MEDI-GROUND

SAMPLE / 見本

WEB SECURITY ASSESSMENT REPORT

Webセキュリティ点検 報告書

あおば整骨院 会員予約システム

対象	**https://stg.aoba-reserve.example（検証環境）**
点検期間	**2026年10月6日 〜 10月8日**
点検内容	**外部点検 ＋ ソースコード点検（オプション）**
報告書番号	**MG-SR-SAMPLE-001**
報告日	**2026年10月10日**

> 本書は、点検でお渡しする報告書の**見本**です。点検の対象は、当社が検証用に構築した**架空のシステム**であり、実在する事業者・人物とは関係ありません。掲載している指摘は、すべて当該システムで実際に再現・確認したものです。

株式会社Medi-Ground
〒150-0044 東京都渋谷区円山町5番3号 MIENUX渋谷ビル8階
https://www.medi-ground.net

CONFIDENTIAL — 本書には貴社システムの弱点に関する情報が含まれます。取り扱いにご注意ください。

01 点検結果

総合判定

要緊急対応

公開前の修正を強く推奨

ログインしていない第三者が、全会員の氏名・電話番号・住所と、施術メモ（持病・手術歴など）を見られる状態です。
施術メモの病歴は、法律上とくに慎重な扱いが求められる**「要配慮個人情報」**です。
漏えいした場合は、**1件でも国（個人情報保護委員会）への報告と、ご本人への通知が義務**になります。

点検項目	判定	ひとことで言うと	詳細
会員情報ののぞき見防止	🔴 要対応	ログインしていない人や他の会員から、個人情報が見えてしまう	A-1, A-2, B-2
入力欄からの不正操作の防止	🔴 要対応	検索欄の作りに不備があり、他の会員の予約が見えてしまう	A-3, B-4
秘密の鍵・パスワードの管理	🔴 要対応	決済サービスの鍵が、誰でも見られる場所に置かれている	A-4, B-1
ログインまわりの守り	🟡 要注意	パスワードを何度間違えてもロックされない	A-5, A-7, B-3
他のサイトからの悪用防止	🟡 要注意	外部サイトとの連携の設定が緩すぎる	A-6
エラー表示・ブラウザの防御設定	🟡 要注意	内部の情報が画面に出てしまう、防御設定が入っていない	A-8, A-9
使っている部品の更新状況	🟢 良好	既知の弱点がある部品は使われていない	—
ログイン状態の乗っ取り防止	🟢 良好	他人のログイン状態を推測して奪うことはできない	—

「詳細」の番号は、第2部（技術詳細）の指摘番号です。

02 実際の画面でご確認ください

重大 ログインしていない状態で、全会員の個人情報が表示される

stg.aoba-reserve.example/api/admin/members　ログインしていません

プリティ プリント


```
[
  {
    "id": 1,
    "name": "管理 太郎",
    "email": "admin@aoba.example",
    "tel": "03-0000-0000",
    "birthday": "1980-01-01",
    "address": "東京都千代田区1-1-1"
  },
  {
    "id": 2,
    "name": "テスト 一郎",
    "email": "test1@aoba.example",
    "tel": "090-1111-1111",
    "birthday": "1990-04-01",
    "address": "東京都渋谷区2-2-2"
  },
  {
    "id": 3,
```

会員5名全員の**氏名・メールアドレス・電話番号・生年月日・住所**が、ログインなしで表示されました（赤枠は当社の注記）。同じように、施術メモ（持病・手術歴）を含む全予約も表示されます。

重大 施術メモ（病歴）も、ログインなしで表示される

stg.aoba-reserve.example/api/admin/reservations　ログインしていません

プリティ プリント


```
  },
  {
    "id": 3,
    "member_id": 4,
    "store": "渋谷店",
    "menu": "整体60分",
    "reserved_at": "2026-10-22 11:00",
    "memo": "腰痛・右膝の手術歴あり",
    "name": "山田 花子"
  },
  {
    "id": 4,
    "member_id": 5,
    "store": "大宮店",
    "menu": "マッサージ30分",
    "reserved_at": "2026-10-23 18:00",
    "memo": "持病：高血圧",
```


「腰痛・右膝の手術歴あり」「持病：高血圧」など、**要配慮個人情報にあたる内容**が会員名と一緒に表示されます。

重大　修正すれば、確実に止められます

修正前

.../api/admin/members

プリティ プリント


```
[
  {
    "id": 1,
    "name": "管理 太郎",
    "email": "admin@aoba.example",
    "tel": "03-0000-0000",
    "birthday": "1980-01-01",
    "address": "東京都千代田区1-1-1"
  },
  {
```


ログインなしで全会員の情報が表示される

修正後

.../api/admin/members

プリティ プリント


```
{
  "error": "ログインしてください"
}
```


「ログインしてください」と表示され、情報は出ない

重大　会員が、他の会員の個人情報を見られる

点検用に頂いた「テスト一郎」さんのアカウントでログインし、もう一つの点検用アカウント「テスト二郎」さんの情報が見えるかを確認しました。

修正前

.../api/members/3　テスト一郎でログイン中

プリティ プリント


```
{
  "id": 3,
  "name": "テスト 二郎",
  "email": "test2@aoba.example",
  "tel": "090-2222-2222",
  "birthday": "1992-08-15",
  "address": "神奈川県横浜市3-3-3"
}
```


他人（テスト二郎さん）の住所や電話番号が表示される

修正後

.../api/members/3　テスト一郎でログイン中

プリティ プリント


```
{
  "error": "この情報を表示する権限がありません"
}
```


「この情報を表示する権限がありません」と表示される

高 決済サービスとメール配信サービスの鍵が、誰でも見られる

stg.aoba-reserve.example/js/config.js　ログインしていません

```
window.APP_CONFIG = {
  apiBase: '/api',
  // 予約完了メール送信用
  mailApiKey: 'SG.SAMPLE-DO-NOT-USE.k3Jd9fLq2xVb7NwPz8RtYh4MuCe6',
  // 決済
  paymentSecretKey: 'sk_live_SAMPLE_DO_NOT_USE_51Hx9aBcD2eF',
};
```

決済の鍵があると、貴社のアカウントで**返金の実行**や**顧客の決済情報の参照**ができてしまいます。

メール配信の鍵があると、**貴社のドメインから会員に、なりすましメールを送れて**しまいます。

03 もし悪用されたら — 貴社に起きること

要配慮個人情報が漏えいした場合に、法律上求められる対応の一般的な流れです。技術的な修正とは別に、これだけの対応が経営の側に発生します。

発覚

会員からの問い合わせや外部からの指摘で発覚

「身に覚えのないメールが届いた」などの連絡が、きっかけになることが多くあります。原因が分かるまで、予約サイトを止めるかの判断を迫られます。

3〜5日以内

個人情報保護委員会への報告（速報）

要配慮個人情報を含むため、**漏えいが1件でも報告が義務**です。

速やかに

会員ご本人への通知と、問い合わせ窓口の設置

どの会員の、どの情報が漏れたのかを特定する必要があります。アクセスの記録が残っていなければ、**全会員の全情報が漏れた前提**での対応になります。

並行して

原因の調査と修正

外部の専門家に調査を依頼するケースも少なくありません。

30日以内
（不正アクセスの場合は60日以内）

個人情報保護委員会への報告（確報）

原因・影響の範囲・再発防止策をまとめて報告します。

その後

再発防止策の実施と、信頼の回復

会員や取引先への説明、口コミや評判への対応が続きます。

※個人情報保護法と、個人情報保護委員会のガイドラインに基づく一般的な流れです。個別の事案での対応は、専門家にご確認ください。

まず対応していただきたいこと

1. **決済サービスとメール配信サービスの鍵を、いますぐ無効化して再発行する**
   鍵はすでに公開ページから見える状態です。修正より先に、鍵そのものを使えなくする必要があります。

2. **管理用のデータに、ログインと管理者権限の確認を追加する**
   修正が終わるまでの間は、社内からしか見られないように制限するだけでも、被害を止められます。

3. **他の会員の情報が見える不具合と、検索欄の不具合を修正する**
   いずれも数時間程度で修正できる規模です。修正後に再点検を行い、解消を確認します。

04 点検の概要

対象システム	あおば整骨院 会員予約システム（会員向けWeb予約・マイページ・管理画面）
対象環境	検証環境 https://stg.aoba-reserve.example 本番環境の会員データには一切触れていません。検証環境は、本番と同じプログラムで動いていることを事前に確認しています。
ご提供いただいたもの	点検用の会員アカウント2つ（テスト一郎・テスト二郎）／ソースコード一式（オプションの点検用）
点検期間	2026年10月6日〜10月8日（平日10時〜18時）
実施方法	自動点検ツールとAIによる解析で、確認すべき箇所を洗い出しました。そのうえで、エンジニアが一つずつ手作業で再現し、誤検知を除いています。本書に載せている指摘は、すべて実際に再現できたものです。
事前の合意	点検の範囲・期間・実施する作業を書面で取り交わしてから実施しています。サービスを止めるような負荷のかかる試験は行っていません。

点検した範囲

基本 外部点検

利用者と同じ立場で、ブラウザや通信を通して確認します。ソースコードは使いません。

- ログインしていない人が、見えてはいけない情報を見られないか
- ある会員が、他の会員の情報を見たり操作したりできないか
- 入力欄に細工した文字を入れて、想定外の動きをしないか
- ページの中に、秘密にすべき情報が含まれていないか
- ログイン周り（試行回数の制限、Cookieの設定）
- 通信やサーバーの設定（セキュリティ関連の設定項目）

オプション ソースコード点検

お預かりしたプログラムを読み、外から見えない部分を確認します。

- パスワードの保存方法
- 権限を確認する仕組みが、すべての画面・APIに行き渡っているか
- 外部点検で見つかった問題と同じ書き方が、他の箇所に残っていないか
- ログイン状態の管理（有効期限など）
- 利用している外部部品（ライブラリ）に、既知の弱点がないか

点検の範囲外：サーバーやネットワーク機器そのものの設定、社内PC・メール環境、負荷試験、ソーシャルエンジニアリング（なりすまし電話など）。

指摘事項の一覧

番号	重大度	内容	対応の目安
A-1	重大	ログインしていない人が、全会員の個人情報と施術メモを取得できる	小（数時間）
A-2	重大	会員が、URLの数字を変えるだけで他の会員の個人情報を見られる	小（数時間）
A-3	重大	予約検索の入力欄から、他の会員の予約情報を引き出せる（SQLインジェクション）	小（数時間）

A-4	高	決済・メール配信サービスの秘密鍵が、公開ページから読み取れる	中（1〜3日）
A-5	中	パスワードを何度間違えてもロックされない	小（数時間）
A-6	中	どのWebサイトからでも、ログイン中の会員として情報を読み出せる設定になっている	小（数時間）
A-7	低	ログイン状態を保持するCookieの保護設定が不足している	小（数時間）
A-8	低	エラー時に、プログラムの内部情報が画面に表示される	小（数時間）
A-9	情報	ブラウザ側の防御機能を有効にする設定が入っていない	小（数時間）
B-1	高	パスワードが、復元されやすい方式で保存されている	中（1〜3日）
B-2	中	権限を確認する仕組みが、APIごとの書き忘れに頼っている（A-1・A-2の根本原因）	中（1〜3日）
B-3	中	ログイン状態に有効期限がない	小（数時間）
B-4	低	入力値をそのままSQLに埋め込む書き方の残存箇所（全件確認の結果、A-3の1箇所のみ）	—（A-3で対応）

A：外部点検で見つかったもの／B：ソースコード点検で見つかったもの。「対応の目安」は修正作業そのもののおおよその規模です。

重大度は、次の基準で判定しています。重大 個人情報の大量漏えいや金銭被害に直結し、容易に悪用できる　高 深刻な被害につながるが、悪用に条件がある　中 他の弱点と組み合わさると被害が拡大する　低 単独での被害は限定的　情報 対策を推奨する事項

05 指摘事項の詳細 — 外部点検

基本 **外部点検** 利用者と同じ立場で、ブラウザと通信から確認

A-1 ログインしていない人が、全会員の個人情報と施術メモを取得できる 重大

何が起きるか

管理画面が表示に使っているデータの取得先（API）に、**ログインの確認がありません。**管理画面のページ自体は「ログインしていなければトップへ戻す」作りになっていますが、それはブラウザ上の見た目の制御にすぎず、データの取得先には誰でも直接アクセスできます。

取得できる情報：全会員の**氏名・メールアドレス・電話番号・生年月日・住所**、全予約の**施術メモ（持病・手術歴など）**。

確認した内容

ログインしていない状態で、次のURLにアクセスしました。

```
# ログインしていない状態（Cookieなし）でアクセス
GET /api/admin/members        → 200 OK 全会員5件を取得
  { "name": "管理 太郎", "email": "admin@aoba.example", "tel": "03-0000-0000",
    "birthday": "1980-01-01", "address": "東京都千代田区1-1-1" }, …

GET /api/admin/reservations   → 200 OK 全予約を取得
  { "name": "山田 花子", "store": "渋谷店", "memo": "腰痛・右膝の手術歴あり" },
  { "name": "佐藤 健",   "store": "大宮店", "memo": "持病：高血圧" }
```

※管理画面のURL（/admin/）は推測しやすく、ページを開くと、データの取得先URLがそのまま読み取れます。

対応方法

1. 管理用APIのすべてで、**ログインしているか**と**管理者権限を持っているか**を、サーバー側で確認する。
2. 修正までの応急処置として、管理用API（/api/admin/）へのアクセスを、社内のIPアドレスからに限定する。
3. アクセスログを確認し、過去に外部からこのAPIへのアクセスがなかったかを調べる（漏えいの有無の判断に必要です）。

A-2 会員が、URLの数字を変えるだけで他の会員の個人情報を見られる 重大

何が起きるか

マイページは、会員番号を含むURL（/api/members/**2**）から本人の情報を取得しています。サーバーは「ログインしているか」は確認していますが、**「その番号が本人のものか」を確認していません。**番号を1つずつ変えていけば、ログインできる人なら誰でも、全会員の情報を順番に取得できます。

確認した内容

点検用アカウント「テスト一郎」でログインし、もう一つの点検用アカウント「テスト二郎」の情報を取得できるかを確認しました。

```
# テスト一郎（会員番号 2）でログインした状態
GET /api/members/2   → 200 OK  { "name": "テスト 一郎", … }  ← 本人。正常
GET /api/members/3   → 200 OK  { "name": "テスト 二郎", "tel": "090-2222-2222",
                                 "address": "神奈川県横浜市3-3-3", … }  ← 他人の情報が返る
```

※点検は、ご提供いただいた2つのアカウントの間だけで行い、実際の会員の情報は取得していません。

対応方法

サーバー側で、**要求された会員番号が、ログイン中の本人のものか**を照合する。マイページは、URLで番号を受け取らず、ログイン情報から本人を特定する作りに改めるのが確実です。

A-3　予約検索の入力欄から、他の会員の予約情報を引き出せる（SQLインジェクション）　重大

何が起きるか

予約履歴の「店舗名で絞り込み」に入力した文字が、データベースへの命令文にそのまま組み込まれています。**入力欄に命令文の一部を書き込むと、本来の絞り込み条件を外せてしまいます。**今回は、他の会員の予約がすべて表示されることを確認しました。同じ手口を応用すると、会員テーブル（パスワード情報を含む）など、データベース全体の読み出しにつながるおそれがあります。

確認した内容

```
store=渋谷              → 200 OK  本人の予約 1件  ← 正常
store='                 → 500 エラー  "unrecognized token"  ← 記号1文字で命令文が壊れる
store=' OR 1=1 --       → 200 OK  全会員の予約 4件  ← 絞り込み条件が無効になる
```

※影響を確かめるのに必要な最小限の操作にとどめ、データの書き換え・削除にあたる操作は行っていません。

対応方法

入力値を命令文に直接つなげず、**プレースホルダ（入力値を「値」としてだけ扱う仕組み）**を使う。同じシステム内のログインや予約登録の処理ではすでにこの方式が使われているため、それに揃える修正です。

A-4　決済・メール配信サービスの秘密鍵が、公開ページから読み取れる　高

何が起きるか

トップページが読み込む設定ファイルに、**決済サービスの秘密鍵**と**メール配信サービスの鍵**が書かれています。このファイルは誰でも閲覧できます。

決済の秘密鍵があれば、貴社のアカウントで**返金の実行や、顧客の決済情報の参照**ができてしまいます。メール配信の鍵があれば、**貴社のドメインから会員になりすましメールを送れます**（フィッシング詐欺に悪用されやすい手口です）。

確認した内容

```
GET /js/config.js   → 200 OK（誰でも閲覧可能）
  mailApiKey:       'SG.SAMPLE-…(以下略)',     // 予約完了メール送信用
  paymentSecretKey: 'sk_live_SAMPLE_…(以下略)', // 決済（本番用の秘密鍵の形式）
```

※鍵が実際に使えるかどうかの確認（外部サービスへの送信）は行っていません。鍵の形式から、本番用の秘密鍵と判断しています。

対応方法

1. **最優先：**決済サービスとメール配信サービスの管理画面で、該当の鍵を**無効化して再発行**する。ページを修正しても、すでに読み取られた鍵は使えてしまうためです。
2. 新しい鍵はサーバー側だけで保持し、決済やメール送信はサーバーから行う作りに改める。
3. 各サービスの利用履歴を確認し、心当たりのない返金やメール送信がないかを調べる。

A-5　パスワードを何度間違えてもロックされない　中

何が起きるか

ログインに何回失敗しても制限がかからないため、よく使われるパスワードを機械的に次々と試す攻撃を止められません。

確認した内容

```
誤ったパスワードで30回連続ログイン → 30回とも 401（失敗）。待ち時間・ロックなし
31回目に正しいパスワード           → 200 OK（ログイン成功）
```

対応方法

同じアカウント・同じ接続元からの連続失敗に、待ち時間や一時的なロックを設ける。管理者アカウントには、二段階認証の導入を推奨します。

A-6　どのWebサイトからでも、ログイン中の会員として情報を読み出せる設定になっている　中

何が起きるか

他のWebサイトから貴社のAPIを呼び出してよいかを決める設定（CORS）が、**「どのサイトからでも、ログイン情報付きで許可する」**状態です。会員が悪意のあるサイトを開いただけで、そのサイトが裏側で会員の情報を読み出し、外部に送ることができます。

確認した内容

```
Origin: https://evil.example（無関係なサイトを想定）からの要求に対して
  Access-Control-Allow-Origin:      https://evil.example   ← 要求元をそのまま許可
  Access-Control-Allow-Credentials: true                  ← ログイン情報付きでも許可
```

対応方法

許可する接続元を、貴社のサイト（https://aoba-reserve.example など）だけに限定する。

A-7　ログイン状態を保持するCookieの保護設定が不足している　低

何が起きるか

ログイン状態を表すCookieに、盗まれにくくするための3つの設定がありません。他の弱点と組み合わさると、ログイン状態を乗っ取られる危険が高まります。

確認した内容

```
Set-Cookie: sid=f814c1b6…; Path=/
            HttpOnly なし（プログラムから読み取れる）／Secure なし（暗号化されない通信でも送られる）／SameSite なし
```

対応方法

Cookieの発行時に `HttpOnly`・`Secure`・`SameSite=Lax` を付ける。

A-8　エラー時に、プログラムの内部情報が画面に表示される　低

何が起きるか

エラーが起きると、プログラムのファイルの置き場所や、使っている部品の名前が表示されます。攻撃者が次の手を考えるための手がかりになります（A-3の発見にも、この表示が役立ちました）。

確認した内容

```
500 エラー時の応答
  "stack": "Error: unrecognized token … at /var/www/aoba-reserve/routes/reservations.js:9:6
            at Layer.handle … (node_modules/express/lib/router/layer.js:95:5)"
```

対応方法

利用者に返すエラーは「エラーが発生しました」などの定型文にとどめ、詳細はサーバーのログにだけ残す。

A-9　ブラウザ側の防御機能を有効にする設定が入っていない　情報

確認した内容

```
Content-Security-Policy / X-Frame-Options / X-Content-Type-Options /
Strict-Transport-Security / Referrer-Policy   → いずれも未設定
X-Powered-By: Express                         → 使用している部品名を表示
```

対応方法

上記の設定を追加し、X-Powered-By は表示しないようにする。どれも設定の追加だけで済み、ブラウザが標準で備える防御機能を有効にできます。

06 指摘事項の詳細 — ソースコード点検

オプション **ソースコード点検** お預かりしたプログラムから、外では見えない部分を確認

外部点検は「実際に入れてしまう穴」を見つけるのが得意ですが、**外からは見えない保存方法や、見つかった穴と同じ書き方が他にも残っていないか**は確認できません。ソースコード点検では、プログラムを読んでその部分を補います。

B-1 パスワードが、復元されやすい方式で保存されている 高

何が起きるか

パスワードは、**MD5**という古い方式で変換して保存されています。この方式は非常に高速に計算できるため、データベースが流出すると、**短いパスワードや単純なパスワードの大半は、短時間で元に戻されてしまいます。**同じパスワードを他のサービスでも使っている会員は、そちらのアカウントまで乗っ取られるおそれがあります。
A-3の弱点からデータベースが読み出される可能性があるため、この問題は実際の被害に直結します。

該当箇所

routes/auth.js

```
  7  router.post('/login', (req, res) => {
  8    const hash = crypto.createHash('md5').update(password || '').digest('hex');
  9    const m = db.prepare('SELECT * FROM members WHERE email = ? AND password =
?').get(email, hash);
```

対応方法

パスワード保存専用の方式（bcrypt や Argon2）に切り替える。既存の会員には、次回ログイン時に新しい方式で保存し直す移行処理を入れることで、パスワードの再設定をお願いせずに切り替えられます。

```
const ok = await bcrypt.compare(password, m.password_hash);  // 修正例
```

B-2 権限を確認する仕組みが、APIごとの書き忘れに頼っている（A-1・A-2の根本原因） 中

何が起きるか

ログインや権限の確認は、APIを1つ作るたびに個別に書き足す作りです。管理者かどうかを確認する処理は、システムのどこにも存在しません。A-1とA-2は、この**「書き忘れても動いてしまう」構造**から生まれています。A-1とA-2の箇所だけを直しても、今後APIを追加するたびに同じ問題が起きる可能性があります。

該当箇所

```
routes/admin.js
  4  // 管理画面（/admin/）から呼ばれる
  5  router.get('/members', (req, res) => {        ← ログイン確認・管理者確認なし
routes/members.js
  6  router.get('/:id', requireLogin, (req, res) => {
  7    … WHERE id = ?').get(req.params.id);        ← 本人の番号かを確認していない
```

対応方法

「/api/admin/ 以下は、管理者でなければ一律で拒否する」のように、**入口でまとめて確認する仕組みを作る。**個々のAPIに書き忘れがあっても、安全側に倒れる構造にします。

B-3　ログイン状態に有効期限がない　中

何が起きるか

一度ログインすると、その状態がサーバーを再起動するまで無期限に有効です。ログアウトの機能もありません。共用PCでログインしたままにしたり、Cookieが盗まれたりした場合に、いつまでも悪用され続けます。

該当箇所

```
routes/session.js
  7    sessions.set(sid, { id: member.id, role: member.role });  ← 期限の情報を持たない
```

対応方法

ログイン状態に有効期限（例：最後の操作から2時間）を設け、ログアウト機能を追加する。

B-4　入力値をそのままSQLに埋め込む書き方の残存箇所　低

確認した内容

A-3と同じ書き方が他の箇所に残っていないかを、**データベースを操作する全6箇所**で確認しました。該当したのは**A-3の1箇所のみ**で、他の5箇所は、プレースホルダを使うか入力値を使わない、安全な書き方になっていました。

```
routes/reservations.js
  9    .prepare(`… WHERE member_id = ${req.user.id} AND store LIKE '%${store}%'`)
```

対応方法

A-3の修正で解消します。外部点検だけでは「他にもあるかもしれない」という不安が残りますが、全箇所を確認したことで、修正範囲がここだけだと確定できました。

外部点検とソースコード点検で、見つかったものの違い

確認したこと	外部点検	ソースコード点検
実際に情報を取り出せる穴（A-1〜A-4）	**発見**	**発見・原因箇所を特定**
パスワードの保存方法（B-1）	外からは見えない	**発見**
同じ問題が今後も起きる構造か（B-2）	外からは見えない	**発見**
同じ書き方が他に残っていないか（B-4）	見つけた箇所のみ	**全箇所を確認**

07 良好だった点

指摘事項だけでなく、適切に作られていた点もご報告します。今後の改修で、これらの作りを崩さないようご注意ください。

- ✓ **ログイン状態を表す値は、推測できない十分な長さの乱数**で作られています。他人のログイン状態を推測して乗っ取ることはできません。
- ✓ **ログインと予約登録の処理は、安全な書き方（プレースホルダ）**で作られています。B-4のとおり、問題があったのは1箇所だけです。
- ✓ **利用している外部部品（ライブラリ）に、既知の弱点はありません**（点検日時点で、公開されている脆弱性情報と照合しました）。
- ✓ ログインに失敗したときのメッセージが「メールアドレスまたはパスワードが違います」となっており、**登録済みのメールアドレスかどうかを第三者に推測させない**作りになっています。

08 今後の進め方のご提案

今日〜3日以内

被害を止める

- 秘密鍵の無効化と再発行（A-4）
- 管理用APIの接続元を限定（A-1の応急処置）
- アクセスログと各サービスの利用履歴の確認

1〜2週間以内

穴をふさぐ

- 重大・高の指摘の修正（A-1〜A-4、B-1）
- 権限確認の仕組みを入口に集約（B-2）
- 修正後の再点検

1か月以内

守りを固める

- 中・低・情報の指摘への対応
- 管理者アカウントの二段階認証
- 定期点検の仕組みづくり

修正後の再点検について

指摘事項を修正されたあと、同じ手順で再度確認し、すべて解消されたことを確認する「再点検」を承ります。修正作業そのものも、当社でお引き受けできます。

継続的な保守について

今回の問題の多くは、公開後に誰も点検しないまま運用が続いたことで残っていたものです。機能の追加や部品の更新のたびに、新しい弱点は生まれます。当社では、定期的な点検と部品の更新を含めた保守運用を承っています。

本書の取り扱いと免責事項

- 本書は、点検期間中の対象環境の状態に基づいて作成しています。点検後のプログラムや設定の変更による影響は含みません。
- 点検は、限られた期間と合意した範囲で行うものです。本書に記載のない弱点が存在しないことを保証するものではありません。
- 本書には、貴社システムを攻撃する手がかりとなる情報が含まれます。社外への共有は、修正を依頼する開発会社など必要な範囲にとどめてください。
- 点検で取得したデータは、報告書の作成後に当社の環境から削除します。

株式会社Medi-Ground | https://www.medi-ground.net | j-funayama@medi-ground.net