TechStyle
Quality for All Professionals

Internet Week 2002
tutorial - T6

WEBセキュアプログラミング

～脆弱性を作らないWEBアプリ開発～

岡田 良太郎

株式会社テックスタイル
http://techstyle.jp/
riotaro@techstyle.jp

December 17, 2002

TechStyle
Quality for All Professionals

about Speaker

- 岡田 良太郎
  - 1989年 神戸市立神戸工業高等専門学校電気工学科卒業
  - 1999年 日本Linux協会運営委員
  - 2001年 有限会社テューンビズ代表取締役就任
    Allabout Linuxガイド担当
    http://allabout.co.jp/computer/linux/
  - 2002年 株式会社テックスタイル代表取締役就任
    PHPカンファレンス2002プログラム委員長
    CISA
    http://okdt.org/

- 本資料・講演に関するお問い合わせは岡田（ryotaro@techstyle.jp）まで。

TechStyle
Quality for All Professionals

Webアプリのセキュリティ要件

- 耐性の高いシステムであること
  - 攻撃に耐えられるように意図して、保護の方策を施したアプリケーションを設計すること

  → トラブル対応の軽減・市場評価の維持、向上

- ユーザにとって心配のないシステムであること
  - プライバシーの問題
  - システム停止・データ損失の問題

- 開発会社にとって「コスト」を甘受できるかどうかの問題…

TechStyle
Quality for All Professionals

セキュリティ3要素と脅威

- 機密性 :対象 (人)、内容 (データ)の関係を保証
  - ユーザID、管理権限、パーソナリゼーション
    - 脅威 :顧客データ漏洩, ID不正利用
- 完全性 :正確な伝達
  - メール、メッセージ
    - WEBサイト改ざん, ウイルス感染, 盗聴, なりすまし
- 可用性 :期待されるときはいつでも応じられる
  - パフォーマンス、アベイラビリティ
    - DoS対策,サーバダウン, タイムアウト

cf. What types of attacks exist?
There are three main types of attacks (Saltzer 1975):

1. Unauthorized release of privileged information.
2. Unauthorized modification of privileged information.
3. Denial of service.

TechStyle
Quality for All Professionals

セキュリティ設計

- いつ行うか
  - 後からセキュリティ機能を追加する方法は機能に影響
  - 影響範囲
    - 機能
    - ユーザインターフェース
    - エラー処理
- 「コスト」

TechStyle
Quality for All Professionals

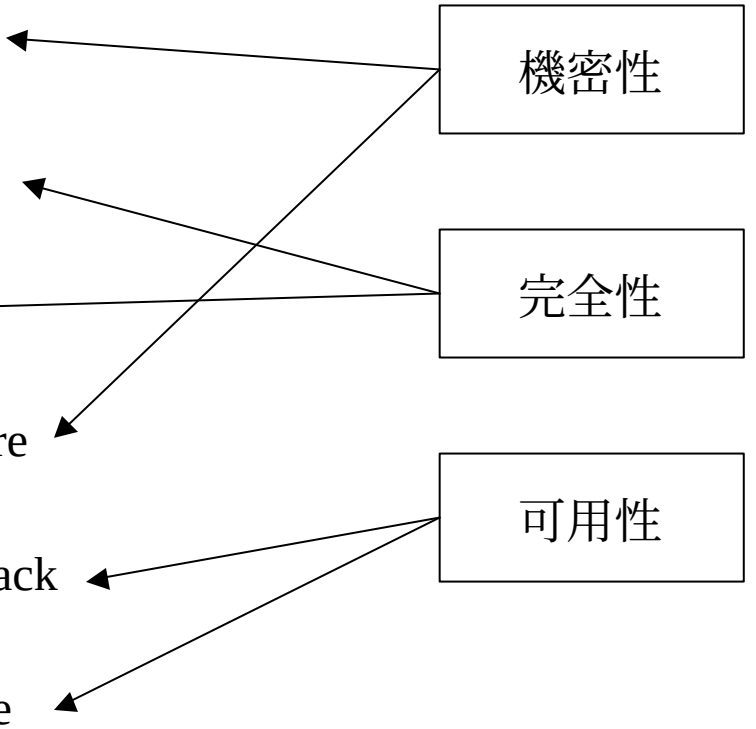
基本的な原則

- セキュリティ要件決定
  - 脅威モデルによる脅威の類推
  - 外部の危険を認識
  - 事実に基づかない楽観論を排除する
- 開発プロセスの確立
  - 開発方式のガイドライン化と監査
  - 「デフォルト」より緩めない
  - 普遍のセキュリティ方策はない
- 重層的な保護策
  - 最小限の権限の使用
  - 動作と被害の記録
  - 障害時の対策の準備

TechStyle
Quality for All Professionals

脅威例 :脅威モデルの分析

- 誰から何を保護するのかをモデル化して整理
- 脅威モデル（例:STRIDE）
  - なりすまし
    - Spoofing identity
  - データ改ざん
    - Tampering with data
  - 否認（とぼけ）
    - Repudiation
  - 情報漏えい
    - Information disclosure
  - サービス停止
    - Denial of Service attack
  - アクセス権の昇格
    - Elevation of Privilege



TechStyle
Quality for All Professionals

対象例 :動作前提となる依存関係

- 物理的な依存
  - IDC:電源、空調、ネットワーク
  - HW: ボード、CPU、メモリ、ディスク、ケーブル
- 外部ネットワーク的な依存
  - 上位ISP、ルーティング、DNS、ipフィルタ
  - 管理ソフト、監視ソフト、FTPサイト、バックアップ、バックドア
- 内部ネットワーク的な依存
  - www、smtp、configuration、logrotate、
  - ssh、サーバ内管理ツール、管理チーム（人）、

TechStyle
Quality for All Professionals

脅威と対象からアタックをイメージ

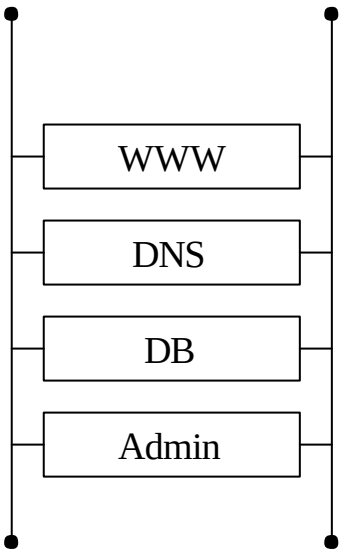
- ブラウザからのアタック（例）

1. Port Scanして開きポートをチェック
   - 20,22,80,8080,…
2. 開きポートごとにExploitアタックを発射
   - Linux(ICMP), telnet, OpenSSH, Apache, …

   → サーバダウン（DoS/DDoS）

   → バッファオーバフロー,侵入成功
   （ユーザ情報/システムファイルの取得）

   → HEADコマンドでサーバ情報をGET
3. HTTP/HTTPSをブラウザで
   - FORMにタグを埋め込んでみる
     - クロスサイトスクリプティング
   - FORMにシェルエスケープ文字を入れてみる
   - URLなどを渡しているようなら、別のドメインを入れてみる
   - 不正に(？)ファイルを取得

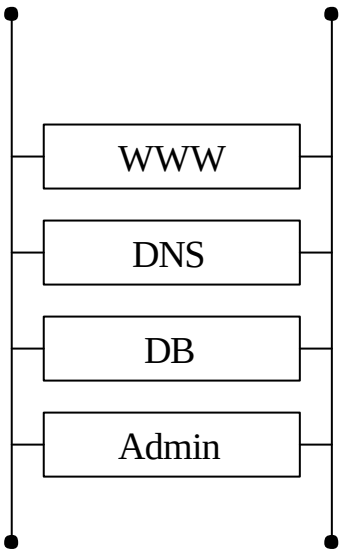


TechStyle
Quality for All Professionals

脅威と対象からアタックをイメージ

- 偽サーバ立ち上げによる権限取得（例）

1. ネットワーク情報からDNSサーバを改ざん
   - JPNICにDNS情報の書き換え依頼（不正）
   - 既存DNSをクラックしてAレコードを追加（ラウンドロビン）
   - 同一ドメインで偽サーバ立ち上げ
2. アプリケーションハイジャック
   - Cookieの分析
   - 偽サーバで偽Cookieを発行(/読み取り)
   - あるいはXSSでCookie情報から不正取得



TechStyle
Quality for All Professionals

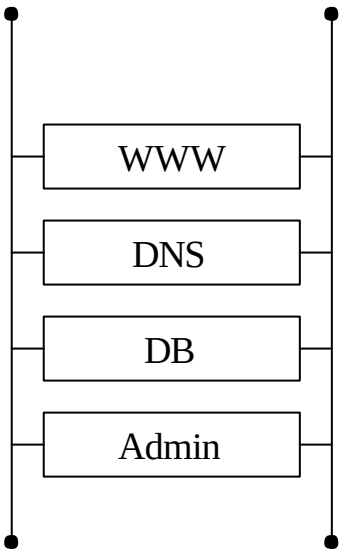
脅威と対象からアタックをイメージ

その他・・・

- 管理用バックドア
- FORMの不正入力
- URLに不正パラメータを書いたリンクを掲載（掲示板など）
  - ex. http://www.yahooo.co.jp?cmd=jump&url=http://foo.com&...
- (D)DoS：集中アクセス、集中書き込みによるサービス停止
- 依存関係のある別サーバへのアタック（DBサーバなど）

→ 動作前提となる対象と想定される脅威の2次元やアプリケーションの動作基盤の構成図を活用して、対策を策定していく必要がある。

→ アプリケーションプログラミングだけでは防衛は完結しない。構成、体制などの種々の土台が関係する。



TechStyle
Quality for All Professionals

本日のお題 :WEBプログラムにおける 脆弱性」を考える

TechStyle
Quality for All Professionals

問題：バッファオーバーフロー

- 基本原理
  - スタック上のバッファサイズより大きなデータのコピーによって発生
  - 非常に注意すべき関数strcpyなど 後述）
- 被害
  - リターンアドレスの変更が可能

  → プログラムの停止

  → 任意のコードを実行
- ヒープオーバーフロー
  - 動的メモリーアロケーションの場合にもnervous でいられるか

- コピーバッファの長さのチェックを怠らないように
  - 1チャンクの長さもチェックするように

TechStyle
Quality for All Professionals

注意関数：strcpy/strncpy

- strcpy(d,s);
  - 基本的に安全ではない。

```
if(strlen(s)< sizeof(d)){
        strcpy(s, d);
}
```

- strncpy(d,s,n);
  - s の内容を「ちょんぎる」ことの影響を調べておくこと
  - NULLで終了している「はず」の罠を意識

```
d[sizeof(d)-1] =‘¥0’;
strncpy(d, s, sizeof(d));
// s はちょんぎられない
if(d[sizeof(d)-1]!= ‘¥0’){
// s は不正なデータかもしれない
}
```

TechStyle
Quality for All Professionals

注意関数 :gets/fgets

- 標準入力からの読み込み
  - CR/LFまでの読み込み
  - バッファリング予測ができるか？

  → 使うべきではない

TechStyle
Quality for All Professionals

問題 :format string問題

- 書式指定文字列問題
  - 可変個の引数をとる関数で、実際に渡される引数の個数がわからないことに起因

```
printf(inputbuf);
// inputbuf に書式指定文字が含まれていたら？

prinft(%s, inputbuf);
// 安全なコード
```

  - sprintf などは完全に安全な方法はない、、、

TechStyle
Quality for All Professionals

要注意関数

- 文字列のサイズに注目
  - strcpy関連、strcat関連、strncpy関連、strncat関連
- メモリーを扱う関数
  - memcpy関連、sprintf関連、printf関連、strlen関連（限度あり）
- 標準入出力関数
  - gets, scanf,
- パラメータデータに注意
  - include, readfile, fopen, file, link, unlink, symlink, rename, rmdir, chmod, chown, chgrp, exec, system, passthru, popen (など)
- コネクションプーリング（pg関数）
  - すなおにアベイラビリティのためのものだと割り切る

※ TechStyle WEBシステムセキュリティ監査ガイドライン2002 より

TechStyle
Quality for All Professionals

問題 :最大のリスク・ユーザ入力

- 信頼してはいけない（回避してもいけない）
  - FORMデータはWEBプログラムで評価・転用する。表示するだけにとどまることもあろう。
    しかし、それでも問題は起きる。
    - SQLクエリ文字列
    - クロスサイトスクリプティング
    - includeするリモートURL
    - HTMLタグチェック
    - 認証のすり抜け
    - 認証情報の漏洩
    - Referer詐称
    - ファイルアップロード
    - シェルコマンドの実行

TechStyle
Quality for All Professionals

問題：XSS（クロスサイトスクリプティング）

- 被害
  - JavaScriptの実行に至り、Cookieデータ漏洩やサイトの不正表示・プライバシ漏洩が可能
    - Not FoundやSQL Errorなどのエラーメッセージの中に表示させることも
  - アプリ動作とHTMLを知り尽くしたexploit

→ Cookieをdisableするユーザは増えているご時世に。

- 対策
  - Cookieなしで動作するよう極力努力すべき
  - input validation（後述）

TechStyle
Quality for All Professionals

問題：Cookie Spoofing

- Cookieのドメインスコープ
- RFC2965（7.2 Cookie Spoofing）より超訳

1. victim.cracker.eduをアクセスした際に、デフォルトドメインvictim.cracker.eduで、クッキーsession_id=“1234” をセットしたとします。

2. 同じブラウザでspoof.cracker.edu をアクセスしてしまい、session-id=“1111” を、Domain=“.cracker.edu“ と共にセットされたとします。

3. もう一度 victim.cracker.edu にアクセスすると、このプログラムがクッキーを読み出すと、下記のものが取り出せます。

```
Cookie: $Version="1"; session_id="1234",
        $Version="1"; session_id="1111"; $Domain=".cracker.edu"
```

victim.cracker.edu サーバは、二番目のものを自分のサイトのクッキーではないことを判断し、無視できるプログラムであるべきです。

- 対策：Cookieのより厳密なドメインスコープ管理をすべき

TechStyle
Quality for All Professionals

対策 :input validation

- すべての入力をフィルタリング
  - input validation, data cleaning という
  - フォームデータ・URLパラメータ・環境変数を含む
- 入力値チェック・フィルタリング
  - 数字、半角英数、全角
  - 予想外のプログラムエラーを招かないようにする
  - ユーザビリティを不必要に犠牲にしない
- スペシャルキャラクタの深刻な影響
  - メタキャラクタ
  - HTMLタグ記号
  - 全角の “片割れ ”が¥ 記号のようなもの

TechStyle
Quality for All Professionals

対策例 :フィルタすべきキャラクタ

- %(パーセント)
  - GETメソッドによるパラメータでは、URLエンコーディングによりパラメータ文字が %HH(H: 16進数)に変換される。URLの改ざんによりバイナリ化したときに問題を引き起こす文字を混入できる。

    ex.
    %00 (NULL), %0A(改行), %20(スペース), %25(パーセント)

  - URLエンコードされた文字列のvalidationでは、1度のフィルタでクリアしなければならない。

    ex.
    %2500 → %00 → NULL

cf. URLに使用できる文字はRFC2396 (Uniform Resource Identifiers (URI):Generic Syntax)で規定されているので、参考にすること。

TechStyle
Quality for All Professionals

対策例 :フィルタすべきキャラクタ

- ! (bang) ; (semi-colon) : (colon) , (comma) – (minus) ¥ (backslash)
  - exec, system関数、メールエージェント、シェルプログラムなどに渡される文字列にこれらが含まれると、別の任意のプログラムを動作させたり、プログラムに不必要なパラメータを与えてしまう可能性がある。

```
ex.
   !/bin/bash
   -f /etc/passwd
```

- * (asterisk) / (slash) .. (dot/dotdot) ? (question) []{} (brace)
  - ファイル名に利用される可能性のある場合に注意すべき

```
ex.
  ../../../../
  *.png
```

TechStyle
Quality for All Professionals

対策例 :フィルタすべきキャラクタ

- <(lesser than) >(grater than)
  “(double quote) ‘(single quote)
  - タグの初めと終了、オプションアイテムのクローズに用いられるため、タグの埋め込み、クロスサイトスクリプティングexploitのようなスクリプトコードの埋め込みに頻繁に利用 される。他サイトのコンテンツの埋め込み、不正なCookie読み出しに悪用される。
    ＜＞はシェルにおけるリダイレクト記号であるためにコンテンツファイル破壊に悪用される可能性がある。

```
ex.
        ><script>alert(window.location);</script>
        “;alert(document.cookie);
        ‘ onmouseover=‘alert(document.cookie);’ ‘
        “><script> alert(document.cookie);</script>
        “></a> <script> alert(document.cookie);</script>
ex.
        ;cat /etc/passwd >>/home/html/htdocs/index.html
```

TechStyle
Quality for All Professionals

対策例 :フィルタすべきキャラクタ

- HTMLタグ
  - 必要に応じて許可せざるを得ないケース
    - 一般に、<p>,<bold>,<i>,,,<pre>,
は無害とされるが、せめてWell-formedにはしておきたいところ。
    - exploitがないわけではない
      - Javascriptは思わぬところでも動作する。

ex.
…
<img src="javascript:[code]>
<style type="text/javascript">[code]</style>

TechStyle
Quality for All Professionals

対策 :データクリーニング関数例 (PHP)

- ereg_replace(“[^0-9]”,””,$data)
  - 正規表現はクリーニングの基本
- addslashes($data)
  - addcslashes (string str, string charlist)
  - クオート (single,double), バックスラッシュ、NULL文字などをslashing(スラッシュ付加)する
- quotemeta($data)
  - .+?[^](*)$ などをquoteする
- escapeshellcmd($data)
  - メタ文字をescapeする
- nl2br ($str)
  - すべての改行文字の 前に ‘
’ を挿入して返す
- htmlspecialchars ("Test", ENT_QUOTES);
  - 特殊文字をHTMLエンティティに変換する

TechStyle
Quality for All Professionals

対策 :最小限権限の原則

- WEBサーバ
  - apache ユーザ
- DBサーバ
  - mysql / postgresql
  - DBユーザ　削除 ･更新権限）
- ファイルシステム
  - ユーザの読める範囲 ･書き込める範囲 ･書き込む内容
- 管理者権限
  - 運用管理者とアプリケーションへの影響
- メールユーザ
  - ログインアカウントの発行の必要性の検討
- アプリケーションユーザ
  - アプリケーション実装のユーザIDによるACL
- IPアドレス制限

TechStyle
Quality for All Professionals

対策 :ファイル (DB)保存

- 保存」内容により必要性を検討
  - ID / Password
  - 不可逆なhashだと漏れても意味不明
  - 管理担当からでさえ守れる
  - hashデータでその他のプライバシーデータを暗号化
- 保存する場所 •テーブル
  - 一時ファイル
    - 共通 •IDごと
  - 保存ファイル
    - WEBサーバからinvisibleな場所であるべき
    - 保存ファイルの 脅威」を最低限にできるパーミッション
    - バックアップ方策
- 自動保存されるファイルのクリーニング
  - /tmp/
- 削除」
  - 削除」はWEBアプリではpayしない。 無効化」をひたすら用いる。
    - SQLのDELETE文、unlink関数は使用不可とする、など。

TechStyle
Quality for All Professionals

対策 :暗号 ･乱数

- 暗号化すれば安全か
  - XORは見慣れればわかる
  - 暗号関数は自作しないこと
- rand関数出力が類推できる問題
  - 乱数関数出力は「予測」できる！
  - 乱数サーバの利用
  - 乱数関数の作成
  - 重層化
- 不可逆データで保護
  - MD5 / ハッシュ関数

TechStyle
Quality for All Professionals

対策：DoS攻撃と過負荷

- DoS攻撃
  - 各層により対策は異なる

    例：
    - ネットワーク層の場合
    - アプリケーション層の場合

- CPU過負荷
  - WEBアプリケーションの過剰ロード回避
    - 不正データ時にはできるだけ早期に処理を終了させる
    - 入力処理頻度を規定する（例：slashdot/ECサイトのカゴ）
    - 高負荷を与えるクライアントサイトを記録する（NATに注意）

TechStyle
Quality for All Professionals

対策：パフォーマンスとセキュリティ

- 高可用性のために安定動作は重要
  - CGIスクリプトがメインの環境で主なボトルネックはCPU
    - スタティックなHTMLあるいは画像では、メモリーとネットワークがボトルネックになる
  - 「遅いPentium(II) 400Mhzマシンはスタティックな HTMLページで T3回線（45Mbps）を費やし尽くします。」
    （Lim, John, Tuning Apache and PHP for Speed on Unix より）
    - CPUパワーに余力があり、ネットワークの帯域が問題の場合
      - mod_gipや zlib.output_compression を利用

TechStyle
Quality for All Professionals

追加題 :WEB運用の「セキュリティ品質保持」を考える

TechStyle
Quality for All Professionals

セキュリティ対応

- セキュリティ対応のための重要なファクタ
  - 事象・原理に関する知識
  - ソフトウエア入手の確保
  - 実行動作環境
  - ソフトウエア配布の手段
  - 検証環境・検証プロセス
  - 性能要件・機能要件
  - 情報源と問題把握
  - 予防体制・対応体制
  - …

TechStyle
Quality for All Professionals

事象・原理に関する知識

- 横断的な判断の難しさ...
- セキュリティ対応に「唯一無二」の方法などない。
- 一体、なにが起こりえるのか？
  - ネットワーク
    - TCP/IP, ルーティング
  - サーバアーキテクチャ
    - Intel アーキテクチャ
    - クラスタリング
  - カーネル
    - DoS対応、ipchainsによる「境界線」など
  - サーバソフトウエア
    - Apache, SSHなど基盤
  - ミドルウエア
    - 開発プラットフォーム
  - アプリケーション
    - 誰が、どんな機能を作っているのか
  - ユーザイメージ
    - どんなユーザがどんな環境で使うのか

TechStyle
Quality for All Professionals

ソフトウエア入手の確保

- ベンダー依存できるものとできないもの
  - ネットワークからアプリケーションまでの各レイヤーを整理
    - 保守対応でまかなえないことを果たしているか
  - ベンダーOS、UNIX, Linux (Debian,RedHat,Turbo,Miracle)
  - いずれにせよ、確保できているか

TechStyle
Quality for All Professionals

実行動作環境

- 記録！記録！記録！
  - ドキュメントのないシステムは本当に「システム」といえるか
- サーババージョン管理
  - メジャーバージョン
    - 7.2 ? 7.3?
  - パッケージ・ライブラリと配布手段の確保
    - rpm –rebuilddb
    - rpm -qa
- 開発アプリケーション管理
  - CVS, RCSなどの管理

TechStyle
Quality for All Professionals

検証環境 •検証プロセス

- 検証環境は必須
  - VMwareなどの活用

- アプリケーションテスト
  - アップデートテスト用のスクリプトの事前作成
  - 設計段階で利用ライブラリ範囲を作る
  - 検収段階で利用関数の一覧などを作る
  - テストスクリプトをあわせて開発

  - アップデート担当者はテストスクリプトの検証まで行う

TechStyle
Quality for All Professionals

性能要件 •機能要件

- 外部テストの重要性
  - 同一セグメントでは問題が見えないケースが多い
  - abコマンドでできることも多い
  - webalizerで外部からどういうパターンで見られているか調査
- 内部テストの重要性
  - テストスクリプト...

TechStyle
Quality for All Professionals

情報源と問題把握

- 情報源の特性を知っておく
  - ML
  - WEB
  - ディストリビューション情報
  - TechStyle
  - 最重要 :ユーザの権益
- 問題把握
  - その問題は、どうして起きるのか？
  - どこに現れるのか？
  - セキュリティ3要素の何を脅かすのか？ (リスク分析)
  - 記録！記録！記録！

TechStyle
Quality for All Professionals

予防体制 •対応体制

- 人間というリソース
  - 最大のセキュリティホール
  - もっとも冗長性が低い
  - もっとも可用性も低い

→ 瞬間最大風速で最適な運用をしていかなければならない

→ 複雑すぎる手順を、もう少し間便で確実な手順へと改訂

補足：セキュリティ会議は広報や営業などの顧客の顔が見える部署も含めて開催するのが良いと思われる。技術だけのマターではない。

TechStyle
Quality for All Professionals

WEBという「システム」

- 信頼できないユーザの無制限の要求
- ブランド・顧客満足などの重要なマターを課されている

- しっかりと立ち向かっていこう

TechStyle
Quality for All Professionals

参考資料

- Saltzer, J.H., and M.D. Schroeder, "The Protection of Information in Computer Systems," Proc. IEEE, Vol. 63, No. 9, Sept. 1975, pp. 1278-1308.
- Wall, Larry and Schwartz, Randal L. "Programming Perl" : Sebastopol, California : O'Reilly And Associates, 1992.
- Al-Herbish , Thamer, Secure UNIX Programming FAQ,1999
- Wheeler, David A. Secure Programming for Linux and Unix HOWTO, 2002
- Howard, M. and LeBlanc, David, WRITING SECURE CODE, 2002
- RFC 2396, 2965
- Lim, John, Tuning Apache and PHP for Speed on Unix,2001(http://php.weblogs.com/tuning_apache_unix)

TechStyle
Quality for All Professionals

TechStyleニュースレター

TechStyle
Supported by TUNEBiZ

Linux
Apache
PHP
MySQL
PostgreSQL
Ruby
PHP
Perl
JavaScript
HTML
Flash

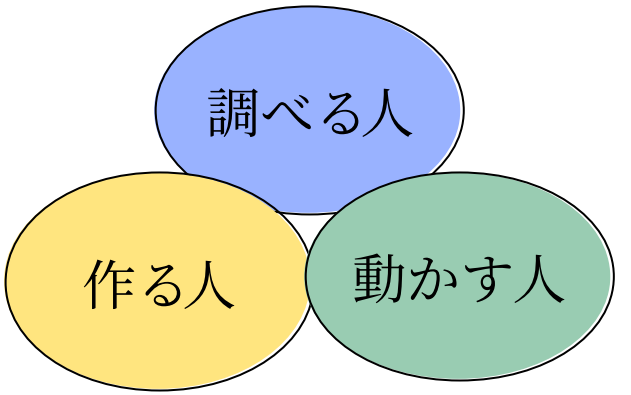
日本Linux協会
日本UNIXユーザ会
日本PHPユーザ会
日本Apacheユーザ会
日本Postgresユーザ会
日本MySQLユーザ会

RFC
Ietf-draft
W3C
ODL文書
BS7799
…

セキュリティ
運用管理
コンサルロジック
ビジネス分析
(T)
スキルアップ
業界再編
株価動向

高信頼の
情報流通

プロフェッショナル



ハードウェアベンダ
ソフトウエアベンダ
ディストリビューション
教育・資格会社
Sier
コンサルタント会社
PR・マーケティング
出版

December 17, 2002

Thank You

Please feel free to mail me

riotaro@TechStyle.jp