

# 発注者のための Web システム/Web アプリケーション セキュリティ要件書



このドキュメントは、[クリエイティブコモンズ・ライセンス](#)の下でライセンスされています。

※下記のクレジットを明記のうえ、二次加工・配布、商用利用をすることができます。

(C)Some rights Reserved:2010 脆弱性診断.jp(株式会社アイアクト) / 株式会社トライコーダ

## 目次

1. 本ドキュメントの目的.....	3
2. 本ドキュメントの対象読者 .....	3
3. 本ドキュメントがカバーする範囲 .....	3
4. 要求仕様項目 .....	4
5. 参考文献 .....	11
6. 改訂履歴 .....	11

## 1. 本ドキュメントの目的

本ドキュメントは、以下の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

## 2. 本ドキュメントの対象読者

本ドキュメントは、以下の方を対象読者としています。

- 発注者の方で要求仕様や RFP（提案依頼書）を作成する方
- 受注者（開発者）の方で要件定義書を作成する方

## 3. 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のしやすさを優先して記載しているため、一般的であろ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受注者（開発者）と折衝してください。

## 4. 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。

<b>1. 認証</b>
<b>1.1 以下の箇所では、認証を実施すること</b> <ul style="list-style-type: none"><li>● 認証済みユーザーのみに表示・実行を許可すべき画面や機能</li><li>● 上記画面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)</li><li>● 管理者用画面</li></ul>
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、認証を行う必要があります。また、認証が成功した後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。
<b>1.2 以下の箇所では、認証済みの場合でも再認証を実施すること ※</b> <ul style="list-style-type: none"><li>● 個人情報や機密情報を表示するページに遷移する際</li><li>● 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</li></ul>
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。
<b>1.3 パスワードについて</b> <ul style="list-style-type: none"><li>● パスワード文字列は英字と数字の両方を含み、最低 7 文字以上であること</li><li>● 画面 (hidden パラメーターなどのソースコード内も含む) にパスワード文字列を表示しないこと</li><li>● 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</li><li>● 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</li><li>● 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</li><li>● 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 ※</li></ul>
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うこととなります。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。
<b>1.4 アカウントロック機能について</b> <ul style="list-style-type: none"><li>● 認証時に無効なパスワードで 10 回試行があった場合、最小 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 ※</li><li>● ロックアウトは自動解除を基本とし、手動でのロックアウトの解除は管理者のみ実施可能とすること ※</li></ul>
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ことが有効な手段になります。アカウントロック機能を実装することが望ましいです。

よう。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。

## 2. 認可（アクセス制御）

### 2.1 Web ページや機能、データへのアクセスは認証情報・状態を元にして判別すること

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・書き・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。Web ページや機能、データにアクセスする際には認証情報・状態を元にして権限があるかどうかを判別する必要があります。

## 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（時間はサービスの内容に応じて調整すること）
- 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

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うこととなります。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ようにするためには、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調整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の体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 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 文字列はログインごとに乱数により生成すること
- 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 ※
- セッション ID の発行は認証成功後とすること ※
- 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 を盗聴や推測されたり、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するのが困難な場合にはログイン後にセッション ID を振り直すなどの代替案を取り入れましょう。

セッション ID は原則として Cookie にのみ格納すべきですが、Cookie の利用に制限がある携帯電話用サイトを構築する場合、セッション ID を POST アクセスの hidden パラ

メーターに格納、または URL に埋め込むことで対応します。

### 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

- CSRF 対策を実施すべき箇所を特定すること
- CSRF 対策を実施すべき箇所では再認証の実施すること
- CSRF 対策を実施すべき箇所では再認証の実施を提供することが困難な場合にはアクセスを POST に限定し秘密情報の埋め込みと確認を実施すること

利用者本人以外の意図により操作されては困る箇所が CSRF 対策を実施すべき箇所となります。再認証を実施することが望ましい対策ですが、利便性低下を回避したい場合などには、サーバーが発行した秘密情報の埋め込みと確認を実施することで正しい画面遷移を経てきているかを確認する必要があります。

## 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、Referer 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 アプリケーション要件に基づいて入力値の文字種や文字列長の検証を行うこと

Web インタフェースを通して受け取る入力値に、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。

## 5. 文字列処理

### 5.1 クロスサイトスクリプティング (XSS) 対策を講じること

- すべての HTML 出力で特殊文字 (<>'&) をエスケープすること  
( ' のエスケープはオプション扱い)
- ユーザーの入力など、外部から入力した URL を出力するときは「http://」や「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

外部からの特殊文字の入力により不正な HTML タグなどが実行されてしまう可能性があります。「<」→「&lt;」や「>」→「&gt;」、「&」→「&amp;」のようにエスケープを行

う必要があります。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)が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 SQL コマンドを組み立てる際に Prepared Statement などバインド機構を使用すること

SQL コマンドの組み立て時に不正な SQL コマンドを挿入されることで、不正な SQL コマンド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、サーバー側で用意されたバインド機構 (Prepared Statement など) を使用して SQL コマンドを組み立てるようにする必要があります。

## 5.4 HTTP レスポンスヘッダーの Content-Type に文字コードを指定すること

一部のブラウザの仕様を悪用してコンテンツの文字コード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に「Content-Type: text/html; charset=utf-8」のように文字コードを指定する必要があります。

# 6. HTTPS

## 6.1 https://で指定すべき画面を特定すること

- 入力フォームのある画面
- 入力フォームデータの送信先
- 重要情報や問い合わせ先など改ざんや偽のページが表示されては困る画面

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。重要な情報を扱う場合には HTTPS で通信を行う必要があります。

## 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 SSL2.0 を無効にすること

SSL2.0 にはセキュリティ上の問題があります。SSL3.0/TLS1.0 などを使用する必要があります。

## 7. Cookie

### 7.1 HTTPS 利用時の Cookie には secure 属性を付けること

Cookie に secure 属性を付けることで、http://へのアクセスの際には Cookie を送出不いようにできます。http://と https://にまたがった Web サイトの場合には、secure 属性がある Cookie とない Cookie の 2 つを使い分ける必要があります。

### 7.2 Cookie の値にはセッション ID 以外を格納しないこと ※

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 の値は盗聴される可能性があります。それらの値はセッション ID に紐付け、Cookie にセッション ID のみを格納して管理することが望ましいでしょう。

### 7.3 Cookie に HTTPOnly 属性を付けること ※

Cookie に HttpOnly 属性を付けることで、一部のブラウザでは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。攻撃による Cookie の盗難を防ぐためには HttpOnly 属性を付けることが望ましいでしょう。

## 8. 画面設計

### 8.1 利用者の 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

利用者の 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。利用者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利用者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。

## 9. その他

### 9.1 エラーメッセージでユーザーIDや登録メールアドレスなどの存在証明をしないこと

たとえば認証失敗時に「パスワードが間違えています」というエラーメッセージを表示した場合、ユーザーIDは合っているということを意味します。攻撃者は、これを利用して存在するユーザーIDを探ることができます。エラーメッセージでユーザーIDや登録メールアドレスなどの存在がわかるようなメッセージを出力すべきではありません。

### 9.2 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。これらの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。

### 9.3 ハッシュ関数、疑似乱数生成系、暗号アルゴリズムは CRYPTREC の電子政府推奨暗号リストに記載のもののみを使用すること

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、CRYPTREC の電子政府推奨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。

### 9.4 暗号処理を実装する場合には CRYPTREC の電子政府推奨暗号リストに記載の暗号アルゴリズムのみを使用すること

暗号アルゴリズムを自作せずとも、必要な暗号アルゴリズムはすでに提供されています。暗号処理を実装する場合には、推奨されている暗号アルゴリズムを使う必要があります。

### 9.5 データファイルを公開ディレクトリに格納しない

データファイルを使ってデータを格納する場合、データファイル自体にアクセスされてしまう可能性があるため、インターネット経由でアクセスできない場所に格納する必要があります。

### 9.6 パラメーターに任意の URL を指定することで自動的に画面を遷移できるオープンリダイレクタを設けないこと

オープンリダイレクタを不正に利用することで、攻撃者が指定した任意の URL に遷移させることができます。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必要があります。

### 9.7 メール送信処理には専用のライブラリを使用すること

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、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。

### 9.8 HTTP ヘッダーフィールドの生成には専用のライブラリを使用すること

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、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。

### 9.9 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

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。

## 10. 提出物

### 10.1 提出物として下記を用意すること

- 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。

## 5. 参考文献

- NPO 日本ネットワークセキュリティ協会  
『「Web システム セキュリティ要求仕様 (RFP)」 編 β 版』  
➤ [http://www.jnsa.org/active/2005/active2005\\_1\\_4a.html](http://www.jnsa.org/active/2005/active2005_1_4a.html)
- 独立行政法人 産業技術総合研究所 情報セキュリティ研究センター 高木浩光  
『安全な Web アプリ開発の鉄則 2006』  
➤ <http://www.nic.ad.jp/ja/materials/iw/2006/proceedings/T21.pdf>
- 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 3 版』  
➤ <http://www.ipa.go.jp/security/vuln/websecurity.html>
- 株式会社トライコーダ  
『安全な Web サイト構築のための設計ガイドライン』  
➤ <http://www.tricorder.jp/education.html>
- 制作：株式会社トライコーダ
- 協力：HASH コンサルティング株式会社

## 6. 改訂履歴

Ver.1.0	初版 (2009 年 3 月 17 日)
Ver.1.1	2009 年 4 月 22 日改訂 「4. 要求仕様項目」の 1.3、3.1、3.2、5.1、8.1 の項目を追記・修正 「4. 要求仕様項目」の 1.1、3.2、8.2 の補足説明を追記・修正
Ver.1.2	2010 年 3 月 2 日改訂 「4. 要求仕様項目」の 6.1 の補足説明を修正