No.16 チェックリスト・全8ページ

公開ページの設定を自分で確認

公開ページ設定チェック10項目



HTTPS、HSTS、CSP、フレーム制御、nosniff、Cookie 属性、
HTTP リソース参照ほか。確かめ方・設定例・書き込み欄つき

Securiy

CHECKLIST

この10項目で分かること、分からないこと

公開ページの応答だけで確認できる設定を10項目にしました。手元のブラウザーで確認できます。

ここから先は見えない

この10項目で分かること

ログイン前の、公開ページの基本設定

接続の設定
HTTPS・転送・HSTS

ブラウザーへの指示
CSP・フレーム制御・nosniff

公開応答の手がかり
Cookie 属性・HTTP リソース参照（＋手元で2項目）

確認するのはトップページ1枚の公開応答だけ。
攻撃や負荷のかかる診断は行いません。
確かめ方は P.04、設定例は P.05。

この10項目では分からないこと

ログイン後の権限、顧客データ、AI 攻撃への耐性

ログイン後の権限
他組織のデータが見えないか、承認を迂回できないか

顧客データへのアクセス
ID の書き換え、一括出力、検索 API からの取得

AI 攻撃への耐性
AI エージェントが経路を探して繰り返す攻撃に、成立する操作があるか

この10項目がすべて揃っていても「安全」とは言えません。
確かめたのは公開応答の設定だけです。

10項目の意味: 何を見て、未設定だと何が起きるか

「未設定＝即事故」ではなく、他の対策と合わせて判断します。09・10 は手元で追加確認する項目です。

No.	項目	何を見るか	未設定だと何が起きるか
接続の設定			
01	**HTTPS 接続**	ページが HTTPS で配信されているか	通信が盗み見られ、改ざんされるおそれがある
02	**HTTP → HTTPS 転送**	http:// で来た要求を https:// に転送するか	古いリンクやブックマークから平文のまま接続される
03	**HSTS**	常時 HTTPS をブラウザーに指示しているか	http:// で入るたびに最初の要求が平文で流れ、HTTPS への切り替えを妨害されうる（初回アクセスは HSTS でも守れず、preload 登録で補う）
ブラウザーへの指示			
04	**CSP**	スクリプト等の読み込み元を制限しているか	スクリプトを差し込まれたとき、実行を抑える仕組みがない
05	**フレーム制御**	他サイトへの埋め込みを制限しているか（X-Frame-Options か CSP の frame-ancestors）	攻撃者のページに見えない状態で埋め込まれ、意図しないボタンを押させられる
06	**nosniff**	ファイル種別の推測をブラウザーに禁止しているか	アップロードされたファイルなどが、宣言と違う種類（HTML やスクリプト）として扱われる余地が残る
公開応答の手がかり			
07	**Cookie 属性**	Secure ／ HttpOnly ／ SameSite が付いているか	Secure なし: http 接続時に平文で送られる。HttpOnly なし: スクリプトから読める。SameSite なし: 他サイトからの要求にも付く
08	**HTTP リソース参照**	ページ内に http:// の画像・スクリプトが残っていないか	スクリプトや iframe はブロックされ画面や機能が壊れる。画像・動画は https に自動で切り替わり、なければ表示されない
手元で追加確認			
09	**Referrer-Policy**	参照元 URL をどこまで外部に送るか	既定値より緩い値（unsafe-url など）だと、URL に含めた ID やトークンが外部に送られる。既定値に頼らず明示する
10	**サーバー情報の露出**	応答にサーバー・フレームワークの版が出ていないか	既知の脆弱性の当たりを付けやすくなる（隠しても脆弱性はなくならない。更新が本筋）

確かめ方: 3分でできる

自社サイトのトップページを Chrome で開き、開発者ツールで応答ヘッダーを見ます。ターミナルが使えるなら curl 1行でも同じものが取れます。

01
Chrome でトップページを開き、F12（Mac は option + command + I）

02
［ネットワーク］タブを開いたまま、ページを再読み込み

03
一覧の一番上（ドキュメント）を選び、［ヘッダー］→「レスポンス ヘッダー」を見る

04
右の名前があるか、値は何かを P.07 のシートに写す

ターミナルなら

```
curl -sI https://自社のドメイン/ | sort
```

-I で応答ヘッダーだけを取得します。HTTP → HTTPS 転送は http:// に対して同じ要求を送り、301 と Location を見ます。自社が管理するサイトなら、MDN HTTP Observatory や securityheaders.com でも同じ項目を確認できます。

```
// レスポンス ヘッダーで見る名前（例）
strict-transport-security: max-age=31536000; includeSubDomains   → 03 HSTS
content-security-policy: default-src 'self'; ...                 → 04 CSP
x-frame-options: DENY                                            → 05 フレーム制御
x-content-type-options: nosniff                                  → 06 nosniff
set-cookie: session=...; Secure; HttpOnly; SameSite=Lax          → 07 Cookie 属性
referrer-policy: strict-origin-when-cross-origin                 → 09 Referrer-Policy
server: nginx/1.25.3                                             → 10 版が出ている
x-powered-by: Express                                            → 10 版が出ている
```

01・02・08 はヘッダー以外で見る

01 は URL が https:// で鍵マークが出るか。02 は http:// で開いて https:// に変わるか。08 は［コンソール］に Mixed Content の警告が出ないか。

設定例と、どこで設定するか

「改善提案」の欄に書く内容の例です。値は一般的な推奨で、自社の構成に合わせて調整します。CDN や WAF（境界で攻撃の通信を見張る仕組み）の側で付ける構成もあります。

項目	設定例	どこで設定するか	注意
HTTP → HTTPS 転送	`301` で `https://` の同じパスへ	CDN・ロードバランサー・Web サーバー	全ページ共通。http のまま残すページを作らない
HSTS	`max-age=31536000; includeSubDomains`	Web サーバー・CDN	全サブドメインが HTTPS になってから includeSubDomains を付ける。短い max-age から始める
CSP	まず `Content-Security-Policy-Report-Only` で様子を見る	Web サーバー・フレームワーク	いきなり本番で有効にすると画面が壊れる。報告だけ受ける期間を置く
フレーム制御	`X-Frame-Options: DENY` ／ `frame-ancestors 'none'`	Web サーバー・フレームワーク	自社の別サイトに埋め込む場合は SAMEORIGIN か許可する origin を書く
nosniff	`X-Content-Type-Options: nosniff`	Web サーバー・CDN	ファイル配信の Content-Type が正しいことを先に確認する
Cookie 属性	`Secure; HttpOnly; SameSite=Lax`	フレームワークのセッション設定	外部サイトからの遷移でログインを維持したい場合だけ SameSite を見直す
HTTP リソース参照	`src・href` の `http://` を `https://` に	HTML・テンプレート・CMS	外部の画像・スクリプトは https 版があるか確認する
Referrer-Policy	`strict-origin-when-cross-origin`	Web サーバー・フレームワーク	URL に ID やトークンを含める設計を先にやめる
サーバー情報の露出	`Server` ヘッダーを消すか一般名に。`X-Powered-By` を消す	Web サーバー・フレームワーク	隠すより更新。版を上げる運用を決める

自社で回るもの

P.04 の確かめ方と、この設定例で10項目を直すところまでは自社でできます。

外に出す価値があるもの

ログイン後の操作のうち、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）を広く試し、未検証範囲まで書いた記録にするところ。

結果の読み方: 4つの状態

結果は「安全／危険」ではなく4つの状態で読みます。未観測は脆弱性ありを意味せず、確認済みも値が妥当とは限りません。安全性の総合点は付けません。

```
strict-transport-security: max-age=31536000
```

確認済み

設定があり、値も妥当。値を記録して終わり。

```
set-cookie: session=...; SameSite=None
```

要確認

設定はあるが、値を見直したい。意図どおりかを担当に確認。

```
// content-security-policy が無い
```

未観測

応答に見当たらない。脆弱性ありの意味ではない。meta タグや別ページだけに付いている場合もある。設定を足すか、足さない理由を書く。

```
// ログイン後の応答: 取得していない
```

未確認

範囲外か、応答が取れなかった。防御の成功も脆弱性ありも意味しない。ログイン後の権限はすべてここ。

「未観測」と「未確認」の行が、次の一手の一覧になります。CSP が未観測でも、XSS があることを示すものではありません。

6 ／ 書き込みシート

「未観測」の行が、次に設定するものの一覧になる

URL ごとに1枚。状態に丸を付け、観測した値・改善提案（P.05 から）・担当を書きます。

No.	項目	状態	観測した値・事実	改善提案	担当・期限
01	**HTTPS 接続**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			
02	**HTTP → HTTPS 転送**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			
03	**HSTS**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			
04	**CSP**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			
05	**フレーム制御**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			
06	**nosniff**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			
07	**Cookie 属性**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			
08	**HTTP リソース参照**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			
09	**Referrer-Policy**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			
10	**サーバー情報の露出**	確認済み ／ 要確認 ／ 未観測 ／ 未確認			

対象 URL: _____　確認日: ____／____／____　確認者: _____

公開ページの次は、ログイン後の権限を。

10項目は、P.04 の手順で自社で確かめられます。ログイン後の操作も、決めた条件で画面と API から試すところまでは自社でできます（No.09 P.09）。外に出す価値があるのは、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）を広く洗い出して試し、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。アクセス制御の不備は OWASP Top 10 でも最上位のリスクで、公開応答では見えません。

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

※ 本シートは公開応答の基本設定を確認するもので、侵入テストや認可の評価とは異なります。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

「未観測」「未確認」の行をお持ちください。ログイン後に確かめる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 →

無料URLチェック（体験版）

8項目を URL ひとつで確認する予定

現在は入力画面と結果のサンプルを公開中で、実チェックは準備中です。
公開までは P.04 の手順で手元で確認できます。

次に読む

課題編「AI攻撃時代のセキュリティ対策のススメ」

不正操作が成立する4つの理由と、空いている工程はこちら。