

Web アプリケーション セキュリティとは

長谷川 武

セントラル・コンピュータ・サービス株式会社
セキュリティソリューション部

2003年4月1日

この記事は、株式会社バガボンド発行の小冊子「Scan Security Handbook Vol .9, Dec 2002b アプリケーションセキュリティ」向けに2002年10月に執筆したものです。今回、独立した資料として再編集しました。当記事の著作権はセントラル・コンピュータ・サービス株式会社が、当記事の著作人格権は長谷川武がそれぞれ有しています。この記事の複製・再配布は、非営利目的かつ一切の改変を加えない形に限り自由です。

1. Web アプリケーションのセキュリティ問題

1.1 Web アプリケーション

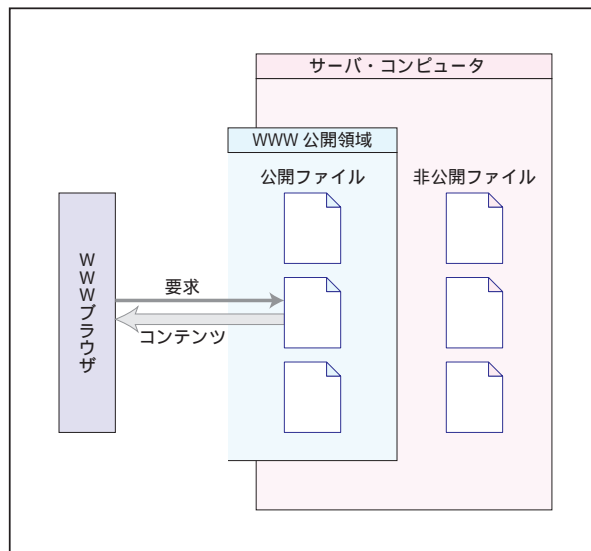
「Webアプリケーション」とは、WWWサーバ側で何らかのプログラムを動かすようになっている一連のコンテンツのことである。こうしたコンテンツではWWWブラウザから送り込んだデータを受け取り、それに応じた動作をするため、高度な情報サービスの提供が可能だ。

現在、オンライン・ショッピング、インターネット・オークション、インターネット・バンキング、電子掲示板、求人への応募、企業への資料請求など、さまざまなWebアプリケーションがインターネットで稼動している。

1.2 静的コンテンツから動的コンテンツへ

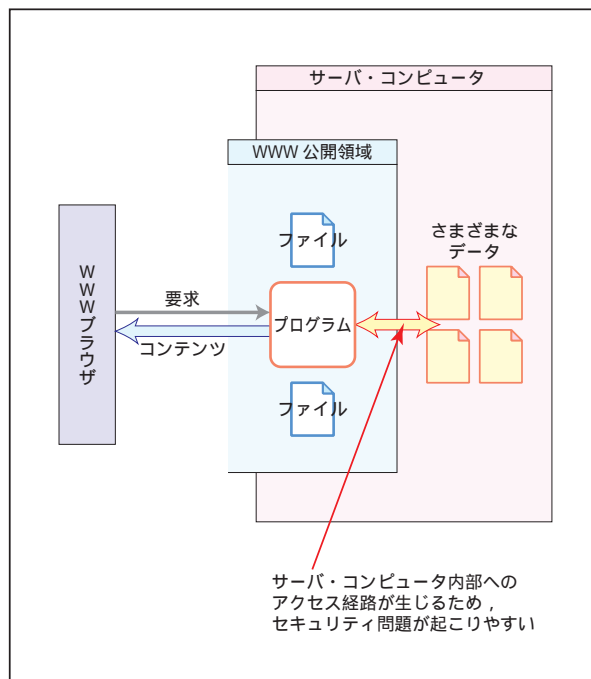
もともとのWWW (World-Wide Web) システムは、サーバ・コンピュータの中の一定のファイルを外部のユーザに公開するという単純な仕組みから始まった (図1)。

図1 静的 WWW コンテンツ



これを少しだけ拡張し、WWWサーバに特定の形のURLを送るとファイルの代わりにプログラムを呼び出す、という仕組みが作られるようになった (図2)。

図2 動的 WWW コンテンツ



WWWブラウザの操作にWWWサーバ側のプログラムが対応するようにすることでWWWシステムの能力は飛躍的に高まった。当初CGI(Common Gateway Interface)という形式でのプログラム呼び出しの仕組みからはじまった動的コンテンツのサポートは、Active Server Pages、Java Servlet などさまざまな発展を遂げている。

ところが、ここに問題がある。動的コンテンツの場合、プログラムがサーバ・コンピュータの中のファイル、データベース、ネットワーク機能など、さまざまな資源にアクセスする「アクセス経路」ができあがっている。プログラミングにミスがあると、この経路を伝って外部から侵入されてしまうことがあるのだ(図2の注意書き)。

1.3 Web アプリケーションに起こる問題

さまざまな理由から、Web アプリケーションには次のような問題が起こる。

- (1) 蓄積している個人情報や取引情報が広く一般に流出する
- (2) オンライン・ショッピングで不正をはたかれる
- (3) サーバ内のデータベースが改竄されたり破壊される
- (4) Webページの内容が改竄される
- (5) ユーザ認証メカニズムが迂回される
- (6) 正規ユーザの会話セッションが別人に乗っ取られる
- (7) サーバ・マシンが乗っ取られる

1.4 攻撃の手口

こうした問題は次のような手口を使って攻撃されることから起こる。

- (1) サーバが不用意に露出しているファイルから情報を盗まれる
- (2) サーバが不用意に露出している項目から重要なキー値が類推されデータの呼び出しに悪用される
- (3) サーバが不用意に露出している価格などの項目が改竄される
- (4) 不正なSQL文が混入される
- (5) 不正なOSコマンドが投入される
- (6) クロスサイトスクリプティング攻撃を受ける
- (7) プログラムへのパラメタが操作されプログラムの振る舞いが攪乱される
- (8) プログラムへのパラメタにバッファオーバーフローが仕組まれる
- (9) プログラムのデバッグ機能が部外者に悪用される
- (10) サーバ・システムの重要なファイルが流出する
- (11) WWWサーバ・ソフトウェア、WWWサーバとともに使っている各種パッケージ・ソフトウェア、サーバ・マシンのOSの脆弱性や不用心な設定が攻撃される

1.5 対策の指針

さまざまな脆弱性(セキュリティ上の弱点)が生じないようにWebアプリケーションを構築するには、少なくとも次のような指針をもつ必要がある。

- (1) ファイルの露出を減らす 外部に流出しては困るファイルをWWWブラウザから「確実に見えない」場所に配置する。
- (2) パラメタの露出を減らす WWWサーバからWWWブラウザに渡すデータを厳選する。
- (3) セッション機構 ページ間のデータ受け渡しはサーバ側の「セッション維持機構」を利用する。
- (4) 暗号化 重要な情報を取り扱う場面ではSSLを使用する。暗号化区間で使うセッションIDの安全対策を行う。
- (5) 分岐パラメタの無害化 複数のページの中から一つを選ぶ場面で「生の」キーをページ呼び出しのパラメタに使わない。
- (6) 入力すべてを疑う WWWブラウザから送られてくるデータ項目はすべて改竄されたものであるおそれがある。これらすべてをチェックする。
- (7) クロスサイトスクリプティング対策 ページ上でスクリプトを形成しうる特殊文字[<」「>」「"」「'」)を入力させず出力もしない。
- (8) 不正SQL混入対策 SQL文の構造に干渉できる特殊記号['」「;」)を入力させない。
- (9) システムコマンド不正投入対策 Linux(および Unix)のシェルコマンドで使われる特殊記号([';」「|」「&」「`」「(」「)」など)を入力させない。
- (10) WWWサーバ・ソフトウェア, WWWサーバとともに使う各種パッケージ・ソフトウェア, サーバ・マシンのOSについて脆弱性対策を施すとともに, デフォルトの設定をより安全なものに変更しておく。

注 これらは大まかな指針である。それぞれのプログラミングの場面で注意すべき点については情報処理振興事業協会セキュリティセンターの『セキュア・プログラミング講座』(<http://www.ipa.go.jp/security/awareness/vendor/programming/index.html>)の記事などを参考にしていきたい。

2. これだけは避けよう 初歩的なミス

最近では「クロスサイトスクリプティング」などの不正手口が広く知られるようになり、Webアプリケーションの開発時点でさまざまな対策がとられるようになってきた。

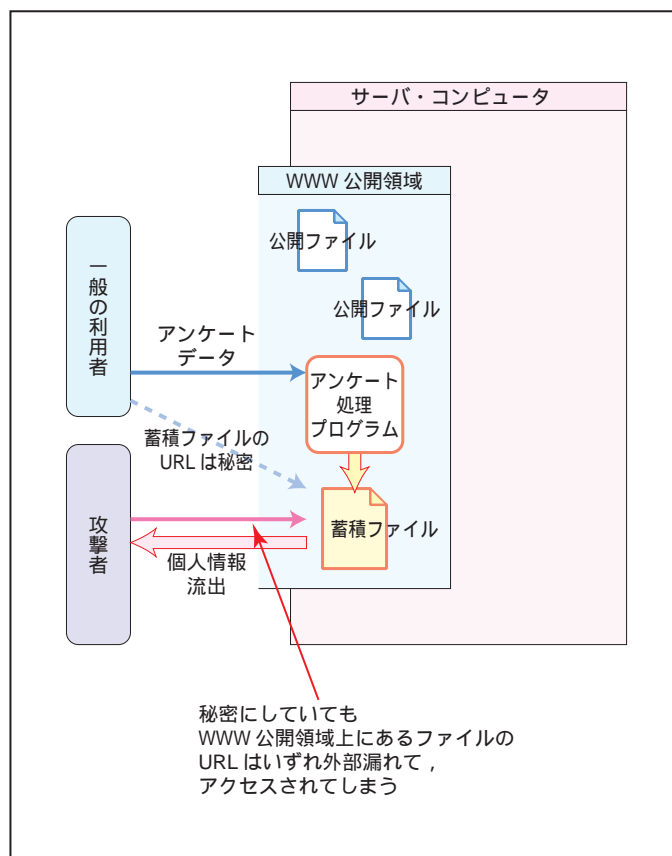
その一方で、ごく普通のユーザからでも簡単に攻撃を受けてしまうような初歩的なミスを犯すWWWサーバもまだまだ後を絶たない。

Webアプリケーションのセキュリティ対策の最初のステップとして、次のような初歩的なミスの回避に努めたい。

2.1 データの危険な蓄積場所

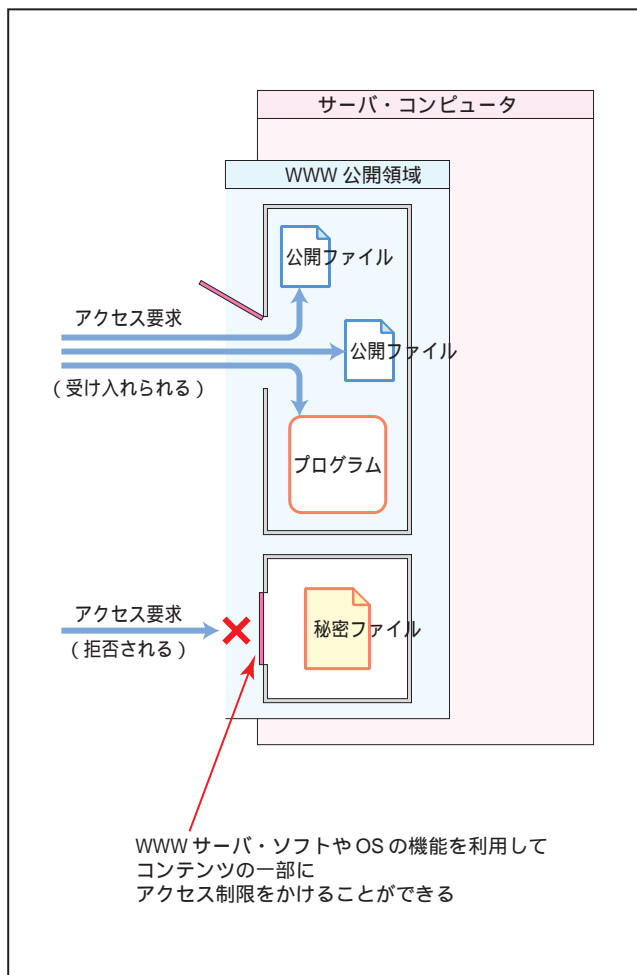
消費者からのアンケート回答を蓄積するといったWebアプリケーションがある。こうしたアプリケーションが不用心にもデータの蓄積ファイルの置き場所としてWWWサーバがコンテンツを公開しているディレクトリの上に置いていることがある。この「データ蓄積ファイル」の存在はユーザには知らされていないから一見安全に見える。しかし、データファイルのパス名が推定され、見つめられると電子掲示板などを通じて不特定多数の人々に情報が伝わり、事実上「誰にでも見える」状態になってしまう(図3)。

図3 重要なファイルが丸見えに



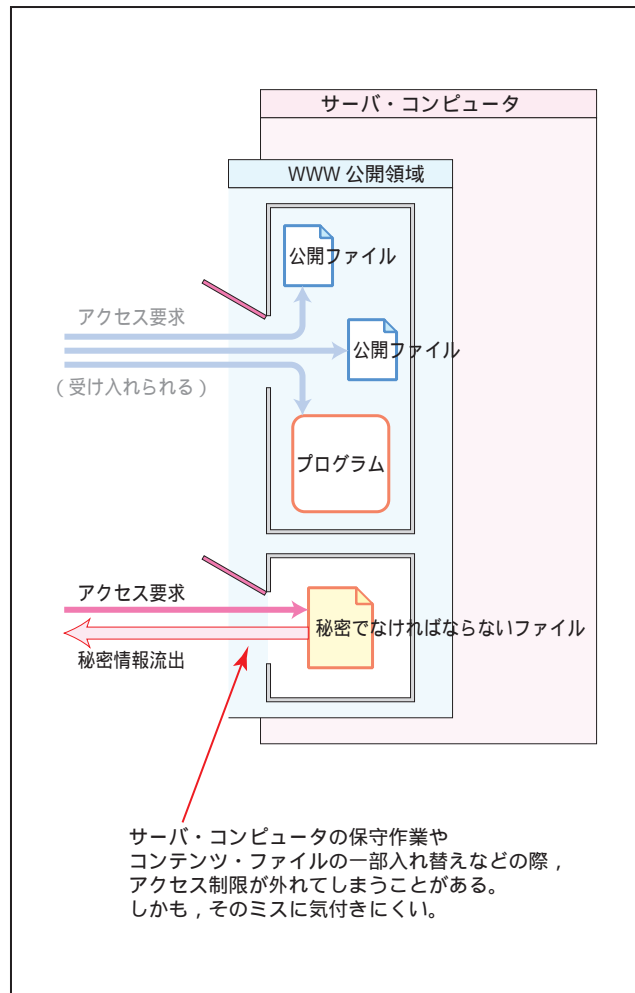
何の保護も講じずにファイルを置いておくような不用心なやり方はまずいと考える人はさすがに多くて、WWWサーバの公開領域に置いていながら、WWWサーバやOSのアクセス制御機能を利用してファイルを保護している Web アプリケーションもある（図4）。

図4 部分的なアクセス制限



ところが、サーバ・コンピュータの保守や、WWW コンテンツ・ファイルを入れ替えたりする際にこのアクセス制御がデフォルトの状態、すなわち「アクセスを許す」状態に戻ってしまうことがある。しかも、アクセス制限が解除されていることにはなかなか気づきにくい。結局重要なファイルが「丸見え」の状態でも長時間放置されることになる（図5）。

図5 アクセス制限が外れていても気づきにくい

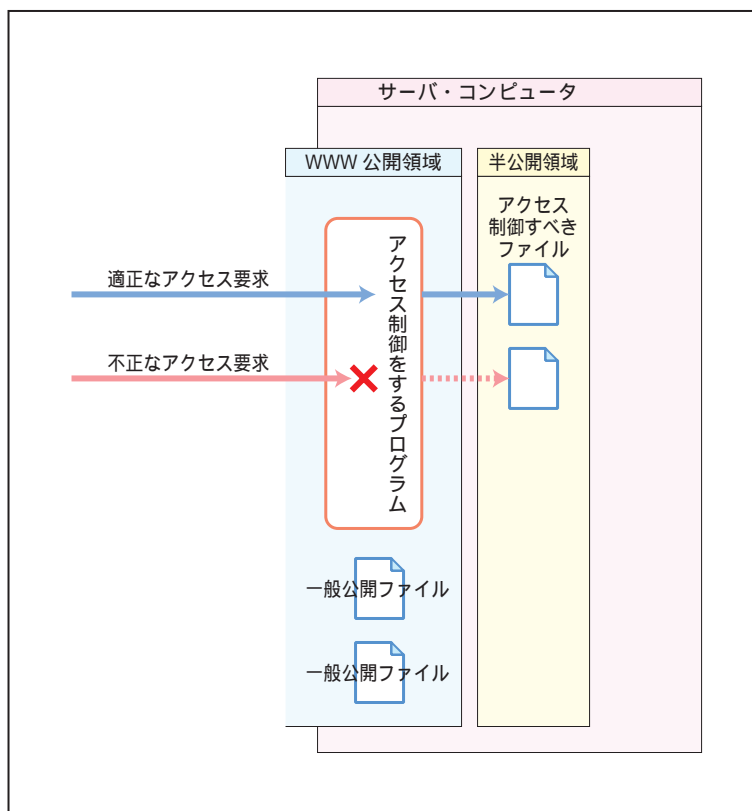


たとえアクセス制限をかけたとしても小さなミスで破綻してしまうことがあるから、WWW 公開領域に重要な秘密ファイルを置くことは危険である。必ず外部からアクセスできない場所に配置すべきだ。

2.2 危険な静的コンテンツ

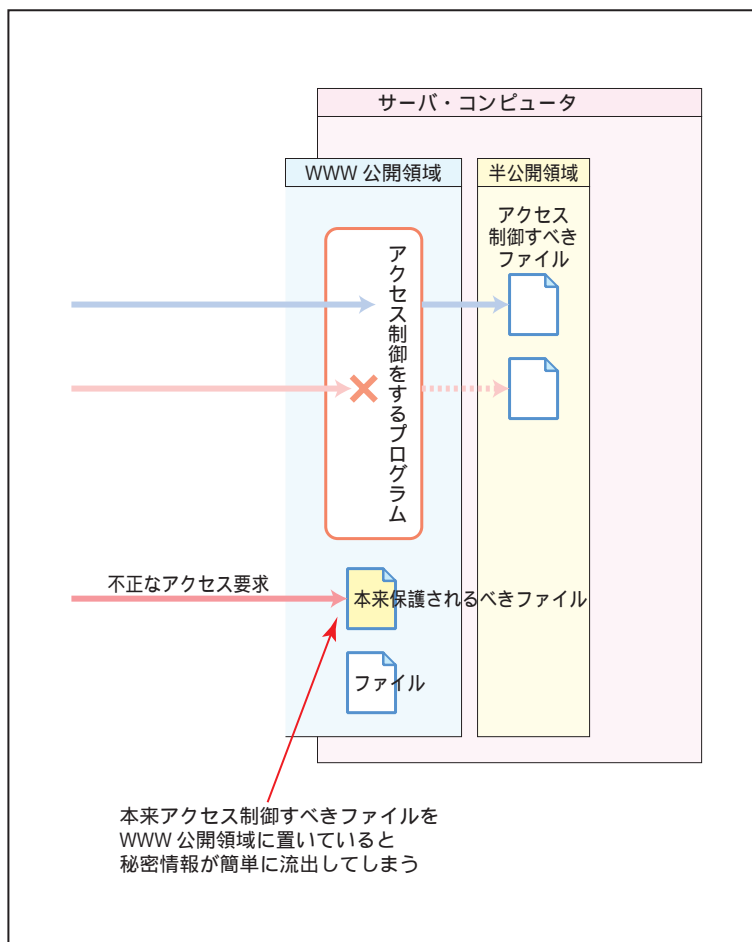
ある会員向けWebサイトでは、ユーザIDとパスワードによるユーザ認証の仕組みを使っていた。こういうサイトの場合、特定の会員しか見ることのできないページや画像を動的コンテンツ（プログラム）を経由して提供し、このプログラムによって適切な保護が行われるようにするものである（図6）。

図6 プログラムでガードする



ところが、このサイトでは会員の顔写真のJPEG画像をWWWサーバから送り出す際に動的コンテンツを経由させることなく、JPEGファイルの静的コンテンツとして提供していた。これらのJPEGファイルのファイル名の付け方には規則性があり、不特定多数の人々がこれらの画像ファイルを観覧できる状態になっていたのである（図7）。

図7 ガードし忘れ

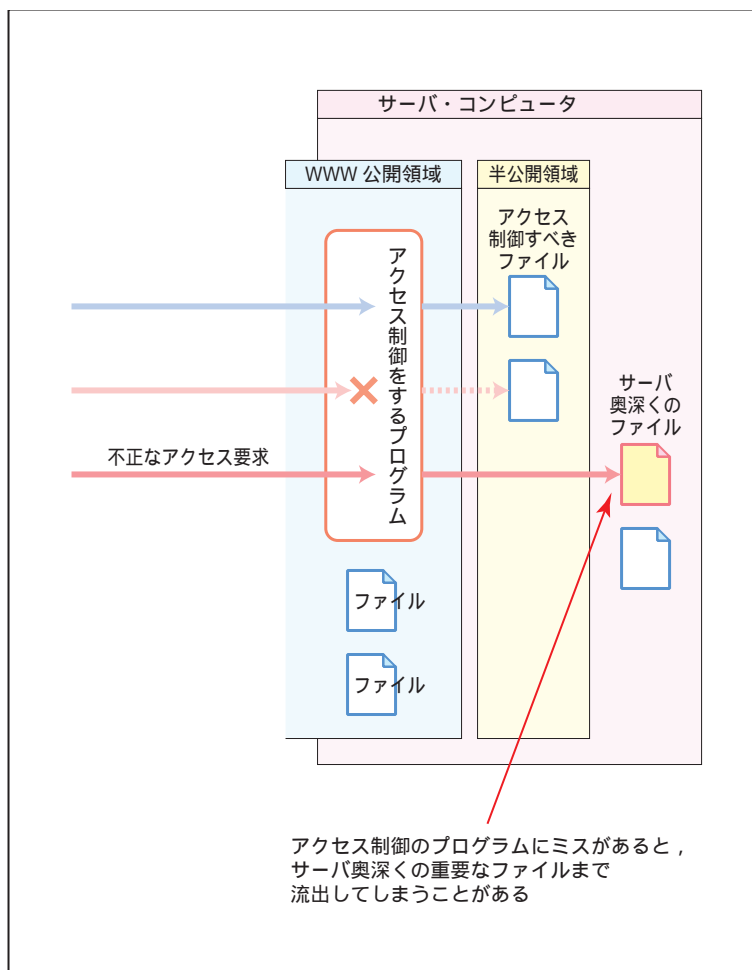


ユーザ本人以外に見せてはならない守秘義務のあるコンテンツは、画像や各種ファイルも含めて、すべてプログラムによる保護機構を経由させて提供すべきである。

2.3 ガードにしくじる

あるWebサイトでは、画像ファイルについてもプログラムによる保護機構を経由して提供していた。ところが、この保護機構のプログラムは表示させる画像ファイルのファイル名を受け取るパラメタのチェックが甘く、特定のパターンのパス名を指定されるとサーバシステム内の奥深くにあるファイルの内容を暴露してしまうという問題を抱えていた（図8）。

図8 ガードにしくじる



3. さまざまな問題

上記の初歩的な問題を手始めに，Webアプリケーションには次のようなセキュリティ問題が起こり得る。

3.1 「見えてしまう」問題

3.1.1 パラメタを露出しているミス

後続のページを呼び出すリンクのクエリストリングに重要なキーが含まれていると，それはすぐユーザの目にとまり，キーを類推する機会を与えてしまう。都合の良いキーの値が指定され，ページが呼び出されて，秘密の情報が部外者に漏れてしまう。

クエリストリングに指定されたパラメタは，「直前のページ」を示す Referer:ヘッダを通じて別の WWW サーバに傍受されたり，プロキシサーバのログに記録されるなど露出する機会が多い。重要なパラメタはクエリストリングに載せないようにすべきである。

3.1.2 「見えない」つもりのミス

Web の HTML フォームの入力項目のなかにはページ上に表示されない hidden (ヒドゥン) フィールドというものがある。hidden フィールドはページ間でデータを受け渡すには便利であるが，ここにおかれたデータは全く保護されていないと考えるべきである。

ユーザは WWW ブラウザの簡単な操作で「隠されている」はずの hidden フィールドの値を見ることができし，それを自分の都合の良いように書き変えてしまうことも難しくない。

hidden フィールドだけでなく，ラジオボタン，チェックボックス，選択肢の項目についても同様の注意が必要だ。これらの各項目にはプログラムによる処理のためにそれぞれ値が割り当てられているのだが，WWW ブラウザを使いこなしているユーザにとってはそれらも「丸見え」なのである。

3.1.3 ソースコードの流出

WWW サーバの公開領域に，スクリプトプログラムのバックアップファイルを置いたまま消し忘れていたり，サーバ側で動いているプログラムの一部分が外部に漏れることになる。

また，WWW サーバやアプリケーションサーバのバグによりスクリプトのソースコードが露出してしまふことがある。こうして，ソースコードが外部に漏れることにより，有力な手がかりを攻撃者に与えてしまうことがある。

3.2 ユーザ認証とセッションの問題

3.2.1 クエリストリングのユーザID

先に述べた「パラメタを露出しているミス」で最もやってはならないのは、ログインで守られているはずのページを呼び出す URL のクエリストリングの中にユーザID を含めてしまうことである。

このようにしていると、認証ページを経由せずにユーザID を指定して直接そのページを呼び出すことが可能になってしまう。有効なユーザID が1つ外部に漏れれば、誰でもそのページを呼び出せるようになる。

3.2.2 見えないつもりのIDの漏洩

クエリストリングではなく、hidden フィールドやCookie にユーザID を記録してそれに依存して動作する Web アプリケーションについても、クエリストリングを使ったときと同様、ユーザ認証ページを迂回される問題が生じる。

3.2.3 SSL 区間とセッションIDの付け替え

Web アプリケーションを構築するための各種の環境 (Java Servlet, Active Server Pages, PHP など) では、セッション管理の機能が備わっていてプログラムから簡単にそれを利用できる。

こうした環境を利用する場合、設定にもよるが、ユーザが当該 Web アプリケーションのどれかのページに最初にアクセスした際にセッション用の記憶域が確保されセッションIDが発行されることが多い。

ログインする前の何ページかはSSLを使わず、ログインのページから先の通信をSSLで保護するような Web アプリケーションの場合、ログインの時点でセッションIDを付け替える必要がある。

セッションIDの付け替えを行わなければ、最初の非SSL区間で第三者にセッションIDが傍受されるとその後SSLで通信を暗号化するかどうかに関わりなくセッションの乗っ取りを許してしまうからである。

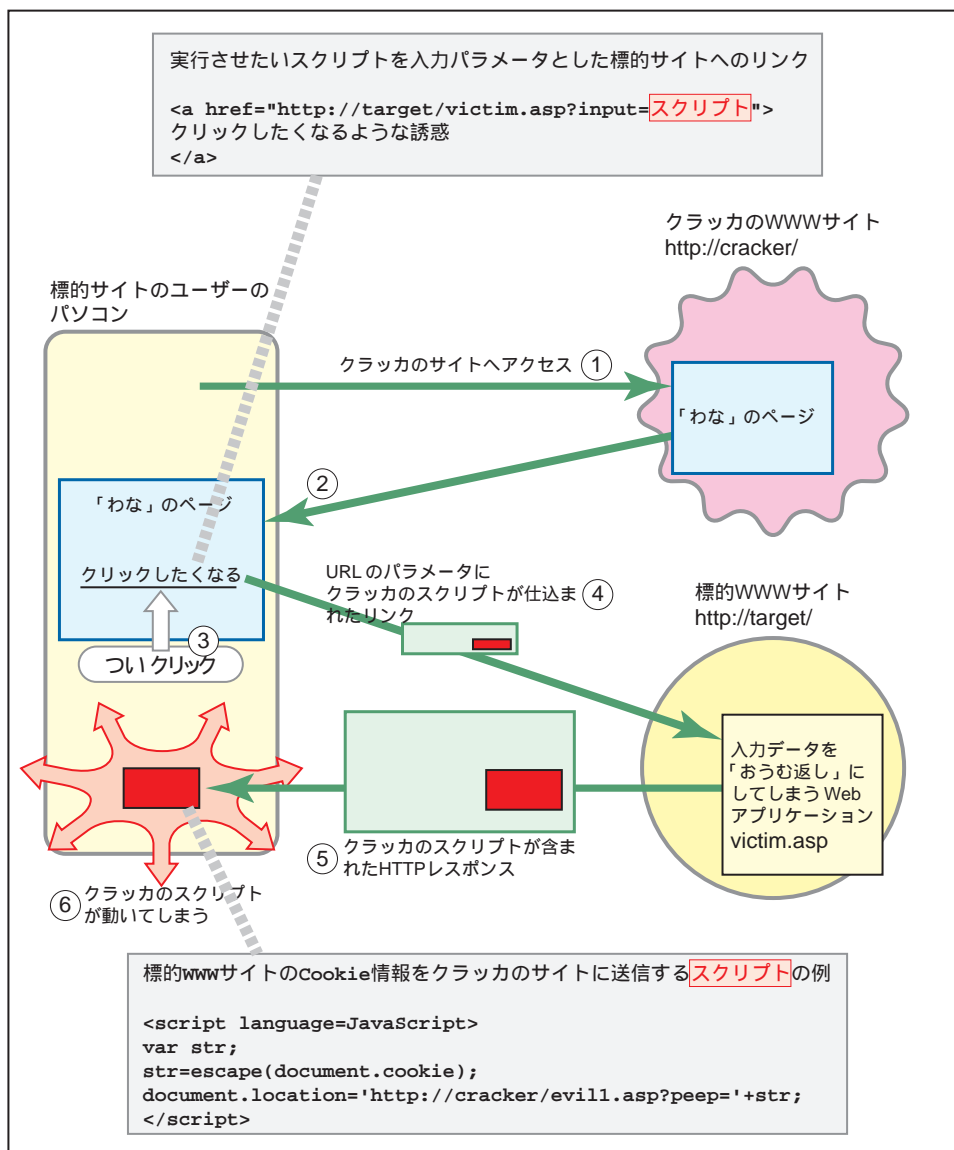
3.2.4 クロスサイトスクリプティング攻撃

スクリプトをダウンロードつもらのないWebサイトに、他のWebサイトから干渉して悪意のJavaScriptプログラムをWWWブラウザにダウンロードさせてしまうという攻撃手法がある。本来のWebサイトとは別のサイトからスクリプトを送り込むというところから、この攻撃の手口は「クロス・サイト・スクリプティング」と呼ばれている。

入力データの値を加工せずに「おうむ返し」してしまうページをもつWebアプリケーションはクロスサイトスクリプティングに悪用されるおそれがある。

ユーザのWWWブラウザに悪意のJavaScriptプログラムが送り込まれると、軽度の被害としては、正規のWebサイトのページがあたかも改竄されたかのようにユーザに見せられるといったことが起こる。最悪の場合、Cookieに保持されていたWebアプリケーションのセッションIDが盗まれ、そのユーザとWWWサーバとの間の会話セッションが乗っ取られる（図9）。

図9 クロスサイトスクリプティング



3.3 SQL 文や OS コマンドを不正投入される問題

3.3.1 SQL 組み立て上のミス

メタキャラクタ「'」がデータに含まれている場合の対策が施されていないと、プログラムの中でSQL文を組み立てる部分が攪乱され、攻撃者に都合の良いようなSQL文の断片が挿入されてしまう。

その結果、認証の迂回、情報の流出、データの破壊などが起きる。

3.3.2 OS コマンド不正投入攻撃

ユーザからの入力データを埋め込んでOSコマンドを組み立てて実行するような場面で、さまざまな機能を持つ特殊文字（「;」「f」「&」「'」「(「)）」などが混入されていることに対して対策が施されていないと、こうした特殊文字の機能が悪用されて、攻撃者に都合の良い不正なOSコマンドが送り込まれ、実行されてしまう。

最悪の場合、サーバマシンが乗っ取られることになる。

3.4 そのほかの問題

3.4.1 スпамメールの踏み台

任意の宛先に電子メールを発信できる機能を持っているWebアプリケーションは、そのページが悪用されて不特定多数へのスパムメールの発信に使われるおそれがある。

4. Webアプリケーションはなぜ危ないのか

Webアプリケーションは、セキュリティ上の問題が起きやすい次のような特性を持っている。

- (1) WWWブラウザに対して開かれている基本構造をもつ
- (2) さまざまな技術が使われていて構成が複雑である
- (3) 開発している人々が熟練していないことがある
- (4) 運用している人々がセキュリティ対策を十分に実施していない

4.1 ブラウザに対して開かれている

Webアプリケーションはその構造上いくつか課題を抱えている。

アプリケーションの「会話処理の流れ」にあたる「ページの流れ」(図10)を維持するための一定の情報をWWWブラウザに預けつつ動作する必要がある点である(図11)。

図10 ページ進行

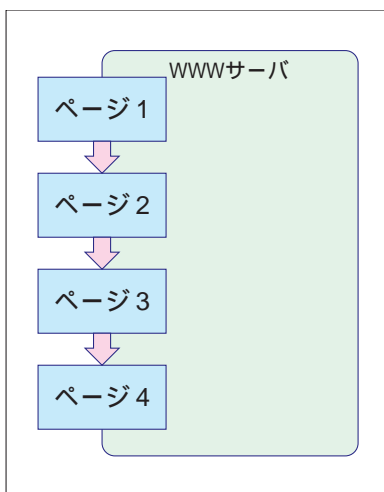
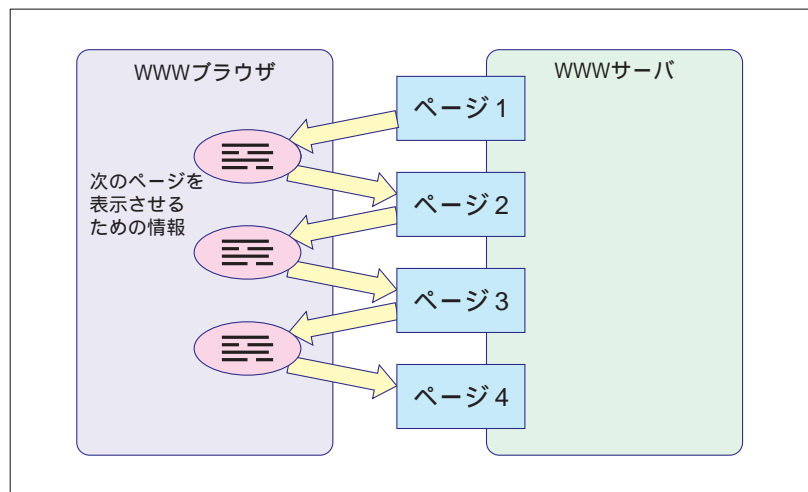


図11 WWWブラウザに情報を預ける



以前のWebアプリケーションでは、顧客番号、商品番号、単価、数量といったデータがhiddenフィールド(HTMLページには現れないが次の送信の際にはサーバに送られる特別なフォーム入力項目)としてWWWブラウザに預けられていることすらあった。

これらのデータは通常ユーザの目に触れないが、WWWブラウザの簡単な操作で読み出すことができ、しかも値を書き変えてしまうことも難しくない。

さすがに現在では重要なデータはWWWサーバ内に「セッションデータ」として保持しておき、その「セッション」を識別するセッションIDのみをWWWブラウザに預けるといった手だてが講じられているのが普通だ。

他の情報を預けないようにしても、このセッションIDだけはWWWブラウザに預けておく必要がある。

それは、World-Wide Webの通信プロトコルであるHTTPがWWWブラウザとWWWサーバの間の一回分の通信（リクエストとそれに対する応答）についてしか対応を維持するように設計されていないため、複数のWebページの連鎖を維持するには最小限1つの識別データをWWWブラウザに預けておく必要があるためである。

4.2 ページの分岐

セッションIDのほかにもWWWブラウザに情報を預ける必要が生じる場合がある。

それはたとえば、オンライン・ショッピングの注文一覧のページに表示されているリストの中から1つをユーザが選んだときのような場面である（図12）。リストの1件分の明細を次のページに表示する場合にその選択肢の情報をブラウザに預けることになるからである（図13）。

図12 ページの分岐

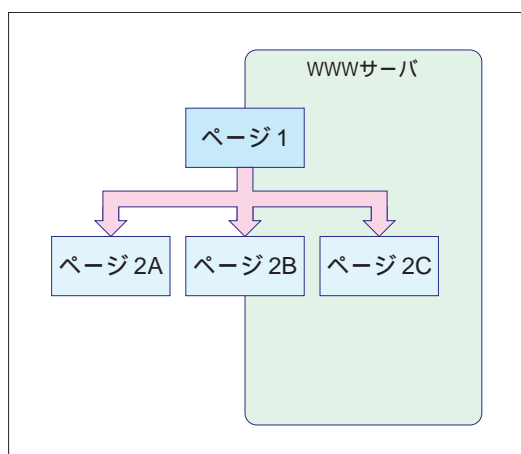
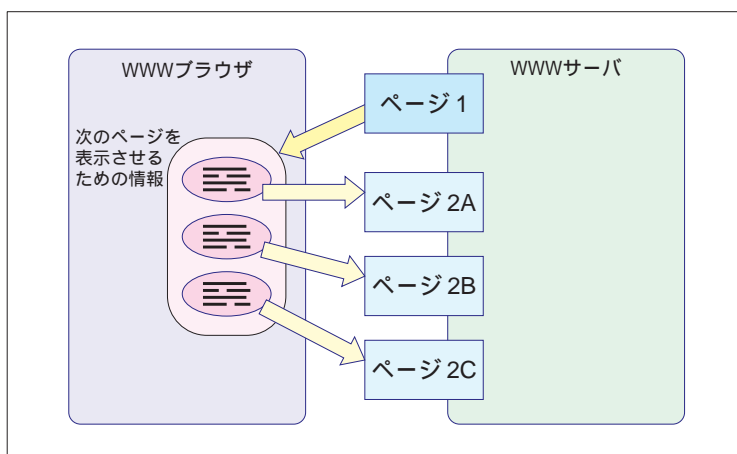


図13 預ける情報が増える



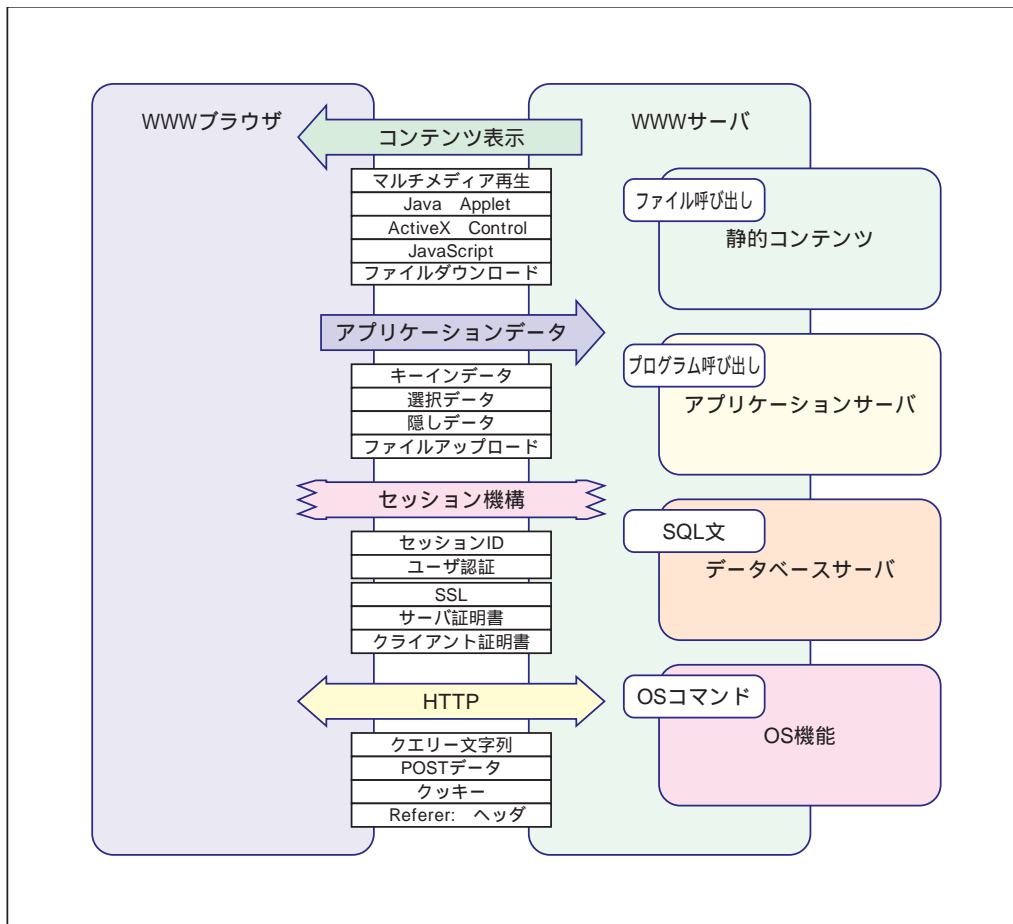
一覧のページに配置される各項目は、対応する明細ページの呼び出しの形で準備される。多くの場合、パラメタ付きのプログラム呼び出しのURLの形をとる。

この明細内容の呼び出しの際のパラメタに実際のデータのキーの値を直接記述するようになっていると、ユーザは明細ページの呼び出しの仕組みを推定することができ、勝手にその呼び出しを行って他人の注文内容明細を呼び出してしまう場合がある。

4.3 さまざまな種類の技術

Webアプリケーションは、その内部でさまざまな種類の技術をつなぎ合わせて使っている。それらには、HTML、HTTP、SSL、アプリケーションサーバ、DBサーバ、ネットワーク機能、OSが提供する各種機能がある（図14）。

図14 考慮すべき技術要素が多い



これらを安全に使いこなすにはそれぞれ「こつ」があるのだが、その1つを見落としていたりするとアプリケーションに脆弱性が生まれてしまう。

4.4 開発面 技術的なばらつきが多い

インターネットには多くのWebアプリケーションが存在し、その開発や運用に携わっている人々の技量はさまざまである。

品質が十分考慮されセキュリティ対策の十分なWebサイトがある一方で、ソフトウェアにあまり詳しくない人が簡単なCGIスクリプトの書き方を覚えて掲示板サイトを構築しているといった場合もある。

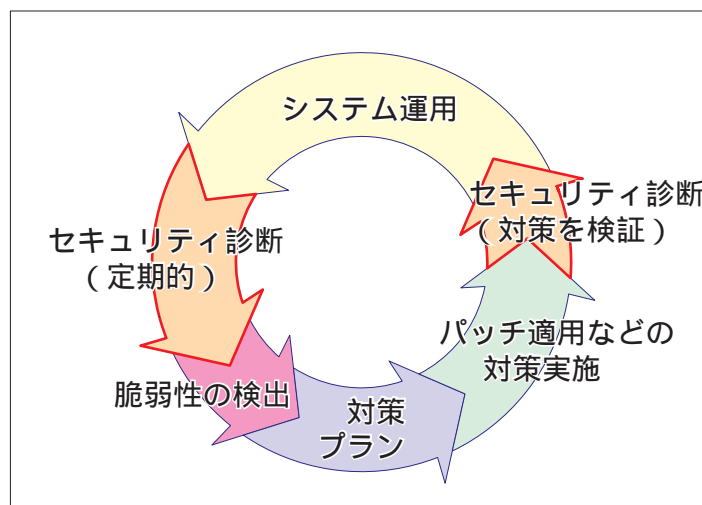
見た目が似ている2つのWebサイトであっても、その背後で動作しているプログラムの品質に大きな差がある場合がある。

4.5 運用面 セキュリティ・サイクルが回っていない

自分が運営しているWebアプリケーションに脆弱性が潜んでいることを心配している運用者は多いが、必ずしも的確な手だてが講じられていないのが現実である。

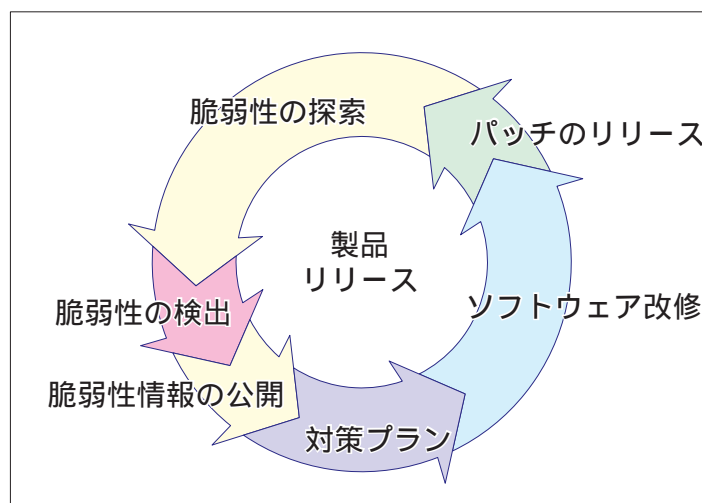
パッケージとして売られているソフトウェアの場合は、セキュリティレベルを維持するための「セキュリティ・サイクル」が回転している。セキュリティ対策に敏感なユーザは自己の運営する各種ソフトウェアを定期的に診断し、セキュリティ脆弱性があるべくふさがれるようにしているのである（図15）。

図15 ユーザのセキュリティサイクル



そして、このユーザのセキュリティサイクルは、脆弱性を見つけて告知しパッチをリリースするベンダのセキュリティサイクルによって支えられている（図16）。

図16 ベンダのセキュリティサイクル



4.6 Web アプリケーションは誰が守る？

しかしながら，Web アプリケーションではこうした「セキュリティ・サイクル」は回転していないことが多い。

そうすると，潜んでいる脆弱性を見つけるための定期的な作業も，脆弱性が見つかったときのソフトウェア改修の体制も整っていないことになる。

5. 診断サービスを利用したセキュリティ対策

5.1 脆弱性検出の努力の必要性

ソフトウェアのセキュリティ脆弱性は、そのソフトウェアの運用が始まってから、次第に見つかるものである。どんなに品質が高い開発チームでもバグをゼロにすることが困難なのと同様、ソフトウェアのセキュリティ脆弱性をあらかじめ皆無にしておくことは難しい。

Webアプリケーションの場合は、非常に多くの人々がアクセスすることができ、運用者が気付かぬうちに第三者によってさまざまな脆弱性が見つけたされてしまうことがよくある。中には善意で脆弱性について通知してきてくれる人もいるので、そうした情報提供はたとえ的はずれであったとしても尊重すべきである。

ただし、こうした善意の第三者にのみ依存するだけでは、次々と脆弱性が明るみにすることによるセキュリティレベルの低下は防げない。

Webサイトの運用者は、自分のWebアプリケーションを自ら定期的に診断する必要がある。

その際、各種の診断手法やツールを自分で用いるやり方もあるが、専門家が提供している診断サービスを利用する方法もある。

第三者の目で診断を行うことには利点がある。Webサイトの運用者やWebアプリケーションの開発者が見落としている点があると、自分たちで行う診断ではその点をふたたび見落としてしまうおそれがあるからである。

5.2 Webアプリケーションの診断サービス

筆者が属すセントラル・コンピュータ・サービス株式会社(CCS)では「TRUSNET^(R)セキュリティ・サービス」を提供している。この「TRUSNET^(R)」のサービスの中には、Webアプリケーションの診断を行うメニューも含まれている。

これらのメニューには自動診断ツールを活用して手軽に行うことのできる「スキャンング診断」と、Webアプリケーションの内部をじっくり調べることのできる「ソースコード診断」といったものが用意されている。

6. まとめ

Webアプリケーションは、多くの人々の目にさらされ、狙われやすい。しかも、WWWブラウザに対して開かれた構造を持ちさまざまな技術要素が組み合わされているため、セキュリティ確保のためには各種の注意と工夫が欠かせない。

そうした努力をしても思わぬところに脆弱性が生まるおそれがある。

こうした問題に対しては、開発者・運用者とは異なる立場の第三者の目による「診断」が有効である。

参考文献

IPA ISEC「セキュア・プログラミング講座」

<http://www.ipa.go.jp/security/awareness/vendor/programming/index.html>

「セキュアWebプログラミング」

<http://www.trusnet.com/secinfo/docs/webprog1/index.html>

「Webアプリケーションのセキュリティを守る TRUSNET^(R)セキュリティ・サービス」(PowerPointファイル)

http://www.trusnet.com/secinfo/docs/0210/trusnet_web.lzh

「TRUSNET^(R)セキュリティ・サービス」

<http://www.trusnet.com/>

TRUSNET はセントラル・コンピュータ・サービス株式会社の登録商標です。

以上

TRUSNET®