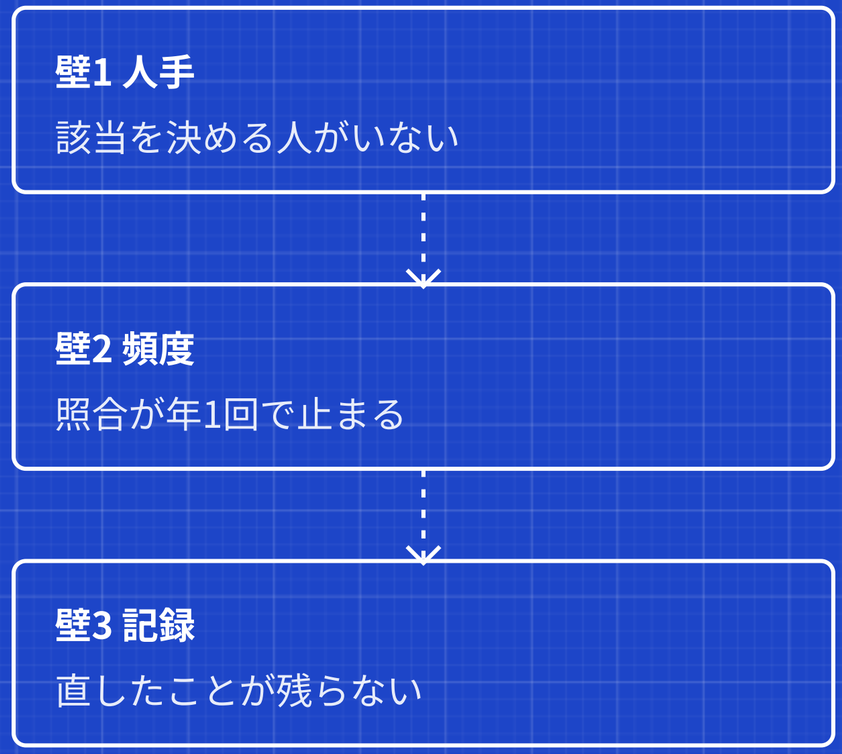
SECURIY DOCS — No.10 ノウハウ

公開情報と自社環境の照合

既知の脆弱性が放置される3つの壁

情報取得 → 照合 → 検証 → 修正 → 適用 → 再検証。人手・頻度・記録の壁と、2系統で考える対策



Securiy

10 PAGES · 2026.10
TEMPURA TECHNOLOGIES

この資料でわかること

脆弱性は公開されています。遅れるのは、自社のどこに該当するかの照合と、修正までの工程です。どこで止まるかを3つの壁に分け、
自社で回すための最小の手順を示します。課題編 P.12 の「3つの壁」を、既知の脆弱性の工程に当てはめたものです。

想定読者

ライブラリの更新が後回しになっている開発責任者

セキュリティ専任がいない SaaS・Fintech・EC 事業者。依存関係の脆弱性通知が溜まっている方。

扱うこと

公開情報の読み方と、照合から再検証までの工程

JVN・GitHub Advisory・OSV・CISA KEV・EPSS の使い分け。人手・頻度・記録の壁と、それぞれの最小の対策。検証が要るとき、要らないとき。

扱わないこと

個別の脆弱性の解説、ツールの設定手順

特定の CVE の詳細や、Dependabot などの設定方法は扱いません。公式ドキュメントをご覧ください。修正・適用で止まる場合（互換性の問題）は別の課題です。

1 どこで止まるか

公開から再検証までの6工程 P.03

公開情報の使い分け: まずはこの5つ P.04

2 3つの壁

壁1 人手: 該当するかを決める人がいない P.05

壁2 頻度: 公開は毎日、照合は年1回 P.06

壁3 記録: 「直した」が残らない P.07

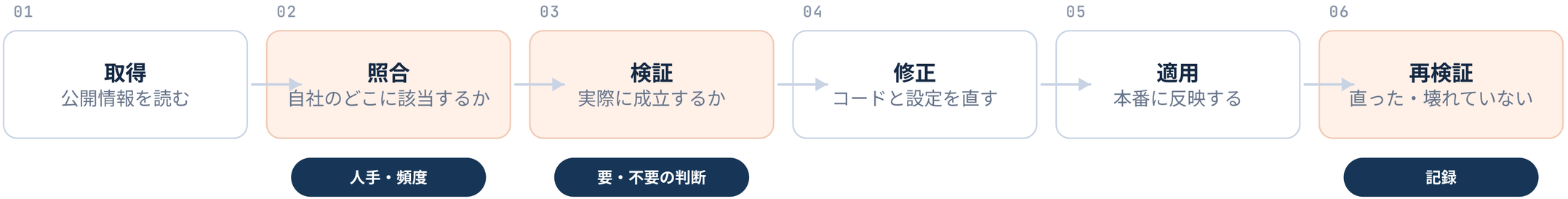
3 回すために

検証が要るとき、要らないとき P.08

点検表 P.09

公開から再検証までの6工程と、3つの壁

脆弱性が公開されてから自社で直し終わるまでは6つの工程です。止まりやすいのは照合・検証・再検証。照合は人手と頻度で、再検証は記録で止まります。検証は、すべての該当に要るわけではありません（P.08）。



止まり方の例

02 照合で止まる	通知は来るが「うちのどこで使っているか」を調べる人がいない。年1回まとめて見るので未読が溜まる。
03 検証で止まる	該当は分かったが、すぐ更新できない。「実際に悪用できるのか」を試す時間がなく、次の年次診断まで待つ。
06 再検証で止まる	ライブラリは上げたが、何を直したか・壊れていないかの記録がなく、取引先に説明できない。

この資料は照合・検証・再検証に絞ります。修正・適用で止まる場合（互換性・保守終了）は別の課題です。

情報は誰でも同じものを読めます。差が付くのは、自社のどこに該当するかを決める速さと、直したことを残す仕組みです。

公開情報の使い分け: まずはこの5つ

読むのは通知だけにして、優先順位は「悪用済みか（KEV）」と「悪用されそうか（EPSS）」で決めます。CVSS（深刻度を 0〜10 で表す点数）は、悪用されているかを表しません。

情報源（すべて無料）	何が分かるか	使いどころ
JVN ／ JVN iPedia	JPCERT/CC と IPA が運営。国内で使われる製品の脆弱性を日本語で公開。JVN iPedia は海外製品（NVD 由来）も日本語で収録	**日本語で概要と対策を確かめたいとき。国内ベンダーの製品を使っているとき**
GitHub Advisory Database	オープンソースのライブラリの脆弱性。Dependabot alerts の元（対象は GitHub がレビューした Advisory）	**リポジトリの依存関係と自動で照合させる（Dependabot alerts を有効にする）**
OSV（osv.dev）	オープンソース全般の脆弱性 DB。API で問い合わせできる	**OSV-Scanner でロックファイル（使っている版を固定した記録）やコンテナを一括で照合するとき**
CISA KEV	米国の政府機関 CISA が公開する、実際に悪用が確認された脆弱性の一覧（対処方法があるものだけ）	**「今週中に直す」の優先順位を決めるとき。載っていない＝悪用されていない、ではない**
EPSS	今後30日間に悪用の試みが観測される確率の推定（FIRST が毎日更新）。悪用が成功する確率ではない	**該当が多いとき、後回しにしてよいものを決める**

優先順位の目安（この資料の案）: KEV に載っている → 今週中。EPSS が高い → 次のリリースで。それ以外 → 期限を決めて定期更新に乗せる。NVD と各ベンダーのアドバイザリは、必要なときに参照します。

壁1 人手:「自社のどこに該当するか」を決める人がいない

情報は公開されています。足りないのは読む人ではなく、自社の構成表と、通知を見て該当・非該当を決める担当です。GitHub を使っていない場合は、OSV-Scanner や Trivy、OWASP Dependency-Check でロックファイルを照合できます。

公開情報

JVN
国内の脆弱性情報

GitHub Advisory
依存ライブラリ

OSV
OSS の脆弱性 DB

CISA KEV
悪用が確認済み

↓

自社の構成表
どのライブラリの、どの版を、どこで使っているか

ここが無いと、通知を読んでも「うちに関係あるか」が決められない

↓

該当 ／ 非該当 ／ 要調査
判断できる

壁1を越える最小の形: 構成表は機械に出させ、人は「該当するか」の判断だけをする。

自分でやるなら: 構成表を機械に出させる

人が一覧を作ると続きません。ビルドの仕組みにすでにある依存関係の一覧をそのまま使います。

ロックファイル（使っている版を固定して記録したファイル）
package-lock.json、poetry.lock、Gemfile.lock、go.mod など

リポジトリの通知機能
Dependabot alerts を有効にすると、依存関係と Advisory を自動で照合

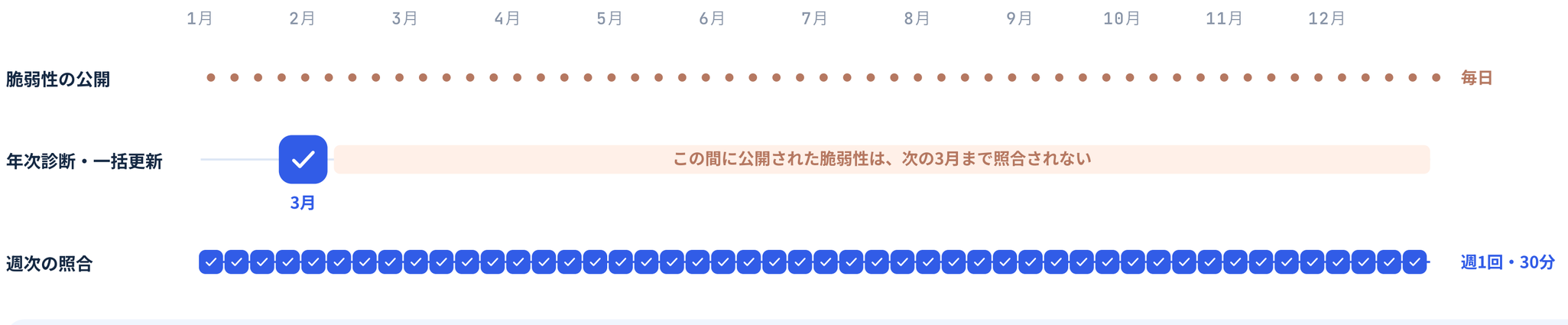
SBOM（部品表。SPDX・CycloneDX などの標準形式）
取引先に「使っているライブラリの一覧」を求められたら、そのまま出せる形

OS・ミドルウェア・コンテナのイメージはロックファイルに出てきません。クラウドやコンテナの脆弱性スキャン（Trivy など）で一覧にし、同じ表に入れます。

担当を決めるのは「一覧を作る人」ではなく「通知を見て、該当・非該当を決める人」。目安は週に1回、30分。

壁2 頻度: 公開は毎日、照合は年1回

年次診断と一括更新だけに頼ると、公開から次の診断までの間が丸ごと「照合されない期間」になります。全部を即日直すのではなく、
見る頻度を上げて、悪用済みから直します。



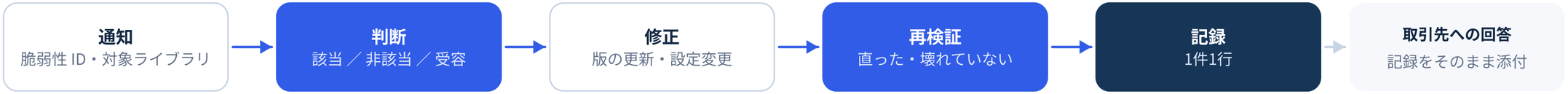
自分でやるなら: 週次30分のルーチン（目安）

1. **通知を開く**
   Dependabot alerts、JVN・KEV の新着
2. **該当・非該当を決める**
   構成表と突き合わせ、1件1行で記録
3. **悪用済みを先に**
   CISA KEV に載っているものは今週中
4. **残りは次回に**
   EPSS が低いものは期限を決めて待つ

公開された脆弱性は、会社の規模に関係なく、該当する構成を自動で探して試されることがあります。見る頻度を上げると、照合されない期間は最長でも1週間に縮みます。

壁3 記録:「直した」が残らない

直すところまではやっているのに、何に対応したか・壊れていないかが残っていない。取引先の「修正済みですか」に根拠つきで答えられず、次の通知でも同じ調査をやり直します。



「直した」が残らない典型

- ライブラリは上げたが、どの脆弱性に対応したのか紐づいていない
- 非該当と判断した理由が残っておらず、次に同じ通知で同じ調査をする
- 上げた後に壊れていないかを確かめた記録がなく、取引先の「修正済みですか」に「はい」としか書けない

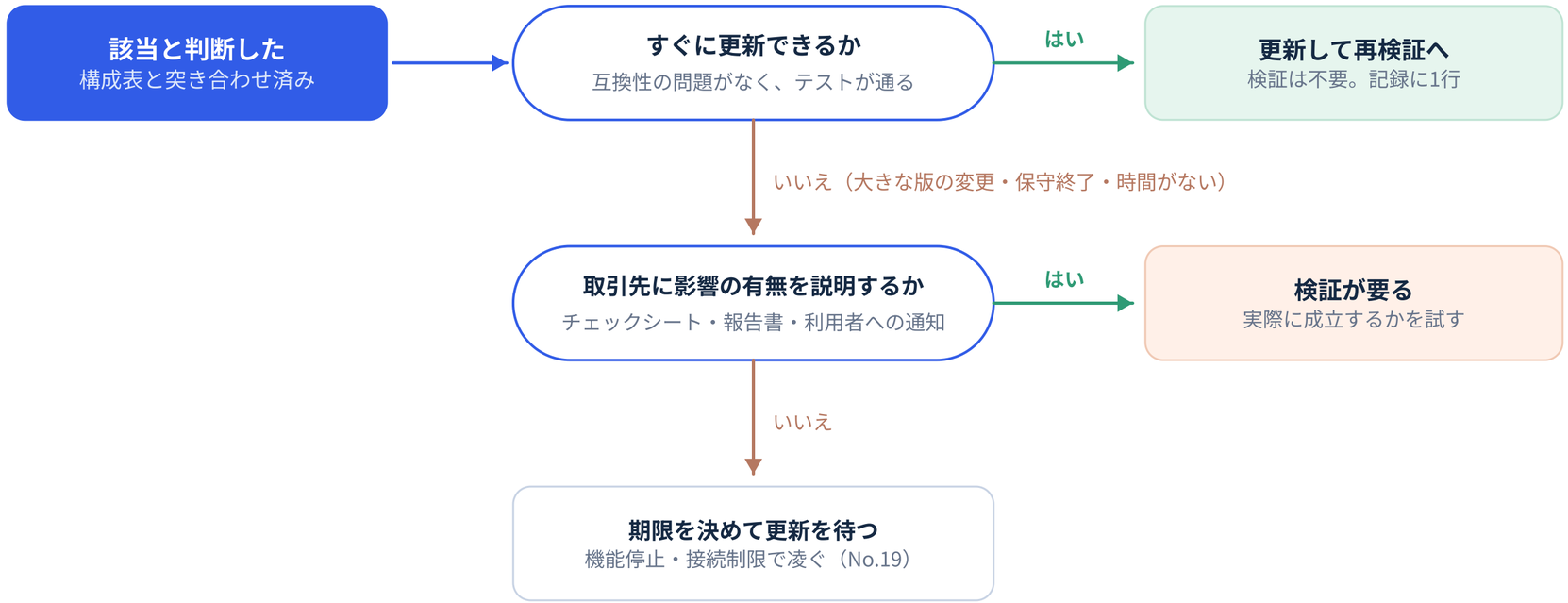
自分でやるなら: 1件1行の判断記録（説明のための架空の例。スプレッドシートで足ります）

日付	脆弱性 ID	該当箇所（ライブラリ・版・利用機能）	判断	根拠	修正日・版	再検証日・結果
○/○	CVE-20XX-XXXXX	lib-foo 1.2.3 ／ CSV 出力で使用	該当	悪用済み（KEV）	○/○ ／ v2.7.1	○/○ ／ 出力正常・拒否確認
○/○	CVE-20XX-YYYYY	lib-bar 0.9 ／ テストでのみ使用	非該当	本番に含まれない	—	—

記録は「直した証明」であると同時に、次回の照合を速くする資産です。非該当の理由こそ残します。

検証が要るとき、要らないとき

「該当したら更新するだけでは？」その通りです。すぐ更新できるなら、検証は要りません。実際に成立するかの検証が要るのは、すぐ更新できないときと、取引先に影響の有無を説明するときです。



「検証」で確かめること

該当するライブラリの機能を、
自社のどの操作が実際に通っているか。
その操作を、公開されている条件で
試したときに成立するか。

成立しなければ「該当するが成立しない」と記録し、期限を決めて更新します。
成立すれば、機能の停止か接続の制限で凌ぎ、アプリ経由なら上限も下げます。

試すのは検証環境。本番の認証情報や実データは使いません。

ここが、自社だけでは人手と時間が足りなくなりやすい工程です。

該当したら全部を検証する必要はありません。すぐ更新できるなら更新する。検証が要るのは、更新を待つ間の影響を知りたいときと、取引先に説明するときです。

点検表: 照合から再検証まで、止まらずに回るか

7つが揃えば、3つの壁は越えられます。全部を一度に整える必要はなく、Q1 と Q2 だけで照合は回り始めます。公開される前の脆弱性には、この資料の工程は効きません。その対策は No.19 で扱います。

No.	点検すること	目安	状態	担当
Q1	**構成表が機械から出る**	ロックファイル・SBOM・コンテナのスキャン。人手で一覧を作っていない	ある ／ 一部 ／ ない	
Q2	**通知を見て該当・非該当を決める担当と頻度が決まっている**	例: 開発リーダーが毎週月曜30分（目安）	ある ／ 一部 ／ ない	
Q3	**悪用済み（KEV）を先に直す優先順位がある**	CVSS の点数だけで並べていない	ある ／ 一部 ／ ない	
Q4	**1件1行の判断記録がある**	日付・ID・該当箇所・判断・根拠・修正日・再検証日	ある ／ 一部 ／ ない	
Q5	**非該当の理由も残している**	次回の照合で同じ調査をしない	ある ／ 一部 ／ ない	
Q6	**修正後に「直った・壊れていない」を確かめた記録がある**	取引先の「修正済みですか」に添付できる	ある ／ 一部 ／ ない	
Q7	**すぐ更新できない該当について、実際に成立するかを確かめた記録がある**	更新を待つ間の影響と、取引先への説明の根拠	ある ／ 一部 ／ ない	

自社で回るもの

Q1〜Q5 は、ロックファイルと通知機能とスプレッドシートで揃います。Q6 も、元の条件で試し直すところまでは自社でできます。ツールの費用は掛かりません。

外に出す価値があるもの

Q7 の「実際に成立するか」を、まだ決めていない条件と経路まで広く試し、試さなかったものも未検証範囲として書いた記録にするところ。Q6 の、元の条件での確認は自社でできます。すぐ更新できない該当を起点に、共同検証プログラムの初回スコープ（2操作）で始められます。

照合は自社で回る。
検証から、一緒に。

構成表・週次の照合・判断記録と、更新後に元の条件で画面と API から試すところまでは、この資料の手順で自社で回せます（No.09 P.09）。外に出す価値があるのは、すぐ更新できない該当が「実際に成立するか」を、まだ決めていない条件と経路（承認内容の書き換え・連携キー・AI 機能）まで広く試し、試さなかったものも未検証範囲として書いた、取引先に出せる記録にするところです。初回のご相談は30分・無料です。

お申し込み・お問い合わせ　**securiy-preview.pages.dev**　のフォームから
Tempura technologies株式会社

※ 本資料は工程の設計を扱うものです。個別の脆弱性の影響判断は、各情報源の一次情報と自社の構成で行ってください。すべての攻撃の網羅や「安全」の総合判定は保証しません。金融の規制適合審査、決済基盤全体の監査、カード不正判定は対象外です。

無料相談

30分無料相談

溜まっている通知の一覧をお持ちください。該当の決め方と、初回に検証する2操作を一緒に決めます。本番の認証情報や顧客データは不要です。

相談を申し込む →

共同検証プログラムとは

検証テーマを一緒に作る企業に、初回スコープで提供する枠組み

1アプリ・1検証環境・2操作、準備完了後10営業日・合計40時間（修正案と再検証は2件・12時間まで）。費用は着手前に提示します。

次に読む

No.19 ゼロデイ前提の不正操作対策
No.11 権限・承認・上限の設計

公開される前の脆弱性でも、アプリの操作を経由する被害を小さくする2系統目の考え方と設計はこちら。