セキュリティに配慮したインターネット応用システムの構築

松本哲始*　　岡田　正**

Construction of the Internet Application Systems Designed in Consideration of Security

Satoshi MATSUMOTO and Tadashi OKADA

When designing the Internet application systems, it is important to consider in security. This paper deals with causes and measures where security holes happen. On the basis of our experience of constructing the systems, we report our practices such as a note in regard to Web-based programming, a code review for programs developed, an adoption of safe mechanisms.

Keywords. Internet application system, Security, Web-based system

１．はじめに

我々の研究室では，以前から地域企業と共同でインターネットを利用した種々のシステム開発を行っている [1-4]．最近開発したシステムはインターネットに常時接続で公開運用されており，セキュリティ面の脅威に常にさらされている．さらに，設置先の企業に情報通信に関する高度な知識を持った技術者がいる場合は少なく，地域と連携したシステム開発において，セキュリティの確保と安全な保守への配慮は重要な課題である．

本論文では，地域企業と共同で安全なシステムを安価に開発するための手法と実践例を取り上げている．インターネットを利用したシステムは，中核となるネットワークサーバとその上で動作するプログラムが重要なので，これらの安全性を高める方法を検討する．さらにこれらの検討結果が，実際のシステムにおいてどのように活かされているかを具体的に報告したい．

２．では安全性を保つ上で重要なセキュリティホールの発生要因を整理し，３．においてセキュリティに配慮したシステム開発の要点を述べる．４．で地域企業と共同で開発したシステムにおいて，セキュリティ対策がどのように適用されているか具体的に述べる．

原稿受付　平成１９年８月３１日

*専攻科電子・情報システム工学専攻科修了生（平成１８年度）
**情報工学科

２．セキュリティホールのできる要因

セキュリティホールとは，何らかの要因により生じるシステムのセキュリティ上の弱点のことをいう [5]．セキュリティホールは，不正行為の糸口として使われることが多く，特にソフトウェアの欠陥によるセキュリティホールの場合，修正に時間がかかり，その間無防備となってしまうことが多いため問題である．ここでは，セキュリティホールのできる一般的要因と，我々が構築するシステムで重要な Web アプリケーションに対する問題を検討する．

２．１　一般的なセキュリティホール

セキュリティホールのできる一般的要因として，次の３つが重要である．

(1)設定のミス
(2)セキュリティ意識の不足
(3)システムの欠陥やメーカの対応の不備

セキュリティ侵害の主な原因が(1)の設定ミスによるもので，約半数は設定ミスに起因すると言われている．設定ミスをなくすために，セキュリティ機能をすべて使えば安全なコンピュータとなる．しかし，使い勝手が悪くなることが多いので，広く使われているパソコンではデフォルトで制限をかけていない．そのため，無意識のうちに脅威にさらされたり，制限をかけるときに設定ミスを引き起こしやすい．いずれにしても，セキュリティ対策と使い勝手

のバランスをとることが大切となる．

次に，(2)のセキュリティ意識の不足がある．これはソーシャルエンジニアリングといわれるもので，電話で管理者になりすましてパスワードを聞き出すといった手口がある．電話以外にも，ゴミ置き場に積んであるゴミの中やサーバルームに忍び込んでパスワードのメモを探したり，パスワードを入力しているところを後ろからのぞき見たりして入手する．管理用パスワードが重要であるのはもちろんのこと，一般ユーザのパスワードであっても入手できればLANやサーバへのアクセスが容易になるため，どのようなパスワードであっても漏れることがあってはならない．しかし，パスワードの管理はユーザ個人に委ねられるため，各個人のセキュリティ意識の問題となり対策が難しい．

最後に，(3)のシステムの欠陥やメーカの対応の不備がある．これはセキュリティ問題の中で設定ミスに次いで多い問題で，しかもエンドユーザでは対処できないこともあり困った問題である．プログラムにセキュリティ問題が発覚すると，メーカはパッチを配布して問題を解決している．しかし，問題が発見されてからパッチがリリースされるまでの間は無防備な状態にさらされる．さらに，パッチを入手し，適用する作業はユーザが行わなければならない．パッチを当てることでほかの不具合が起こることもあるし，パッチを入手して当てるには手間がかかるため，面倒くさがりのユーザならたいした問題ではないとパッチを当てず，セキュリティホールが放置される可能性がある．

2．2　Web アプリケーション開発に対する注意

2004年の調査では，Webアプリケーションの約6割に致命的な欠陥があり，それ以外にも何らかのバグがあり，安全な Web アプリケーションは数%しかなかったという報告がある[6]．それだけでなく，脆弱な Web アプリケーションに対して修正を行ったとしても，98 %は脆弱性が残ってしまうというショッキングな調査結果もある[6]．Web アプリケーションの開発では，正しい知識と手法を理解しておかなければならない．

Web アプリケーションに対する代表的な攻撃手法に，次のようなものがある[7]．

- Path Traversal
- SQL Injection
- OS Command Injection
- Session Hijacking/ Replay
- Buffer Overflow
- Cross Site Scripting
- Parameter Manipulation
- Backdoor & Debug Options
- Forceful Browsing
- Client Side Comment
- Error Codes

これらの攻撃は従来の OS や HTTP デーモンへの攻撃と異なり

- 必ずしもサーバの管理者権限を奪うことが目的ではない
- Web アプリケーションごとに攻撃のパターンが違う
- 必ずしもファイルを改ざんするわけではない
- ウイルスやワームを送りつけているわけではない
- ほとんどの攻撃がログに残らない

などの特徴があり，基本的な対策に加えて，利用段階だけでなく開発段階でも，攻撃意図に応じた対策をとる必要がある．

例えば，Cross Site Scripting（XSS）は，なりすましなどの問題を引き起こす脆弱性の一つである．掲示板やフォームの入力確認画面で，ユーザが入力した値をそのまま表示しているアプリケーションで起こりうる．HTML タグを入力した際にタグが有効な表示をする作りになっているプログラムは，非常に危険であると認識すべきである．XSS の脆弱性をついたなりすまし攻撃が問題となるのは，ユーザは特に被害に遭うようなことを行っていないのに被害に遭ってしまうことがあるためである．XSS 対策として，入力値チェックをシステムレベルで行わなければならない．

安全なプログラムを作成するためには，設計段階からセキュリティについて確認を行い，すでに完成された安全性の高いモジュールがある場合，それを流用する．また，再利用できるような形でサブルーチンを設計し，セキュリティに重点を置いて開発することで，後の開発に役立てることができる．一方，システムを一から開発する場合は，安全性を考慮したプログラミング言語を利用することで，比較的簡単にセキュリティ性能を上げることができる．

プログラムを実際に書く上での最も重要な留意点は，わかりやすいプログラムを書くことである．読みにくいプログラムでは，作っている本人もミスをしやすくなるし，後からコードレビューをしている人も分かりづらく，バグを発見することが困難になる．わかりやすく，バグを作りにくいプログラムを書くためのポイントとして，次のようなものがある．

(1)変数に明示的に初期値を代入する．
(2)生成した変数やオブジェクトは，必要がなくなれば明示的に解放する．
(3)変数を宣言して使用しているか確認する．変数宣言の必要がない Perl であっても，変数名を{ }で囲んで明確にする．

(4)必要のないグローバル変数を使っていないか確認し,できるだけローカル変数を使うようにする.
(5)コメントが書かれているか確認し,他人が読むことを想定してコメントをつけるようにする.
(6)プログラムのあちこちで定数を定義していないか確認し,定数はプログラムの初期化部分に変数に入れて利用するようにする.
(7)インデントの方法や深さなど,書式はそろっているか確認する.

3．セキュリティに配慮したシステム開発

安全なシステムを実現するためには,基盤となるプログラムにセキュリティホールがあってはならない.この問題には我々は,オープンソースソフトウェアを活用することで対応している（詳細については文献4)に譲る).

次に,システムを実現するのに必要で適切な技術を採用し,プログラム作成での注意を守りながら開発する.さらに,開発したプログラムは,動作確認だけでなく,セキュリティ配慮を含めたコードレビューを行うべきである.最後に,実運用に入るときは正しい設定を行って安全な運用を始めるとともに,その後に発見されたセキュリティホールを速やかにふさぐ体制を準備しなければならない.

3．では我々が採用した技術の概要とコードレビューの方法について述べる.

3.1　セキュリティに関連した利用技術

先にも述べたように,システムを安全に保つためには,適切な技術を選択し正しく使わなければならない.ここでは,インターネット応用システムの構築で採用した技術の概要を取り上げる.

まず,サーバマシンの基盤となるOSに関連したことを述べる.我々は従来からOSとしてFreeBSD[8)]を採用している.FreeBSDは,ネットワークやセキュリティなどに関して最新機能を実装し,かつ高負荷時にも安定で強力なUnix系OSである.オープンソースソフトウェアとしてセキュリティパッチも迅速に提供されるため,安全で安定なサーバを構築するための基盤になっている.

さらに,FreeBSDには安全を高める技術が実装されており,その一つにchrootがある.これはUnixシステムコールの一つで,ファイルシステムのルートディレクトリの位置を変更するというものである.この機能を活用することで,ユーザがシステムファイルや他のユーザのディレクトリへアクセスすることを完全に防止できる.さらにchrootの概念を拡張して,FreeBSDにはJailが実装されている[9)].Jailとは,仮想FreeBSDマシンを実現する機能で,FreeBSDマシン内に,もう一つのFreeBSD環境を作ることができる.Jailはchrootを拡張したもので,ネットワークプロセスを含めて,全プロセスを完全に隔離できるようにしたものである.Jailの仮想マシンを利用することで,攻撃やシステムダウンがあった場合でも被害を局所空間に閉じこめることが可能となる.

今回構築したシステムは,WWWを応用したものである.この中心技術は実績を積み,安全な運用に問題は出ない.しかし,データベースとの連携やマルチメディア処理の付加機能など,拡張が容易なことが問題を引き起こすことがある.例えば,CGI（Common Gateway Interface）はWebサーバの機能を補助する仕組みで[10)],WWWクライアントからWWWサーバ上のプログラムを起動して処理を行うことにより,動的HTMLコンテンツを生成できる.CGIは標準出力の使える言語であれば,どのような言語を使っても実現可能である.一般的にLarry Wall氏が開発したPerl[11)]と呼ばれる言語で使われており,我々もこれを使っている.

WWWサーバは,オープンソースソフトウェアであるApache[12)]を使って構築する.Apacheは,バーチャルドメインという機能をもっている.この機能を使うと,１台のサーバマシンに最低一つのIPアドレスを割り振るだけで,複数台のWWWサーバと同じ役割を果たすことができる.Apaceのバーチャルドメインには２種類の方式がある.一つはIPアドレスでホストを区別する方式であり,もう一つはWebブラウザがサーバに送信するホスト名を元にして応答するホストを決定するネームベース方式である.

最後に,暗号化技術を取り上げる.ネットワークを介して安全なデータ交換を行うためには,暗号化して通信を行うことが欠かせない.このために一般的に使われいるのは,SSH（Secure SHell）である[13)].SSHを使えば,暗号化された安全な通信が行えるほか,ポートフォワーディングによる接続の拡張ができ,多様で安全な利用方法を容易に実現できるものとなっている.さらに,ファイル送受信に一般的に利用されているFTPは,暗号化が行われておらず,データが盗聴されてしまう可能性がある.それを改善するために開発されたのがSFTP（SSH File Transfer Protocol）で,パスワードやデータをSSHで暗号化して送受信するため,安全なファイルの送受信が可能となる.

3.2　コードレビュー

セキュリティホールやバグを発見するための手段として,コードレビューがある.コードレビューと

は，ソースコードを読み，問題点がないか検証することである[14]．コードレビューには，ツールを使って行う自動検査と，人がソースコードを読んでチェックを行う手動検査がある．自動化ツールについて，今回はRats[15]を使って調査を行った．Ratsは，“C，C++，PHP，Perl，Python”コードの調査を行うことができるツールである．一方，手動検査は，人が仕様書やソースコードを読み，データの扱いをチェックするほか，あえて不正な入力を行い，動作をチェックすることも行われる．

自動検査の特徴として，チェックする人の技量に依存しないというメリットがある一方で，検査漏れや検査不能な場合がある．それに対して手動調査は，ソースコードだけではなく，仕様書などに基づく詳細なチェックが可能になる上，実際に動作させてチェックすることが可能となる．しかし，時間がかかる上，チェックする人の技量に大きく左右されるという問題がある．

4．セキュリティ対策の適用事例

4．1　コードレビューの結果

これまでの調査を元に，学内掲示板システム[3]のコードレビューを行った．学内掲示板システムは，学生への連絡事項を従来の張り紙による告知から，液晶画面を利用した電子的な表示に切り替えることで，掲示板管理の効率化を図るシステムである．なお，このプログラムはPerlで書かれており，掲示板管理プログラムと新規投稿プログラムの２つのCGIからなる合計500行程度のプログラムである．

手動でのコードレビューを行った結果，大きなセキュリティ上の問題点はなかったものの，下記のような小さな問題点が見つかったので，ただちに修正した．

(1) チェックが不完全なためpngファイル以外がアップロード可能
(2) エラーになった場合でもエラーメッセージの表示なし
(3) 排他処理が不完全なためファイルが上書きされてしまう可能性あり

次に，ネットプリントシステム[3]のCGIについてもコードレビューを行った．ネットプリントシステムは，インターネットを利用したデジカメ画像プリント注文システムで，４つのCGI（注文プログラム，ユーザ登録プログラム，サムネイル表示プログラム，履歴表示プログラム）からなる合計2000行程度のプログラムである．

まず，Ratsによる自動検査を行ったところ，Open関数の警告が15カ所，Mkdir関数の警告が11カ所，Rand関数の警告が2カ所表示された．Open関数とMkdir関数については，引数にユーザ入力値を利用している場合，パスの乗り越えなどの問題を引き起こす可能性があることを示している．一方，Rand関数については，標準の乱数ジェネレータは性能が悪く，結果が偏りがちになるため，セッションIDを推測しやすくなってしまうことが指摘された．

指摘された部分をチェックしてみたところ，Open関数とRand関数とについて，ユーザ入力を引数としているものはなかった．一方，Rand関数の問題は，FreeBSDのPortsから別の乱数ジェネレータをインストールすることで改善した．

続いてコードレビューを手動で行ったところ，Ratsで検出されていないOpen関数やMkdir関数がいくつもあった．調べてみると，

```
open（FH,filename）
```

のように括弧でくくって書いた場合はチェックの対象となり，

```
open FH,filename
```

のように括弧でくくらずに書いた場合はチェックの対象となっていなかった．表現の自由度の高いPerlの検査には，あまり向かないツールといえるかもしれない．

その他の部分についてもチェックを行い，不審な部分には実際に不正な入力を行ってみたところ，いくつかの問題点が見つかったため，改善できる部分は改良を行った．見つかった問題点を以下に示す．

(1) ユーザ登録のメールアドレス欄に改行コードを挿入すると，任意のタイトル・本文を挿入することが可能
(2) サニタイジング文字の不足
(3) XSS対策が無意味
(4) JPEGファイル以外のものがアップロード可能
(5) ログイン失敗時のエラーメッセージが不適切
(6) 操作画面やHTMLのHiddenフィールドにメールアドレスを表示
(7) 操作画面のアドレスバーやステータスバーの表示なし
(8) ユーザ登録時・ログオン時のデータを平文で送信
(9) JavaScriptを有効にする必要があることが未表示

4．2　インターネット応用システムと採用技術

我々は，デジタルカメラで撮影した画像と枚数・サイズの情報をインターネットで送信し注文を受け付け，注文30分後に写真を受け取ることができるネットプリントシステムを開発した（Fig.1）[3]．このシステムは，画像登録から注文までの一通りの機能を完成させ，写真店へ光ファイバによるインターネット接続環境を準備し，店舗に設置したサーバで

運用してきた．その後，１台のサーバで複数店舗へ対応することや，運用で生じた問題を解消する機能拡張の要望があり，種々の対応を行っている．この過程で採用した技術を，主にセキュリティ確保の観点から述べる．

Fig.1　Order picture of the NetPrint System

現在のシステムは１社のみが運営するサービスである．これを，他のカメラ店からも利用したいという要望があった．このとき，コストを削減する必要があるため，１台のサーバマシン上で複数の店舗を稼働させることとなった．１台のマシンで複数のシステムを動作させる方法として，Jail とバーチャルドメインが考えられる．

開発当初 Jail を用いてシステムを構築した．Jail を使うと店舗情報を完全に分離できるものの，ネットワークが独立しているため，各店舗ごとに IP アドレスが必要となる．現状では固定 IP アドレス取得費用がかなり負担となっているため，一つの IP アドレスで済むバーチャルドメインを利用することとなった．バーチャルドメインでは，Web ブラウザがバーチャルドメイン対応である必要がある．この点は，当システムの推奨ブラウザがバーチャルドメインに対応した InternetExplorer6 以上，Opera Ver6 以上となっているため，バーチャルドメインを用いてシステムの構築を行った．

バーチャルドメインを利用すると，同じサーバマシンに複数店舗からのアクセスがある．標準状態では他店舗の情報へもアクセスすることが可能となるため，対策を施す必要がある．また，画像のダウンロードを行う人がコンピュータに詳しいとは限らず，システムファイルを書き換えたり，削除したりしてしまうとシステムトラブルの原因となる．

この問題を避けるため，chroot を用いてアクセス制限を行った．まず，各店舗ごとに chroot を設定し，他店舗へのアクセスを禁止した．さらに，各店舗領域の中の，ホームページ・価格データ・注文画像の各ディレクトリに chroot を設定し，用途ごとのユーザを作成することで操作ミスによるシステムトラブルを防いでいる．また，データ転送はすべて SFTP を使い，端末パソコンに必要なプログラムや鍵を自動で導入できるよう，インストール CD も作成した．

現在バーチャルドメインを用いたシステムを構築し，インターネット上で運用を開始した．この検討と並行してコードレビューによりプログラムを見直し，多店舗対応が容易に行えるよう，店舗固有情報の分離統合を行った．この他に，パスワードの変更機能や古い画像の自動削除機能の実装など，より使いやすくする機能の追加も行っている．

今後は，店舗を追加して運用したとき，複数店舗のアクセスが１台のマシンに集中するため，どの程度まで１台のマシンで対応可能なのかについて検証しなければならない．

５．あとがき

本論文では，インターネット応用システムを構築するとき，セキュリティ面で何に注意しどう検証すれば良いかを，具体的なシステム構築や改良の経験をもとに述べた．現在，複数のサーバが実際に安全に稼働しており，我々の開発が適切に行われていると考えている．

開発したシステムは，システム本体と管理を外部に移転しているものも多い [16]．このため，これからは運用管理の面から連携先企業と共同して，安全で安定なシステム運用に努力しなければならないと考えている．

謝　辞

ここで取り上げたシステムの開発には，地元企業関係者の多大なご協力をいただきました．この場を借りて深く御礼申し上げます．

参　考　文　献

1）岡田・近藤: 津山工業高等専門学校紀要 **No.41**（1999）5-11.

2）岡田・野村: 津山工業高等専門学校紀要 **No.44**（2002）21-24.

3）宮岡・衣笠・岡田: 津山工業高等専門学校紀要 **No.46**（2004）59-66.

4）田渕・佐野・岡田: 津山工業高等専門学校紀要 **No.48**（2006）55-60.

5）D. Russell and G.T.Gangemi Sr.（山口監訳）:コンピュータセ

キュリティの基礎，アスキー出版局（1994）.

6）Web アプリケーションの脆弱性は修復してもなお 93 %に脆弱性が残る ,http://internet.watch.impress.co.jp/cda/news/2004/06/30/3698.html.

7）Web アプリケーションに潜むセキュリティホール 第１回～第１２回,http://www.atmarkit.co.jp/fsecurity/rensai/Webhole01/Webhole01.html .

8）The FreeBSD Project: http://www.freebsd.org/

9）Jails: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/jails.html.

10）CGI とは何か：http://mtlab.ecn.fpu.ac.jp/scripting/about_cgi.html

11）The Source for Perl: http://www.perl.com/.

12）The Apache HTTP Server Project: http://httpd.apache.org/.

13）D.J. Barrett, R.E. Silverman, R.G. Byrnes（小島監訳）：実用 SSH 第 2 版，オライリー・ジャパン(2006).

14）M.G. Graff, K.R. van Wyk（新井，一瀬訳）：セキュアプログラミング－失敗から学ぶ設計・実装・運用・管理,オライリージャパン（2004）120-137.

15）Secure Software: http://www.securesoftware.com/.

16）岡田：第 5 回全国高専テクノフォーラム予稿集 (2007-8) 62-63.