情報処理推進機構 ウェブアプリケーション
開発者向けセキュリティ実装講座
2006年2月28日，4月4日 後日配布資料

「CSRF」と「Session Fixation」の諸問題について

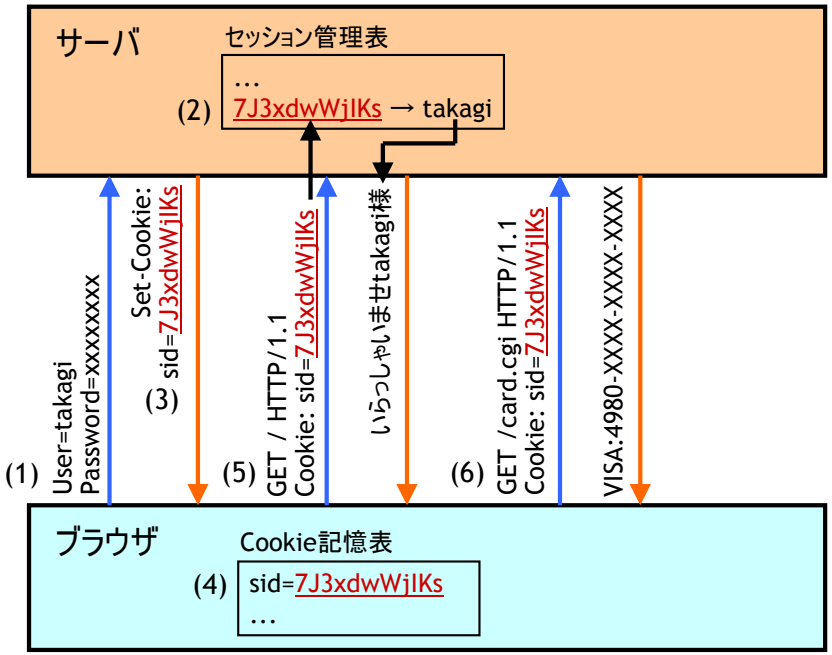
独立行政法人産業技術総合研究所
情報セキュリティ研究センター
高木 浩光
http://staff.aist.go.jp/takagi.hiromitsu/

目次

- 前提知識の確認
  - Webアプリにおけるセッション追跡と認証の実装手段
  - セッションハイジャック攻撃の原理と脅威
- CSRF (Cross-Site Request Forgeries) 別名：Session Riding
  - 歴史的経緯、原因、脅威、技術的対策、適法な存在推定手段
- Session Fixation
  - 歴史的経緯、原因、脅威、技術的対策、適法な存在推定手段

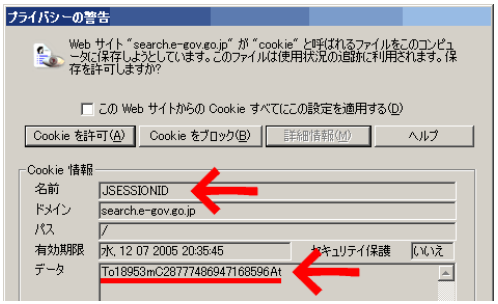
セッション追跡と認証の実装手段

- 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

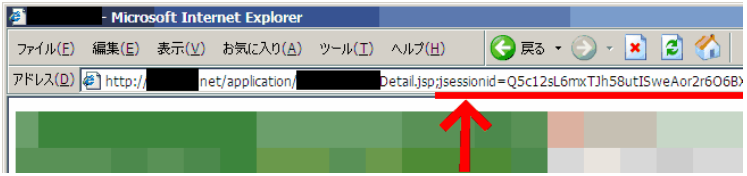


セッションIDの格納場所

- 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

Session Riding

- 別名「CSRF: クロスサイトリクエストフォージェリ」
- 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

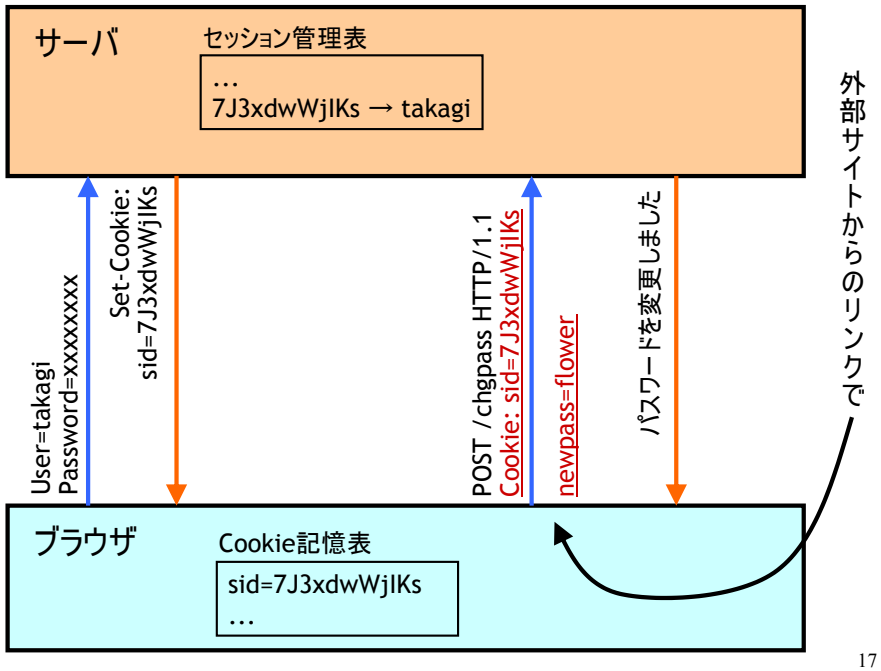
- Thomas Schreiber, "Session Riding — A Widespread Vulnerability in Today's Web Applications", 2004年12月
  http://www.securenet.de/papers/Session_Riding.pdf
  - *In this paper we describe an issue that was raised in 2001 under the name of Cross-Site Request Forgeries (CSRF) [1]. It seems, though, that 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

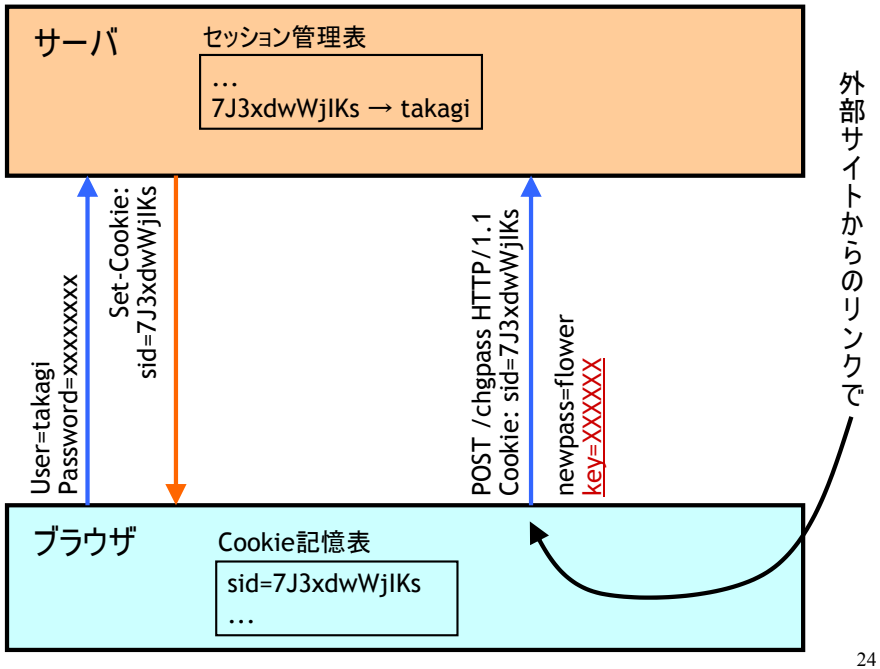
- Webサイト運営者の責任
  - 原状回復可能な被害しか生じない場合や、ユーザに被害が及ばない場合、「荒らし対策」と同じ立場をとることは許されると考えられる
- 製品開発者の責任
  - Webアプリ型のソフトウェア製品にCSRF脆弱性がある場合
  - 製品のユーザ = 当該Webアプリの管理者（運営者）
    - 管理者のみに被害が生じる場合であっても、製品開発者には脆弱性に対応する責任がある（場合がある）と考えられる
    - 原状回復可能な被害しか生じない場合は?

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

適法な脆弱性存在推定手段

- 関係法令
  - 不正アクセス禁止法
  - 刑法 電子計算機損壊等業務妨害罪
  - 刑法改正案 不正指令電磁的記録作成および供用（未成立）
- 確認手段
  - 自分のアカウントで正規の手順でログインする
  - Referer:の送出を止めても動作するかを確認する
  - 重大な状態変更画面の最終ページのHTMLを読み、hiddenパラメタに秘密情報（予測困難な値）が含まれているかを調べる
- 注意点
  - 自分のアカウントだけを使用する
  - 罠のリンクを他人に踏ませない

不正アクセス行為の禁止等に関する法律

(平成11年8月13日公布、平成12年2月13日施行)

- 第三条（不正アクセス行為の禁止）
  - 一　アクセス制御機能を有する特定電子計算機に電気通信回線を通じて当該アクセス制御機能に係る他人の識別符号を入力して当該特定電子計算機を作動させ、当該アクセス制御機能により制限されている特定利用をし得る状態にさせる行為（当該アクセス制御機能を付加したアクセス管理者がするもの及び当該アクセス管理者又は当該識別符号に係る利用権者の承諾を得てするものを除く。）
  - 二　アクセス制御機能を有する特定電子計算機に電気通信回線を通じて当該アクセス制御機能による特定利用の制限を免れることができる情報（識別符号であるものを除く。）又は指令を入力して当該特定電子計算機を作動させ、その制限されている特定利用をし得る状態にさせる行為（当該アクセス制御機能を付加したアクセス管理者がするもの及び当該アクセス管理者の承諾を得てするものを除く。次号において同じ。）
  - 三　電気通信回線を介して接続された他の特定電子計算機が有するアクセス制御機能によりその特定利用を制限されている特定電子計算機に電気通信回線を通じてその制限を免れることができる情報又は指令を入力して当該特定電子計算機を作動させ、その制限されている特定利用をし得る状態にさせる行為

電子計算機損壊等業務妨害罪

- 刑法第二百三十四条の二
  - 人の業務に使用する電子計算機若しくはその用に供する電磁的記録を損壊し、若しくは人の業務に使用する電子計算機に虚偽の情報若しくは不正な指令を与え、又はその他の方法により、電子計算機に使用目的に沿うべき動作をさせず、又は使用目的に反する動作をさせて、人の業務を妨害した者は、五年以下の懲役又は 百万円以下の罰金に処する
- 脆弱性の指摘と業務妨害との関係

不正指令電磁的記録に関する罪（改正案）

- 第十九章の二　不正指令電磁的記録に関する罪
  （不正指令電磁的記録作成等）
  第百六十八条の二
  - 人の電子計算機における実行の用に供する目的で、次に掲げる電磁的記録その他の記録を作成し、又は提供した者は、三年以下の懲役又は五十万円以下の罰金に処する。
    - 一　人が電子計算機を使用するに際してその意図に沿うべき動作をさせず、又はその意図に反する動作をさせるべき不正な指令を与える電磁的記録
    - 二 前号に掲げるもののほか、同号の不正な指令を記述した電磁的記録その他の記録
  - 2　前項第一号に掲げる電磁的記録を人の電子計算機における実行の用に供した者も、同項と同様とする。
  - 3　前項の罪の未遂は、罰する。
- （不正指令電磁的記録取得等）
  第百六十八条の三
  - 前条第一項の目的で、同項各号に掲げる電磁的記録その他の記録を取得し、又は保管した者は、二年以下の懲役又は三十万円以下の罰金に処する。

「抵触しないと推察される行為の例」

- 「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」p.21
  （2）不正アクセス禁止法に抵触しないと推察される行為の例
  - 1）ウェブアプリケーションの利用権者が、正規の手順でログインするなどして通常のアクセスをした際に、ブラウザとサーバとの通信の内容を観察したところ、それだけで脆弱性の存在を推定できた場合。
  - 2）ウェブページのデータ入力欄にHTML のタグを含む文字列を入力したところ、入力した文字列がそのまま表示された。この段階ではアクセス制御機能の制限を回避するに至らなかったが、悪意ある者に別の文字列を入力されれば、このサイトにセキュリティ上の問題が引き起こされかねないと予想できた場合。
  - 3）アクセス制御による制限を免れる目的ではなく、通常の自由なページ閲覧を目的として、日付やページ番号等を表すと推察されるURL中の数字列を、別の数字に差し替えてアクセスしてみたところ、社会通念上、本来は利用できてはならないはずと推定される結果が、偶発的に起きてしまった場合。（ただし、積極的に多数の数字列を変えて試す行為等は、制限を免れる目的とみなされる可能性があります。）

パブリックコメントより

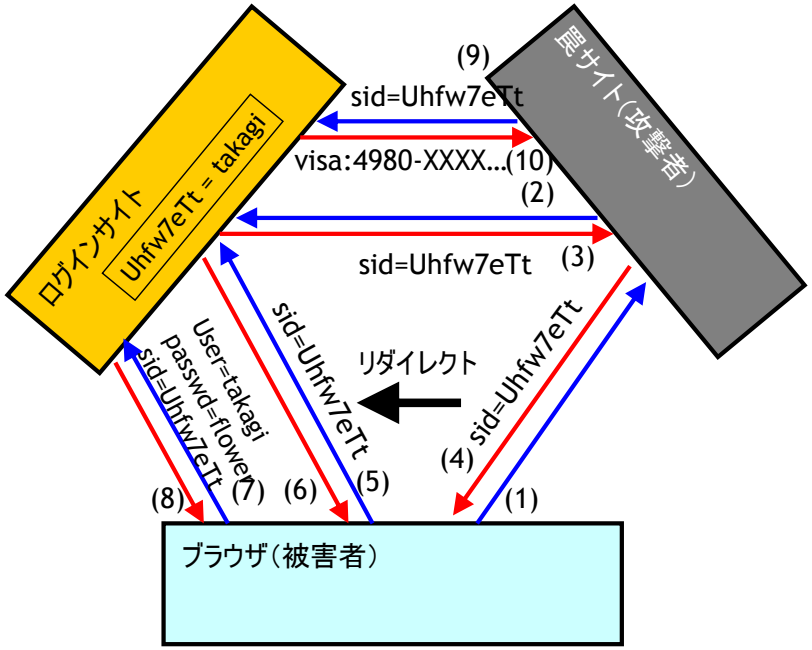
- 「ソフトウエア等脆弱性関連情報取扱基準（案）」等に関するパブリック・コメント（意見募集）の結果について http://www.meti.go.jp/feedback/data/i40706aj.html より
- 「善意の脆弱性発見者を保護する旨を主旨に明記すべき。」
  - 「発見者については、本制度が経済産業省告示を前提として検討されていることから勘案すると、当省としては既存法令に抵触しない範囲内で行動することを要請する以外にないと考える。ただし、届出後の発見者の個人情報の適切な管理・取扱いや、届出内容に関する受付機関からの照会等の発見者負担の軽減等に取り組んでまいりたい。」
- 「ウェブアプリケーションの脆弱性における発見者基準について、「違法な方法により脆弱性関連情報を発見又は取得しない」というだけでは基準として不十分であるように思われる。ウェブアプリケーションの場合、その脆弱性を偶然発見する事は少なく、意図的にそのホームページを調べることで脆弱性が明らかとなることが多いことから、発見者の遵守すべき事項を明確に規定する必要がある。
  さらに、発見された脆弱性については厳格に管理するため、「原則開示しないこと。開示する場合は受付機関の許可を得るようにする」等の発見者の責務を明確に規定する必要がある。」
  - 「本取扱基準（案）は、主旨のところで述べているとおり、関係者とその役割について国が「推奨」する行為を示したものという位置付けであるため、強制力はなく、関係者の自主的な運用に拠るところが大きい。また、本取扱基準（案）は、関係者が行うべき必要最低限の行動基準をとりまとめたものであり、関係者に対しては各種関係の法律に抵触しないように行動することを求めている。なお、本取扱基準（案）の前提としては、偶然発見してしまった脆弱性についての対処を想定しており、ウェブアプリケーションの脆弱性を積極的に発見することを奨励するものでない。
    脆弱性情報の取扱いは、一義的には発見者に委ねられており、「表現の自由」との兼ね合いもあるため、本取扱基準（案）で一律に発見者の言動を規制することは難しく、届出を行った発見者に対して一定期間、特定の言動を差し控えるよう、協力を要請する旨を規定した。」

Session Fixation

- セッション固定（据付?）攻撃
- 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
  - 対策する気はないと公言している
    - 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を使って完全な対策をしているという未確認情報あり

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、Java Servletコンテナの一部(?)など
- 設定で回避?、脆弱性として修正させるべき?

適法な脆弱性存在推定手段

- 方法
  - 自分のアカウントで正規の手順でログインする
  - セッション追跡用パラメタがどれなのかを推定する
  - そのパラメタの発行タイミングがいつなのかを調べる
  - ログイン成功前に発行されている場合は、ログイン成功時にその値が変化しているかどうかを調べる

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

Cross Site Cooking

- Michal Zalewski, Cross Site Cooking, 2006年1月
  http://www.securityfocus.com/archive/107/423375
  - Cookie Monster問題を改めて指摘している
  - Session Fixationの問題も指摘しつつ、別の問題も指摘
  - 正規サイトで使用するのと同名のcookieを、攻撃者の罠サイトで発行する攻撃
- 問題の所在
  - Set-Cookie: の domain 指定の制限範囲の問題
  - 同じ名前（で異なるdomain指定（pathも）の）cookieが複数存在し得る
  - サーバ側でcookieの値を取得するAPIの実装方法よる問題
    - 複数の同名cookieのうち、最初の1個しか返さないAPIの場合
    - 無効なセッションIDしか読めなくなり、ユーザへのDoS攻撃になる