

## 安全なウェブアプリケーション発注のあり方

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/>

1

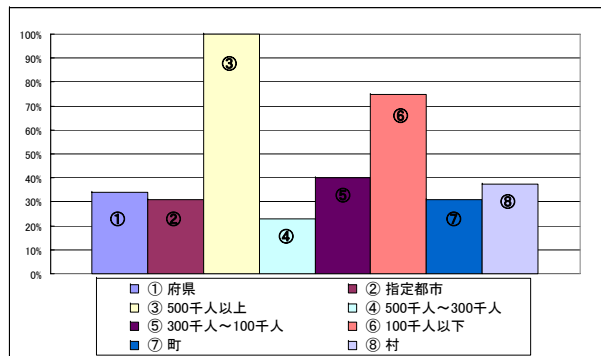
## 経済産業省 受託研究

- 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)

2

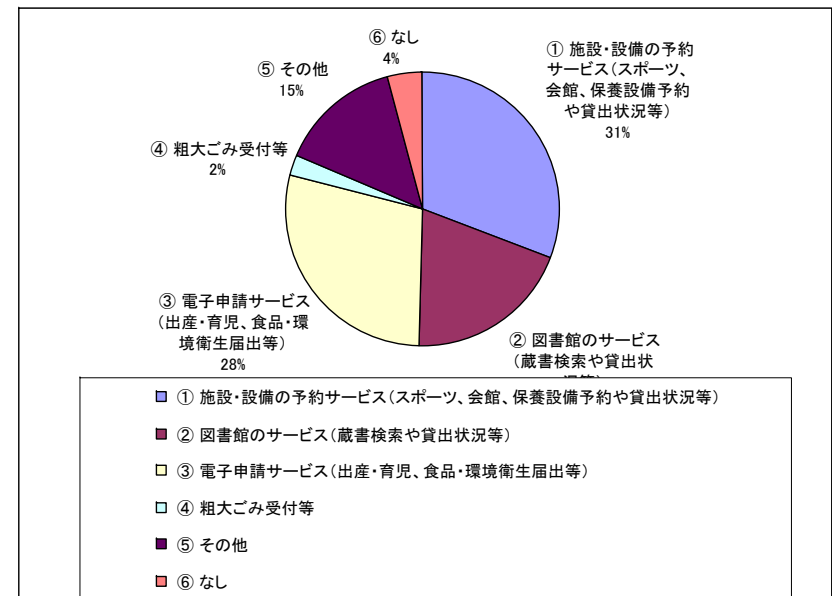
## 自治体の実態調査アンケート

- 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%



3

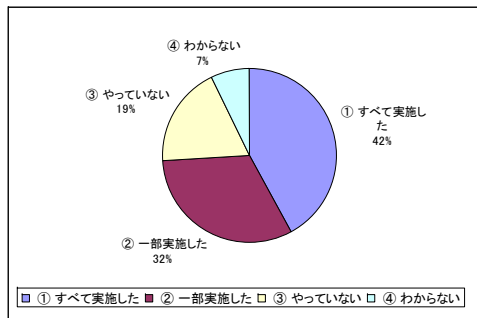
## 「個人情報を入力するアプリケーションはありますか？」



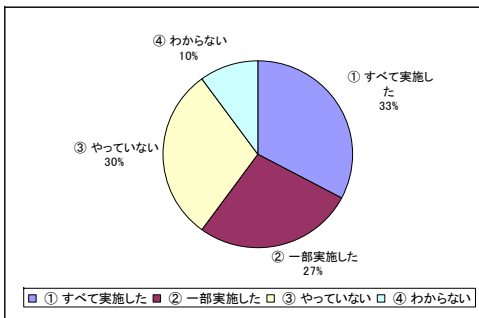
4

# 「脆弱性検査を実施していますか？」

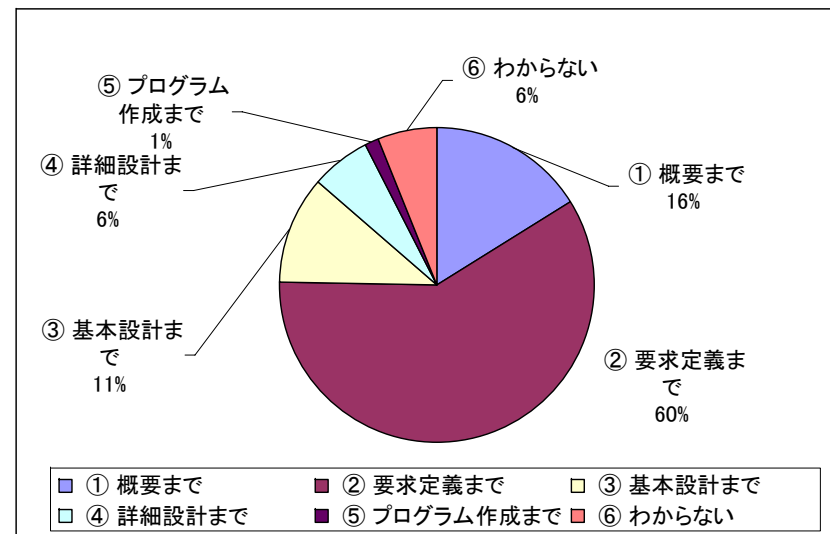
OSの脆弱性について



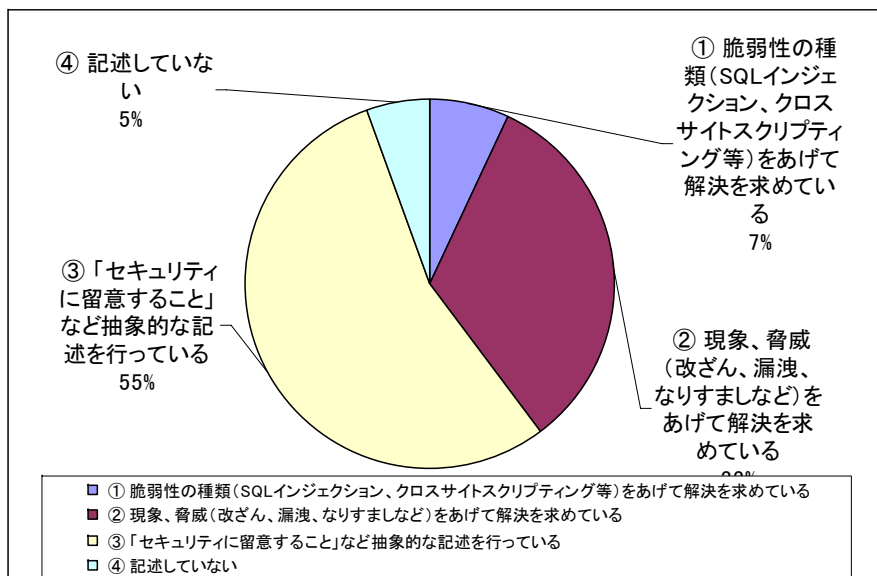
Webアプリケーションの脆弱性について



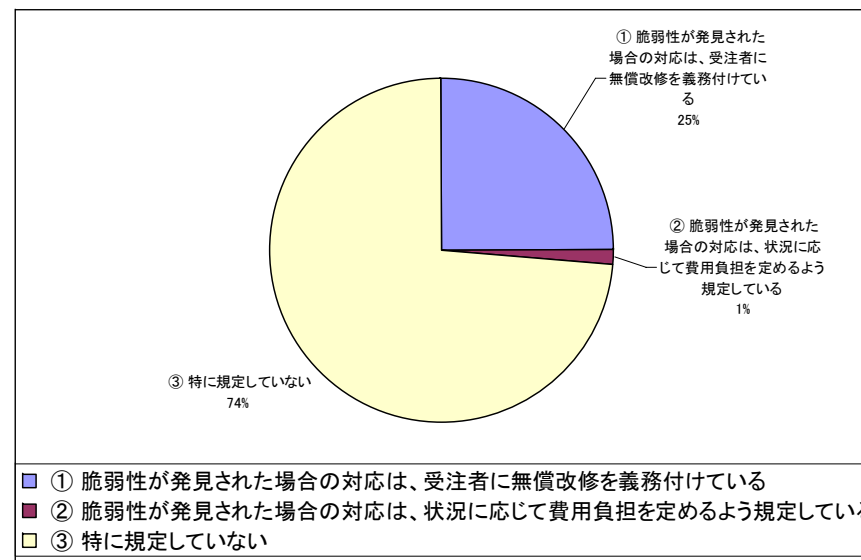
# 「ウェブアプリケーションの開発（改修）発注にあたり（略）通常、職員が直接作成されるのはどの段階までですか？」



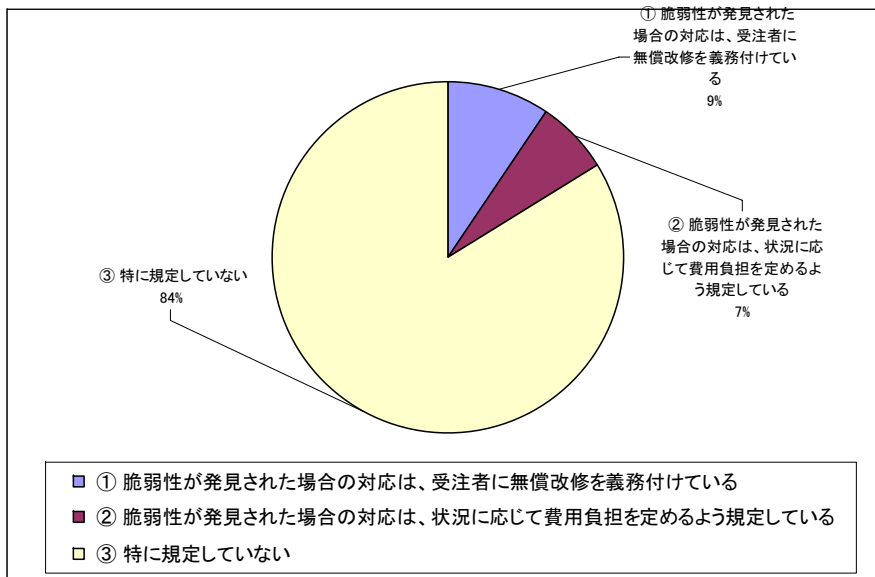
# 「要求仕様書を出している場合、セキュリティに対する要求はどの程度されましたか？」



# 「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？」

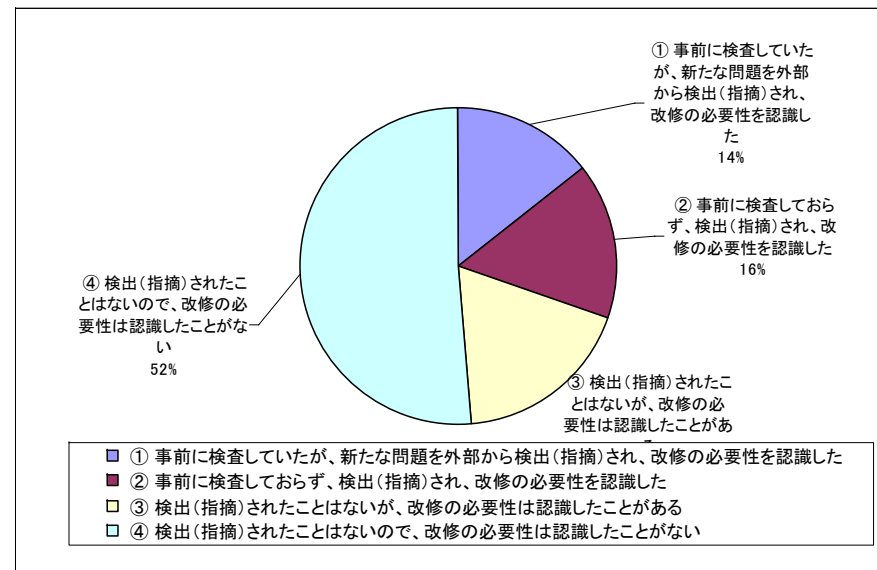


## 「成果物の納品時点で未知である脆弱性について、契約書規定がありますか？」



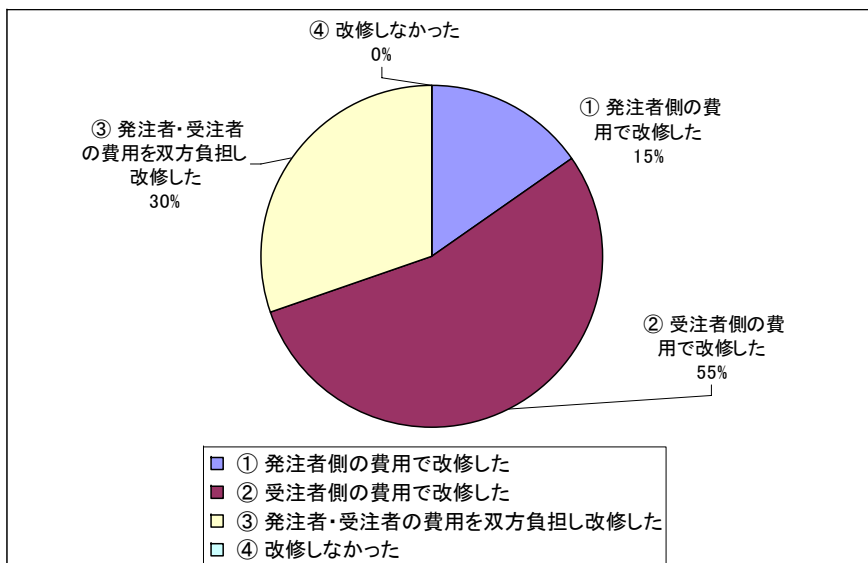
9

## 「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？」



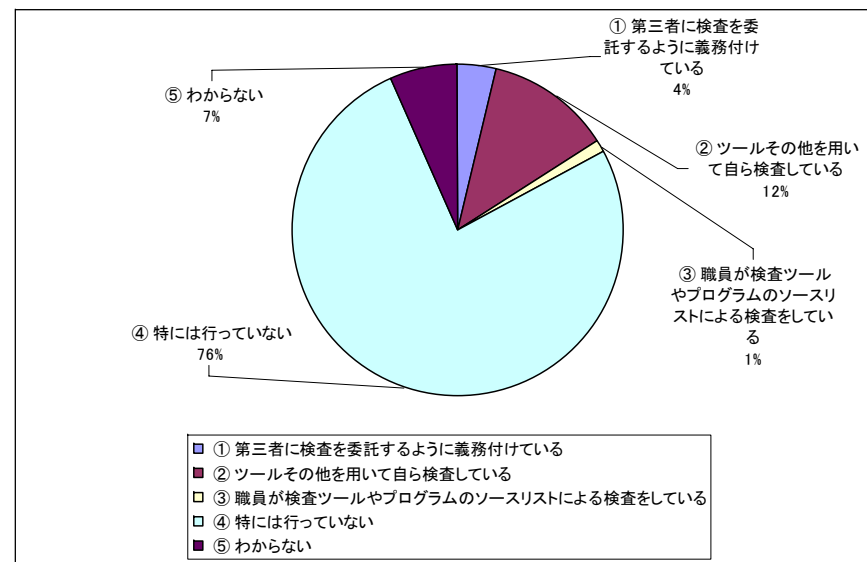
10

## 「実際に改修しましたか？」



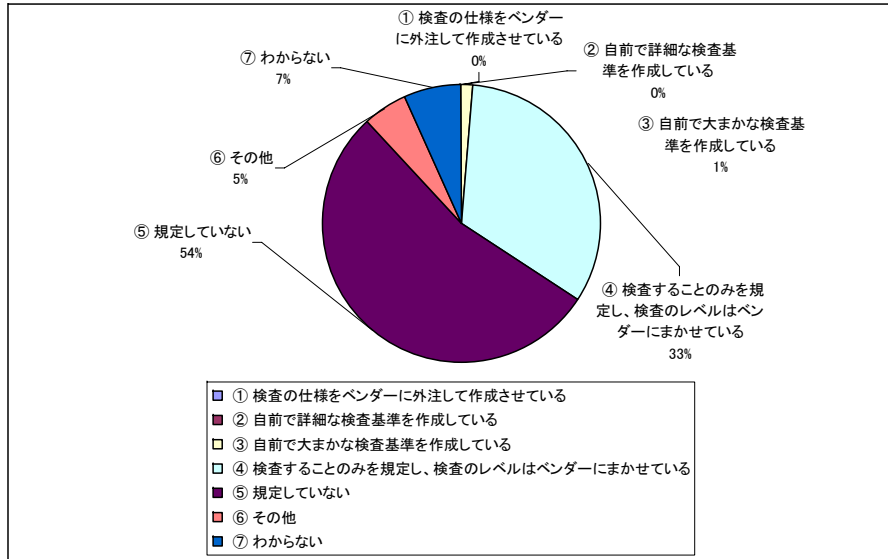
11

## 「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？」



12

## 「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？」



13

## 「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イケない点
- 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）

14

## 「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いが、予算の確保が難しいのが現状（など）

15

## 「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）

16

## 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](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
  - 「ソフトウェ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/>

17

## 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と言えるか

18

## 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](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

19

- 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
    - ユーザー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ー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
    - 正規ユーザーの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。
      - ユーザー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。

20

## 問題意識

- 「〇〇脆弱性がないこと」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

21

## 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

22

## 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、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

23

## 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

24

## 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
  - 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る
  - 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
- リダイレクト
  - 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ー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、暗号学的に安全な疑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)

25

## 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

26

## 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?)

27

## 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

28

## 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

29

## 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サーバを稼働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

30

## 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

31

## 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

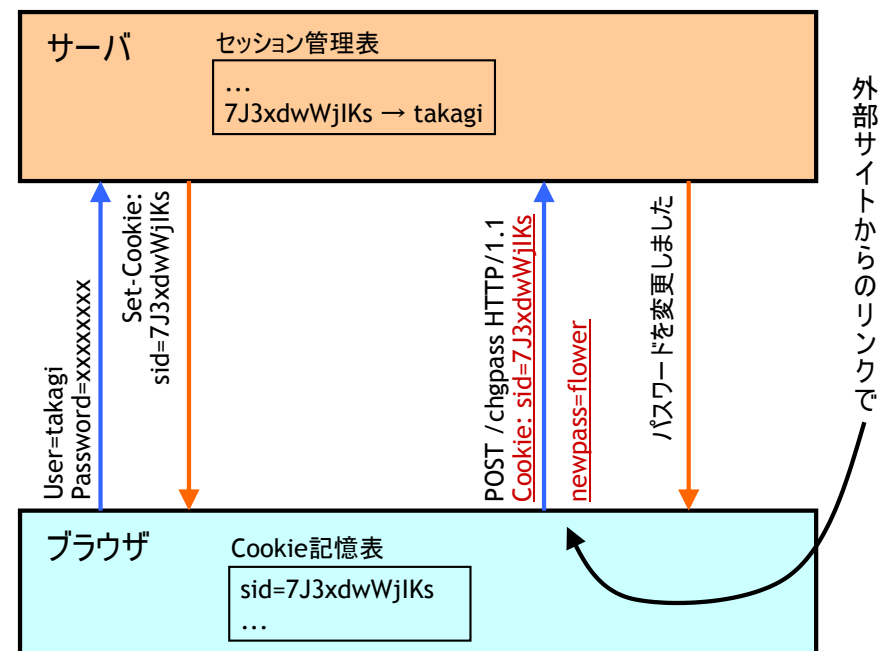
- 目的
  - セッションIDを格納するcookieにsecure属性を付けさせる
  - セッションIDをhiddenパラメータに格納する場合、格納ページと送信先ページが https:// であるように
- 例外に関する考察
  - ひとつのWebアプリでセッションが二重構成になっている場合
    - ログイン前からログイン中まで引き継がれるセッションと、ログイン中だけに対応するセッションの二つ
    - ログイン中のセッションについてのみこの要件を適用
  - ログイン状態の確認をセッションIDではなく、認証トークンで行っている場合
    - 「セッションID」を認証トークンに読み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

32

## 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

33



34

## 画面設計時から考慮する

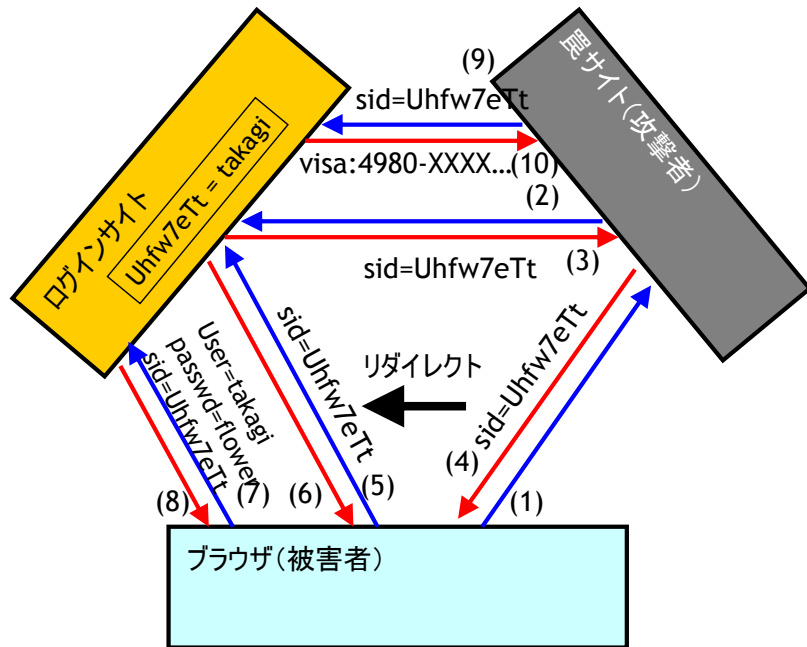
- 画面(アクセス)ごとに以下を検討「特定副作用を有するアクセス」
  - それは「状態変更」するアクセスか
    - セッションで破棄されない状態変更
  - その状態変更は重大か?
    - 重大でない例: ショッピングカートに商品番号と数量を入れる
- 重大な状態変更画面について
  - アクセスはすべてPOSTにする
  - 前のページのhiddenパラメータに秘密情報を自動挿入する

35

## 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

36



37

## 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

38

## 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)

39

## 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、その攻撃成功の可能性を低減するため

40

## 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](http://blogs.msdn.com/michael_howard/archive/2005/01/14/353379.aspx)

41

## 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て実装する(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ーフロー系の脆弱性が生じ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)

42

## 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...

43

## HTTPヘッダを直接出力しない

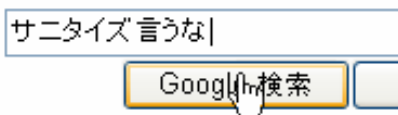
- 目的
  - HTTPレスポンス分割、HTTPヘッダインジェクション脆弱性の防止
  - 必要な場合は、WebアプリサーバやCGIライブラリに用意されているヘッダセットAPIを必ず使用する
    - リダイレクションの際に Location: を自力で出力しないで、専用APIを使用する
    - Cookieの発行は専用のAPIを使用し、Set-Cookie: を自前で送信しない
  - 注意: HTTPヘッダインジェクション脆弱性のあるWebアプリサーバやCGIライブラリを使用しないこと(修正されたものを使用する)

44

## 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
  - 「入力をサニタイズする」という対策は誤り

続きはWebで



45

## 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

46

## その他、検討中のもの

- 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。
- フォーマット文字列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

47

## 位置付け

- 安全な設計・実装の教科書(ノウハウ集)ではない
  - やらないよりはやった方がよいので「多層的防御」として、あれもこれも入れておく.....という性質のものではない
- これを満たさないサイトは脆弱性があるという意味になるものではない
  - 「この設計でないと脆弱性と見なされる」という、脆弱でないための必要条件の要件と、この実装ならば安全にしやすいという、十分条件の要件が混在
- これを満たせばサイトに脆弱性がないという意味でもない
  - 脆弱性発覚時に責任が受注者側にあることを明確にできるよう、契約で使用するもの
- 発注者が使う使わないは自由だが、この要件を必須としてもさほど設計、実装上の困難が生じないレベルを目指す

48