



Forest Book

FOREST SECURITY WHITEPAPER

Shopifyアプリ開発者のための 保護対象顧客データ Level 2 実務ガイド

要件を自社の構成に当てはめ、証跡を残し、公式セキュリティチェック
36項目の先を確認する

発行 株式会社Forest Book (Forest Book Inc.)

Forest Security (株式会社Forest Book)

2026年10月22日 版1.0

本資料の複製・社内配布は自由。再配布時は出典を明記のこと。

この資料について

項目	内容
題名	Shopifyアプリ開発者のための 保護対象顧客データ Level 2 実務ガイド
副題	要件を自社の構成に当てはめ、証跡を残し、公式セキュリティチェック36項目の先を確認する
発行	Forest Security（株式会社Forest Book）
版	1.0（2026年10月22日）
想定読者	Shopify公開アプリ・カスタムアプリを開発・保守する会社のテックリード、PM、情報セキュリティ担当
想定構成	Node.js / TypeScript で書かれたアプリを AWS サーバーレス（API Gateway、Lambda、DynamoDB、SQS、S3、CloudWatch）で運用している構成。他の構成でも「証跡として残すもの」の列はそのまま使える
一次情報の確認日	2026年10月22日。shopify.dev の該当ページを同日に直接確認した。出典一覧は末尾
この資料が保証しないこと	Shopify のアプリ審査・データ保護レビューの可否。要件の解釈は公式文言の要約であり、判断は Shopify が行う

本文中の「公式」は shopify.dev の記載を指す。公式に書かれていないことは「公式には記載なし」と明記した。AWS 側の実装例は Forest Book が実務で使っている構成の一例であり、Shopify が要求しているものではない。

この資料の使い方

保護対象顧客データ（Protected customer data、以下 PCD）は、Shopify が「顧客または見込み顧客に直接関係するデータ」と定義する範囲である。そのうち氏名、住所（番地、地理情報、郵便番号を含む）、メール、電話は「保護対象顧客フィールド」として個別に申請する。

Level 1 はこれら4フィールドを含まない顧客データ、Level 2 は4フィールドを含む顧客データを扱う場合の区分である。要件は Level 1 が9項目、Level 2 はそれに7項目を加えた計16項目になる。

公開アプリとカスタムアプリで何が違うか

公式ページの表（2026-10-22確認）をそのまま転記する。

Level	公開アプリ	カスタムアプリ	管理画面で作成したカスタムアプリ
1	Requires review（レビュー必要）	Always available（常時利用可）	Always available
2	Requires review	Always available	Varies by plan（プランによる）

加えて、開発ストアにのみインストールするアプリはレビュー提出が不要である。

この表が意味することは次の3点。

読者の立場	表が意味すること	この資料の使いどころ
公開アプリを出す	Level 1・2 ともパートナーダッシュボードで申請し、レビューを受ける。Shopify は「詳細なレビューを求める場合がある」と明記しており、その際は要件を満たす証拠の提示が必要になる	要件一覧の「証跡として残すもの」を先に揃える。申請フォームの Data protection details で確認する内容と同じ
カスタムアプリを作る・運用する	レビューなしで Level 2 のデータに常時アクセスできる。つまり、同じ氏名・住所・電話・メールを扱っていても、第三者の確認が一度も入らない。16項目の要件は適用される（ページは要件を全アプリに対して書いている）が、満たしているかを誰も見ない	要件一覧と自己点検チェックリストを、社内または発注元への説明資料として使う
発注側（加盟店・元請け）	委託先が作ったカスタムアプリが PCD 要件を満たすかは、自分で確認するしかない	自己点検チェックリストを委託先に渡し、根拠の提出を求める

「Varies by plan」の具体的な条件は公式ページに記載がない（公式には記載なし）。

この資料で扱わないこと

Shopify 本体、Shopify Payments、チェックアウトのプラットフォーム側の統制。Shopify Functions や Checkout UI 拡張に固有の要件。各国の個人情報保護法そのものの解説。

Level 1・Level 2 の要件一覧

公式ページは Level 1 を1～9、Level 2 を9～15と番号付けしている（9が重複する）。本書では公式番号を併記しつつ、区別のため Level 2 側を L2-1～L2-7 と呼ぶ。

各要件に「AWSサーバーレス構成での実装例」「証跡として残すもの」「よくある不備」を付けた。実装例は一例であり、要件を満たす唯一の方法ではない。

Level 1（保護対象顧客データを扱う全アプリ）

公式番号	要件（公式文言の要約）	AWSサーバーレス構成での実装例	証跡として残すもの	よくある不備
1	アプリ機能の提供に必要な最小限の個人データのみ処理する	GraphQL クエリの選択フィールドを機能ごとに絞る。DynamoDB に保存する属性を設計書で列挙し、それ以外は書き込まない	データ棚卸し表（保存する属性、用途、取得元クエリ、保持期間）。shopify.app.toml のスコープ一覧	顧客オブジェクトを丸ごと保存している。使っていないフィールドを「将来のため」に取得している
2	どの個人データを何のために処理するかを加盟店に伝える	プライバシーポリシーとデータ処理契約に、棚卸し表と同じ項目を書く	公開中のプライバシーポリシー URL と改訂履歴。App Store リスティングの記載	ポリシーの記載と実際に保存している属性が一致していない
3	処理を明示した目的に限定する	分析・マーケティング用途の二次利用を行う Lambda を分け、設計レビューで目的外利用を止める	設計書の「目的」欄。目的追加時の変更記録	サポート目的で取得したメールをメール配信に流用している

公式番号	要件（公式文言の要約）	AWSサーバーレス構成での実装例	証跡として残すもの	よくある不備
4	該当する場合、顧客の同意判断を尊重し適用する	同意状態を示すフィールド（マーケティング同意等）を条件に処理を分岐する	分岐箇所のコード参照（ファイルと行）。単体テスト	同意フラグを読まずに一律処理している
5	該当する場合、データ共有（販売等）からのオプトアウトを尊重する	第三者連携がある場合は、オプトアウト顧客を除外するフィルタを連携バッチに入れる	除外ロジックのコード参照とテスト結果	第三者連携がないのに「該当なし」の判断を記録していない
6	自動化された意思決定を行う場合、顧客が手動処理を求められるようにする	与信・不正判定などを自動化するなら、人手レビューへの切り替え経路を用意する	該当有無の判断記録。該当する場合は手順書	該当有無を検討した記録がない
7	加盟店とプライバシー・データ保護契約を結ぶ	利用規約に DPA（データ処理契約）を含め、インストール時に同意を取る	DPA の文面、版、同意取得のログ	利用規約に個人データの扱いが書かれていない
8	保持期間を定め、必要以上に保持しない	DynamoDB の TTL 属性、S3 ライフサイクルルール、shop/redact 受信時の削除 Lambda	TTL 設定の CLI 出力。ライフサイクルルールの JSON。削除処理のログ	アンインストール後もデータが残る。TTL が一部テーブルにしか設定されていない
9	保存時・通信時にデータを暗号化する	DynamoDB・S3・SQS・CloudWatch Logs の保存時暗号化（既定または KMS）。API Gateway のカスタムドメインで TLS 1.2 以上。Shopify への通信は HTTPS のみ	各サービスの暗号化設定の CLI 出力（describe-table、get-bucket-encryption 等）。TLS 設定のスクリーンショット	ログ出力先やバックアップ先の暗号化を確認していない。開発環境だけ HTTP

Level 2（氏名・住所・電話・メールを扱う場合に追加）

本書番号	公式番号	要件（公式文言の要約）	AWSサーバーレス構成での実装例	証跡として残すもの	よくある不備
L2-1	9	データのバックアップを暗号化する	DynamoDB のポイントインタイムリカバリとオンデマンドバックアップは保存時暗号化される。AWS Backup を使う場合はボールの KMS 鍵を確認。S3 のクロスリージョンレプリケーション先も暗号化	バックアップ設定とボールの暗号化設定の CLI 出力。復元テストの記録	手動エクスポート（CSV、ダンプ）がローカル PC や共有ドライブに残る
L2-2	10	テストデータと本番データを分ける	本番とステージングを別 AWS アカウントに分け、ステージングには合成データのみを投入する。開発ストアを使う	アカウント構成図。ステージングへのデータ投入手順書	本番の DynamoDB エクスポートをステージングにそのまま復元している

本書番号	公式番号	要件（公式文言の要約）	AWSサーバーレス構成での実装例	証拠として残すもの	よくある不備
L2-3	11	データ損失防止（DLP）の戦略を持つ	顧客データを含む S3 バケットのパブリックアクセスブロック、VPC 外への持ち出し経路の棚卸し、Macie 等での検出、ダウンロード権限の限定	DLP 方針書（技術統制、ポリシー、基準の組み合わせであることが公式の記述）。設定の CLI 出力	方針書がなく、設定だけがある。または方針書だけがあり設定がない
L2-4	12	保護対象顧客データへのスタッフアクセスを制限する	本番 DynamoDB・S3 への読み取りは IAM Identity Center の限定グループのみ。一時的昇格は期限付き。コンソールからの直接閲覧を監査	IAM ポリシーと所属一覧。アクセス権限の棚卸し記録（四半期ごと等）	全開発者が本番の管理者権限を持つ。退職者のアクセスキーが残る
L2-5	13	スタッフアカウントに強固なパスワードを求める	IAM Identity Center または IAM のパスワードポリシー（最小長、文字種）と MFA 必須化。パートナーダッシュボード、GitHub、パスワードマネージャーにも同じ基準を適用	パスワードポリシーの CLI 出力（ <code>get-account-password-policy</code> ）。MFA 有効化率の一覧	AWS だけ厳しく、パートナーダッシュボードや GitHub が個人アカウント・MFA なし
L2-6	14	保護対象顧客データへのアクセスログを保持する	CloudTrail のデータイベント（DynamoDB・S3）を有効化。アプリ層では「誰が、どのストアの、どの顧客を、何の目的で」読んだかを構造化ログに出す。ログは別アカウントの S3 に転送し改ざんを防ぐ	CloudTrail 設定の CLI 出力。ログの保持期間設定。ログを定期的にレビューした記録（公式は「保持し、レビューする」と記載）	ログを取っているがレビューしていない。ログ本体に氏名やメールがそのまま出ている（後述の確認7を参照）
L2-7	15	セキュリティインシデント対応方針を実装する	方針書に重大度区分、役割、エスカレーション経路、証拠保全、必要な措置（加盟店・Shopify・当局への連絡を含む）を書く。GuardDuty や CloudWatch アラームの通知先を方針書の連絡経路に合わせる	方針書の版と承認記録。年1回程度の机上演習の記録	方針書が「作っただけ」で、通知先やオンコールが現実と合っていない

要件と「技術で確認できるもの」「文書で示すもの」の区別

区分	該当要件	確認の仕方
設定・コードで確認できる	1、8、9、L2-1、L2-2、L2-4、L2-5、L2-6	CLI 出力、コード参照、テスト結果
文書と運用記録で示す	2、3、7、L2-3、L2-7	方針書、契約、レビュー記録、演習記録
該当有無の判断記録が主	4、5、6	「該当する／しない」の判断と理由を残す

第三者が確認できるのは主に1行目で、2行目は文書の有無と現実との整合、3行目は判断が記録されているかを見る。Forest Security Review の要件対応表もこの区分で作る。

データ保護レビューの流れと準備

公式に記載されている申請手順

手順	内容（公式文言の要約）
1	パートナーダッシュボードの Apps でアプリを選ぶ。先に配布方法（distribution method）を選んでおく必要がある。開発ストア向けでも同じ
2	サイドバーの API access requests を開く
3	Protected customer data access の Request access を押す
4	Protected customer data を選び、利用理由を記入して保存する
5	必要なら保護対象顧客フィールド（氏名、住所、メール、電話）を個別に選び、それぞれ理由を記入して保存する
6	Data protection details で、要件を満たしていることを確認する
7	アプリをレビューに提出する

開発ストアだけで使う場合は手順5まででデータにアクセスできる。

データ保護レビューについて公式が書いていること

公式の記述	準備への含意
Shopify は実務の詳細なレビューを求める場合があり、その際は要件遵守の証拠を提示する必要がある	本書の「証跡として残すもの」を、申請前に1か所に集めておく
どのアプリも対象になりうるが、加盟店インストール数が多い、顧客レコード量が多い、承認フィールド数が多い、個人データの保持期間が長いアプリに重点が置かれる見込み	フィールド数と保持期間は自分で減らせる。申請前に棚卸し表で見直す
承認は「アプリ機能の提供に必要な最小限を要求しているか」による	理由欄には機能と結びつけて書く。「将来のため」は最小限ではない
承認されなかったフィールドは null で返る。GraphQL は HTTP 200 で <code>errors</code> に理由が入る	未承認状態でもアプリが壊れないよう、null とエラーの処理を実装しておく

レビューの所要日数、審査基準の詳細、不承認時の再申請手順は公式ページに記載がない（公式には記載なし）。本書は合否を予測しない。

申請前に揃える書類一式

書類	対応する要件	形式の例
データ棚卸し表	1、2、3、8	属性、取得元、用途、保存先、保持期間、削除手段の表
プライバシーポリシーと DPA	2、7	公開 URL と PDF 控え
暗号化・バックアップ設定の出力	9、L2-1	CLI 出力をそのまま保存（日付入り）
環境分離の構成図	L2-2	アカウント ID と用途の表
DLP 方針書	L2-3	1〜2ページで可。統制の一覧表を含む
アクセス権限一覧とパスワードポリシー	L2-4、L2-5	IAM の出力と、パートナーダッシュボード・GitHub の権限一覧
アクセスログ設定とレビュー記録	L2-6	設定出力と、月次レビューの議事メモ
インシデント対応方針	L2-7	版管理された文書

最近の変更点（2024年～2025年）

PCD 要件そのものの2026年の改定は、チェンジログの検索では見つからなかった。ただし「変更がない」と断定はしない。関連する変更で一次情報が確認できたものを挙げる。

公開日	施行日	変更（公式文言の要約）	開発者への影響
2024-03-15	同日 （Action Required）	Admin GraphQL・REST の全バージョンで <code>order.statusPageUrl</code> の取得に Level 2 が必要になった	注文ステータス URL を使う通知系アプリは Level 2 申請が要る
2024-08-26	同日	Customer Account API の利用に Level 1 以上の承認が必要になった。カスタムアプリ、Hydrogen、ヘッドレスは既にアクセスを持つため対象外	公開アプリで顧客アカウント連携をするなら申請が前提
2025-11-10	2025-12-10	Web Pixel 拡張のイベントペイロードに氏名・メール・電話・住所が含まれるのは、対応する保護スコープ（ <code>read_customer_name</code> 、 <code>read_customer_email</code> 、 <code>read_customer_phone</code> 、 <code>read_customer_address</code> 、 <code>read_customer_personal_data</code> ）の承認済みアプリのみ。未承認は該当フィールドが null になり、データ構造は変わらない。ストアフロント、チェックアウト、顧客アカウントの各面で適用。カスタムピクセルは対象外	計測・広告系アプリは Web Pixel でも PCD 承認が必要。null を前提にした実装とテストが必要

このほか、PCD とは別軸だが同時期に確認しておくべき期限として、公開アプリは2027年1月1日以降、期限なしのオフラインアクセストークンで GraphQL Admin API を呼べなくなる（移行ガイドに記載。カスタムアプリと加盟店作成アプリは対象外）。

Shopify 公式 shopify app security check 36項目の読み方

ツールの位置づけ（公式の記述）

項目	公式の記述
公開	2026年10月15日、Shopify CLI / AI Toolkit に追加
何をするか	アプリのコードベース、各拡張ポイントの設定、依存関係を確認する。指摘ごとに該当ファイルと行、重大度、理由、修正方法へのリンクを示す。ローカルで動く
いつ実行するか	アプリ提出前と、すべてのリリースで
方式	決定的スキャン（ローカルで動く固定のコード検査）と、エージェントレビュー（コード検査だけでは判定できない項目をコーディングエージェントに渡す）。両方を使う項目もある
制約	「このコマンドはアプリのソースコードの確認に限定され、多層的なセキュリティ対策の一部とすべき」（CLI リファレンス）。決定的チェックの一部は React Router テンプレートのアプリのみ対応で、他フレームワークはエージェントレビューに頼る
出力先	<code>.shopify/app-security/</code> （アプリ設定ごとに別フォルダ）。 <code>shopify app security clean</code> で削除
終了コード	<code>--blocking high medium low</code> で、その重大度以上の指摘があるときに非ゼロ終了。既定は <code>none</code> （指摘があっても失敗しない）
審査との関係	審査で必須とは書かれていない（公式には記載なし）

バックエンドを別ディレクトリに置く構成では `--include-dir ./backend` を付けないとバックエンドが検査されない。CI では `--no-input --skip-instructions --blocking high` の組み合わせで決定的スキャンだけを回し、エージェントレビューは人が別途行うのが扱いやすい。

36項目の分類

公式ページは項目を重大度で分けている（High 25、Medium 9、Low 2）。本書では、是正の担当と手順が異なるため、項目を6つの観点に分類し直した。分類は Forest Book によるもので公式のものではない。

観点	High	Medium	Low
認証・セッション	UNAUTHENTICATED_ENDPOINT 、 SESSION_LIFECYCLE_AND_REPLAY 、 WEAK_SHOP_VALIDATION 、 APP_PROXY_UNVERIFIED_SIGNATURE	OPEN_REDIRECT 、 CSRF_MISSING_PROTECTION 、 EXPIRING_OFFLINE_TOKEN	—
認可・テナント分離	MISSING_AUTHORIZATION_CHECK 、 MISSING_TENANT_ISOLATION 、 REQUEST_DERIVED_SHOP_SCOPE 、 REQUEST_CONTROLLED_ADMIN_CONTEXT 、 UNSCOPED_SHOP_CONFIG_WRITE 、 METAFIELD_OFFLINE_TOKEN	OVERBROAD_DATA_ACCESS	—
シークレットと資格情報	COMMITTED_SECRET 、 CREDENTIAL_LOG_LEAKAGE 、 CREDENTIAL_BROWSER_LEAKAGE	—	—

観点	High	Medium	Low
インジェクション・出力	CROSS_SITE_SCRIPTING、 UNSAFE_INNERHTML、SQL_INJECTION、 SSRF_REQUEST_FORGERY、 APP_PROXY_LIQUID_INJECTION、 TEXT_SETTING_HTML_SMUGGLING、 THEME_EXTENSION_XSS、 SCRIPT_TAG_URL_INJECTION、 ACTIVE_UPLOADS_AND_PRIVILEGED_PREVIEWS	LIQUID_UNSAFE_RENDER	—
設定・プラットフォーム要件	STATIC_FRAME_ANCESTORS、 INSECURE_WEBHOOK_URL、 SCOPE_OVER_REQUEST	MISSING_EMBEDDED_CSP、 MISSING_COMPLIANCE_WEBHOOKS、 DEPRECATED_SCRIPT_TAG_SCOPE	EOL_API_VERSION
依存関係	—	DEPENDENCY_REACHABILITY	MISSING_DEPENDENCY_SECURITY_AUTOMATION

出力を是正順に並べる方法

公式の重大度だけで並べると、High が25件あるため順番が決まらない。次の3段階で並べ替える。

段階	基準	理由
第1段	「他テナントのデータに到達できる」指摘を最優先。認可・テナント分離の観点と、 APP_PROXY_UNVERIFIED_SIGNATURE、 UNAUTHENTICATED_ENDPOINT	PCD の文脈で最も影響が大きいのは、ストアAの利用者がストアBの顧客データを読める状態。1件で全加盟店に影響する
第2段	シークレットと資格情報、次にインジェクション	流出したシークレットは全ストアのアクセストークン取得に使える。インジェクションは影響範囲が1ストア～全体まで幅があるので個別に判定
第3段	設定・プラットフォーム要件、依存関係	修正が機械的で工数が読みやすい。審査で止まる原因にはなるが、データ流出に直結するものは少ない

同じ段の中では、到達可能性（ステージングで再現できるか）と、修正の波及範囲（共通ミドルウェアで直せるか、個別ルートを全部直すか）で決める。ツールの指摘には誤検知も含まれるため、各指摘に「再現した／誤検知と判断した理由／該当なし」のいずれかを記録してから着手する。

ツールが扱わない確認7つ

shopify app security check はソースコードを見る。次の7つは、コードだけでは判定できないか、コードの外（AWS、ログ、運用）にあるため、人が確認する必要がある。各項目に、確認方法とコード上の見どころを付けた。コード例は架空のものだが、挙動は正確に書いた。

マルチテナント分離（ツールの指摘の先を実機で確かめる）

ツールは「where 句に shop がないクエリ」を指摘する。ツールが見られないのは、shop がどこから来たか、そして DynamoDB のキー設計でストアをまたいだ読み取りが物理的に可能かどうかである。

確認項目	方法
shop の出所	認証済みセッション（ID トークンの dest ）由来か。 req.query.shop 、 req.body.shop をそのまま使っていないか
DynamoDB のキー設計	パーティションキーに shop が含まれるか。 Scan に依存していないか。 GSI にも shop が入っているか
実機確認	開発ストアAのセッションで、ストアBのリソース ID を指定して 404 または 403 になるか。 HTTP ログを証跡として保存

```
// 見どころ：shop をリクエスト入力から取っている（不備）
const shop = req.query.shop as string;
await ddb.send(new GetCommand({ TableName, Key: { pk: `SHOP#${shop}`, sk: `ORDER#${id}` } }));

// 修正：検証済みトークンの dest から取る
const { dest } = await verifySessionToken(req.headers.authorization);
const shop = new URL(dest).hostname;
await ddb.send(new GetCommand({ TableName, Key: { pk: `SHOP#${shop}`, sk: `ORDER#${id}` } }));
```

Webhook の冪等性と重複配信

公式は「Shopify は重複を最小化するが、同じ Webhook を複数回受け取ることがある」「処理を冪等にするのが望ましく、できない場合は X-Shopify-Webhook-Id で重複を除外する」と書いている。また「トピック内でも順序は保証されない」と明記し、X-Shopify-Triggered-At ヘッダーまたはペイロードの updated_at で並べることを推奨している。HMAC 検証はツールが見るが、重複と順序はツールの対象外である。

確認項目	方法
重複排除	Webhook-Id を DynamoDB に条件付き書き込み（ attribute_not_exists(pk) ）し、既存なら 200 を返して処理しない。TTL で古い ID を消す
順序	updated_at を比較し、保持している値より古いイベントで上書きしない
応答時間	受信後すぐ 200 を返し、処理は SQS に退避する（公式：接続1秒、全体5秒のタイムアウト。失敗時は4時間で8回再試行、Admin API で設定した購読は8回連続失敗で自動削除）
同一トピック複数購読	Webhook-Id は配信ごとに異なり、X-Shopify-Event-Id が同じ。関連にはイベント ID を使う

```
// 見どころ：条件付き書き込みで重複を弾く
try {
    await ddb.send(new PutCommand({
        TableName: 'webhook-dedupe',
        Item: { pk: webhookId, ttl: Math.floor(Date.now() / 1000) + 86400 * 3 },
        ConditionExpression: 'attribute_not_exists(pk)',
    }));
} catch (e) {
    if ((e as Error).name === 'ConditionalCheckFailedException') return { statu
sCode: 200 }; // 重複
    throw e;
}
await sqs.send(new SendMessageCommand({ QueueUrl, MessageBody: rawBody }));
return { statusCode: 200 };
```

App Proxy 署名の計算方法

ツールは「署名検証がない」ことを指摘するが、「検証はあるが計算が間違っている」ケースは検出しにくい。公式の計算方法は、signature を除く全パラメータを key=value にし、配列値はカンマで結合、ソートして区切り文字なしで連結し、共有シークレットで HMAC-SHA256 を hex で出す。公式は「署名検証はリクエストが改ざんされていないことだけを保証する」「logged_in_customer_id が要求データの所有者と一致することをアプリ自身が確認しなければならない」とも書いている。

確認項目	コード上の見どころ
連結の区切り	& で結合していないか（Webhook や OAuth の hmac と混同しやすい）
空値	logged_in_customer_id= のように値が空のパラメータを落としていないか
複数値	同一キーの複数値をカンマ結合しているか
比較	crypto.timingSafeEqual か。長さ不一致で例外を投げないか
所有者確認	署名が正しくても、logged_in_customer_id が空なら顧客データを返さないか。パスの顧客 ID と一致を見ているか

```
function verifyAppProxy(query: Record<string, string | string[]>, secret: string): boolean {
    const { signature, ...rest } = query;
    const message = Object.keys(rest).sort()
        .map((k) => `${k}=${Array.isArray(rest[k]) ? (rest[k] as string[]).join(',') : rest[k]}`)
        .join(''); // 区切り文字なし
    const digest = crypto.createHmac('sha256', secret).update(message).digest('hex');
    const a = Buffer.from(digest, 'utf8');
    const b = Buffer.from(String(signature ?? ''), 'utf8');
    return a.length === b.length && crypto.timingSafeEqual(a, b);
}
```

セッショントークン (ID トークン) の発行先照合

公式の検証手順は、HS256 でアプリのシークレットによる署名検証、`exp` が未来、`nbf` が過去、`iss` と `dest` のトップレベルドメインが一致、`aud` がアプリのクライアント ID と一致、のすべてを満たすこと。いずれかに失敗したら「ペイロードを破棄し、処理を止め、エラーで応答する」。トークンは発行から1分で期限切れになり、リクエストごとに取得する。

ツールは `jwt.decode` (検証なし) のような明白な不備は見つけられるが、`aud` の照合漏れや、`algorithms` 未指定のライブラリ呼び出しは見落とすことがある。

確認項目	コード上の見どころ
アルゴリズム固定	<code>jwt.verify(token, secret, { algorithms: ['HS256'] })</code> のように固定しているか
<code>aud</code>	<code>payload.aud === SHOPIFY_API_KEY</code> の比較があるか
<code>iss</code> と <code>dest</code>	両者のホストが一致するか。 <code>dest</code> をテナント識別に使っているか
期限	<code>ignoreExpiration: true</code> や独自のゆるい許容がないか
失敗時	401 を返して処理を止めているか。フォールバックでクッキー認証に落ちていないか

```
const payload = jwt.verify(token, SHOPIFY_API_SECRET, { algorithms: ['HS256'], clockTolerance: 5 }) as JwtPayload;
if (payload.aud !== SHOPIFY_API_KEY) throw new Unauthorized('aud mismatch');
if (new URL(payload.iss!).hostname !== new URL(payload.dest!).hostname) throw new Unauthorized('iss/dest mismatch');
const shop = new URL(payload.dest!).hostname; // 以後のテナント識別子
```

オフラインアクセストークンの保管

公式は「アクセストークンとリフレッシュトークンはバックエンドに安全に保存する」「クライアントシークレットとアクセストークンはサーバー側に置く」と書いている。ツールはトークンがログやブラウザに漏れる経路を見るが、保存先の暗号化と鍵管理、バックアップ、ログ基盤への複製は AWS 側の確認になる。

確認項目	方法
保存先の暗号化	DynamoDB のテーブル暗号化 (<code>describe-table</code> の <code>SSEDescription</code>)。KMS カスタマー管理鍵を使うならキーポリシー
鍵の権限	アプリ層で追加暗号化しているなら、復号できる IAM プリンシパルを列挙
複製先	DynamoDB Streams、エクスポート、バックアップ、CloudWatch Logs にトークンが流れていないか
期限付きトークン	公開アプリは2027年1月1日までに期限付きオフライントークンへ移行。セッションストレージが <code>expires</code> 、 <code>refreshToken</code> 、 <code>refreshTokenExpires</code> を持てるか
返却経路	loader / API 応答にセッションオブジェクトをそのまま返していないか

AWS 側 IAM の最小権限

ツールは AWS を一切見ない。Lambda 実行ロールが `dynamodb:*` や `s3:*` を持っていれば、1つのインジェクション脆弱性が全テーブルへの到達に変わる。

確認項目	方法
実行ロールの範囲	関数ごとに必要なアクションとリソース ARN に限定しているか。Resource: "*" を grep する
ワイルドカード	"Action": "dynamodb:*"、"s3:*"、"secretsmanager:*" の有無
環境変数	クライアントシークレットが Lambda の環境変数に平文で入っていないか。Secrets Manager か SSM Parameter Store (SecureString) か
横展開	Webhook 受信 Lambda と管理画面 API の Lambda が同じロールを共有していないか
人のアクセス	本番アカウントへの人の権限は L2-4 の表と一致するか

```
# 見どころ: SAM / CloudFormation のロール定義
Policies:
  - Version: "2012-10-17"
    Statement:
      - Effect: Allow
        Action: [dynamodb:GetItem, dynamodb:Query, dynamodb:PutItem]
        Resource: !GetAtt AppTable.Arn # "*" にしない
      - Effect: Allow
        Action: secretsmanager:GetSecretValue
        Resource: !Ref ShopifySecret
```

ログ内の個人情報

L2-6 は「アクセスログを保持しレビューする」ことを求めるが、ログに氏名・メール・住所がそのまま出ていれば、ログ基盤が新たな保護対象データの保存先になる。ツールはクレデンシャルのログ漏えい (CREDENTIAL_LOG_LEAKAGE) を見るが、個人情報は対象外である。

確認項目	方法
ログ出力の実態	CloudWatch Logs Insights で @message like /@/ (メールの形) や郵便番号のパターンを検索し、件数を記録
構造化ログ	顧客を指す値は Shopify の顧客 ID (数値) だけにし、メール・氏名は出さない方針をロガーで強制
Webhook ペイロード	受信ペイロード全体を console.log していないか。デバッグ時の出力が残っていないか
保持期間	ロググループの保持日数が無期限になっていないか
アクセスログの中身	「誰が (ユーザー ID)、どのストアの、どの顧客 ID を、どの機能で」読んだかが出ていれば L2-6 を満たす材料になる

```
// 見どころ: ペイロード丸ごとの出力 (不備)
console.log('webhook received', JSON.stringify(payload));

// 修正: 識別子だけを構造化して出す
logger.info({ topic, shop, webhookId, customerId: payload.customer?.id }, 'webhook received');
```

自己点検チェックリスト

印刷して使う。判定は「根拠あり／根拠不足／未実装／該当なし」のいずれかを記入し、根拠の所在（ファイル名、CLI 出力名、文書名）を添える。空欄のまま申請や納品に進まない。

要件（16項目）

No.	確認項目	判定	根拠の所在
1	保存する顧客データ属性の一覧があり、各属性に用途がある		
2	プライバシーポリシーの記載と保存属性が一致する		
3	目的外利用を止める設計レビューの記録がある		
4	顧客の同意判断を処理に反映している（または該当なしの判断記録）		
5	データ共有のオプトアウトを反映している（または該当なしの判断記録）		
6	自動化された意思決定の該当有無を判断し記録した		
7	加盟店との DPA があり同意取得のログがある		
8	保持期間が定義され、TTL・ライフサイクル・shop/redact で削除される		
9	保存先と通信路の暗号化設定の出力がある		
L2-1	バックアップとレプリカの暗号化設定の出力がある		
L2-2	本番とテストが別アカウントで、テストに本番データがない		
L2-3	DLP 方針書と対応する設定がある		
L2-4	本番データへの人のアクセス一覧があり、棚卸し記録がある		
L2-5	パスワードポリシーと MFA が AWS・パートナーダッシュボード・GitHub に適用されている		
L2-6	アクセスログを取得し、レビュー記録がある		
L2-7	インシデント対応方針が版管理され、連絡経路が現実と一致する		

実装（Shopify 固有）

No.	確認項目	判定	根拠の所在
S-1	shopify app security check を --include-dir 付きで実行し、全指摘に再現／誤検知／該当なしを記録した		
S-2	ID トークンの署名・exp・nbf・aud・iss / dest を検証し、失敗時に 401 で止める		
S-3	テナント識別子は dest 由来のみ。リクエスト入力の shop を使うクエリがない		
S-4	開発ストアAからストアBのリソースに到達できないことを実機で確認した		
S-5	Webhook は生ボディで HMAC 検証し、timingSafeEqual で比較し、失敗時 401		

No.	確認項目	判定	根拠の所在
S-6	Webhook-Id で重複を除外し、 updated_at で順序を扱い、処理は SQS に退避		
S-7	必須コンプライアンス Webhook 3種を購読し、受信だけでなく削除・提供の実処理がある（App Store 配布アプリは必須。カスタムアプリへの適用は公式には記載なし）		
S-8	App Proxy 署名を公式の計算方法で検証し、 logged_in_customer_id の所有者確認がある		
S-9	shop ホスト名を <code>^[a-zA-Z0-9][a-zA-Z0-9-]*\.myshopify\.com\$</code> で前後アンカー付きで検証		
S-10	アクセススコープが実際のクエリと一致し、write を read 用途で要求していない		
S-11	期限付きオフライントークンへの移行計画がある（公開アプリ、2027-01-01）		
S-12	Web Pixel を使う場合、保護スコープの承認状況と null 処理を確認した		

実装（AWS 側）

No.	確認項目	判定	根拠の所在
A-1	Lambda 実行ロールに Resource: "*" とサービス全体のワイルドカードがない		
A-2	クライアントシークレットが Secrets Manager / SSM SecureString にあり、環境変数平文・リポジトリにない		
A-3	DynamoDB・S3・SQS・CloudWatch Logs の保存時暗号化を出力で確認した		
A-4	CloudTrail のデータイベントを顧客データのテーブル・バケットで有効化した		
A-5	CloudWatch Logs にメール・氏名・住所が出ていないことを検索で確認した		
A-6	ロググループと S3 の保持期間が定義されている		
A-7	本番アカウントへの人のアクセスが IAM Identity Center の限定グループのみで、MFA 必須		

Forest Security について

Forest Security は、株式会社Forest Book が提供する、ソースコードとクラウド設定の両方からセキュリティ上の問題を確認し、直す順番を決め、直すまでの進捗を管理するサービスである。Shopify アプリ（公開・カスタム）と AWS サーバーレスを開発してきた知見を使う。

商品	内容
Forest Security Review	Shopify アプリ1本または AWS アカウント1つを対象に、本書の要件16項目について「根拠あり／根拠不足／未実装」の要件対応表と指摘一覧を作る。資料・権限受領から5営業日

商品	内容
Forest Security Assessment	AWS 1アカウント+アプリ1本を対象に、Shopify 固有、AWS サーバーレス、Web/API、CI/CD の4領域を確認し、技術報告書、リスク台帳、修正チケット案を納める。是正後の再確認1回を含む
Forest Security Care	月額で、AWS 設定差分、依存パッケージ、検出結果のレビューと是正チケットの進捗管理を行い、月次報告書を出す

やらないこと

24時間365日の監視、インシデント発生時の復旧作業やフォレンジック、網羅的なペネトレーションテスト、Shopify プラットフォーム本体への検査、脆弱性が存在しないことの保証、App Store 審査やデータ保護レビューの可否保証、自社が開発した資産に対する独立した第三者としての監査意見。

金額は個別見積。範囲と「含まれないもの」は提案時に書面で示す。

問い合わせ	info@forest-book.jp
サービスサイト（予定）	https://security.forest-book.jp
会社	株式会社Forest Book（Forest Book Inc.） 神奈川県横浜市西区北幸二丁目10番48号 むつみビル3階

出典一覧

すべて 2026年10月22日に直接確認した。

文書	URL
Work with protected customer data	https://shopify.dev/docs/apps/launch/protected-customer-data
App security checks（36項目一覧）	https://shopify.dev/docs/apps/build/security/app-security-checks
Shopify CLI <code>app security check</code> リファレンス	https://shopify.dev/docs/api/shopify-cli/app/app-security-check
Changelog: Build with confidence: Shopify AI Toolkit's security check（2026-10-15）	https://shopify.dev/changelog/build-more-secure-apps-with-shopify-s-ai-toolkit-security-check
Security（アプリの一般的なセキュリティ要件、TLS、開放ポート）	https://shopify.dev/docs/apps/build/security
Subscribe to webhooks（HTTPS 配信、HMAC、再試行、重複）	https://shopify.dev/docs/apps/build/webhooks/subscribe/https
Webhooks overview（順序非保証、照合ジョブ）	https://shopify.dev/docs/apps/build/webhooks

文書	URL
Privacy law compliance（必須コンプライアンス Webhook）	https://shopify.dev/docs/apps/build/compliance/privacy-law-compliance
Session tokens（ID トークン）概要	https://shopify.dev/docs/apps/build/authentication-authorization/session-tokens
Set up session tokens（検証手順）	https://shopify.dev/docs/apps/build/authentication-authorization/session-tokens/set-up-session-tokens
Authorization code grant（OAuth、hmac、state、shop 検証）	https://shopify.dev/docs/apps/build/authentication-authorization/access-tokens/authorization-code-grant
Authenticate app proxy requests	https://shopify.dev/docs/apps/build/online-store/app-proxies/authenticate-app-proxies
Migrate to expiring offline access tokens	https://shopify.dev/docs/apps/build/authentication-authorization/migrate-to-expiring-offline-access-tokens
App requirements checklist	https://shopify.dev/docs/apps/launch/app-requirements-checklist
Changelog: order.statusPageUrl に Level 2 が必要（2024-03-15）	https://shopify.dev/changelog/level-2-protected-customer-data-requirements-are-now-needed-to-access-the-order-statuspageurl-field
Changelog: Customer Account API に PCD アクセスが必要（2024-08-26）	https://shopify.dev/changelog/protected-customer-data-access-required-to-use-the-customer-account-api
Changelog: Web Pixel の PII に保護スコープが必要（2025-11-10 公開、2025-12-10 施行）	https://shopify.dev/changelog/protected-customer-data-scopes-required

本書の AWS に関する記述は AWS の公開ドキュメントに基づく一般的な構成例であり、個々の設定の既定値は利用時点の AWS ドキュメントで確認すること。