SQL Anywhere 10 サーバのセキュリティ保護

ホワイトペーパー - iAnywhere Solutions, Inc.

目次

はじめに

今日の IT 環境では、あらゆる企業でコンピュータとコンピュータ対応のデバイスが重要な役割を担っており、適切な情報セキュリティを実装することはもはや守るべき責務となっています。日常業務で取り扱う電子データの量が増え続けている状況で、適切なセキュリティ・プロシージャ (手続き) の実施に失敗するようなことがあれば、その企業は厄介な問題に悩まされ、無用の危険にさらされることになります。

情報セキュリティに関して言えば、IT インフラストラクチャの構成要素の中でも、実際に情報の格納と伝送を行うデータベース・サーバに注意を払うことが極めて重要な意味を持ちます。データベース・サーバのセキュリティ保護が不十分であれば、重要な業務データを不正に入手する経路を提供することになりかねません。また、盗聴の試みや意図的なデータ破壊を防ぐ意味でも、データベース・サーバとのデータ通信のセキュリティを適切な手段で保護する必要があります。
ここ最近で国内のメディアの注目を集めているセキュリティ侵害の事例には、必ずと言っていいほど、機密情報である個人データの紛失や盗用が関連しています。その一例として米国保険福祉省の事例を挙げますが、これは保険会社の従業員がホテルのコンピュータを利用してデータを呼び出し、そのファイルを削除し忘れたために、メディケア (高齢者公的医療保険) 受給者約 17,000 人の社会保障番号とその他の個人情報が漏えいした可能性があるというものです。このような事件は、関与した組織の名声や評判を損なうだけでなく、個人情報の盗難や金融詐欺といった実質的な被害をもたらすことにもなります。

このホワイトペーパーでは、最初に業務データに対する不正アクセスを防ぐためのセキュリティ・インフラストラクチャについて考察してから、SQL Anywhere 10 データベース・サーバのインストールのセキュリティ保護に必要な手順に重点を置いて説明します。SQL Anywhere 10 は、中小規模のエンタープライズ・データベース市場とともに、モバイル・デバイス向けのデータベース市場でもトップの地位を占めており、その強力なモバイル・ソリューションでは、データベース・サーバ自体のセキュリティ保護に加えて、データベース・サーバとモバイル・データベースの通信伝送のセキュリティ保護にも注意を払うことが重視されています。ただし、このホワイトペーパーで取り上げる技法や手法は、必ずしもモバイル環境とサーバ環境の両方で効果を発揮するわけではありません。たとえば、ハードウェアの物理的なセキュリティは、モバイル・デバイスの環境では、サーバやデスクトップ・ハードウェアの環境よりも厄介な問題となります。

SQL Anywhere 10 データベースをセキュリティ保護する上で解決すべき問題は、次に示す 4 種類の一般的な領域に分類されます。

◆ インストール前段階のセキュリティ - データベースを実行しているサーバの物理的なセキュリティおよびネットワーク・セキュリティに加えて、システムの基盤となるオペレーティング・システムのセキュリティを含みます。

◆ データベース・アクセスとパーミッション - データベースのセキュリティ保護には、そのアクセスのセキュリティ保護が大いに関係するため、この領域には最も注意を払う必要があります。具体的には、ユーザ・アカウントとパーミッション、認証方式、パスワード・ポリシー、バックアップ・セキュリティなどを含みます。

◆ データベースの暗号化 - 最大限のセキュリティを確保するには、常に強力な暗号化キーを使用してデータベースを暗号化する必要があります。

◆ 通信の暗号化 - どれだけ強力な手段でデータベース・ファイルをセキュリティ保護していても、データがネットワーク上でやり取りされれば、傍受や悪用の対象となる可能性があります。SQL Anywhere の組み込みのトランスポート・レイヤ・セキュリティを利用して、通信の暗号化と伝送データのセキュリティ保護を実現することをおすすめします。

セキュリティが頭痛の種になる理由は?

インターネットと企業コンピュータ・ネットワークの普及により、今や情報は企業にとって重要な商品となりました。事例によっては、情報が失われることで、企業が大きな困難に直面することもあります。結果として、ほとんどの大企業は、データが盗用や紛失に遭った場合の脅威に対処するために必要なセキュリティ・プロシージャや予防策について少なくとも認識しており、実際に取り組みを進めている企業もあります。

このような取り組みがきっかけとなり、セキュリティに対してどの程度の費用負担が必要であるかという問題が提起されています。この問題に対する答えは決して単純なものではありません。なぜなら、目に見える成果をもたらす本来の情報セキュリティの費用負担に加えて、目に見えない二次的なコストも存在するからです。確かに、ほとんどの事例において、情報セキュリティに直接的に関連するコストは少額ではありませんが、二次的なコストについてはさらに慎重に考慮する必要があると考えられます。なぜなら、直接的なコストの額は、二次的なコストによって左右されるからです。

直接的なセキュリティ・コストには、予防策を講じる上で企業が負担する一定量の金額が含まれます。具体的には、セキュリティ担当者の人件費、運営管理担当者の人件費、設備のコスト、コンピュータ機器のコスト、セキュリティ・コンサルタントの人件費、物理的なセキュリティ手段のコストなどが挙げられます。推計では、企業は IT 予算の約 30% をセキュリティ関連に費やしており、直接的なセキュリティ・コストの合計額が数百万ドルに達している大企業も珍しくはありません。

二次的なコストとは、将来の最悪の事態に対処するためのコストのことですが、その性質上、具体性に欠けるため、このようなコストの額を正確に算出することは不可能です。明確に言うと、これはデータの紛失や盗用が発生した場合に企業が負担する可能性のあるコストのことであり、その額は管理する対象のデータの量や種類によって大きく左右されるものの、ゼロになることはなく、企業活動を停止に追いやるほどの金額に達する可能性もあります。このコストには、次のようなものが含まれています。

◆ 時間 - 紛失や盗用にあったデータの種類を問わず、そのデータの入手に要した時間の量は常にコストとして換算されます。広く普及しているデータのパブリック・ストアも、データの整合性が損なわれていないかを確認する必要があるという点では、その確認に要する時間がコストに含まれます。

◆ 法的コスト - (国や州の法律によっては) 企業は自社で管理する顧客データに対して法的に責任を負っている場合があります。データの紛失に起因する訴訟では、企業が負担するコストが非常に大きくなる可能性があります。また、膨れ上がる可能性のある解決費用に加えて、訴訟には必ず高額な弁護士費用も発生します。

◆ 機会逸失コスト　データが失われた場合、通常は結果としてダウンタイムを伴います。定期的にバックアップを作成していたことで、ダウンタイムを最小限に抑制できたとしても、情報とインターネットが主導する今日の環境では、時間と売上の両方の意味において、ダウンタイムは取り戻せない生産性の損失に等しいと考えられます。

◆ 信用喪失　セキュリティ侵害に起因して、利害関係者 (従業員、顧客、パートナー企業など) からの信用が失われる可能性があります。信用を取り戻すことは困難であり、時間を要します。

結局のところ、直接的なセキュリティ・コストへの支出額は、企業の判断に委ねられます。この判断を適切に行うには、最悪の事態に対処するための二次的なコストに留意する必要があります。情報セキュリティに関して言えば、予防策は解決策よりも優先されるだけでなく、企業の死活問題にかかわる可能性があります。

インストール前段階のセキュリティ

この項では、データベース・サーバのセキュリティ向上に寄与するため、SQL Anywhere の外部で取るべき手順の概要について説明します。各手順では、一般的なサーバの注意事項やオペレーティング・システムのセキュリティについて取り上げます。

物理的なマシンのセキュリティ保護

サーバ・セットアップをセキュリティ保護する際には、まずサーバの物理的な特性に注目する必要があります。どれだけサーバ・ソフトウェアを保護していたとしても、不正なユーザがマシンを直接操作した場合、ほとんどのソフトウェアのセキュリティ対策は無効になります。また、ハードウェア自体を盗む方法以外にも、各種ソフトウェア・ツールを使用してサーバの管理者権限を取得すれば、後からさまざまな不正行為を働くことができます。

したがって、データベース・サーバは、物理的に保護された場所に配置する必要があります。具体的な場所の例としては、特定の入室資格を要求することでセキュリティ保護されたサーバ・ルームが挙げられます。また、このようなサーバ・ルームには、浸水検知、火災検知、および消火に対応する各種システムが配備されていることが理想的です。

ネットワーク・セキュリティ

サーバの物理的な安全性が確保されたら、ネットワーク・アクセスのセキュリティ保護に伴う問題を解決する必要があります。まず第一に、データベース・サーバは、インターネットに直接公開すべきではありません。データには常にミドルウェア・ソフトウェア・システムを介してアクセスするようにすることで、データベースへの接続メカニズムが外部から直接アクセスされることはなくなります。たとえば、SQL Anywhere Mobile Link のデータベース同期機能の使用時に、デフォルトでは、モバイル・デバイス自体がデータベースと直接情報をやり取りすることはありません。代わりに、Mobile Link サーバと通信してから、バックエンドのデータベース・サーバと情報をやり取りします。

ほとんどの企業は、適切なファイアウォールの必要性を認識しています。内部ネットワークのコンピュータとの通信のみを許可するように、ファイアウォール・ポリシーは可能なかぎり安全性を維持できるように設定する必要があります。ファイアウォールに穴を開けるような設定 (デフォルトの SQL Anywhere 接続ポート 2638 の開放など) は安易に行わないでください。

オペレーティング・システムのセキュリティ保護

データベース・サーバを実行するオペレーティング・システムは、その目的に特化させることをおすすめします。不要なサービスおよびデーモン (FTP や Telnet など) をすべて無効にして、別のソフトウェアにおける安全性の欠陥 (セキュリティ・フロー) が原因でデータベースへの不正なアクセスが発生する可能性をなくします。最後に、関連性のあるセキュリティ更新プログラムをすべて定期的にダウンロードしてインストールする必要があります。

また、オペレーティング・システムのユーザ・アカウントについても注意を払う必要があります。データベースの実行可能ファイルやデータ・ファイルへのアクセスを許可するのは、管理者権限を持つユーザのみに限定する必要があります。さらに、ネットワーク認証を使用する場合は、ローカル (非ネットワーク) ユーザ・アカウントでデータベース・サーバを起動しないでください。ユーザが意図的にまたは誤ってデータベース・サーバを停止することがないように、プロセスを開始および停止するパーミッションも厳密に管理する必要があります。

モバイル・デバイスのセキュリティ保護

適切なセキュリティ対策を講じていない状況でモバイル・デバイスを使用した場合、重大なセキュリティの危険が企業にもたらされる可能性があります。企業のデータベースに記録されているデータにモバイル・デバイスがリモート環境からアクセスできる場合、この危険は大幅に増加します。モバイル・デバイスを使用する場合に、そのさまざまな利点を受けながらも、起こり得る問題を回避するには、次の 3 種類の領域で、モバイル・デバイスをセキュリティ保護する必要があります。

デバイスへのアクセス

モバイル・デバイスは小サイズであることから、紛失や盗難に遭う危険性が高いと言えます。モバイル・デバイス上のデータに対するセキュリティの脅威に関しては、マルウェア、ウイルス、ワームなどよりも、紛失や盗難の方が危険であると専門家は分析しています。したがって、不正なユーザがモバイル・デバイスへのアクセスに成功した場合でも、デバイスを悪用できないように適切な予防策を講じておく必要があります。

この領域の主な目的は、次のような強力かつ確実な認証手続きを実装して、不正なアクセスからモバイル・デバイスを保護することです。

◆ オペレーティング・システムのパワーオン・パスワード - 現在、ほとんどのモバイル・オペレーティング・システムに用意されている機能です。モバイル・デバイスのセキュリティで最も基本的な形態ですが、同時に最も効果的なものでもあります。最新のオペレーティング・システムには、誤ったシステム・パスワードが入力された場合にペナルティを課す機能も組み込まれています。たとえば、Windows Mobile では、再度パスワード入力が可能になるまでの時間が延長され、BlackBerry では、パスワード入力に 10 回失敗した時点でデバイス上のデータがすべて消去されます。
◆ リモート・デバイス管理 - ユーザの手動による操作を必要としない、リモート環境でのネットワーク管理者によるモバイル・デバイスのセキュリティ・オプションの変更を指しています。この機能を実装しない場合、一部のモバイル・デバイスは、組織の他のデバイスと比較して、セキュリティ・ポリシーを厳密に実施していないと考えられます。

◆ アプリケーション・レベルのセキュリティ - モバイル・デバイスにはオペレーティング・システムのパワーオン・パスワードが用意されていますが、重要なアプリケーションの実行もすべて同様にパスワードで保護することをおすすめします (対応するオプションが用意されている場合)。

◆ 証明書 - 今日のモバイル・デバイスは通常、認証目的の証明書ストレージをサポートしています。この機能は、特定のユーザまたはユーザ・グループのログイン権限をリモート環境から追加および削除する場合に特に便利です。
◆ 非従来型の認証 - モバイル・セクタの発展と、モバイル認証が直面している危険を実際に評価する動きを受けて、最近では、さまざまな非テキスト型のパスワード・オプションが採用されています。各方式は、従来型のパスワード入力による認証を置き換えるか、または連携して機能することで、モバイル・デバイスのセキュリティ・レベルを向上させています。具体的な例としては、署名認証、手書きパスワード、指紋認証、Smart Card/SecurID Card 認証などが挙げられます。

デバイスに保存されているデータへのアクセス

デバイス・アクセスのセキュリティ保護は不正なアクセスを防ぐ上で重要な手順ですが、フラッシュ・ドライブや USB ドライブなど、リムーバル・メディアの使い方にも、デバイス自体に保管されているデータを保護する上で非常に重要な意味があります。(パスワードのクラッキングや盗用などにより) デバイスの認証保護が回避された場合や、デバイスが認証保護されていない場合は、データ・アクセスのセキュリティ保護が唯一の防御策になります。
データ・アクセスのセキュリティ保護を実現する手段としては、第一にデータの暗号化が挙げられます。SQL Anywhere には強力な暗号化機能が用意されており、そのデータベース格納データを保護しています。ただし、通常は、保護を強化するために、モバイル・デバイス上の他のファイルも暗号化することをおすすめします。具体的には、ストレージ・メディア上のデータや、通常の操作時に RAM に格納されるデータも、データ・スヌーピングを防ぐため、暗号化の対象になります。

企業ネットワークへのアクセス

ネットワーク・アクセスに対して強力なセキュリティ・ポリシーを適用することも十分検討に値します。この領域は、企業ネットワークへのアクセスのセキュリティ保護と、企業ネットワークでの通信のセキュリティ保護という 2 つの相に区分されます。

企業ネットワークでは、セキュリティ機能を採用し、接続を許可する対象は、悪意がないと認められたデバイスのみに限定する必要があります。これにより、従業員が自分のデバイスからネットワークに接続できると同時に、企業スパイ行為 (ネットワークに接続可能なモバイル・デバイスを悪用して、データを自動収集するような行為など) の機会を減らすこともできます。企業ネットワークへのアクセスを制限する方法としては、ダイヤルアップ・パスワード、セキュアな Web サイト向けのネットワーク認証プロトコル、VPN 認証などが挙げられます。

ネットワーク認証後、デバイスとネットワークの間を流れるトラフィックをすべて暗号化することも重要です。SQL Anywhere にはトランスポート・レイヤ・セキュリティ (TLS) 機能が用意されており、業界標準の承認済みアルゴリズムを使用して通信を暗号化します。これにより、既存のテクノロジでは、復号化は非常に困難かつ時間を要するものになります。そのため、SQL Anywhere が配備された統合環境とモバイル・デバイスとの業務データ通信は安全であると考えられますが、ネットワークおよびシステムの管理者は、モバイル・デバイスとのその他の通信もすべて保護する必要があります。したがって、重要なアプリケーションはすべて、通信データのネイティブ暗号化をサポートしている必要があると考えられます。

なお、モバイル・デバイスが VPN に接続できる場合は、実績ある業界標準の暗号化技法ですべての通信を暗号化する必要があります。

iAnywhere Afaria

モバイル・デバイスのセキュリティに関しては、iAnywhere Afaria を使用すれば、前述の問題の多くを解決することができます。Afaria は業界トップレベルのモバイル・デバイス総合管理ソリューションであり、パッチ管理やセキュリティ管理から、リモート・コントロールやネットワーク制御に至るまで、前述のセキュリティ領域をすべてカバーします。Afaria は、モバイル IT 管理の次の領域で、企業を支援します。

◆ セキュリティ - Afaria は次の機能を提供し、デバイスのセキュリティ向上を実現します。

・ パッチ管理 - Afaria では、適用済みのパッチと未適用のパッチを正確に管理できます。パッチ配布は自動化され、ログに正確に記録されます。

・ セキュリティ管理 - Afaria には、パワーオン・パスワードの設定、証明書の自動発行、安全な全システム・データ削除、リモート・リセット機能といった各種機能と連携する全ディスク暗号化機能が用意されています。

・ データ・バックアップ - Afaria には、リモート・デバイスのデータ、プロファイル、および構成の自動バックアップおよび復元機能が用意されています。

◆ プロセス自動化 - Afaria は、ファイル転送やハードディスク・チェックといったプロセスの自動化およびスケジュール実行に対応します。
◆ データおよびコンテンツの管理 - Afaria では、ドキュメントの配布や管理に加えて、さまざまな形式のドキュメント間での同期化を簡単かつ安全に実施できます。

◆ システム管理拡張機能 - Afaria では、企業本社から単一のコンソールを使用して、あらゆるモバイル・デバイス上のソフトウェア設定を管理できます。
◆ ソフトウェアおよびインベントリ管理 - Afaria には、ソフトウェア・インストール、ライセンス付与、更新、ロールバック、リモート・コントロールなどの管理全般に対応する機能が用意されています。

モバイル・デバイスに関する IT ポリシーおよびプロシージャ

モバイル・デバイスとそのデータへのアクセス権のセキュリティ保護に加えて、IT 部門は具体的な一連のポリシーやプロシージャを適用してモバイル・インベントリを管理するとともに、割り当てられたハードウェアに関する責任について全ユーザを教育する必要があります。次のようなポリシーやプロシージャを実施することで、結果として組織全体のセキュリティに寄与します。

◆ 社内のモバイル・デバイスとその現在の使用者を網羅したインベントリの維持管理。

◆ リモート・パッチ管理プロシージャの実施。これにより、グループ単位でデバイスにパッチを適用することで、必要なセキュリティ・バグの修正プログラムが適用されずにデバイスが放置されるリスクが軽減されます。

◆ 紛失や盗難に遭ったモバイル・デバイスのリモート・アクセス権をすべて無効にするプロシージャの策定。これには、デバイスに保管されているユーザ認証情報 (パスワードなど) をすべて消去する手段も含まれます。
◆ 現在未使用のモバイル・デバイスの全データ消去、または現在未使用のモバイル・デバイスの所有者が変更された場合や、紛失や盗難に遭った場合に、当該デバイスの全データ消去。これにより、デバイスが盗難に遭った場合でも、危険にさらされるデータの量は最小限に抑制されます。

◆ モバイル・デバイスのデータ復旧に特化したサードパーティ企業から提供されるサービスの利用。

◆ 社内のモバイル・デバイスにインストールするソフトウェア・アプリケーションの共通化と、ユーザによるソフトウェアの追加インストールの制限。

データベース・アクセスとパーミッション

この項では、データベース操作を安全に行う上で必要なデータベース・アカウントの構成方法の概要について説明します。

ユーザ・アカウントとパーミッション

ユーザ ID は、一意なユーザ (ユーザ数が少ない場合) またはロール (データベースに接続するユーザ数が多い場合) ごとに作成する必要があります。これにより、DBA は各ユーザ ID のパーミッションを調整して、ユーザに許可するデータベース操作の種類を限定することができます。また、一意なユーザ ID を設定することで、特定のアクションに関与したユーザを判別するプロセスも簡素化されます。18 ページの「データベース監査」を参照してください。

SQL Anywhere データベース・サーバをセキュリティ保護するため、次のような問題を解決する必要があります。

◆ DBA 権限の制限 - DBA 権限を持つユーザは、データベース・サーバに対してあらゆる操作を実行できます。一般に、DBA 権限は通常のユーザ操作には不要であり、この権限を安易に割り当てた場合、セキュリティの脅威をもたらすことになります。また、DBA のログイン証明書は安全な場所に保管し、保管場所については、このような機密情報の漏えいを防ぐ法的義務がある一部のユーザに対してのみ知らせる必要があります。

◆ データベースの起動 - デフォルトでは、動作しているデータベース・サーバ上で任意のユーザがデータベースを起動できます。データベースの起動および停止に必要なパーミッションを設定するには、-gd オプションを使用します。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の -gd サーバ・オプションに関する項を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf（P168）

◆ データベースの作成 - デフォルトでは、CREATE DATABASE 文を使用して、任意のユーザがデータベースを起動できます。データベースの作成に必要なパーミッションを設定するには、-gu オプションを使用します。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の -gu サーバ・オプションに関する項を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf（P176）

◆ データベース・サーバの停止 - デフォルトでは、任意のユーザが dbstop ユーティリティを使用して、データベース・サーバをシャットダウンできます。データベース・サーバの停止に必要なパーミッションを設定するには、-gk オプションを使用します。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の -gk サーバ・オプションに関する項を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf（P170）

◆ データのロードおよびアンロード - LOAD TABLE、UNLOAD TABLE、および UNLOAD 文はすべて、サーバの基本ファイル・システムにアクセスします。これはセキュリティ上の問題になる可能性があるため、上記の文へのアクセスは、-gl コマンドを使用して制限することをおすすめします。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の -gl サーバ・オプションに関する項を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf（P171）

◆ 重要でないパーミッションをすべて無効にする - GRANT および REVOKE 文を使用する場合、日常業務では必要のないパーミッションをユーザやグループに付与しないでください。

☞ 詳細については、『SQL Anywhere サーバ - SQLリファレンス』の GRANT 文と REVOKE 文に関する項を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbrfja10.pdf （P565/P655）

統合化ログイン

統合化ログインでは、Windows オペレーティング・システムとデータベースにログオンする際に、この両方でユーザが使用するログイン名とパスワードが共通になります。外部ログイン名 (またはユーザ・グループ) がデータベースのユーザ ID と関連付けられます。

ユーザ ID の統合により、さまざまなセキュリティ上の利点が得られます。第一に、ユーザ ID を SQL Anywhere に統合した場合、そのユーザはデータベースのユーザ ID とパスワードを記憶しておく必要がありません。ユーザの利便性が向上することに加えて、(意図的にまたは誤って) データベースのセキュリティ侵害に使用される可能性がある ID とパスワードの組み合わせをユーザが知ることはなくなります。また、DBA はユーザのデータベース・アクセス・パーミッションを迅速に付与または削除できるようになります。さらに、Windows ユーザ・グループ全体で共通するデータベースのユーザ ID とパスワードを使用するように構成できるため、DBA の視点からは、ユーザ・アカウント管理が容易になります。

Microsoft Windows で統合化ログインを使用する場合、セキュリティ上の問題を最小限に抑えるため、いくつかの点で注意が必要になります。最初に、Windows の Guest ユーザ・アカウントを無効にする必要があります。Guest ユーザ・アカウントにはパスワードが設定されていないため、統合化ログインの構成が不十分な場合、悪意のあるユーザがデータベース・サーバに対するデータベース管理者のアクセス権を取得できる可能性があります。また、Windows オペレーティング・システムでセキュリティが侵害され、ユーザのログイン情報が盗まれた場合、悪意のあるユーザがデータベースに自由にアクセスできる可能性があります。

☞ 統合化ログインの詳細については、『SQL Anywhere サーバ - データベース管理』の「統合化ログインの使用方法」を参照してください。

ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.98)

Kerberos 認証

SQL Anywhere は、Kerberos 認証をサポートしています。Kerberos はネットワーク認証プロトコルの 1 つであり、秘密キー暗号方式に基づく強力な認証機能と暗号化機能を備えています。SQL Anywhere は、Windows の統合化ログインと同様の方法で、Kerberos を認証に使用しています。Kerberos サーバにすでにログインしているユーザは、ユーザ ID やパスワードを入力せずにデータベースに接続することができます。Kerberos ログインにより、セキュリティ・システムが共通化されて利便性が向上します。ただし、セキュリティに関連するいくつかの重要なポイントについて (単一障害点やクロック同期化など)、データベース管理者は理解しておく必要があります。

☞ Kerberos 認証ログインの詳細については、『SQL Anywhere サーバ - データベース管理』の「Kerberos 認証の使用」を参照してください。

ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.108)

ディレクトリ・サービス

SQL Anywhere には、LDAP サポートが組み込まれています。これにより、LDAP ディレクトリ・サービスのクエリを実行して、データベース・サーバを検索することができます。また、クライアント・データベースは、サーバ・ロケーション・ユーティリティ (dblocate) を使用して次の例のようなディレクトリ・サービスのクエリを実行することで、正確なネットワーク・アドレスが不明でも、統合データベースを検索することができます。

```
dblocate customerDatabaseServer
```

今日のエンタープライズ環境で LDAP アクセスによるディレクトリ・サービスが普及した理由の 1 つとしては、ディレクトリ・サービスのセキュリティ品質の向上が挙げられます。ディレクトリ・サービスでは、ユーザ管理、リソース管理、およびセキュリティ・ポリシー管理が容易になると同時に、データベース・サーバのネットワーク IP 情報を事実上、非公開にすることができます。

dblocate ユーティリティによるデータベース検索では、管理上の視点からはさまざまな利点が得られるものの、ユーティリティによる検索に対してデータベース・サーバを非公開にする方が望ましい場合もあります。その場合は、データベース・サーバの起動時に -sb オプションを使用します。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の -sb サーバ・オプションに関する項を参照してください。

ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.191)

ユーザ・パスワード・ポリシー

強度に問題がある脆弱なパスワードは、しばしばコンピュータ・システムの「アキレスの踵」に例えられます。データベース・サーバは、企業の存続にかかわる重要なデータが格納されている場所です。そのデータベース・サーバへのアクセスに、容易に推測できるようなパスワードを使用しているようでは、悲惨な結果も想像に難くありません。パスワードのセキュリティを向上させる方法としては、次のような技法が挙げられます。

◆ パスワードの最小長 - パスワードが長ければ長いほど、有効な文字の組み合わせの数が増えるため、コンピュータベースのパスワード推測プログラムを使用したクラック行為は困難になります。パスワードの最小長を設定するには、min_password_length オプションを使用します。たとえば、次の例では、パスワードの最小長を 8 文字に設定しています。

```
SET OPTION PUBLIC.min_password_length = 8
```

◆ パスワード管理ポリシーの適用 - ユーザは常に、自分の組織で許容されるパスワード・ポリシー (複雑性やランダム性の度合いなど) を意識する必要があります。また、このようなポリシーは、厳密に適用されなければ意味がありません。ユーザ・パスワードが安全基準を満たしていることを確認するには、パスワード検証オプション verify_password_function を使用します。

☞ 安全なパスワードを選択する際のガイドラインについては、36 ページの「付録 B ：強力なパスワードのガイドライン」を参照してください。

☞ パスワード検証関数の例については、39 ページの「付録 D ：パスワード検証関数の例」を参照してください。

◆ ユーザ・パスワードの有効期限 - verify_password_function オプションを使用して、パスワードが無効になるまでの期限を指定することもできます。これにより、古いパスワードが不正に使用された場合でも、古いパスワードは期限切れにより自動的に無効になっているため、データベース・セキュリティは向上します。

☞ 具体的な実装例については、39 ページの「付録 D ：パスワード検証関数の例」を参照してください。

◆ デフォルト DBA パスワードの変更 - データベース DBA (管理者) アカウントのデフォルトのパスワードは sql です。このパスワードは、データベース作成直後に、データベース・サーバをセットアップしてネットワーク経由での接続を許可する前に変更してください。最大限のセキュリティを確保するには、DBA パーミッション (および十分な長さのパスワード) を設定した新しいユーザ ID を作成して、組み込みの DBA アカウントを無効にしてください。

◆ ODBC データ・ソースにパスワードを含めない - ODBC データ・ソースの作成時に、データ・ソースのパスワードを格納するオプションが用意されています。セキュリティを向上させるには、このオプションを使用してパスワードを格納することは避けてください。

◆ ログイン試行の失敗回数の制限 - ログイン・プロシージャの機能を利用して、ユーザごとにログイン試行の失敗回数を制限できます。ログイン試行の失敗回数が上限に達した時点で、ユーザ・アカウントをロックアウトし、管理者に解決を依頼するように設定する必要があります。

カスタム・ログイン・プロシージャの使用

データベース・サーバへのログイン・パーミッションを詳細に管理するには、ユーザのログイン試行時に自動的に呼び出されるストアド・プロシージャを作成します。これは login_procedure オプションを使用して行われ、これにより、SQL Anywhere データベース・サーバのセキュリティ保護がさらに強化されます。

☞ カスタム・ログイン・ストアド・プロシージャのセットアップの詳細については、『SQL Anywhere サーバ - データベース管理』の login_procedure オプションに関する項を参照してください。

ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.460)

次の SQL の例は、INVALID_LOGON エラーを通知して、接続を拒否する方法を示しています。

```
CREATE PROCEDURE DBA.login_check()
    BEGIN
        DECLARE INVALID_LOGON EXCEPTION FOR SQLSTATE '28000';
        // Allow a maximum of 3 concurrent connections
        IF( DB_PROPERTY('ConnCount') > 3 ) THEN  SIGNAL
        INVALID_LOGON;
        ELSE
            CALL      sp_login_environment;
        END IF;
    END
    go

GRANT EXECUTE ON DBA.login_check TO PUBLIC
    go

SET OPTION PUBLIC.login_procedure='DBA.login_check'  go
```

データベース機能の無効化

実際の環境でデータベース・サーバの特定の機能が不要な場合は、-sf サーバ・オプションを使用して無効にしてください。無効にした機能は、クライアント・アプリケーション、ストアド・プロシージャ、トリガ、およびデータベース・イベントでは利用できなくなります。無効にした機能を実行時に再び有効にするには、-sk オプションを使用してサーバを起動します。このマニュアルで説明するように、特定の機能または機能のグループを指定して無効にすることができます。最大限のセキュリティを確保するには、サーバの起動時に -sk オプションを使用して、次の機能グループを無効にすることをおすすめします。

◆ local_call - データベース・サーバの直接の構成要素ではなく、データベース・サーバの管理下にないコードの実行を可能にする機能をすべて無効にします。この対象には、cmdshell、external_procedure、および Java 機能が含まれています。

◆ local_db - データベース・ファイルに関連する機能をすべて無効にします。この対象には、バックアップ、復元、データベース、および dbspace 機能が含まれています。この場合、バックアップを実行できるのは、データベースがオフラインの状態であるときに限定されます。

◆ local_io - ファイルとその内容への直接アクセスを可能にする機能をすべて無効にします。この対象には、xp_read_file、xp_write_file、ディレクトリ、load_table、およびアンロード機能が含まれています。

◆ local_log - ディスク上のファイルにデータを直接作成したり書き込んだりするロギング機能をすべて無効にします。この対象には、request_log および log_file 機能が含まれています。

-sf オプションを使用して機能を無効にする場合は、必要に応じて後から当該機能の一部を有効にできるように、-sk オプションを使用してキーを指定するようにしてください。

☞ 無効にできる機能および機能グループの詳細や全リストについては、『SQL Anywhere サーバ - データベース管理』の -sf サーバ・オプションに関する項を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.192)

SQL Anywhere ユーティリティへのアクセスのセキュリティ保護

SQL Anywhere ユーティリティはダウンロード可能な状態で公開されているため、「ユーザが望めば自由に入手できるもの」と認識しておく必要があります。ここで重要なポイントは、データベースのセキュリティに関して、SQL Anywhere ユーティリティが何らかの形で影響を及ぼす可能性があるということです。

次のユーティリティは、昇格したパーミッションによるデータベース接続を必要とします。ユーザ・パーミッションを制限することで、これらのユーティリティを使用した場合にセキュリティ上の問題が発生する可能性を抑制することができます。

ユーティリティ	セキュリティ上の問題
dbbackup	このバックアップ・ユーティリティを使用すると、データベースのバックアップ・コピーを作成できます。悪意のあるユーザが使用すれば、時間をかけて暗号化キーのクラッキングを実行することが可能になると考えられます。
dbinfo	この情報ユーティリティは、データベースに関する情報を表示します。この情報は、実施されているセキュリティ対策を突破する際に、攻撃者によって利用される可能性があります。
dbinit	この初期化ユーティリティを使用すると、データベースを作成できます。悪意のあるユーザが使用すれば、作成したデータベースを稼動サーバにロードし、リソースの使用量を増加させて、ディスク・スペースを減少させることが可能になると考えられます。
dbstop	このデータベース停止ユーティリティを使用すると、データベースを停止できます。悪意のあるユーザが使用すれば、不要なダウンタイムを発生させることが可能になると考えられます。
dbunload	このアンロード・ユーティリティを使用すると、データベース・スキーマとデータをカンマ区切り (.csv) 形式で保存できます。悪意のあるユーザが使用すれば、重要なデータを入手することが可能になると考えられます。
dbconsole	この SQL Anywhere コンソール・ユーティリティは、データベースの接続およびプロパティに関する情報を提供します。悪意のあるユーザが使用すれば、データベース・サーバのセキュリティを侵害することが可能になると考えられます。
dbisql	この対話型 SQL ユーティリティ (GUI またはコマンド・ライン) を使用すると、データベースに対して任意の SQL 文を実行できます。悪意のあるユーザが使用すれば、マイナスの結果をもたらす可能性があります。
Sybase Central	Sybase Central は、データベースのあらゆる要素へのアクセスを可能にする GUI 管理ツールであり、上記のユーティリティと同じ機能を提供するように設計されたウィザードも用意されています。ユーザがログイン証明書を入手すれば、このツールを使用して、データベースのあらゆる要素の表示および変更が可能になると考えられます。

次のユーティリティは、データベース・パーミッションを必要とせず、使用の際にデータベース・ファイルへのアクセス・パーミッションのみを必要とします。サーバ環境では、データのセキュリティ侵害を防ぐには、物理的なサーバおよびデータベース・ファイルをセキュリティ保護することが最適な方法です。

一方、モバイル環境では、この他にも検討すべきポイントがあります。たとえば、dbtran は主要なユーティリティの 1 つですが、利用する際にユーザ ID やパスワードを必要としません。それにもかかわらず、このユーティリティを使用すると、トランザクション・ログ・ファイルに含まれている SQL の全文を変換することが可能です。このような処理を防ぐため、いくつかのオプション (組み合わせる形でも使用可能) が用意されています。

◆ データベース・ファイルの暗号化 - データベース・ファイルが暗号化されている場合、トランザクション・ログ・ファイルを変換するには、暗号化キーが必要となります。
◆ 高頻度でバックアップを実施する - トランザクション・ログ・ファイルを頻繁にバックアップしてファイルの内容をトランケートするように、バックアップ・プロシージャを設計します。これにより、独立したディレクトリでトランザクション・ログ・ファイルを管理すると同時に、ライブ・トランザクション・ログ・ファイルのサイズを制限します。また、dbtran からの保護をより強化するには、ファイル・システム暗号化によってバックアップ・ディレクトリを暗号化する必要があります。

◆ チェックポイントでログをトランケートする - データベース・サーバに対して -m オプションを使用します。これにより、チェックポイントの発生時にトランザクション・ログをトランケートして、ログ・ファイルのデータ量を制限します。ただし、このオプションを使用すると復元が困難になることもあるため、使用はおすすめしません。

◆ モバイル・デバイスの管理 - IT ポリシーおよびプロシージャを使用して、リモート・データを管理します。たとえば、iAnywhere Afaria のようなツールを使用してリモート環境からデータを削除することで、悪意のあるユーザが実行できる操作を制限します。

ユーティリティ	セキュリティ上の問題
dbdsn	このデータ・ソース・ユーティリティは、悪意のあるユーザが使用すれば、ネットワーク上の SQL Anywhere データベースを簡単に特定することが可能になると考えられます。また、悪意のあるユーザが既存のデータ・ソースを変更および削除することも可能であり、アプリケーション・レイヤで問題を引き起こす可能性があります。
dberase	このデータベース消去ユーティリティを使用すると、重要なデータベース・ファイルを削除できます。悪意のあるユーザが使用すれば、データの損失をもたらす可能性があります。
dblog	このトランザクション・ログ・ユーティリティを使用すると、データベースに関連付けられているトランザクション・ログ・ファイルの名前を変更することも、トランザクション・ログをすべて無効にすることもできます。結果として、データの損失やパフォーマンスの低下をもたらす可能性があります。
dbtran	このログ・トランザクション・ユーティリティは、悪意のあるユーザが使用すれば、データベースのトランザクション・ログ・ファイルに基づいて、SQL 文を生成してデータベースを再現することが可能になると考えられます。これは、トランザクション・ログ・ファイルが暗号化されていない場合、特に危険です。

次のユーティリティは、データベース・ファイルへのアクセスをまったく必要とせず、ネットワークに接続さえしていれば実行することができます。これらのユーティリティの悪意ある使い方としては、単純に攻撃の対象を検索して特定する行為などが考えられます。

ユーティリティ	セキュリティ上の問題
dblocate	このサーバ列挙ユーティリティは、悪意のあるユーザが使用すれば、ネットワーク上のデータベース・サーバを直接検索することが可能になると考えられます。
dbns10	この DBNS ブロードキャスト・リピータ・ユーティリティは、攻撃者である疑いがある人物に対して、同一サブネット上または別サブネット上に存在するデータベース・サーバに関する情報を提供する可能性があります。

☞ 各ツールの詳細については、『SQL Anywhere サーバ - データベース管理』の「管理ユーティリティの概要」を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.637)

データベース・アクティビティのデバッグ

SQL Anywhere には、データベース・アクティビティのデバッグに対応するさまざまな機能が用意されています。これらの機能は、データベース・サーバのステータスや、データベース・サーバに送られた SQL 文の内容を判別する際に役立ちます。ただし、用意されている機能は確かに便利であるものの、これらの機能は重要な情報にアクセスするために悪用される可能性があるため、実際に使用する際には注意が必要です。

データベース要求ログ

SQL Anywhere の要求ログには、データベース・サーバに対する要求がすべて記録されます。ログに記録される情報としては、タイムスタンプ、接続 ID、要求タイプなどが挙げられます。また、クエリの場合は、アイソレーション・レベル、フェッチされたロー数、カーソル・タイプなども含まれ、INSERT、UPDATE、および DELETE 文の場合は、影響を受けるロー数や起動されたトリガ数なども含まれます。ただし、ログイン・データなどの機密情報は、要求レベル・ログには記録されません。稼動サーバに対する要求ロギングをオンにするには、サーバ上で動作しているデータベースに対する DBA 権限が必要ですが、要求ロギングを有効にした状態であれば任意のユーザが新しいデータベース・サーバを起動することができます。

要求ログは暗号化できず、すべての情報がプレーン・テキスト形式で記録されるため、データベース関連の問題を診断する場合に限定して使用する必要があります。それでも、侵入者である疑いがある人物がアプリケーションによるデータベースの呼び出しを要求ログから正確に判別し、この情報を悪用してアクセス権を不正に取得する可能性があります。

したがって、要求ログを使用する場合は、次の推奨事項に従ってください。

◆ 要求ログはすべて、不要になった時点で速やかに削除します。
◆ 要求ログのサイズに上限を設定して、セキュリティ侵害が発生した場合に悪用されるデータの量を最小限に抑制します。要求ログ・ファイルのサイズを制限するには、-zs オプションを使用します。
☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の -zs サーバ・オプションに関する項を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.215)

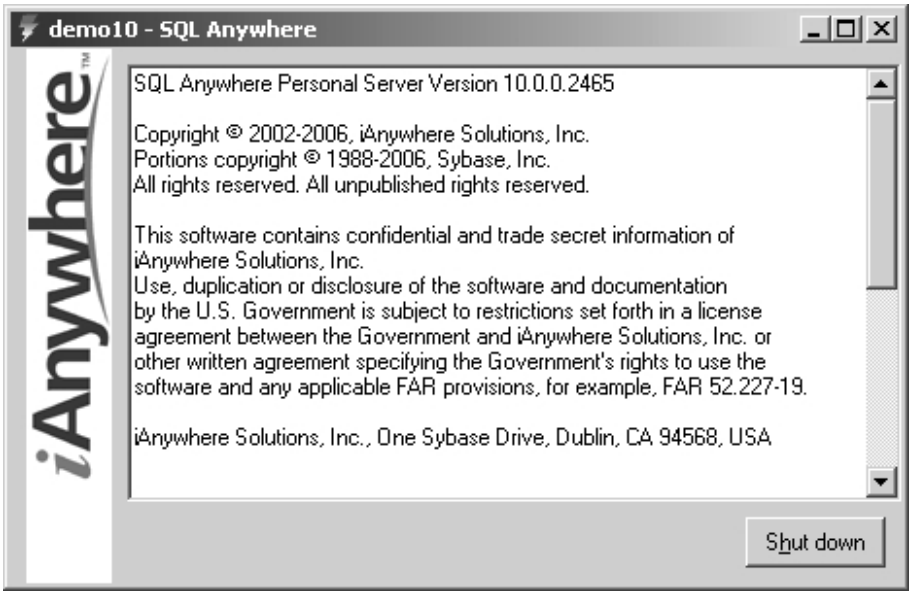
◆ アプリケーション・プロファイリング機能を利用して、要求ログ出力を別のトレーシング・データベースに格納します。これは SQL Anywhere 10 の新しい機能であり、トレーシング・データベースのデータを暗号化することでセキュリティを強化します (要求ログ情報をファイルに保存した場合、そのファイルは暗号化されていません)。

☞ 詳細については、『SQL Anywhere サーバ – SQLの使用法』の「Tracing session data」を参照してください。

ftp://ftp2.ianywhere.jp/public/tech/dbugja10.pdf (P216)

[Server Messages] ウィンドウ

[Server Messages] ウィンドウには、SQL Anywhere データベース・サーバに関するステータス情報が表示されます。Windows コンピュータでは、システム・トレイのアイコンが、データベース・サーバが動作していることを示す唯一の表示になります。



最大限のセキュリティを確保するには、データベース・サーバの起動時に -qw オプションを使用して、[Server Messages] ウィンドウを無効にすることをおすすめします。これでデータベース・サーバが動作していることを示す表示はなくなったため、攻撃者である疑いがある人物の目をそらすことができると考えられます。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の -qw サーバ・オプションに関する項を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P189)

SQL Anywhere 構成ファイルのセキュリティ保護

データベース・サーバの構成ファイルのセキュリティについても考慮する必要があります。悪意のあるユーザがこのファイルのコピーを入手できた場合、サーバの悪用に利用できる弱点を判別することが可能になると考えられます。

SQL Anywhere では、構成ファイルはデータベース・サーバの起動を簡素化するために使用されます。具体的には、構成ファイルとは必要なオプションおよびパラメータをすべてファイルに保存したものであり、@data オプションを使用してデータベース・サーバに渡されます。これにより、(複数のパラメータが必要な場合は特に) サーバの起動が簡素化されると同時に、重要な情報の格納が可能になります。

構成ファイルには機密情報が保存される場合もあるため、ファイル難読化ユーティリティ (dbfhide) を使用して構成ファイルの内容を隠蔽することが重要になります。dbfhide を使用すると、簡単な暗号化方式により、ファイルに保存された情報の内容を判別することがさらに困難になります。簡単な暗号化とは弊社内で開発された暗号化方式であり、難読化と同義です。ただし、AES などの業界標準の強力な暗号化アルゴリズムと比較すると、安全性は劣ります。dbfhide を使用する場合でも、機密情報 (暗号化キーなど) は構成ファイルに保存しないでください。また、オペレーティング・システムのユーザ・パーミッションを管理者ユーザのみに限定することで、構成ファイルへのアクセスもセキュリティで保護する必要があります。

データベース監査

監査は、データベースに対して実行されたアクティビティを追跡する方法の 1 つです。アクティビティの記録はトランザクション・ログに残されており、SQL Anywhere データベース・サーバでは、データベースごとにトランザクション・ログが保持されています。トランザクション・ログには、データベースに変更を加えたすべてのトランザクションが記録され、この情報はデータの整合性および復元性を確保する目的で使用されます。データベース監査が有効な場合、サーバはセキュリティ関連の情報をトランザクション・ログに追加します。これにより、随時調査を行うことが可能になり、ユーザがデータベースに不正なアクセスを試みた時刻の情報など、ログに記録されていた監査情報に基づいて、悪意のある動作を識別することが容易になります。また、ユーザ (またはユーザ・グループ) が実際には不正なアクセスを試みていなかった場合、データベース証明書が改ざんされたことを示している可能性があるため、迅速に対処する必要があります。

監査を有効にすることで、データベースでは次の項目が保存の対象になるため、トランザクション・ログに保存されるデータの量が増加します。

◆ すべてのログイン試行 (成功および失敗)、ユーザの IP アドレスを含む。

◆ すべてのイベントの正確なタイムスタンプ (ミリ秒単位の精度)。

◆ すべてのパーミッション・チェック (成功および失敗)、パーミッションがチェックされたオブジェクトを含む (該当する場合)。

◆ DBA 権限を必要とするすべてのアクション。

なお、監査はデフォルトでは無効になっているため、次のコマンドを実行して明示的に有効にする必要があります。

```
SET OPTION PUBLIC.auditing = 'On'
```

このコマンドの実行後は、無効にしないかぎり監査は有効なままとなります。データベース監査情報を取得するには、SQL Anywhere に用意されているログ・トランザクション・ユーティリティ (dbtran) を使用します。このユーティリティは、動作中のデータベース・サーバまたは特定のトランザクション・ログから監査情報を抽出する場合に使用します。ただし、ログ・ファイルが暗号化されていない場合は、悪意のあるユーザがこのユーティリティを使用してデータベースの完全なコピーを作成する上で必要な SQL 文をすべて生成することが可能になると考えられます。SQL Anywhere ユーティリティによる不正なアクセスからデータを保護する場合の詳細については、14 ページの「SQL Anywhere ユーティリティへのアクセスのセキュリティ保護」を参照してください。

データベース監査のパフォーマンス分析

データベース監査を有効にすると、ユーザがデータベースにアクセスするたびに追加のロギング情報が生成されるため、パフォーマンスに影響を及ぼします。監査が有効な状態と無効な状態でデータベースのパフォーマンスを比較すると、次のような結果が明らかになりました。

データの挿入操作では、監査が無効な状態のデータベースは、監査が有効な状態のデータベースと比較して、約 14% 高速に処理を実行しました。このテストは、Pentium 4 2.4GHz、1024MB の RAM を搭載したシステムで、単一のテーブルに 200,000 件のローを挿入するシミュレーションに基づいています。また、データの取得操作では、監査が無効な状態のデータベースは、監査が有効な状態のデータベースと比較して、約 14% 高速に処理を実行しました。このテストは、Pentium 4 2.4GHz、1024MB の RAM を搭載したシステムで、単一のテーブルから 200,000 件のローを取得するシミュレーションに基づいています。

バックアップのセキュリティ保護

データベースのバックアップのセキュリティは見過ごされがちであり、セキュリティの取り組みの多くはライブ・データベースを対象としているのが現状です。ただし、バックアップしたデータベースにも運用データベースとほぼ同じデータが格納されている以上、同様の重要性を持つものと考えられます。バックアップのセキュリティが重視されていない場合、悪意のあるユーザは、ライブ・データベースを保護する目的で採用されている高コストなセキュリティ対策を迂回して、バックアップのコピーを入手することに専念する可能性があります。この項では、考慮すべきバックアップのセキュリティに伴う問題の概要について説明します。

物理的な保護

バックアップ・システムやバックアップ・レコード (テープなど) の物理的なセキュリティに関して言えば、運用データベース・システムと同じレベルの注意を払う必要があると考えられます。さらに、バックアップのロケーションが運用現場から離れている場合は、専門のセキュリティ担当者が 24 時間 365 日体制で保護に対応する必要があります。また、データが人手を介して輸送される場合 (ネットワークを介して伝送されない場合)、そのデータの価値に応じて、輸送手段についてもセキュリティ保護の対策が必要になります。重要な企業データの場合、専用の輸送車両と専門訓練を受けた人員を提供できる警備会社に輸送を委託することをおすすめします。

ネットワーク・バックアップ・プロシージャ

ネットワーク・バックアップを行うには、データベース・ファイルの単純コピー、dbbackup ユーティリティ、dbbackup API、または BACKUP DATABASE SQL 文を実行します。提供されているバックアップ・ユーティリティを使用すると、セキュリティに関する重要な利点が得られます。また、バックアップ・ユーティリティは標準的なデータベース接続上で実行されるため、トランスポート・レイヤ・セキュリティ (TLS) に基づいてネットワーク伝送中のデータを保護することができます。

データベース・ファイルのバックアップ・コピーをネットワーク上で伝送する場合、いくつかの注意事項があります。理想的なケースとして、データが SQL Anywhere で強力に暗号化されている場合 (推奨)、最初に復号化を行ってからデータベースからデータを抽出します。

次に、(伝送を高速化するために) データを圧縮して、伝送中の保護のために再度暗号化します。バックアップ・システムに到着したデータは、復号化後に再度バックアップ暗号化キー (ライブ・データベースの暗号化キーとは異なるキー) を使用して暗号化され、最後にバックアップ・ストレージ・メディアに保存されます。

バックアップ暗号化

バックアップ・コピーは、専用の特別なバックアップ暗号化キーを使用して暗号化した状態で保管してください。(ライブ・データベースと比較して) 使用頻度が低いバックアップ・データについては、さらに複雑なアルゴリズムと、キーの長さを拡張した複雑な暗号化キーを使用してバックアップ・コピーを暗号化する方法も考えられます。この場合、データは二重に暗号化されることになりますが (運用システムでは非実用的)、バックアップ・データの場合は特に支障はありません。

Web サービスのセキュリティ保護

SQL Anywhere には HTTP サーバが組み込まれており、データベースに格納されている Web サービスにクライアント・アプリケーションがアクセスする場合に使用されます。これにより、標準的な HTTP 呼び出しで必要な情報が返されるため、データに対するユーザ要求は簡素化されると同時に、(ODBC 接続などでは必要とされた) データベースとの直接接続も不要になります。ただし、Web サービスを介したデータへのアクセスはセキュリティ保護する必要があります。なぜなら、セキュリティ保護が不十分な場合、Web サービスに接続すれば誰でも Web サービスから返されるデータを確認することができるからです。そのため、データベースで Web サービスを使用する場合、次の項目について十分に考慮する必要があります。

◆ 認証 - HTTP を介して Web サービスにアクセスする場合、情報はプレーン・テキスト形式で送信されるため、データベース・ログイン情報の入力をユーザに要求することは避けるべきです。たとえば、次の HTTP URL では、ユーザがユーザ ID DBA、パスワード sql を使用してサービス serviceName にログインしていることになります。

```
http://DBA:sql@localhost:8080/serviceName
```

悪意のあるユーザが、送信された URL に含まれているプレーン・テキスト形式のユーザ ID とパスワードを傍受すれば、データベースのセキュリティが侵害される可能性があります。

したがって、Web サービスへのアクセスにユーザ認証が必要な場合は、HTTP の代わりに HTTPS を使用して、URL 情報をすべて確実に暗号化する必要があります。

SQL Anywhere には、ユーザのログイン後に、(実際のユーザのアクセス・パーミッションにかかわらず) すべてのサービス・クエリを特定のユーザとして実行するオプションが用意されています。ただし、この機能を使用する際には、ユーザ・アカウントが悪用された場合にデータ損失の被害が拡大する可能性があるため、十分な注意が必要です。

◆ セキュアな接続の使用 - SQL Anywhere の Web サービスは、HTTP とセキュアな HTTPS 接続の両方をサポートしています。最大限のセキュリティを確保するには、Web サービスで HTTPS 接続を使用することを強くおすすめします (特にデータがインターネットを介して送信される場合)。

◆ パラメータ - SQL Anywhere では、Web サービスにパラメータを渡して、サービスに関連する SQL クエリを実行する際に必要な情報を追加で指定できます。

悪意があると考えられる攻撃を回避するには、適切なセキュリティ対策を実施する必要があります。詳細については、32 ページの「SQL インジェクション」を参照してください。

このような攻撃を回避するには、データベース・サーバに対する SQL クエリで使用する前に、すべてのパラメータについて、型を厳密に設定すると同時に、値が正しいことをチェックする必要があります。

暗号化によるデータ保護

SQL Anywhere では、データベースに格納されているデータを暗号化して、外部ツール (バイナリ・エディタなど) を用いた不正なユーザによるデータ表示を防ぐことができます。データベース内の重要なデータはすべて暗号化することをおすすめしますが、暗号化の必要性の有無と、暗号化の対象とするデータの範囲については、DBA の判断に委ねられます。なお、暗号化によるパフォーマンスのオーバーヘッドは一定ではなく (後述)、セキュリティと運用時のパフォーマンスのバランスを考慮する必要があります。

暗号化の種類

SQL Anywhere では、簡単な暗号化と高度な暗号化という 2 種類のデータ暗号化をサポートしています。

簡単な暗号化では、難読化アルゴリズムをデータに適用することで、データをセキュリティ保護します。これは高度な暗号化ほど安全ではありませんが、何者かがディスク・ユーティリティを使用してデータベース内のデータを解読するような行為を防ぐことはできます。また、簡単な暗号化は高度な暗号化と比較してパフォーマンスの面で優れおり、取得や挿入の操作におけるデータの復号化に伴うオーバーヘッドを最小限に抑えることができます。
一方、SQL Anywhere で採用される高度な暗号化は、米国標準技術局 (NIST) によって選定されたブロック暗号の Advanced Encryption Standard (AES) アルゴリズムに基づいています。これは、ユーザ指定のキーを使用してデータベースおよびトランザクション・ログ内のデータを暗号化し、元のキーが提供されなければデータの暗号を復号化できなくするというものです。この SQL Anywhere の高度な暗号化は、米国連邦情報処理規格 (FIPS) によって承認された AES アルゴリズムの実装を採用している一部のプラットフォームでも利用することができます。なお、AES_FIPS でデータベースを暗号化するには、個別ライセンスのコンポーネントを入手する必要があります。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の「安全なデータの管理」を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.917)

高度な暗号化のキーとアルゴリズム

セキュアな暗号化キーの選定に関しては、セキュアなパスワードの選定と同じ原則が適用されます (36 ページの「付録 B ：強力なパスワードのガイドライン」を参照)。選定したキーを失った場合、データベース内のデータ、データベースのログ・ファイル、およびデータベースのミラー・ファイル (使用可能な場合) にはまったくアクセスできなくなるため、物理的に保護された場所に選定したキーのコピーを保管しておく必要があります。

暗号化したデータベースを起動するには、暗号化キーの入力が必要になるため、データベースを起動できるのは特定の 1 人の人物 (DBA) のみに限定することをおすすめします。これにより、複数の人物が暗号化キーを知ることはなくなるため、セキュリティ侵害の機会 (ソーシャル・エンジニアリングによる情報入手など) は減少します。さらに、データベース・サーバのコマンド・ラインでは暗号化キーを指定しないでください。これは、ほとんどのオペレーティング・システムには、実行中のプロセスのコマンド・ラインを表示する手段が用意されており、キーが簡単に悪用される可能性があるためです。代わりに、データベース・サーバを起動してユーティリティ・データベースに接続してから、データベースの起動コマンドを使用してデータベースを起動する際に、データベースの起動コマンドでキーを指定するようにしてください。

組み込みアプリケーションでは、暗号化キーをユーザと共有したくない場合、次のようにキーを非公開にする手段として利用できるオプションが用意されています。
◆ キーをアプリケーションに埋め込む形で保存する - キーを簡単に見つけられるため、このオプションは通常おすすめしません。ただし、実装は非常に容易であり、一般のユーザに対しては、キーを非公開にできると考えられます。

◆ 実行時にキーを派生させる目的で使用できるアルゴリズムを策定する - キーをマシン上で保管する必要はなくなりますが、キー生成のアルゴリズムは保護する必要があります。このアルゴリズムでは、マシン、ユーザ、データベースなど、データベースを保持する環境ごとに異なるキーを生成する必要があります。
◆ キーを Windows の INI ファイルまたはレジストリで非表示にする - このオプションで非表示にした値もオペレーティング・システムのユーティリティで簡単にスヌーピングできるため、おすすめしません。ただし、セットアップは容易であり、一般のユーザに対しては有効な対策になると考えられます。

◆ Web サービスを使用してキーを取得する - インターネット・アクセスを必要とするアプリケーションや、インターネットに常時アクセスできる環境では、データベースを起動する前にアプリケーションから HTTPS Web サービスを呼び出してキーを取得するように設定できます。

◆ 組み込みの暗号化 API を使用する - Windows では、CryptProtectData や CryptUnProtectData など、組み込みの暗号化 API を使用してデータベース・キーを暗号化して保存できます。

☞ ソーシャル・エンジニアリングの詳細については、38 ページの「付録 C ：ソーシャル・エンジニアリング」を参照してください。

データベース作成用の暗号化オプション

次のコマンド・ライン・オプションは、初期化ユーティリティ (dbinit) を使用してデータベースを作成する際に、データベースとテーブルの暗号化を有効にするために使用します。

◆ -e - このオプションは、簡単な暗号化 (難読化) を使用してデータベースを作成する場合に使用します。

◆ -ek key - このオプションは、データベース全体の暗号化を指定してデータベースを作成する場合に使用し、続いて暗号化キーをパラメータとして指定します。このパラメータは、暗号化したデータベースを起動する際に、データベース暗号化キーを指定する場合にも使用します。

◆ -ep - このオプションは、データベース全体の暗号化を指定してデータベースを作成する場合に使用し、ユーザに暗号化キーの入力を要求するダイアログ・ボックスを表示します。ユーザは、データベースの作成時とデータベース・サーバの起動時の両方で、暗号化キーの入力を要求されます。

◆ -et - このオプションは、データベース内の個別のテーブルの暗号化を有効にしてデータベースを作成する場合に使用します。単独で使用した場合は、単純な暗号化が指定されます。続けて -ek または -ep を指定した場合は、暗号化するテーブルに対して高度な暗号化が使用されます。

☞ 詳細については、『SQL Anywhere サーバ－ SQLの使用法』の「Creating databases (SQL)」を参照してください。

ftp://ftp2.ianywhere.jp/public/tech/dbugja10.pdf (P185)

データベース全体の暗号化

セキュリティの観点からは、データベース全体の暗号化 (データベース・テーブルおよびデータベース・トランザクション・ログをすべて暗号化すること) をおすすめします。データベースは、簡単な暗号化または高度な暗号化を使用して暗号化することができます。

簡単な暗号化

暗号化データベースを作成するには、dbinit、CREATE DATABASE SQL 文、または Sybase Central Create Database ウィザードを使用します。たとえば、簡単な暗号化を使用してコマンド・プロンプトからデータベースを作成するには、次のコマンドを実行します。

```
dbinit -e database-file
```

高度な暗号化

高度な暗号化データベースを作成するには、コマンド・ライン・ユーティリティ、SQL 文、または Sybase Central を使用します。SQL 文を使用する場合、ENCRYPTED ON KEY 句を CREATE DATABASE 文に含める必要があります。

☞ 詳細については、『SQL Anywhere サーバ－ SQLの使用法』の「Creating databases (SQL)」を参照してください。

ftp://ftp2.ianywhere.jp/public/tech/dbugja10.pdf (P185)

コマンド・ラインで、-ek (コマンド・ラインで指定した暗号化キー) または -ep (ユーザに暗号化キーの入力を要求するダイアログ・ボックスを表示) パラメータを指定して dbinit ユーティリティを使用します。コマンド・ラインではユーザがキーをプレーン・テキスト形式で入力するのに対して、ダイアログではユーザが入力したキーがマスクされるため、-ep パラメータの使用をおすすめします。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の -ep サーバ・オプションに関する項を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.164)

データベースを作成してから暗号化するには、CREATE ENCRYPTED FILE 文を使用します。CREATE ENCRYPTED FILE 文は、データベースの暗号化キーを変更する場合にも使用することができます。

☞ 詳細については、『SQL Anywhere サーバ - SQL リファレンス』の「CREATE ENCRYPTED FILE 文」を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbrfja10.pdf (P.394)

テーブルの暗号化

テーブルの暗号化は、簡単な暗号化または高度な暗号化を使用してテーブル全体が暗号化されるという点ではデータベースの暗号化と似ています。ただし、テーブルの暗号化では、指定したテーブルのみが暗号化されるという点が異なります。この方法では、機密データを含むテーブルを暗号化すると同時に、データベースを暗号化した場合に発生する可能性があるパフォーマンスへの影響を回避することができます。ただし、データベース暗号化を有効にした場合、個別のテーブルごとに暗号化することはできなくなります。

簡単な暗号化

◆ SQL - 次の SQL 文は、テーブルの簡単な暗号化でデータベースを作成します。

```
CREATE DATABASE database-file-name ENCRYPTED TABLE
```

◆ コマンド・プロンプト - 次のコマンドは、dbinit ユーティリティを使用して、テーブルの簡単な暗号化でデータベースを作成します。

```
dbinit database-file-name -et
```

高度な暗号化

◆ SQL - 次の SQL 文は、テーブルの高度な暗号化で、キー abc および AES 暗号化アルゴリズムを指定して、データベースを作成します。

```
CREATE DATABASE database-file-name
ENCRYPTED TABLE
KEY 'abc'
ALGORITHM AES
```

◆ コマンド・プロンプト - 次のコマンドは、上記の SQL 文と同じデータベースを作成します。ただし、ここでは dbinit ユーティリティを使用しています。

```
dbinit database-file-name -et -ek abc -ea AES
```

ユーザにはコマンド・ラインで暗号化キーを入力するように要求する代わりに、ダイアログ・ボックスを表示して暗号化キーを入力するように要求することをおすすめします (この場合、入力した暗号化キーはマスクされ、プレーン・テキスト形式では表示されません)。具体的には、次のコマンドを実行します。

```
dbinit database-file-name -et -ep -ea AES
```

テーブル・レベルの暗号化でデータベースを作成してから暗号化したテーブルを作成するには、ENCRYPTED 句を CREATE TABLE 文に追加します。次に例を示します。

```
CREATE TABLE Customers(
    MemberID CHAR( 40 ),
    CardNumber  VARCHAR(  30  ) )
    ENCRYPTED;
```

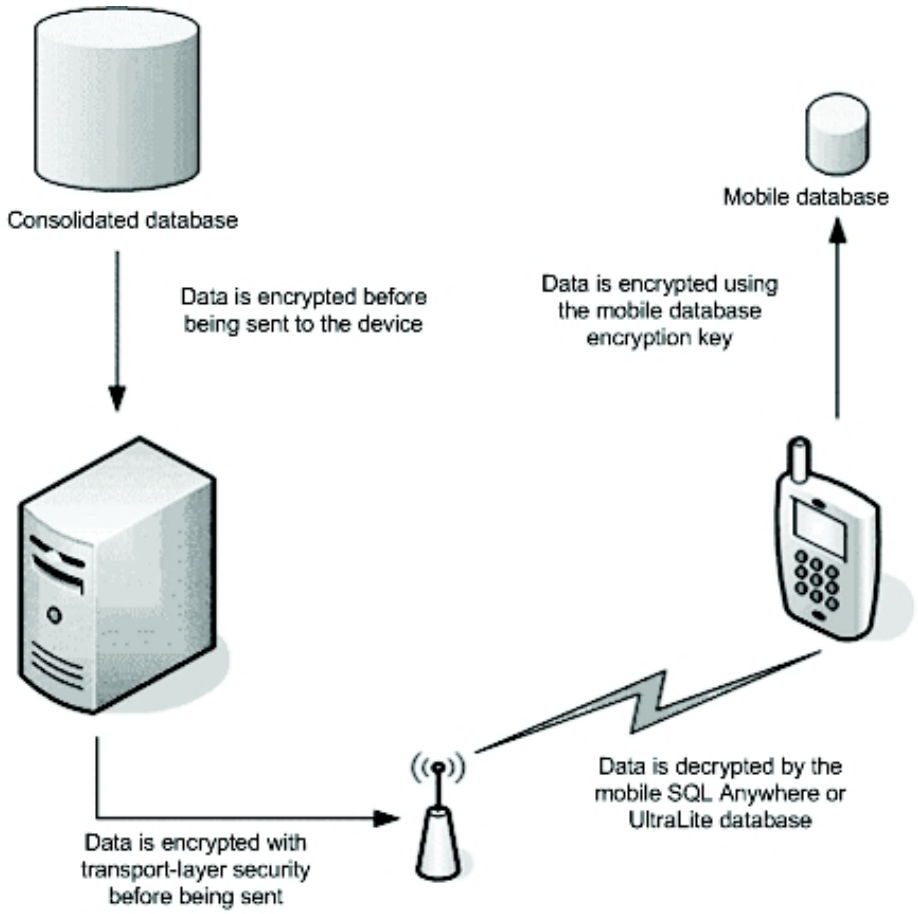
テーブルに含まれる個別の値の暗号化

SQL Anywhere では、ENCRYPT 関数を使用してテーブル内の任意のデータを暗号化することができます。この関数を使用する場合、データベース作成時に特別なパラメータは不要ですが、INSERT や SELECT の操作ごとに暗号化キーを入力する必要があるという欠点があります。なお、ENCRYPT 関数の使用時には、高度な AES 暗号化が使用されます。
ENCRYPT 関数の使用時には、単一のカラム (パスワード・カラムなど) を暗号化することはできますが、テーブル内の他のデータを暗号化することはできません。この場合、挿入データを自動的に暗号化すると同時に、SELECT 文用に自動的に非表示のビューを作成して列を復号化するトリガを作成すると便利です。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の「データベースの暗号化」を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.936)

モバイル・データの暗号化ポリシー

モバイル・デバイスで SQL Anywhere を実行していても、デスクトップ・コンピュータと同じデータ暗号化機能を利用することができます。SQL Anywhere のモバイル版インストールを使用する場合にもデータを暗号化することをおすすめします。



セキュリティの脅威を軽減するため、SQL Anywhere データベース・ネットワーク内の統合データベースがキー X を使用して暗号化されている場合、データは最初にそのキーを使用して復号化されてからモバイル・デバイスに送信されます。これにより、モバイル・デバイス上のデータベースは、必要な場合にのみ暗号化されることになります。また、モバイル・デバイス上のデータが暗号化されている場合 (キー Y を使用)、キー Y にキー X と同じキーを使用することは許容されません。このルールに従っていれば、単一のモバイル・デバイスの暗号化キーが悪用された場合でも、企業データ全体への不正なアクセスを防ぐことはできます。さらに、データベース管理プロシージャでは、モバイル・データベース・インスタンスを暗号化する場合に使用するキーを一定の期間で無効にする (つまり、新しいキーでデータを強制的に再暗号化する) ことをおすすめします。これにより、モバイル・デバイスの暗号化のセキュリティが侵害された場合でも、悪意のあるユーザがキーを悪用することは困難になります。

パフォーマンス分析

データが暗号化されている場合でも、SQL Anywhere はデータの暗号化／復号化の処理効率に優れており、パフォーマンスへの影響はごくわずかです。ラボで行ったテストでは、データベースが AES または AES_FIPS で暗号化された場合、パフォーマンスの低下はデータの取得操作で 2 ～ 3%、データの挿入操作でも 5% に留まっています。この結果は、Pentium 4 2.4GHz、1024MB の RAM を搭載したシステムで、単一のテーブルに対して行った 250 万件のローの挿入／取得操作に基づいています。

通信の暗号化

SQL Anywhere は、ユーザの作業場所にかかわらず重要な業務データを提供できるように設計されたフロント・ライン・データベースの開発において市場をリードしています。業務データを社外に送信する際に、データの暗号化や、悪意のあるユーザからのデータ保護に対して注意を払わなかった場合、セキュリティ上の問題につながる可能性があります。SQL Anywhere にはセキュリティ・メカニズムが組み込まれており、現場の従業員の業務に支障を来すことなく、最大限の保護を実現します。SQL Anywhere は (データベースのユーザ ID とパスワードのみを必要とする) データベース間の非暗号化通信をサポートしていますが、この項では、トランスポート・レイヤ・セキュリティに重点を置いて説明します。

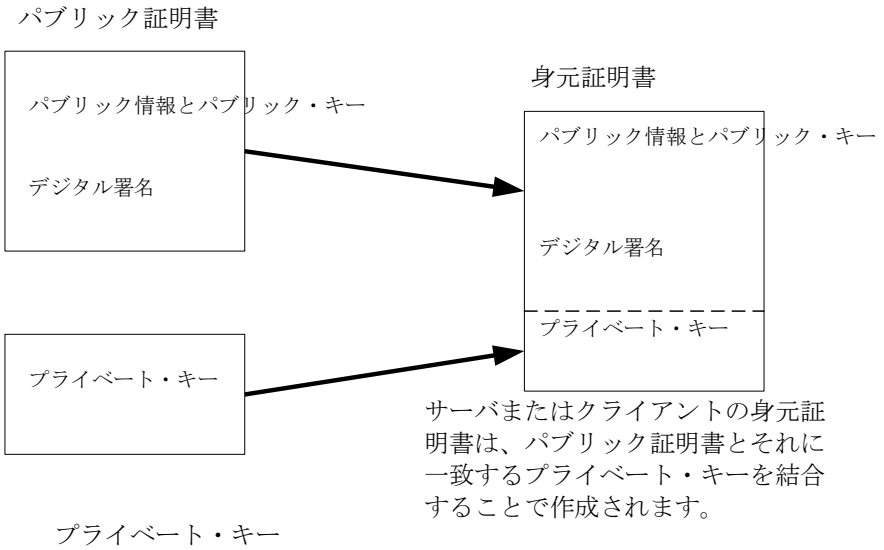
はじめに

SQL Anywhere では、トランスポート・レイヤ・セキュリティ (TLS) を使用して通信の暗号化が行われます。TLS は SSL の後継として位置付けられる暗号プロトコルであり、TCP/IP 上での通信の暗号化を実現する目的で設計されたものです。TLS は Internet Engineering Task Force (IETF) の標準プロトコルであり、デジタル証明書とパブリック・キー暗号方式を使用してクライアント／サーバの通信をセキュリティ保護します。また、SQL Anywhere は、サーバ認証をサポートしています。サーバ認証では、データベース・サーバによって作成されたパブリック証明書と身元証明書をクライアント・アプリケーションが使用して、中央データベース・サーバの身元を検証します。なお、通信が暗号化されているかどうかにかかわらず、クライアント・アプリケーションがデータベース・サーバに接続するために送信するログイン・パケットは常に暗号化されていることに注意してください。

TLS プロトコルは、パブリック・キー暗号方式と対称キー暗号方式を組み合わせた形で採用し、通信のパフォーマンスを向上させています。パブリック・キー暗号方式は優れた認証技法ですが、計算の負荷が大きくなります。そのため、セキュアな通信の確立後、クライアントとサーバは、128 ビット長のキーによる処理効率の高い対称暗号方式を使用して以降の通信を行います。

デジタル証明書

トランスポート・レイヤ・セキュリティは、当事者間でのデジタル証明書の交換に基づいています。通信を開始する前に、各当事者はパブリック証明書とプライベート・キーを結合して身元証明書を生成します。

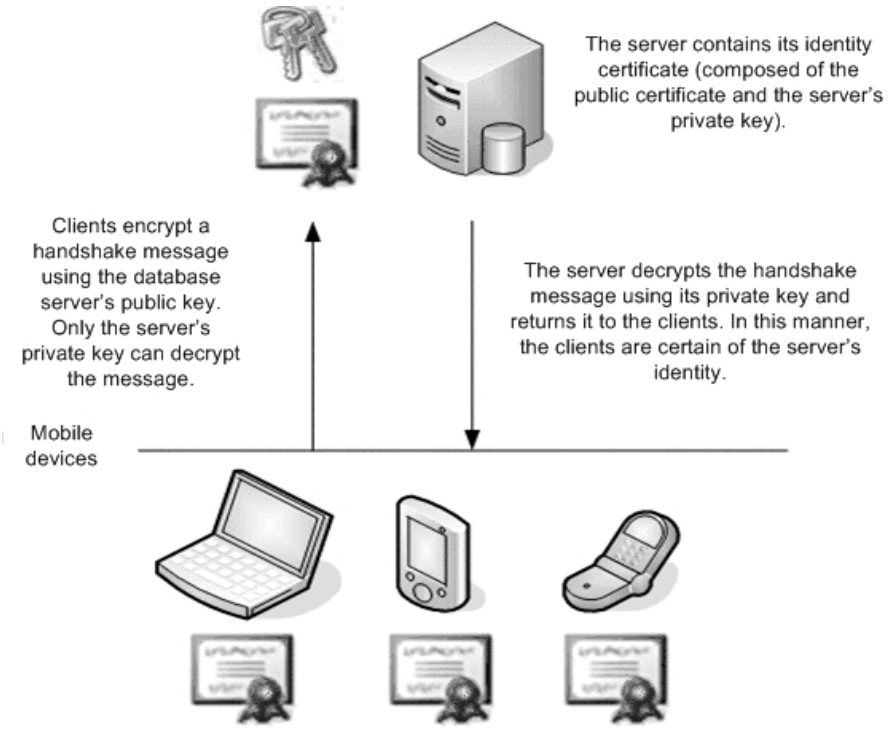


プライベート・キー

パブリック証明書の重要な特徴としては、デジタル署名が挙げられます。デジタル署名は、伝送時のデータの整合性の維持と、改ざんの検出を目的として使用されます。デジタル署名は、証明書に基づいて作成した一意なハッシュ値を署名する (つまりプライベート・キーで暗号化する) ことで生成されます。署名プライベート・キーの保護は非常に重要です。なぜなら、署名プライベート・キーが悪用された場合、悪意のあるユーザは不正なデータ伝送を正当なものとして偽装し、データベース・サーバを欺くことが可能になるからです。なお、証明書の署名方法には、次の 3 種類があります。

◆ 自己署名証明書 - 自己署名されたサーバまたはクライアントの証明書は、(1 台の中央データベース・サーバと複数のリモート・データベースのように) 単純なセットアップ環境向けに使用されます。この場合、パブリック証明書の署名に使用されるプライベート・キーは、中央データベース・サーバ (サーバ認証用) またはリモート・データベース (クライアント認証用) で保管されます。

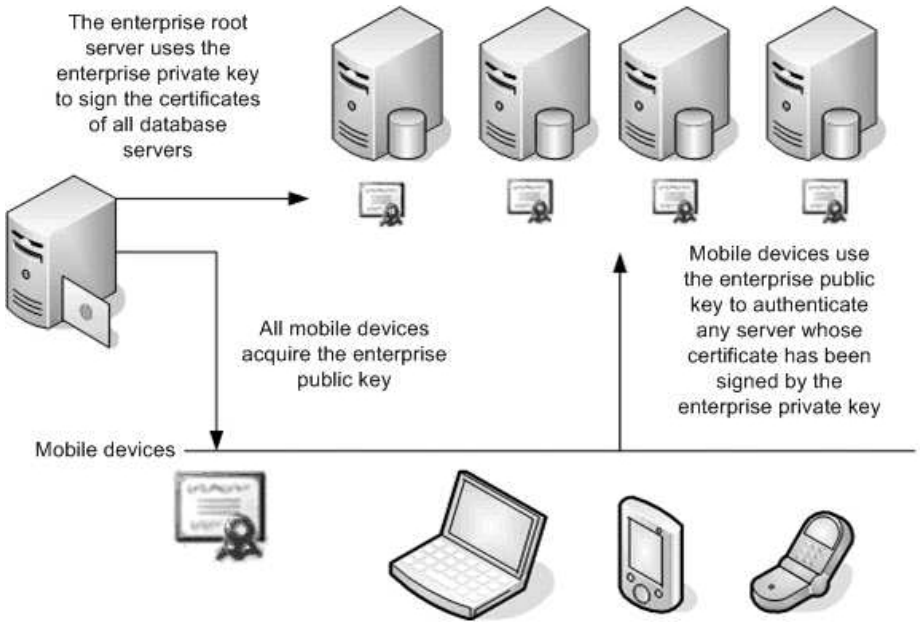
☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の「自己署名ルート証明書」を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.956)



◆ エンタープライズ・ルート証明書 - この署名方法では、複数サーバおよび／または複数クライアントの配備構成でデータの整合性と拡張性が向上します。特定のサーバが認証局の役割を果たし、データベース・システムで必要なすべての署名に使用されるプライベート・キーを保管します。この方法では、クライアントまたはサーバの再構成を行わずに双方を追加できるため、拡張性の面で利点があります。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の「エンタープライズ・ルート証明書」を参照してください。

ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.958)



◆ 商用 (グローバル) 認証局 - 商用認証局 (VeriSign など) を使用する場合でも、証明書の手続きはエンタープライズ・ルート証明書を使用する場合とほぼ同じですが、署名プライベート・キーがローカル環境では保管されない点が異なります。この方法は、他の方法と比較して多くの利点があります。第一に、商用認証局は証明書の署名処理に特化しており、このような要求の処理に最高級のシステムで対応しています。また、物理的なセキュリティと仮想的なセキュリティを複雑に組み合わせる形で実装しています。グローバル認証局は、2 つの異なる組織間でデータをやり取りする場合にも有用です。同じ第三者機関によって署名された信頼できる証明書を利用することで、このようなプロセスに要する時間を大幅に短縮することができます。ただし、グローバル署名証明書を使用する場合、各クライアントまたはサーバは証明書の値 (組織、共通名など) を検証して、認証局が他のユーザ向けに署名した証明書を回避する必要があります。

☞ 詳細については、『SQL Anywhere サーバ - データベース管理』の「グローバル署名証明書」を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.958)

通信の暗号化の有効化

通信の暗号化を有効にするには、使用する暗号化方式 (ECC または RSA) を最初に選択する必要があります。また、FIPS 準拠のアプリケーションを必要とする場合、RSA_FIPS セキュリティも使用する必要があります。選択後、次の手順に従ってクライアントとサーバ間の通信の暗号化を有効にする必要があります。

◆ デジタル証明書の作成 - 身元証明書を作成して、サーバ上の安全な場所に保管する必要があります。また、クライアント証明書を生成して、クライアント・アプリケーションに配布する必要があります。SQL Anywhere には、RSA 証明書または ECC 証明書の生成および署名を支援する gencert.exe というユーティリティが用意されています。

さらに、証明書の読み取りに対応する readcert.exe というユーティリティも用意されています。

次の例では、gencert -r オプションを使用して、RSA 自己署名サーバ証明書を生成しています。

```
gencert -r
Certificate Generation Tool
Choose certificate type ((R)SA or (E)CC): R Enter key
length (512-2048): 1024 Generating key pair...
Country: CA
State/Province: ON Locality: Waterloo
Organization: Sybase, Inc.
Organizational Unit: IAS
Common Name: MobiLink Serial
Number: 123
Certificate valid for how many years: 12
Enter password to protect private key: mypwd123
Enter file path to save certificate: public_cert.crt
Enter file path to save private key: server_key.pri
Enter file path to save server identity: server_cert.crt
```

◆ データベース・サーバの起動 - -ec オプションを使用して、暗号化を有効にした状態でサーバを起動します。このオプションを使用すると、セキュリティの種類とサーバ証明書の情報を指定できます。次の例では、-ec サーバ・オプションを使用して、ECC セキュリティ、サーバ証明書、およびサーバのプライベート・キーのパスワード保護を指定しています。

```
dbsrv10 -ec tls(tls_type=ecc;certificate=c:¥test¥serv1_
        ecc.crt;        certificate_password=mypwd)        -x        tcpip
        c:¥ test¥secure.db
```

◆ クライアント・アプリケーションの構成 - トランスポート・レイヤ・セキュリティを使用するようにクライアント・アプリケーションをセットアップするには、上記の方法で生成した証明書をアプリケーションに適用します。次に、接続文字列で Encryption [ENC] 接続パラメータを使用します。

次の例の接続文字列では、trusted_certificates 暗号化接続パラメータを使用して、パブリック証明書 public_cert.crt を指定しています。

```
"UID=DBA;PWD=sql;ENG=myeng;LINKS=tcpip;
ENC=tls(tls_type=ecc;trusted_certificates=public_cert.crt)"
```

☞ 通信の暗号化をセットアップする場合の詳細については、『SQL Anywhere サーバ - データベース管理』の「トランスポート・レイヤ・セキュリティ」を参照してください。
ftp://ftp2.ianywhere.jp/public/tech/dbdaja10.pdf (P.949)

パフォーマンス分析

通信の暗号化は、送信するデータすべての暗号化と受信する通信トラフィックすべての復号化に伴うオーバーヘッドにより、データベース通信のパフォーマンスに大きな影響を及ぼします。

一般に、RSA は ECC よりも高速であり、ECC は RSA_FIPS よりも高速です。このテストはすべて、変更を加えていない標準的な既製のハードウェアに対して実施されています。

データベース・フェッチ - 暗号化通信チャネル上でデータをフェッチした場合、RSA 暗号化通信は非暗号化通信よりも 50% 低速になります。この結果は、Pentium 4 2.4GHz、1024MB の RAM を搭載したシステムで、250 万件のローを取得するテストに基づいています。
データベース挿入 - データの挿入時も、フェッチ時と同様に暗号化通信は非暗号化通信よりも低速になりましたが、その差はデータベースのフェッチ結果の場合よりも小さなものでした。データの挿入時の RSA 暗号化通信と非暗号化通信のパフォーマンスの差は 40% でした。この結果は、Pentium 4 2.4GHz、1024MB の RAM を搭載したシステムで、250 万件のローを挿入するテストに基づいています。

SQL インジェクション

SQL インジェクションとは、動的に生成された SQL 文が処理を目的としてアプリケーション・レイヤからデータベース・サーバに渡された場合に発生する可能性のあるセキュリティの脆弱性のことを指します。この脆弱性が動的な SQL 文の生成時に発生すると、ユーザ指定の値は適切にエスケープされず、ユーザが自身の SQL コードを入力することで不正なデータへのアクセス権を取得することが可能になります。今日、インターネットの拡大に伴い、大量の機密情報を不正に取得できる可能性のある商用サイトが SQL インジェクション攻撃の最も一般的なターゲットになっています。SQL インジェクションは単純であることから見過ごされがちであり、セキュリティの脅威としては評価されず、防護策では企業の他の領域が重視されています。

例
次のコードは、データベースに渡されたときにユーザのログインを認証する SQL 文を生成します (返されたデータ・セットが空の場合、ログインに失敗しています)。

```
String statement = "SELECT * FROM users WHERE
  userId = '" + _userId + "' AND password = '" + _password + "';" ;
```

ユーザが適切なパスワードを入力せずにユーザ名のフォーム・ボックスに anyUser と入力し、次の内容をパスワード・ボックスに入力した場合、どのような現象が発生するでしょうか。

```
password' OR ''='
```

動的に生成される SQL 文は、次のようになります。

```
SELECT * FROM USERS WHERE uid = 'anyUser' AND password =
        'password' OR ''='';
```

この文の実行結果は意図した結果と異なります。データベースはすべてのユーザを返すため、結果として、悪意のあるユーザがシステムに対する不正なアクセス権を取得する可能性があります。

悪意のあるユーザが特にデータベース内のデータに損傷を与えることを目的とした場合、次の内容をパスワード・フィールドに入力するという方法が考えられます。

```
password'; DROP TABLE accounts; SELECT * FROM users WHERE uid =   'admin
```

この場合、アカウント・テーブルが削除され、悲惨な結果がもたらされると考えられます。

SQL インジェクションの防止

データベース・サーバでは SQL インジェクションは完全に正当なものと認識されてしまうため、この問題の予防策の対象となるのは、アプリケーション・レイヤと、アプリケーションによるデータベースからの SQL 実行の呼び出し方法に関するものがほとんどです。

アプリケーション・レイヤでの SQL インジェクションの防止

SQL インジェクション攻撃に対処するためのツールがほとんどのプログラミング言語やデータ・アクセス API に用意されるようになったことで、ここ数年では、従来よりも単純な修正で、SQL インジェクション攻撃に対するアプリケーションの弱点を克服できるようになりました。

エスケープ文字

プログラミング言語によっては、入力された文字列値をチェックして特殊なエスケープ文字を含んでいるかどうかを判別する関数 (Perl の DBI::quote など) が個別に用意されています。このような関数では SQL コードが検出されるわけではなく、エスケープ文字 (文字列を終端する文字など) が削除された結果、データベース・サーバから SQL 構文エラーが返されて、文の実行に失敗する場合があります。

値のプレースホルダ

インジェクション攻撃を防ぐ適切な方法としては、クエリの作成時に値のプレースホルダを使用する方法が挙げられます。この技法は、本来 SQL コードを持たない値 (パスワード変数など) に SQL コードを検出する場合に便利です。次に示すのは、C# コードの例です。

```
string commandText = "SELECT * FROM Customers WHERE
            Country=@CountryName";
SqlCommand cmd = new SqlCommand(commandText, conn);
cmd.Parameters.Add("@CountryName",countryName);
```

この場合、ADO.NET には強力な型のチェックが用意されているため、countryName に単純な文字や数値以外の要素が含まれている場合、C# はエラーのフラグを立てます。ただし、プレースホルダを使用できるかどうかは、DBMS 自体から提供される機能や、使用するデータ・アクセス API (上記の例の ADO.NET など) から提供される機能に依存します。

また、SQL クエリを動的に作成する代わりに、ホスト変数を利用する方法も考えられます。

ストアド・プロシージャ

インジェクション攻撃に対抗する最も有効な方法としては、データベースのストアド・プロシージャを使用する方法が挙げられます。ストアド・プロシージャを使用する方法は、値のプレースホルダを使用する方法と似ていますが、重要な相違点がいくつかあります。第一に、ストアド・プロシージャはデータベースに格納されており、アプリケーション・コードに必要なパラメータを渡して呼び出す必要があります。

データベース・セキュリティを適切に設定した場合 (34 ページの「データベース・レイヤでの防止」を参照)、プロシージャ自体の内部処理を公開せずに、ストアド・プロシージャの処理に失敗したことがアプリケーションに通知されます。

第二に、ストアド・プロシージャでは、型のチェックが強制されます。ストアド・プロシージャで INTEGER 値を想定している場合、有効な整数値以外の要素はサーバ・エラーを引き起こします。また、プロシージャでサイズ 10 の VARCHAR 変数を受け取る場合、データベース・サーバ・エラーを引き起こさずにコードのインジェクトを行うことは困難です。ストアド・プロシージャを適切に実装すれば、SQL インジェクションの機会を排除する上で大いに役立つと考えられます。ただし、変数値については、アプリケーション側でもインジェクション・コードの可能性をチェックすれば、保護の強化につながるという点に留意する必要があります。

データベース・レイヤでの防止

SQL インジェクション攻撃は正当な SQL 文に基づくものですが、データベース・サーバで悪意のあるコードを識別し、データベースのセキュリティを強化して悪意のあるコードによる危険を限定することは可能です。
通常の運用配備では、厳密にアプリケーションによって使用される独立したユーザ・アカウントが常に存在するはずです。DROP TABLE などの文について、悪意のあるユーザによる使用を防ぐには、次のようにアプリケーション・ユーザのパーミッションの範囲を制限する必要があります。

◆ 通常の操作時に必要な SQL 関数のみに制限する。

◆ 必要な情報を含むテーブルのみに制限する。

◆ ストアド・プロシージャを使用する場合、アプリケーション・ユーザに付与するパーミッションの範囲は、必要なストアド・プロシージャの実行のみに制限する。

また、監査により、成功した SQL インジェクション攻撃のインスタンスを識別して、システムに被害が発生する可能性を無効にすることもできます。

URL のインジェクション

SQL インジェクションについて論じる際には、URL の内容についても認識しておくことが重要です。具体的には、Web サイトがその Web ページの URL に SQL 文を公開していた場合 (複数のページを対象とする SQL 文を作成する場合など)、不正行為を働く余地があるということです。URL に SQL コード (SQL 文の作成に使用するパラメータ) を公開することは、予期しないセキュリティの侵害をもたらす可能性があり、手法としては避けるべきです。この場合、基本的な SQL 知識しか持たない初心者ユーザでも、不正行為を働いてシステムに被害を与えることが可能になります。

まとめ

今日、組織における情報の価値は上昇し続け、データベース・セキュリティは重要な課題になっています。あらゆる組織が年度末の会計報告と同じくらい真剣にデータベース・セキュリティに取り組むことが求められています。

企業の所有するデータが悪用された場合、財務的な観点と広報的な観点の両方で、被る損害は多大なものとなる可能性があります。

SQL Anywhere 10 は、データを確実にセキュリティ保護する上で必要なツールを組織に提供します。このマニュアルでは、問題の主要な領域について簡潔に説明するとともに、企業の所有するデータについて、SQL Anywhere データベースに格納する場合や、ネットワーク上のいかなる場所に伝送する場合でも、安全性とセキュリティを確保するために必要な手順について説明してきました。

付録 A ：SQL Anywhere セキュリティ・チェックリスト

◆ 適切に保護されたセキュアな環境にデータベース・サーバ・マシンが配備されている。

◆ 必要なネットワーク・セキュリティ対策が実施されている。

◆ オペレーティング・システムの不要なサービスやデーモンがすべて無効になっている。

◆ 厳密なオペレーティング・システム・ユーザ・アカウント・ポリシーが実装されており、データベース・サーバ・マシンにログインできるユーザが少数に限定されている。

◆ 統合化ログインの使用時に、Windows の Guest ユーザ・アカウントが無効になっており、その他のユーザのログインでは強力なパスワードが常に使用されている。

◆ モバイル・デバイスのセキュリティ対策が実施されている。

◆ パスワード、証明書、または非従来型の認証方式に基づいて、デバイスへのアクセスがセキュリティ保護されている。

◆ アプリケーション・レベルのセキュリティ対策が実施されている。
- リモート・デバイス管理およびパッチ管理を利用できる。
- モバイル・デバイス上のデータが強力な暗号化方式で暗号化されている。
- モバイル・デバイスと企業ネットワークの通信がセキュリティ保護されている。
- 企業所有のデバイスのみが企業ネットワークへの接続を許可されている。
- 前述のように適切な IT ポリシーおよびプロシージャが実施されている。

◆ DBA 権限が可能なかぎり少数のユーザに制限されている。

◆ データベースの起動、停止、作成、ロード、およびアンロードを許可するパーミッションが DBA ユーザのみに制限されている。

◆ データベース・ユーザ・パスワードに関するセキュリティ対策が実施されている。
- カスタム・ログイン・プロシージャが実装されている。
- パスワードに最小長が設定されている。
- パスワードが最小レベルの複雑性を必ず満たしている。
- デフォルトの DBA アカウント・パスワード (sql) が変更されているか、または (推奨) DBA アカウントが無効化され、DBA 権限を持つ別のアカウントで置き換えられている。

・ パスワードが ODBC データ・ソースに含まれていない。
・ ログイン試行の失敗回数が制限されている。

◆ 不要な SQL Anywhere ユーティリティがサーバ上に存在しておらず、ユーティリティによるセキュリティ侵害を防ぐための対策が実施されている。

◆ データベース要求ログがデフォルトで有効になっておらず、必要に応じて、使用後即座に削除される。

◆ データベース・サーバのメッセージ・ウィンドウが無効になっている。

◆ データベース監査が有効になっており、dbtran ユーティリティへのアクセスが制限されている。

◆ バックアップおよびバックアップ・プロシージャがすべて適切にセキュリティ保護されている。

◆ データベースの Web サービスが HTTPS に基づいてセキュリティ保護されており、ユーザ認証を必要とする。

◆ 強力なデータベースおよびテーブルの暗号化が使用されている。

◆ 暗号化キーが強力である。

◆ データベースの起動時に、ダイアログ・ボックスを使用して、データベース暗号化キーを入力するようにユーザに要求している。

◆ モバイル／統合データベースとの通信がトランスポート・レイヤ・セキュリティに基づいてセキュリティ保護されている。

◆ 可能な場合、強力な型チェックに基づくストアド・プロシージャを使用して、不正なコードの実行や SQL インジェクションを防いでいる。

◆ 情報セキュリティ・プロシージャや実践方法について、従業員に対して十分なセキュリティ・トレーニングと情報が提供されている。

付録 B ：強力なパスワードのガイドライン

今日、コンピュータ・システムをセキュリティ保護する方法の定番として挙げられるのがパスワードです。デスクトップ・ワークステーションから顧客の信用情報を保管する重要なデータベースに至るまで、パスワードはさまざまな場面で利用されています。個々のパスワードの目的にかかわらず、パスワードをセキュリティ保護し、組織のネットワークにおける侵入経路を塞ぐことが共通の重要課題となっています。
強度に問題がある脆弱なパスワードは、依然として今日のセキュリティ・システムが直面している大きな問題の 1 つです。この問題の理由としては、特に強制されないかぎり、ユーザが自分の名前、物の名前、重要な日付など、覚えやすいパスワードを選ぶ傾向があるということが挙げられます。これでは、攻撃者が膨大な量の一般的な単語を使用してユーザのパスワードを推測する辞書攻撃 (ディクショナリ・アタック) のような技法を駆使すれば、脆弱なパスワードは簡単に破られてしまいます。その具体的な攻撃方法としては、パスワードを変化させながらログインの試行を延々と繰り返す方法や、暗号化されているパスワード格納ファイルを入手して、その値を事前に暗号化した数百万語もの辞書の単語と比較する方法などが考えられます。

このような処理は驚くほど短時間で終了します。たとえば、デュアルプロセッサを搭載したワークステーション 2 台で 100,000 語の辞書ファイルを使用すると、数百個のパスワードをわずか数分で破ることが可能です。

強力なパスワードは、次のように特徴付けられます。

◆ 長さが少なくとも 8 文字以上

◆ 少なくとも 1 つの数字を含む

◆ 少なくとも 1 つの小文字を含む

◆ 少なくとも 1 つの大文字を含む

◆ 少なくとも 1 つの記号 (!、@、#、$、^) を含む

また、次のような文字は、パスワードとして使用すべきではありません。

◆ ユーザ名

◆ 従業員番号

◆ 名前 (ファースト・ネーム)

◆ 家族、友人、同僚の名前

◆ ニックネーム

◆ 社会保障番号や社会保険番号

◆ 運転免許証番号

◆ 誕生日

◆ ナンバー・プレートの番号

◆ 住所や地名

◆ 電話番号

◆ 市区町村名、都道府県名、国名

◆ 社名、部署名、部門名

◆ コンピュータ用語、コマンド、Web サイト、ハードウェア用語、ソフトウェア名

◆ 一般的な用語や頭字語

◆ 単語や数字のパターン (aaddoo22 や a1s2d3f4 など)

◆ 車両、航空機、船舶のメーカー名やモデル番号

◆ 俗語や俗語表現

◆ わいせつな語句

◆ 学校名や学校の通称

◆ 好きな食べ物、スポーツ、テレビ番組、飲み物などの名前

◆ 辞書に掲載されている語句

◆ 上記のいずれかの語順を逆にしたもの

◆ 公共の Web サイトで使われているパスワード

◆ パスワードによく使われる語句

付録 C ：ソーシャル・エンジニアリング

コンピュータのセキュリティ・システムは、セキュリティ保護の対象や、セキュリティ対策の実装方法についてはそれぞれ異なります。ただし、「人的要因(ヒューマン・ファクタ)」、つまり正規のユーザが不正な操作を行えば、システム全体を簡単に停止させることができるという弱点は、すべてのセキュリティ・システムに共通しています。どれだけ強力な手段でシステムをセキュリティ保護しても、悪意のあるユーザが、システムを日常的に操作しているユーザを巧みに操ってシステムにアクセスすることは可能です。このように、コンピュータ・システムに不正にアクセスするために人間関係を利用して機密情報を入手する行為は、「ソーシャル・エンジニアリング」と呼ばれます。
コンピュータ・システムに関連するソーシャル・エンジニアリングで利用されるのは、ユーザ同士の互いの信頼関係です。ソーシャル・エンジニアリングを試みる人物は、このような信頼関係を利用して機密情報にアクセスします。ほとんどの事例では、機密データへのアクセス権を持つ人物のユーザ ID／パスワードを聞き出すことが狙いとされています。一般的な事例としては、システム管理者を装って企業の従業員に電話で連絡を取る方法が挙げられます。一般部門の従業員が IT 部門の担当者の大半と親しい可能性はそれほど高くないと仮定すると、システムのリセットにパスワードが必要であると IT 部門の担当者に要請された場合、従業員が自分のパスワードを教えてしまう確率は高いと考えられます。

ソーシャル・エンジニアリングの働きかけは通常、電話、電子メール、インスタント・メッセージング、時には対面で行われます。

組織がソーシャル・エンジニアリング攻撃に対する弱点を克服する方法としては、さまざまな手順・手続きが考えられます。

◆ 従業員全員に対してセキュリティ関連の講習会を実施し、ソーシャル・エンジニアリングとは何か、ソーシャル・エンジニアリングを試みる人物がどのような方法で情報にアクセスするのか、どのような予防策を講じればこのような攻撃を防ぐことができるのかといった点について説明します。ソーシャル・エンジニアリングが狙い通りに機能する最大の理由としては、一般に従業員は自分が攻撃の対象になっていることを認識していない点が挙げられます。したがって、講習会を実施することでソーシャル・エンジニアリングに対する従業員の認識が高まれば、被害が拡大する可能性は低くなると考えられます。実際の講習会では、次のようなトピックについて説明する必要があります。

- ソーシャル・エンジニアリングを試みる人物の正体と、このような人物が対象の信頼を得るために試みる方法。
- 連絡を取ってきた人物がソーシャル・エンジニアリングを働きかけている可能性を示す危険な兆候の具体例。
- 企業の運営上、どのような情報に価値があり、どのような代償を払ってもその情報を保護する必要があるという点について、従業員として理解しておく必要があること。
- パスワード・ポリシーの具体的な内容。パスワードは個人の責任で管理すべきものであり、他のユーザに安易に教えるものではないということの周知徹底。
- 企業独自の用語、担当者の氏名および役職、サーバ名といった情報は、別部門の従業員に対するソーシャル・エンジニアリング攻撃に使用される可能性があるため安易に口外すべきではないということ。

・ 見知らぬ人物から送信された電子メールを開いたときに、有害な可能性のあるファイル (実行可能ファイルなど) が添付されている場合の注意事項。

◆ 企業のセキュリティ・ポリシーを従業員に対してわかりやすく説明し、その実施を図る必要があります。

◆ 企業のセキュリティ・ポリシーに関して、従業員に周知徹底するプロセスを策定します。

◆ 専用のログを作成し、セキュリティ関連の記録を一元管理します。疑わしい電話を受けた場合や、企業の込み入った事情に関心を示す疑わしい人物に出会った場合に、従業員はその都度ログを更新する必要があります。このようなログにより、ソーシャル・エンジニアリング攻撃の発生を早い段階で察知し、セキュリティ対策を進めることができると考えられます。

◆ 従業員の身元を確認するため、セキュリティ関連の質問集を用意しておきます。具体的には、従業員の個人情報や意図的な虚偽の質問などが役立ちます。

付録 D ：パスワード検証関数の例

この例では、さまざまなプロシージャや関数を定義し、それらを組み合わせて実装することで、高度なパスワードのルールを実現します。具体的には、パスワードに特定の型の文字を要求したり、パスワードの再利用を禁止したり、パスワードの有効期限を設定したりします。これらのプロシージャや関数は、ユーザ ID の作成時、パスワードの変更時、ユーザの接続時などに、verify_password_function や login_procedure オプションを使用してサーバによって呼び出されます。また、パスワードが期限切れになる前に変更する必要があることを通知するため、post_login_procedure によって指定されたプロシージャがアプリケーションから呼び出される場合もあります。

```
-- only DBA should have permissions on this table CREATE
TABLE DBA.t_pwd_history(
        pk              INT             DEFAULT AUTOINCREMENT PRIMARY KEY,
        change_date TIMESTAMP   DEFAULT CURRENT TIMESTAMP,
                                        -- when pwd set
        grace_date  DATE,               -- a day after password expires to
                                          -- allow user to log in
        user_name   CHAR(128),          -- the user whose password is set
        pwd_hash        CHAR(32) ); -- hash of password value to detect
                                        -- duplicate passwords
```

```
-- called on every GRANT CONNECT TO ... IDENTIFIED BY ...
  -- statement to verify that the password conforms to
-- the password rules
CREATE FUNCTION DBA.f_verify_pwd( uid                    VARCHAR(128),
                                                    new_pwd VARCHAR(255) )
RETURNS VARCHAR(255) BEGIN
        -- a table with one row per character in new_pwd DECLARE
        local temporary table pwd_chars(
                    pos INT PRIMARY KEY,            -- index of c in new_pwd
                    c   CHAR( 1 CHAR ) ) NOT TRANSACTIONAL;              --
                character
        -- new_pwd with non-alpha characters removed
        DECLARE pwd_alpha_only              CHAR(255);
        DECLARE num_lower_chars             INT;

        -- enforce minimum length (can also be done with
        -- min_password_length option)
        IF length( new_pwd ) < 6 THEN
                RETURN 'password must be at least 6 characters long'; END IF;

        -- break new_pwd into one row per character
        INSERT INTO pwd_chars SELECT row_num, substr(
                                                new_pwd, row_num, 1 )
                                            FROM dbo.RowGenerator
                                            WHERE row_num <= length( new_pwd );

        -- copy of new_pwd containing alpha-only characters
        SELECT LIST( c, '' ORDER BY pos ) INTO pwd_alpha_only
                FROM pwd_chars WHERE c BETWEEN 'a' AND 'z' OR c BETWEEN 'A'
                AND 'Z';

        -- number of lower case characters IN new_pwd SELECT
        COUNT(*) INTO num_lower_chars
                FROM pwd_chars WHERE CAST( c AS BINARY ) BETWEEN 'a' AND 'z';

        -- enforce rules based on characters contained in new_pwd
        IF ( SELECT count(*) FROM pwd_chars WHERE c BETWEEN '0' AND  '9' )
                    < 1 THEN
                RETURN 'password must contain at least one numeric digit';
        ELSEIF length( pwd_alpha_only ) < 2 THEN
                RETURN  'password must contain at least two letters'; ELSEIF
        num_lower_chars = 0
                    OR length( pwd_alpha_only ) - num_lower_chars = 0 THEN
                RETURN 'password must contain both upper- and lowercase characters';
        END IF;
```

```
        -- not the same as any user name
        -- (this could be modified to check against
          -- a disallowed words table)
        IF EXISTS( SELECT * FROM SYS.SYSUSER
                                WHERE  lower(  user_name  )  IN  (  lower(  pwd_
                alpha_only ),
                                                                        lower( new_pwd
                ) ) ) THEN
              RETURN 'password or only alphabetic characters in
                password ' ||
                        'must not match any user name';
        END IF;

        -- not the same as any previous password for this user
        IF EXISTS( SELECT * FROM DBA.t_pwd_history
                                WHERE user_name = uid
                                   AND pwd_hash = hash( uid || new_pwd, 'md5'
                ) ) THEN
              RETURN 'previous passwords cannot be reused'; END IF;

        -- save the new password
        INSERT INTO DBA.t_pwd_history( user_name, pwd_hash )
              VALUES( uid, hash( uid || new_pwd, 'md5' ) );

        RETURN( NULL );
END;
ALTER  FUNCTION  DBA.f_verify_pwd  SET  HIDDEN;
GRANT EXECUTE ON DBA.f_verify_pwd TO PUBLIC;
SET OPTION PUBLIC.verify_password_function = 'DBA.f_verify_pwd';

-- called on every connection to check for password expiry CREATE
PROCEDURE DBA.p_login_check()
BEGIN
        DECLARE uid                     CHAR(128);
        DECLARE     INVALID_LOGON       EXCEPTION  FOR  SQLSTATE  '28000';
        DECLARE      last_pwd_change    DATE;
        DECLARE grace_date              DATE;
        DECLARE is_dba                  CHAR;
        DECLARE msg_str                 CHAR(255);
```

```
SET uid = connection_property( 'Userid' );
IF ( EXISTS( SELECT * FROM DBA.t_pwd_history WHERE user_name = uid ) )
THEN
      SELECT FIRST  t.change_date, t.grace_date INTO
      last_pwd_change, grace_date
                  FROM  t_pwd_history  t  WHERE  t.user_name  =  uid
                  ORDER BY t.change_date DESC;
END IF;
IF last_pwd_change IS NULL THEN
      -- no password change, so create one now.
      INSERT INTO DBA.t_pwd_history( user_name, pwd_hash )
            VALUES( uid, 'unknown' );
      COMMIT WORK;
ELSE
      IF EXISTS( SELECT * FROM SYS.SYSUSERAUTHORITY a,
      SYS.SYSUSER u
                        WHERE u.user_name = uid AND u.user_id =
        a.user_id AND
                                    a.auth = 'DBA' ) THEN
            SET is_dba = 'Y';
      ELSE
            SET is_dba = 'N'; END IF;

                        -- remove any locks on t_pwd_history and
        SYSUSERAUTHORITY
      ROLLBACK WORK;
      -- was last password change > five months ago
      IF CURRENT DATE > dateadd( month, 5, last_pwd_change ) THEN
            -- Never expire DBA accounts so that the database
            --  does not get locked out by all users.
            IF CURRENT DATE < dateadd( month, 6, last_pwd_change ) OR
                  is_dba = 'Y' OR
                  ( grace_date IS NOT NULL AND grace_date = CURRENT
        DATE ) THEN
                  SET msg_str = 'The password for user ' || uid || ' expires on ' ||
                                    CAST( dateadd( month, 6, last_pwd_
        change )
                                                AS DATE ) ||
                                    '.  Please change it before it
        expires.';
                  MESSAGE msg_str TO CONSOLE;
```

```
                -- The post_login_procedure option is set to
                -- p_post_login_check, which will cause a dialog
                  -- to bedisplayed notifying the user their
                  -- password willexpire soon.  dbisql and dbisqlc
                                                                -- will
        display this dialog, and user applications
                                                                -- can call
        the post_login_procedure and display
                  -- this dialog.
                                                            -- May want to
        use xp_send_mail to notify user
                  -- and/or administrator. ELSEIF
            grace_date IS NULL THEN
                -- Allow one grace login day. The first login on
                  -- the grace day fails to ensure the user knows
                  -- their password has expired
                UPDATE  DBA.t_pwd_history  t  SET  t.grace_date  =
        CURRENT DATE
                    WHERE t.grace_date IS NULL AND t.user_name =
        uid;
                COMMIT WORK;
                SET msg_str = 'The password for user ' || uid || ' has expired,
                but future logins
        will ' ||
                                'be allowed today only so that the
        password ' ||
                                'can be changed.';
                MESSAGE msg_str TO
                CONSOLE; RAISERROR 28000
                msg_str; RETURN;
            ELSE
                SET msg_str = 'The password for user ' || uid || ' has expired
                and must be reset by
        your DBA.';
                MESSAGE msg_str;
                --  may  want  to  use  xp_send_mail  to  notify
        administrator.
                RAISERROR 28000 msg_str;
                RETURN;
            END IF;
        END IF;
    END IF;

    CALL    dbo.sp_login_environment;
END;
GRANT EXECUTE ON DBA.p_login_check TO PUBLIC;
SET OPTION PUBLIC.login_procedure = 'DBA.p_login_check';
```

```
-- called by dbisql, dbisqlc and some user applications on every  successful
-- connection to check for warnings which should be displayed CREATE
PROCEDURE DBA.p_post_login_check()
RESULT( warning_text VARCHAR(255), warning_action INT ) BEGIN
      DECLARE uid                   CHAR(128);
      DECLARE last_pwd_change DATE;
      DECLARE warning_text          CHAR(255);
      DECLARE warning_action  INT;

      SET uid = connection_property( 'Userid' ); SELECT
      FIRST t.change_date
            INTO last_pwd_change
            FROM DBA.t_pwd_history t WHERE t.user_name = uid
            ORDER BY t.change_date DESC;
      IF CURRENT DATE > dateadd( month, 5, last_pwd_change ) THEN SET
      warning_text = 'Your password expires on ' ||
                                    CAST( dateadd( month, 6, last_pwd_
            change )
                                          AS DATE ) ||
                                    '.  Please change it before it
            expires.';
            SET warning_action = 1;
      ELSE
            -- There is no warning
            SET warning_text = NULL;
            SET warning_action = 0; END
      IF;
      -- Return the warning (if any) through this result set SELECT
      warning_text, warning_action;
END;

GRANT EXECUTE ON DBA.p_post_login_check TO PUBLIC;
SET OPTION PUBLIC.post_login_procedure = 'DBA.p_post_login_  check';
```

法的注意