

# エグゼクティブサマリー

## 背景

PowerBuilder アプリケーションは、金融、医療、物流、産業、公共部門など、多くの分野における重要なプロセスの中心的存在です。これらのアプリケーションの一部は、当初設計された時よりも高いセキュリティとコンプライアンス要件を満たす必要に迫られています。

## 3つの主要な目的

- 1 アプリケーションを書き換えることなく、PowerBuilder の最新バージョンと専用ツールを使用してセキュリティを強化する。
- 2 主要なリスクを特定し、優先順位を付けて修正することで、脆弱性の露出を減らし、監査管理基準を満たします。
- 3 検証可能な証拠を提供することで、基準（ISO 27001、NIST、PCI DSS、HIPAA、SOX、GDPR）への準拠を実証する。

## 9つのテーマ別章

1. 実行ファイルと配布済みコンポーネントの保護
2. 保存時および転送中のデータの暗号化
3. 最新かつ安全なネットワークプロトコルの採用
4. 外部サービスおよび API との通信の保護
5. 組み込みブラウザの攻撃対象領域の削減
6. ビルドおよび配布チェーンのセキュリティ確保
7. 認証とアクセス制御の強化
8. 最新のアーキテクチャ（PowerServer）への移行
9. 最新標準への準拠達成

## 3つの補完ツール

### PowerBuilder

ネイティブセキュリティ機能：コード署名、暗号化、最新プロトコル (TLS 1.3、HTTP/2、OAuth 2.0、REST)、トークンベース認証など

### Visual Expert

PowerBuilder 向け静的コード分析 (SAST): 脆弱性、ハードコードされた秘密情報、および安全でない実行の検出。

### Visual Guard

アイデンティティおよびアクセス管理 (IAM)：MFA 認証、権限、監査、レポート、職務分離 (SoD)、コンプライアンス。

## 得られるもの

- PowerBuilder アプリケーションのセキュリティ強化のためのアクションプラン
- 技術的対策と基準要件を結びつけるコンプライアンスロードマップ

# 2026 年版 PowerBuilder アプリケーションセキュリティガイド

---

## セキュリティ強化のためのツールと手法 および現代の基準への準拠

### 目次

1. はじめに .....	3
2. 実行ファイルと配布済みコンポーネントの保護 .....	5
3. 保存時および転送中のデータの暗号化 .....	8
4. 現代的で安全なネットワークプロトコルの採用 .....	10
5. 外部サービスおよび API との安全な通信 .....	12
6. 組み込みブラウザに関連する攻撃対象領域の削減 .....	14
7. ビルドおよび配布チェーンのセキュリティ確保 .....	16
8. 認証とアクセス制御の強化 .....	19
9. オプション : PowerServer によるモダンアーキテクチャへの移行 .....	21
10. アプリケーションを現代の基準に準拠させる .....	23
PowerBuilder アプリケーションのセキュリティ確保のためのチェックリスト .....	25
よくある質問 .....	29
参考文献 - 本ガイドで引用した情報源 .....	31

## 1. はじめに

### 本ガイドの対象者

- **PowerBuilder アプリケーションを担当するチーム**：開発者、アーキテクト、プロジェクトマネージャー、運用マネージャー。アプリケーションのセキュリティ強化とライフサイクル（開発/テスト/本番環境）の保護方法を説明します。
- **セキュリティおよびコンプライアンスチーム**：アプリケーションセキュリティマネージャー、CISO、GRC チーム、内部または外部監査人。PowerBuilder アプリケーションのセキュリティレベルを評価し、監査やコンプライアンスレビューに備えるのに役立ちます。

### 本ガイドの目的

PowerBuilder アプリケーションは、重要なプロセス（財務、医療、物流、産業、公共部門）の中核を担っています。設計時よりも高いセキュリティおよびコンプライアンス要件（クライアントワークステーションの保護、強固な認証、暗号化、アクセス制御、トレーサビリティ、監査レポートなど）を満たす必要があります。

こうした状況において、組織は次の3つの目標を同時に達成する必要があります：

1. **アプリケーションを書き換えることなく、PowerBuilder の最新バージョンと専用ツールを活用してセキュリティを強化する。**
2. **主要なリスクを特定し、優先順位を付けて修正し、脆弱性の露出を減らして監査にパスさせます。**
3. **検証可能なエビデンスを提供することで、内部または外部の基準・要件（ISO、NIST、PCI、HIPAA、SOX、GDPR）への準拠を実証する。**



#### Strengthen Security

Secure existing applications without rewriting, leveraging modern tools and recent versions.



#### Reduce Risks

Identify and prioritize vulnerabilities to reduce exposure and pass audits.



#### Prove Compliance

Demonstrate adherence to standards and requirements with verifiable evidence.

## 得られるもの

本ガイドを読むことで、以下のものを得られます：

- 実行ファイルの整合性からアクセス制御、監査に至るまで、PowerBuilder アプリケーションのセキュリティ強化に向けた**行動計画**。
- これらの技術的措置を規格や規制の要求事項と結びつける**コンプライアンスロードマップ**。

本ガイドは**モジュール式**で、選択的に読むことができます。

各章は独立しており、コンテキスト、脅威、技術的目標、各ツールの機能、主要標準への準拠、証拠など、読みやすい構成になっています。

セキュリティまたはコンプライアンスプロジェクトの基盤として活用できます。

## 2. 実行ファイルと配布済みコンポーネントの保護

### 背景とリスク

PowerBuilder アプリケーションは、多くの場合、内部配布メカニズム（ネットワーク共有、パッケージ、汎用配布ツール）を介して配布され、配信された実行ファイルに対する暗黙の信頼に基づいています。

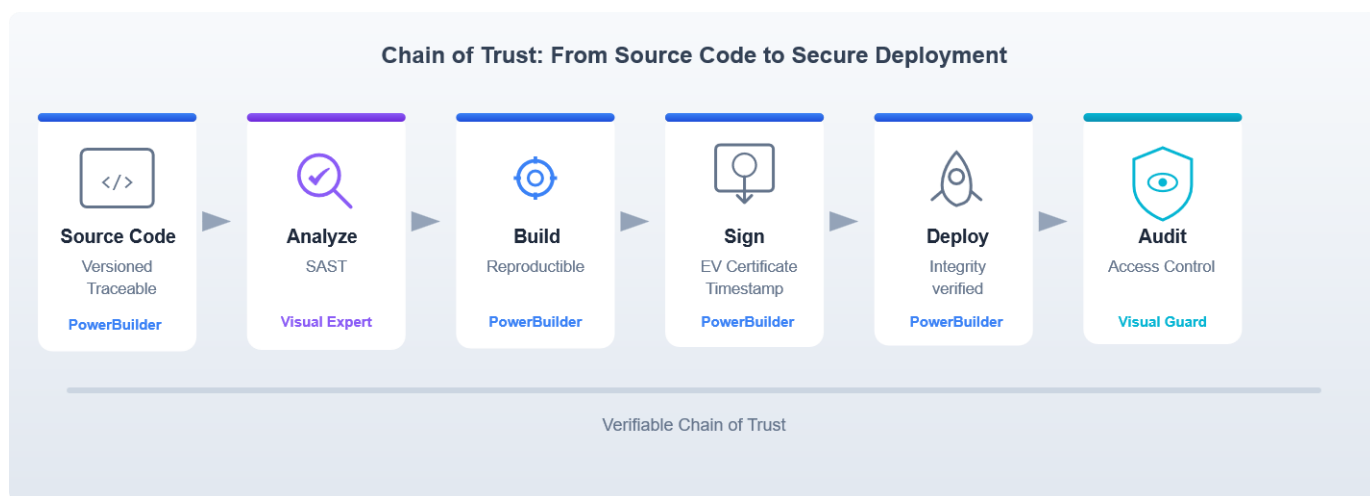
ワークステーションの強化に伴い、署名されていない、または追跡不可能な実行ファイルはしばしばブロックされますが、制御が脆弱な場合、侵害されたバイナリが受け入れられる可能性があります。

このリスクは、導入の容易さを超えたものです。[2023年の3CXインシデント](#)で明らかになったように、ソフトウェアチェーンが侵害されるシナリオに相当します。このインシデントでは、正規のアプリケーションが感染し、大規模に配布され、検出されるまでに複数の侵入の道が開かれてしまいました。

さらに、攻撃者は署名付き要素への信頼を悪用しようとします。Sophos（世界的なサイバーソリューションセキュリティベンダー）は[133の悪意のあるWindowsドライバ](#)を文書化しており、そのうち100はWHCPプログラムを通じて署名されていました。

解決策は検証可能な信頼の連鎖を確立することです。公開前に分析し、成果物に体系的に署名し、配布されたバージョンを追跡し、公開権限を制限します。

コード署名に加えて、多くの組織では現在、実行ファイルレベルのハードニングと、クライアントワークステーション上でのDLL読み込みに対するより厳格な制御を求めています。これにより、機密性の高いエンドポイントにおけるメモリ悪用技法やDLLハイジャックのリスクを低減できます。



## 目的

- アプリケーションのバージョンに関連付けられた、再現可能でバージョン管理されたコンパイルプロセスを実装する。
- 組織が承認した証明書で、実行ファイル、ライブラリ、および配布パッケージに体系的に署名する。
- 配布前に成果物の整合性を検証する（例：ハッシュによる）。
- 配布されたバージョン、そのハッシュ、リリース日、責任者を追跡する。
- 配布とセキュリティに関連する公開権限および管理権限を制限し監査する。
- 対象のビルドモードで可能な場合は、アプリケーション実行ファイルでサポートされる実行ファイルセキュリティフラグを有効化する。
- 承認されたパスのみに DLL の読み込みを制限し、該当する場合は DLL 署名検証を有効化する。

## 機能とツール

PowerServer および PowerClient プロジェクトを含め、 <a href="#">コード署名を</a> コンパイルおよび配布プロセスに統合する。	PB
<a href="#">拡張検証証明書 (EV 証明書) を使用してアプリケーションに署名し</a> 、ユーザーの信頼性と最新のセキュリティポリシーとの互換性を向上させます。	PB
プロジェクトタイプでサポートされる場合、アプリケーション実行ファイルに対して DEP、ASLR、CFG、SafeSEH (32 ビットのみ) などの高度な実行ファイルセキュリティフラグを有効化します。	PB
クライアントワークステーションでの DLL ハイジャック リスクを低減するため、DLL の読み込みパスを承認済みパスに制限し、DLL 署名検証を有効化します。	PB
PowerBuilder 専用セキュリティルール (SAST) による PowerBuilder コードの分析、 <a href="#">脆弱性および不適切な手法の検出により</a> 、安全なデリバリーを実現します。	VE
コード内の <a href="#">ハードコードされた識別子/パスワード、暗号化キー、IP アドレス</a> を検出し、デコンパイルやデプロイメントスクリプトのコピーによる漏洩リスクを低減します。	VE

## コンプライアンス

この章では、ISO 27001 などの規格や SOX 準拠の環境でよく見られる、ソフトウェアの完全性およびリリースサイクル管理に関する期待事項について取り上げます。([セキュリティ基準とコンプライアンス](#))

## 証拠

- 署名手順（証明書、タイムスタンプ、責任範囲）およびパイプライン内での実行証明。
- 有効化された実行ファイルセキュリティフラグと、承認された DLL 読み込みセキュリティ設定を示すビルドエビデンス。

- ビルド後に、PESecurity、dumpbin、WinDbg などのツールで生成された検証結果。
- 署名済みバージョンに関連する Visual Expert レポート（脆弱性、深刻度、修正内容）。
- 機密操作（管理、権限変更）に関する Visual Guard 監査履歴（監査活動）。
- PB 2025 R2 に関する注記：高度な実行ファイルセキュリティフラグは現在ベータ機能であり、アプリケーション実行ファイルのみに適用されます。ランタイムファイルへの対応は、今後のメジャーリリースで予定されています。

### 3. 保存時および転送中のデータの暗号化

#### 背景とリスク

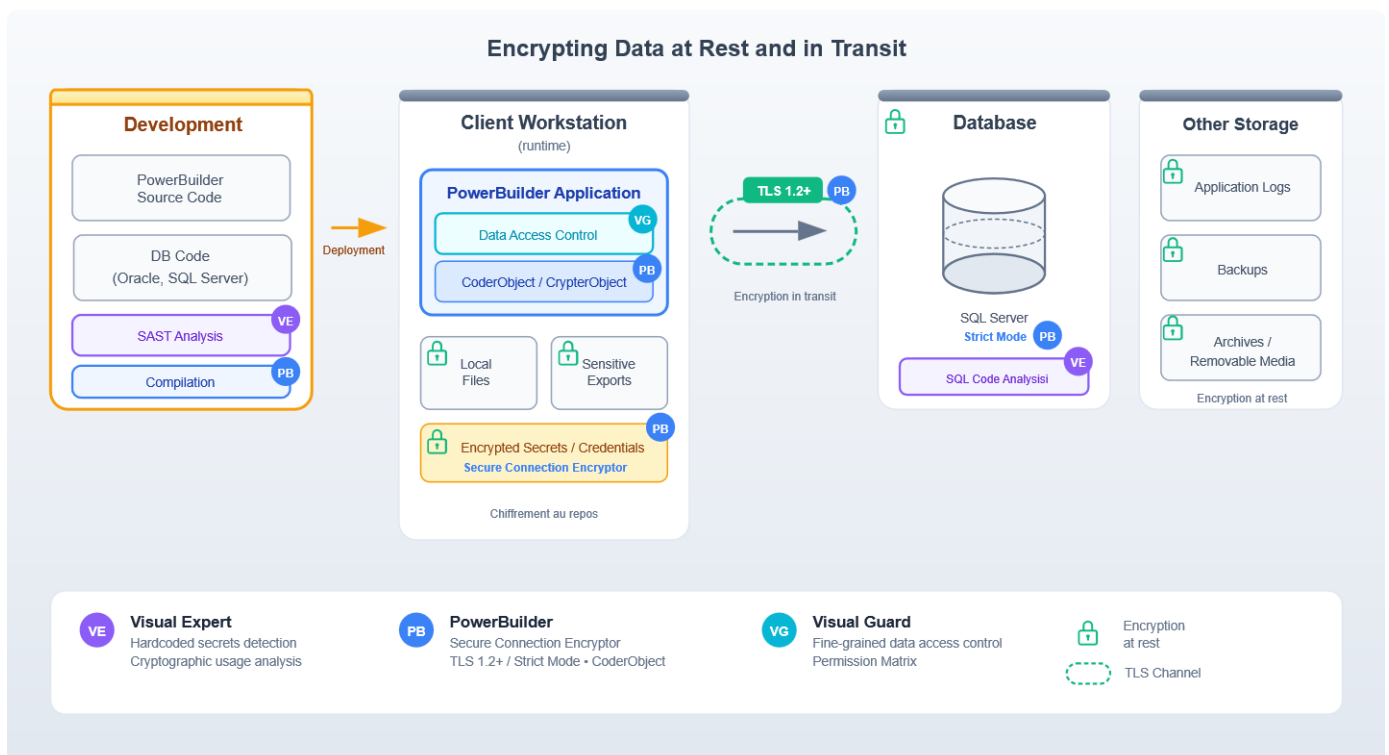
PowerBuilder アプリケーションは、機密データ（業務、個人情報、医療、金融など）を頻繁に処理します。このデータはクライアント、データベース、中間コンポーネント間で転送されます。また、エクスポート、一時ファイル、アプリケーションログ、バックアップにも存在します。

システムおよびデータの可用性喪失は一般的な問題であり、ランサムウェアは確認済み侵害事例の 44% に存在します（[Verizon 2025 DBIR - エグゼクティブサマリー](#)）。

しかし、こうした攻撃は、事前の情報流出（「二重の恐喝」）を伴う場合があります、インシデントをデータ漏洩やコンプライアンスの危機へと発展させる可能性があります（[CISA - ランサムウェアガイド](#)）。

医療分野では、[MD アンダーソン事件](#)が、暗号化されていないストレージ（ワークステーション、ノートパソコン、リムーバブルメディア）のために、規制当局による大きな制裁措置につながりました。

解決策は、保存時の暗号化、転送中の暗号化、厳密なキー管理、関連する権限の制限などにより、そのコンテキスト外ではデータを読み取れないようにすることです。



## 目的

- アプリケーションが扱う機密データとその保存場所（データベース、ファイル、エクスポート、ログ、バックアップ）を特定する。
- 秘密情報や機密性の高いパラメータを暗号化し、コード、スクリプト、設定ファイルに平文で表示されないようにする。
- クライアントワークステーション、アプリケーションサーバー、データベース間の通信を、会社が推奨する暗号化モードを適用して暗号化する。
- クライアントワークステーションに重要なデータが含まれる可能性がある場合、ローカルストレージと機密エクスポートを保護する。
- 使用されているアルゴリズム、鍵サイズ、暗号化パラメータが準拠し維持されていることを確認する。

## 機能とツール

トランザクションオブジェクトの接続プロパティを暗号化し、 <a href="#">Secure Connection Encryptor</a> を使用して暗号化された接続文字列を生成し、スクリプトやファイルに平文の認証情報が含まれないようにします。	PB
<a href="#">SQL Server ドライバー</a> およびデータベース側で、TLS/‘Strict’ 暗号化を有効にして SQL Server 接続を暗号化し、証明書を検証します。	PB
<a href="#">CoderObject</a> および <a href="#">CrypterObject</a> オブジェクトを使用してデータをエンコードおよび暗号化します。	PB
PowerBuilder コードをスキャンして、 <a href="#">DES/3DES</a> 、廃止されたハッシュ関数 ( <a href="#">SHA-1/MD5 など</a> )、 <a href="#">暗号化モード</a> 、 <a href="#">ハードコードされたキー</a> 、 <a href="#">または堅牢ではないキーサイズの使用</a> を検出します。	VE
権限/ロールモデルを介して機密データや機能へのアクセスを厳密に制御し、リスクを軽減します。 ( <a href="#">Permissions</a> )	VG
機密データおよび重要操作へのアクセスを記録する。詳細な監査を可能にする（誰が、何を、いつ、どこから行ったか）。 ( <a href="#">トレーサビリティと監査</a> )	VG

## コンプライアンス

本章は、ISO 27001、NIST SP 800-53、PCI DSS、および状況に応じて HIPAA および GDPR における機密性およびデータ保護の要件に貢献します。 ([セキュリティ基準とコンプライアンス](#))

## エビデンス

- 接続暗号化設定（DB プロファイル、アプリケーション設定）および秘密鍵ローテーション手順。
- ハードコードされた秘密情報の存在がなく、暗号関連リスクが低減されていることを示す Visual Expert レポート。（PowerBuilder コードの[セキュリティ脆弱性検出](#)）
- 機密領域の権限マトリックスおよび Visual Guard 監査履歴のエクスポート。（[権限マトリックス](#)） ([監査活動](#))

## 4. 現代的で安全なネットワークプロトコルの採用

### 背景とリスク

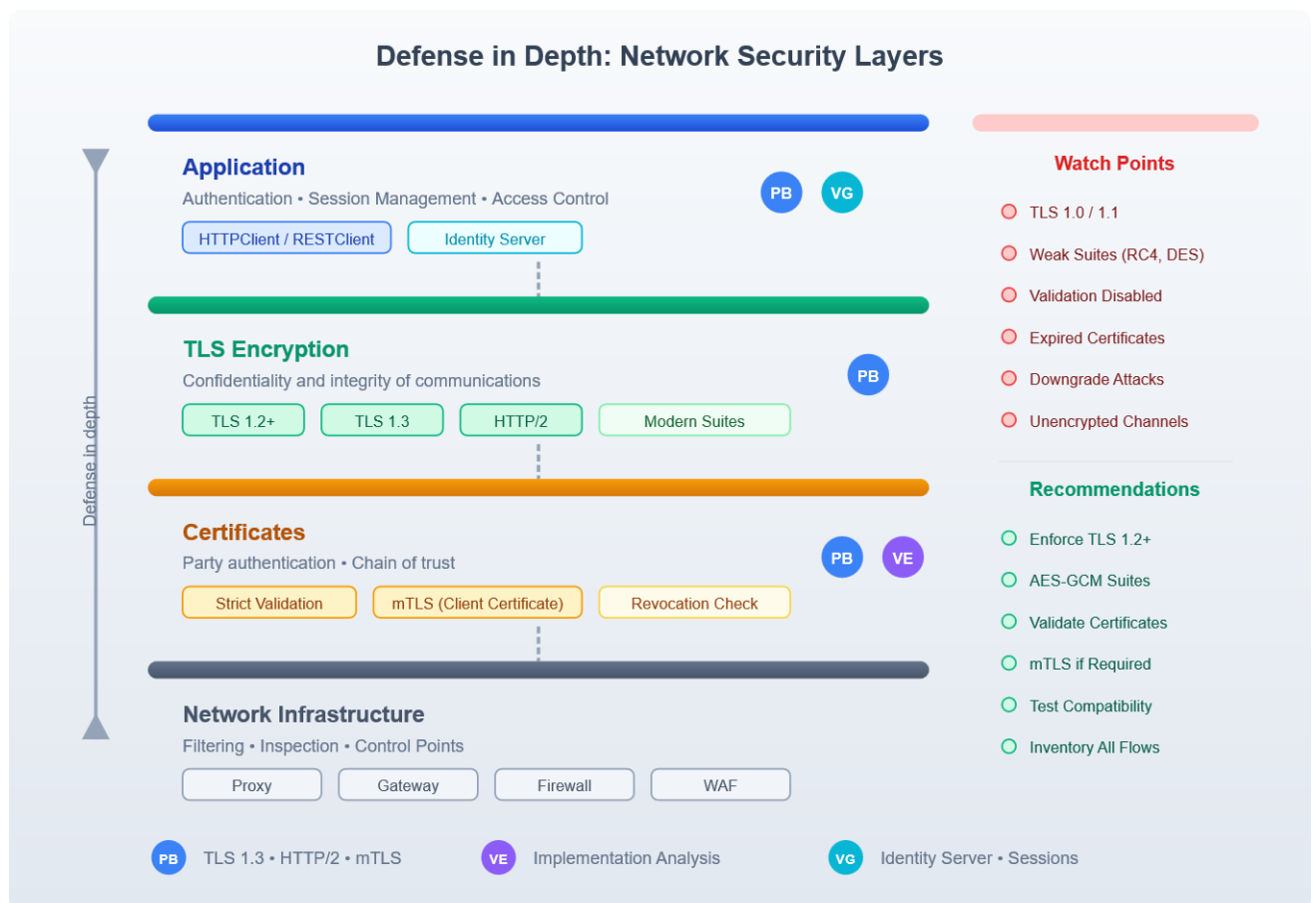
PowerBuilder アプリケーションのインフラストラクチャは、長年にわたって進化してきました（ドライバ、ミドルウェア、プロキシ、ゲートウェイ）。

古いプロトコルの使用、脆弱な暗号スイート、または暗号化されていないチャネルの使用により、トラフィック（認証、データ、トークン）の傍受や改ざん、あるいはプロキシ、サーバー、データベースが TLS 1.2 以上または厳格モードを要求する場合のアプリケーションのブロックといったリスクが生じます。これによりセキュリティインシデントが本番環境の問題に発展します。

[NIST SP 800-52r2](#) 公開文書では、最低限 TLS 1.2 を推奨し、ダウングレード攻撃や暗号化エラーを減らすための設定要件を定めています。

コンプライアンスの観点では、[PCI DSS](#) は公開ネットワーク経由で送信される決済データを保護するため、強固な暗号技術の使用を義務付けています。

解決策は、トラフィックフローのインベントリ作成、TLS 設定の強化、ネットワークコンポーネントの標準化です。



## 目的

- アプリケーションが使用するネットワークフローと、実際にネゴシエートされたプロトコル (TLS バージョン、暗号スイート、証明書) をインベントリ化する。
- 廃止されたプロトコルを排除し、組織のセキュリティポリシーに沿った TLS バージョンと構成を適用する。
- 環境が要求する場合、特に内部 API やパートナーに対して、相互認証 (クライアント証明書) を実装する。
- バイパスやサイレントな劣化を防ぐため、証明書検証とエラー処理を標準化する。
- プロキシ、ゲートウェイ、ネットワーク保護との互換性をテストし、セキュリティ制御による本番環境のブロックを防止する。

## 機能とツール

<a href="#">HTTPClient</a> および <a href="#">RESTClient</a> オブジェクトで <a href="#">TLS 1.3</a> および <a href="#">HTTP/2</a> をサポートします。	PB
内部 API やパートナーサービスなど、 <a href="#">相互 TLS 認証</a> (クライアント証明書) をサポートします。	PB
コード通信ポイント (HTTP クライアント、 <a href="#">Web サービス呼び出し</a> 、証明書管理) を識別します。	VE
最新のセキュリティプロトコルをサポートしていない使用例 (例: <a href="#">SOAP/INET サービスへの呼び出し</a> や <a href="#">非セキュアな認証 API の使用</a> ) を検出します。ネットワークリファクタリングを優先します。	VE
認証、アクセス制御、ログ記録には <a href="#">標準化された Web サービス</a> を専用サーバー経由で使用し、制御を集中するとともにクライアント側のセキュリティロジックを制限します。	VG

## コンプライアンス

本章は、ISO 27001 および NIST の通信セキュリティに関連する統制に貢献し、金融および医療分野における監査要件を頻繁に満たします。 ([セキュリティ基準とコンプライアンス](#))

## エビデンス

- TLS 設定 (バージョン、スイート、証明書、該当する場合は mTLS) および接続性テスト結果。 ([双方向 TLS 認証をサポート - 新機能](#))
- アプリケーションエンドポイントの一覧と使用プロトコルの正当性説明。
- 公開フロー向け集中認証 (Identity Server) の統合証明。 ([Identity Server API の使用方法?](#))

## 5. 外部サービスおよび API との安全な通信

### 背景とリスク

システムの近代化に伴い、多くの PowerBuilder アプリケーションが API を利用したり、SaaS プラットフォームと通信したり、メッセージングや内部サービスを介した交換を自動化しています。

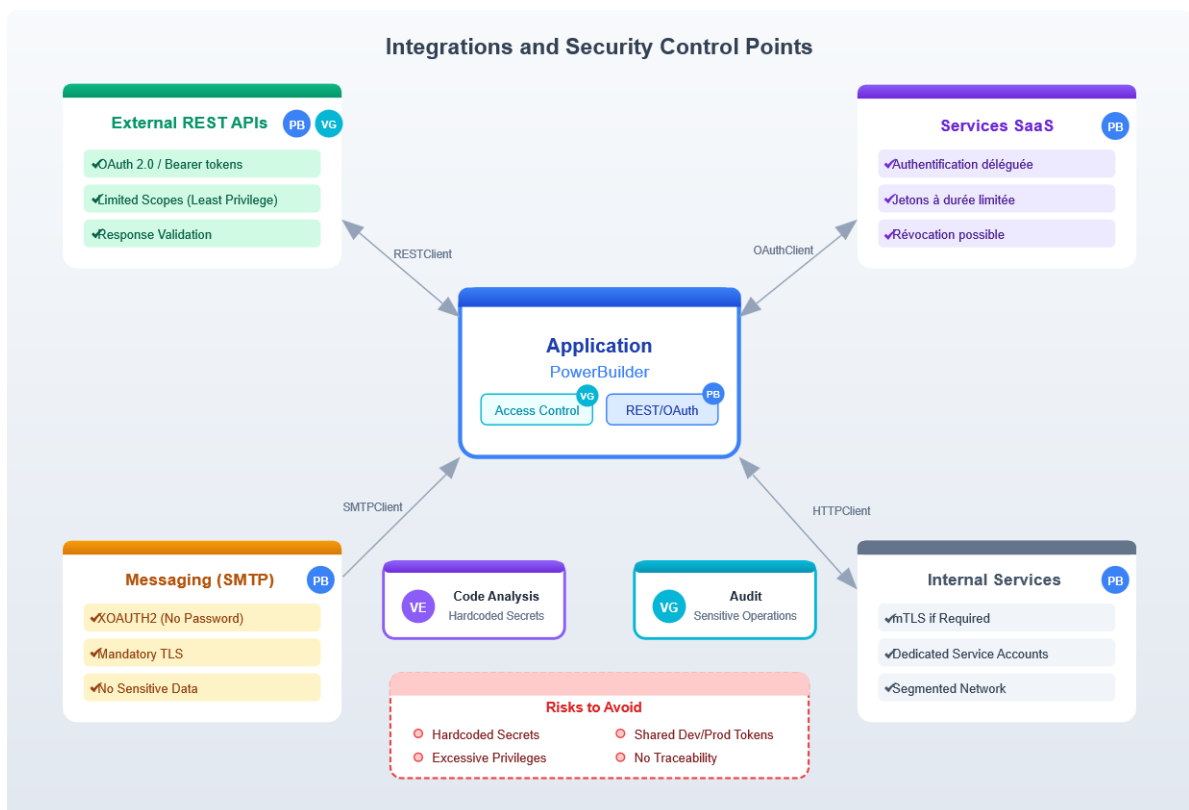
この開放性はプロジェクトを加速させる一方で、侵入経路も生み出します：不完全なアクセス制御、過度に許可的なトークン、不十分な入力検証、環境間で共有されるシークレットなどです。ローカルに保存されたトークンや環境間（開発/テスト/本番）で共有されたトークンを使用して API を呼び出すアプリケーションは、ワークステーションの侵害や設定漏洩が発生した場合に攻撃の入り口となる可能性があります。

OWASP は、API 側の最も重大な弱点は依然として オブジェクトレベルのアクセス制御の欠如であると指摘しています。

ビジネスプロセスに関しては、[FBIIC3 2024 年報告書](#)によると、ビジネスメールの侵害に関連する損失は 27 億 7000 万ドル（電信送金詐欺、請求書ハイジャックなど）と推定されています。

約 1,000 万人の顧客に影響を与えた Optus インシデント ([2022 年](#)) は、顧客データがインターネット経由でアクセス可能になった場合の危険性の大きさを示しています。

対応策としては、認証の標準化、権限と露出面の制限、交換の制御が挙げられます。



## 目的

- 統合箇所（API、内部サービス、SMTP）と交換されるデータを特定する。
- システム間での標準認証（例：OAuth）を実装する。
- トークンおよび技術アカウントに関連する権限を厳しく制限する。
- ハードコードされたシークレットを削除し、その保存、ローテーション、失効を整理する。
- メッセージやログに機密情報を露出させない。
- 影響の大きい操作（エクスポート、送信、重要なアクション）を管理および追跡する。
- 不正利用を検知するメカニズムを導入する。

## 機能とツール

RESTClient を使用した REST ベースの統合を優先する。SOAP が必要な場合は、強力な認証を備えた TLS 1.2 以上の HTTPClient を使用し、レガシー SOAP クライアントの実装は避けてください。	PB
一般的な API 統合の慣行に合わせて、OAuth 2.0 認証プロトコルを使用します。 ( <a href="#">強化された RESTClient オブジェクト</a> - 新機能)	PB
ユーザー名とパスワードを送信する代わりに、トークンベースの認証（OAuth 2.0 ベアータークン）を使用し、HTTPClient および OAuthClient でトークンを管理してください。( <a href="#">HTTPClient および OAuthClient の機能強化</a> - 新機能)	PB
SMTP をネイティブでサポートし、最新の認証方法（XOAUTH2 を含む）を使用します。脆弱な可能性のある社内実装を減らします。( <a href="#">ネイティブメールサポート (SMTP クライアント)</a> - 新機能)	PB
<a href="#">コードをスキャン</a> して、 <a href="#">SQL</a> 、 <a href="#">OS コマンド</a> 、 <a href="#">パスへのコードインジェクション</a> 、ハードコードされたシークレット、不十分なエラー処理、交換データの制御不足による攻撃の可能性を特定します。	VE
機密性の高い呼び出し（エクスポート、送信、ビジネスアクション）をトリガーできるユーザーを制限する	VG
<a href="#">監査</a> および調査のためにこれらの操作を <a href="#">追跡</a> します。重大または異常なイベントが発生した場合に <a href="#">リアルタイムで監視</a> し、通知をトリガーします。	VG

## コンプライアンス

統合管理機能は、ISO 27001、NIST、PCI DSS、および適用範囲に応じて HIPAA のセキュリティとトレーサビリティ要件に貢献します。( [セキュリティ基準とコンプライアンス](#) )

## エビデンス

- 統合機能（API、SMTP、内部サービス）と使用認証方式（OAuth、証明書など）のインベントリ。( [強化された RESTClient オブジェクト](#) - 新機能) ( [ネイティブメールサポート \(SMTP クライアント\)](#) - 新機能)
- ハードコードされたシークレットが存在せず、主要なリスクが修正されていることを証明する Visual Expert レポート（PowerBuilder コードの[セキュリティ脆弱性を検出](#)）

- 機密操作の実行およびアラートを表示する Visual Guard ログ ([トレーサビリティと監査](#))

## 6. 組み込みブラウザに関連する攻撃対象領域の削減

### 背景とリスク

PowerBuilder アプリケーションに組み込まれたブラウザは、HTML コンテンツの表示、ポータル統合、または Web ページ経由の認証を容易にするために使用できます。

これらの統合はレガシーコンポーネントに基づく場合があります。マイクロソフトは 2022 年 6 月に IE11 のサポートを終了し、[現在は Edge の IE モード使用を推奨](#)しています。古い Web コントロールを維持することは、脆弱性のリスクや望ましくないコンテンツの実行リスクを生じさせます。

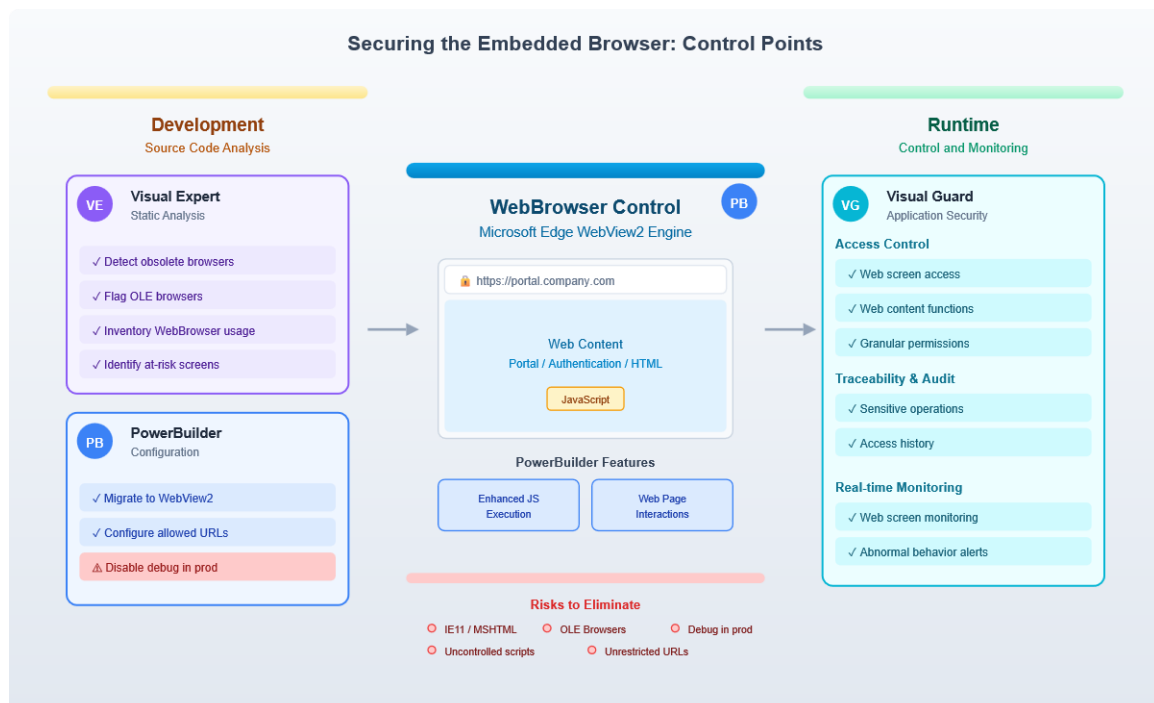
[Microsoft のドキュメント](#)では、WebBrowser コントロールが「完全信頼」モードで動作し、外部サーバー由来の潜在的に危険なスクリプトやコンテンツの実行を防止しない点も指摘されています。

MSHTML エンジンの脆弱性は、CISA アラートを発動させた [CVE-2021-40444 など](#)、標的型攻撃で既に悪用されています。

この状況を踏まえ、アクセスする Web コンテンツを制限し、コンポーネントを隔離し、最新のサポート対象エンジンへの移行が推奨されます。

### 目的

- ブラウザを使用して、画面、期待されるコンテンツ、および許可された URL をリストアップします。
- 最新のブラウザに移行し、サポートされていないコンポーネントを削除します。
- 特に外部サーバーからのコンテンツについては、ブラウジング、URL オープン、スクリプト実行を厳しく制限します。
- アプリケーションとウェブページ間の通信 (JavaScript インタラクション) を監視し、望ましくない実行やアクセスを防止します。
- 本番環境でのリモートデバッグを無効化します。



## 機能とツール

Microsoft Edge WebView2 に基づく最新化された <a href="#">WebBrowser</a> コントロールを使用し、関連する機能強化を適用します。	PB
JavaScript 実行とページ操作のための <a href="#">WebBrowser</a> の改善を活用します。	PB
診断目的で <a href="#">WebBrowser のリモートデバッグ</a> を有効化します。この機能は厳重に管理し、本番環境では無効化する必要があります。	PB
<a href="#">OLE ブラウザや廃止されたコンポーネントの使用</a> を検出します。これらは脆弱性やセキュリティ上の非互換性にさらされやすいためです。	VE
WebBrowser の使用状況（URL の開く操作、スクリプトの実行、コンテンツの操作）を調査・分析し、リスクのある画面を特定して修正の優先順位を決定します。	VE
Web コンテンツを統合する画面や機能への <a href="#">アクセスを制限</a> します。	VG
ブラウザに関連する機密性の高い操作を <a href="#">追跡し監査</a> します。	VG
ウェブコンテンツを表示する画面や機能へのアクセスを <a href="#">リアルタイムで監視し</a> 、異常な動作が発生した場合に <a href="#">チームに通知</a> します。	VG

## コンプライアンス

この章は、ISO 27001 および NIST で一般的に見られる、攻撃対象領域の削減およびアクセス制御に関連する要件をサポートします。[\(セキュリティ基準およびコンプライアンス\)](#)

## エビデンス

- WebBrowser を含む画面の一覧（強化対策適用済み：許可された URL、本番環境でのリモートデバッグ無効化）。[\(WebBrowser でのリモートデバッグのサポート - 新機能\)](#)
- 特定された Web エントリーポイントと実施された修正に関する Visual Expert レポート [\(PowerBuilder コードのセキュリティ脆弱性検出\)](#)
- これらの画面/機能におけるアクセス制御と監査の Visual Guard による証拠 [\(トレーサビリティと監査\)](#)

## 7. ビルドおよび配布チェーンのセキュリティ確保

### 背景とリスク

アプリケーションのセキュリティはソースコードに限定されません。アーティファクトを生成・公開するチェーン（ビルドサーバー、スクリプト、依存関係、サービスアカウント、署名キー、リポジトリ、本番プロセス）にも依存します。

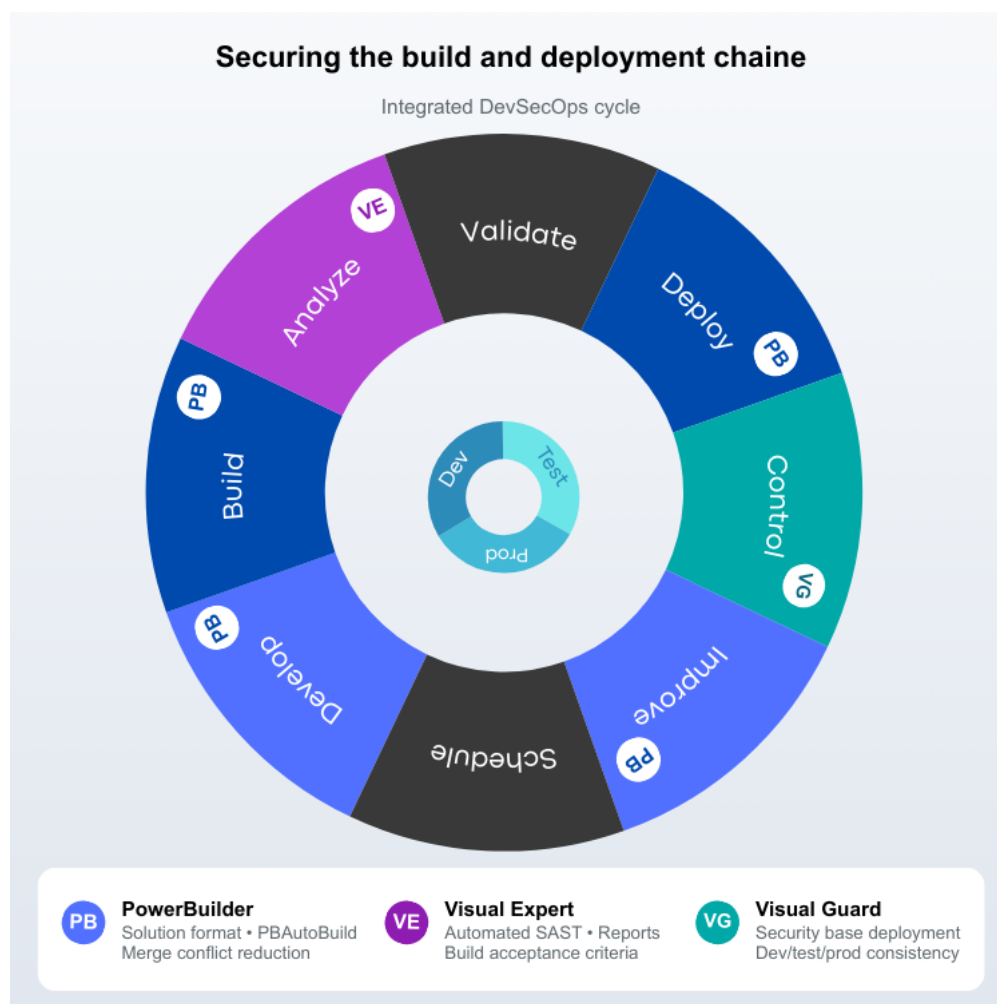
1 つのリンクが侵害されると、攻撃者は更新プログラムに悪意のあるコードを注入し、すべてのユーザーに影響を与える可能性があります（共有ビルドサーバー、配布サービスアカウント、証明書/署名キーの保存場所など）。

[SolarWinds インシデント](#)は、広く導入されたソフトウェアプログラムの製造・流通チェーンが侵害されたことで、このシナリオが大規模に発生する可能性を示しました。

より一般的には、[2025 年の DBIR では](#)、分析対象の侵害事例の 30%にサードパーティが関与していたことが明らかになっており、前年の約 15%から増加しており、依存関係と共有アクセスの管理の重要性を再認識させています。

推奨される対応策は、DevOps プロセスに沿った完全性管理（署名、ハッシュ、SBOM）、技術アカウント権限の削減、シークレットガバナンス（ローテーション、漏洩検知、失効）を組み合わせたものです。

セキュアなビルド パイプラインでは、アーティファクトに署名するだけでなく、リリース前に実行ファイルのハードニング設定と DLL 読み込み制限を適用し、検証する必要があります。これを行わなければ、セキュア コーディングとコード署名を実施していても、ローカル実行経路が不要に露出したままになる可能性があります。



## 目的

- ビルド、テスト、セキュリティ分析、署名、配布を統合した標準パイプラインを定義し、明確な合格基準を設定します。
- ツールのバージョン、依存関係、コンパイルパラメータを管理することで、ビルドを再現可能にします。
- パイプラインで使用されるサービスアカウント、シークレット、署名キーを保護し、その権限を制限します。
- 監査証拠を提供し調査を容易にするため、各バージョンに関連する成果物とレポートを保持します。
- ビルドパイプラインと配布先全体で、実行ファイルのハードニング設定を標準化する。
- リリース検証の一環として、実行ファイルのセキュリティフラグと DLL 読み込み制限を検証する。

## 機能とツール

<a href="#">ソリューション形式</a> に切り替えてコードをテキストに変換し、バージョン管理ツールや自動化ツールとの統合を容易にします。	PB
<a href="#">PBAutoBuild</a> を使用してビルドスクリプトを簡素化し、パイプラインを標準化します。	PB
PB 2025 で <a href="#">マージ競合を減らし</a> 、レビューとトレーサビリティを容易にします。	PB
プロジェクトレベルで実行ファイルのハードニング オプションと DLL 読み込み制限を構成し、ビルドおよびリリース時に一貫して適用されるようにします。	PB
トリガー、ビルド受け入れ基準、ダッシュボードなど、すべての <a href="#">ビルドで静的脆弱性分析 (SAST) を工業化</a> します。	VE
<a href="#">例外が無視されず</a> 、本番環境でコンソールログが使用されていないことを検証します。	VE
環境間（開発/テスト/本番）における <a href="#">アクセス制御設定（権限、ロールなど）のデプロイを自動化</a> し、不一致やエラーを削減します。	VG

## コンプライアンス

この章は、ISO 27001 および該当する場合は内部統制要件（SOX）に関連するガバナンスとトレーサビリティの期待に応えるものです。（[セキュリティ基準とコンプライアンス](#)）

## エビデンス

- パイプライン（ビルド、テスト、VE スキャン、署名、デプロイ）の定義と成果物の保存期間。（[PBAutoBuild は PBC と統合 - 新機能](#)）
- 実行ファイルのハードニングフラグ、DLL 読み込み制限、および署名検証設定を確認するリリース検証エビデンス。

- 環境間で一貫したセキュリティ設定が適用されていることを示す、ビルドおよび配布記録。
- 提供バージョンごとの **Visual Expert** レポート履歴と是正措置の追跡  
([PowerBuilder コードのセキュリティ脆弱性検出](#))
- バージョンおよび環境に関連付けられた **Visual Guard** エクスポート (監査/権限)。( [監査活動](#) ) ( [権限マトリックス](#) )

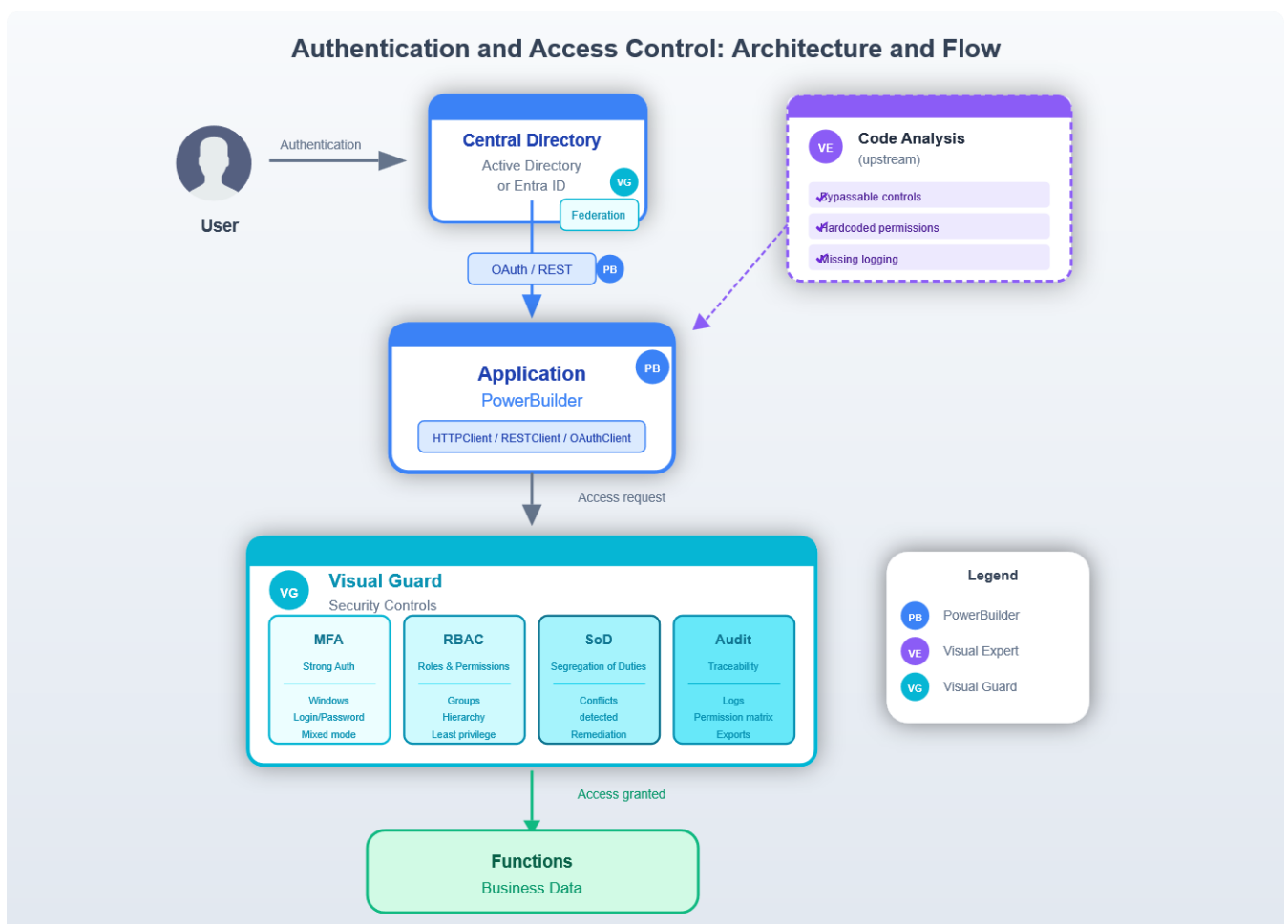
## 8. 認証とアクセス制御の強化

### 背景とリスク

多くの PowerBuilder アプリケーションでは、ユーザー認証にアプリケーションデータベースに保存されたログイン/パスワードアカウントを依然として使用しています。

- パスワード認証は限定的なセキュリティしか提供せず（脆弱なパスワード、共有アカウント）、リスクをもたらします。なぜなら、アイデンティティが主要な攻撃ベクトルとなっているからです：[2025年 DBIR](#) で調査された侵害事例の 22%は、侵害された認証情報の使用から始まりました。
- このアプローチでは、各アプリケーション固有のユーザーアカウント（サイロ化されたアカウント）を管理する必要が生じ、管理コストが増加します。
- また、管理者が管理・監視すべきアカウントが増えるため（非アクティブアカウント、過剰な権限、職務分離の欠如など）、異常発生のリスクも高まります。

適切な戦略は、Active Directory や Entra ID などの単一の ID ストアにユーザーアカウントを集中管理し、機密性の高いアプリケーションには多要素認証（MFA）による追加の保護層を追加することです。



## 目的

- ユーザーID を一元管理します。ローカルアプリケーションアカウントの使用を削減します。例えば、Active Directory または MS EntraID に保存されている Windows アカウントに置き換えます。
- 機密性の高いアプリケーションや特権アカウントには強力な認証（MFA）を導入します。
- 最小権限の原則を適用し、一貫した役割と権限を通じて権限をモデル化します。
- 重要な操作に対する職務分離を実施し、承認の競合を処理します。
- 権限の定期的な見直しを実施し、機密性の高いログインや操作の監査証跡を維持します。

## 機能とツール

アプリケーションが ID サービスと連携する際、PB ネットワークオブジェクト ( <a href="#">HTTPClient/OAuth/RESTClient</a> ) を使用して、最新の認証およびトークンメカニズムを統合します。	PB
コード内の認証/認可の実装を識別し、 <a href="#">一般的な脆弱性</a> （回避可能な制御、ハードコードされた権限、ロギングの欠如） <a href="#">を検出します</a> 。	VE
強力な認証（MFA）を実装し、ケースに応じて、Windows 認証、ログイン/パスワード、または同じアプリ内での混合モードを選択します。	VG
複数のディレクトリに分散した ID を管理（フェデレーション）し、組織構造を反映した階層的なグループにユーザーを編成することで、権限付与の委任と工業化を実現します。	VG
<a href="#">職務分離（SoD）</a> を通じて権限衝突を特定し修正します。	VG
<a href="#">権限マトリックス</a> を生成し、監査対応可能なレポートを作成します。	VG

## コンプライアンス

この章では、ISO 27001、NIST SP 800-53、および PCI DSS の IAM およびアクセスガバナンス要件に直接取り組み、特に金融、医療、および上場企業セクターに関連しています。[\(セキュリティ基準とコンプライアンス\)](#)

## エビデンス

- 役割/グループ/ユーザー別の権限マトリックスと定期的な見直しのエビデンス。[\(権限マトリックス\)](#)
- 文書化された職務分離（SoD）ルール、検出された衝突のリスト、および是正計画。[\(職務分離（SoD）\)](#)
- 機密性の高い接続および操作の監査ログ（保持期間とエクスポート機能付き）。[\(監査活動\)](#)

## 9. オプション : PowerServer によるモダンアーキテクチャへの移行

### 背景とリスク

一部の PowerBuilder アプリケーションは現在、リモートアクセス（VPN、アプリケーションゲートウェイ、仮想化）が可能ですが、データベースに直接接続し、クライアントワークステーションに保存されたパラメータに依存する「厚いクライアント」も維持されています。

このモデルでは、ネットワーク境界やリモートアクセスポイントに配置された機器が重要な侵入経路となるため、一定のリスクが生じます。また、ワークステーションごとに秘密情報や接続パラメータを管理することがより困難になります。

[DBIR 2025](#) によると、脆弱性の悪用は初期アクセスのおよそ 20% を占めています。特にリモートアクセス機器に影響を与える脆弱性は、30 日以内に修正される割合が 54% に留まることから、極めて敏感な問題であることが強調されています。

解決策は、ワークステーションへの依存度と露出を低減するため、多階層アーキテクチャへの移行です。アプリケーションサーバーが .NET API (PowerServer) を公開し、適切な位置（認証、ログ、ネットワークセグメンテーション）に集中管理された ID 管理と制御機能を配置します。

### 目的

- クライアントワークステーションからのデータベースへの直接アクセスを減らすため、API を公開するサーバー層でアクセスを一元化します。
- サーバーサイドのセキュリティ制御（認証、認可、ログ記録）を集中化し、クライアントアプリケーションやローカル設定への依存を制限します。
- API 認証とトークン管理を標準化し、組織の慣行に整合させます。
- 最も機密性の高いモジュールを優先し、段階的な進化を計画します。

## 機能とツール

PowerServer を使用して、サーバー側の n 層アーキテクチャに移行し、クライアントワークステーションからの直接データベースアクセスを削減します。	PB
<a href="#">Web API</a> に OAuth 2.0 や JWT などの認証メカニズムを実装し、不正アクセスを防止します。	PB
サーバーコンポーネントを <a href="#">.Net のサポート対象バージョン</a> に更新します	PB
アーキテクチャ変更の前、最中、後に、アプリケーションのマッピング、依存関係の特定、既存コードのセキュリティ確保を行います。	VE
標準ツールを使用して、ユーザー、 <a href="#">API アクセス権限</a> を管理し、監査レポートを生成します。	VG
<a href="#">複数のアプリケーションのセキュリティを一元化し</a> 、ID と権限の包括的な可視化を実現するとともに、制御、ログ記録、監査を標準化します。	VG

## コンプライアンス

この章は、ISO 27001 および NIST で頻繁に強調される、監査可能で標準化されたアーキテクチャに整合させるアプローチの一部です。([セキュリティ基準とコンプライアンス](#))

## エビデンス

- アーキテクチャ図（ネットワークゾーン、Web API、リバースプロキシ、アイデンティティ）およびフロー文書化。（[PowerServer Web API](#)）
- 機密性の高い操作に対するアクセス制御と監査（Visual Guard）の証拠。（[トレーサビリティと監査](#)）
- 本番環境展開前の脆弱性修正を証明する Visual Expert レポート。（[PowerBuilder コードのセキュリティ欠陥検出](#)）

## 10. アプリケーションを現代の標準に準拠させる

### 背景とリスク

規制対象分野（金融、医療、上場企業）では、セキュリティ強化だけが目的ではありません。

これらの組織は、アプリケーションが現代の基準に準拠していることを実証しなければなりません。また、機密機能にアクセスできるのは誰か、どのような権限が付与されているか、どのようなアクションが実行されているか、そして重要なデータをどのような対策で保護しているかなど、具体的な検証可能なチェックを実施する必要があります。

この要件は、透明性に関する義務によってさらに強化されています。例えば米国では、SEC は上場企業に対して、厳格に規制された手順に従って特定の「[重要な](#)サイバーセキュリティインシデントを報告するよう義務付けています。

さらに、[データ侵害による世界平均の損失額は2024年に488万ドルに達し](#)、コンプライアンスのギャップが直接的な財務リスクへと変化しています。

こうした状況において、組織は今や統制を体系化し、証拠（レポート、ログ、アクセスレビュー、変更のトレーサビリティ）を提示しなければなりません。

重複を避け、監査準備を容易にするため、この章では上記の機能をコンプライアンス要件と関連付けて説明します。

### 目的

- 適用される基準と、アプリケーションのコンプライアンスで期待される範囲（内部および外部の要件）を定義します。
- これらの要件を測定可能な統制に翻訳し、さらに監査のための証拠へと変換します。
- 証拠（レポート、ログ、アクセスレビューなど）のリポジトリを確立します。
- ガバナンス（アクセスレビュー、例外管理、職務分離（SoD）の競合処理）を正式化し、標準化された監査ファイルを準備します。

## 機能とツール

コード署名、DB/ドライバー接続の暗号化、セキュアなネットワークプロトコル、OAuth/REST 統合機能など、頻繁に発生する要件をサポートします。。	PB
脆弱性の低減と <a href="#">継続的な改善アプローチ</a> を実証するのに役立つ、繰り返し可能なバージョン管理された分析（特に PowerBuilder セキュリティルールに関する分析）を提供します。	VE
セキュリティチェックとレポートを自動化し、IAM、RBAC、 <a href="#">職務分離</a> 、追跡可能性、監査といった <a href="#">現代的な基準に準拠</a> します。	VG
<a href="#">権限マトリックス</a> を公開することで「誰が何をアクセスできるか」を文書化します。	VG
ソースコードの脆弱性検出において再現性のある結果を生成するため、検査ルールのリポジトリを活用します。	VE

## コンプライアンス

Visual Guard は、サポートされている標準（ISO 27001、NIST SP 800-53、NIST SP 800-63、CIS コントロール、PCI DSS、GDPR など）のマップを公開し、関連するメカニズム（MFA、RBAC、SoD、ロギング、アクセスレビュー）を指定します。（[セキュリティ標準とコンプライアンス](#)）

## エビデンス

- 対応マトリックス：制御項目／要件／機能（PB、VE、VG）／生成される証拠。（[セキュリティ基準とコンプライアンス](#)）
- 標準レポートセット：VE スキャン、権限マトリックスエクスポート、監査ログ、署名手順。（[PowerBuilder コードのセキュリティ脆弱性検出](#)）（[権限マトリックス](#)）（[PowerBuilder コード署名オプションの使用](#)）
- ガバナンスプロセス：アクセスレビュー頻度、例外管理、SoD 競合処理。（[職務分離 \(SoD\)](#)）

## PowerBuilder アプリケーションのセキュリティ確保のためのチェックリスト

これらのチェックリストは、ほとんどのコンテキストに適用可能なアクションの後に、組織のアーキテクチャと規制上の制約に応じてアクティブ化されるオプションのアクションが続くという実用的なアプローチを提供します。

### チェックリスト A - 汎用対策（ほとんどのケースで推奨）

章	対策
横断的（前提条件）	アプリケーション、環境、依存関係、フロー（ワークステーション、サーバー、データベースなど）をインベントリ化する。
	取り扱うデータを分類し、関連する保護要件を正式に定義する。
	修復プロセスを確立する（優先順位付け、目標タイムライン、検証、追跡）。
	パッチ適用と公開コンポーネントの更新（OS、データベース、ミドルウェア、ライブラリ、リモートアクセス）。
	インシデント対応計画を正式化し、DR/BCP（復旧、回復）をテストする。
2. 実行ファイルとコンポーネント	特定可能なリリースに紐づく、再現可能なバージョン管理されたビルドプロセスを作成する。
	会社承認の証明書を使用して、すべての実行ファイル、ライブラリ、配布パッケージに署名する。
	配布前にアーティファクトの完全性を検証（ハッシュ/フィンガープリント）し、結果を記録する。
	配布されたバージョン（ハッシュ、日付、所有者）を追跡し、履歴を保持する。
	配布に関連する公開権限と管理権限を制限し監査する。
	リリース前に PowerBuilder コードをスキャンし、脆弱性と不適切な慣行を特定する。
	コードやスクリプトからハードコードされた機密情報（ID、パスワード、キー、IP アドレス）を検出し削除する。
	アプリケーション実行ファイルでサポートされる実行ファイルセキュリティ フラグを有効化する。（DEP / ASLR / CFG / SafeSEH）
	承認されたパスのみに DLL の読み込みを制限し、該当する場合は DLL 署名検証を有効化する。（Strict モード / DLL 署名検証）
3. データの保護	機密データとその保存場所（データベース、ファイル、エクスポート、ログ、バックアップ）を特定する。
	DB サーバーおよび PB DB ドライバーで TLS 1.2 以上（証明書検証を含む）を使用するように TLS を有効にする。
	必要に応じて保存中の機密データを暗号化する（データベース、ファイル、エクスポート、バックアップ）。
	暗号化キーを保護する（保管、アクセス制御、ローテーション、失効）。
	接続プロパティを暗号化する。暗号化された接続文字列を生成する（平文のシークレットは使用しない）。
	脆弱な暗号化方式（例：DES/3DES）を検出し排除する。堅牢なモードを強制する。
	廃止されたハッシュ（例：SHA-1/MD5）を検出し排除し、堅牢な代替手段を強制する。
	セキュリティポリシーに対して暗号の強さ（鍵サイズ、動作モード、パディング）を検証する。
	ロール/権限によるアクセスを厳しく制限し、機密データへの露出を減らす。
4. ネットワークプロトコル	ネットワークフローと実際にネゴシエートされたプロトコル（TLS バージョン、スイート、証明書）をインベントリ化する。

章	対策
	<p>廃止されたプロトコルを削除する。TLS 設定を会社のポリシーに合わせる。</p> <p>証明書検証とエラー処理（バイパスやサイレントダウングレードを許さない）。</p> <p>必要に応じて相互 TLS 認証（mTLS）を実装する（内部/パートナーAPI）。</p> <p>プロキシ、ゲートウェイ、ネットワーク保護機能との互換性をテストする（RESTClient / HTTPClient）。</p> <p>準拠していない、またはレガシーなネットワーク呼び出しを検出し、その置換を優先する。</p>
5. 外部サービスおよび API とのやり取り	<p>インベントリ統合（API、内部サービス、SMTP）および交換されるデータ。</p> <p>統合には SOAP/INET より REST を優先する（RESTClient）。</p> <p>SOAP が必要な場合は、TLS 1.2 以上と強力な認証を備えた HTTPClient を使用する。レガシー SOAP クライアントは避ける。</p> <p>HTTPClient/ OAuthClient を使用し、ユーザー名とパスワードではなくトークンベース認証を採用する。</p> <p>トークンおよび技術アカウントの権限を厳格に制限する（スコープ、権限）。</p> <p>ハードコードされたシークレットを削除し、環境ごとに保管、ローテーション、失効を管理する。</p> <p>入力とパラメータにおける注入リスク（SQL、OS コマンド、パス）を検知し修正する。</p> <p>公開フロー（サービス、API、ファイル、メール）における入力の検証と出力のエンコードを実施する。</p> <p>機密操作（エクスポート、送信、重要アクション）の追跡と監視/アラート設定を実施する。</p>
6. 組み込みブラウザ	<p>ブラウザを埋め込んだ画面をインベントリ化し、期待されるコンテンツと許可された URL を定義する。</p> <p>サポートされている最新のブラウザエンジンに移行し、廃止されたコンポーネントを削除する。</p> <p>信頼できないコンテンツに対するブラウジング、URL オープン、スクリプト実行を制限する。</p> <p>望ましくない実行を防ぐため、JavaScript インタラクション（アプリケーションブリッジ）を管理する。</p> <p>本番環境でのリモートデバッグを無効化し、管理された診断に限定する。</p> <p>「Web」画面へのアクセスを制限する。</p> <p>ウェブアクセスを監視し、異常な動作時に通知をトリガーする（トレーサビリティと監査）。</p>
7. ビルドと配布	<p>明示的な合格基準を持つ標準パイプライン（ビルド、テスト、分析、署名、デプロイ）を定義する。</p> <p>バージョン管理と自動化を簡素化するためソリューション形式に移行する（Solution）。</p> <p>マージ競合を減らしトレーサビリティを向上させるため、PB 2025+ へ移行する。</p> <p>ビルドの再現性を確保（バージョン、依存関係、パラメータ）。バージョンごとの成果物とレポートを保持。</p> <p>パイプラインの技術アカウントを保護（最小権限、役割分離、監査）。</p> <p>パイプラインのシークレットとキーを保護する（安全な保管、アクセス制限、ローテーション）。</p> <p>すべてのビルドで SAST 分析を工業化し、重大なルール違反が発生した場合はリリースをブロックする。</p> <p>本番環境での危険な慣行（例：コンソールログ出力）を回避し、例外を無視しない。</p> <p>開発/テスト/本番環境間でセキュリティ設定のデプロイを標準化する（不一致を回避）。</p> <p>ビルド/リリース検証の一部として、実行ファイルのハードニング設定と DLL 読み込み制限を検証する。（PESecurity / dumpbin / WinDbg）</p>
8. 認証とアクセス制御	<p>ID 管理を集中化（AD/Entra ID）し、ローカルアプリケーションアカウントを削減する。</p> <p>機密性の高いアプリケーションと特権アカウントには多要素認証（MFA）を有効化する。</p> <p>最小権限の原則に沿ったロールと権限を通じて権限をモデル化する。</p>

章	対策
	<p>専用サーバー（Identity Server）を介して認証、認可、ログ記録を一元化する。</p> <p>重要な業務における職務分離（SoD）を実施し、競合を解消する。</p> <p>定期的なアクセス権限の見直しを実施し、証拠を保持する。</p> <p>監査対応の権限マトリックスを作成する（誰が何をアクセスできるか）。</p> <p>機密性の高いログインと操作（誰が、何を、いつ、どこで）をログ記録し、監視を一元化する。</p>
10. 現代的な基準への証拠	<p>適用される基準とアプリケーションのコンプライアンス範囲を定義する。</p> <p>要件を測定可能な管理措置と期待される証拠（レポート、ログ、アクセスレビュー）に変換する。</p> <p>証拠リポジトリを確立し、バージョン/環境ごとに履歴を保持する。</p> <p>ガバナンスを標準化する（アクセスレビュー、例外管理、SoD（職務分離）の衝突処理）。</p> <p>要件、統制、ツール、証拠を連携させた再利用可能な監査パックを構築する。</p>

## チェックリスト B - アーキテクチャと環境に応じたオプション対策

章	対策
2. 実行ファイルとコンポーネント	PB 2025 R2 ベータを使用する場合：より広範な展開の前に、互換性のあるプロジェクトタイプで高度な実行ファイルセキュリティフラグを評価する。（DEP / ASLR / CFG / SafeSEH）
5. 外部サービスおよび API との交換	アプリケーションが API を消費する場合：強力な認証と鍵/トークンのローテーション（OAuth 2.0）
	アプリケーションが API を消費する場合：トークンのスコープと権限を制限する（OAuthClient）
	メールの場合：ネイティブメールサポートと最新認証方式を使用（SMTP クライアント / XOAUTH2） メールの場合：セキュアな中継（認証、TLS）、添付ファイルの制限、機密送信の追跡（XOAUTH2）
6. 組み込みブラウザ	Web/HTML コンポーネントの場合：コンテンツを制御し、信頼できないスクリプトを制限する（WebBrowser）
	Web/HTML コンポーネントの場合：レガシーエンジンをサポート対象エンジンに置き換える（WebView2）
7. ビルドとデプロイ	工業化の場合：ビルドスクリプトとパイプラインオーケストレーションを標準化（PBAutoBuild）
	より強力な QA の場合：分析結果に基づく受け入れ基準を構築（SAST）
8. 認証とアクセス制御	権限衝突がある場合：職務分離（SoD）ルールとは是正措置を定義する
	組織が複雑な場合：組織構造を反映したユーザー・グループ階層を構築し、権限を委譲する。
	複数のディレクトリに ID が分散している場合：ID フェデレーションを実装する。 組織が監査を実施する場合：権限マトリックスを生成・活用する（Permission Matrix）
9. PowerServer によるモダンアーキテクチャ	クライアントからデータベースにアクセス可能な場合：PowerServer を用いた n 層アーキテクチャへ移行
	アプリケーションが API を公開/利用する場合：API 認証を標準化（OAuth 2.0）
	アプリケーションが API を公開/利用する場合：標準トークンを使用し、不正アクセスを制限する（JWT）
	サーバーサイド制御（認証、認可、ロギング）を集中化する。
	最も機密性の高いモジュールを優先して段階的な移行を計画する（プログレッシブ移行）。
	サーバーコンポーネントをサポート対象の .NET バージョンに更新する（.NET）。
API アクセス権限を管理し、監査レポートを生成する（権限 / 監査活動）。	
10. 現代的な標準への準拠	厳しい要件がある場合（金融、医療など）：コンプライアンスガバナンスと正式な制御（監査活動）
	厳格な要件がある場合：監査証拠とその保持の工業化（権限マトリックス）

## よくある質問

### 1) 既存の PowerBuilder アプリケーションのセキュリティ強化はどこから始めるべきか？

まず全体像を把握することから始めます。コンポーネント（実行ファイル、DLL、OCX、スクリプト）、フロー（ワークステーション ↔ サーバー ↔ データベース）、機密データ、依存関係を洗い出します。次に PowerBuilder コードに対して SAST スキャンを実行し、脆弱性と不適切な手法を迅速に特定し、修正の優先順位付けを行います。最後に、コード署名、転送中の暗号化、アクセス制御、ログ記録、監査証跡などの「基盤」を整えます。

### 2) PowerBuilder アプリケーションで最もよく見られる脆弱性は何ですか？

一般的な発見事項には、ハードコードされた機密情報（ユーザー名、パスワード、キー）、レガシーなネットワーク呼び出し、「ケースバイケース」で実装され監査が困難なアクセス制御、不十分な入力検証（SQL/コマンド/パスインジェクションリスク）、調査を支援するには脆弱すぎるログ記録が含まれます。PowerBuilder に特化した SAST スキャンは、これらのリスクを可視化し、修正対象を特定する効果的な方法です。

### 3) PowerBuilder からデータベース接続を暗号化する方法は？

ワークステーション、アプリケーションサーバー、データベース間の転送データを暗号化することが目的です。クライアント側とサーバー側の両方で TLS 設定（バージョン、暗号スイート、証明書）を確認してください。データベースに応じて利用可能な最も厳格な暗号化モードを有効化し、プロキシ、ゲートウェイ、強化されたネットワークポリシーが存在する環境での互換性をテストしてください。

### 4) PowerBuilder アプリケーションにおける「転送中」と「保存時」の暗号化の違いは？

転送中の暗号化はネットワーク上のデータ（傍受、改ざん）を保護します。保存中の暗号化は、盗難、不正アクセス、侵害が発生した場合に保存されたデータ（データベース、ファイル、エクスポート、ログ、バックアップ）を保護します。両者は補完関係にあります。暗号化された通信経路は、保護されていないローカルエクスポート経由の漏洩を防げません。同様に、暗号化されたストレージは、通信が暗号化されていない場合、ネットワークスニффイングを防げません。

### 5) 避けるべき暗号化およびハッシュアルゴリズムは？

脆弱または陳腐化したアルゴリズムや関数（例：DES/3DES、MD5、SHA-1）、不適切なモードやパディングは避けるべきです。堅牢なパラメータ（鍵サイズ、動作モード）と適切な鍵管理（保管、ローテーション、失効）を備えた現代的なアルゴリズムを優先してください。「適切なアルゴリズム」+「適切なパラメータ」+「適切な使用法」など、一貫性が重要です。

### 6) PowerBuilder 実行ファイルの公開とインストールをどのように保護しますか？

信頼の連鎖を確立する：再現可能かつバージョン管理されたビルド、アーティファクトの体系的な署名、完全性検証（ハッシュ）、デプロイ版追跡（日付、所有者、フィンガープリント）。公開/デプロイ権限を制限し監査する。目的はサプライチェーン侵害リスクの低減と調査の容易化です。

## 7) ビルド/デプロイパイプラインが重要なセキュリティポイントである理由は？

パイプラインが侵害されると、「正当な」リリースに悪意のあるコードを注入し、大規模に配布できるためです。技術アカウント、シークレット、署名キー、リポジトリを保護します。すべてのビルドでSASTを工業化し、アーティファクトとレポートを保持し、本番環境前に明確なリリース承認基準を定義します。「コードのみ」のセキュリティでは不十分です。

## 8) PowerBuilder アプリケーションに SSO と MFA を実装するには？

原則として、ID 管理を集中化（エンタープライズディレクトリ）し、特に機密性の高いアプリケーションや特権アカウントに対してはMFAを含む強力な認証メカニズムを適用します。PowerBuilder アプリケーションでは、これは一貫したIAMアプローチ(集中認証、ロール/権限管理、ログ記録、エビデンス(アクセス レビュー、レポート)の作成機能)に帰着します。

## 9) 権限管理とアクセスレビューの準備をどう行うか？

最小権限原則に沿ったロール/権限モデルと、権限の完全かつ実用的な可視化機能が必要です。定期的なアクセスレビューを実施し、例外を処理し、承認証拠を保持します。「権限マトリックス」（または同等のもの）は監査を簡素化し、レビューを迅速化します。

## 10) 職務分離（SoD）とは何か？PowerBuilder アプリケーションで重要な理由は？

SoD は、互いに矛盾する権限（例：支払いの作成と承認、管理と監査）を1人が保持することを防止します。内部不正リスクを低減し、コンプライアンスを強化します。PowerBuilder アプリケーションでは、役割のモデリング、SoD ルールの定義、競合の検出、是正、監査証拠の保持を意味します。

## 11) 技術的対策とコンプライアンス要件（ISO、NIST、PCI、HIPAA、SOX）をどのように関連付けるか？

まず適用される基準と範囲を定義します。次に要件を測定可能な統制と証拠（スキャンレポート、ログ、アクセスレビュー、権限マトリックス、SoD ルール、署名手順、TLS 設定など）に変換します。最も有用な成果物は、各監査ごとに再利用可能な「要件→統制→ツール→証拠」のマッピングです。

## 参考文献 – 本ガイドで引用した情報源

- [2023年の3CXインシデント](#) – CISA (DHS)
- [133の悪意のあるWindowsドライバー](#) – Sophos; 著者: アンドルー・ブランド
- [セキュリティ - 新着情報](#) – Appeon
- [PowerBuilder コードのセキュリティ脆弱性の検出](#) – Novalys (Visual Expert)
- [権限](#) – Novalys (Visual Guard)
- [セキュリティ基準とコンプライアンス](#) – Novalys (Visual Guard)
- [監査活動](#) – Novalys (Visual Guard)
- [Verizon 2025 DBIR - エグゼクティブサマリー](#) – Verizon; 著者: Verizon
- [CISA - ランサムウェアガイド](#) – CISA (DHS)
- [MD アンダーソン事件](#) – 米国 HHS
- [Secure Connection Encryptor - 新機能](#) – Appeon
- [SQL Server の厳格な暗号化をサポート - 新機能](#) – Appeon
- [パーミッションマトリックス](#) – Novalys (Visual Guard)
- [NIST SP 800-52r2](#) – NIST; 著者: Kerry A. McKay; David A. Cooper
- [PCIDSS](#) – PCI SSC
- [HTTP/2 のサポート - 新機能](#) – Appeon
- [PowerBuilder 2022 - 新機能](#) – Appeon
- [双方向 TLS 認証のサポート - 新機能](#) – Appeon
- [Identity Server API の使用方法](#) – Novalys (Visual Guard)
- [オブジェクトレベルのアクセス制御の欠如](#) – OWASP
- [FBI IC3 2024](#) – FBI / IC3; 著者: FBI インターネット犯罪苦情センター (IC3)
- [Optus インシデント \(2022 年\)](#) – ロイター; 著者: ロイター
- [ネイティブメールサポート \(SMTP クライアント\) - 新機能](#) – Appeon
- [強化された RESTClient オブジェクト - 新機能](#) – Appeon
- [トレーサビリティと監査](#) – Novalys (Visual Guard)
- [IE モードでの Edge の使用](#) – Microsoft
- [Microsoft ドキュメント](#) – Microsoft
- [CVE-2021-40444](#) – CISA (DHS); 著者: CISA
- [WebBrowser エンジンのアップグレード - 新機能](#) – Appeon
- [WebBrowser の機能強化 - 新機能](#) – Appeon
- [WebBrowser でのリモートデバッグのサポート - 新機能](#) – Appeon
- [SolarWinds インシデント](#) – CISA (DHS)
- [超高速コンパイラと新しいソリューション形式 - 新機能](#) – Appeon
- [DBIR 2025](#) – Verizon
- [職務分離 \(SoD\)](#) – Novalys (Visual Guard)
- [PowerServer Web API](#) – Appeon
- [アーキテクチャ - PowerServer ヘルプ](#) – Appeon
- [「重要な」サイバーセキュリティインシデントの報告](#) – 米国証券取引委員会 (SEC) ; 著者: Erik Gerding
- [データ侵害のグローバル平均コストは 488 万ドルに達する](#) – www.ibm.com