウェア イメージでサポートされます。
- サポートされるアプリケーションは、実行シェル、remote コマンドの実行、および SCP のみです。
- Rivest、Shamir、および Adelman（RSA）キー生成は SSH サーバ サイドの要件です。SSH クライアントとして動作するルータは、RSA キーを生成する必要がありません。
- RSA キー ペアのサイズは、768 以上である必要があります。
- 次の機能はサポートされていません。
  - ポート フォワーディング
  - 圧縮

セキュア シェル バージョン 2 サポートに関する情報

- 「セキュア シェル バージョン 2」（P.2）
- 「セキュア シェル バージョン 2 の機能拡張」（P.3）
- 「セキュア シェル バージョン 2 の RSA キーに関する機能拡張」（P.3）
- 「SNMP トラップ生成」（P.4）
- 「SSH キーボード インタラクティブ認証」（P.5）

セキュア シェル バージョン 2

セキュア シェル バージョン 2 サポート機能で、SSH バージョン 2 を設定できます。

SSH バージョン 2 サーバの設定は、SSH バージョン 1 の設定と同様です。**ip ssh version** コマンドは、設定する SSH バージョンを定義できるように導入されました。このコマンドを設定しない場合、デフォルトで SSH は互換モードで実行されます。バージョン 1 とバージョン 2 両方の接続が利用できます。

（注） SSH バージョン 1 は、標準として定義されていないプロトコルです。ルータを定義されていないプロトコル（バージョン 1）に戻したくない場合は、**ip ssh version** コマンドを使用してバージョン 2 を指定してください。

ip ssh rsa keypair-name コマンドも、Cisco IOS Release 12.3(4)T で導入されたため、設定した RSA キーを使用して SSH 接続をイネーブルにできます。すでに、SSH は生成済みの最初の RSA キーにリンクされています（つまり、最初の RSA キー ペアが生成された時点で SSH はイネーブルになっています）。この動作は存在していますが、**ip ssh rsa keypair-name** コマンドを使用してこの動作を行わないようにすることができます。**ip ssh rsa keypair-name** コマンドを、キー ペアの名前を使用して設定する場合、SSH はキー ペアが存在する場合にイネーブルになるか、キー ペアを後で作成する場合は後からイネーブルになります。このコマンドを使用して SSH をイネーブルにする場合、Cisco IOS ソフトウェアの SSH バージョン 1 では必要な、ホスト名とドメイン名を設定を設定する必要はありません。

（注） ログイン バナーは SSH バージョン 2 でサポートされますが､セキュア シェル バージョン 1 ではサポートされません。

セキュア シェル バージョン 2 の機能拡張

SSH バージョン 2 の機能拡張には、VRF aware SSH、SSH デバッグ機能拡張、および Diffie-Hellman（DH）グループ交換のサポートなどの、追加機能がいくつか含まれています。

Cisco IOS SSH 実装では従来、768 ビット絶対値が使用されていましたが、DH グループ 14（2048 ビット）およびグループ 16（4096 ビット）暗号化アプリケーションに対応するため、より大きなキー サイズの必要性が高まり、優先 DH グループを確立するクライアントとサーバ間のメッセージ交換が必要になっています。**ip ssh dh min size** コマンドが Cisco IOS Release 12.4(20)T で導入され、SSH サーバの絶対サイズを設定できるようになりました。これに加え、**ssh** コマンドが拡張され、SSH クライアント サイドのクライアントの VRF インスタンス名を IP アドレスとともに使用して、正しいルーティング テーブルを検索し、接続を確立する機能に、VRF 認識が追加されました。

SSH debug コマンドが修正され、デバッグが拡張されました。**debug ip ssh** コマンドが拡張され、デバッグ処理を簡素化できるようになりました。これまでは、このコマンドで、SSH に関連するデバッグ メッセージが、特に必要なものとそうでないものにかかわらずすべて印刷されていました。この動作は依然として存在しますが、**debug ip ssh** コマンドをキーワードを指定して設定した場合、メッセージが、キーワードで指定した情報に制限されます。

セキュア シェル バージョン 2 の RSA キーに関する機能拡張

Cisco IOS SSH バージョン 2（SSHv2）は、キーボード インタラクティブでパスワード ベースの認証方式をサポートしています。RSA キーの SSHv2 拡張機能は、クライアントとサーバ向けの RSA ベースの公開キー認証もサポートしています。

ユーザ認証：RSA ベースのユーザ認証では、各ユーザに関連付けられている秘密キー / 公開キーのペアを認証に使用します。ユーザは秘密キー / 公開キーのペアをクライアントで生成し、公開キーを Cisco IOS SSH サーバで設定して、認証を完了します。

クレデンシャルの確立を試行する SSH ユーザは、秘密キーを使用して暗号化されたシグニチャを提示します。シグニチャとユーザの公開キーは、認証のために SSH サーバに送信されます。SSH サーバでは、ユーザから提示された公開キーに対してハッシュを計算します。ハッシュは、サーバに一致するエントリがあるかどうかを判断するために使用されます。一致が見つかった場合、RSA ベースのメッセージ検証が公開キーを使用して実行されます。その結果、暗号化されたシグニチャに基づいて、ユーザのアクセスは認証されるか拒否されます。

サーバ認証：SSH セッションの確立中に、Cisco IOS SSH クライアントは、キー交換フェーズ中に使用できるサーバ ホスト キーを使用して、SSH サーバを認証します。SSH サーバ キーは、SSH サーバの識別に使用されます。これらのキーは SSH がイネーブルになるときに作成され、クライアント側で設定する必要があります。

サーバ認証の場合、Cisco IOS SSH クライアントが各サーバにホスト キーを割り当てる必要があります。クライアントがサーバとの間で SSH セッションを確立しようとすると、キー交換メッセージの一部として、サーバのシグニチャを受信します。厳密なホスト キーのチェック フラグがクライアント側でイネーブルの場合、そのサーバに対応するホスト キー エントリがあるかどうかがクライアントで確認されます。一致が見つかると、クライアントはサーバ ホスト キーを使用してシグニチャの検証を試行します。サーバの認証に成功すると、セッションの確立処理は続行します。失敗すると、処理は終了し、「Server Authentication Failed」メッセージが表示されます。

（注） 公開キーをサーバで格納する際、メモリを使用します。したがって、SSH サーバで設定できる公開キーの数は、1 ユーザに最大 2 つの公開キーを作成した場合 10 ユーザ分に限られます。

（注）

- Cisco IOS サーバは RSA ベースのユーザ認証をサポートしていますが、Cisco IOS クライアントは認証方式として公開キーを提案できません。RSA ベースの認証に対するオープンな SSH クライアントからの要求を Cisco IOS サーバが受信した場合、サーバは認証要求を受け入れます。
- サーバ認証の場合、サーバの RSA 公開キーを手動で設定し、Cisco IOS SSH クライアント側で **ip ssh stricthostkeycheck** コマンドを設定します。

SNMP トラップ生成

Cisco IOS Release 12.4(17) では、Simple Network Management Protocol（SNMP; 簡易ネットワーク管理プロトコル）トラップは、トラップがイネーブルで SNMP デバッグがオンになっている場合、SSH セッションが終了した際に自動的に生成されます。SNMP トラップのイネーブル化の詳細については『*Cisco IOS Network Management Configuration Guide*, Release 15.0』の「Configuring SNMP Support」モジュールを参照してください。

（注） **snmp-server host** コマンドを設定する場合、IP アドレスは SSH（Telnet）クライアントがあり、SSH サーバへの IP 接続が可能な PC のアドレスにしてください。SNMP トラップ生成設定の例については、「例：SNMP トラップの設定」（P.20）を参照してください。

また、SNMP デバッグを **debug snmp packet** コマンドを使用してオンにし、トラップを表示する必要があります。トラップ情報には、送信バイト数や SSH セッションで使用されたプロトコルなどの情報が含まれます。SNMP デバッグの例については、「例：SNMP のデバッグ」（P.22）を参照してください。

SSH キーボード インタラクティブ認証

SSH キーボード インタラクティブ認証機能は、SSH での汎用メッセージ認証とも呼ばれ、異なる種類の認証メカニズムを実装するために使用できる方式です。基本的に、現在サポートされている、ユーザの入力のみが必要な認証方式はすべて、この機能で実行することができます。この機能は自動的にイネーブルになります。

次の方式がサポートされています。

- パスワード
- サーバが送信するチャレンジ に応答する番号またはストリングを印刷する SecurID およびハードウェア トークン
- Pluggable Authentication Module（PAM; プラグイン可能な認証モジュール）
- S/KEY（およびその他の使い捨てキー）

自動的にイネーブルにされた SSH キーボード インタラクティブ認証機能のさまざまなシナリオの例については、「例：SSH キーボード インタラクティブ認証」（P.20）を参照してください。

セキュア シェル バージョン 2 サポートの設定方法

- 「ホスト名およびドメイン名を使用した SSH バージョン 2 のルータ設定」（P.5）（必須）
- 「RSA キー ペアを使用した SSH バージョン 2 のルータ設定」（P.6）（任意）
- 「RSA ベースのユーザ認証を実行するための Cisco IOS SSH サーバの設定」（P.7）（任意）
- 「RSA ベースのサーバ認証を実行するための Cisco IOS SSH サーバの設定」（P.9）（任意）
- 「リモート デバイスでの暗号化セッションの開始」（P.11）（任意）
- 「SSH サーバでのセキュア コピー プロトコルのイネーブル化」（P.12）（任意）
- 「show ssh コマンドを使用したセキュア シェル接続のステータスの確認」（P.13）（任意）
- 「セキュア シェル ステータスの確認」（P.15）（任意）
- 「セキュア シェル バージョン 2 のモニタリングと維持」（P.16）（任意）

ホスト名およびドメイン名を使用した SSH バージョン 2 のルータ設定

このタスクを実行して、SSH バージョン 2 のルータを、ホスト名とドメイン名を使用して設定します。また、SSH バージョン 2 を、RSA キー ペアの設定を使用して設定することもできます（「RSA キー ペアを使用した SSH バージョン 2 のルータ設定」（P.6）を参照）。

手順の概要

1. **enable**
2. **configure terminal**
3. **hostname** *hostname*
4. **ip domain-name** *name*
5. **crypto key generate rsa**
6. **ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*]

7. **ip ssh version** [**1** | **2**]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** **例：** `Router> enable`	特権 EXEC モードをイネーブルにします。 • プロンプトが表示されたら、パスワードを入力します。
ステップ 2	**configure terminal** **例：** `Router# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**hostname** *hostname* **例：** `Router(config)# hostname cisco 7200`	ルータのホスト名を設定します。
ステップ 4	**ip domain-name** *name* **例：** `Router(config)# ip domain-name example.com`	ルータのドメイン名を設定します。
ステップ 5	**crypto key generate rsa** **例：** `Router(config)# crypto key generate rsa`	ローカルおよびリモート認証用の SSH サーバをイネーブルにします。
ステップ 6	**ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*] **例：** `Router(config)# ip ssh time-out 120`	（任意）SSH コントロール変数をルータに設定します。
ステップ 7	**ip ssh version** [**1** | **2**] **例：** `Router(config)# ip ssh version 1`	（任意）ルータで実行する SSH のバージョンを指定します。

RSA キー ペアを使用した SSH バージョン 2 のルータ設定

このタスクを実行して、ホスト名またはドメイン名を設定することなく SSH バージョン 2 をイネーブルにします。SSH バージョン 2 は、設定するキー ペアがすでに存在する場合や、後で作成する場合は、後でイネーブルになります。また、SSH バージョン 2 をホスト名とドメイン名を設定して設定することもできます（「ホスト名およびドメイン名を使用した SSH バージョン 2 のルータ設定」（P.5）を参照）。

手順の概要

1. **enable**
2. **configure terminal**
3. **ip ssh rsa keypair-name** *keypair-name*

4. **crypto key generate rsa usage-keys label** *key-label* **modulus** *modulus-size*
5. **ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*]
6. **ip ssh version 2**

手順の詳細

ステップ 1	**enable** **例：** `Router> enable`	特権 EXEC モードをイネーブルにします。 • プロンプトが表示されたら、パスワードを入力します。
ステップ 2	**configure terminal** **例：** `Router# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**ip ssh rsa keypair-name** *keypair-name* **例：** `Router(config)# ip ssh rsa keypair-name sshkeys`	SSH を使用する際に使用する RSA キー ペアを指定します。 **（注）** Cisco IOS ルータでは、複数の RSA キー ペアを指定することができます。
ステップ 4	**crypto key generate rsa usage-keys label** *key-label* **modulus** *modulus-size* **例：** `Router(config)# crypto key generate rsa usage-keys label sshkeys modulus 768`	ルータでローカルおよびリモート認証を行う SSH サーバをイネーブルにします。 • SSH バージョン 2 では、絶対サイズは 768 ビット以上である必要があります。 **（注）** RSA キー ペアを削除するには、**crypto key zeroize rsa** コマンドを使用します。RSA キー ペアを削除すると、SSH サーバも自動的にディセーブルになります。
ステップ 5	**ip ssh** [**time-out** *seconds*	**authentication-retries** *integer*] **例：** `Router(config)# ip ssh time-out 12`	SSH コントロール変数をルータに設定します。
ステップ 6	**ip ssh version 2** **例：** `Router(config)# ip ssh version 2`	ルータで実行する SSH のバージョンを指定します。

RSA ベースのユーザ認証を実行するための Cisco IOS SSH サーバの設定

このタスクを実行して、RSA ベースのユーザ認証を実行するように Cisco IOS SSH サーバを設定します。サーバに保存されている RSA 公開キーが、クライアントに保存されている公開キーと秘密キーのペアを使用して検証されると、ユーザ認証は成功です。

手順の概要

1. **enable**
2. **configure terminal**
3. **hostname** *name*

4. **ip domain-name** *name*
5. **crypto key generate rsa**
6. **ip ssh pubkey-chain**
7. **username** *username*
8. **key-string**
9. **exit**
10. **key-hash** *key-type key-name*
11. **end**

手順の詳細

ステップ 1	**enable** **例：** `Router> enable`	特権 EXEC モードをイネーブルにします。 • プロンプトが表示されたら、パスワードを入力します。
ステップ 2	**configure terminal** **例：** `Router# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**hostname** *name* **例：** `Router(config)# hostname host1`	ホスト名を指定します。
ステップ 4	**ip domain-name** *name* **例：** `Router(config)# ip domain-name name1`	Cisco IOS ソフトウェアで使用するデフォルトのドメイン名を定義し、不完全なホスト名のドメインを補完します。
ステップ 5	**crypto key generate rsa** **例：** `Router(config)# crypto key generate rsa`	RSA キー ペアを生成します。
ステップ 6	**ip ssh pubkey-chain** **例：** `Router(config)# ip ssh pubkey-chain`	SSH サーバ上のユーザおよびサーバ認証用に SSH-RSA キーを設定し、公開キー コンフィギュレーション モードを開始します。
ステップ 7	**username** *username* **例：** `Router(conf-ssh-pubkey)# username user1`	SSH ユーザ名を設定し、公開キー ユーザ コンフィギュレーション モードを開始します。
ステップ 8	**key-string** **例：** `Router(conf-ssh-pubkey-user)# key-string`	リモート ピアの RSA 公開キーを指定し、公開キー データ コンフィギュレーション モードを開始します。 **（注）** オープン SSH クライアントから（言い換えると .ssh/id_rsa.pub ファイルから）公開キー値を取得できます。
ステップ 9	**exit** **例：** `Router(conf-ssh-pubkey-data)# exit`	公開キー ユーザ コンフィギュレーション モードを終了します。

ステップ 10	**key-hash** *key-type key-name* **例：** `Router(conf-ssh-pubkey-data)# key-hash ssh-rsa key1`	（任意）SSH キー タイプとバージョンを指定します。 • キー タイプは、秘密公開キー ペアの設定では ssh-rsa である必要があります。 • この手順が任意なのは、**key-string** コマンドが設定されている場合のみです。 • **key-string** コマンドまたは **key-hash** コマンドを設定する必要があります。 **（注）** 公開キー ストリングのハッシュを計算するには、ハッシュ処理ソフトウェアを使用します。また、別の Cisco IOS ルータからのハッシュ値をコピーすることもできます。初めて公開キー データを入力する場合、**key-string** コマンドを使用して公開キー データを入力することを推奨します。
ステップ 11	**end** **例：** `Router(conf-ssh-pubkey-data)# end`	現在のモードを終了し、特権 EXEC モードに戻ります。

RSA ベースのサーバ認証を実行するための Cisco IOS SSH サーバの設定

このタスクを実行して、RSA ベースのサーバ認証を実行するように Cisco IOS SSH クライアントを設定します。

手順の概要

1. **enable**
2. **configure terminal**
3. **hostname** *name*
4. **ip domain-name** *name*
5. **crypto key generate rsa**
6. **ip ssh pubkey-chain**
7. **server** *server-name*
8. **key-string**
9. **exit**
10. **key-hash** *key-type key-name*
11. **end**
12. **configure terminal**
13. **ip ssh stricthostkeycheck**

手順の詳細

ステップ 1	**enable** **例：** `Router> enable`	特権 EXEC モードをイネーブルにします。 • プロンプトが表示されたら、パスワードを入力します。
ステップ 2	**configure terminal** **例：** `Router# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**hostname** *name* **例：** `Router(config)# hostname host1`	ホスト名を指定します。
ステップ 4	**ip domain-name** *name* **例：** `Router(config)# ip domain-name name1`	Cisco IOS ソフトウェアで使用するデフォルトのドメイン名を定義し、不完全なホスト名のドメインを補完します。
ステップ 5	**crypto key generate rsa** **例：** `Router(config)# crypto key generate rsa`	RSA キー ペアを生成します。
ステップ 6	**ip ssh pubkey-chain** **例：** `Router(config)# ip ssh pubkey-chain`	SSH サーバ上のユーザおよびサーバ認証用に SSH-RSA キーを設定し、公開キー コンフィギュレーション モードを開始します。
ステップ 7	**server** *server-name* **例：** `Router(conf-ssh-pubkey)# server server1`	ルータでの公開キー認証について SSH サーバをイネーブルにし、公開キー サーバ コンフィギュレーション モードを開始します。
ステップ 8	**key-string** **例：** `Router(conf-ssh-pubkey-server)# key-string`	リモート ピアの RSA 公開キーを指定し、公開キー データ コンフィギュレーション モードを開始します。 **（注）** オープン SSH クライアントから（言い換えると .ssh/id_rsa.pub ファイルから）公開キー値を取得できます。
ステップ 9	**exit** **例：** `Router(conf-ssh-pubkey-data)# exit`	公開キー データ コンフィギュレーション モードを終了し、公開キー サーバ コンフィギュレーション モードを開始します。

ステップ 10	**key-hash** *key-type key-name* **例：** `Router(conf-ssh-pubkey-server)# key-hash ssh-rsa key1`	（任意）SSH キー タイプとバージョンを指定します。 • 秘密キー / 公開キー ペアの設定では、キー タイプを ssh-rsa にする必要があります。 • この手順が任意なのは、**key-string** コマンドが設定されている場合のみです。 • **key-string** コマンドまたは **key-hash** コマンドを設定する必要があります。 **（注）** 公開キー ストリングのハッシュを計算するには、ハッシュ処理ソフトウェアを使用します。また、別の Cisco IOS ルータからのハッシュ値をコピーすることもできます。初めて公開キー データを入力する場合、**key-string** コマンドを使用して公開キー データを入力することを推奨します。
ステップ 11	**end** **例：** `Router(conf-ssh-pubkey-server)# end`	公開キー サーバ モードを終了し、特権 EXEC モードに戻ります。
ステップ 12	**configure terminal** **例：** `Router# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 13	**ip ssh stricthostkeycheck** **例：** `Router(config)# ip ssh stricthostkeycheck`	サーバ認証が実行されることを確認します。 • 障害発生時には接続は終了します。

リモート デバイスでの暗号化セッションの開始

リモート ネットワーキング デバイスで暗号化セッションを開始するには、このタスクを実行します（ルータをイネーブルにする必要はありません。SSH はディセーブル モードで実行できます）。

（注） 接続するデバイスは、Cisco IOS ソフトウェアでサポートされる暗号化アルゴリズムを備えた SSH サーバをサポートしている必要があります。

手順の概要

1. **ssh** [**-v** {**1** | **2**}] [**-c** {**3des** | **aes128-cbc** | **aes192-cbc** | **aes256-cbc**}] [**-m** {**hmac-md5** | **hmac-md5-96** | **hmac-sha1** | **hmac-sha1-96**}] [**l** *userid*] [**-o numberofpasswordprompts** *n*] [**-p** *port-num*] {*ip-addr* | *hostname*} [*command*]

手順の詳細

ステップ 1	**ssh** [**-v** {**1** | **2**}] [**-c** {**3des** | **aes128-cbc** | **aes192-cbc** | **aes256-cbc**}] [**-m** {**hmac-md5** | **hmac-md5-96** | **hmac-sha1** | **hmac-sha1-96**}] [**-l** *userid*] [**-o numberofpasswordprompts** *n*] [**-p** *port-num*] {*ip-addr* | *hostname*} [*command*] **例：** `Router# ssh -v 2 -c aes256-cbc -m hmac-sha1-96 -l user2 10.76.82.24`	リモート ネットワーキング デバイスを使用した暗号化セッションを開始します。

トラブルシューティングのヒント

ip ssh version コマンドは、SSH 設定のトラブルシューティングに使用できます。バージョンを変更すると、どの SSH バージョンが問題かを確認できます。

SSH サーバでのセキュア コピー プロトコルのイネーブル化

このタスクを実行して、SSH サーバ上でセキュア コピー プロトコルをイネーブルにします。このタスクでは SCP に関するサーバサイドの機能を設定します。これは、ルータでリモートのワークステーションからファイルを安全にコピーできる一般的な設定の例です。

前提条件

SCP が正しく機能するかどうかは、AAA 認証と認可に依存しています。したがって、AAA はルータで設定する必要があります。

手順の概要

1. **enable**
2. **configure terminal**
3. **aaa new-model**
4. **aaa authentication login default local**
5. **aaa authorization exec default local**
6. **username** *name* **privilege** *privilege-level* **password** *password*
7. **ip ssh time-out** *seconds*
8. **ip ssh authentication-retries** *integer*
9. **ip scp server enable**

手順の詳細

ステップ 1	**enable** **例：** `Router> enable`	特権 EXEC モードをイネーブルにします。 • プロンプトが表示されたら、パスワードを入力します。
ステップ 2	**configure terminal** **例：** `Router# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**aaa new-model** **例：** `Router(config)# aaa new-model`	AAA アクセス コントロール モデルをイネーブルにします。
ステップ 4	**aaa authentication login default local** **例：** `Router(config)# aaa authentication login default local`	認証時にローカルのユーザ名データベースを使用するように、ログイン時の AAA 認証を設定します。
ステップ 5	**aaa authorization exec default local** **例：** `Router(config)# aaa authorization exec default local`	ユーザ アクセスを制限するパラメータをネットワークに設定します。認証を実行し、ユーザ ID で EXEC シェルを実行するかどうかを定義します。その後、システムで認証にローカル データベースを使用するかを指定します。
ステップ 6	**username** *name* **privilege** *privilege-level* **password** *password* **例：** `Router(config)# username samplename privilege 15 password password1`	ユーザ名ベースの認証システムを確立し、ユーザ名、権限レベル、非暗号化パスワードを指定します。 **（注）** *privelege-level* 引数の最小値は 15 です。権限レベルが 15 未満の場合、接続が切断されます。
ステップ 7	**ip ssh time-out** *seconds* **例：** `Router(config)# ip ssh time-out 120`	ルータが SSH クライアントの応答を待つ時間間隔を、秒で設定します。
ステップ 8	**ip ssh authentication-retries** *integer* **例：** `Router(config)# ip ssh authentication-retries 3`	インターフェイスのリセット後、認証を試行する回数を設定します。
ステップ 9	**ip scp server enable** **例：** `Router(config)# ip scp server enable`	ルータで、リモート ワークステーションから安全にファイルをコピーできるようにします。

トラブルシューティングのヒント

SCP 認証に関する問題のトラブルシューティングには、**debug ip scp** コマンドを使用します。

show ssh コマンドを使用したセキュア シェル接続のステータスの確認

ルータで SSH 接続のステータスを表示するには、**show ssh** コマンドを使用します。

手順の概要

1. **enable**
2. **show ssh**

手順の詳細

ステップ 1	**enable** **例：** `Router> enable`	特権 EXEC モードをイネーブルにします。 • プロンプトが表示されたら、パスワードを入力します。
ステップ 2	**show ssh** **例：** `Router# show ssh`	SSH サーバ接続のステータスを表示します。

例

次は、**show ssh** コマンド表示ステータスからの、さまざまな SSH バージョン 1 およびバージョン 2 接続に関する出力例です。

バージョン 1 およびバージョン 2 接続

```
------------------------------------------------------------------------
Router# show ssh

Connection      Version Encryption      State                   Username
 0              1.5     3DES            Session started         lab
Connection Version Mode Encryption  Hmac                State
Username
1          2.0     IN   aes128-cbc  hmac-md5     Session started       lab
1          2.0     OUT  aes128-cbc  hmac-md5     Session started       lab
------------------------------------------------------------------------
```

バージョン 1 がないバージョン 2 接続

```
------------------------------------------------------------------------
Router# show ssh

Connection Version Mode Encryption  Hmac                State
Username
1          2.0     IN   aes128-cbc  hmac-md5     Session started       lab
1          2.0     OUT  aes128-cbc  hmac-md5     Session started       lab
%No SSHv1 server connections running.
------------------------------------------------------------------------
```

バージョン 2 がないバージョン 1 接続

```
------------------------------------------------------------------------
Router# show ssh

Connection      Version Encryption      State                   Username
 0              1.5     3DES            Session started         lab
%No SSHv2 server connections running.
------------------------------------------------------------------------
```

セキュア シェル ステータスの確認

SSH 設定を確認するには、このタスクを実行します。

手順の概要

1. **enable**
2. **show ip ssh**

手順の詳細

ステップ 1	**enable** **例：** `Router> enable`	特権 EXEC モードをイネーブルにします。 • プロンプトが表示されたら、パスワードを入力します。
ステップ 2	**show ip ssh** **例：** `Router# show ip ssh`	SSH のバージョンおよび設定データを表示します。

例

次は、**show ip ssh** コマンドの、イネーブルになっている SSH のバージョン、認証タイムアウト値、および認証リトライ回数の出力例です。

バージョン 1 およびバージョン 2 接続

```
---------------------------------------------------------------------
Router# show ip ssh

SSH Enabled - version 1.99
Authentication timeout: 120 secs; Authentication retries: 3
---------------------------------------------------------------------
```

バージョン 1 がないバージョン 2 接続

```
---------------------------------------------------------------------
Router# show ip ssh

SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
---------------------------------------------------------------------
```

バージョン 2 がないバージョン 1 接続

```
---------------------------------------------------------------------
Router# show ip ssh

3d06h: %SYS-5-CONFIG_I: Configured from console by console
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
---------------------------------------------------------------------
```

セキュア シェル バージョン 2 のモニタリングと維持

SSH 接続に関するデバッグ メッセージを表示するには、**debug ip ssh** コマンドと **debug snmp packet** コマンドを使用します。

手順の概要

1. **enable**
2. **debug ip ssh**
3. **debug snmp packet**

手順の詳細

ステップ 1	**enable** **例：** `Router> enable`	特権 EXEC モードをイネーブルにします。 • プロンプトが表示されたら、パスワードを入力します。
ステップ 2	**debug ip ssh** **例：** `Router# debug ip ssh`	SSH のデバッグ メッセージを表示します。
ステップ 3	**debug snmp packet** **例：** `Router# debug snmp packet`	ルータで送受信された各 SNMP パケットに関する情報を表示します。

例

次は、ディジット 2 キーワードが割り当てられ、これが SSH バージョン 2 接続であることを示す **debug ip ssh** コマンドの出力例です。

```
Router# debug ip ssh

00:33:55: SSH1: starting SSH control process
00:33:55: SSH1: sent protocol version id SSH-1.99-Cisco-1.25
00:33:55: SSH1: protocol version id is - SSH-2.0-OpenSSH_2.5.2p2
00:33:55: SSH2 1: send: len 280 (includes padlen 4)
00:33:55: SSH2 1: SSH2_MSG_KEXINIT sent
00:33:55: SSH2 1: ssh_receive: 536 bytes received
00:33:55: SSH2 1: input: packet len 632
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: ssh_receive: 96 bytes received
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: input: padlen 11
00:33:55: SSH2 1: received packet type 20
00:33:55: SSH2 1: SSH2_MSG_KEXINIT received
00:33:55: SSH2: kex: client->server aes128-cbc hmac-md5 none
00:33:55: SSH2: kex: server->client aes128-cbc hmac-md5 none
00:33:55: SSH2 1: expecting SSH2_MSG_KEXDH_INIT
00:33:55: SSH2 1: ssh_receive: 144 bytes received
00:33:55: SSH2 1: input: packet len 144
00:33:55: SSH2 1: partial packet 8, need 136, maclen 0
00:33:55: SSH2 1: input: padlen 5
00:33:55: SSH2 1: received packet type 30
00:33:55: SSH2 1: SSH2_MSG_KEXDH_INIT received
```

```
00:33:55: SSH2 1: signature length 111
00:33:55: SSH2 1: send: len 384 (includes padlen 7)
00:33:55: SSH2: kex_derive_keys complete
00:33:55: SSH2 1: send: len 16 (includes padlen 10)
00:33:55: SSH2 1: newkeys: mode 1
00:33:55: SSH2 1: SSH2_MSG_NEWKEYS sent
00:33:55: SSH2 1: waiting for SSH2_MSG_NEWKEYS
00:33:55: SSH2 1: ssh_receive: 16 bytes received
00:33:55: SSH2 1: input: packet len 16
00:33:55: SSH2 1: partial packet 8, need 8, maclen 0
00:33:55: SSH2 1: input: padlen 10
00:33:55: SSH2 1: newkeys: mode 0
00:33:55: SSH2 1: received packet type 2100:33:55: SSH2 1: SSH2_MSG_NEWKEYS received
00:33:56: SSH2 1: ssh_receive: 48 bytes received
00:33:56: SSH2 1: input: packet len 32
00:33:56: SSH2 1: partial packet 16, need 16, maclen 16
00:33:56: SSH2 1: MAC #3 ok
00:33:56: SSH2 1: input: padlen 10
00:33:56: SSH2 1: received packet type 5
00:33:56: SSH2 1: send: len 32 (includes padlen 10)
00:33:56: SSH2 1: done calc MAC out #3
00:33:56: SSH2 1: ssh_receive: 64 bytes received
00:33:56: SSH2 1: input: packet len 48
00:33:56: SSH2 1: partial packet 16, need 32, maclen 16
00:33:56: SSH2 1: MAC #4 ok
00:33:56: SSH2 1: input: padlen 9
00:33:56: SSH2 1: received packet type 50
00:33:56: SSH2 1: send: len 32 (includes padlen 13)
00:33:56: SSH2 1: done calc MAC out #4
00:34:04: SSH2 1: ssh_receive: 160 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #5 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 50
00:34:04: SSH2 1: send: len 16 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #5
00:34:04: SSH2 1: authentication successful for lab
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #6 ok
00:34:04: SSH2 1: input: padlen 6
00:34:04: SSH2 1: received packet type 2
00:34:04: SSH2 1: ssh_receive: 64 bytes received
00:34:04: SSH2 1: input: packet len 48
00:34:04: SSH2 1: partial packet 16, need 32, maclen 16
00:34:04: SSH2 1: MAC #7 ok
00:34:04: SSH2 1: input: padlen 19
00:34:04: SSH2 1: received packet type 90
00:34:04: SSH2 1: channel open request
00:34:04: SSH2 1: send: len 32 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #6
00:34:04: SSH2 1: ssh_receive: 192 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #8 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: pty-req request
00:34:04: SSH2 1: setting TTY - requested: height 24, width 80; set: height 24,
width 80
00:34:04: SSH2 1: input: packet len 96
00:34:04: SSH2 1: partial packet 16, need 80, maclen 16
00:34:04: SSH2 1: MAC #9 ok
```

```
00:34:04: SSH2 1: input: padlen 11
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: x11-req request
00:34:04: SSH2 1: ssh_receive: 48 bytes received
00:34:04: SSH2 1: input: packet len 32
00:34:04: SSH2 1: partial packet 16, need 16, maclen 16
00:34:04: SSH2 1: MAC #10 ok
00:34:04: SSH2 1: input: padlen 12
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: shell request
00:34:04: SSH2 1: shell message received
00:34:04: SSH2 1: starting shell for vty
00:34:04: SSH2 1: send: len 48 (includes padlen 18)
00:34:04: SSH2 1: done calc MAC out #7
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #11 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #8
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #12 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #9
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #13 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #10
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #14 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 17)
00:34:08: SSH2 1: done calc MAC out #11
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #15 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 16)
00:34:08: SSH2 1: done calc MAC out #12
00:34:08: SSH2 1: send: len 48 (includes padlen 18)
00:34:08: SSH2 1: done calc MAC out #13
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #14
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #15
00:34:08: SSH1: Session terminated normally
```

セキュア シェル バージョン 2 サポートの設定例

- 「例：セキュア シェル バージョン 1 の設定」（P.19）
- 「例：セキュア シェル バージョン 2 の設定」（P.19）
- 「例：セキュア シェル バージョン 1 および 2 の設定」（P.19）
- 「例：リモート デバイスでの暗号化セッションの開始」（P.19）
- 「例：サーバサイド SCP の設定」（P.19）
- 「例：SNMP トラップの設定」（P.20）
- 「例：SSH キーボード インタラクティブ認証」（P.20）
- 「例：SNMP のデバッグ」（P.22）
- 「例：SSH のデバッグの強化」（P.23）

例：セキュア シェル バージョン 1 の設定

次に、SSH バージョン 1 を設定する例を示します。

```
Router# configure terminal
Router(config)# ip ssh version 1
Router(config)# end
```

例：セキュア シェル バージョン 2 の設定

次に、SSH バージョン 2 を設定する例を示します。

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip ssh version 2
Router(config)# end
```

例：セキュア シェル バージョン 1 および 2 の設定

次に、SSH バージョン 1 と SSH バージョン 2 の両方を設定する例を示します。

```
Router# configure terminal
Router(config)# no ip ssh version
Router(config)# end
```

例：リモート デバイスでの暗号化セッションの開始

次に、リモート デバイスで暗号化セッションを開始する例を示します。

```
Router# ssh -v 2 -c aes256-cbc -m hmac-sha1-160 -l shaship 10.76.82.24
```

例：サーバサイド SCP の設定

次は、SCP のサーバサイド機能の設定方法の例です。この例では、ルータでの AAA 認証および認可の設定も示しています。この例では、ローカルに定義されたユーザ名とパスワードを使

用します。

```
Router# configure terminal
Router(config)# aaa new-model
Router(config)# aaa authentication login default local
Router(config)# aaa authorization exec default local
Router(config)# username samplename privilege 15 password password1
Router(config)# ip ssh time-out 120
Router(config)# ip ssh authentication-retries 3
Router(config)# ip scp server enable
Router(config)# end
```

例：SNMP トラップの設定

次に、設定済みの SNMP トラップの例を示します。トラップ通知は、SSH セッションが終了すると自動的に生成されます。この例の a、b、c、d は SSH クライアントの IP アドレスです。SNMP トラップデバッグ出力の例については、「例：SNMP のデバッグ」（P.22）のセクションを参照してください。

```
snmp-server
snmp-server host a.b.c.d public tty
```

例：SSH キーボード インタラクティブ認証

次は、キーボード インタラクティブ認証機能が自動的に配置されたさまざまなシナリオの例です。

クライアントサイドのデバッグ

次の例では、クライアントサイドのデバッグがオンになっており、プロンプトの最大数が 6（SSH キーボード インタラクティブ認証方式とパスワード方式の認証に 3 ずつ）です。

```
Password:
Password:
Password:
Password:
Password:
Password: cisco123
Last login: Tue Dec 6 13:15:21 2005 from 10.76.248.213
user1@courier:~> exit
logout
[Connection to 10.76.248.200 closed by foreign host]

Router1# debug ip ssh client

SSH Client debugging is on

Router1# ssh -l lab 10.1.1.3
Password:
*Nov 17 12:50:53.199: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version exchange successful
*Nov 17 12:50:53.203: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.335: SSH CLIENT0: key exchange successful and encryption on
*Nov 17 12:50:53.335: SSH2 CLIENT 0: using method keyboard-interactive
Password:
Password:
Password:
*Nov 17 12:51:01.887: SSH2 CLIENT 0: using method password authentication
```

```
Password:
Password: lab

Router2>
*Nov 17 12:51:11.407: SSH2 CLIENT 0: SSH2_MSG_USERAUTH_SUCCESS message received
*Nov 17 12:51:11.407: SSH CLIENT0: user authenticated
*Nov 17 12:51:11.407: SSH2 CLIENT 0: pty-req request sent
*Nov 17 12:51:11.411: SSH2 CLIENT 0: shell request sent
*Nov 17 12:51:11.411: SSH CLIENT0: session open
```

TACACS+ ACS がバックエンド AAA サーバ、ChPass がイネーブル、ブランク パスワードの変更あり

次の例では、TACACS+ Access Control Server（ACS; アクセス コントロール サーバ）がバックエンド AAA サーバです。ChPass 機能がイネーブルで、ブランク パスワードの変更が、SSH キーボード インタラクティブ認証方式を使用して行われています。

```
Router1# ssh -l cisco 10.1.1.3
Password:
Old Password: cisco
New Password: cisco123
Re-enter New password: cisco123

Router2> exit
[Connection to 10.1.1.3 closed by foreign host]
```

TACACS+ ACS がバックエンド AAA サーバ、ChPass がイネーブル、パスワードは最初のログインで変更

次の例では、TACACS+ ACS はバックエンド AAA サーバで、ChPass 機能がイネーブルです。パスワードは、SSH キーボード インタラクティブ 認証方式を使用して最初のログインで変更されています。

```
Router1# ssh -l cisco 10.1.1.3
Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123

Router2> exit
[Connection to 10.1.1.3 closed by foreign host]

Router1# ssh -l cisco 10.1.1.3
Password:cisco1
Your password has expired.
Enter a new one now.
New Password: cisco
Re-enter New password: cisco12
The New and Re-entered passwords have to be the same.
Try again.
New Password: cisco
Re-enter New password: cisco

Router2>
```

TACACS+ ACS はバックエンド AAA サーバ、ChPass はイネーブル、パスワードは 3 回のログイン後失効

次の例では、TACACS+ ACS はバックエンド AAA サーバで、ChPass 機能がイネーブルです。パスワードは SSH キーボード インタラクティブ認証方式を使用して、3 回のログイン後期限切れになります。

```
Router# ssh -l cisco. 10.1.1.3
Password: cisco

Router2> exit
[Connection to 10.1.1.3 closed by foreign host]

Router1# ssh -l cisco 10.1.1.3
Password: cisco

Router2> exit

Router1# ssh -l cisco 10.1.1.3
Password: cisco

Router2> exit
[Connection to 10.1.1.3 closed by foreign host]

Router1# ssh -l cisco 10.1.1.3
Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123

Router2>
```

例：SNMP のデバッグ

次は、**debug snmp packet** コマンドの出力例です。出力には、SSH セッションの SNMP トラップ情報が含まれます。

```
Router1# debug snmp packet

SNMP packet debugging is on

Router1# ssh -l lab 10.0.0.2

Password:

Router2# exit

[Connection to 10.0.0.2 closed by foreign host]
Router1#
*Jul 18 10:18:42.619: SNMP: Queuing packet to 10.0.0.2
*Jul 18 10:18:42.619: SNMP: V1 Trap, ent cisco, addr 10.0.0.1, gentrap 6, spectrap 1
local.9.3.1.1.2.1 = 6
tcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 4
ltcpConnEntry.5.10.0.0.1.22.10.0.0.2.55246 = 1015
ltcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 1056
ltcpConnEntry.2.10.0.0.1.22.10.0.0.2.55246 = 1392
local.9.2.1.18.2 = lab
*Jul 18 10:18:42.879: SNMP: Packet sent via UDP to 10.0.0.2
Router1#
```

例：SSH のデバッグの強化

次は、**debug ip ssh detail** コマンドの出力例です。出力には、SSH プロトコルとチャネル要求に関するデバッグ情報が含まれます。

```
Router# debug ip ssh detail

00:04:22: SSH0: starting SSH control process
00:04:22: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
00:04:22: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
00:04:22: SSH2 0: SSH2_MSG_KEXINIT sent
00:04:22: SSH2 0: SSH2_MSG_KEXINIT received
00:04:22: SSH2:kex: client->server enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2:kex: server->client enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2 0: expecting SSH2_MSG_KEXDH_INIT
00:04:22: SSH2 0: SSH2_MSG_KEXDH_INIT received
00:04:22: SSH2: kex_derive_keys complete
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS sent
00:04:22: SSH2 0: waiting for SSH2_MSG_NEWKEYS
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS received
00:04:24: SSH2 0: authentication successful for lab
00:04:24: SSH2 0: channel open request
00:04:24: SSH2 0: pty-req request
00:04:24: SSH2 0: setting TTY - requested: height 24, width 80; set: height 24, width 80
00:04:24: SSH2 0: shell request
00:04:24: SSH2 0: shell message received
00:04:24: SSH2 0: starting shell for vty
00:04:38: SSH0: Session terminated normally
```

次は、**debug ip ssh packet** コマンドの出力例です。出力には、SSH パケットに関するデバッグ情報が含まれます。

```
Router# debug ip ssh packet

00:05:43: SSH2 0: send:packet of  length 280 (length also includes padlen of 4)
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 280 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 24 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 4 bytes
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 144 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 6 bytes
00:05:43: SSH2 0: signature length 143
00:05:43: SSH2 0: send:packet of  length 448 (length also includes padlen of 7)
00:05:43: SSH2 0: send:packet of  length 16 (length also includes padlen of 10)
00:05:43: SSH2 0: newkeys: mode 1
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: input: total packet length of 16 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 8 bytes, maclen 0
```

```
00:05:43: SSH2 0: input: padlength 10 bytes
00:05:43: SSH2 0: newkeys: mode 0
00:05:43: SSH2 0: ssh_receive: 52 bytes received
00:05:43: SSH2 0: input: total packet length of 32 bytes
00:05:43: SSH2 0: partial packet length(block size)16 bytes,needed 16 bytes, maclen 20
00:05:43: SSH2 0: MAC compared for #3 :ok
```

関連情報

SSH バージョン 2 をサポートする SSH リモート デバイスを使用する必要があります。また、Cisco IOS ルータに接続する必要があります。

その他の参考資料

関連資料

内容	参照先
Cisco IOS コマンド	『*Cisco IOS Master Commands List, All Releases*』
AAA	『*Cisco IOS Security Configuration Guide*』の「*Securing User Services*」の章
• ホスト名およびホスト ドメインの設定 • セキュア シェルの設定	『*Cisco IOS Security Configuration Guide: Securing User Services*』の「Configuring Secure Shell」の章
コマンドのデバッグ	『*Cisco IOS Debug Command Reference*』
シスコ ソフトウェア イメージのダウンロード	『*Cisco IOS Configuration Fundamentals Configuration Guide*』 『*Cisco IOS Network Management Configuration Guide*』
Cisco IOS 設定の基礎	
IPsec	『*Cisco IOS Security Configuration Guide*』の「*Secure Connectivity*」の章
セキュリティ コマンド	『*Cisco IOS Security Command Reference*』
SNMP、トラップの設定	『*Cisco IOS Network Management Configuration Guide*』の「Configuring SNMP Support」の章

規格

規格	タイトル
IETF Secure Shell Version 2 Draft 規格	Internet Engineering Task Force の Web サイト

MIB

MIB	MIB リンク
新しい MIB または変更された MIB はサポートされていません。また、既存の MIB に対するサポートに変更はありません。	選択したプラットフォーム、Cisco ソフトウェア リリース、および機能セットの MIB の場所を検索しダウンロードするには、次の URL にある Cisco MIB Locator を使用します。 http://www.cisco.com/go/mibs

RFC

RFC	タイトル
新しい RFC または変更された RFC はサポートされていません。また、既存の RFC に対するサポートに変更はありません。	—

シスコのテクニカル サポート

説明	リンク
右の URL にアクセスして、シスコのテクニカル サポートを最大限に活用してください。以下を含むさまざまな作業にこの Web サイトが役立ちます。 ・テクニカル サポートを受ける ・ソフトウェアをダウンロードする ・セキュリティの脆弱性を報告する、またはシスコ製品のセキュリティ問題に対する支援を受ける ・ツールおよびリソースへアクセスする - Product Alert の受信登録 - Field Notice の受信登録 - Bug Toolkit を使用した既知の問題の検索 ・Networking Professionals（NetPro）コミュニティで、技術関連のディスカッションに参加する ・トレーニング リソースへアクセスする ・TAC Case Collection ツールを使用して、ハードウェアや設定、パフォーマンスに関する一般的な問題をインタラクティブに特定および解決する この Web サイト上のツールにアクセスする際は、Cisco.com のログイン ID およびパスワードが必要です。	http://www.cisco.com/cisco/web/support/index.html

セキュア シェル バージョン 2 サポートの機能情報

表 1 に、このモジュールで説明した機能をリストし、特定の設定情報へのリンクを示します。

Cisco Feature Navigator を使用すると、プラットフォームおよびソフトウェア イメージのサポート情報を検索できます。Cisco Feature Navigator を使用すると、ソフトウェア イメージがサポートする特定のソフトウェア リリース、機能セット、またはプラットフォームを確認できます。Cisco Feature Navigator には、http://www.cisco.com/go/cfn からアクセスします。Cisco.com のアカウントは必要ありません。

（注） 表 1 には、一連のソフトウェア リリースのうち、特定の機能が初めて導入されたソフトウェア リリースだけが記載されています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

表 1　セキュア シェル バージョン 2 サポートの機能情報

機能名	リリース	機能情報
セキュア シェル バージョン 2 サポート	12.2(25)S 12.3(4)T 12.2(11)T	セキュア シェル バージョン 2 サポート機能を使用して、Secure Shell（SSH; セキュア シェル）バージョン 2 を設定できます（SSH バージョン 1 のサポートは、以前の Cisco IOS ソフトウェア リリースで実装されていました）。SSH は信頼できる転送レイヤの上位で実行され、強化認証および暗号化機能を実現します。 12.3(11)T では、Cisco 10000 のサポートが追加されました。 この機能に関する詳細については、次の各項を参照してください。 • 「セキュア シェル バージョン 2 サポートに関する情報」（P.2） • 「セキュア シェル バージョン 2 サポートの設定方法」（P.5） 導入または変更されたコマンド：**debug ip ssh**、**ip ssh min dh size**、**ip ssh rsa keypair-name** 、**ip ssh version**、**ssh**。
セキュア シェル バージョン 2 クライアントおよびサーバ サポート	12.0(32)SY 12.3(7)JA 12.4(17)	Cisco IOS イメージが、SSH セッション終了時に SNMP トラップを自動的に生成するよう更新されました。 この機能に関する詳細については、次の各項を参照してください。 • 「SNMP トラップ生成」（P.4） • 「例：SNMP のデバッグ」（P.22）
SSH キーボード インタラクティブ認証	12.4(18) 12.2(33)SXH3	この機能は、SSH での汎用メッセージ認証とも呼ばれ、異なる種類の認証メカニズムを実装するために使用できる方式です。基本的に、現在サポートされている、ユーザの入力のみが必要な認証方式はすべて、この機能で実行することができます。 この機能に関する詳細については、次の各項を参照してください。 • 「SSH キーボード インタラクティブ認証」（P.5） • 「例：SSH キーボード インタラクティブ認証」（P.20）

表 1　セキュア シェル バージョン 2 サポートの機能情報（続き）

機能名	リリース	機能情報
セキュア シェル バージョン 2 の機能拡張	12.4(20)T	セキュア シェル バージョン 2 の機能拡張には、VRF aware SSH、SSH デバッグ機能拡張、および DH グループ 14 および 16 交換のサポートなどの、追加機能がいくつか含まれています。 この機能に関する詳細については、次の各項を参照してください。 • 「セキュア シェル バージョン 2 の機能拡張」（P.3） • 「例：サーバサイド SCP の設定」（P.19）
セキュア シェル バージョン 2 の RSA キーに関する機能拡張	15.0(1)M 15.1(1)S	RSA キーのセキュア シェル バージョン 2 機能拡張には、SSH 向け RSA キー ベースのユーザ認証や、SSH サーバ ホスト キーの保存や検証のサポートなどの、追加機能がいくつか含まれています。 この機能に関する詳細については、次の各項を参照してください。 • 「セキュア シェル バージョン 2 の RSA キーに関する機能拡張」（P.3） • 「RSA ベースのユーザ認証を実行するための Cisco IOS SSH サーバの設定」（P.7） **ip ssh pubkey-chain** および **ip ssh stricthostkeycheck** の各コマンドが導入または変更されました。