ORACLE
OPTIMIZED SOLUTIONS

完全に暗号化されたデータセンター

Oracle SPARCサーバーを用いたデータセンターの暗号化

Oracle ホワイト・ペーパー | 2016 年 6 月

ORACLE

目次

概要

このドキュメントでは、Oracle Solaris 11 のセキュリティ機能、およびオラクルの SPARC サーバーのハードウェアによる暗号化機能を使用してアプリケーションを保護する方法について説明します。このドキュメントでは、Oracle Solaris 11 が動作する SPARC サーバーを使用した多層アプリケーションでのエンド・ツー・エンドのアプリケーション・セキュリティ・シナリオ、技術的な前提条件、構成・導入・検証のガイドラインについて説明します。さらに、パフォーマンスとデータ保護の主要な機能となる、SPARC プロセッサの Oracle ハードウェアによる暗号化機能についても説明します。それによってもたらされるセキュリティ上の利点は、アプリケーション・ソフトウェア、ミドルウェア、インフラストラクチャ・ソフトウェアといったさまざまなソリューションに活用できます。

対象読者および前提となる知識

このドキュメントはセキュリティ担当者、およびセキュリティに関わるアプリケーションの開発者と管理者を対象としています。開発者および管理者は、Oracle SPARC サーバー、Oracle Solaris 11、Oracle Advanced Security とその Transparent Data Encryption 機能、ネットワーク暗号化、Oracle HTTP Server、および Secure Sockets Layer（SSL）と Transport Layer Security（TLS）プロトコルを使用してセキュアな通信を行うためのアプリケーション・セキュリティ技術に精通していることを前提にしています。

Oracle SPARCプロセッサの役割

IT 業界ではセキュリティがますます重要視され、組織はビジネスと情報を不正アクセスから保護し機密性と整合性を確保するために、データ送信時や保存時に積極的に暗号化技術を採用しています。暗号化は演算の負荷が非常に高く、CPU サイクルとネットワーク帯域幅の増加がシステムに負担となるため、システムとアプリケーションのスループットが大幅に低下します。たとえば、1 秒間に 1,000 件のトランザクションを処理できるサーバーが、アプリケーションとの通信を保護するために SSL を導入すると、1 秒間に 10 件のトランザクションしか実行できなくなります。

暗号化を高速化するため、セキュリティ専門家は暗号化アプライアンスを推奨し、使用しています。これにより、暗号化をオフロードし、CPU サイクル数を削減して、システムのスループットを強化できます。暗号化をオフロードするための特別なアプライアンスを採用することは有用ですが、新たなコスト、複雑さ、調達の問題、追加のインストール、構成、テスト手順、管理、そして電源要件と導入プロジェクトのコストが大幅に増加します。Oracle は、暗号化ワークロードの要件を満たすことができる特定目的のハードウェアが必要になると見越して、業界初の最速オンチップ・ハードウェア暗号化機能を Ultra SPARC T1 プロセッサに導入しました。そして、Oracle は Ultra SPARC T1 プロセッサを 2005 年に発売して以来、SPARC プロセッサの新しい世代ごとに暗号化サポートを強化しています。

SPARC M7プロセッサ—統合された暗号化アクセラレーション

新しい Software in Silicon 機能と画期的なキャッシュおよびメモリ階層を組み合わせることで、オラクルの SPARC M7 プロセッサは劇的に処理速度を高速化し、マルウェアおよびソフトウェア・エラーに対する革新的な保護を可能にしています。

SPARC M7 プロセッサの Silicon Secured Memory 機能は、ポインタに関連するソフトウェア・エラーやポインタを悪用するマルウェアからデータを保護するためにリアルタイムなデータ整合性チェックを実行可能です。これにより、ソフトウェアで実現しようとした場合に非常にコストのかかる機能を、低オーバーヘッドのハードウェア監視により実現しています。Silicon Secured Memory により、アプリケーションのメモリ・アクセスのバグや不正なメモリ・アクセスを特定したり、原因を診断したり、適切なリカバリ・アクションを講じたりできるようになります。また、SPARC M7 プロセッサは、特定のソフトウェア機能を高速化するハードウェア・ユニットを組み込んでいます。8 つのオンチップ・アクセラレータはデータベースの問合せ処理をオフロードし、リアルタイムのデータ圧縮解凍を実行します。インメモリ・クエリ・アクセラレーション機能は、他のプロセッサと比べて最大で 10 倍速いパフォーマンスを実現します。インライン圧縮解凍機能により、パフォーマンスを損なうことなく、同じメモリ・フットプリントで最大で 3 倍多いデータを保存できます。

SPARC M7 プロセッサには、各プロセッサ・コアに直接統合された暗号化命令アクセラレータが搭載されています。このアクセラレータにより、業界標準の 10 数種類以上の暗号スイートで高速暗号化が可能になり、セキュアなコンピューティングに通常伴うパフォーマンスやコストの障壁が排除されます。

表 1 は、SPARC プロセッサでサポートされている暗号化アルゴリズムを示しています。競合するプロセッサの、オンチップ/オンコアでの実装と比較して、SPARC プロセッサは多くの公開鍵暗号化、対称キー暗号化、メッセージ・ダイジェスト・アルゴリズムをサポートする一連のアルゴリズムを提供しています。

表1：SPARCプロセッサでサポートされる暗号化アルゴリズム

アルゴリズムの種類	アルゴリズム
アクセラレータ・ドライバ	Userland（ドライバ不要）
公開鍵暗号化	RSA,、DSA、DH、ECC
バルク暗号化	AES、DES、3DES、R4、Kasumi、Camelia
メッセージ・ダイジェスト	CRC32c、MD5、SHA-1、SHA-224、SHA-256、SHA-384、SHA-512
API	PKCS#11 標準、Ucrypto API、Java Cryptography Extension、OpenSSL

SPARCプロセッサの暗号化操作モデル

SPARC プロセッサでは、アプリケーションがコアの暗号化機能に直接アクセスして、ハードウェアでこれらの機能を実行できます。特別な構成またはドライバ、カーネル・パラメータ、管理権限は必要ありません（図 1 を参照）。

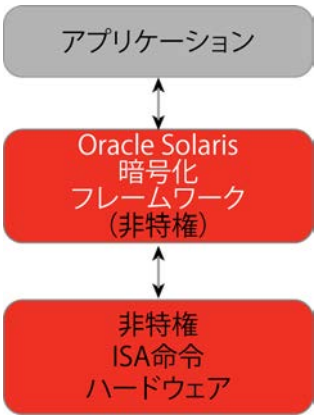

図1：SPARCプロセッサの暗号化操作モデル

実際には、Oracle Solaris の暗号化フレームワークは、アプリケーションと下層に位置付けされるハードウェアとの間のコアの役割として機能します。このフレームワークにより、ユーザーレベルのアプリケーションがハードウェアによる暗号化アクセラレーション機能を自動的に使えるようになります。暗号化フレームワーク・ライブラリは一連の暗号化サービスとアプリケーション・プログラミング・インタフェース（API）を提供しています。これにより、カーネルレベルとユーザーレベルの両方のアプリケーション・ユーザーは、新しいコードをアプリケーションに追加することなく、暗号化操作をハードウェアに透過的に行わせること（委譲）ができます。

SPARCプロセッサを使用したエンド・ツー・エンドのアプリケーション・セキュリティ

アプリケーションの暗号化操作を SPARC プロセッサのオンコアの暗号化機能にオフロードして委譲することで、セキュリティのパフォーマンスを大幅に高めることができます。このオンコアの暗号化アクセラレーション機能には、Oracle HTTP Server、Oracle Application Server、Oracle Database Server を含むアプリケーション・インフラストラクチャ・コンポーネントを使用したさまざまな方法でアクセスできます。

エンド・ツー・エンドのセキュリティ・トポロジ（図 2 を参照）を有効化したアプリケーション・インフラストラクチャの一般的な導入シナリオでは、送信中、処理中、保存中のデータを確実に保護するためにすべてのレベルで暗号化を使用する必要があります。SPARC プロセッサベースの暗号化アクセラレーションは、暗号化メカニズムの使用が重要視されるエンド・ツー・エンドのセキュリティ・トポロジに大きく貢献します。高パフォーマンスなセキュリティの提供は、Oracle Solaris 暗号化フレームワークによって実現され、このフレームワークによりアプリケーションが透過的に暗号化操作を SPARC プロセッサのオンコア暗号化機能にオフロードして委譲できるようになります。さらに、暗号化フレームワークのサポートによって、アプリケーションは Oracle Solaris に備わっている暗号鍵の保存および管理機能も利用できます。

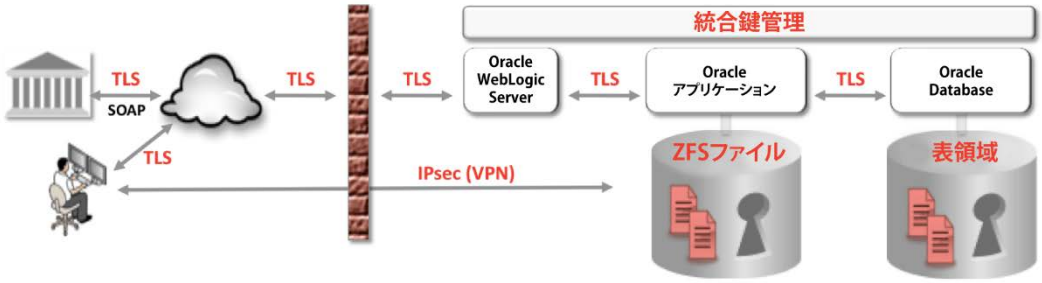

図2：SPARCプロセッサによるエンド・ツー・エンドのセキュリティ・トポロジを使用した多層アプリケーションのデプロイメント

SPARCの暗号化のパフォーマンス

SPARC プロセッサベースの暗号化アクセラレーションでは、オンチップの暗号化アクセラレーション機能により SSL の応答性を改善するだけではなく、暗号化のオーバーヘッドを低減することができます。このテクノロジーの有効性を評価するため、Oracle が提供するテストおよびパフォーマンス監視スイートである Oracle Application Testing Suite を使用して、ワークロードを生成することができます。テストでは、1,000 人のユーザーが、Web サーバーを同時に使用するシミュレーションが行われました。各ユーザーは 1 分間にできるだけ多く、セキュアな SSL 通信を使用して Web アプリケーションに問い合わせを行い、問い合わせの間のキャッシュはクリアされました。ワークロードはピーク時のパフォーマンスの能力ではなく継続的なワークロードを検証するため、10 分間維持されました。この負荷テストはサーバーの限界に挑戦するためではなく、合理的な負荷における暗号化のオーバーヘッドと、ハードウェアによる暗号化アクセラレーションの使用効果を検証することを目的にしました。

図 3 に示すように、まったく保護されていないアプリケーションと、オンチップの暗号化アクセラレーションを使用するエンド・ツー・エンドの完全な SSL 暗号化を使用した場合とで、SSL のオーバーヘッドによる CPU 使用率の差はわずかでした。SPARC 暗号化を有効にすることにより、即効性がありコスト効果のある結果が得られ、高速でセキュアなトランザクションと迅速な応答時間が実現します。また、セキュリティ機器の追加コスト、パフォーマンスの低下、電源使用プロファイルの変更は生じず、複雑なシステム構成も必要ありません。

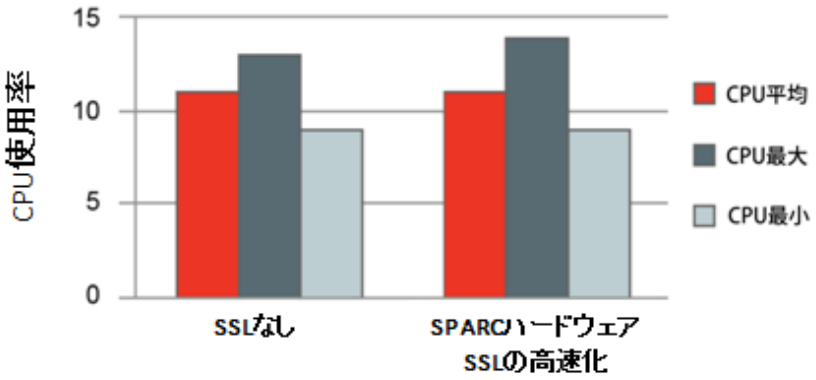

図3：SPARCにより高速化された暗号化でも、SSL暗号化を使用しないシステムとの間でCPUの負荷にほとんど差がなく、CPUリソースをアプリケーションの実行のために引き続き使用できる。

セキュリティの構成

以下のセクションでは、Oracle Solaris 11 および SPARC サーバーで実行されているアプリケーションにセキュリティを構成するための情報を提供します。

Oracle Ucryptoプロバイダを使用したOracle WebLogic Serverセキュリティ・アクセラレーション

デフォルトでは、Oracle WebLogic Server を SPARC サーバーに構築すると、Java Development Kit（JDK）とその Oracle Ucrypto のプロバイダ環境に応じて暗号化操作を処理します。Java Cryptography Extension（JCE）の Oracle Ucrypto プロバイダには、オラクルの SPARC ベースのオンコア暗号化命令でサポートされる暗号化操作をオフロードして委譲するための Oracle Solaris 11 Ucrypto API を活用する専用の機能が統合されています。この機能を有効にするためには、デフォルトの SSL プロバイダとして Java Secure Socket Extension（JSSE）プロバイダを使用するように Oracle WebLogic Server SSL を構成する必要があります。JSSE プロバイダはそのすべての暗号化操作で基礎となる JCE プロバイダのみを使用するため、Oracle WebLogic Server SLL の構成では、Oracle Ucrypto プロバイダによるハードウェア支援型暗号化アクセラレーション機能を自動的に活用できます。

Oracle Ucrypto プロバイダを利用するには、JDL 7 Update 4 以降を Oracle Solaris 11 にインストールし、Java Runtime Environment（JRE）として使用する必要があります。インストールしたら、`$JAVA_HOME/jre/lib/security/`ディレクトリにある Java セキュリティ・プロパティ・ファイル `java.security` に、Oracle Ucrypto プロバイダがデフォルトのプロバイダとして指定されていることを確認します。

```
security.provider.1=com.oracle.security.ucrypto.UcryptoProvider
                        ${java.home}/lib/security/ucrypto-solaris.cfg
```

JDK 7 Update 4 のリリースで、Oracle は PKCS#11 をバイパスし、オラクルの SPARC T4（またはそれより新しい）プロセッサのハードウェア支援型暗号化アクセラレーション機能を自動的に活用する専用のインタフェースを提供する Oracle Ucrypto プロバイダを導入しました。Oracle Solaris 11 および SPARC サーバーに JDK 7（JDK 7 Update 4 以降）をインストールすると、通常はデフォルトで Oracle Ucrypto プロバイダを使用するように JRE が事前構成されます。これにより、Java および Oracle WebLobic Server がホストするアプリケーションと XML Web サービスが、オラクルの SPARC オンコア暗号化命令（図 4 を参照）を使用する暗号化フレームワーク経由で処理される、暗号化の操作を自動的に委譲できるようになります。

Oracle Ucrypto プロバイダだけでなく JDK も PKCS#11 プロバイダ実装（SunPKCS11）を提供します。これにより、Oracle Solaris 暗号化フレームワークにより提供される PKCS#11 ベースの暗号化プロバイダ実装に Java アプリケーションがアクセスできるようになります

図4：オラクルのSPARCサーバーを使用するOracle WebLogic Serverセキュリティ

Oracle Ucryptoプロバイダを使用したSSLの高速化

以下の手順では、オラクルの SPARC プロセッサベースのサーバーのオンチップ暗号化アクセラレーション機能を使用して SSL を高速化するために Oracle WebLogic Server を構成する方法について説明します。

» Oracle WebLogic Server が SSL を Listen するように構成します。構成を行う前に、必要な秘密鍵、サーバー証明書（公開鍵を含む）、認証局（CA）からの信頼できる証明書を取得して、Oracle WebLogic Server 環境内に構成した Java キーストア（ID キーストアと信頼キーストア）に保存します。開発とテストのため、Java の `keytool` ユーティリティを使用して作成した自己署名証明書、秘密鍵、CA からの信頼できる証明書を使用することもできます。ID キーストアと信頼キーストアを構成するには、Oracle WebLogic Server 管理コンソールを使用します。『*Oracle Fusion Middleware—Securing Oracle WebLogic Server 12c Release 1 (12.1.1)*』のドキュメントに指定された SSL 構成のガイドラインに従ってください。

» Oracle WebLogic Server が SSL ポートを Listen し、応答しているかどうかを確認します。これは、管理対象サーバーの Oracle WebLogic Server コンソール・ログから確認できます。

» SSL ベースの暗号化の自動委譲を有効にして、Oracle Ucrypto プロバイダを自動的に活用するために、SSL 構成が JSSE プロバイダに依存していることを確認します。これを行うには、Java ランタイムのオプション `Dweblogic.ssl.JSSEEnabled=true` を、Oracle WebLogic Server の管理対象サーバー・インスタンスの JRE 設定に追加します。

» ハードウェア支援型暗号化機能を実行するには、Oracle WebLogic Server の SSL のプロバイダ構成で、ハードウェアによりサポートされる暗号化アルゴリズムを含む SSL 暗号スイートが定義されていることが重要です。この構成は、脆弱な SSL 暗号スイートを無効化する場合にも役立ちます。これを行うには、Oracle WebLogic Server ドメインの `config.xml` ファイルを次のように編集します。

```
<ssl>
      <enabled>true</enabled>
      <ciphersuite>TLS_RSA_WITH_AES_128_CBC_SHA</ciphersuite>
      <ciphersuite>TLS_RSA_WITH_AES_256_CBC_SHA</ciphersuite>
      ...
</ssl>
```

» Oracle WebLogic Server の管理サーバー・インスタンスを再起動します。これを行うには、Oracle WebLogic Server コンソールを使用します。

Oracle Ucryptoプロバイダを使用したWebサービス・セキュリティの高速化

Web サービス・セキュリティ（WS-Security）は、SOAP メッセージの機密性、整合性を確保し、アクセス制御を確実に実行することで XML Web サービスにメッセージレベルのセキュリティを提供するうえで、重要な役割を果たします。Oracle WebLogic Server は XML Web サービスのメッセージレベルのセキュリティに関連する暗号化操作をサポートするために JCE プロバイダを利用しています。また、暗号化、署名、メッセージ・ダイジェスト操作に関連する暗号化を高速化するための機能については Oracle Ucrypto プロバイダを利用しています。

Oracle WebLogic Server では、WS-SecurityPolicy 標準を使用することを強く推奨しています。Oracle WebLogic Server では、メッセージレベルのセキュリティ要件を指定する、あらかじめ定義された WS-SecurityPolicy ファイルが提供されます。WS-SecurityPolicy 標準には、SOAP メッセージに要求されるメッセージレベルのセキュリティ要件が記載されています。また、要求された操作に対して、ポリシーに定義されたアルゴリズムを使用してどのようにデジタル署名または暗号化を行うかについても記載されています。そして、Web サービス定義言語（WSDL）により、公開されている Web サービスに対してそのセキュリティ・ポリシーを利用可能にします。

以下の手順では、SPARC サーバーのオンチップ暗号化アクセラレーション機能を使用した XML Web サービスのセキュリティ操作および高速化のために Oracle WebLogic Server を構成する方法について説明します。作成される Web サービスは、Oracle WebLogic Server Web サービスや XML Web サービス（JAX-WS）API の Java API を使用して実装される JSON Web 署名（JWS）として構築されるものとします。

必要な WS-Security メカニズムを実行するための WS-Policy および WS-SecurityPolicy の定義を表す事前定義ポリシー・ファイルを指定するため、JWS ファイルを Oracle WebLogic Server 固有の `@Policy` および`@Policies` JWS のアノテーションを含めて更新します。

```
@WebService(name="Simple", targetNamespace="http://oracle.org")
@WLHttpTransport(contextPath="/wssp12/wss10",
serviceUri="UsernameTokenPlainX509SignAndEncrypt")
@Policy(uri="policy:Wssp1.2-2007-Wss1.0-UsernameToken-Plain-X509-
Basic256.xml") public class UsernameTokenPlainX509SignAndEncrypt {
@WebMethod
@Policies({
@Policy(uri="policy:Wssp1.2-2007-SignBody.xml"),
@Policy(uri="policy:Wssp1.2-2007-EncryptBody.xml")}) public String
echo(String s)
{
   return s;
}
```

上の WS-Policy アノテーションは、WS-SecurityPolicy を示しており、サービスがユーザー名トークンを使用してクライアントを認証すること、および要求と応答の両方のメッセージが X.509 証明書を使用して署名され、暗号化されることを指定しています。

ポリシーを追加したら、Web サービスを再コンパイルして、構築します。Web サービスの構成および構築の手順については、『*Oracle Fusion Middleware—Securing Oracle WebLogic Server 12c Release 1 (12.1.1)*』のドキュメントを参照してください。

構築された Web サービスを起動するクライアント・アプリケーションをクライアント側のセキュリティ・ポリシー・ファイルに関連付けることも重要です。通常、セキュリティ・ポリシー・ファイルはサーバー側で起動する Web サービスで構成されるものと同じです。ただし、サーバー側のファイルはクライアントの Java ランタイムに公開されないため、クライアント・アプリケーションが独自のローカル・コピーをロードする必要があります。ユーザーはクライアント・アプリケーションで `weblogic.jws.jaxws.ClientPolicyFeature` クラスを使用して、サービスに定義された有効なポリシーをオーバーライドすることもできます。

セキュリティ・ポリシー・ファイルに指定された暗号化メカニズムによって、Oracle Ucrypto プロバイダまたは SunPKCS11 プロバイダでサポートされる WS-SecurityPolicy アルゴリズム・スイートが特定されるようにします。暗号化アルゴリズムのサポート対象については、『*Java PKCS#11 Reference Guide*』を参照してください。指定されたアルゴリズム・スイートが Basic256 である場合、バルク暗号化の AES-256 アルゴリズムを表し、Sha256 アルゴリズムは SHA-256 ベースのメッセージ・ダイジェストを、Rsa-oaep-mgf1p は鍵ラップのための RSA を表しています。Oracle Ucrypto プロバイダは AES-256 と SHA256withRSA の両方のアルゴリズムをサポートし、オラクルの SPARC ハードウェア支援型暗号化アクセラレーションを活用します。

クライアント側のポリシー・ファイルが確実にロードされるように Java クライアント・アプリケーションを更新して、クライアント・アプリケーションをリビルド/再構築します。クライアントが Web アプリケーションである場合は、クライアントが構築されている Oracle WebLogic Server の管理サーバー・インスタンスを再起動します。

最新バージョンのSSL/TLSの使用

SPARC の暗号化ハードウェア・アクセラレーション機能を利用する方法はいくつかありますが、そのすべてを推奨できるわけではありません。古いドキュメントの多くで、Kernel SSL（KSSL）のプロキシ・アプローチが記載されています。KSSL は基本的に、システム全体に負荷をかけずに SSL ワークロードをインターセプトし、SPARC プロセッサの暗号化機能を使用して暗号化および複合化を実行するための、双方向プロキシとして動作します。KSSL はオラクルの SPARC T2、T3、T4 プロセッサといった初期に、アプリケーションが暗号化アクセラレーションの利点を手軽に利用できるように導入されました。しかし、オラクルの新しいプロセッサ SPARC T5、M6、M7 の機能を活用するためのよりセキュアで高パフォーマンスなオプションが存在するため、KSSL を使用することはもはや推奨されません。

多くの場合で、グレードの低い SSL（バージョン 2.0 および 3.0）を Web 暗号化メカニズムとして使用することも推奨されなくなりました。これは、最近になってハッキングに遭いやすいことが判明したためです（CVE-2014-2566 を参照）。現在、SSL 2.0 および 3.0 を無効化して、代わりに TLS のバージョン 1.1 または 1.2 を使用することが推奨されています。TLS 1.1 および 1.2 を使用するには、JDK 7 Update 1（またはそれ以降）が必要であり、Java Secure Socket Extension（JSSE）を有効にする必要があります。

Oracle WebLogic Server バージョン 10.3.6 および 12c では、JSSE および JDK 7 が認証されており、Java スタートアップ・オプションを使用することで TLS 機能が有効になり、使用可能な最小バージョンの TLS プロトコルを指定することができます。必要となる `JAVA_OPTIONS` パラメータは、Oracle WebLogic Server のスタートアップ・スクリプトが実行される Oracle Solaris シェルの環境変数から構成できます。

```
export JAVA_OPTIONS=-Dweblogic.security.SSL.protocolVersion=TLS1 \
-Dweblogic.security.SSL.minimumProtocolVersion=TLSv1.1
```

TLS を正しく利用するためには、Oracle HTTP Server も正しく構成する必要があります。これを行うには、Oracle HTTP Server の最低バージョンとして 12.1.3 を使用し、Oracle HTTP Server の `ssl.conf` ファイル（`DOMAIN_HOME/config/fmwconfig/components/OHS/componentName` にある）のマスター・コピーに以下の行が含まれていることを確認します。

```
SSLProtocol nzos_Version_1_1 nzos_Version_1_2
```

上のように TLS のバージョンが指定されると、ハードウェア・アクセラレータが自動的に使用されます。Oracle WebLogic Server で TLS のセキュリティを有効にする方法について詳しくは、My Oracle Support のドキュメント番号 1936300.1 を参照してください。

Oracle Application Serverのハードウェア支援型セキュリティの確認

ハードウェア支援型暗号化アクセラレーションが使用できるように構成され、確実にセキュリティ・シナリオに基づいて動作するためには、以下の Oracle Solaris DTrace スクリプトを使用することを推奨します。DTrace は Oracle Solaris の機能です。

```
#!/usr/sbin/dtrace -s
pid$1:libsoftcrypto:yf*:entry,
pid$1:libmd:yf*:entry
{
 @[probefunc] = count();
}
tick-10sec
{
 printa(@);
 clear(@);
 trunc(@,0);
}
tick-100sec
{exit(0);}
```

スクリプトをファイル **cryptoverify.d** として保存してから、スクリプト（コマンドライン引数として J2EE サーバーの Java プロセス ID の Oracle コンテナを含む）を実行します。

```
# dtrace -s cryptoverify.d <Server Process ID>
```

たとえば、TLS_RSA_WITH_AES_128_CBC_SHA 暗号スイートを使用する SSL/TLS 暗号化シナリオでは、AES ジョブの数が正で、増加していると、暗号化アクセラレーションがターゲットの AES バルク暗号化ペイロードで機能していることを示します。次のサンプル出力をご覧ください。

```
# dtrace -s cryptoverify.d 5774

dtrace: script 'crypto-t4.d' matched 51 probes
CPU   ID          FUNCTION:NAME
 65 83719           :tick-10sec
 yf_aes128_ecb_decrypt                  39922
 yf_aes128_load_keys_for_decrypt             39922
```

```
 65 83719           :tick-10sec
 yf_aes128_ecb_decrypt                  44108
 yf_aes128_load_keys_for_decrypt             44108
 65 83719           :tick-10sec
 yf_aes128_ecb_decrypt                  44534
 yf_aes128_load_keys_for_decrypt             44534
..
```

データベース層のセキュリティ

エンタープライズ・アプリケーションは、すべてのレベルで暗号化を使用することにより送信中および保管中のデータの機密性および整合性を確保するための機能を、Oracle Database Security により実現します。Oracle Advanced Security の機能である Transparent Data Encryption（TDE）は、データベースに保存されたデータと送信中のデータを暗号化および復号化して、ネットワーク通信、表領域および列レベルの暗号化、暗号化されたバックアップに関連するすべての操作に対するサポートを提供します。Oracle Database 11*g*（リリース 11.2.0.3）以降、TDE は SPARC プロセッサおよび Oracle Solaris 11 を使用したハードウェア支援型暗号化アクセラレーションのサポートを拡張し、表領域暗号化およびマスター・キーベースの操作に関連する暗号化処理のオフロードをサポートしています（図 5 を参照）。

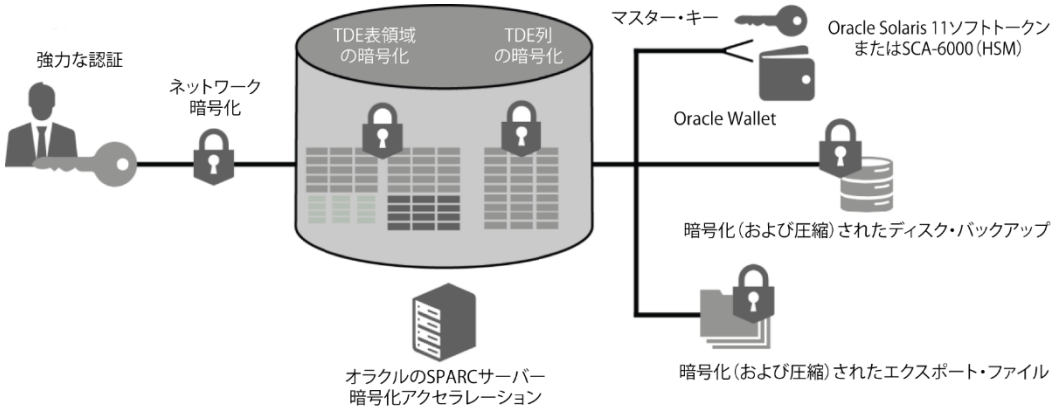

図5：Oracle Database Security：適用されるセキュリティ・シナリオ

Oracle Database Security：適用されるシナリオ

多くの暗号化操作に SPARC プロセッサのハードウェア支援型暗号化アクセラレーションを使用するため、TDE をテスト、検証しました。以下のようなセキュリティ・シナリオを適用しました。

- Oracle Database 12*c* による TDE の有効化
- Oracle Solaris PKCS#11 ソフトトークンを使用したマスター・キー管理
- マスター・キーのバックアップおよびリカバリ
- 表領域および列の暗号化
- データベースのバックアップおよびリストア操作の暗号化および復号化
- ネットワーク・データの暗号化

Oracle Database 12*c* によるTDEの有効化

以下のセクションでは、ソフトウェア・キーストアを使用して Oracle Database 12*c* で TDE を構成する方法について説明します。詳しくは、Oracle Database オンライン・ドキュメント 12*c* リリー

ス 1（12.1）の『*Database Advanced Security ガイド*』を参照してください。

1. wallet（ウォレット）ディレクトリを作成します。

```
% mkdir /export/home/oracle/wallets/<keystore_location>
```

2. ソフトウェア・キーストアの場所を sqlnet.ora ファイルに設定します。デフォルトでは、このファイルは*$ORACLE_HOME*/network/admin/ディレクトリにあります。

```
ENCRYPTION_WALLET_LOCATION=
(SOURCE=
  (METHOD=FILE)
    (METHOD_DATA=
      (DIRECTORY=/export/home/oracle/wallets/<keystore_location>)))
```

3. キーストアを作成します。ADMINISTER KEY MANAGEMENT または SYSKM 権限が付与されているユーザーとして、データベース・インスタンスにログインします。次に、ADMINISTER KEY MANAGEMENT SQL 文を実行して、キーストアを作成します。この例では、パスワードベースのソフトウェア・キーストアを作成しています。

```
$ sqlplus c##sec_admin as syskm
Enter password: password
Connected.

SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'keystore_location'
     IDENTIFIED BY software_keystore_password;

keystore altered.
```

4. 暗号化ウォレットのステータスを調べることで、処理が成功したかどうかを確認できます。この例では、ウォレットが開いているものの、使用できるマスター・キーがまだありません。

```
SQL> select STATUS  from V$ENCRYPTION_WALLET;

STATUS
------------------------------
OPEN_NO_MASTER_KEY

SQL> select WALLET_TYPE from V$ENCRYPTION_WALLET;

WALLET_TYPE
--------------------
PASSWORD
```

5. キーストアを閉じて、再度開きます。

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE;

keystore altered.
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
     IDENTIFIED BY software_keystore_password;

keystore altered.
```

6. ソフトウェアの TDE マスター暗号化キーを設定して、キーストアのバックアップを作成します。

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY software_keystore_password
     WITH BACKUP USING 'backup_key';

keystore altered.
```

7. 暗号化ウォレットのステータスを調べることで、プロセスが成功したかどうかを確認できます。

```
SQL> select STATUS  from V$ENCRYPTION_WALLET;

STATUS
------------------------------
OPEN
SQL exit
```

8. これで、希望の暗号スイートを使用して暗号化された表領域を作成できるようになりました。この例では、ENCRYPTION USING 'AES256'文によって暗号化アルゴリズムと暗号化のキーの長さが指定されています。

```
$ sqlplus "/as sysdba" << !
  CREATE BIGFILE TABLESPACE O_cust_space
       DATAFILE '$DB_DIR/O_cust' SIZE 100000M REUSE
       ENCRYPTION USING 'AES256'
       DEFAULT STORAGE (ENCRYPT);
  !
```

9. 注：データベースをシャットダウンする場合は、起動後にウォレットを開く必要があります。

```
% sqlplus "/as sysdba" << !
       startup
       ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
          " software_keystore_password "
       exit;
  !
```

Oracle Solaris PKCS#11ソフトトークンを使用したマスター・キー管理

Oracle Wallet Manager は、暗号化および復号化に使用するマスター・キーを保護するための一元化されたキーストアです。Oracle Database 11*g* より、PKCS#11 ベースのハードウェア・セキュリティ・モジュール（HSM）キーストアが Oracle ウォレットとして使用できるようになりました。Oracle Solaris PKCS#11 ソフトトークン・ベースの Oracle ウォレットを使用すると、データベースおよびファイル・システムのバックアップ中にマスター・キーが複製およびコピーされなくなります。これは、"Sun Software PKCS#11 Softtoken"と呼ばれる Oracle Solaris PKCS#11 ソフトトークンを使用して行います。

1. Oracle Solaris 暗号化フレームワークの `pktool` ユーティリティを使用して、Sun Software PKCS#11 Softtoken のキーストアを構成します。ソフトトークンのキーストアにアクセスするための PIN/passphrase（パスフレーズ）を設定します。

```
# pktool setpin keystore=pkcs11
 Create new passphrase:
 Re-enter new passphrase:
```

2. Sun Software PKCS#11 Softtoken のトークンを、メタスロットとして有効にします。

```
# cryptoadm enable metaslot
  token="Sun Software PKCS#11 Softtoken"
```

3. PKCS#11 ライブラリを Oracle が提示するディレクトリ構造にコピーします。

   » Oracle Solaris SPARC 環境で、最初に PKCS#11 ライブラリのディレクトリを作成します。

```
# mkdir –p /opt/oracle/extapi/64/hsm/sun/1.0.0/lib
```

   » 次に、Oracle Solaris の `libpkcs11.so` ファイルを PKCS#11 ライブラリのディレクトリにコピーします。

```
# cp /usr/lib/sparcv9/libpkcs11.so /opt/oracle/extapi/64/hsm/sun/1.0.0/lib
```

4. ユーザーおよびグループ（`oracle:install`）が PKCS#11 ライブラリのディレクトリに正しく設定され、このディレクトリに読み書きの権限が割り当てられていることを確認します。

```
# chown –R oracle:oinstall <directory>
```

   » Oracle Solaris の環境では、`SOFTTOKEN_DIR` 環境変数を設定することもできます（オラクルのデフォルト・ユーザー・シェル内）。

```
# export SOFTTOKEN_DIR=/export/home/oracle/.sunw
```

Transparent Data Encryptionのためのマスター・キー管理の保護

Oracle Solaris PKCS#11 ソフトトークンを使用するように TDE を構成するには、最初にマスター・キーのソースを HSM として特定する HSM ベースの Oracle ウォレットを設定する必要があります。

1. `$TNS_ADMIN/sqlnet.ora` ファイルを編集して、`ENCRYPTION_WALLET_LOCATION` パラメータを追加します。

```
ENCRYPTION_WALLET_LOCATION =
            (SOURCE=(METHOD=HSM) (METHOD_DATA=
   (DIRECTORY = /export/home/oracle/11g/network/admin/)))
```

2. SQLPlus に `system` または `sysdba` としてログインし、HSM ウォレットを作成します。

```
$ sqlplus "/ as sysdba"

SQL> alter system set encryption key
    identified by "HSM Username:Password";
```

注：`Username:Password` は、Oracle Solaris PKCS#11 ソフトトークン・キーストアを使用したマスター・キー管理操作の実行をサポートするための、TDE 専用ユーザー・アカウントの資格証明です。

データベースが以前に Oracle ソフトウェア・ウォレットを使用していた場合、ユーザーはマスター・キーを構成済みの Oracle Solaris PKCS#11 ソフトトークンに移行できます。この移行プロセスによって、自動的に既存のデータ・オブジェクトが復号化され、Oracle Solaris PKCS#11 ソフトトークンで新しく作成されたマスター暗号化キーを使用して再度暗号化されます。

TDE が"ソフトウェア・ウォレット"を使用して以前に構成されている場合は、`MIGRATE USING "software_wallet_password"`句を前の `sqlplus` コマンドに追加して、ソフトウェア・ウォレットから HSM にマスター・キーを移行する必要があります。*`software_wallet_password`* はソフトウェア・ウォレットの元のパスワードです。

```
SQL> alter system set encryption key identified by
   "HSM Password" migrate using "software_wallet_password";
```

表領域および列の暗号化

Oracle Database 11.2.0.3 では、TDE のためにオラクルの SPARC T4 以上のハードウェア支援型暗号化アクセラレーションがサポートされています。インストール・プロセスでホスト・マシンのプロセッサが自動的に識別されるため、SPARC サーバーでハードウェアにより高速化される暗号化を使用するための設定は技術的に必要ありません。

Oracle Database は、一度デプロイされると、表領域暗号化、ネットワーク暗号化、暗号化されたバックアップ、ファイルのリストア、暗号化されたダンプ・ファイルに関係する、暗号化および復号化操作の両方でハードウェア支援型暗号化を使用します。

TDE が Oracle Solaris PKCS#11 ソフトトークンに保存されているマスター暗号化キーを使用していることをテストして確認するには、以下の推奨される SQL の例を使用します。この例は Oracle Solaris PKCS#11 ソフトトークンに存在するマスター・キーに依存する TDE の操作を示しています。

1. データベースが起動し実行されていることを確認します。sqlplus に system としてログインし、接続します。

```
$ sqlplus "/ as sysdba"
SQL> startup;
SQL> connect as system/password;
```

2. Oracle Solaris PKCS#11 ソフトトークン・ベースの HSM ウォレットのオープンとクローズを確認します。ユーザーが TDE にアクセスするために作成したユーザー名とパスワードを使用していることを確認します。

```
SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "tdepassword";
System altered.

SQL> select WRL_TYPE, STATUS from v$encryption_wallet;
WRL_TYPE          STATUS
--------------------   --------------
HSM            OPEN

SQL> ALTER SYSTEM SET WALLET CLOSE IDENTIFIED BY   "tdepassword";
System altered.
```

3. HSM ウォレットを使用して暗号化された表領域を作成します。

   » HSM ウォレットが開いていることを確認します。

```
SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "tdepassword";
```

   » 暗号化された表領域を作成します。

```
SQL> CREATE TABLESPACE SCASecuredTablespace
 2 DATAFILE '/export/home/oracle/11g/oradata/scasecuretbs1.dbf'
 3 SIZE 50M
 4 ENCRYPTION
 5 DEFAULT STORAGE(ENCRYPT);
```

4. 暗号化された表領域に表を作成します。この表に保存されているすべてのデータ・オブジェクトが自動的に暗号化されます。

```
SQL> CREATE TABLE PERSON
 2 (first_name VARCHAR2(11),
 3 last_name VARCHAR2(10),
 4 social_security_number NUMBER(9),
 5 address VARCHAR2(25),
 6 city VARCHAR2(25),
 7 state VARCHAR2(2)) TABLESPACE SecuredTablespace;
```

データベースのバックアップとリストアの暗号化および復号化

Oracle Database Data Pump ユーティリティを使用する、暗号化されたエクスポート/インポート・ファイルでは、ダンプ・ファイルを暗号化および復号化するために HSM に存在するマスター・キーを使用できます。

» デフォルトでは、（暗号化を指定することなく）すべてのエクスポート・ダンプ・ファイルが暗号化されないファイルとして保存されます。以下の例では、エクスポート・ファイルが暗号化なしでダンプされます。

```
$ expdp system/oracle@sid tables=employee
```

» マスター・キーを使用してエクスポート・ダンプ・ファイルの暗号化を強制するには、最初に HSM ウォレットが開いたままになっていることを確認します。ユーザーは `ENCRYPTION_MODE=TRANSPARENT` を使用して、HSM ウォレットに保存されたマスター・キーを使用するダンプ・ファイルの暗号化を有効にする必要があります。オプション `ENCRYPTION_MODE=DUAL` を指定すると、ウォレットに保存されたマスター・キーと、暗号化のためのパスワードも追加で使用してダンプ・セットが暗号化されます。

```
$ expdp system/oracle@sid tables=employee encryption=all
  encryption_password=pwd4encrypt encryption_algorithm=AES256
  encryption_mode=DUAL
```

» マスター・キーを使用して暗号化されたダンプ・ファイルをインポートするには、HSM ウォレットが開いたままになっていることを確認して、暗号化に使用するパスワードを指定するためのオプションを設定します。次に例を示します。

```
$ impdp system/oracle@Ssid encryption_password=pwd4encrypt
tables=employee table_exists_action=replace
```

Oracle Recovery Manager（Oracle RMAN）を使用したデータベースのバックアップとリストアでは、HSM に存在するマスター・キーを使用できます。

» データベースのバックアップ、リストア、リカバリのコマンドを実行する前に HSM ウォレットが開いていることを確認し、さらにデータベースが `archivelog` モードになっていることも確認します。

```
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount;
ORACLE instance started.
Database mounted.
```

```
SQL> alter database archivelog;
Database altered.
SQL> alter database open;
Database altered.
SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "oracle:password";
System altered.
SQL> exit
```

» backup コマンドを実行する前に、rman ユーティリティを使用して暗号化を有効にします。

```
RMAN> connect target sysoper/oracle;
connected to target database: sid (DBID=1555558107)

RMAN> set encryption on;

RMAN> backup as compressed backupset database;
```

ネットワーク・データの暗号化

ネットワーク・データの暗号化により、Oracle Database サーバーと Oracle Database クライアントの間のネットワーク上で送信中のデータの暗号化が有効になります。Oracle Database は、暗号化操作を実行するために SPARC プロセッサのハードウェアによる暗号化アクセラレーションを利用する、Oracle Solaris PKCS#11 キーストアの使用をサポートしています。このサポートを有効にするには、Oracle ウォレットのウォレット・タイプを PKCS#11 に構成する必要があります。これにより、ファイル・システムベースのウォレットではなく、Oracle Solaris PKCS#11 ソフトトークンを使用できるようになります。Oracle Wallet Manager アプリケーションにより、通信の保護、およびクライアント/サーバー認証のための SSL および TLS プロトコルに必要な、秘密鍵、証明書、信用できる証明書を含む PKI 証明書の資格証明の保存と管理をサポートするための、PKCS#11 ベースのウォレットを構成できるようになります。

"Sun Software PKCS#11 Softtoken"キーストアを構成していることが前提となります。前のセクション「Oracle Solaris PKCS#11 ソフトトークンを使用したマスター・キー管理」の手順 1 から 3 を実行します。キーストアが構成されたら、Oracle Wallet Manager ユーティリティを使用して PKCS#11 ベースの Oracle ウォレットを設定します。SSL/TLS を構成するには、『*Oracle Database Advanced Security 管理者ガイド*』のセクション「Secure Sockets Layer 認証の構成」で説明されている手順に従います。サーバーがオラクルの SPARC T4 および SPARC T3 プロセッサでサポートされるアルゴリズムを含む SSL 暗号スイートを選択することが重要です。たとえば、SSL_RSA_WITH_AES_128_CBC_SHA がサポートされていますが、これは、バルク暗号化に RSA（ハンドシェイクと認証）および AES-128 を使用するためです。オラクルのデフォルトの SSL 暗号スイート SSL_RSA_WITH_RC4_128_SHA では、バルク暗号化に RC4 を使用する必要がありますが、これはサポートされていません。

ZFS暗号化によるデータの保護

デフォルトでは、ZFS は Oracle Solaris 11 の暗号化サービス API を使用します。これにより、SPARC プロセッサで使用できる AES アルゴリズムのハードウェア・アクセラレーションが自動的に利用できます。暗号化ポリシーは、データ・セット（ファイル・システムまたは ZFS ボリューム）が作成される際に、データ・セット・レベルで設定されます。それぞれの ZFS ディスク・ブロック（最小サイズは 512 バイト、最大は 128 キロバイト）は、CCM または GCM モードのどちらかで AES アルゴリズムを使用して暗号化されます。ラッピング鍵は、ファイル・システムを作成する Oracle Solaris 管理者が提供する必要があります。この鍵はファイル・システムをオフラインにしなくても、いつでも変更できます。データ暗号化キーはデータ・セットの作成時にランダムに生成されます。ラッピング鍵を作成する最も簡単な方法は、Oracle Solaris の既存の `pktool` コマンドを使用する方法です。

```
$ pktool genkey keystore=file keytype=aes
    keylen=128 outkey=/export/home/user/mykey
```

ZFS 暗号化サポートは以下のように簡単に使用できます。

```
# zfs create -o encryption=on -o
    keysource=raw,file:///export/home/user/mykey myfilesystem/cryptofs
```

または、ラッピング鍵のセキュアな保存および取得を行うには、Oracle Solaris PKCS#11 ソフトトークンを、ラッピング鍵を保存するためのキーストアとして使用することを推奨します。キーストアとして Oracle Solaris PKCS#11 ソフトトークンを使用することで、ラッピング鍵が暗号化されて保存され、キーストアが PIN で保護されるようになります。Oracle Solaris PKCS#11 ソフトトークンのキーストアにラッピング鍵を作成して保存し、この鍵を使用して暗号化された ZFS データ・セットを作成する手順は以下のとおりです。

```
# pktool genkey keystore=pkcs11 keytype=aes keylen=128 label=mykey
Enter PIN for Sun Software PKCS#11 softtoken:

# zfs create -o encryption=on
       -o keysource=raw,pkcs11:object=mykey myfilesystem/cryptofs
Enter PKCS#11 token PIN for 'myfilesystem/cryptofs'
```

上の例では、AES キーはユーザーのデフォルトのソフトトークン・キーストアに作成されます。このキーストアに鍵を作成して保存している鍵を使用するには認証が必要になるため、ユーザーにはキーストアの PIN の入力が求められます（実際にはパスフレーズですが、PKCS#11 の用語では慣例により用語 *PIN* が使用されます）。`keysource` プロパティで使用される PKCS#11 URI の構文により、PIN ファイルのパスを指定できます。この方法を使用することで、実際のラッピング鍵が PKCS#11 キーストアで暗号化され、保護されます。

ZFS 暗号化については、『*Oracle Solaris ZFS 管理ガイド*』を参照してください。

まとめ

このホワイト・ペーパーでは、オラクルの SPARC プロセッサの Oracle Solaris 11 のセキュリティ機

能、およびハードウェア支援型暗号化アクセラレーション機能を使用してアプリケーションを保護するためのさまざまな方法について説明しました。また、高パフォーマンスでエンド・ツー・エンドのセキュリティ・ソリューションを提供するための、中心となるメカニズム、構成、導入方法や、Oracle Solaris 暗号化フレームワークおよび Java Cryptography Extension ベースの技術を使用する際の役割および関連性について紹介しました。多層ビジネス・アプリケーションにエンド・ツー・エンドのセキュリティを提供し、規制遵守の要件を満たすため、送信中のデータや保管中の暗号化データに対して SSL/TLS の暗号化を採用することが重要になっています。

エンド・ツー・エンドのセキュリティのデプロイメントに SPARC プロセッサのハードウェア支援型暗号化アクセラレーションを使用することで、即効性があり、コスト効果のある結果が得られ、高速でセキュアなトランザクションと迅速な応答時間が実現します。セキュリティ機器の追加コスト、電源使用プロファイルの変更は生じず、複雑なシステム構成も必要ありません。派生するパフォーマンス特性によって、高速化していない暗号化ワークロードがサーバーにもたらすと考えられる膨大な負荷も明らかになります。

まとめると、オラクルの SPARC プロセッサベースのサーバーは、高パフォーマンスのエンタープライズ・セキュリティにアプリケーションのための一貫したスケーラビリティを提供しながらも、スペース、電力消費、コストを削減しています。

参考資料

詳しくは、表 2 のリファレンスを参照してください。

表2：詳細に関するリファレンス

Web サイト	
Oracle Optimized Solution	oracle.com/optimizedsolutions
Oracle SuperCluster	oracle.com/supercluster
オラクルの SPARC サーバー	oracle.com/goto/tseries
Oracle Solaris	oracle.com/solaris
Oracle Solaris Cluster	oracle.com/jp/solaris/cluster/overview/index.html
Oracle ZFS Storage Appliance	oracle.com/jp/storage/nas/overview/index.html
オラクルの Exadata Storage Expansion Rack	oracle.com/us/products/database/exadata/expansion-storage-rack/overview/index.html
Oracle E-Business Suite	oracle.com/us/products/applications/ebusiness/overview/index.html
Oracle Optimized Solution for Secure Backup and Recovery	oracle.com/solutions/optimized-solutions/backup-and-recovery
Oracle Optimized Solution for Secure Disaster Recovery	oracle.com/solutions/optimized-solutions/disaster-recovery
Oracle Technology Network	oracle.com/technetwork/index.html
Oracle Consulting	oracle.com/jp/products/consulting/overview/index.html
Oracle SuperCluster ホワイト・ペーパー	
『Oracle SuperCluster の技術概要』	oracle.com/technetwork/jp/server-storage/sun-sparc-enterprise/documentation/o13-045-sc-t5-8-arch-1982476-ja.pdf
『Oracle SuperCluster T5-8：すぐに実行可能な最適化済みのサーバー、ストレージ、ネットワーク、ソフトウェア』	oracle.com/jp/products/servers-storage/servers/sparc/supercluster/supercluster-t5-8/ssc-t5-8-wp-1964621-ja.pdf
Oracle Exadata Database Machine ホワイト・ペーパー	
『Oracle Exadata 上の Oracle E-Business Suite Oracle Maximum Availability Architecture』	oracle.com/technetwork/jp/database/availability/maa-ebs-exadata-197298-ja.pdf
Oracle Solaris ホワイト・ペーパー	
『Oracle Solaris および Oracle SPARC T4 サーバー—エンタープライズ・クラウド導入のための設計』	oracle.com/jp/products/servers-storage/solaris/solaris-and-sparc-t4-497273-ja.pdf
Oracle Database ホワイト・ペーパー	
『Oracle Database 11gRelease 2 による高可用性の実現』	oracle.com/technetwork/jp/database/features/availability/twp-databaseha-11gr2-1-132255-ja.pdf 11gr2-1-132255-ja.pdf
バックアップ、リカバリ、高可用性、ディザスタ・リカバリのホワイト・ペーパー	
『Oracle Optimized Solution for Secure Backup and Recovery: Oracle SuperCluster Backup and Recovery』	oracle.com/technetwork/server-storage/hardware-solutions/oos-ocs-backup-recovery-1973464.pdf
『Oracle Data Guard 11*g* Oracle Database 向けのデータ保護と可用性』	oracle.com/technetwork/jp/content/twp-dataguard-11gr2-134591-ja.pdf
Oracle Optimized Solution Support ドキュメント	
Oracle Support ドキュメント 1558827.1、『Oracle Optimized Solution for Oracle E-Business Suite』	support.oracle.com/epmos/faces/DocumentDisplay?id=1558827.1

セキュリティ・リソース	
Oracle E-Business Suite System Administrator's Guide – Security (Release 12.1)	docs.oracle.com/cd/E18727_01/doc.121/e12843/T156458T156461.htm
Oracle E-Business Suite Security Guide (Release 12.2)	docs.oracle.com/cd/E26401_01/doc.122/e22952/T156458T156461.htm
Oracle Database セキュリティ・ガイド11g リリース2（11.2）	docs.oracle.com/cd/E16338_01/network.112/b56285/title.htm
My Oracle Support ドキュメント 946372.1 ー Secure Configuration of Oracle E-Business Suite Profiles	support.oracle.com/epmos/faces/DocumentDisplay?id=946372.1
Oracle Database Security Guide 12c Release 1 (12.1)	docs.oracle.com/database/121/DBSEG/
Oracle Solaris 11 Security Guidelines	docs.oracle.com/cd/E26502_01/pdf/E29014.pdf
Architecture Matters:Increasing Oracle Database Security Without Application Performance Loss	community.oracle.com/docs/DOC-999059

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

海外からのお問い合わせ窓口
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Integrated Cloud Applications & Platform Services

完全に暗号化されたデータセンター2016 年 6 月

Oracle is committed to developing practices and products that help protect the environment