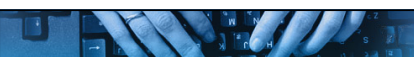


☞ 早めのチェック ☞
3工程によるセキュリティ品質確保

「第1回 要件定義工程における保護対象の識別が重要である対策」

2010年4月
IPA セキュリティセンター 企画グループ

〔セキュア・プログラミング講座 Webアプリケーション編 にもとづく〕



アジェンダ

1-1 総論・その1

- 1-1-1 総論と対策の分類
- 1-1-2 開発工程と脆弱性対策

1-2 暴露対策


- 1-2-1 Webサーバからのファイル流出対策
- 1-2-2 プログラムからのファイル流出対策
- 1-2-3 コンテンツ間パラメータ対策
- 1-2-4 デバッグオプション対策
- 1-2-5 プロキシキャッシュ対策



1-1 総論・その1

1-1-1 総論と対策の分類

1-1-2 開発工程と脆弱性対策



1-1-1 総論と対策の分類

Webアプリケーションの特性

- Webアプリケーションの特性
 - 緩く結合されたクライアント・サーバ構成
 - WebブラウザとWebサーバ
 - キャッシュ機能をもつ設備によって中継されることもある
 - ユーザインタフェースはWebブラウザで動作
 - サーバから供給されるHTMLあるいはXMLの記述にもとづく
 - アプリケーションの仕組みがユーザの目に触れやすい
 - ユーザはサーバへ送られるリクエスト内容を改ざんできる
 - 高機能のスクリプトエンジン
 - Webブラウザの多くが備える
 - スクリプトからの画面内容の書き換えやネットワークアクセスが可能
 - 悪意のスクリプトが動作するおそれ
- Webアプリケーションには情報セキュリティ問題が生じがちである

セキュリティ問題の分類

- Webアプリケーションのセキュリティ問題の分類
 - 暴露問題
 - エコーバック問題
 - 入力問題
 - セッション問題
 - アクセス制御問題
 - 各種の問題

暴露問題

- 暴露問題

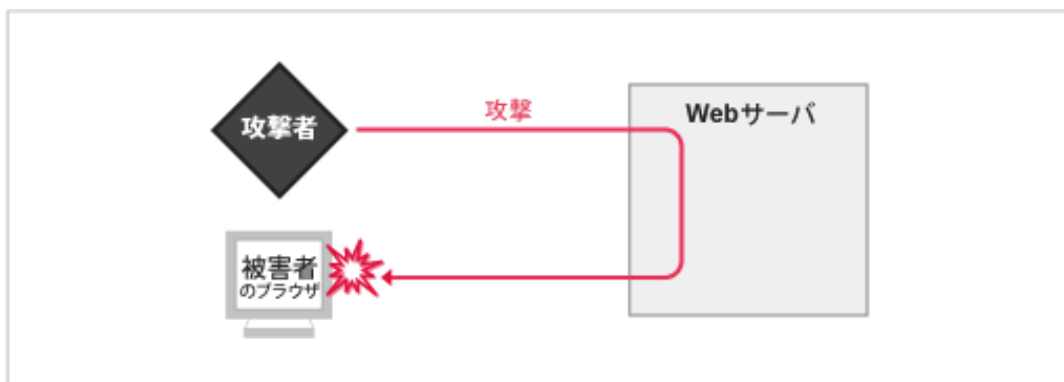
- Webサーバは予定外のファイルを開示するおそれがある
- あるWebコンテンツに埋め込まれたURLによって別のWebコンテンツを呼び出す際、そこに置かれたパラメータが重要な情報を暴露していたり、干渉を受けるおそれがある



エコーバック問題

- エコーバック問題

- Webアプリケーションが入力パラメータに何も対策を施さずにHTMLページやHTTPレスポンスヘッダへエコーバック出力を行うロジックをもつことは、「スクリプト注入」や「HTTPレスポンスによるキャッシュ偽造」の問題を起こす



入力問題

入力問題

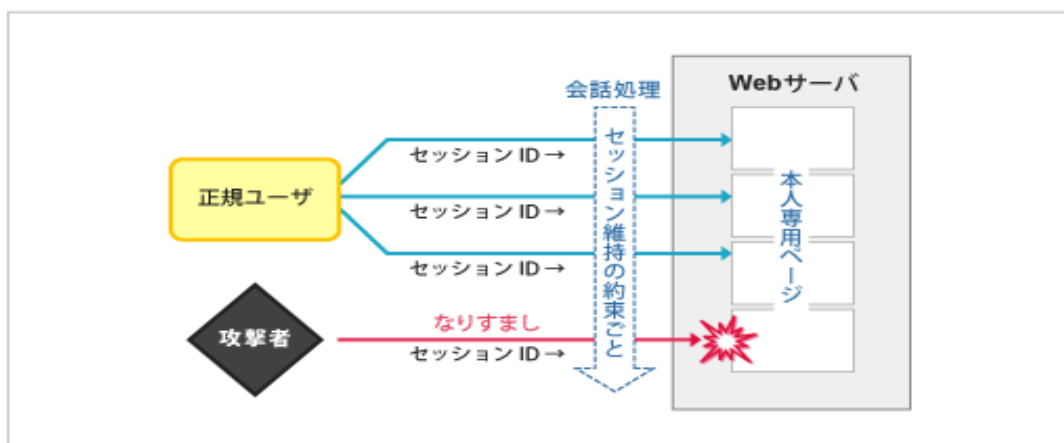
- Webアプリケーションが取り込む入力パラメータには「SQL注入」「コマンド注入」をはじめとする攻撃を意図した悪意ある内容が含まれているおそれがある



セッション問題

セッション問題

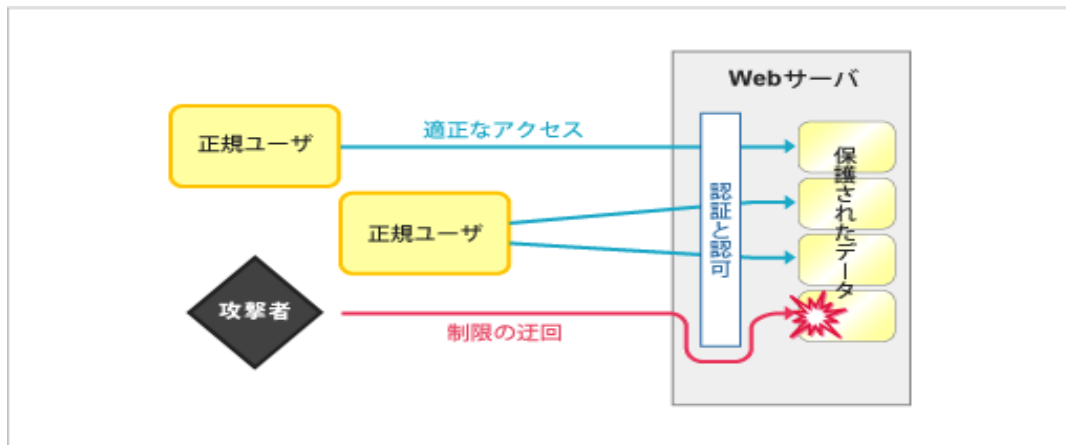
- Webアプリケーションがセッションを維持する仕組みは必ずしも堅固なものではなく、他者からの干渉やセッションの乗っ取りのおそれがある



アクセス制御問題

- アクセス制御問題

- Webアプリケーションを構成するコンテンツのひとつひとつはそれぞれURLで呼び出される形式をとるものであり、実装方法によってはアクセス制御が迂回されるおそれがある



各種の問題

- 各種の問題

- 上記5つのカテゴリには分類されないもの
 - メールの第三者中継
 - 偽ページ 等

本セミナーの構成

- **第1回 要件定義および暴露対策の論点**
 - 総論・その1
 - » 「総論と対策の分類」「開発工程と脆弱性対策」
 - 暴露対策
 - » 「Webサーバからのファイル流出対策」「プログラムからのファイル流出対策」等
- **第2回 設計を中心とした論点**
 - セッション対策
 - » 「リクエスト強要(CSRF)対策」「セッション乗っ取り対策」等
 - アクセス制御対策
 - » 「ユーザ認証対策」「アクセス許可対策」
 - 総論・その2
 - » より良いWebアプリケーション設計のヒント」
- **第3回 実装を中心とした論点**
 - 入力対策
 - » 「SQL注入」「コマンド注入攻撃対策」等
 - エコーバック対策
 - » 「スクリプト注入(XSS)」「HTTPレスポンスによるキャッシュ偽造」
 - サイトデザインにかかわる対策
 - » 「メールの第三者中継対策」「真正性の主張」

1-1-2 開発工程と脆弱性対策



開発工程と脆弱性対策

- **開発工程と脆弱性対策**

- Webアプリケーションの脆弱性対策にはさまざまなものがある
 - 実装段階で考慮すれば済むもの
 - 設計の上流段階から考慮したほうがよいもの 等
- この一連のコンテンツが述べる脆弱性対策は、Webアプリケーションの開発工程と次のように関連づけられる



開発・運用工程の想定

- **開発・運用工程の想定**

- Webアプリケーションの開発・運用工程については、さまざまな考え方があり得るが、このセミナーでは次のように想定する
 - (1) 要件定義
 - (2) 基盤の選定
 - (3) 設計
 - (4) 実装
 - (5) テストと統合
 - (6) 運用・保守

工程：要件定義、基盤選定

(1) 要件定義

- 構築対象システムがどのようなものであらねばならないかを定める段階
- どのようにコンピュータおよびネットワークを配し、どのようなデータ処理を行うか、人々はシステムにどのようにアクセスするかといった枠組みを決める

(2) 開発基盤の選定

- 対象システムを稼働させるための技術的な基盤を選定する段階
- プログラミング言語、アプリケーションフレームワーク、OS、サーバマシン等を決める
- これ以降の工程は、ここにおける意思決定に強く束縛される

工程：設計

(3) 設計

- 対象システムをどのような構造や仕様で実現するかを決める段階
- 設計の中は、さらに次のような工程に分かれる
 - (3-1) システム基本設計
 - システムの主要な構造を決める工程。ここでは骨格のレベルで、サブシステム、コンポーネント、データベース、主要ファイル等の配置を決める
 - (3-2) 業務仕様設計
 - ページ遷移の体系、各ページのレイアウト、各入出力項目の仕様を決める
 - (3-3) モジュール分割設計
 - プログラム実装にあたってのモジュール構造、共通モジュールの持ち方等を決める
 - (3-4) テスト計画
 - 実装の誤りを検出して解消するためのテスト計画を立てる

工程：実装、テスト、運用

(4) プログラム実装

- それぞれのモジュールを実装する段階。詳細ロジックの設計、コーディング、および単体テストを含む

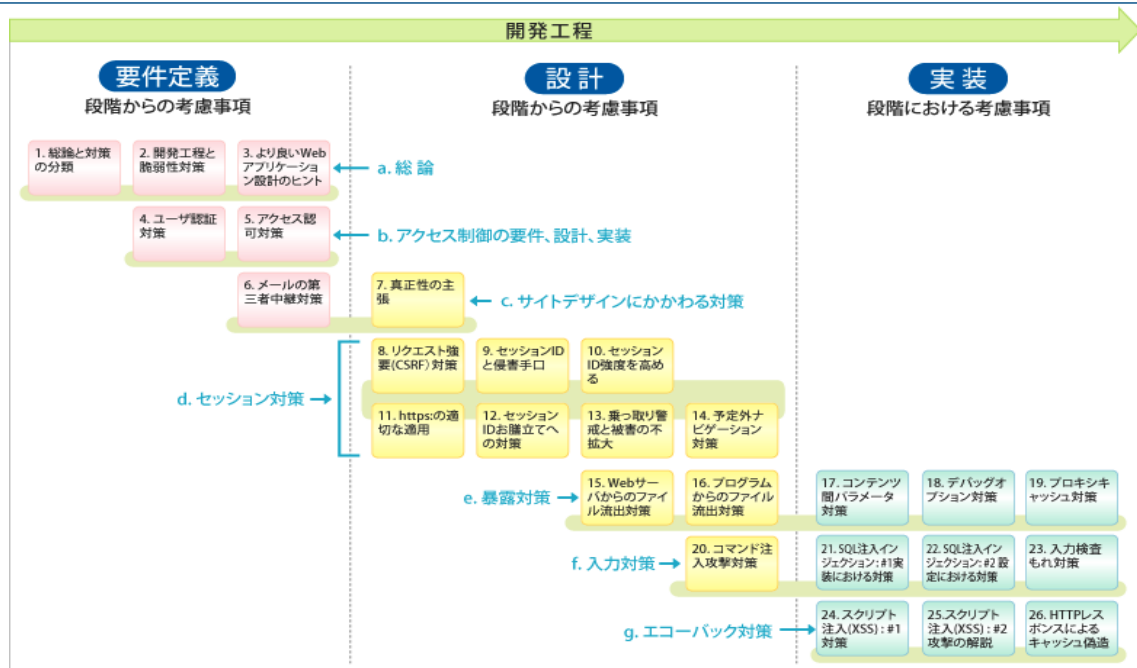
(5) テストと統合

- 実装されたプログラムをテスト計画にもとづいてテストし、段階的に大きな単位に統合する段階

(6) 運用・保守

- 対象システムを運用する段階
- 定期保守、障害発生に伴う臨時保守、業務環境の変化に追従するための保守等があり得る

脆弱性対策項目と工程



要件定義段階からの考慮事項

- 要件定義段階あるいはそれ以前から考慮するとよいもの
 - 総論
 - 総論と対策の分類
 - 開発工程と脆弱性対策(本稿)
 - より良いWebアプリケーション設計のヒント
 - アクセス制御対策
 - ユーザ認証対策
 - アクセス認可対策
 - サイトデザインに関わる対策
 - メールの第三者中継対策

設計段階からの考慮事項

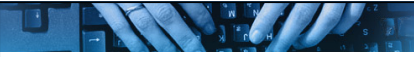
- 設計段階から考慮するとよいもの
 - サイトデザインに関わる対策
 - 真正性の主張
 - セッション対策
 - リクエスト強要(CSRF)対策
 - セッション乗っ取り: #1 セッションID侵害手口
 - セッション乗っ取り: #2 セッションIDの強度を高める
 - セッション乗っ取り: #3 https:の適切な適用
 - セッション乗っ取り: #4 セッションIDお膳立てへの対策
 - セッション乗っ取り: #5 兆候の警戒と被害の不拡大
 - 暴露対策
 - Webサーバからのファイル流出対策
 - プログラムからのファイル流出対策
 - 入力対策
 - コマンド注入攻撃対策
 - SQL注入: #2 設定における対策

実装段階における考慮事項


- 概ね実装段階で考慮するもの
 - 暴露対策
 - コンテンツ間パラメータ対策
 - デバッグオプション対策
 - プロキシキャッシュ対策
 - 入力対策
 - SQL注入 : #1 実装における対策
 - 入力検査漏れ対策
 - エコーバック対策
 - スクリプト注入(XSS): #1 対策
 - スクリプト注入(XSS): #2 攻撃の解説
 - HTTPレスポンスによるキャッシュ偽造攻撃対策

1-2 暴露対策

Webサーバからのファイル流出対策
プログラムからのファイル流出対策
コンテンツ間パラメータ対策
デバッグオプション対策
プロキシキャッシュ対策



1-2-1 Webサーバからの ファイル流出対策



Webサーバからのファイル流出対策

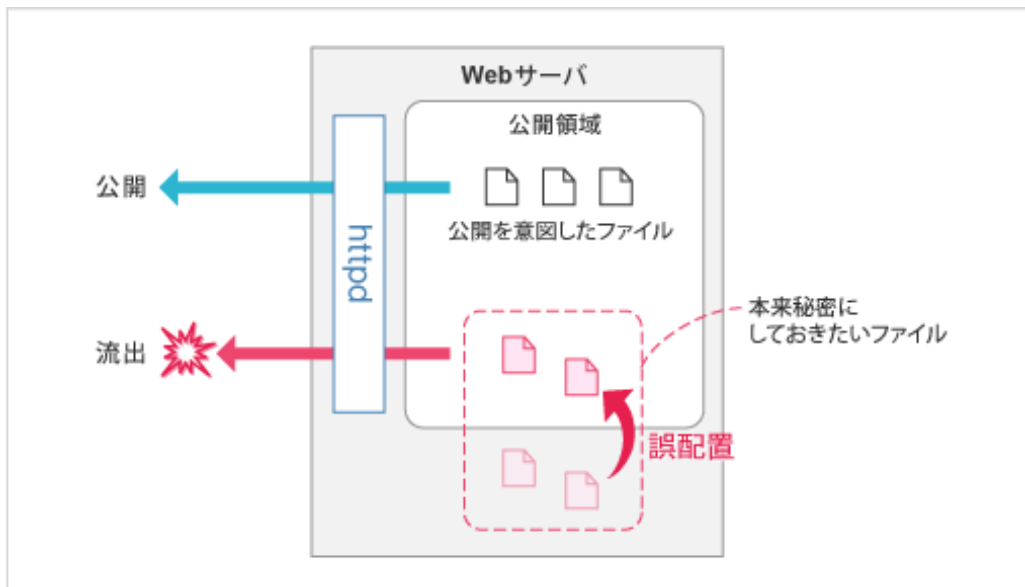
- Webサーバからのファイル流出対策
 - Webサーバコンピュータ上の本来保護されるべき重要なファイルの内容を、インターネット越しに容易に読み出せる場合がある。
- ふたつのWebサーバからのファイル流出
 - 「データファイルの誤った公開」
 - 「Webプログラムソースコードの流出」

データファイルの誤った公開

- データファイルの誤った公開は、個人情報を含む顧客リスト等の重要なデータファイルをWeb公開領域に誤って設置してしまう問題
- 要因1:
 - Webアプリケーションの設計時に問題があると意識せずに公開領域に重要なファイルを置く設計を行った
 - あるいは、Webプログラムの実装時にプログラマの裁量で重要なファイルの置き場所を決めたが、それがWeb公開領域であった
- 要因2:
 - 画像、文書、表計算ワークシート等、ファイルの形をしたコンテンツであって、特定のユーザのみに開示すべきものを、ファイルの形のままWeb公開領域に配置した
- 要因3:
 - Webサーバコンピュータの運用時に、誤って重要なファイルのバックアップコピーをWeb公開領域に置いたままにした

データファイルの誤った公開

- データファイルの誤った公開



データファイルの誤った公開

- 対策1 – [要因1]
 - 重要なファイルはWebサーバ公開領域に置かない設計・実装・設定を行う
- 対策2 – [要因2]
 - 特定のユーザのみに開示すべきコンテンツは、それがファイルの形で存在したとしても、アクセス制御ロジックをもつプログラムを通じて提供するよう、設計・実装する
- 対策3 – [要因3]
 - 一時的バックアップコピーの制限。本番運用しているWebサーバ上に重要なファイルの一時的バックアップコピーを作らないようにする
- 対策4 – [要因1, 要因2, 要因3]
 - 初期設定ならびに運用時、あらかじめリストに掲載されているもの以外のファイルをWeb公開領域に置かないようにする
 - 違反がないか、シェルスクリプト等を用いて定期的に検査することも、ひとつの方法である

Webプログラムソースコードの流出

- JSP、Perl、PHP等、スクリプトの形で記述されるWebアプリケーションプログラムの場合、これらのソースコードが誤ってWebサーバコンピュータから流出することがある。
- 要因4
 - include, require, use等の命令で取り込むためのインクルードファイル名に「.inc」「.pm」等、標準ではWebサーバ(ソフトウェア)やWebアプリケーションサーバがスクリプトとして認識しない拡張子をもたせている
 - これらのインクルードファイルはURLさえ見当がつけばインターネットから閲覧できるおそれがある
- 要因5
 - Webアプリケーションの修正を行った際ソースコードのバックアップ・ファイルが作られて、それがWebサーバコンピュータ上に放置されている。
 - プログラマが意図的に旧バージョンの写しを作る場合
 - テキストエディタが自動でバックアップファイルを作ってしまう場合

Webプログラムソースコードの流出

- 対策5 – [要因4]
 - 標準的な拡張子の使用。インクルードファイルに与える拡張子には、スクリプトの標準的な拡張子(PHPであれば.php等)を用いる。非標準の拡張子(.inc、.pm等)を用いない
- 対策6 – [要因5]
 - 非標準拡張子の動作の定義
 - インクルードファイルに.incのような標準では動作が定義されていない拡張子を与えるのであれば、WebサーバもしくはWebアプリケーションサーバにおいて、この拡張子をもつファイルの内容をブラウザに開示するのではなくスクリプトとして処理するよう定義する
- 対策7 – [要因5]
 - プログラム保守の制限。本番運用しているWebサーバコンピュータにおいてWebアプリケーションプログラムの修正を行わないようにする
 - 別のコンピュータで動作確認を済ませてから必要なもののみファイルの入れ替えを行う

厳格なアクセス許可設定

- Webサーバからのファイル流出に関する共通の対策として、プログラムのソースコード上の論点ではないが、厳格なアクセス許可設定が挙げられる
- 対策8
 - 厳格なアクセス許可設定
 - Webサーバ(ソフトウェア)、Webアプリケーションサーバ、およびそれらの配下で動作するWebアプリケーションプログラムのすべてを権限の小さなアカウントで稼働させる。
 - Webサーバコンピュータ内のすべてのファイルには、次のものを除き、Webサーバ(ソフトウェア)、Webアプリケーションサーバ、Webアプリケーションプログラムのそれぞれのアカウントからのアクセスを一切許さない設定を施す
 - Web公開領域上で一般に公開するファイル。これらには、Webサーバ(ソフトウェア)のアカウントに対し読み出し許可を与える
 - Webプログラムが内容を読み取る必要があるファイル。これらには、Webアプリケーションプログラムのアカウントに対し読み出し許可を与える
 - Webサーバ、Webアプリケーションサーバの諸設定ファイル。これらにはWebサーバ、Webアプリケーションサーバのアカウントに対する読み出し許可をそれぞれ与える

対策と開発フェーズ

- ファイル流出の対策は、Webアプリケーション設計、プログラム実装、サーバ設定、および運用の4つのソフトウェア開発フェーズに対し、次のように関連する

－ 凡例:

- ◎ 大きな関連がある
- △ 関連がある
- － 関連がない

	設計	実装	設定	運用
対策1	◎	△	◎	－
対策2	◎	◎	△	－
対策3	－	－	－	◎
対策4	△	－	△	◎
対策5	◎	－	△	－
対策6	△	－	◎	－
対策7	－	－	－	◎
対策8	△	△	◎	△

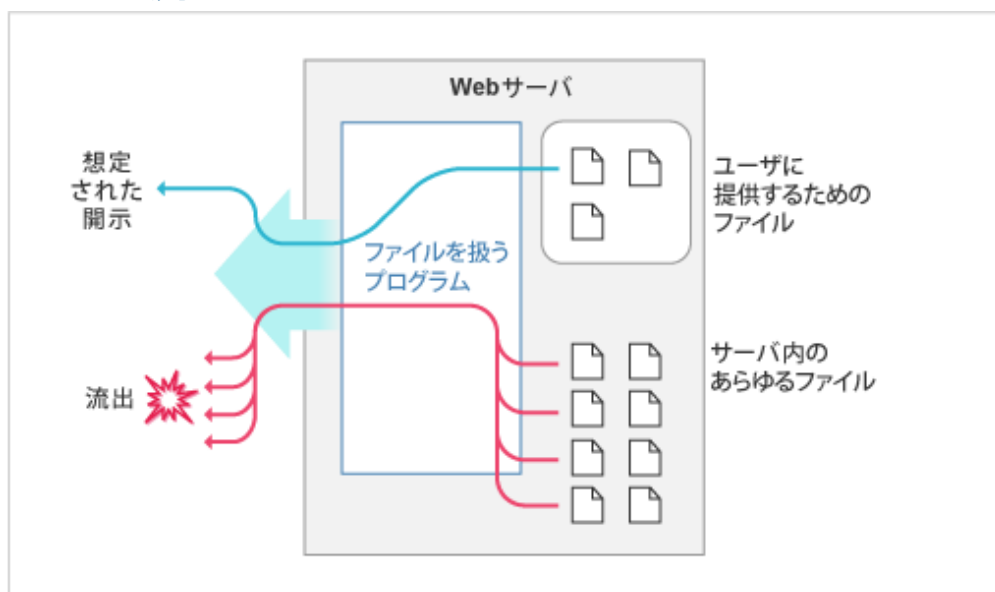
1-2-2 プログラムからの ファイル流出対策

プログラムからのファイル流出対策

- 保護されるべき、Webサーバコンピュータ上の重要なファイルの内容が、何らかのミスによってインターネットから容易に読み出せるようになっているのがファイル流出問題である
- ファイル流出はふたつのタイプに分かれる
 - 「Webサーバソフトウェアからのファイル流出」
 - 「Webアプリケーションプログラムからのファイル流出」

ファイル流出

- ファイル流出



プログラムからのファイル流出

- Web アプリケーションプログラムからのファイル流出は、ファイル名をパラメータとして受け取って、その内容を表示するWebプログラムに不備があることから起こる
- 工夫した値のファイル名パラメータを与えることによって、インターネット上の人物が誰でもサーバコンピュータ内の任意のファイルを読み出せる問題である
- ふたつのバリエーション
 - フルパス名を受け入れてしまう問題
 - ディレクトリトラバーサル攻撃による流出

フルパス名を受け入れてしまう問題

- カレントディレクトリ内のファイルをオープンすることを想定し、ディレクトリ修飾を付けることなくファイル名のみを用いてファイルをオープンしようとするプログラムは、実はフルパス名を与えられると、そのパス名が表すファイル内容を流出させてしまう
- 例えば、次のPerlスクリプト、

```
open(HANDLE, "<$filename");
```

- の\$filenameに「/etc/passwd」等のパス名を与えられて、ファイル流出が起こる

ディレクトリトラバーサル攻撃による流出

- ディレクトリトラバーサル攻撃というのは、「../」等の親ディレクトリを示す表記をファイル名パラメータの中に混入し、サーバ内の任意のファイルを読み出そうとする攻撃である
- ファイルをオープンする際、与えられたファイル名の先頭にディレクトリ修飾を追加してパス名を組み立てている場合でも、ユーザから与えられたファイル名の中に「../」あるいは「..\」の親ディレクトリを示すパターンが含まれていると、予定外のファイル内容が流出する
- 例えば、次のPerlスクリプト、

```
dir = "/var/data1";  
open(HANDLE, "<$dir/$filename");
```

- の\$filenameに「../etc/passwd」等のパス名が与えられて、ファイル流出が起こる。

プログラムからのファイル流出

- 対策1: ファイル名パラメータを警戒
 1. 慎重な設計・実装
 - ファイル名をパラメータとして受け取るプログラムは要注意プログラムである
 - そのようなプログラムをWebサイトに置くことを決めた場合は、経験豊かな技術者を割り当てて、慎重に設計・実装する
 2. プログラム自己裁量の禁止
 - それと並行して、事前に計画されていない、プログラム自己裁量によるファイル名パラメータの新たな設置を禁止し、違反を警戒する

プログラムからのファイル流出

- 対策2: ファイル名パラメータの検査
 - ファイル名をパラメータとして受け取るプログラムは、次の入力検査を行う
 1. Unix, GNU/Linuxの場合
 - 可能なら、ファイル名パラメータの仕様として、ディレクトリ修飾のあるファイル名(「/」を含むパス名)を禁止するものとし、その仕様に沿った入力検査を行う
 - 「/」ではじまるパス名(絶対パス名)を受理しない
 - 「./」(親ディレクトリ修飾)を含むパス名を受理しない
 2. Windowsの場合
 - 可能なら、ファイル名パラメータの仕様として、ディレクトリ修飾のあるファイル名(「/」もしくは「\」を含むパス名)を禁止するものとし、その仕様に沿った入力検査を行う
 - 「英字:\」ではじまるパス名(絶対パス名)を受理しない
 - 「\」ではじまるパス名(絶対パス名のバリエーション)を受理しない
 - » Windowsにはプレフィックス「\\?」を用いた「\\?\c:\foo.bar」のようなパス名表記があり「c:\foo.bar」とほぼ同じ意味をもつ。そのようなパス名を排除する
 - 「./」もしくは「..\」(親ディレクトリ修飾)を含むパス名を受理しない
 - ライブレットの直後《以外》に「:」を含むパス名を受理しない
 - » Windowsのファイルシステム形式のひとつNTFSのパス名には「:」を用いて副次ストリーム名を示す表記法があり、それを用いたパス名を排除する

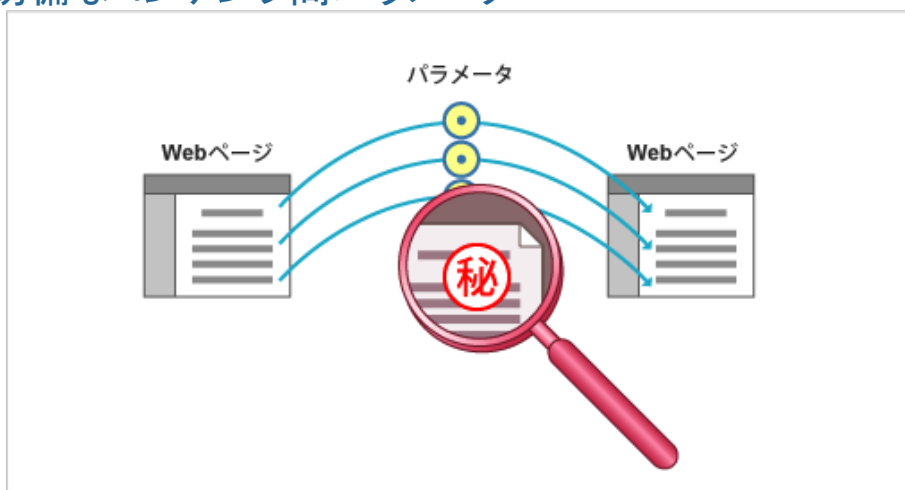
1-2-3 コンテンツ間パラメータ対策

コンテンツ間パラメータ対策

- コンテンツ間パラメータ
 - Webアプリケーションの場合、連鎖するURLのそれぞれには、いくつかのパラメータが伴う
 - 呼び出されたWebプログラムはこれらのパラメータを読み込んで柔軟な情報処理を行う
 - あるWebページから別のWebページに移り変わる際HTTPリクエストに添えられるパラメータを、ここでは「コンテンツ間パラメータ」と呼ぶことにする
- 3種類のコンテンツ間パラメータ
 1. URLパラメータ(クエリストリングパラメータ)
 - Webプログラム呼び出しのURLの中に含まれるパラメータ
 2. POSTパラメータ
 - Webプログラム呼び出しのHTTPリクエストのボディ部に含まれるパラメータ
 3. Cookie
 - Webサーバ側がブラウザに預けておくことのできる小さなデータ

コンテンツ間パラメータに生まれる脆弱性

- コンテンツ間パラメータは、第三者への漏えい・改ざん、ユーザ本人によって改ざんされうる無防備な存在である
- 無防備なコンテンツ間パラメータ



第三者への漏えい

- 平文通信による漏えい
 1. 平文通信 (http:) を用いているために URL パラメータ、POST パラメータ、Cookie が第三者に傍受される
- 暗号通信 (https:) を使用していたとしても起こり得る漏えい
 2. URL パラメータがキャッシュやログに残留し、そこから漏えいする
 - ブラウザのキャッシュ
 - プロキシサーバのキャッシュ
 - ファイアウォールのログ
 - Webサーバのアクセスログ
 3. URL パラメータが Referer: ヘッダを通じて別 Webサーバに傍受される
 - ハイパーリンクに埋め込んだパラメータが第三者に傍受される
 - フォームデータが URL パラメータで送信され、傍受されるケースもある
 - <form> タグに method="post" 属性を明示しなかった場合
 4. Cookie が別サーバに傍受される
 - Cookie 発行の際の domain 属性の値が広いドメイン範囲を指しているとき起こる
 5. Cookie が別アプリケーションに傍受される
 - Webサーバに複数のアプリケーションが同居しており、Cookie 発行の際の path 属性の値が / 等の広すぎる範囲であるとき起こる
 6. 平文通信が混在していて Cookie が第三者に傍受される
 - Webアプリケーションの中に暗号通信を使っていない部分 (http:) が混在していてかつ、Cookie 発行の際 secure 属性があたえられていないとき起こる

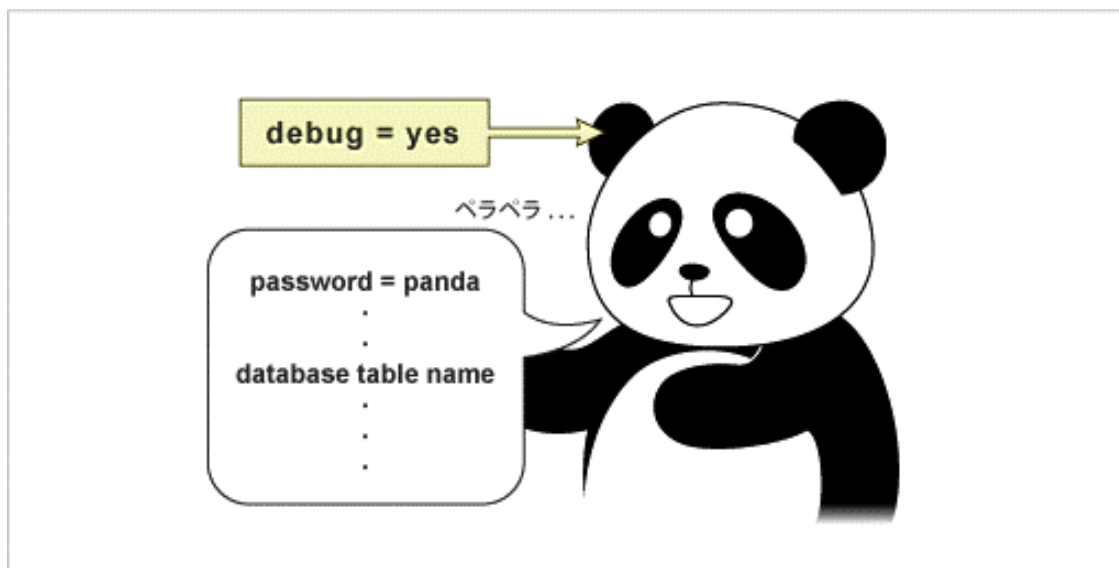
ユーザ本人による改ざん

7. URL パラメータ、POST パラメータ、Cookie のいずれかに含まれる個人識別に関わるパラメータを改ざんして別人になりすまし、Webアプリケーションを不正にオペレーションする
8. 同じくユーザの権限に関わるパラメータを改ざんして本来許可されないコンテンツにアクセスする
9. 同じくリソースの識別に関わるパラメータを改ざんして本来許可されないコンテンツにアクセスする

コンテンツ間パラメータ対策

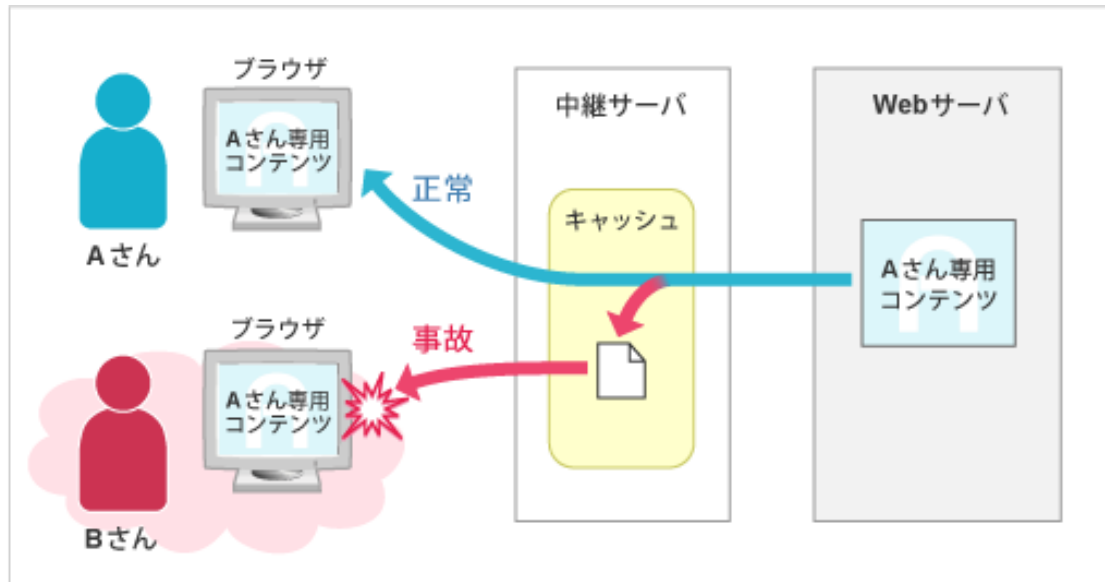
- コンテンツ間パラメータに関わる問題を回避するには次のようにWebアプリケーションを実装する
- 対策1: パラメータ用途の限定
 - コンテンツ間パラメータ(URLパラメータ、POSTパラメータ、Cookie)には、第三者に傍受されては困る秘密情報を含めない
- 対策2: セッション変数の使用
 - コンテンツ間パラメータの受け渡しにはなるべくセッション変数を用いる
- 対策3: <form>タグへのmethod="post"の明記
 - フォームデータがURLパラメータではなくPOSTパラメータで送信されるよう、<form>タグには必ずmethod="post"を明記する
- 対策4: Cookie属性の厳密な指定
 - Cookieの属性をより厳しい条件に設定する
 - domain、path、max-ageまたはexpires、secure

1-2-4 デバッグオプション対策



1-2-5 プロキシキャッシュ対策

- キャッシュによる情報漏えい問題



ご質問をどうぞ

Q & A