

ソフトウェアのセキュリティ欠陥は 誰が直すのか

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

高木 浩光

<http://staff.aist.go.jp/takagi.hiromitsu/>

1

「セキュリティが重要」とは？

- 「これからの時代、セキュリティが重要だ」と聞いて、あなたは何を連想しますか？

A

セキュリティ技術

の開発、採用

B

セキュリティ欠陥

の防止、排除

「セキュリティ」にはこれら対照的な2つの文脈がある

2

技術開発と欠陥防止

- セキュリティ技術**（の開発、採用）
 - 暗号、認証、侵入検知、アクセス制限、ウイルスフィルタなど、危険を機械的に排除しようとする手法や製品
 - 「技術で安全を確保するから、(きわどいけれど)便利なることをやろう」と利用者を説得する
- セキュリティ欠陥**（の防止、排除）
 - 明示的、暗黙的な仕様上、安全であるはずのものが、実際には安全でないという、ソフトウェアの欠陥
 - これを解決せずして、「安全を確保するので……」と説得するのは、いわば過失による詐欺

3

技術開発で欠陥を防止できるか

- ある程度は可能だが完全(十分)ではない(もしくは現実的でない)
 - 安全な言語の選択によるbuffer overflow排除
 - コード監査ツールによる欠陥の発見
 - 脆弱性スキャナによる欠陥の発見
 - 検証理論の適用
- 誤った認識に基づく愚かな考え
 - セキュリティ技術の研究開発には予算を投入するが、セキュリティ欠陥の予防には投入しない

4

技術採用で欠陥を無視できるか

- ある程度は可能だが完全(十分)ではない
 - ウイルスフィルタ
 - 広範無差別攻撃には対応できるが、個別プログラムによる特定対象を狙った攻撃に対応する性質のものではない
 - ファイアウォール
 - 本質的には管理コストを下げるための道具に過ぎない
 - 侵入検知
 - 検知した後どうするのか
- 誤った認識に基づく愚かな考え
 - ファイアウォール、ウイルスフィルタ、侵入検知、高度な暗号システムを導入したので対策は万全と考える

5

欠陥の防止と排除

- 欠陥を未然に防ぐ
 - なぜできないのか、どうすればできるのか
- 欠陥を見つけ出す
 - 誰が見つめるのか、どうやって見つけるのか
- 欠陥を修理する
 - 誰が直すのか、誰が直す必要性を告知するのか

6

脆弱性はなぜ生まれるのか

- 品質管理上の問題に帰着されるもの
 - buffer overflowなど、既に原因と解決策が広く知られているもの
 - うっかりミス、一部のスキルの低いコーダにより発生
 - 他の一般的なバグと同様に、品質管理、プロジェクト管理の問題に帰着され、コストとのトレードオフとなる
- 不知が原因によるもの
 - 脆弱性の発生要因は日々新たなものが発見される
 - 知らなかったというだけで熟練開発者でも欠陥を生む
 - 周知の仕組みが確立していない

7

情報セキュリティ欠陥の特殊性

- 100%の再現性がある
 - 自動車の構造的欠陥や、家電製品の発火の問題等では確率的にしか事象が発生しないのと対照的
 - 誰にでも一発で欠陥の存在を確認できる(確認方法が知らされれば)
- 欠陥の要因が日々激しく増加する
 - ハードウェア製品の物理的欠陥の原因に比べ、急激に欠陥の要因が増加している(ドッグイヤー)
 - OSや開発言語、開発環境、使用ライブラリ等に複雑に依存していて、それらの影響を受ける
- 開発に従事し得る者の絶対数が多い

8

不知が原因の脆弱性の典型事例

- 「PostPet」、「Becky!」、「Shuriken」ほかいくつかのメールソフトに存在していた脆弱性
 - Windowsにおいては、固定パスのディレクトリにインターネットから取得したファイルをそのまま置いてはならないという知識が欠けていたために発生
- 「CyberSupport」に存在していた脆弱性
 - ActiveXコントロールの開発はネットアクセスを行わないものであってもセキュリティ上の配慮が必要という知識が欠けていたために発生
- 「クロスサイトスクリプティング」の問題
 - CERT CA-2000-02ではほとんど周知されなかった
 - 最近では広く知られるようになったので、もはや品質管理上の問題となったといえる

9

可能な現実的欠陥防止策

- 不知による欠陥は、低コストで防止可能(では?)
 - 他山の石以て玉を攻むべし
 - Outlook Expressに過去に発覚した脆弱性の原因を会得していたなら、同類機能の他社製品は、同じ欠陥を生むことはなかった(あるいは自力で直せた)
 - 日々発覚する他社製品の脆弱性をウォッチ
 - 開発者全員がウォッチする必要はない(それは無理)
 - ウォッチ専門のスタッフを置く(全社横断的に)(少人数で十分)
 - ある製品に欠陥が見つかったら自社の同類製品に同じ欠陥がないか確認を実施または要求

10

脆弱性パターンの集積

- CERT Advisory.....
 - 製品ごとの脆弱性情報は提供してきたが、脆弱性の原因ごとの整理は(ほとんど)行われていない
- 脆弱性の原因パターンごとの整理が必要
 - IPA: セキュアプログラミング講座
<http://www.ipa.go.jp/security/awareness/vendor/programming/>
 - OWASP: The Open Web Application Security Project
<http://www.owasp.org/>
 - 産業技術総合研究所: SecurIT
<http://SecurIT.etl.go.jp/>

11

ISO/IEC 15408で解決するか

- 品質管理上のは解決されるのだろう
 - それだけのコストを払う覚悟がある
- 不知が原因のものは解決されるのか?
 - 検証が必須の脆弱性パターンのリストはあるのか?
 - 特に心配なのはWebアプリケーションの脆弱性

12

Webは脆く弱い

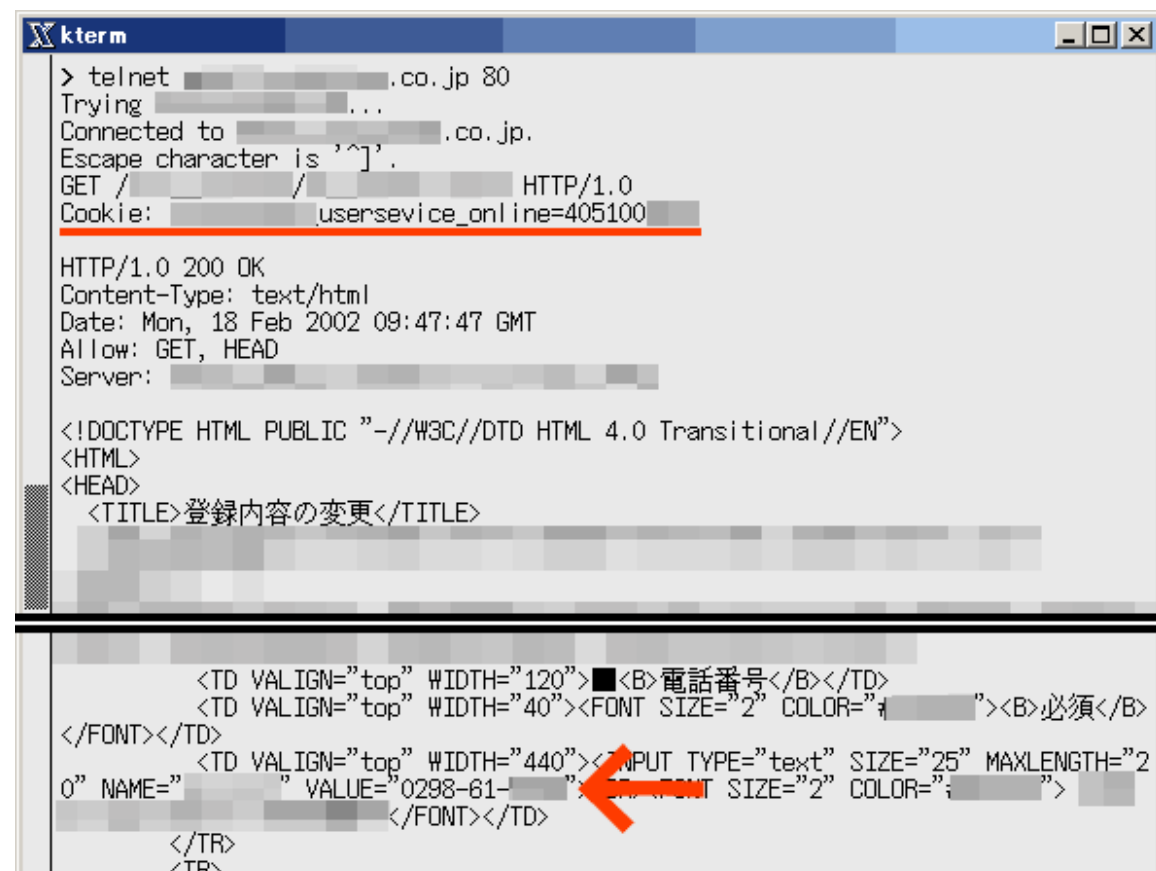
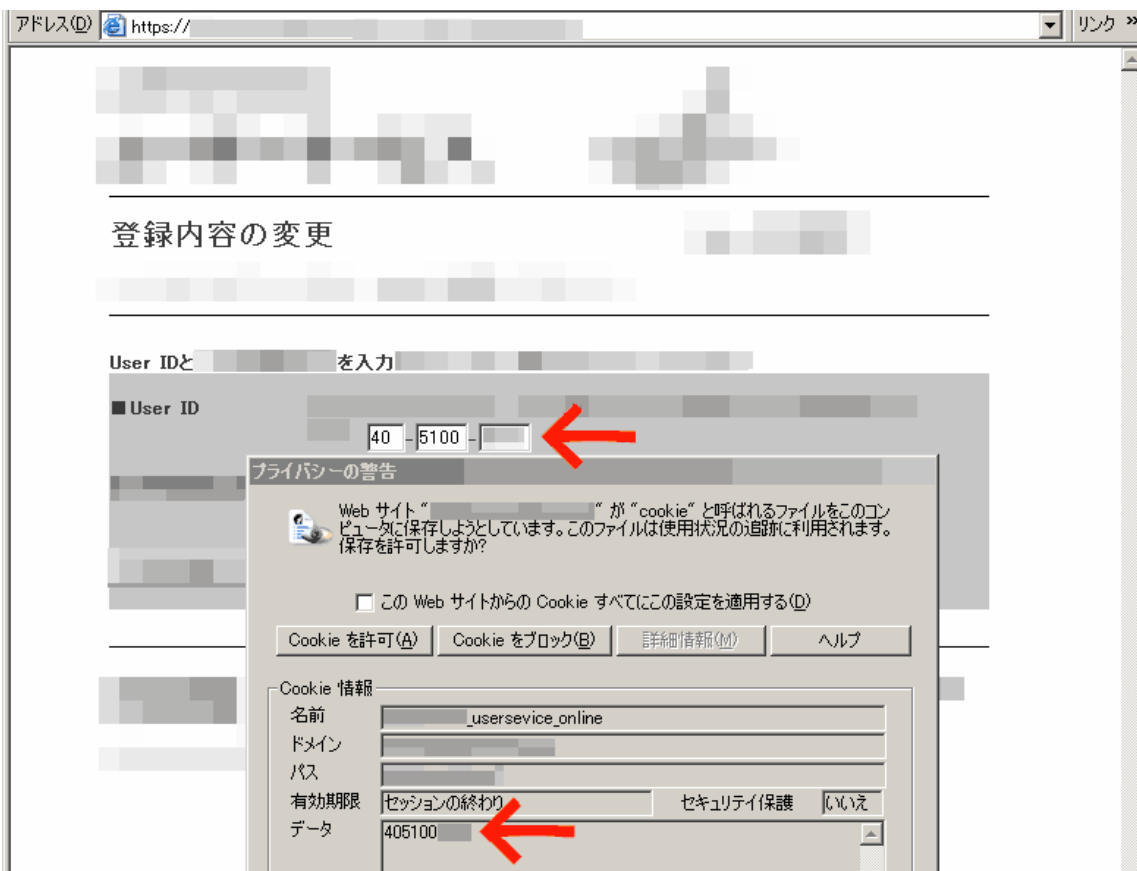
- Webの仕組みは本質的に危うい
 - ブラウザは、同時に複数のサイトにアクセスできる
 - cookieなどの機構の設計が汎用的過ぎて、使い方を誤るとセキュリティ欠陥となる
- 個々のサイトごとに個別に設計されている
 - アクセス制御プロトコルを素人が設計している
- HTTPの設計が悪かったのか
 - 汎用性が高かったからこそ普及した

13

Webアプリの欠陥パターン

- 例えば
 - 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.etl.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**
 - 他、クロスサイトスクリプティング脆弱性など

14



発注者の立場

- 発注時から、後に欠陥が発覚することを想定して、契約するのがよい
- しかし、実装レベルの欠陥を防ぐために、セキュリティ仕様をどう書けばよいのか
 - 発注者は通常、実装レベルにどんな欠陥があり得るのかなど知らない
 - 中途半端なセキュリティ仕様ではかえって受注者の責任を回避させてしまいかねず、むしろ書かずにおいたりすることがある
- チェック項目の標準が必要

17

安全なWebアプリ開発31箇条の鉄則

- JNSAセキュリティセミナー in Internet Week 2002
日本ネットワークセキュリティ協会, 2002年12月17日
- 4グループの鉄則
 - 画面設計の鉄則
 - セッション実装の鉄則
 - 万が一に備える鉄則
 - SSL使用時の鉄則 (1件は取り消し)
- 画面設計の鉄則
 - 1. **アドレスバーを隠さない** 2. FRAMEを使わない 3. ブラウザの設定変更を強要しない 4. ユーザ名だけでログインさせない 5. 認証キーには秘密情報を 6. パスワードを4桁数字にしない 7. 認証エラーで存在を暴露しない 8. 認証で秘密情報を暴露しない 9. パスワードリマインダの鉄則

18

- セッション実装の鉄則
 - 10. 公開ディレクトリに置かない 11. 全てのアクセスを認証チェック 12. アクセス許可対象者を限定する 13. URLに秘密情報を入れない 14. セッションIDを使う 15. セッションIDは予測不能に 16. 状態はすべてサーバ側に持たせる 17. XSS脆弱性を排除する 18. HTMLタグの入力をさせない 19. ログアウト機能を用意する 20. 際どい操作はPOSTにする 21. ログイン前にセッションID発行しない
- 万が一に備える鉄則
 - 22. Cookieの有効期限を短く 23. Cookieの有効ドメインを狭く 24. POSTによる画面推移方式を検討 25. 個人情報閲覧に再度パスワードを 26. カード番号は全桁表示しない 27. パスワード変更には現パスワードを
- SSL使用時の鉄則
 - 28. 自己発行証明書で運用しない ~~29. BASIC認証使用時は全部httpsに~~ 30. Cookieのsecureフラグを立てる 31. 何を暗号化するかを明確に

19

アドレスバーを隠さない

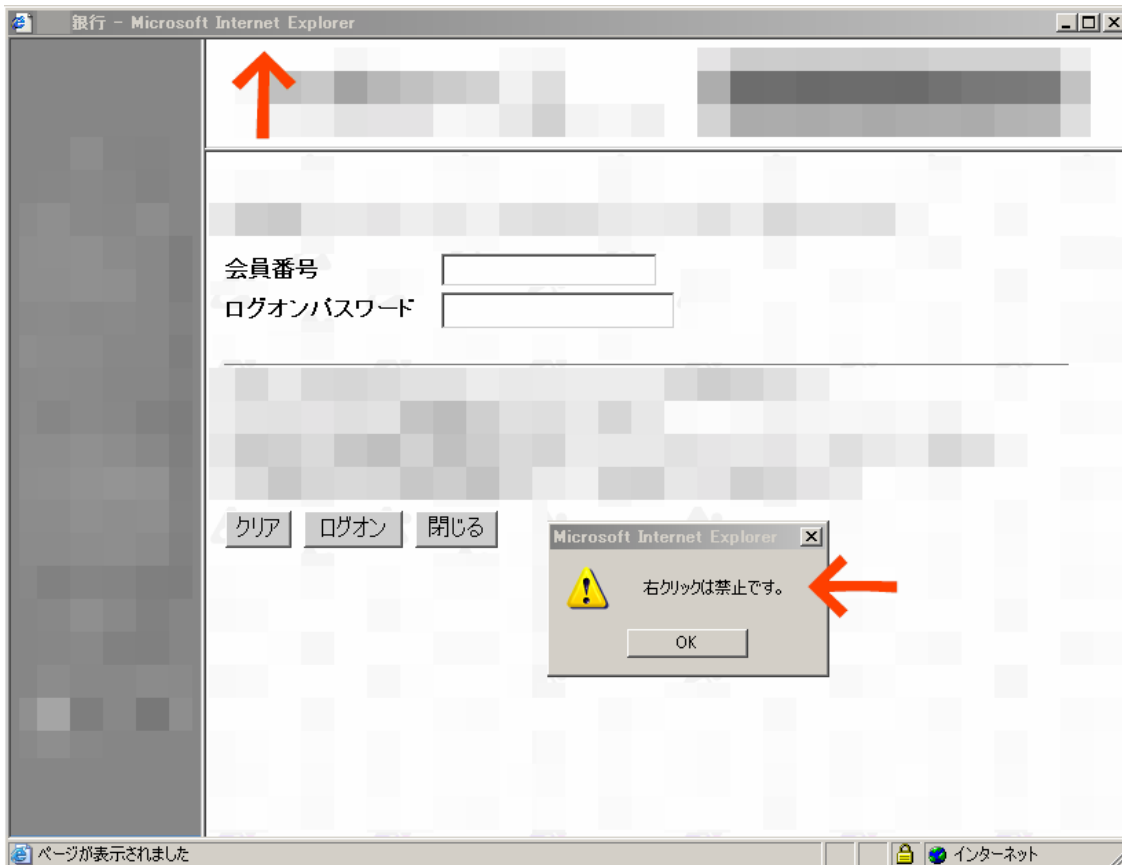
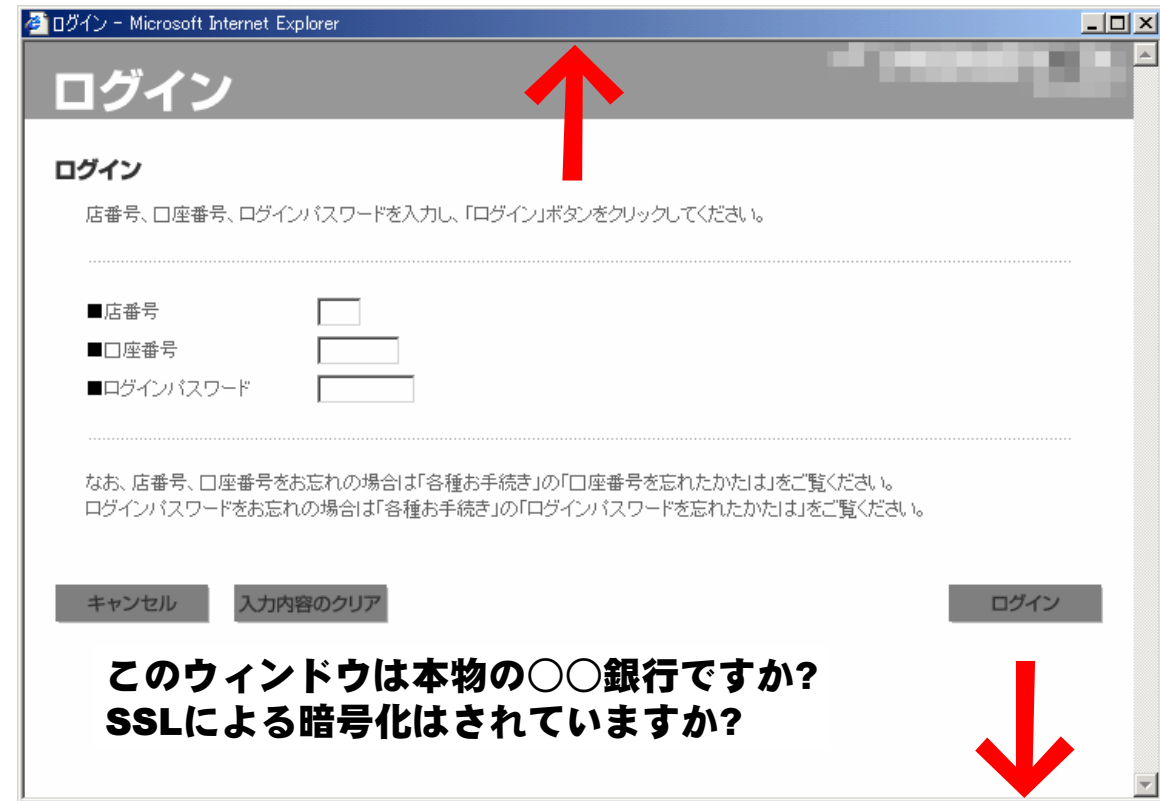
- ドメイン名は、ブランド名、社名に相当する、利用者から見た信頼の起点
- 隠すことに何ら意義はない
 - URL中のパラメタ部を弄られたくない?
 - 弄られるとセキュリティ上の問題が起きるシステムは、アドレスバーを隠したところで攻撃される
 - 戻るボタンを押されたくないのと同じ理由?
 - 推奨しない操作により利用者の利便性が損なわれる(セッションが途切れるなど)のは、利用者の責任
- 同じ理由で: 右クリックの無効化をしたりしない

20

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
 - 「こちらは〇〇銀行です。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

21



わかっていてもできない

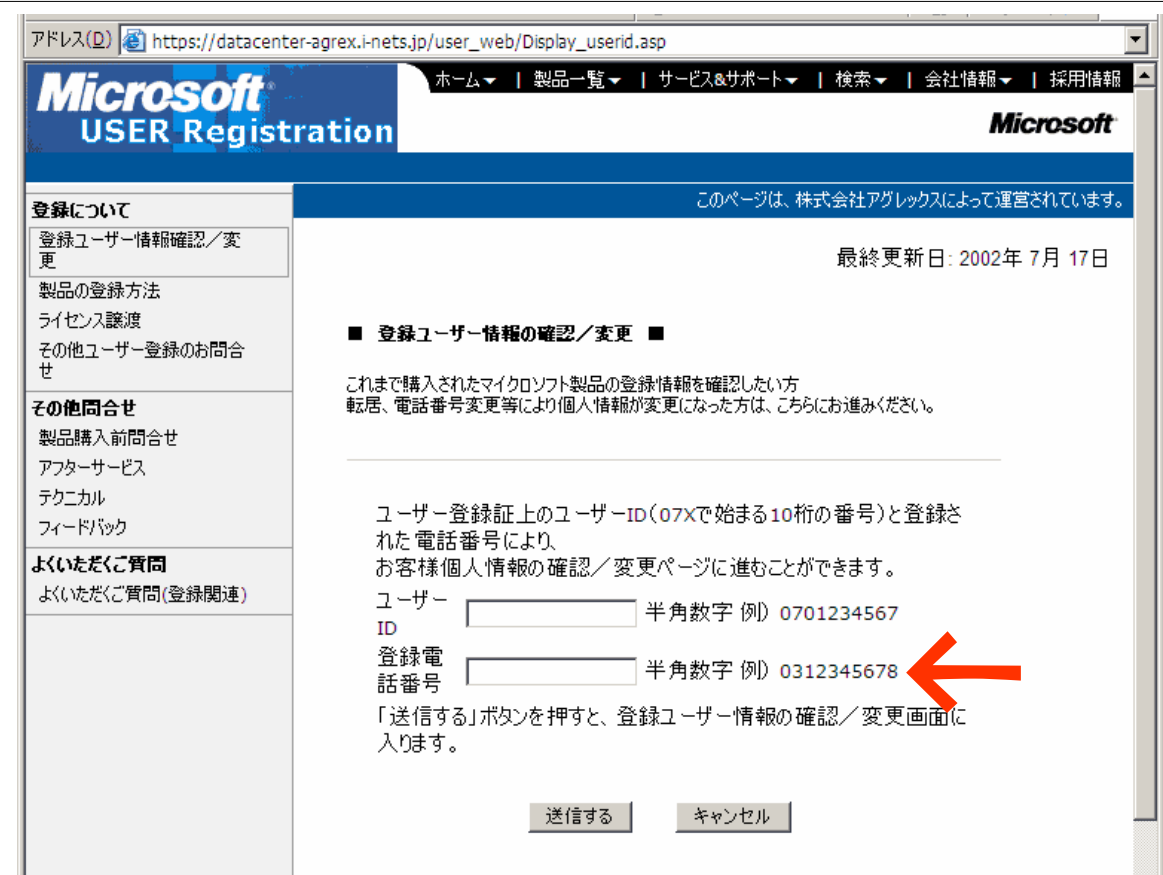
- 現場の技術者が危険性を理解していても、お客様の要求仕様を否定できない(しようとならない)ことがあるようだ
 - 日本人の「コミュニケーション不全」気質の問題か?
 - 権威付けされたチェックリストが援護射撃となるか
- その種の脆弱性(鉄則)
 - アドレスバーを隠さない
 - FRAMEを使わない
 - 認証キーには秘密情報を
 - パスワードを4桁数字にしない

24

認証キーには秘密情報を

- パスワードのないサービス
 - 本人確認のために入力させるキーが、例えば、ユーザ番号と登録した電話番号になっているシステム
 - 実例
 - ジャストシステムのユーザ登録の変更画面
 - マイクロソフトのユーザ登録の変更画面
 - 旅の窓口のログイン画面
- 誰がこれを止められるのか
 - 明白に問題だと指摘できるのか
 - 程度問題であり妥当だと判断されているのかも

25



法における識別符号の定義

- 不正アクセス禁止法第二条2:
 - この法律において「識別符号」とは、(中略)
 - (1) 当該アクセス管理者において当該利用権者等を他の利用権者等と区別して識別することができるように付される符号であって、
 - (2) 次のいずれかに該当するもの又は次のいずれかに該当する符号とその他の符号を組み合わせたものをいう。
 - 当該アクセス管理者によってその内容をみだりに第三者に知らせてはならないものとされている符号
 - (以下略)

27

警察庁担当者の見解

- Q: 電話番号をパスワード代わりにするのは、不正アクセス禁止法の「識別符号」の要件を満たしていないのではないかと?
- A: 電話番号は(1)の条件、(2)の条件も満たさない。会員番号は(1)の条件は満たす。会員番号が(2)の条件を満たすかどうか:
- 「会員番号」は、一般に、アクセス管理者からみだりに第三者に知らせないように求められていると認識されるものとは考えがたく、規約等にアクセス管理者が第三者に知らせてはならないものと規定されていないならば、(2)の要件を満たさないと考えられる。したがって、
 - 会員番号が、規約等にアクセス管理者が第三者に知らせてはならないものと規定されていれば、「会員番号、電話番号」は、「識別符号」に当たると考えられる。また、
 - 会員番号が、規約等にアクセス管理者が第三者に知らせてはならないものと規定されていないならば、「会員番号、電話番号」は、「識別符号」に当たらないと考えられる。

警察庁生活安全局セキュリティシステム対策室担当者の見解（結論部分の高木による要約）（2003年1月）

28

この見解を紹介する意図

- 他人の会員番号と電話番号を入力してログインしても、不正アクセス禁止法違反にならないから、やってよいということではない
- 法による秩序の維持の恩恵に与れない
このようなシステム設計は不適切である

と言いたい
(規約で回避できるのかもしれないが)

29

改めて問う

- ISO/IEC 15408で解決するのか?
プライバシーマーク制度で解決するのか?
- 実装方式に踏み込んだ具体性のあるガイドライン、標準の制定が必要なのでは?
- 業界ガイドラインの事例:
 - 日本銀行検査局:「インターネット利用システムにおける情報セキュリティ対策のチェックポイント」(2000年4月)
 - 「インターネットを利用した個別システムにおける情報セキュリティ対策をチェックするうえでの主な項目を整理したもの」とされ、「技術面での対策のポイント」が挙げられている
 - そこには、ファイアウォール、暗号、電子署名、業務妨害対策の4点が挙げられているのみで、まったく不十分

30

欠陥は誰が見つかるのか

- 開発、販売当事者が自ら見つける事例は少ない
 - そう見えるだけで、単に表沙汰にされないため、気づかないだけなのかもしれない
 - 開発担当技術者自身は見つけて知ってはいるが、コストの都合で直させてもらえないのかもしれない
 - フリーソフトウェア、シェアウェアなどの個人の意思が直接働くものでは、自力で見つけ修正されることがある
- 製品の多くの欠陥が第三者によって発見されている
 - ひとりのユーザである技術者による発見
 - 客ですらない第三者に発見されることもある

31

脆弱性発覚の実態

- BUGTRAQ: メーリングリスト
<http://online.securityfocus.com/archive/1>
 - 流れるメール数: 約80通/週
 - DB登録される脆弱性数: 約20件/週
 - 多種多様な製品の欠陥に関する情報が流れる
 - ここ数年で急速に増加 —— 脆弱性情報のオープン化が進んでいるようだ
 - Webアプリケーション構築環境や、特定のCGIプログラムの脆弱性など、マイナーにも思えるソフトウェアの欠陥も多数流れている

32

わが国の状況

- BUGTRAQ-JP: LACの三輪氏がモデレート
 - 流れるメール数: 1.3 通/週 (Newsletterの和訳を除く)
(2002年は66通が流れた)
- 報告が少ない —— なぜか?
 - 日本には発見する技術を持つ者が少ない??
 - 日本の製品には欠陥が少ない???
 - 日本にはそもそも独自製品が少ない?
 - 基幹ソフトウェアは輸入品ばかり?
 - 日本には「告発」を避ける文化がある?
 - 企業活動として発表する者が少ない(LAC社のみ)
 - 個人で発表する者が少ない

33

「告発」は必要なのか

- ベンダーが修正しない場合
- ベンダーが修正はしたがその事実をユーザに知らせない場合
- ベンダーが修正と告知はしたものの、危険性を正しく伝え切っていない場合
- 発見者による告知:
 - ユーザに正確な知識を与え、回避策をとる機会を与えるという公益性がある

34

不適切な告知の事例

- 「ウイルスに対する高い安全性」を積極的にアピールしている大手ソフトウェアベンダーのメールソフトに、添付ファイルが勝手に起動してしまう脆弱性が発覚 (2002年9月)
 - この欠陥は、1999年7月に国内で他のメールソフトに発見され毎日新聞などでも報道された古典的脆弱性と同一原理の欠陥
 - 外部から指摘され、欠陥修正パッチをリリースしたが、
セキュリティをさらに強化しました!
という台詞で告知した

35

- ソニーのVAIOにプリインストールされたソフトに任意コマンド実行の脆弱性 (2002年1月)
 - 「[セキュリティ強化プログラム](#)」という名の欠陥修正CDを配布

- 「セキュリティ強化」とは、**現在もそれなりに安全**だが、**もっと安全にする**という意味のはずではないか?
- 実際には、悪意あるWebサイトを訪れただけでファイルを抹消されるなどの脅威があった

- セキュリティ脆弱性の修正パッチをCDで配布するのは他に類を見ない適切な対応といえる



36

報告と対応についての標準

- 世界的にも未整備(合意がない)
 - CERT/CCの脆弱性公表ポリシー
 - Internet-Draft “Responsible Disclosure Process”
- わが国の状況
 - まだ何もない
- 整備すべきもの
 - ベンダーの告知方法についてのガイドライン
 - 発見者による公表方法のガイドライン

37

どのように情報を公開するか

- 攻撃方法がある程度公開されないと、危険の現実性を認識できない(者が多い)
- 攻撃方法が公開されることへの抵抗感
 - 自分が発見者となったときどのような認識を持つか
 - 経験前と経験後の認識の違い
 - 自分の知らないことを他人が知っていることに対する恐怖
- 報告から何日後まで公表を控えるべきなのか
 - CERTには45日という基準がある

38

性質の異なる2つの脆弱性

- 製品の脆弱性
 - 欠陥を修正したら、ユーザにそのことを正しく伝える必要がある
 - 適切に危険性を伝えなければ、ユーザは面倒な修正作業を行わないことがある
- Webサイトの脆弱性
 - 欠陥を修正した時点で、ユーザの危険は回避される
 - 欠陥が存在していたことを告知する必要はある?
 -

39

発見者の保護

- 内部告発者保護の流れ
 - 日弁連:公開シンポジウム「内部告発者保護制度を考える～重要論点の検証～」より:
 - 昨今、製品の安全性に関する事件等企業不祥事が後を絶ちませんが、その多くは内部告発によって明らかにされています。このようなことから、内部告発者保護制度導入の世論が高まり、国民生活審議会の「21世紀型の消費者政策のあり方」に関する中間報告の中でも、早期実現が提言されています。
- ソフトウェアの脆弱性の場合
 - 発見者は内部でなく外部の者であることが多い
 - 内部告発よりは安全であるはずなのに、それでも、見てみぬふりをせざるを得ない者は多いと考えられる

40

報告に伴う摩擦

- 善意の報告者の視点
 - － 門前払いされる(適切な報告先が存在しない)
 - － 報告しても無視される(対応がない)
 - － 対応がある場合でも、返事がなく、伝わっているのか不安になる
 - 対応がないならば、利用者に危険性を伝えた方がよいのではないかと迷う
 - － 口止めされたら? 脅迫されたら?
 - － 恐喝と誤解されるのではないか

41

報告に伴う摩擦

- ベンダー側の視点
 - － 報告者は何者なのか?
 - － これは恐喝か?
 - － 一個人の指示に従うのは耐え難い
 - 他の不具合(100%の再現性がないもの)では、多数の報告があってはじめて対応する
 - 可能性だけであって実際に被害が出ている証拠がない
 - － 報告内容の意味がわからない
 - － 重大な問題でないように思える

42

業務妨害? 名誉毀損?

- 弁護士の見解
 - － 「脆弱性が存在する事実」を公表しても、公益目的であり、真実性の証明が出来る限り、民事上の不法行為責任にも刑事上の名誉毀損罪の責任にも抵触しない。
 - － 公表前に、注意・自主改善を促すことは上記法的責任を発生させないためには、不必要。
 - － 「注意・自主改善を促したのに、改善せず、脆弱性を隠したまま販売を継続しようとしている事実」を公表することも結論は上の場合と同じ。
 - － 公開が違法かどうかのポイントは、表現の自由－国民の知る権利の保護(憲法21条)と名誉の保護(憲法13条)との衝突の調整という点。あとは妥当性の問題で、いつ、どのように公表するかは裁量で行って良い。

エルティ総合法律事務所 藤谷護人弁護士の見解(高木による要約)(2003年1月)

43

匿名掲示板による暴露?

- しばしば頂くご意見:
 - Q. 匿名で暴露する場があればよいのでは?
 - － 匿名掲示板等での暴露は、信憑性が低く見られ、アナウンス効果が小さいため、ユーザへの危険性の周知が十分に行き届かない(場合が多い)
 - － その結果、脅威の方が増してしまう
 - － ベンダーへの報告の際に、実名で連絡をとってしまうと、告発の際に匿名とすることにあまり意味がなくなってしまう
 - － 他方、ベンダーへの報告の際に、匿名で連絡をとると、信頼して話を聞いてもらえないように思える

44

仲裁機関の必要性

- 米CERT/CCにはそのような機能がある
- 日本の類似名称を名乗るJPCERTにはそのような機能はない
 - 今後できないだろうと私は思う

- アイデアについては、夕方のパネル討論で

45

エンドユーザの立場

- Webを安全に利用するための基本は、じつはあまり知られていない
 - ブラウザの「アドレスバー」のURLを常に目視確認する
 - ブラウザのセキュリティ設定の安全性を下げて使用するよう指示するサイトを信用しない
 - 自己発行のルート証明書を安全でない方法でインストールさせようとするサイトを信用しない

- リテラシ教育カリキュラム
 - どこが作る?

46

IPAの事例

- 電子申請システムで、自己発行のルート証明書をユーザにインストールさせている
<https://www.ipa.go.jp/ipa/contribution/e-ipa.htm>
 - クライアント認証を使用するためというが、なぜ認証サービス事業者から購入しないのか
 - ルート証明書は、VeriSignのサーバ証明書で運用中の <https://> のページから安全に取得可能ではある
- 問題点
 - 認証局運用規定(CPS)がいまだに用意されてない(ずいぶん前から指摘されているのに.....)
 - 「情報処理振興事業協会では、ルート証明書の秘密鍵の管理については、**十分注意を致します**」とWebページに書いているだけ

47

セキュリティ欠陥は誰が直すのか

それはあなたです

48