セキュリティチェックシート

はじめに

本チェックシートは、株式会社SOBAプロジェクトが開発するサービスについて安全・信頼性に係る情報を記載したものです。
本チェックシートの項目は、IPAの「安全なウェブサイトの作り方」を参考に策定しています。
サーバは主にGoogle Compute EngineならびにGoogle Cloud SQLを利用しております。
本チェックシートは、改善のために予告なく変更することがあります。
情報開示の年月日（西暦）2022年4月29日

1) SQLインジェクション対策

1-1) SQL文の組み立ては全てプレースホルダで実装しているか

> 原則的にRuby on RailsのO/Rマッパーであ るActiveRecordを用いておりSQL文は生成しません。例外的にSQLを生成する場合においても、全てプレースホルダを用います。

1-2) SQL文の構成を文字列連結により行う場合は、アプリケーションの変数をSQL文のリテラルとして正しく構成する

> 原則的にRuby on RailsのO/Rマッパーであ るActiveRecordを用いておりSQL文は生成しません。例外的にSQLを生成する場合においても、全てプレースホルダを用います。

1-3) ウェブアプリケーションに渡されるパラメータにSQL文を直接指定しない

> パラメータでSQL文を受け渡す事はありません。

1-4) エラーメッセージをそのままブラウザに表示しない

> エラーメッセージをそのままブラウザに表示することはありません。

1-5) データベースアカウントに適切な権限を与える

> データベースアカウントには必要最低限の権限のみを付与しております。

2) OSコマンド・インジェクション

2-1) シェルを起動できる言語機能の利用を避ける

> 原則的にRuby on Railsからシェルコマンドを起動する事はありません。

2-2) シェルを起動できる言語機能を利用する場合は、その引数を構成する全ての変数に対してチェックを行い、あらかじめ許可した処理のみを実行する

> 例外的にシェルコマンドを実行する必要がある場合があれば、引数を構成する変数に対し正当性のチェックをおこない、想定する動作のみを実行するように実装します。また、アプリケーションの動作に使用するシェルアカウントは一般ユーザでrootへの昇格不可能な様にしています。

3) パス名パラメータの未チェック／ディレクトリ・トラバーサル

3-1) 外部からのパラメータでウェブサーバ内のファイル名を直接指定 する実装を避ける

> 外部からのパラメータでウェブサーバ内のファイル名を指定する事はありません。

3-2) ファイルを開く際は、固定のディレクトリを指定し、かつファイル名にディレクトリ名が含まれないようにする

> ファイルの永続化は可能な場合はクラウドストレージを用います。ホストのファイルシステムにファイルを保存する場合においても、ファイルを開く際はディレクトリを固定し、かつファイル名を外部から指定できない様に実装します。

3-3) ウェブサーバ内のファイルへのアクセス権限の設定を正しく管理する

> アプリケーションの動作に使用するシェルアカウントは一般ユーザでrootへの昇格不可能な様にしています。ホストのファイルシステムにファイルを保存する場合においても、そのユーザの権限内でアクセスできる領域にのみ保存されます。

3-4) ファイル名のチェックを行う

> 外部からのパラメータでウェブサーバ内のファイル名を指定する事はありません。

4) セッション管理の不備

4-1) セッションIDを推測が困難なものにする

> セッションIDには推測が困難な文字列が払い出されます。

4-2) セッションIDをURLパラメータに格納しない

> セッションIDをURLパラメータとして受け渡す事はありません。セッションにはCookieを用います。

4-3) HTTPS通信で利用するCookieにはsecure属性を加える

> HTTPによる通信は原則行いませんが、セッションで用いるCookieにはsecure属性を付与してます。

4-4) ログイン成功後に、新しくセッションを開始する

> ユーザーのログイン・ログアウト時にセッションが自動的に切れるようになっております。

4-5) ログイン成功後に、既存のセッションIDとは別に秘密情報を発行し、ページの遷移ごとにその値を確認する

> ユーザーのログイン・ログアウト時にセッションが自動的に切れるようになっております。

4-6) セッションIDを固定値にしない

> ユーザーのログイン毎に新たなセッションIDが発行されます。

4-7) セッションIDをCookieにセットする場合、有効期限の設定に注意する

> ユーザー認証の有効期限は2週間となっております。

5) クロスサイト・スクリプティング

HTMLテキストの入力を許可しない場合の対策

5-1) ウェブページに出力する全ての要素に対して、エスケープ処理を施す

> 原則的に画面に表示する要素へのエスケープ処理が実施します。

5-2) URLを出力するときは、「http://」や「https://」で始まるURLのみを許可する

> URLを出力するときは、「http://」や「https://」で始まるURLのみを出力します。

5-3) 要素の内容を動的に生成しない

> 一部の例外(SOBA Framework Cloud利用時)を除き、scriptタグを動的に生成しません。

5-4) スタイルシートを任意のサイトから取り込めるようにしない

> スタイルシートを外部から指定できる様にはしておりません。

5-5) 入力値の内容チェックを行う

HTMLテキストの入力を許可する場合の対策

5-6) 入力されたHTMLテキストから構文解析木を作成し、スクリプトを含まない必要な要素のみを抽出する

> 入力されたHTMLテキストから構文解析木を作成し、スクリプトを含まない必要な要素のみを抽出する事は行っておりません。

5-7) 入力されたHTMLテキストから、スクリプトに該当する文字列を排除する

> 入力されたHテキストは適切にバリデーションを実施します。

全てのウェブアプリケーションに共通の対策

5-8) HTTPレスポンスヘッダのContent-Typeフィールドに文字コード(charset)の指定を行う

> HTTPレスポンスヘッダのContent-Typeフィールドに文字コードの指定を行います(text/html; charset=utf-8)。

5-9) Cookie情報の漏えい対策として、発行するCookieにHttpOnly属性を加え、TRACEメソッドを無効化する

> 発行するCookieにはHttpOnly属性を付記します。また、Webサーバで TRACEメソッドを受け付けません。

5-10) クロスサイト・スクリプティングの潜在的な脆弱性対策として有効なブラウザの機能を有効にするレスポンスヘッダを返す

> レスポンスヘッダで「X-XSS-Protection: 1; mode=block」を返します。

6) CSRF (クロスサイト・リクエスト・フォージェリ)

6-1) 処理を実行するページを**POST**メソッドでアクセスするようにし、その「**hidden**パラメータ」に秘密情報が挿入されるよう、前のページを自動生成して、実行ページではその値が正しい場合のみ処理を実行する

> Ruby on RailsによるCSRF保護機構によりセキュリティトークンがページに埋め込まれます。

6-2) 重要な操作を行った際に、その旨を登録済みのメールアドレスに自動送信する

> アカウント作成時などの重要な操作を行う場合には、登録メールアドレスに通知が送信されます。

7) HTTPヘッダ・インジェクション

7-1) ヘッダの出力を直接行わず、ウェブアプリケーションの実行環境や言語に用意されているヘッダ出力用**API**を使用する

> nginx + ruby on railsの組み合わせにおいてはhttpヘッダ・インジェクションの脅威は極めて低いと思うが、set_headerの使用は原則しない。

7-2) 外部からの入力の全てについて、改行コードを削除する

> nginx + ruby on railsの組み合わせにおいてはhttpヘッダ・インジェクションの脅威は極めて低いと思うが、set_headerの使用は原則しない。

8) メールヘッダ・インジェクション

8-1) メールヘッダを固定値にして、外部からの入力はすべてメール本文に出力する

> メールヘッダの項目を外部からの入力値で作成することはありません。

9) クリックジャッキング

9-1) HTTPレスポンスヘッダに、X-Frame-Optionsヘッダフィールドを出力し、他ドメインのサイトからのframe要素やiframe要素による読み込みを制限する

> HTTPレスポンスヘッダに X-Frame-Options: SAMEORIGINをセットします。

10) バッファオーバーフロー

10-1) 直接メモリにアクセスできない言語で記述する

> ruby on rails ではメモリへの直接アクセスが出来ない様になっています。

11) アクセス制御や認可制御の欠落

11-1) アクセス制御機能による防御措置が必要とされるウェブサイトには、パスワード等の秘密情報の入力を必要とする認証機能を設ける

> 認証機能に加えて認可制御の処理を適切に実装します。

11-2) 認証機能に加えて認可制御の処理を実装し、ログイン中の利用者が他人になりすましてアクセスできないようにする

> 認証機能に加えて認可制御の処理を適切に実装します。