IPA
INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN

安全なウェブサイトの作り方

ウェブアプリケーションのセキュリティ実装と
ウェブサイトの安全性向上のための取り組み

2006 年 1 月 31 日

独立行政法人 情報処理推進機構
セキュリティセンター

目次

目次 1
1. はじめに 1
1.1 はじめに 1
1.2 本資料の内容 1
1.3 問題の解決、対策について 2
2. ウェブアプリケーションのセキュリティ実装 3
2.1 SQL インジェクション 3
2.2 OS コマンド・インジェクション 6
2.3 クロスサイト・スクリプティング 7
2.4 セッション管理の不備 11
2.5 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 13
2.6 メールの第三者中継 15
3. ウェブサイトの安全性向上のための取り組み 17
3.1 ウェブサーバのセキュリティ対策 17
3.2 DNS 情報の設定不備 18
3.3 ネットワーク盗聴への対策 19
3.4 パスワードの不備 21
3.5 フィッシング詐欺を助長しないための対策 22
4. おわりに 24
参考資料 24

1. はじめに

1.1 はじめに

インターネット上では、多くのウェブサイトがそれぞれサービスを提供しています。2005 年度現在、ウェブサイトを開設している企業は約 8 割に達し、ウェブを通じた情報のやりとりやサービスの提供は今後も増え続けることが予想されます。一方、ウェブサイトを悪用した事件も後を絶ちません。最近は営利目的の犯行も目立ち、悪質化が進む傾向にあります。ウェブサイトから個人情報を不正取得される事件も頻発し、その多くでは、ウェブサイトで稼動していたウェブアプリケーションのセキュリティ上の問題が悪用されました。

ウェブアプリケーションは、それぞれのウェブサイトで独自に開発されている場合が多く、セキュリティを考慮した実装はその開発者に委ねられています。ウェブアプリケーションにセキュリティ上の問題が発覚した場合、すでに運用を開始しているウェブアプリケーションを設計レベルから修正することは難しい場合が多く、攻撃を回避するためのその場しのぎの対策で済まさざるを得ないことがあります。ウェブアプリケーションの開発には、開発者がセキュリティを考慮した正しいプログラミング知識を身に付け、開発段階からセキュリティ上の欠陥を作らないための取り組みが必要です。

本資料は、独立行政法人 情報処理推進機構(IPA)が届出[1] を受けたソフトウエア製品およびウェブアプリケーションの脆弱性関連情報を基に、届出件数の多かった脆弱性や攻撃による影響度が大きい脆弱性を取り上げ、その根本的な解決策と、保険的な対策を示しています。また、ウェブサイト全体の安全性を向上するための取り組みについても触れています。

本資料が、ウェブサイトのセキュリティ問題を解決する一助となれば幸いです。

1.2 本資料の内容

本資料は、「2. ウェブアプリケーションのセキュリティ実装」において、ウェブアプリケーションに関連する脆弱性を取り上げ、その根本的な解決策と、保険的な対策を解説しています。また、「3. ウェブサイトの安全性向上のための取り組み」においては、ウェブサーバの運用や通信の暗号化など、ウェブサイト全体の安全性を向上するための取り組みを示しています。

なお、具体的なコーディング例や設定方法などの詳細には触れていません。利用しているウェブアプリケーションの開発言語やウェブサイトの環境に合わせて対応いただくことを想定しています。IPA のウェブサイトで公開している技術情報が参考となる項目については、その URL を掲載しています。また、本資料で示す内容は、あくまで解決策の一例であり、必ずしもこれらの実施を強要するものではありません。

対象読者は、個人や企業を問わず、ウェブアプリケーション開発者やサーバ管理者など、ウェブサイトの運営に関わる方全てとしています。まずは、自身が関わるウェブサイトやウェブアプリケーションに問題がないかを確認し、必要に応じて対処を検討してください。

---

[1] IPA セキュリティセンターでは、経済産業省の告示に基づき、脆弱性情報に関する届出を受け付けています。
■ 脆弱性関連情報に関する届出について
http://www.ipa.go.jp/security/vuln/report/index.html

1.3　問題の解決、対策について

セキュリティ対策は、その内容によって得られる効果が異なります。ある問題への取り組みを考えたとき、問題の原因そのものを取り除く、根本からの解決を目指す方法もあれば、外因である攻撃手法に注目し、その攻撃による影響を低減する対策を施す方法もあります。大切なことは、自分が選択する取り組みが、どのような性質を持っているのか、期待する効果を得られるものなのか、ということを正しく理解することです。

本資料では、特にウェブアプリケーションのセキュリティ実装について、その性質を基に「根本的解決」と「保険的対策」の二つに分類しています。

■ 根本的解決

根本的解決では、「脆弱性の原因を作らない実装」を実現するための取り組みを解説しています。ウェブアプリケーションのセキュリティ対策は、「攻撃を回避する」機能を付加的に実装する傾向がありましたが、最近は開発段階からセキュリティを考慮し、脆弱性の原因を作り込まない取り組みが注目されはじめています。どのような設計において、どのような問題が発生しうるのかを理解し、「問題のある実装」を避け、「安全な実装」を実現してください。

■ 保険的対策

保険的対策では、攻撃による影響を低減する対策等を解説しています。根本的解決との違いは、脆弱性そのものをなくすものではないという点です。根本的解決と併せることで、より高い安全性を確保することが期待できます。時間的制約や運用の事情などにより、根本的解決をすぐに実施できない場合は、この対策がいわば「セーフティネット」になります。ただし、この対策が、結果として特定の文字の取り扱いや本来の機能を制限する場合があるため、この影響を考慮した上で実施を検討する必要があります。

セキュリティを考慮したウェブアプリケーションを開発するためには、正しいプログラミング知識が必要です。ウェブアプリケーション開発者は、本資料とあわせ、次の資料を参照することをお勧めします。

参考 URL	セキュア・プログラミング講座 http://www.ipa.go.jp/security/awareness/vendor/programming/ セキュア・プログラミング講座 「WEB プログラマコース」 http://www.ipa.go.jp/security/awareness/vendor/programming/a00.html

2．ウェブアプリケーションのセキュリティ実装

ここでは、ウェブアプリケーションのセキュリティ実装として、IPA への届出件数の多かった脆弱性や攻撃による影響度が大きい脆弱性を主に取り上げ[2] 、それらに対する根本的解決および保険的対策を示します。

1） SQL インジェクション

2） OS コマンド・インジェクション

3） クロスサイト・スクリプティング

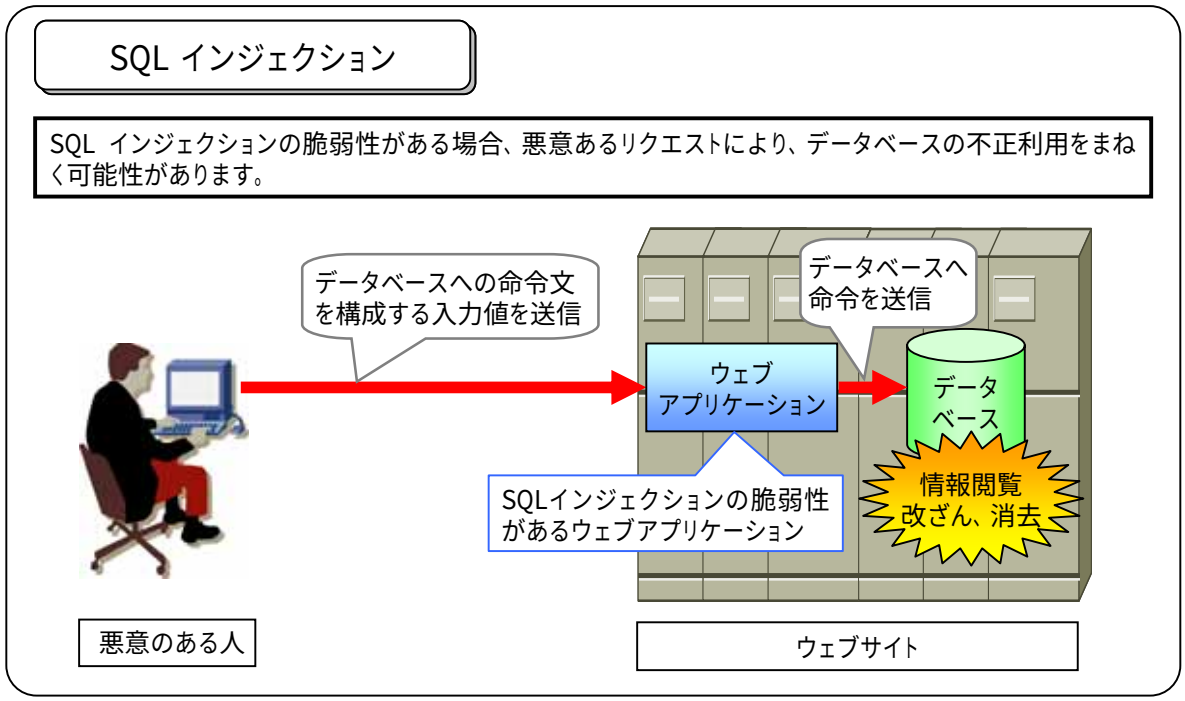
4） セッション管理の不備

5） 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

6） メールの第三者中継

2.1 SQL インジェクション

データベースと連携したウェブアプリケーションの多くは、利用者からの入力情報を基にデータベースへの命令文を組み立てます。ここで、命令文の組み立て方法に問題がある場合、攻撃によってデータベースの不正利用をまねく可能性があります。この問題を悪用した攻撃手法は、一般に「SQL インジェクション」と呼ばれています。



SQL インジェクション攻撃を受けた場合、データベースに格納された情報を不正に取得されたり、改ざんされたりする可能性があります。SQL インジェクションの対策として、次の内容をご検討ください。

[2] 資料の構成上、脆弱性の深刻度や攻撃による影響を考慮して項番を割り当てていますが、これは対策の優先順位を示すものではありません。優先順位は運営するウェブサイトの状況に合わせてご検討ください。

■ 根本的解決

1）SQL 文の組み立てにバインド機構を使用する

解説	これは、SQL 文が混入する原因を作らない実装です。バインド機構とは、実際の値がまだ割り当てられていない記号文字（プレースホルダ）を使用してあらかじめ SQL 文の雛形を用意し、後に実際の値（バインド変数）を割り当てて SQL 文を完成させる、データベースの機能です。バインド変数の値はエスケープ処理されてプレースホルダに渡されるため、利用者に入力された悪意ある SQL 文の実行を防ぐことができます。データベースエンジンや開発環境によっては専用の機能が用意されている場合があるので、この機能の活用をお勧めします。
参考 URL	セキュア DB プログラミング「バインドメカニズムを活用しよう」 http://www.ipa.go.jp/security/awareness/vendor/programming/a02_01.html

2）バインド機構を使用できない場合は、SQL 文を構成する全ての変数に対しエスケープ処理を行う

解説	これは、根本的解決 1）のバインド機構を実装できない場合に実施すべき実装です。利用者から入力されるパラメータや、データベースに格納された情報など、SQL 文を構成する全ての変数や演算結果に対し、エスケープ処理を行ってください。エスケープ処理には、文字の置換（たとえば、「’」→「’’」、「¥」→「¥¥」 など）や関数（たとえば、DBI の quote() など）を利用する方法があります。なお、SQL 文にとって特別な意味を持つ記号文字は、データベースエンジンによって差異があるため、利用しているデータベースエンジンに合わせて対策を行ってください。
参考 URL	セキュア DB プログラミング 「入力文字列はエスケープしよう」「LIKE 句では%と_に気をつけよう」 http://www.ipa.go.jp/security/awareness/vendor/programming/a02_01.html

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

解説	これは、いわば「論外」の実装ですが、フォームの hidden 形式などに SQL 文をそのまま指定するといった問題のあるウェブサイトが少なからず存在するため、避けるべき実装として紹介します。ウェブアプリケーションに渡されるパラメータに SQL 文を直接指定していると、そのパラメータの改変により、データベースの不正利用につながる可能性があります。
参考 URL	セキュア Web プログラミング「hidden は危険」 http://www.ipa.go.jp/security/awareness/vendor/programming/a01_05.html

■ 保険的対策

4）データベースアカウントに適切な権限を与える

解説	これは、SQL インジェクション攻撃による影響を低減するための対策です。ウェブアプリケーションがデータベースに接続する際に使用するアカウントの権限が必要以上に高い場合、攻撃による被害が深刻化する恐れがあります。ウェブアプリケーションからデータベースに渡す命令文を洗い出し、その命令文の実行に必要な最小限の権限をデータベースアカウントに与えてください。
参考 URL	セキュア DB プログラミング「データベースとアクセス権」 http://www.ipa.go.jp/security/awareness/vendor/programming/a02_03.html

5）エラーメッセージをそのままブラウザに表示しない

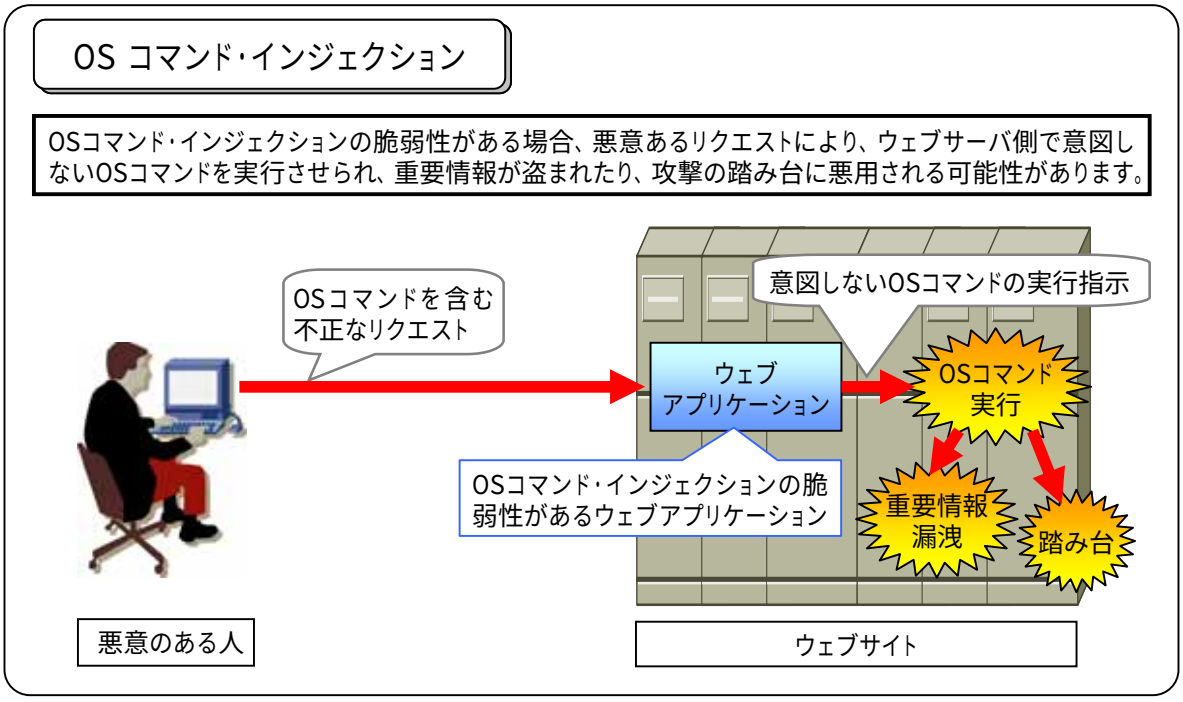
解説	これは、利用者に必要以上の情報を与えないための対策です。エラーメッセージの内容に、データベースの種類やエラーの原因、実行エラーを起こした SQL 文などの情報が含まれる場合、これらは SQL インジェクションの攻撃につながる有用な情報となりえます。データベースに関連するエラーメッセージは、利用者のブラウザ上に表示させないことをお勧めします。

以上の内容を実装することにより、SQL インジェクションに対する安全性の向上が期待できます。データベースと連携したウェブアプリケーションの構築については、次の資料もご参照ください。

参考 URL	セキュアデータベース プログラミング http://www.ipa.go.jp/security/awareness/vendor/programming/a02.html

2.2　OS コマンド・インジェクション

ウェブアプリケーションによっては、外部からの攻撃により、ウェブサーバの OS コマンドを不正に実行されてしまう問題を持つものがあります。この問題を悪用した攻撃手法は、一般に「OS コマンド・インジェクション」と呼ばれています。



OS コマンド・インジェクション攻撃を受けた場合、ウェブサーバに保存されている重要情報を不正に取得されたり、ウェブサーバが攻撃の踏み台として悪用される可能性があります。OS コマンド・インジェクションの対策として、次の内容をご検討ください。

■ 根本的解決

1）シェルを起動できる言語機能の利用を避ける

解説	これは、OS コマンド・インジェクションの原因を作らない実装です。ウェブアプリケーションに利用されている言語によっては、シェルを起動できる機能を持つものがあります。たとえば、Perl 言語の open 関数は、引数として与えるファイルパスに「|」(パイプ)を使うことで OS コマンドを実行できるため、外部からの入力値を引数として利用する実装は危険です。このような、シェルを起動できる言語機能の利用は避けてください。処理の目的によっては、他の関数などで代替できる場合があります。たとえば、Perl 言語でファイルを開きたい場合は、open 関数ではなく、sysopen 関数を利用します。
参考 URL	セキュア Perl プログラミング 「Perl の危険な関数」 http://www.ipa.go.jp/security/awareness/vendor/programming/a04_02_main.html

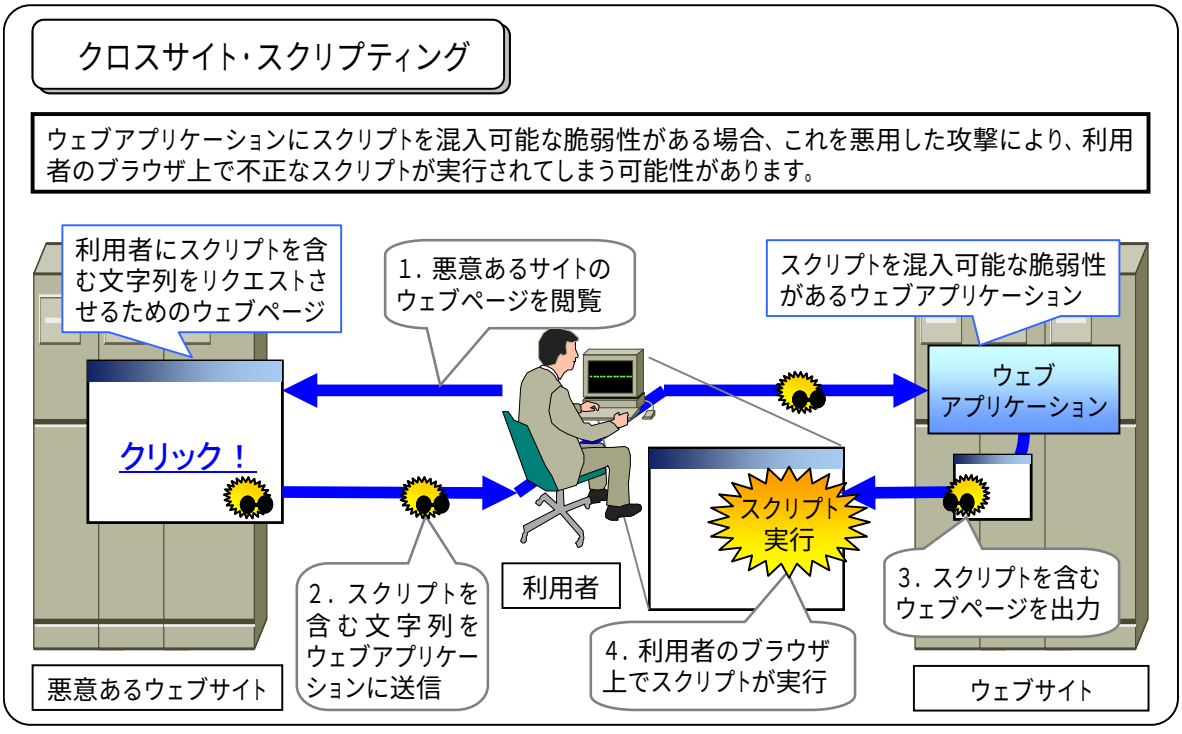
■ 保険的対策

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、あらかじめ許可された処理のみが実行されるようにする

解説	これは、上記解決 1）を実施できない場合や、実施に漏れがあった場合にセーフティネットとなる対策です。シェルを起動できる言語機能の引数を構成する変数に対し、引数に埋め込む前にチェックをかけ、本来想定する動作のみが実行されるようにしてください。チェック方法には、その引数に許可する文字の組み合わせを洗い出し、その組み合わせ以外は許可しない「ホワイトリスト方式」をお勧めします。チェックの結果、許可しない文字の組み合わせが確認された場合は、引数へ渡さず、処理を中止させます。なお、チェック方法には、OS コマンド・インジェクション攻撃に悪用される記号文字（「|」、「<」、「>」など）など、問題となりうる文字を洗い出し、これを許可しない「ブラックリスト方式」もありますが、この方法はチェックに漏れが生じる可能性があるため、お勧めできません。

2.3 クロスサイト・スクリプティング

ウェブアプリケーションの中には、検索のキーワードや個人情報登録時の確認画面、掲示板、ウェブのログ統計画面など、利用者からの入力内容や HTTP ヘッダの情報を処理し、ウェブページとして出力するものがあります。ここで、ウェブページへの出力処理に問題がある場合、そのウェブページにスクリプトを混入されてしまう恐れがあります。この問題を悪用した攻撃手法の一つに、「クロスサイト・スクリプティング」があります。クロスサイト・スクリプティングの影響は、ウェブサイト自身にではなく、そのウェブサイトのページを閲覧している利用者におよびます。



クロスサイト・スクリプティングへの対策を、ウェブアプリケーションの仕様に合わせて示します。まず、ウェブアプリケーションの仕様として、「HTML テキストの入力を許可するかどうか」を確認してください。たとえば、掲示板やブログのようなウェブサイトでは、利用者が入力文字の色やサイズを変更したり、画像を挿入したりできる機能を実装するために、HTML テキストの入力を許可する場合があるかもしれません。一方、検索サイトや個人情報の登録フォームのようなウェブページでは、多くの場合、HTML テキストの入力を許可する必要はありません。ウェブアプリケーションの仕様に合わせ、次の内容をご検討ください。

2.3.1 HTML テキストの入力を許可しない場合

■ 根本的解決

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

解説	これは、スクリプト混入の原因を作らない実装です。ウェブページを構成する要素として、ウェブページの本文や HTML タグの属性値などに相当する全ての出力要素にエスケープ処理を行います。エスケープ処理には、ウェブページの表示に影響する特別な記号文字(「<」、「>」、「&」など)を、HTML エンティティ文字(「<」、「>」、「&」など)に置換する方法があります。また、HTML タグを出力する場合は、その属性値を必ず「"」(ダブルクォート)で括るようにします。そして、「"」で括られた属性値に含まれる「"」を、HTML エンティティ文字「"」にエスケープします。エスケープ処理は、外部からウェブアプリケーションに渡される「入力値」以外に、データベースやファイルから読み込んだ値、HTTP ヘッダ、Cookie、演算によって生成した値などが対象となる場合があります。「入力値」ではなく、「出力値」に注目してエスケープ処理を行うようにしてください。

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

解説	これは、スクリプト混入の原因を作らない実装です。URL には、「http://」や「https://」から始まるものだけでなく、「javascript:」の形式で始まるものもあります。ウェブページに出力するリンク先や画像の URL が、外部からの入力に依存する形で動的に生成される場合、その URL にスクリプトが含まれていると、クロスサイト・スクリプティング攻撃が可能となる場合があります。たとえば、利用者から入力された "リンク先の URL" を <a href="リンク先の URL">の形式でウェブページに出力するウェブアプリケーションは、"リンク先の URL" に「javascript:」などから始まる文字列を指定された場合に、スクリプトを埋め込まれてしまう可能性があります。リンク先の URL には「http://」や「https://」から始まる文字列のみを許可する、「ホワイトリスト方式」で実装してください。

3）<script>…</script> 要素の内容を動的に生成しないようにする

解説	これは、スクリプト混入の原因を作らない実装です。ウェブページに出力する〈script〉...〈/script〉要素の内容が、外部からの入力に依存する形で動的に生成される場合、任意のスクリプトが埋め込まれてしまう可能性があります。危険なスクリプトだけを排除する方法も考えられますが、危険なスクリプトであることを確実に判断することは難しいため、〈script〉...〈/script〉要素の内容を動的に生成する仕様は、避けることが望ましいです。

4）スタイルシートを外部サイトから取り込めるようにしない

解説	これは、スクリプト混入の原因を作らない実装です。スタイルシートには、expression() などを利用してスクリプトを記述することができるため、外部スタイルシートを取り込むことで、生成するウェブページにスクリプトが埋め込まれてしまう可能性があります。取り込んだスタイルシートの内容をチェックし、危険なスクリプトを排除する方法も考えられますが、確実に排除することは難しいため、スタイルシートを外部から指定可能な仕様は、避けることが望ましいです。

■ 保険的対策

5）入力値の内容チェックを行う

解説	これは、上記根本的解決を実施できない場合や、実施に漏れなどがある場合にセーフティネットとなる対策です。入力チェック機能をウェブアプリケーションに実装し、条件に合わない値を入力された場合は、処理を先に進めず、再入力を求めるようにします。ただし、チェックを通過した後の演算処理の結果がスクリプト文字列を形成してしまう場合などには対処できないため、この対策のみに頼ることはお勧めできません。

2.3.2　HTML テキストの入力を許可する場合

■ 根本的解決

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

解説	これは、スクリプト混入の原因を作らない実装です。入力された HTML テキストに対して構文解析を行い、「ホワイトリスト方式」で許可する要素のみを抽出します。ただし、これには複雑なコーディングが要求され、処理に負荷がかかるといった影響もあるため、実装には十分な検討が必要です。

■ 保険的対策

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

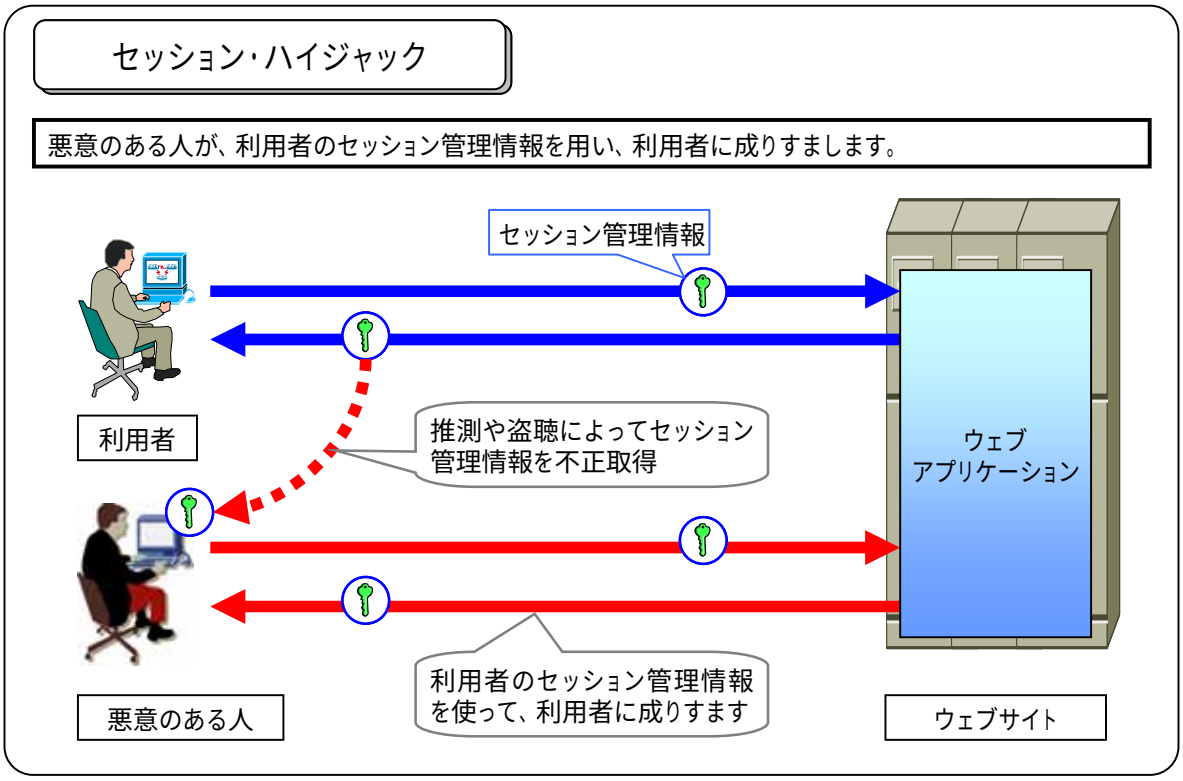
解説	これは、根本的解決 6）を実施できない場合にセーフティネットとなる対策です。入力された HTML テキストに含まれる、スクリプトに該当する文字列を抽出し、排除してください。抽出した文字列の排除方法には、無害な文字へ置換することをお勧めします。たとえば、「<script>」や「javascript:」を無害な文字列へ置換する場合、「<xscript>」「xjavascript:」のように、その文字列に適当な文字を付加します。他の排除方法として、文字列の削除が挙げられますが、削除した結果が危険な文字列を形成してしまう可能性があるため、お勧めできません。なお、この対策は、危険な文字列を完全に抽出することが難しいという問題があります。ウェブブラウザによっては、「java script:」や「java(改行コード)script:」のような文字列を「javascript:」と解釈してしまうため、単純なパターンマッチングでは危険な文字列を抽出することができません。このような「ブラックリスト方式」による対策のみに頼ることはお勧めできません。

以上の内容を実装することにより、クロスサイト・スクリプティングに対する安全性の向上が期待できます。クロスサイト・スクリプティングに関する情報については、次の資料もご参照ください。

参考 URL	セキュア Web プログラミング「クロスサイト・スクリプティング」 http://www.ipa.go.jp/security/awareness/vendor/programming/a01_02.html Web サイトにおけるクロスサイト スクリプティング脆弱性に関する情報 http://www.ipa.go.jp/security/ciadr/20011023css.html

2.4　セッション管理の不備

ウェブアプリケーションの中には、利用者を識別するための情報を発行し、セッション管理を行っているものがあります。このセッション管理情報の発行や管理に不備がある場合、悪意のある人に利用者のセッション管理情報を不正に取得され、その利用者への成りすましにつながる可能性があります。この問題を悪用した攻撃手法は、一般に「セッション・ハイジャック」と呼ばれています。



セッション・ハイジャックが成立した場合、攻撃者は利用者に成りすまし、その利用者本人に許可されている全ての操作を不正に行われてしまいます。セッション・ハイジャックの対策として、次の内容をご検討ください。

■ 根本的解決

1）セッション管理情報を推測が困難なものにする

解説	これは、セッション管理情報が推測され、第三者に入手されてしまうことを回避する方法です。セッション管理情報の生成規則が単純な場合、その値の推測を容易にしてしまいます。そして、推測したセッション管理情報を使ってウェブページにアクセスすることで、セッション・ハイジャックが行われてしまう可能性があります。セッション管理情報の値にはランダムな数字や文字を与え、攻撃者の推測が困難なものにしてください。

2）HTTPS 通信で利用する Cookie には secure 属性を加える

解説	これは、盗聴による Cookie の不正取得を防止するための方法です。ウェブサイトが発行する Cookie には、secure 属性と呼ばれる設定項目があり、これが設定された Cookie は、HTTPS 通信のみで利用されます。Cookie に secure 属性がない場合、HTTPS 通信で発行した Cookie は、経路が暗号化されていない HTTP 通信でも利用されるため、この HTTP 通信の盗聴により Cookie 情報を不正に取得されてしまう可能性があります。HTTPS 通信で利用する Cookie には secure 属性を必ず加えてください。また、HTTP 通信で Cookie を利用する場合は、HTTPS で発行する Cookie とは別のものを発行してください。
参考 URL	経路のセキュリティと同時にセキュアなセッション管理を http://www.ipa.go.jp/security/ciadr/20030808cookie-secure.html

■ 保険的対策

3）セッション管理情報を固定値にしない

解説	これは、セッション・ハイジャックが行われる機会を低減する対策です。利用者に発行するセッション管理情報が固定値の場合、この情報が攻撃者に入手されると、時間の経過に関係なく、いつでもセッション・ハイジャックを行われてしまいます。セッション管理情報は、利用者のログイン毎に新しく発行し、固定値にしないようにしてください。

4）セッション管理情報を Cookie にセットする場合、有効期限の設定に注意する

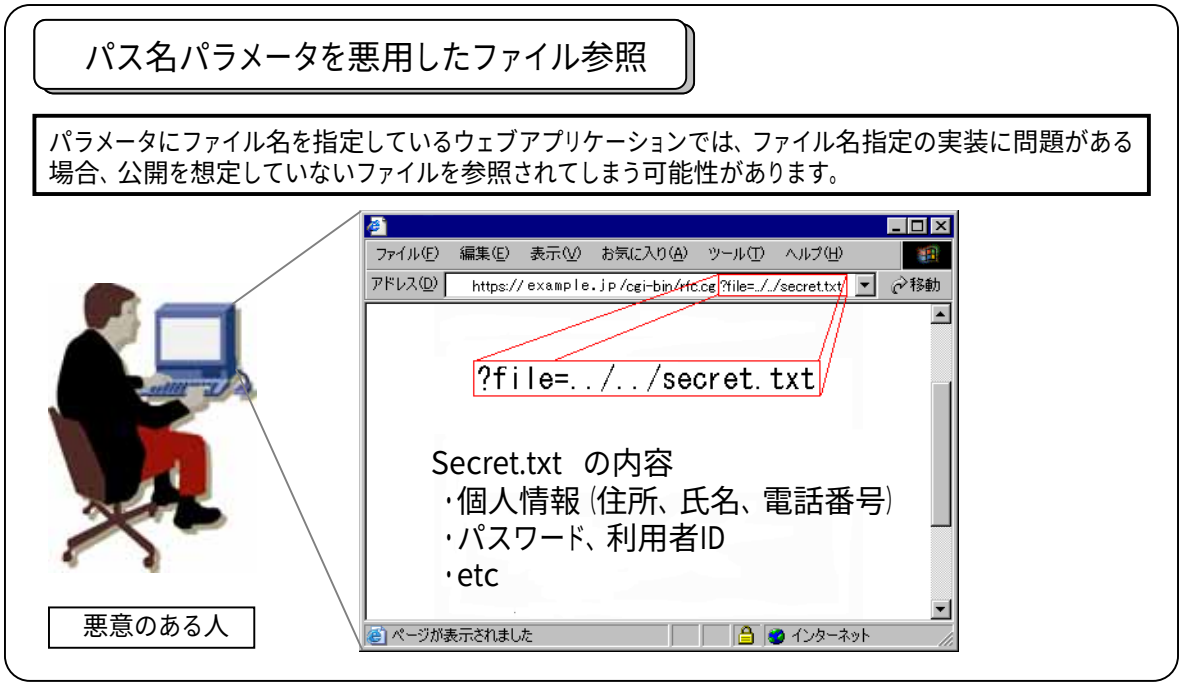
解説	これは、Cookie が盗まれてしまう可能性を低減する対策です。Cookie は有効期限が過ぎるまでブラウザに保持されるため、ブラウザの脆弱性を悪用するなど何らかの方法で Cookie を盗むことが可能な場合、その時点で保持されていた Cookie が盗まれてしまう可能性があります。Cookie を発行する場合は、有効期限の設定に注意してください。たとえば、Cookie をブラウザに残す必要が無い場合は、有効期限の設定（expires=）を省略することにより、発行した Cookie をブラウザ終了後に破棄させます。

以上の対策により、セッション・ハイジャックが行われる可能性を低減することができます。セッション管理に関する情報ついては、次の資料もご参照ください。

参考 URL	セッション管理 http://www.ipa.go.jp/security/awareness/administrator/secure-web/chap6/6_session-1.html セッション管理の留意点 http://www.ipa.go.jp/security/awareness/administrator/secure-web/chap6/6_session-2.html

2.5　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

ウェブアプリケーションの中には、外部からのパラメータにウェブサーバ内のファイル名を直接指定しているものがあります。このようなウェブアプリケーションでは、ファイル名指定の実装に問題がある場合、攻撃者に任意のファイルを指定され、ウェブアプリケーションが意図しない処理を行ってしまう可能性があります。



ウェブアプリケーションの処理内容によっては、ウェブサイト運営者が公開を想定していないファイルの閲覧につながる場合があります。攻撃者による任意のファイル指定への対策として、次の内容をご検討ください。

■ 根本的解決

1）外部からのパラメータにウェブサーバ内のファイル名を直接指定できる実装を避ける

解説	これは、任意ファイルを参照される問題の原因を作らない実装です。外部からのパラメータにウェブサーバ内のファイル名を指定できる場合、そのパラメータが改変され、任意のファイル名を指定され、公開を想定しないファイルの閲覧につながる可能性があります。外部パラメータからウェブサーバ内のファイル名を指定する実装が本当に必要かどうか、他の処理方法で代替できないかどうかなど、仕様や設計から見直すことをお勧めします。
参照URL	セキュア Perl プログラミング「ファイルオープン時のパスにご用心」 http://www.ipa.go.jp/security/awareness/vendor/programming/a04_01.html

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

解説	これは、根本的解決 1) を実施できず、外部からの入力値でファイル名を指定する必要がある場合に、任意ディレクトリのファイルが指定されることを回避する実装です。たとえば、カレントディレクトリ上のファイル「filename」を開くつもりで、open(filename)のような形でコーディングしている場合、open(filename)の filename に絶対パス名が渡されることにより、任意ディレクトリのファイルが開いてしまう場合があります。この絶対パス名による指定を回避する方法として、あらかじめ固定のディレクトリ「dirname」を指定し、open(dirname+filename)のような形でコーディングする方法があります。さらに、「../」などを利用したディレクトリ・トラバーサル攻撃を回避するために、basename()などの関数を利用して、filename に与えられたパス名からファイル名のみを取り出すようにします。

■ 保険的対策

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

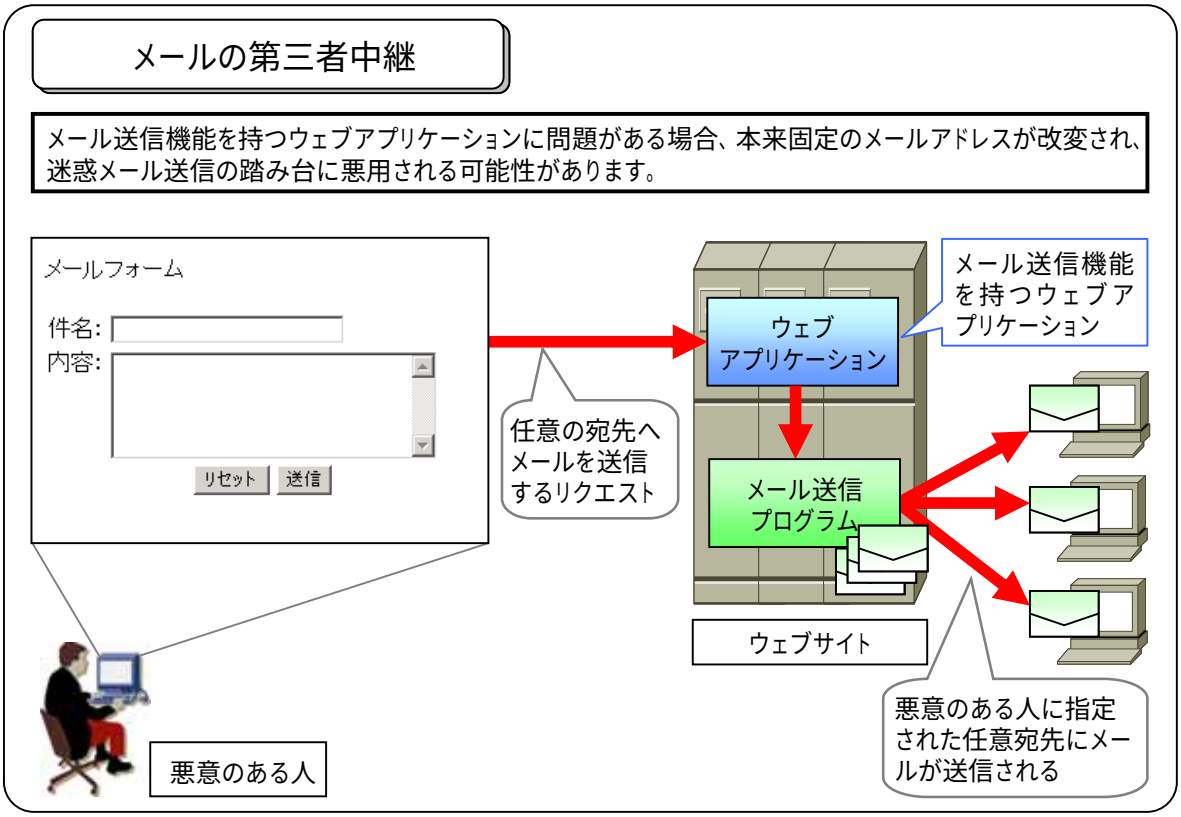
解説	これは、攻撃による影響を低減するための対策です。ウェブサーバ内に保管しているファイルへのアクセス権限が正しく管理されていれば、ウェブアプリケーションが任意ディレクトリのファイルを開く処理を実行しようとしても、ウェブサーバ側の機能でそのアクセスを拒否し、被害の低減が期待できます。

4) ファイル名のチェックを行う

解説	これは、上記根本的解決を実施できない場合にセーフティネットとなる対策です。ファイル名を指定したパラメータにチェックをかけ、"/"、"../"、"..¥" などから始まる文字列を検出した場合は、処理を中止します。ただし、デコード処理を行っている場合は、URL エンコードした "%2F"、"..%2F"、"..%5C"、さらに二重エンコードした "%252F"、"..%252F"、"..%255C" がファイル指定の入力値として有効な文字列となる場合があります。チェックを行うタイミングには注意してください。
参考 URL	セキュア Unix/Linux プログラミング 「Unix パス名の安全対策」 http://www.ipa.go.jp/security/awareness/vendor/programming/b07_07_main.html セキュア Windows プログラミング 「Windows パス名の落とし穴」 http://www.ipa.go.jp/security/awareness/vendor/programming/b08_01_main.html

2.6　メールの第三者中継

ウェブアプリケーションの中には、利用者が入力した商品申し込みやアンケートなどの内容を、指定のメールアドレスに送信する機能を持つものがあります。一般に、このメールアドレスは固定で、ウェブアプリケーションの管理者以外の人は変更できませんが、実装によっては、外部の利用者がこのメールアドレスを自由に指定できてしまう場合があります。この問題を悪用した攻撃は、「メールの第三者中継」と呼ばれています。



問題のあるウェブアプリケーションは、悪意のある人が迷惑メールを送信する際に、身元を隠すための踏み台として悪用される可能性があります。メールの第三者中継の対策として、次の内容を検討してください。

■ 根本的解決

1）外部からのパラメータをメールヘッダの内容に指定しない

解説	これは、ウェブアプリケーションがメールの第三者中継に悪用されないための実装です。To、Cc、Bcc、Subject などのメールヘッダの内容が外部からの入力に依存する場合や、メール送信プログラムへの出力処理に問題がある場合、ヘッダおよび本文を含むメール全体を改ざんされ、任意の宛先に任意の内容のメールを送信されてしまう可能性があります。たとえば、フォームの hidden 形式で指定したメールアドレスを送信先（To）とする場合、その hidden 形式のパラメータを改変され、任意の宛先にメールを送信されてしまいます。外部からのパラメータをメールヘッダの内容に指定しない実装をお勧めいたします。

■ 保険的対策

2）外部からのパラメータをメールヘッダに指定する場合は、危険な文字を排除する

解説	これは、上記根本的解決を実施できない場合にセーフティネットとなる対策です。外部からのパラメータをメールヘッダに指定する場合は、危険な文字を排除してください。たとえば、perl 言語の open 関数を利用したパイプ出力で sendmail コマンドにメール内容を渡している場合、メールヘッダの内容に該当する変数に対して、メールヘッダの区切りとして認識される改行コード（¥n, ¥r¥n, ¥r など）を許可しないようにします。これにより、改行コードに続けて任意の他のメールヘッダを指定されたり、さらに続けてメール本文を指定されたりするメール内容の改変を回避することができます。

3. ウェブサイトの安全性向上のための取り組み

ここでは、ウェブサイト全体の安全性を向上するための取り組みを掲載しています。前章「ウェブアプリケーションのセキュリティ実装」では、設計や実装レベルでの解決や対策を示しましたが、ここで取り上げている内容は、主に運用レベルでの解決や対策となります。

3.1 ウェブサーバのセキュリティ対策

ウェブアプリケーションのセキュリティ対策が十分でも、ウェブサーバへの不正侵入を許してしまっては意味がありません。ここで示す項目は、一般に知られている基本的なことです。以下を参考に、管理しているウェブサーバの設定や運用に問題がないかご確認ください。

1) OS やソフトウェアの脆弱性情報を継続的に入手し、脆弱性への対処を行う

解説	これは、攻撃の糸口を与えないための運用です。OS やソフトウェアの脆弱性をついた攻撃を受けると、たとえサーバへのアクセスに認証をかけていても、不正侵入されてしまう場合があります。脆弱性は日々発見されるので、OS やソフトウェアの開発者から提供される脆弱性情報を継続的に入手し、ソフトウェアの更新や問題の回避を行ってください。
参考 URL	情報処理推進機構 セキュリティセンター http://www.ipa.go.jp/security/ JP Vendor Status Notes http://jvn.jp/

2) ウェブサーバをリモート操作する際の認証方法として、パスワード認証以外の方法を検討する

解説	これは、認証強度を高める運用です。サーバ管理の上で、ウェブサーバのリモート操作を許可する運用は一般的ですが、その際の認証方法にパスワード認証のみを利用している場合、総当り攻撃などにより、パスワード認証を突破されてしまう可能性があります。より高い安全性を確保するための方法として、公開鍵認証などの利用を検討することをお勧めします。

3) パスワード認証を利用する場合は、十分に複雑な文字列を設定する

解説	これは、上記 2) の実施が難しい場合に必要な運用です。ウェブサーバへ接続する際のパスワードには、十分に複雑な文字列を設定してください。また、パスワードの運用にについて、下記情報を参考にすることをお勧めします。
参考 URL	パスワードの管理と注意 http://www.ipa.go.jp/security/fy14/contents/soho/html/chap1/pass.html

4） 不要なサービスやアカウントを停止または削除する

解説	これは、攻撃の糸口を与えないための運用です。ウェブサイト運営に必要のないサービスがウェブサーバ上で稼動している場合、そのサービスに対する管理が十分でなく、脆弱性が存在するバージョンをそのまま利用している可能性などが考えられます。また、用途が明確でないユーザアカウントが存在している場合、そのアカウントに対する管理が十分でなく、不正利用される可能性が考えられます。必要の無いサービスやアカウントは停止または削除してください。

5） 公開を想定していないファイルを、ウェブ公開用のディレクトリ以下に置かない

解説	これは、情報漏えいを回避するための運用です。ウェブ公開用のディレクトリに保管されているファイル群は、基本的に外部から閲覧することが可能です。公開ウェブページにファイルへのリンクが無くても、外部から直接指定することで閲覧されてしまいます。公開を想定していないファイルは、ウェブ公開用のディレクトリに保管しないようにしてください。

安全なウェブサーバの構築方法と運用方法について、下記の資料もご参照ください。

参考 URL	セキュアな Web サーバの構築と運用に関するコンテンツ http://www.ipa.go.jp/security/awareness/administrator/secure-web/

3.2 DNS 情報の設定不備

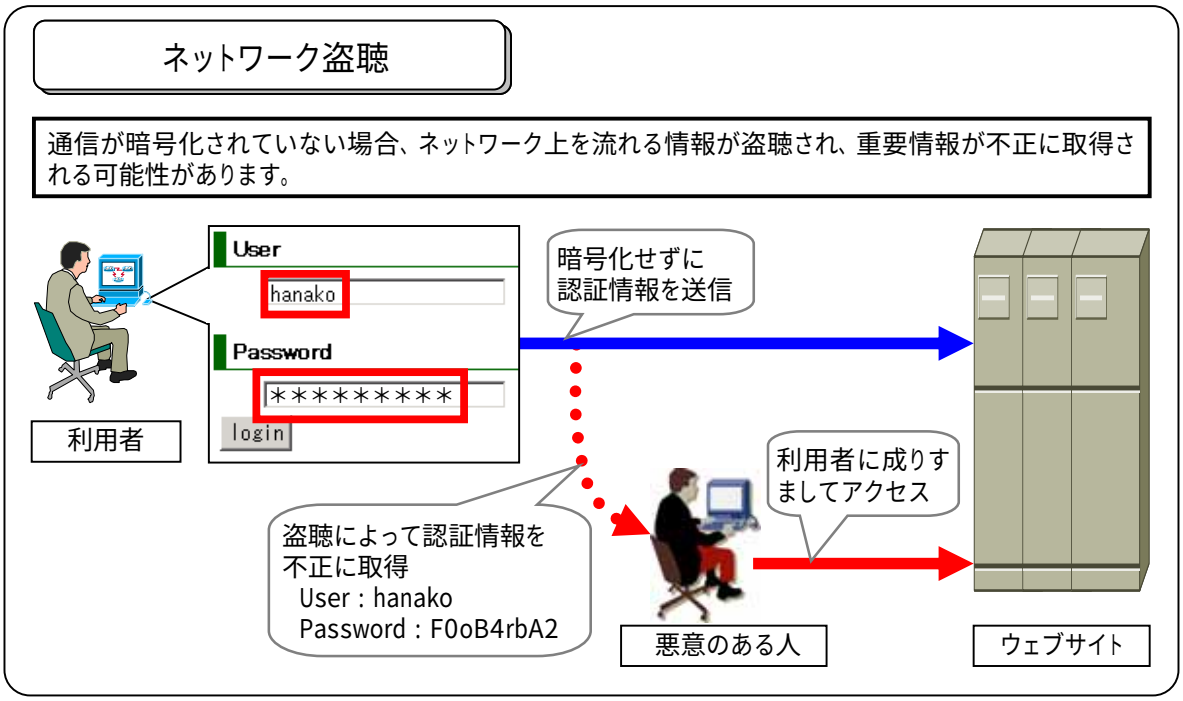
ウェブサイトが利用しているドメイン名およびその DNS サーバについて、問題のある運用や設定は、悪意ある人によるドメイン名乗っ取りにつながる可能性があります。ドメイン名の乗っ取りを行われた場合、利用者が本物のウェブサイトの URL を指定しても、そのドメイン名を乗っ取った人が用意したウェブサイトに接続してしまいます。ドメイン名の乗っ取りによる影響は、ウェブサイトだけでなく、電子メールなどのインターネットを利用するサービス全てに及びます。DNS に関する問題ですが、ウェブサイトにも直接影響する問題であるため、注意が必要です。

1）ドメイン名およびその DNS サーバの登録状況を調査し、必要に応じて対処を行う

対策	これは、ドメイン乗っ取りを回避するための対策です。ドメイン名およびその DNS サーバについて、登録状況を確認し、必要に応じて対処を行ってください。DNS サーバの運用を外部に依託している場合は、その委託先に対処を依頼する必要があります。詳細は下記の情報をご参照下さい。
参考 URL	ドメイン名の登録と DNS サーバの設定に関する注意喚起 http://www.ipa.go.jp/security/vuln/20050627_dns.html

3.3　ネットワーク盗聴への対策

ウェブサイトと利用者の間で交わされる情報は、ネットワークの盗聴によって不正に取得される可能性があります。通信や情報が暗号化されていない場合、盗聴によって取得された情報が悪用され、成りすましなどにつながる可能性があります。



ネットワーク盗聴はウェブサイトと利用者との経路上で行われるため、この行為自体をウェブサイト側の運用や設定のみで防ぐことは困難です。しかし、盗聴による影響を低減することは可能です。特に認証情報や個人情報を扱うウェブサイトでは、ネットワーク盗聴への対策として、次の内容をご検討ください。

1）重要な情報を取り扱うウェブページでは、通信を暗号化する

解説	これは、盗聴による情報漏えいを回避するための対策です。通信を暗号化する主な手段として、SSL（Secure Socket Layer）や TLS（Transport Layer Security）を用いた HTTPS 通信の利用があります。個人情報の登録ページや認証情報をリクエストするログインページなど、保護するべき情報を扱うウェブサイトでは、通信を暗号化することをお勧めします。レンタルサーバを利用してウェブサイトを運営している場合、レンタルサーバのサービスが HTTPS 通信を提供していない場合があります。このようなウェブサイトで重要情報を扱うことはお勧めできません。

2）利用者へ通知する重要情報は、メールで送らず、暗号化された https://のページに表示する

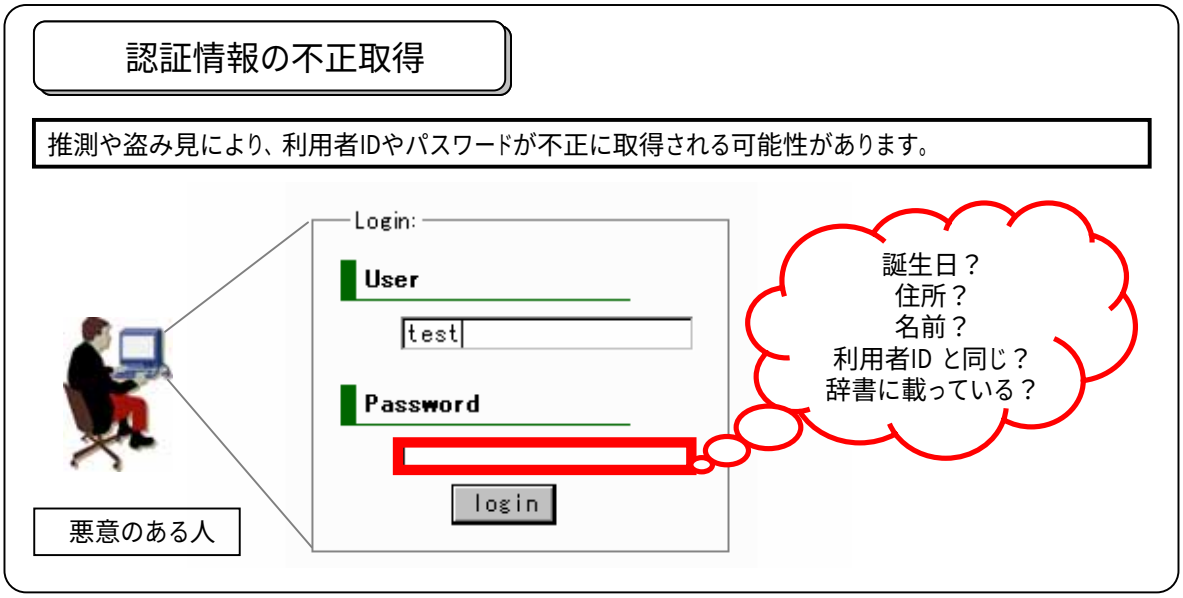
解説	これは、盗聴による情報漏えいを回避するための対策です。ウェブサイトの運営によっては、ウェブサイトの利用者に、個人情報やパスワードなどの重要情報を通知する場合があります。ここで、ネットワークを経由して情報を送信する場合には、盗聴対策として通信の暗号化か、重要情報の暗号化が必要になります。暗号化が必要な情報を利用者に通知する場合は、HTTPS 通信を利用し、ウェブページに表示することをお勧めします。メールを利用する場合には、メール本文の暗号化として、S/MIME（Secure / Multipurpose Internet Mail Extensions）や PGP（Pretty Good Privacy）などの技術がありますが、利用者側に暗号化環境や秘密鍵が必要となるため、現実的ではないかもしれません。

3）ウェブサイト運営者がメールで受け取る重要情報は、暗号化を施す

解説	これは、盗聴による情報漏えいを回避するための対策です。ウェブページに入力された個人情報などの重要情報を、ウェブアプリケーションに実装されたメール通知機能を利用して、ウェブサイト運営者がメールで受け取る場合は、S/MIME や PGP などを利用してメールを暗号化するようにしてください。S/MIME や PGP を利用できない場合には、その他の方法でメール本文を暗号化するようにします。なお、盗聴対策として、メールサーバ間の通信の暗号化（SMTP over SSL）やメールサーバとウェブサイト運営者との通信の暗号化（POP/IMAP over SSL）なども考えられますが、ネットワーク構成によっては、途中経路が暗号化されない可能性があるため、安全とは言えません。
参考 URL	S/MIME を利用した暗号化と電子署名 http://www.ipa.go.jp/security/fy12/contents/smime/email_sec.html

3.4　パスワードの不備

ウェブサイトにおける利用者の認証は、利用者 ID とパスワードを用いる方法が一般的です。しかし、パスワードの運用やウェブページ上のパスワードの取り扱い方法に問題がある場合、利用者の認証情報が悪意ある人に不正取得される可能性が高まります。



認証情報の不正取得の手段の一つに、利用者 ID やパスワードの「推測」があります。これは、推測されやすい単純なパスワードで運用している場合に悪用される手段ですが、ウェブページの表示方法によっては、さらに推測のヒントを与えてしまう場合があります。利用者の認証を行うウェブサイトでは、次の内容に注意してください。

1）初期パスワードは、推測が困難な文字列で発行する

解説	これは、認証情報の推測を困難にするための対策です。初期パスワードは、乱数を利用して規則性をなくし、可能であれば英数字や記号を含めた長い文字列で発行してください。パスワード発行に規則性がある場合、調査のためのテストユーザを複数登録され、その際に発行されたパスワードから規則性を導き出されてしまうかもしれません。利用者によっては、初期パスワードを変更せずに継続して利用することも考えられるため、初期パスワードが推測しやすい仕様は避けるべきです。

2）パスワードの変更には、現行パスワードの入力を求める

解説	これは、第三者がパスワードを変更できないようにするための対策です。パスワードの変更には、必ず現行パスワードの入力を求めるようにしてください。

3）入力後の応答メッセージが認証情報の推測のヒントとならない工夫する

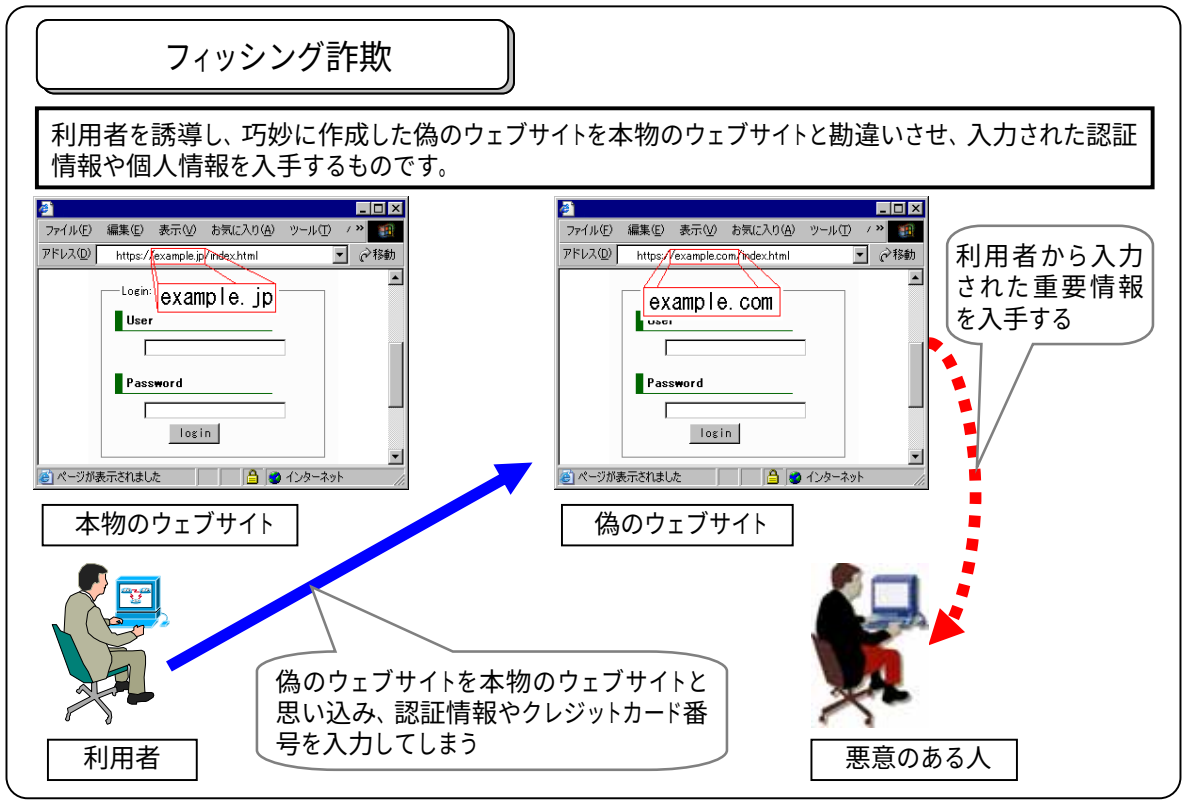
解説	これは、認証情報の推測を困難にするための対策です。認証画面で利用者が入力を誤った際、遷移後の画面で「パスワードが間違っています」というエラーメッセージを表示するものは、「利用者 ID は正しく、パスワードが間違っている」ということを示していることになります。このような表示内容は、登録されている利用者 ID の割り出しを容易にしてしまうため、お勧めできません。入力後の応答メッセージには、「利用者 ID もしくはパスワードが違います」というような表示を用い、認証情報の推測のヒントを与えない工夫をしてください。

4）パスワード入力のフォームでは、入力文字列を伏せ字で表示する

解説	これは、認証情報の盗み見を回避するための対策です。利用者が入力したパスワード文字列は、伏せ字（アスタリスク ″*″）で表示してください。

3.5 フィッシング詐欺を助長しないための対策

フィッシング詐欺とは、悪意のある人が、金融サイトやショッピングサイトなどを装った偽のウェブサイトに利用者を巧みに誘導し、利用者の認証情報やクレジットカード番号などを不正に取得するものです。フィッシング詐欺の回避には、利用者側の注意が必要ですが、ウェブサイトの運用によっては、利用者の注意を妨げ、結果としてフィッシング詐欺を助長してしまう可能性があります。



ウェブサイト利用者がフィッシング詐欺の被害に遭わないためには、利用者自身がアクセスしたウェブサイトを注意深く確認し、本物のウェブサイトかどうかを見極める必要があります。利用者が本物のウェブサイトであることを正しく確認できるよう、次の内容をご検討ください。

1）ウェブサーバの電子証明書を取得し、サイト運営主体の実在性の証明をする

解説	これは、ウェブサイトにアクセスした利用者に対し、そのウェブサイトが本物であることを確認する手段を提供する方法です。運営するウェブサイトが本物であることを証明するために、ウェブサーバ向けの電子証明書を取得し、実在証明を行ってください。電子証明書は、第三者である認証局に発行を依頼します。なお、発行を依頼する認証局には、ウェブブラウザの多くに登録されているルート証明機関や、ルート証明機関の認証を受けた中間証明機関を選択してください。
参考URL	PKI 関連技術解説 「認証局と電子証明書」 http://www.ipa.go.jp/security/pki/031.html

2）ウェブページの URL 情報を表示する

解説	これは、ウェブサイトにアクセスした利用者に対し、そのウェブサイトが本物であることを確認する手段を提供する方法です。URL を表示していないウェブサイトを訪れた利用者は、そのウェブサイトが本物か偽のものかを一目で確認することができません。アドレスバーやステータスバーを隠さず、URL を表示してください。

3）フレームを利用する場合、子フレームの URL を外部パラメータから生成しないように実装する

解説	これは、ウェブサイト自身がフィッシング詐欺に悪用されることを回避する実装です。フレームを利用しているウェブページで、子フレームの URL を外部パラメータから生成する実装は、フィッシング詐欺に悪用される危険性があります。そのパラメータに任意の URL を指定したリンクを仕掛けられた場合、そのリンクをアクセスした利用者は、本物サイトの親フレーム内に、偽サイトのウェブページを子フレームとして埋め込まれた画面を閲覧することになります。表示上のドメインは本物であるため、利用者が子フレームを偽サイトと見分けることは困難です。

4．おわりに

本資料で取り上げたウェブアプリケーションのセキュリティ実装やウェブサイトの安全性向上のための取り組みにより、ウェブサイト運営上の脅威の低減が期待できます。また、組織内部でセキュリティ対策の確認、実施を行った上で、外部組織によるペネトレーションテストやウェブアプリケーションのコードチェック等の監査を受けることは、セキュリティ上、より効果的です。ウェブサイトの重要度に応じて、監査を受けることをお勧めします。

本資料が、ウェブサイトのセキュリティ問題を解決する一助となれば幸いです。

参考資料

・ 脆弱性関連情報に関する届出について
http://www.ipa.go.jp/security/vuln/report/index.html
・ セキュア・プログラミング講座
http://www.ipa.go.jp/security/awareness/vendor/programming/
・ セキュア・プログラミング講座 「WEB プログラマコース」
http://www.ipa.go.jp/security/awareness/vendor/programming/a00.html
・ セキュア DB プログラミング
「バインドメカニズムを活用しよう」「入力文字列はエスケープしよう」「LIKE 句では%と_に気をつけよう」
http://www.ipa.go.jp/security/awareness/vendor/programming/a02_01.html
・ セキュア Web プログラミング 「hidden は危険」
http://www.ipa.go.jp/security/awareness/vendor/programming/a01_05.html
・ セキュア DB プログラミング 「データベースとアクセス権」
http://www.ipa.go.jp/security/awareness/vendor/programming/a02_03.html
・ セキュアデータベース プログラミング
http://www.ipa.go.jp/security/awareness/vendor/programming/a02.html
・ セキュア Perl プログラミング 「Perl の危険な関数」
http://www.ipa.go.jp/security/awareness/vendor/programming/a04_02_main.html
・ セキュア Web プログラミング 「クロスサイト・スクリプティング」
http://www.ipa.go.jp/security/awareness/vendor/programming/a01_02.html
・ Web サイトにおけるクロスサイト スクリプティング脆弱性に関する情報
http://www.ipa.go.jp/security/ciadr/20011023css.html
・ 経路のセキュリティと同時にセキュアなセッション管理を
http://www.ipa.go.jp/security/ciadr/20030808cookie-secure.html
・ セッション管理
http://www.ipa.go.jp/security/awareness/administrator/secure-web/chap6/6_session-1.html
・ セッション管理の留意点
http://www.ipa.go.jp/security/awareness/administrator/secure-web/chap6/6_session-2.html

- セキュア Perl プログラミング 「ファイルオープン時のパスにご用心」
  http://www.ipa.go.jp/security/awareness/vendor/programming/a04_01.html
- セキュア Unix/Linux プログラミング 「Unix パス名の安全対策」
  http://www.ipa.go.jp/security/awareness/vendor/programming/b07_07_main.html
- セキュア Windows プログラミング 「Windows パス名の落とし穴」
  http://www.ipa.go.jp/security/awareness/vendor/programming/b08_01_main.html
- 情報処理推進機構 セキュリティセンター
  http://www.ipa.go.jp/security/
- JP Vendor Status Notes
  http://jvn.jp/
- パスワードの管理と注意
  http://www.ipa.go.jp/security/fy14/contents/soho/html/chap1/pass.html
- セキュアな Web サーバの構築と運用に関するコンテンツ
  http://www.ipa.go.jp/security/awareness/administrator/secure-web/
- ドメイン名の登録と DNS サーバの設定に関する注意喚起
  http://www.ipa.go.jp/security/vuln/20050627_dns.html
- S/MIME を利用した暗号化と電子署名
  http://www.ipa.go.jp/security/fy12/contents/smime/email_sec.html
- PKI 関連技術解説 「認証局と電子証明書」
  http://www.ipa.go.jp/security/pki/031.html