Web システム／Web アプリケーションセキュリティ要件書 2.0

October 2015

1. Web システム／Web アプリケーションセキュリティ要件書について

1.1. 本ドキュメントについて

Web システム／Web アプリケーションセキュリティ要件書（以下、本ドキュメント）は、安全な Web アプリケーションの開発に必要なセキュリティ要件書です。発注者、開発者、テスト実施者、セキュリティ専門家、消費者が活用することで、以下のことを達成することを目的としています。

- 開発会社・開発者に安全な Web システム／Web アプリケーションを開発してもらうこと
- 開発会社と発注者の瑕疵担保契約の責任分解点を明確にすること
- 要求仕様や RFP（提案依頼書）として利用し、要件定義書に組み込むことができるセキュリティ要件として活用していただくこと

1.2. 本ドキュメントがカバーする範囲

本ドキュメントでは Web システム／Web アプリケーションに関して一般的に盛り込むべきだと考えられるセキュリティ要件について記載しています。また、開発言語やフレームワークなどに依存することなくご利用いただけます。ただし、ネットワークやホストレベル、運用などに関するセキュリティ要件については記載していません。

対象とする Web システム／Web アプリケーションは、インターネット・イントラネット問わず公開するシステムで、特定多数または不特定多数のユーザーが利用するシステムを想定しています。この中でも特に認証を必要とするシステムが、本ドキュメントの主なターゲットとなっています。

本ドキュメントは、セキュリティ要件としての利用しやすさを優先して記載しているため、一般的であろうというシステムを想定し、例外の記載を少なくしたセキュリティ要件となっています。そのため具体的な数値や対策を指定していることもありますが、要件定義書に記載する内容は開発者と折衝してください。

2. about OWASP

2.1. OWASP について

The Open Web Application Security Project (OWASP) は、信頼できるアプリケーションの開発・購入・運用の推進を目的として設立されたオープンなコミュニティです。OWASP では、以下をフリーでオープンな形で提供・実施しています。 （https://www.owasp.org/）

- アプリケーションセキュリティに関するツールと規格
- アプリケーションセキュリティ検査、セキュア開発、セキュリティ・コードレビューに関する網羅的な書籍
- 標準のセキュリティ制御とライブラリ
- 世界中の支部
- 先進的な研究
- 世界中での会議
- メーリング・リスト

全ての OWASP のツール、文書、フォーラム、および各支部は、アプリケーションセキュリティの改善に関心を持つ人のため、無料で公開されています。アプリケーションセキュリティに対する最も効果的なアプローチとして、我々は人、プロセス、技術という 3 つの課題から改善することを提唱しています。

OWASP は新しい種類の組織です。商業的な圧力が無い中、アプリケーションセキュリティに対して、偏見無く実用的かつコスト効果の高い情報の提供を行っています。OWASP はいかなる IT 企業の支配下にもありませんが、商用のセキュリティ技術の活用を支持しています。他のオープンソース・ソフトウェアプロジェクトと同様に、OWASP も協同かつオープンな形で多様な資料を作成しています。

The OWASP Foundation は、このプロジェクトの長期的な成功を目指す非営利組織です。OWASP 理事会、グローバル委員会、支部長、プロジェクトリーダー、プロジェクトメンバーを含む、OWASP の関係者はほとんどボランティアです。革新的なセキュリティ研究に対して、助成金とインフラの提供で支援しています。

2.2. OWASP Japan について

主に日本で活動している OWASP メンバーによる日本支部です。OWASP の膨大なドキュメントやツール類の日本語化を初めとして、日本からのセキュリティ情報の発信を行っています。是非ご参加下さい。
https://www.owasp.org/index.php/Japan

3. 要求仕様項目

「※」：必須事項ではないが、あると望ましい要件を表しています。囲み内は解説および補足です。

1. 認証

1.1 以下の箇所では、ユーザー認証を実施すること

- 特定のユーザーのみに表示・実行を許可すべき画面や機能
- 上記画面や機能に含まれる画像やファイルなどの個別のコンテンツ（非公開にすべきデータは直接 URL で指定できる公開ディレクトリに配置しない）
- 管理者用画面

特定のユーザーのみにアクセスを許可したい Web システムでは、ユーザー認証を行う必要があります。
また、ユーザー認証が成功した後にはアクセス権限を確認する必要があります。そのため、認証済みユーザーのみがアクセス可能な箇所を明示しておくことが望ましいでしょう。
リスクベース認証や二要素認証など認証をより強固にする仕組みもあります。

1.2 以下の箇所では、ユーザー認証済みの場合でも再認証を実施すること ※

- 個人情報や機密情報を表示するページに遷移する際
- パスワード変更や決済処理などの重要な機能を実行する際

ユーザー認証はセッションにおいて最初の一度だけ実施するのではなく、重要な情報や機能へアクセスする際には再認証を行うことが望ましいでしょう。

1.3 パスワードについて

- パスワード文字列は少なくとも大小英字と数字の両方を含み、最低 8 文字以上であること
- 画面（hidden フィールドなどの HTML ソース内も含む）にパスワード文字列を表示しないこと
- パスワード文字列の入力フォームは input type="password"で指定すること
- ユーザーが入力したパスワード文字列を次画面以降で表示しないこと
- パスワード文字列は「パスワード文字列＋salt（ユーザー毎に異なるランダムな文字列）」をハッシュ化したものと salt のみを保存すること（salt は 20 文字以上であることが望ましい）
- ユーザー自身がパスワードを変更できる機能を用意すること ※
- 登録可能なパスワード文字列の最大文字数は 127 文字以上であること ※
- 登録可能なパスワード文字列の文字は大小英字、数字、記号が利用可能であること ※

認証を必要とする Web システムの多くは、パスワードを本人確認の手段として認証処理を行います。そのためパスワードを盗聴や盗難などから守ることが重要になります。
パスワード文字列のハッシュ化をさらに安全にする手法としてストレッチングがあります。

1.4 アカウントロック機能について

- 認証時に無効なパスワードで 10 回試行があった場合、最低 30 分間はユーザーがロックアウトされた状態にすること ※
- ロックアウトは自動解除を基本とし、手動での解除は管理者のみ実施可能とすること ※

パスワードに対する総当たり攻撃や辞書攻撃などから守るためには、試行速度を遅らせるアカウントロック機能の実装が有効な手段になります。アカウントロックの試行回数、ロックアウト時間については、サービスの内容に応じて調整することが必要になります。

1.5 パスワードリセット機能について

- パスワードリセットを実行する際にはユーザー本人しか受け取れない連絡先（あらかじめ登録しているメールアドレス、住所、電話番号など）に再設定方法を通知すること
- パスワードはユーザー自身に再設定させること

連絡先については、事前に受け取り確認をしておくことでより安全性を高めることができます

2. 認可（アクセス制御）

2.1 Web ページや機能、データにアクセスする際には認証情報・状態を元に権限があるかどうかを判別すること

認証により何らかの制限を行う場合には、利用しようとしている情報や機能へのアクセス（読み込み・書き込み・実行など）権限を確認することでアクセス制御を行うことが必要になります。
画像やファイルなどのコンテンツに対しても、個別にアクセス権限を設定、確認する必要があります。
これらのアクセス権限の一覧表を作成することが望ましいでしょう。

3. セッション管理

3.1 セッションの破棄について

- 認証済みのセッションが一定時間以上アイドル状態にあるときはセッションタイムアウトとし、サーバー側でセッションを破棄しログアウトすること
- ログアウト機能を用意し、ログアウト実行時にはサーバー側でセッションを破棄すること
- ログアウト機能の実行後にその成否をユーザーが確認できること ※

認証を必要とする Web システムの多くは、認証状態の管理にセッション ID を使ったセッション管理を行います。認証済みの状態にあるセッションを不正に利用されないためには、使われなくなったセッションを破棄する必要があります。セッションタイムアウトの時間については、サービスの内容に応じて設定することが必要になります。

3.2 セッション ID について

- Web アプリケーション開発ツールが提供するセッション管理機能を使用すること
- 上記のセッション管理機能の使用が困難な場合、セッション ID は疑似乱数生成系により 80 ビット（使用する文字が 16 進数の場合 20 文字）以上の文字列を生成すること
- セッション ID をクライアントと受け渡しする際は cookie にのみ格納すること ※
- 認証に使用するセッション ID の発行は認証成功後とすること ※
- セッション管理機能を利用して機密情報を管理する場合、クライアントが送信した機密情報をサーバー側で受け取った時点でセッション ID を発行または再生成すること ※
- 認証済みユーザーの特定はセッション ID でのみ行うこと

セッション ID を用いて認証状態を管理する場合、セッション ID の盗聴や推測、攻撃者が指定したセッション ID を使わされるなどの攻撃から守る必要があります。フレームワークなどの制約によりセッション ID の発行を認証成功後に行うことが困難な場合には、ログイン後にセッション ID を再生成するなどの代替策が必要になります。
また、セッション ID は原則として cookie にのみ格納すべきです。

3.3 CSRF（クロスサイトリクエストフォージェリー）対策の実施について

- ユーザーにとって重要な処理を行う箇所では、ユーザー本人の意図したリクエストであることを確認できるようにすること

正規ユーザー以外の意図により操作されては困る処理を行う箇所では、フォーム生成の際に他者が推測困難なランダムな値（トークン）を hidden フィールドに埋め込み、リクエストを POST メソッドで送信します。フォームデータを処理する際にトークンが正しいことを確認することで、正規ユーザーの意図したリクエストであることを確認することができます。
また、別の方法としてパスワード再入力による再認証を求める方法もあります。

4. パラメーター

4.1 URL パラメーターにユーザーID やパスワードなどの秘密情報を格納しないこと

URL に付加される name=value のパラメーターは、Referrer 情報などにより外部に漏えいする可能性があります。そのため URL パラメーターには秘密にすべき情報は格納しないようにする必要があります。

4.2 パラメーターにパス名を含めないこと

ファイル操作を行う機能などにおいて、URL パラメーターやフォームで指定した値でパス名を指定できるようにした場合、想定していないファイルにアクセスされてしまうなどの不正な操作を実行されてしまう可能性があります。

4.3 入力値の文字種や文字列長の検証を行うこと

入力値に想定している文字種や文字列長以外の値の入力を許してしまう場合、バッファオーバーフローなどの不正な操作を実行されてしまう可能性があります。入力値としてファイルを受け付ける場合には、拡張子やファイルフォーマットなどの検証が必要になります。
サーバー側でパラメーターを受け取る場合、クライアント側での入力値検証の有無に関わらず、入力値の検証はサーバー側で実施する必要があります。

5. 出力処理

5.1 HTML を生成する際の処理について

- HTML として特殊な意味を持つ記号（< > " ' &）を文字参照によりエスケープすること
  （ ' のエスケープはオプション扱い）
- 外部から入力した URL を出力するときは「http://」または「https://」で始まるもののみを許可すること
- <script>...</script>要素の内容やイベントハンドラ（onmouseover=””など）を動的に生成しないようにすること ※
- スタイルシートを外部サイトから取り込めないようにすること

外部からの入力により不正な HTML タグなどが挿入されてしまう可能性があります。「<」→「<」や「&」→「&」、「"」→「"」のようにエスケープを行う必要があります。スクリプトによりクライアント側で HTML を生成する場合も、同等の処理が必要です。実装の際にはこれらを自動的に実行するフレームワークやライブラリを使用することが望ましいでしょう。また、その他にもスクリプトの埋め込みの原因となるものを作らないようにする必要があります。
<script>...</script>要素の内容やイベントハンドラは原則として動的に生成しないようにすべきですが、jQuery などの Ajax ライブラリを使用する際はその限りではありません。ライブラリについては、アップデート状況などを調べて信頼できるものを選択するようにしましょう。

5.2 HTML タグの属性値を「"」で囲うこと

HTML タグ中の name=”value”で記される値(value)にユーザーの入力値を使う場合、「”」で囲わない場合、不正な属性値を追加されてしまう可能性があります。

5.3 CSS を動的に生成しないこと

外部からの入力により不正な CSS が挿入されると、ブラウザに表示される画面が変更されたり、スクリプトが埋め込まれる可能性があります。

5.4 HTTP レスポンスヘッダーの Content-Type を適切に指定すること

一部のブラウザではコンテンツの文字コードやメディアタイプを誤認識させることで不正な操作が行える可能性があります。これを防ぐためには、HTTP レスポンスヘッダーを「Content-Type: text/html; charset=utf-8」のように、コンテンツの内容に応じたメディアタイプと文字コードを指定する必要があります。

5.5 HTTP レスポンスヘッダーフィールドの生成時に改行コードが入らないようにすること

HTTP ヘッダーフィールドの生成時にユーザーが指定した値を挿入できる場合、改行コードを入力することで不正な HTTP ヘッダーやコンテンツを挿入されてしまう可能性があります。これを防ぐためには、HTTP ヘッダーフィールドを生成する専用のライブラリなどを使うようにすることが望ましいでしょう。

5.6 SQL 文を組み立てる際に静的プレースホルダを使用すること

SQL 文の組み立て時に不正な SQL 文を挿入されることで、SQL インジェクションを実行されてしまう可能性があります。これを防ぐためには SQL 文を動的に生成せず、プレースホルダを使用して SQL 文を組み立てるようにする必要があります。
静的プレースホルダとは、JIS/ISO の規格で「準備された文(Prepared Statement)」と規定されているものです。

5.7 プログラム上で OS コマンドやアプリケーションなどのコマンド、シェル、eval()などによるコマンドの実行を呼び出して使用しないこと

コマンド実行時にユーザーが指定した値を挿入できる場合、外部から任意のコマンドを実行されてしまう可能性があります。コマンドを呼び出して使用しないことが望ましいでしょう。

5.8 リダイレクタを使用する場合には特定の URL のみに遷移できるようにすること

リダイレクタのパラメーターに任意の URL を指定できる場合（オープンリダイレクタ）、攻撃者が指定した悪意のある URL などに遷移させられる可能性があります。

5.9 メールヘッダーフィールドの生成時に改行コードが入らないようにすること

メールの送信処理にユーザーが指定した値を挿入できる場合、不正なコマンドなどを挿入されてしまう可能性があります。これを防ぐためには、不正な改行コードを使用できないメール送信専用のライブラリなどを使うようにすることが望ましいでしょう。

6. HTTPS

6.1 重要な情報を扱う画面や機能は HTTPS で保護すること

適切に HTTPS を使うことで通信の盗聴・改ざん・なりすましから情報を守ることができます。次のような重要な情報を扱う画面や機能では HTTPS で通信を行う必要があります。

- 入力フォームのある画面
- 入力フォームデータの送信先
- 重要情報が記載されている画面
- セッション ID を送受信する画面

HTTPS の画面内で読み込む画像やスクリプトなどのコンテンツについても HTTPS で保護する必要があります。また、Web サイトを全て HTTPS で保護することでより安全性を高めることができます。

6.2 サーバー証明書はアクセス時に警告が出ないものを使用すること

HTTPS で提供されている Web サイトにアクセスした場合、Web ブラウザから何らかの警告がでるということは、適切に HTTPS が運用されておらず盗聴・改ざん・なりすましから守られていません。適切なサーバー証明書を使用する必要があります。

6.3 安全な暗号化通信を使用すること

- TLS1.0 以上を使用し、SSL2.0／3.0 を無効にすること
- 安全な暗号スイートを使用すること

SSL2.0／3.0 には脆弱性があるので、TLS1.0 以上を使用する必要があります。

7. cookie

7.1 cookie の属性を適切に設定すること

- HTTPS 利用時には Secure 属性を付けること
- HttpOnly 属性を付けること
- Domain 属性を指定しないこと　※

Secure 属性を付けることで、http://へのアクセスの際には cookie を送出しないようにできます。特に認証状態に紐付けられたセッション ID を格納する場合には、Secure 属性を付けることが必要です。
HttpOnly 属性を付けることで、クライアント側のスクリプトから cookie へのアクセスを制限することができます。
セッションフィクセイションなどの攻撃に悪用されることがあるため、Domain 属性は特に必要がない限り指定しないことが望ましいでしょう。

7.2 cookie の値には機密情報を格納しないこと

cookie には name=value として任意の値を格納することができますが、cookie の値はユーザーが参照・改変できる値であるため、アプリケーションの挙動に影響を及ぼすような重要な値を格納するべきではありません。
それらの値はセッション ID などの代替え情報に紐付け、cookie には代替え情報のみを格納して管理することが望ましいでしょう。

8. 画面設計

8.1 ユーザーの Web ブラウザ環境を操作せずデフォルト状態で動作すること

- ユーザーに指示してセキュリティ設定の変更をさせない
- アドレスバーやステータスバーを隠すなど画面の変更を行わない

ユーザーの Web ブラウザのセキュリティ設定などを変更した場合、その変更は他のサイトにも影響します。ユーザーのセキュリティを守るために設定を変更させるべきではありません。また、アドレスバーやステータスバーなどの情報はユーザーがセキュリティの状態を確認するためなどに必要になりますので、隠すなど画面の変更をするべきではありません。

8.2 フレーム、IFRAME を使用しないこと ※

フレームや IFRAME 内に表示されている画面は、ブラウザ上で一目で確認することができません。フレームや IFRAME 内に偽サイトを表示させられるのを防ぐためには使用を避けることが望ましいでしょう。

8.3 レスポンスヘッダーに X-Frame-Options を指定すること

クリックジャッキング攻撃に悪用されることがあるため、X-Frame-Options ヘッダーフィールドに DENY または SAMEORIGIN を指定する必要があります。

9. その他

9.1 エラーメッセージに詳細な内容を表示しないこと

ミドルウェアやデータベースのシステムが出力するエラーには、攻撃のヒントになる情報が含まれているため、エラーメッセージの詳細な内容はエラーログなどに出力するべきです。

9.2 ハッシュ関数、暗号アルゴリズムは『電子政府における調達のために参照すべき暗号のリスト（CRYPTREC 暗号リスト）』に記載のものを使用すること

広く使われているハッシュ関数、疑似乱数生成系、暗号アルゴリズムの中には安全でないものもあります。安全なものを使用するためには、『電子政府における調達のために参照すべき暗号のリスト（CRYPTREC 暗号リスト）』に記載されたものを使用する必要があります。

9.3 鍵や秘密情報などに使用する乱数的性質を持つ値を必要とする場合には、暗号学的な強度を持った疑似乱数生成系を使用すること

鍵や秘密情報に予測可能な乱数を用いると、過去に生成した乱数値から生成する乱数値が予測される可能性があるため、ハッシュ関数などを用いて生成された暗号学的な強度を持った疑似乱数生成系を使用する必要があります。

9.4 公開ディレクトリには公開を前提としたファイルのみ配置すること

公開ディレクトリに配置したファイルは、URL を直接指定することでアクセスされる可能性があります。そのため、機密情報や設定ファイルなどの公開する必要がないファイルは、公開ディレクトリ以外に配置する必要があります。

9.5 基盤ソフトウェアはアプリケーションの稼働年限以上のものを選定すること

脆弱性が発見された場合、修正プログラムを適用しないと悪用される可能性があります。そのため、言語やミドルウェア、ソフトウェアの部品などの基盤ソフトウェアは稼働期間またはサポート期間がアプリケーションの稼働期間以上のものを利用する必要があります。もしアプリケーションの稼働期間中に基盤ソフトウェアの保守期間が終了した場合、危険な脆弱性が残されたままになる可能性があります。

9.6 管理者がアカウントの有効・無効を設定できること ※

不正にアカウントを利用されていた場合に、アカウントを無効化することで被害を軽減することができます。

9.7 重要な処理が行われたらログを記録すること ※

ログは、情報漏えいや不正アクセスなどが発生した際の検知や調査に役立つ可能性があります。認証の失敗やアカウント情報の変更などの重要な処理が実行された場合には、その処理の内容やクライアントの IP アドレスなどをログとして記録することが望ましいでしょう。ログに機密情報が含まれる場合にはログ自体の取り扱いにも注意が必要になります。
また、重要な処理が行われたことをユーザーに通知することによって異常を早期に発見できる可能性があります。

10. 提出物

10.1提出物として次の資料を用意すること

- サイトマップ
- 画面遷移図
- ディレクトリツリー

認証や再認証、CSRF 対策が必要な箇所、アクセス制御が必要なデータ、HTTPS による保護が必要な画面やデータを明確にするためには、Web サイト全体の構成を把握し、扱うデータを把握する必要があります。そのためには上記の資料を用意することが望ましいでしょう。

4. 参考文献

- 独立行政法人 情報処理推進機構
  『安全なウェブサイトの作り方　改訂第 7 版』
  ➢ http://www.ipa.go.jp/security/vuln/websecurity.html
- 株式会社トライコーダ
  『セキュア Web アプリケーション開発講座』
  ➢ http://www.tricorder.jp/education.html
  - 書籍『体系的に学ぶ　安全な Web アプリケーションの作り方　脆弱性が生まれる原理と対策の実践（ソフトバンククリエイティブ）』徳丸浩 著
  - 総務省「電子政府における調達のために参照すべき暗号のリスト（CRYPTREC 暗号リスト）
  ➢ http://www.soumu.go.jp/menu_news/s-news/01ryutsu03_02000038.html

制作：OWASP Japan セキュリティ要件定義書 WG
上野　宣（OWASP Japan リーダー／ 株式会社トライコーダ）：WG リーダー
池田　雅一（テクマトリックス株式会社）
岩井　基晴（株式会社リクルートジョブズ）
小河　哲之（三井物産セキュアディレクション株式会社）
亀田　勇歩（SCSK 株式会社）
国分　裕（三井物産セキュアディレクション株式会社）
洲崎　俊（ソフトバンク株式会社）
野口　睦夫（NEC ソリューションイノベータ株式会社）

協力：HASH コンサルティング株式会社、株式会社トライコーダ

5. 改訂履歴

株式会社トライコーダ版
Ver.1.0　　初版（2009 年 3 月 17 日）
Ver.1.1　　2009 年 4 月 22 日改訂

OWASP 版
Ver.1.0　　初版（2013 年 11 月 1 日）
Ver.2.0　　2015 年 10 月 13 日改訂