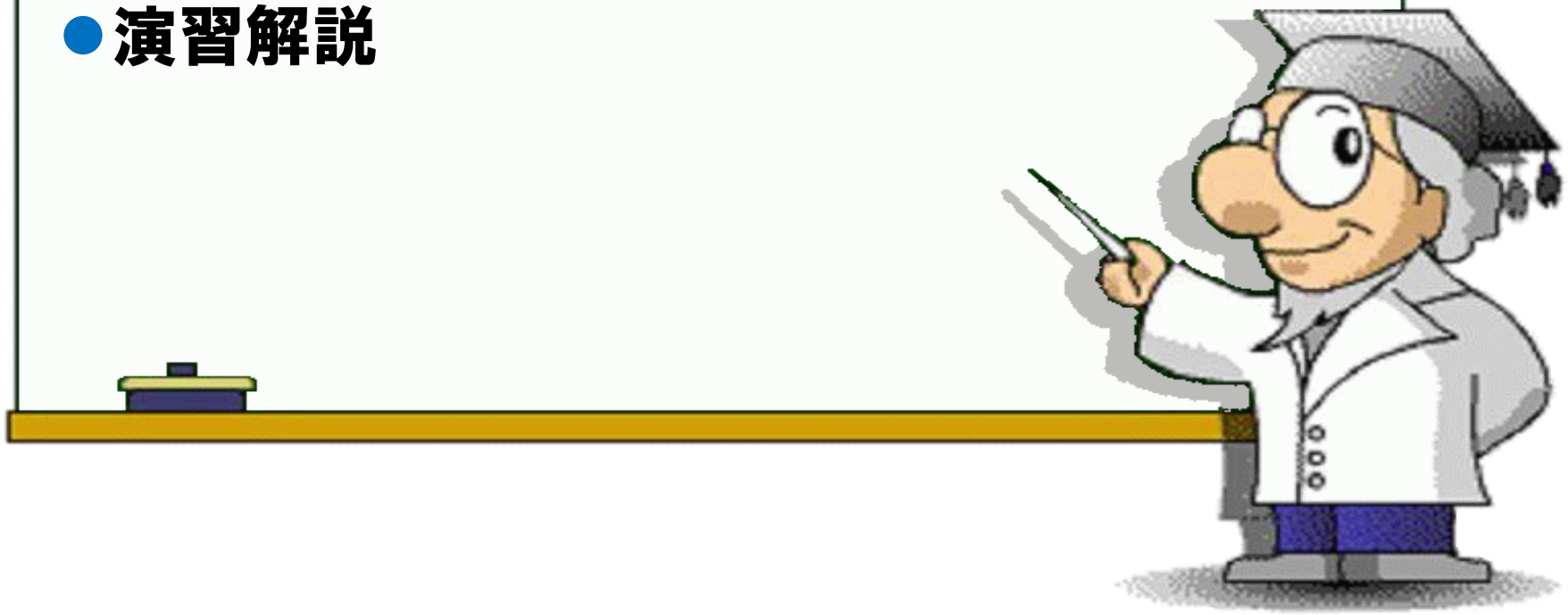
AppGoatを利用した集合教育補助資料 -SQLインジェクション編-

独立行政法人情報処理推進機構（IPA）
セキュリティセンター

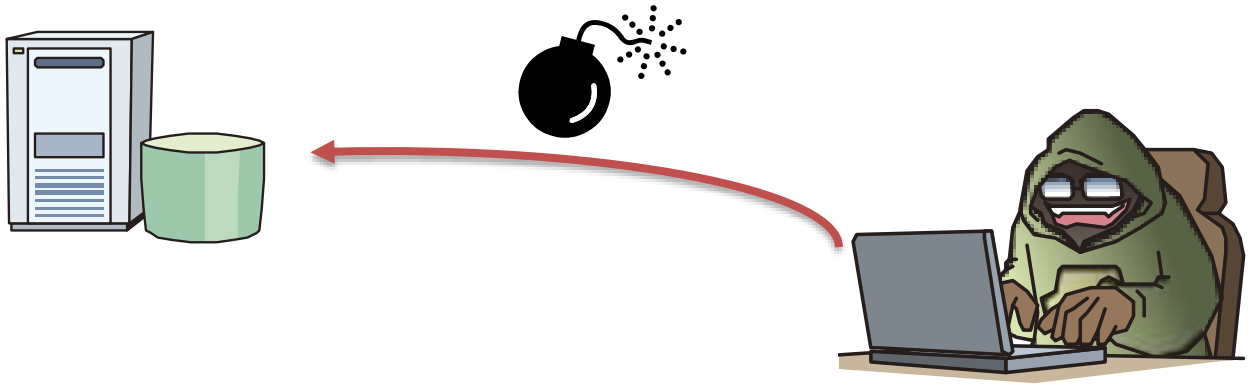
内容（演習１）

- 脆弱性の原理解説・基礎知識
- 脆弱性の発見方法
- 演習１：不正なログイン（文字列リテラル）
- 演習解説



SQLインジェクションとは？

- SQLとは、データベースを操作する為の問合せ言語
- SQL　Injection = SQLの注入
- すなわち、SQLインジェクションとは、外部から意図しないSQLを注入し、不正にデータベースを操作する攻撃
- ウェブアプリケーションで、最も狙われ易い脆弱性の一つ



SQLの基礎知識1－1

IPA

- SQLの基本形
  - 特定のテーブルから情報を取り出す場合

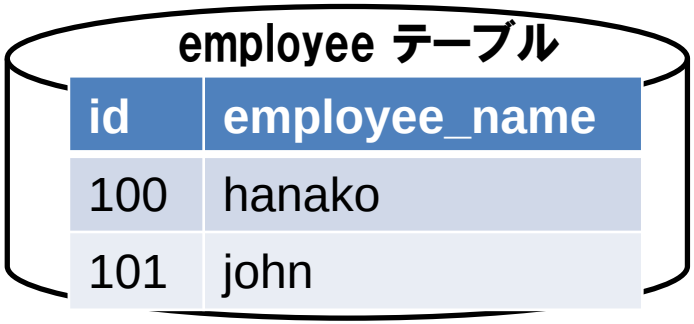
```
SELECT [A] FROM [B] WHERE [C]
```

[B]（テーブル）から［C］の条件で［A］（列の値）を取り出す

例

```
SELECT id FROM employee WHERE employee_name = 'john'
```

employee テーブルから
employee_name = 'john' となる条件で
id 列の値を取り出す



101

SQLの基礎知識1-2

IPA

リテラル（定数）

SQL文中で使用される定数をリテラルと呼び、文字列型、数値型、日付型などがある

文字列型

```
SELECT id FROM employee WHERE employee_name = 'john'
```

数値型

```
SELECT name FROM employee WHERE age >= 25
```

日付型

```
hire_date >= '2014/12/01'
```

クォート（'）

SQLにおいて、データの区切り文字として使われる特殊な意味を持つ記号。
シングルクォートを使い、文字列リテラルの範囲を示す。2つ（始まり・終わり）で一組。

```
SELECT id FROM employee WHERE employee_name = 'john'
```

'john Smith'

文字列リテラルの始まり　終わり　スペース

コメント（--）

SQLにおいて、-- 以降は、命令文ではなくコメントとして扱う。

```
SELECT id FROM employee -- WHERE employee_name = 'john'
```

SQL文として無効となる

SQLを使った認証判定の例(通常時)

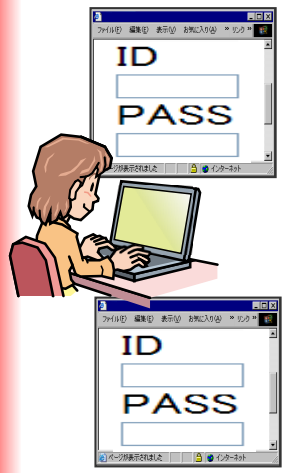
- ログインページでの認証判定では、SQLが使われることが多い。

認証判定SQL: `SELECT * FROM user WHERE id='ID' AND pass='PASS'`

user テーブルで id が ID で pass が PASS な行が

1つ以上あれば認証可、0なら認証不可

例）

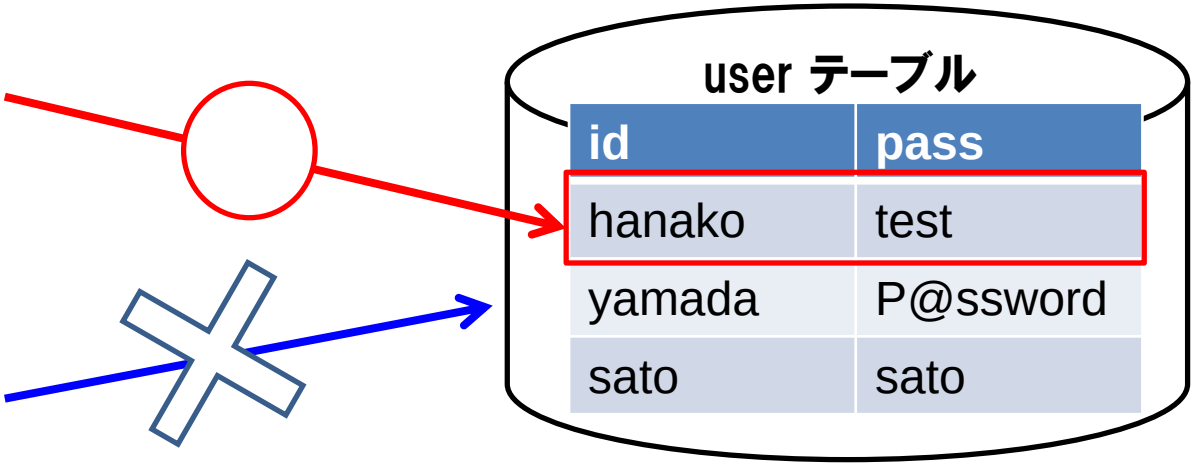


id=hanako

pass=test

id=hanako

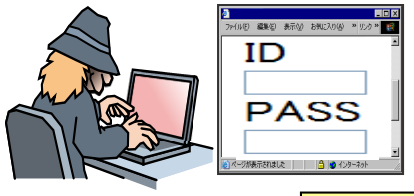
pass=123



SQLを使った認証判定の例(攻撃時)

- 攻撃者は‘(シングルクォート)と--(コメント)を使って攻撃

認証判定SQL: `SELECT * FROM user WHERE id= ‘ ID ’ AND pass= ‘ PASS ’`



id=hanako’ --

pass=123

コメント

`SELECT * FROM user WHERE id= ‘hanako’ --’ AND pass= ‘ 123’`

user テーブルで id が hanako で ~~pass が 123~~ な行が

1つ以上あれば認証可、0なら認証不可

user テーブル

id	pass
hanako	test
yamada	P@ssword

パスワードを知らなくてもログインされてしまう

SQLインジェクションを発見するために

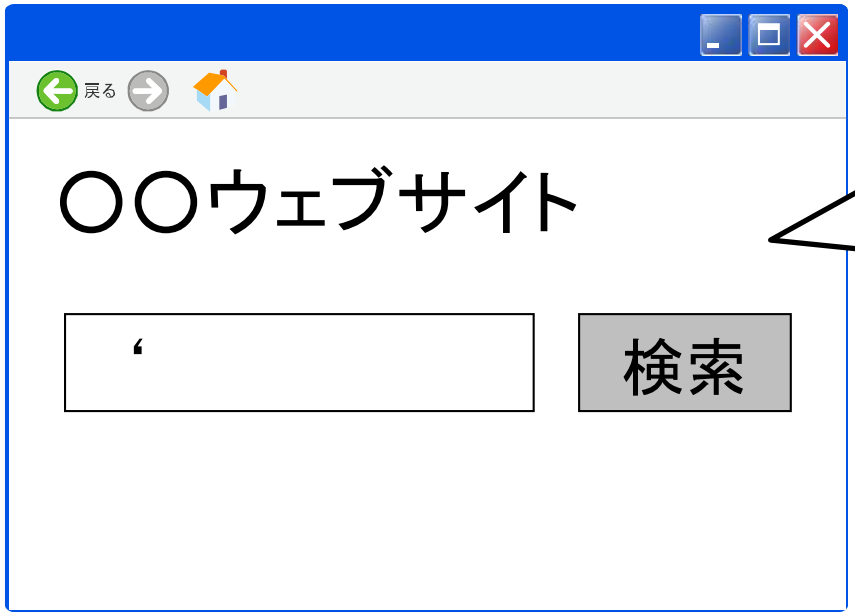
IPA
AppGoat
～突いてみますか？脆弱性！～

- STEP1：脆弱性が存在する箇所を発見する

Input フィールド

URLパラメータ

「‘」（シングルクォート1つ）

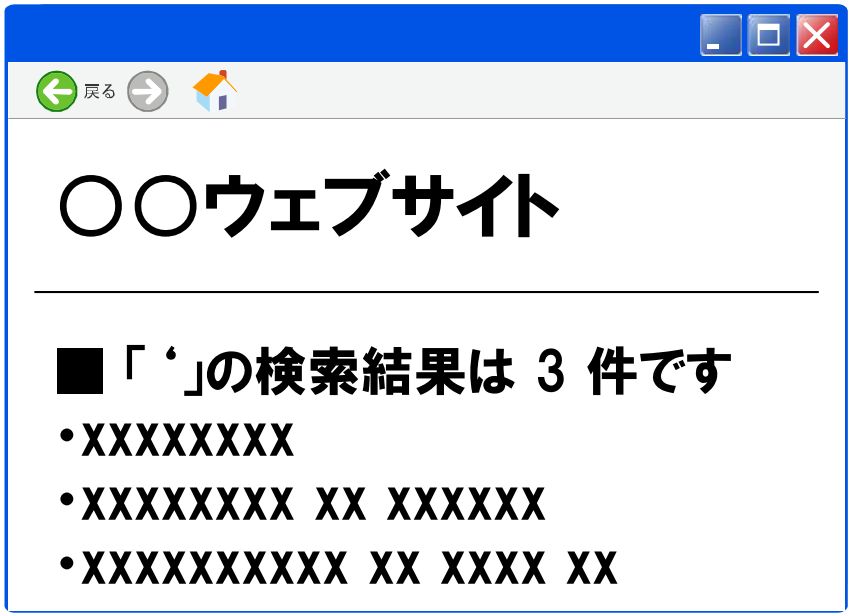


ウェブサイト内の入力項目に、検出パターンを入力して送信する。
（この例の場合、検索フォームに「’」と入力して検索ボタンを押す）

SQLインジェクションを発見するために

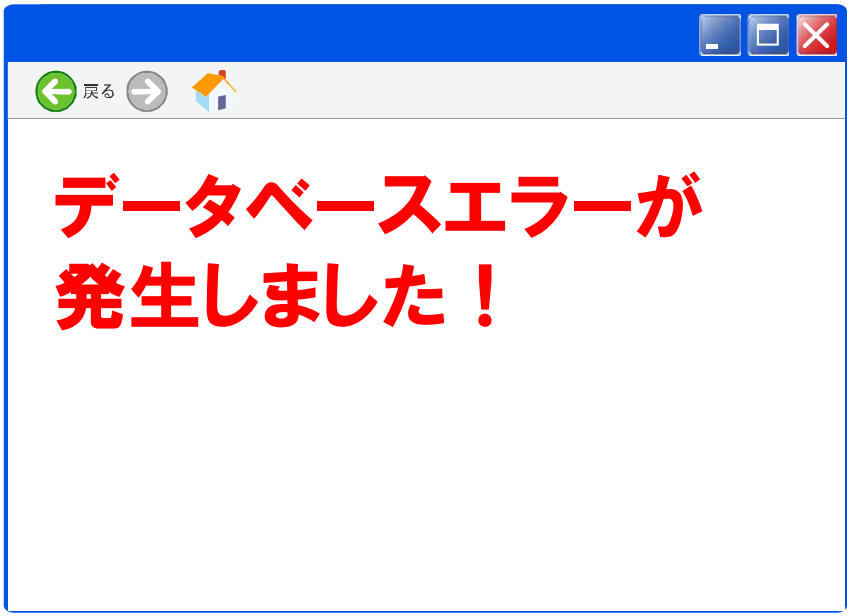
IPA
AppGoat
～突いてみますか？脆弱性！～

【正常】



機能が正常に稼働していれば、正常と判定。

【脆弱性あり】



データベース関連のエラーが発生していれば、脆弱性ありと判定。

SQLインジェクションを発見するために

IPA
AppGoat
～突いてみますか？脆弱性！～

なぜデータベースエラーが起きるのか

データベースエラー
が発生しました！

```
SELECT * FROM kensaku WHERE input = ' keyword ' ;
```

keywordに「’」を入れると

```
SELECT * FROM kensaku WHERE input = '' ' ;
```

‘ ’ ‘
始まり 終わり 余り

[演習] AppGoatの準備

①AppGoatを起動します

②以下の遷移で演習画面に移動します

[演習] AppGoatを用いた疑似攻撃体験

IPA

AppGoat
～突いてみますか？脆弱性！～

- **演習テーマ：**

  「不正なログイン（文字列リテラル）」

- **ミッション：**

  パスワード認証回避に挑戦!

Congratulations!!演習の目標を達成しました。

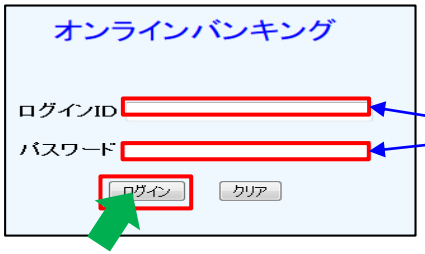
[演習] 演習の進め方

IPA

AppGoat ～突いてみますか？脆弱性！～

Step1：脆弱となる箇所を特定する

・ログイン画面の入力フィールドに様々な値を入力して、動作を確認する。



検査文字列の入力

Step2：脆弱性を突く攻撃を考える

・「前提条件」を参考に、認証を回避するようなSQL文を考える。

ユーザ認証処理では、次のSQL文が使用されています。

```
SELECT * FROM user WHERE id = 'ログインID' AND password = 'パスワード';
```

Step3:認証を回避してログインしてみる。

ログインID「yamada」でログインしてみましょう

演習

IPA

演習はじめてください。

※演習が終わったら次のページで解説を行います。

[Step1] ログイン画面の入力フィールドに様々な値を入力する

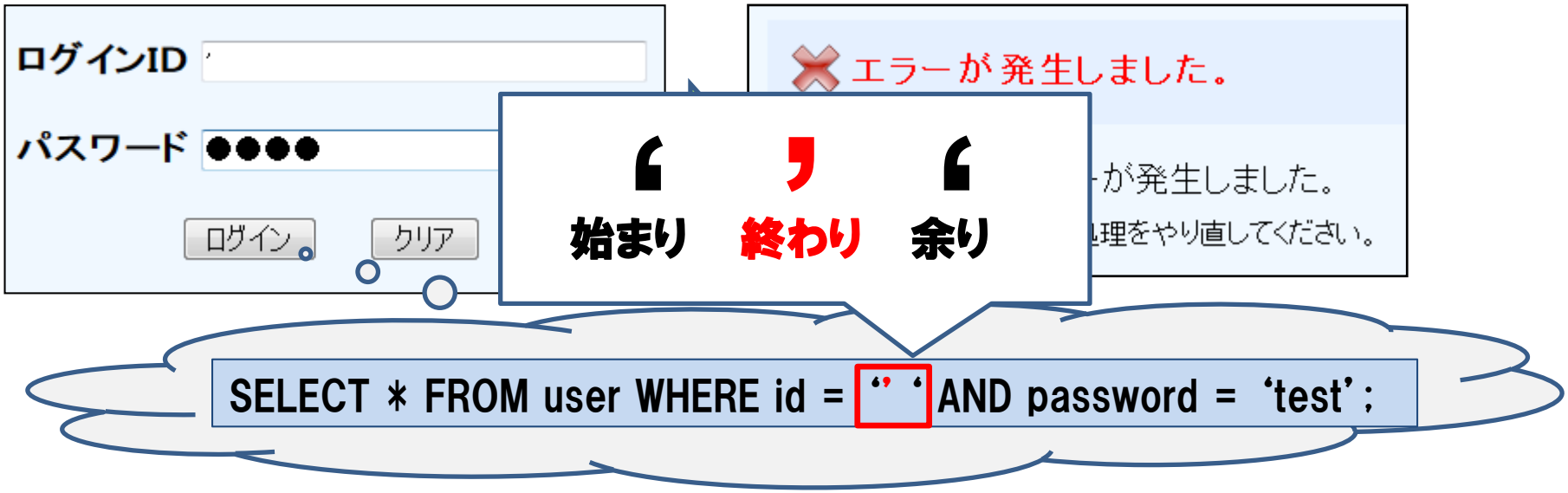
IPA AppGoat ～突いてみますか？脆弱性！～

演習の手順

SQL文のエラーを起こす値を入力して、脆弱な箇所を発見してみましょう

- 入力フィールドにシングルクォート「‘」を入力し、ログインしてみましょう。

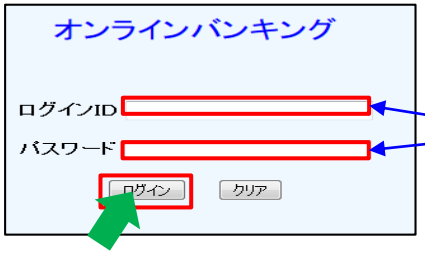
- その結果、下記のようなSQL文が作成され、構文エラーが起きたことが推測できます。



[演習] 演習の進め方

IPA AppGoat ～突いてみますか？脆弱性！～

- Step1：脆弱となる箇所を特定する
  - ・ログイン画面の入力フィールドに様々な値を入力して、動作を確認する。



検査文字列の入力

- Step2：脆弱性を突く攻撃を考える
  - ・「前提条件」を参考に、認証を回避するようなSQL文を考える。

ユーザ認証処理では、次のSQL文が使用されています。

```
SELECT * FROM user WHERE id = 'ログインID' AND password = 'パスワード';
```

- Step3:認証を回避してログインしてみる。

ログインID「yamada」でログインしてみましょう

[Step2] 
認証を回避するようなSQL文を考える

IPA AppGoat ～突いてみますか？脆弱性！～

演習の手順

SQL文のパスワード認証部分の条件式を書き換えましょう

```
SELECT * FROM user WHERE id = 'ログインID' AND password = 'パスワード';
```

- **論理演算子(OR)と(')を使って条件式を書き換える**

```
SELECT * FROM user WHERE id = 'yamada' AND password = 'A' OR 'A' = 'A'--;
```

常に正しい条件

- **[参考] (')と コメント(--)を使って条件式を書き換える**

id= yamada' --　　pass=a

```
SELECT * FROM user WHERE id = 'yamada' -- ' AND password = 'a';
```

SQL文として無効となる

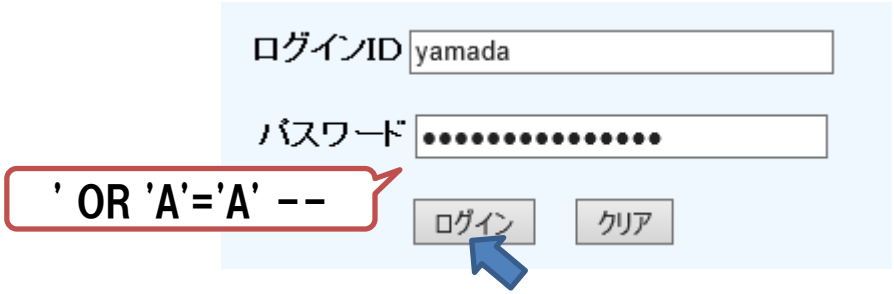
[Step3]  
攻撃により、手順2のSQL文を発行させる

IPA AppGoat ～突いてみますか？脆弱性！～

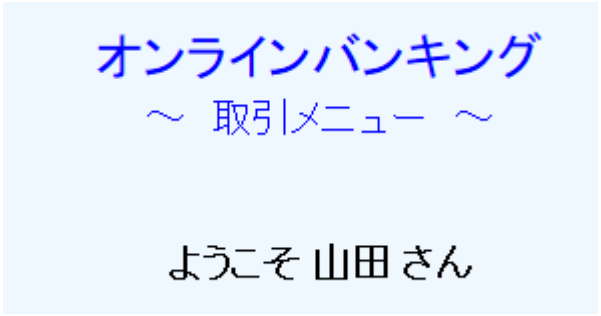
演習の手順

ログインID「yamada」で認証を回避してログインしてみましょう

1. **攻撃者**がオンラインバンキングのログインフォームのID欄に「yamada」を入力、パスワード欄に「' OR 'A'='A' --」を入力しログインを試みます。

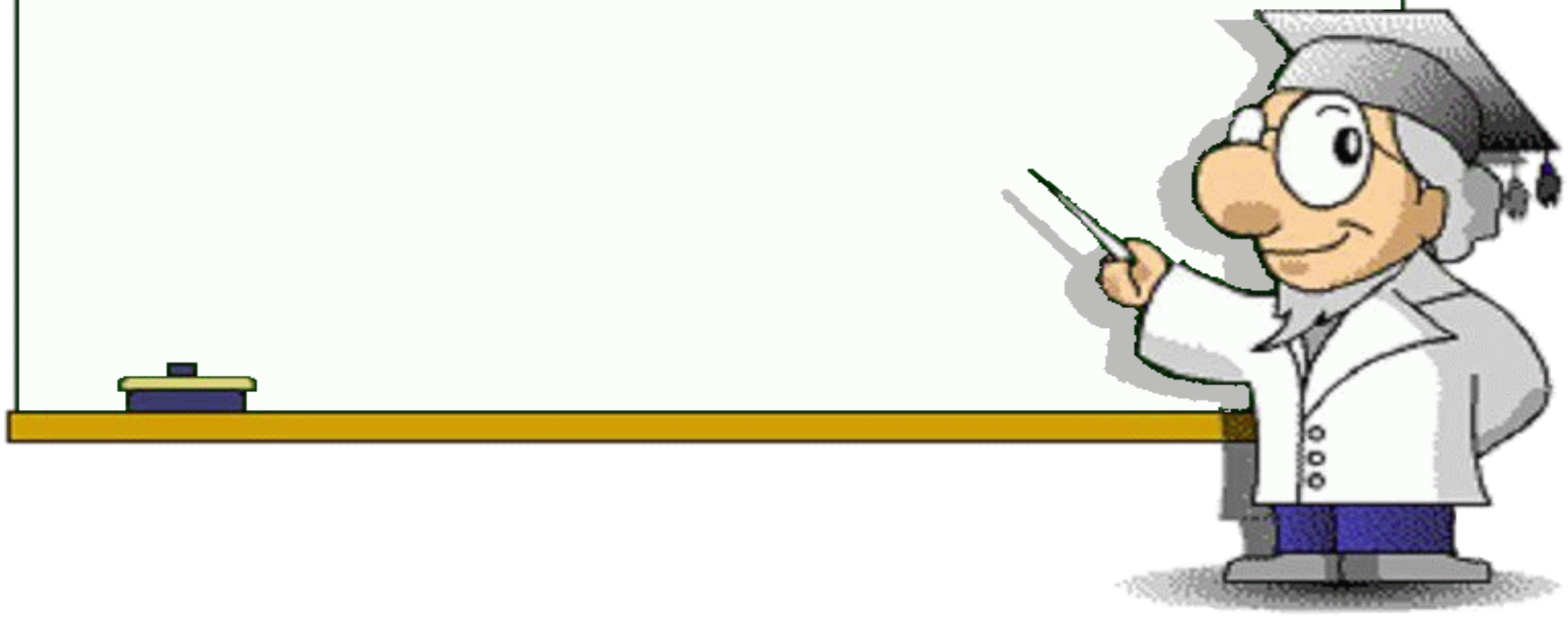


2. その結果、**攻撃者**が山田さんとしてログインできてしまいます。



内容（演習2）

- 脆弱性の原理解説・基礎知識
- 演習2：情報漏えい（数値リテラル）
- 演習解説



SQLの基礎知識2

IPA

論理演算子（AND、OR）

SQL文中のWHERE句で複数の条件を指定する演算子

```
SELECT id FROM employee WHERE age >= 25 AND unit = ‘soumu’
```

年齢が25歳以上”かつ”所属が総務

```
SELECT id FROM employee WHERE age >= 25 AND unit = ‘soumu’ OR unit= ‘keiri’
```

年齢が25歳以上 ”かつ” 所属が総務 ”または” 所属が経理

先にANDを判定し、ORが次になる

Where 句の特殊な使われ方

常に正になる条件式：条件式が同じリテラル値同士で成立している条件式

```
SELECT id FROM employee WHERE ‘john’ = ‘john’
```

常に正になるため、条件式の意味がない

=

```
SELECT id FROM employee
```

SQLを使ったデータ抽出(通常時)

- 特定ユーザの情報を取り出すページではSQLが使われることが多い。(例:成績確認、残高照会　等)

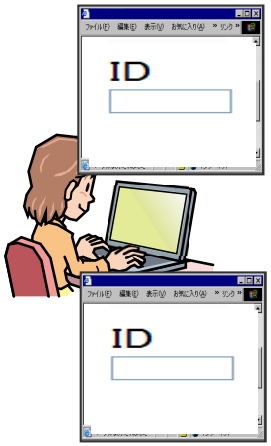
認証判定SQL: `SELECT * FROM seiseki WHERE sid=` [ID]

seiseki テーブルで sid が [ID] な行を出力

例)

seiseki テーブル

sid	kamoku	tokuten
100	English	60
100	Math	80
200	Math	40
300	Math	100



sid=100

sid	kamoku	tokuten
100	English	60
100	Math	80

sid=400

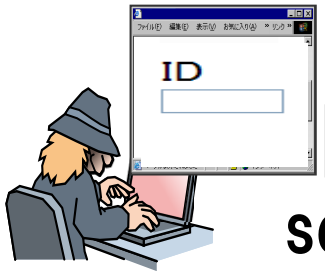
出力なし(0件)

SQLを使ったデータ抽出(攻撃時)

IPA

- 攻撃者は OR 演算子を使って攻撃をしてくる

認証判定SQL: SELECT * FROM seiseki WHERE sid= ID



sid=400 OR 1=1

SELECT * FROM seiseki WHERE sid= 400 OR 1=1

seiseki テーブルで sid が 400 または 1=1 な行を出力

400 → ヒット 0件

1=1 → 常に正しい(ヒット 全件)

SELECT * FROM seiseki

seiseki テーブル

sid	kamoku	tokuten
100	English	60
100	Math	80
200	Math	40
300	Math	100

sid	kamoku	tokuten
100	English	60
100	Math	80
200	Math	40
300	Math	100

sid の値に関わらず、すべての情報が出力される

[演習] AppGoatの準備

①以下の遷移で演習画面に移動します

[演習] AppGoatを用いた疑似攻撃体験

IPA
AppGoat
～突いてみますか？脆弱性！～

- **演習テーマ：**

  「情報漏えい（数値リテラル）」

- **ミッション：**

  全ユーザの口座残高を閲覧

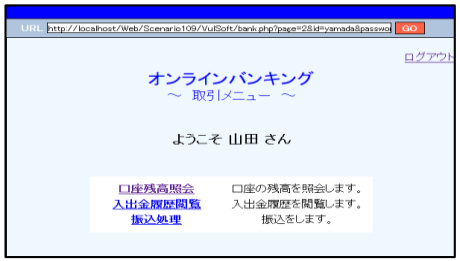
Congratulations!!演習の目標を達成しました。

[演習] 演習の進め方

IPA

AppGoat ～突いてみますか？脆弱性！～

- **Step1：脆弱となる箇所を特定する**
  - ・SQL文のエラーを起こす値を入力して、脆弱な箇所を発見してみましょう
  - －－



- **Step2：前提条件を参考に脆弱性を突く攻撃を考える**
  - ・論理演算子 OR を使用してWHERE句を常に正しくなる条件にしましょう

```
SELECT * FROM account WHERE id = 'ログインID' AND account_id = 口座番号;
```

- **Step3:SQL文を発行し、他人の口座残高情報を取得する**
  - ・ログインユーザ以外の口座残高情報が取得できるか確認しましょう。

演習

IPA

演習はじめてください。

※演習が終わったら次のページで解説を行います。

[Step1]  URLパラメータに様々な値を入力して、動作を確認する

IPA AppGoat ～突いてみますか？脆弱性！～

演習の手順

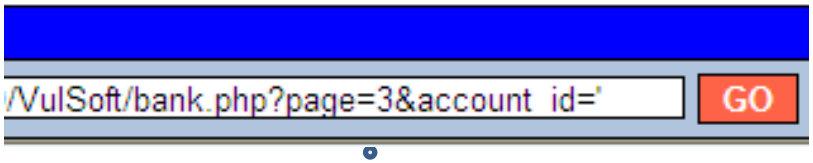
SQL文のエラーを起こす値を入力して、脆弱な箇所を発見してみましょう

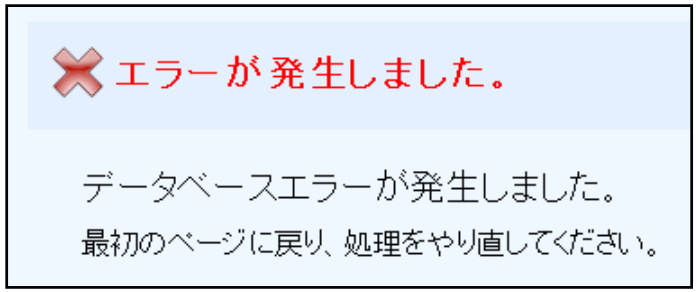
- URLパラメータ(account_id)に ‘ や test という文字列を入力し、アクセスしてみましょう。

① account_id : ‘ → エラーが発生
② account_id :test → エラーが発生

account_id にSQLインジェクションの脆弱性が存在する可能性

- その結果、下記のようなSQL文が作成され、構文エラーが発生することが推測できます。





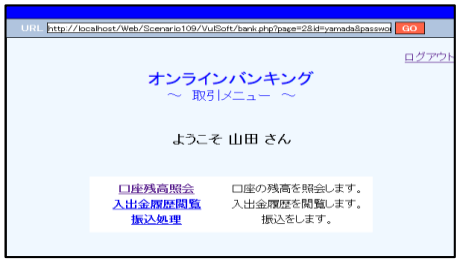

```
SELECT * FROM account  WHERE id = ‘sato’ AND account_id = ‘;
```

```
SELECT * FROM account  WHERE id = ‘sato’ AND account_id = test ;
```

[演習] 演習の進め方

IPA AppGoat ～突いてみますか？脆弱性！～

- Step1：脆弱となる箇所を特定する
  - ・SQL文のエラーを起こす値を入力して、脆弱な箇所を発見してみましょう



- Step2：前提条件を参考に脆弱性を突く攻撃を考える
  - ・論理演算子 OR を使用してWHERE句を常に正しくなる条件にしましょう

```
SELECT * FROM account WHERE id = 'ログインID' AND account_id = 口座番号;
```

- Step3:SQL文を発行し、他人の口座残高情報を取得する
  - ・すべてのログインユーザの口座残高情報が取得できるか確認しましょう。

[Step2] 
他人の口座残高情報を参照するSQL文を考える

IPA AppGoat ～突いてみますか？脆弱性！～

演習の手順

論理演算子 OR を使用してWHERE句が常に正しくなる条件にしましょう

```
SELECT * FROM account WHERE id = 'ログインID' AND account_id = 口座番号;
```

- **論理演算子(OR)を使って条件式を書き換える**

```
SELECT * FROM account WHERE id = 'ログインID' AND account_id = 99 OR 1=1
```

SELECT * FROM account WHERE id = 'sato' AND account_id = 99 OR 1=1;

ANDが先に評価されるため、「idがsatoかつaccount_idが99」または「1=1」という意味になる。つまり、条件は常に正しい。

SELECT * FROM account ;

[手順3] 脆弱性を突いてみる

IPA

AppGoat ～突いてみますか？脆弱性！～

演習の手順

ログインユーザ以外の口座残高情報が取得できるか確認しましょう

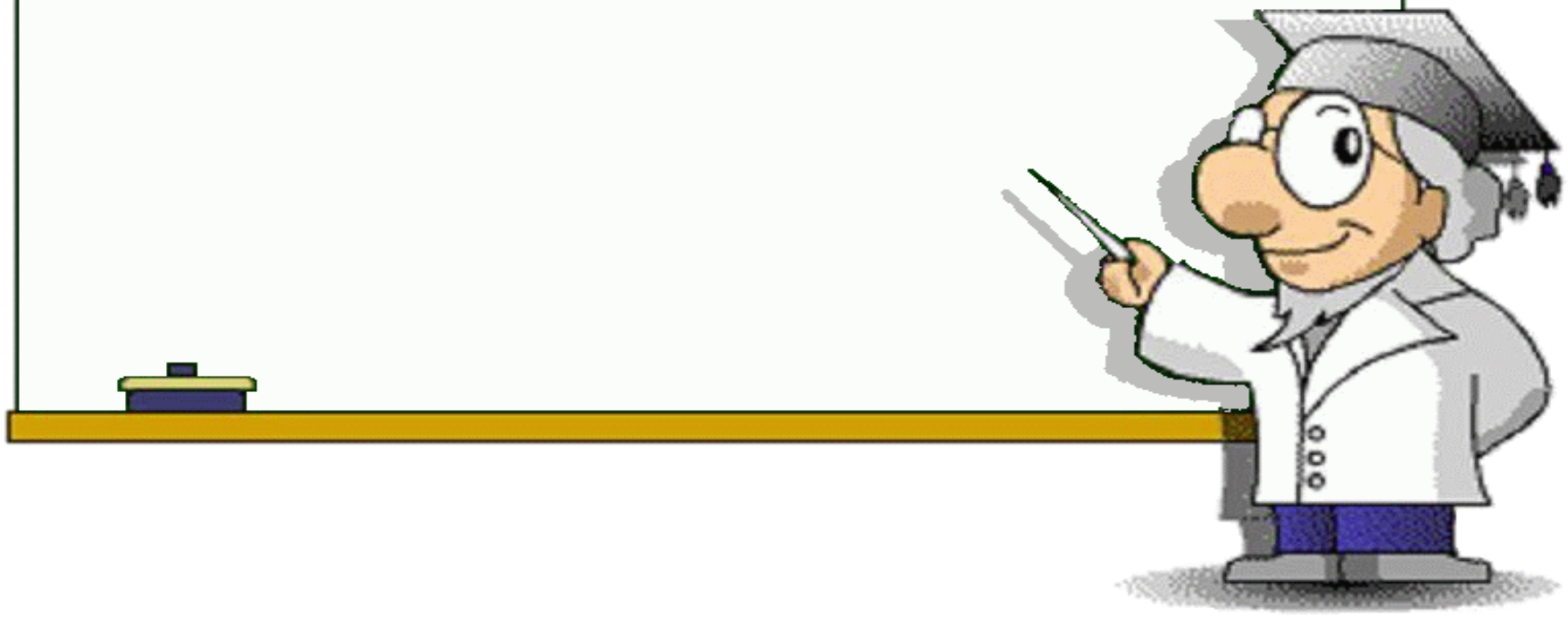
1. 攻撃者がオンラインバンキングの口座照会ページのURLのaccount_idパラメータに「99 OR 1=1」の文字列を入力し、アクセスを試みます。

URL rsonal/Web/Scenario1222/VulSoft/bank.php?page=3&account_id=99 OR 1=1 GO

2. その結果、攻撃者が全ユーザの口座情報を盗み見ることができることを確認します。

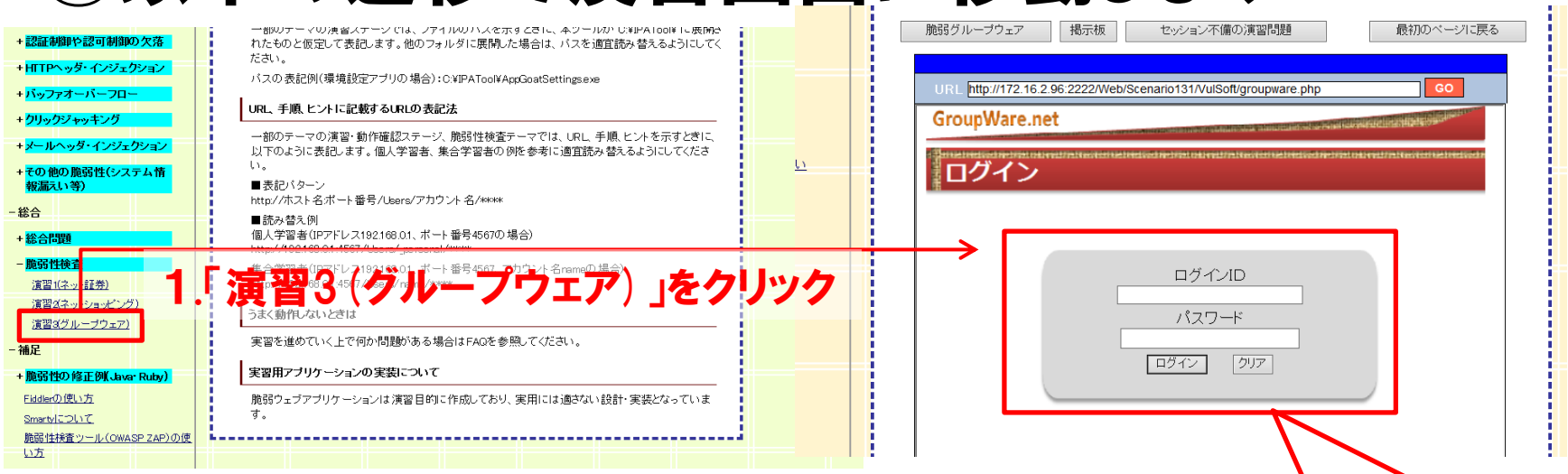
内容（演習3）

- 演習3：演習3（グループウェア）
- 演習解説
- 対策方法

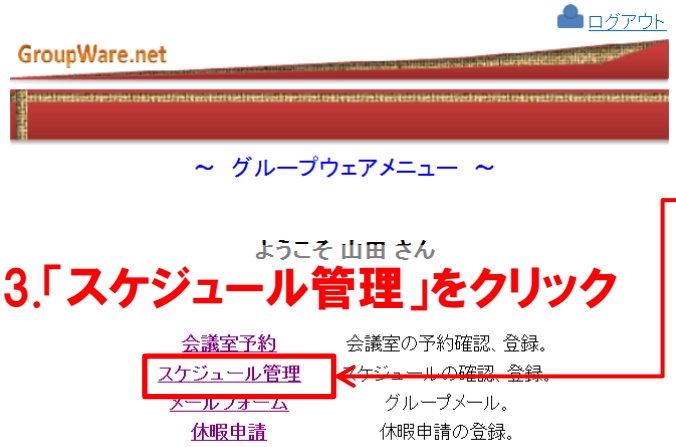


[演習] AppGoatの準備

①以下の遷移で演習画面に移動します

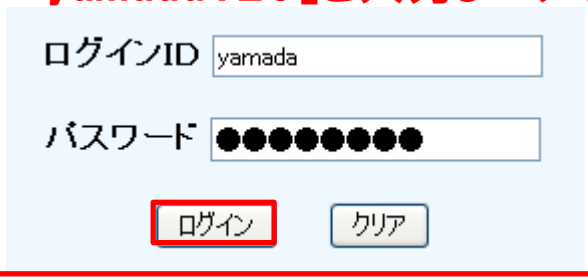


1.「演習3（グループウェア）」をクリック



3.「スケジュール管理」をクリック

2.IDに「yamada」、パスワードに「yamada123」と入力しログイン



[演習] AppGoatを用いた疑似攻撃体験

IPA
AppGoat
～突いてみますか？脆弱性！～

- **演習テーマ：**

  「演習3（グループウェア）」

- **ミッション：**

  全ユーザのスケジュールを閲覧

Congratulations!!演習の目標を達成しました。

[演習] 演習の進め方

IPA

AppGoat
～突いてみますか？脆弱性！～

- **Step1：脆弱となる箇所を特定する**
  - ・SQL文のエラーを起こす値を入力して、脆弱な箇所を発見してみましょう

- **Step2：前提条件を参考に脆弱性を突く攻撃を考える**

SELECT * FROM schedule WHERE id = ‘ログインID’ AND date = ’日付’

- **Step3:SQL文を発行し、他人のスケジュールを取得する**
  - ・ログインユーザ以外のスケジュールが取得できるか確認しましょう。

演習

IPA

演習はじめてください。

※演習が終わったら次のページで解説を行います。

[Step1] URLパラメータに様々な値を入力して、動作を確認する

IPA AppGoat ～突いてみますか？脆弱性！～

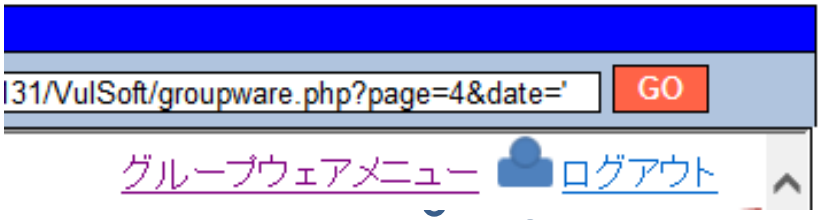
演習の手順

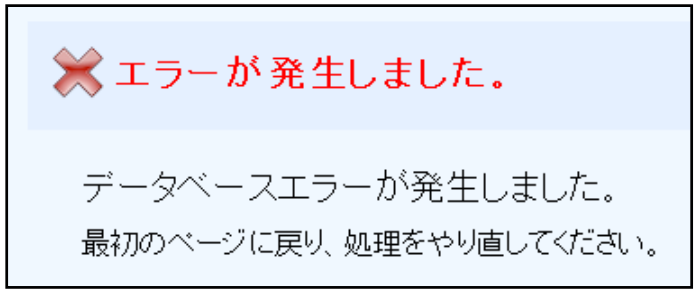
SQL文のエラーを起こす値を入力して、脆弱な箇所を発見してみましょう

- URLパラメータ(date)に ‘ という文字列を入力し、アクセスしてみましょう。

  date = ‘ → エラーが発生 ⇒ date にSQLインジェクションの脆弱性が存在する可能性

- その結果、下記のようなSQL文が作成され、構文エラーが発生することが推測できます。






```
SELECT * FROM schedule WHERE id = ‘yamada’ AND date = ‘’ ‘ ;
```

[Step2] 
他人のスケジュールを参照するSQL文を考える

IPA
AppGoat
～突いてみますか？脆弱性！～

演習の手順

論理演算子 OR を使用してWHERE句が常に正しくなる条件にしましょう

```
SELECT * FROM schedule WHERE id = ‘ログインID’ AND date = ‘日付’
```

- **論理演算子(OR)を使って条件式を書き換える**

```
SELECT * FROM schedule WHERE id = ‘ログインID’ AND date = ‘2017/09/03’ OR ‘a’ = ‘a’
```

ポイント！！
前後にあるシングルクォートでエラーにならないようにする

```
SELECT * FROM schedule WHERE id = ‘yamada’ AND date = ‘2017/09/03’ OR ‘a’ = ‘a’;
```

```
SELECT * FROM schedule ;
```

[手順3] 脆弱性を突いてみる

IPA

AppGoat ～突いてみますか？脆弱性！～

演習の手順

ログインユーザ以外のスケジュールが取得できるか確認しましょう

1. 攻撃者がグループウェアのスケジュールページのURLのdateパラメータに「2017/09/03’ OR ‘a’ = ‘a」の文字列を入力し、アクセスを試みます。

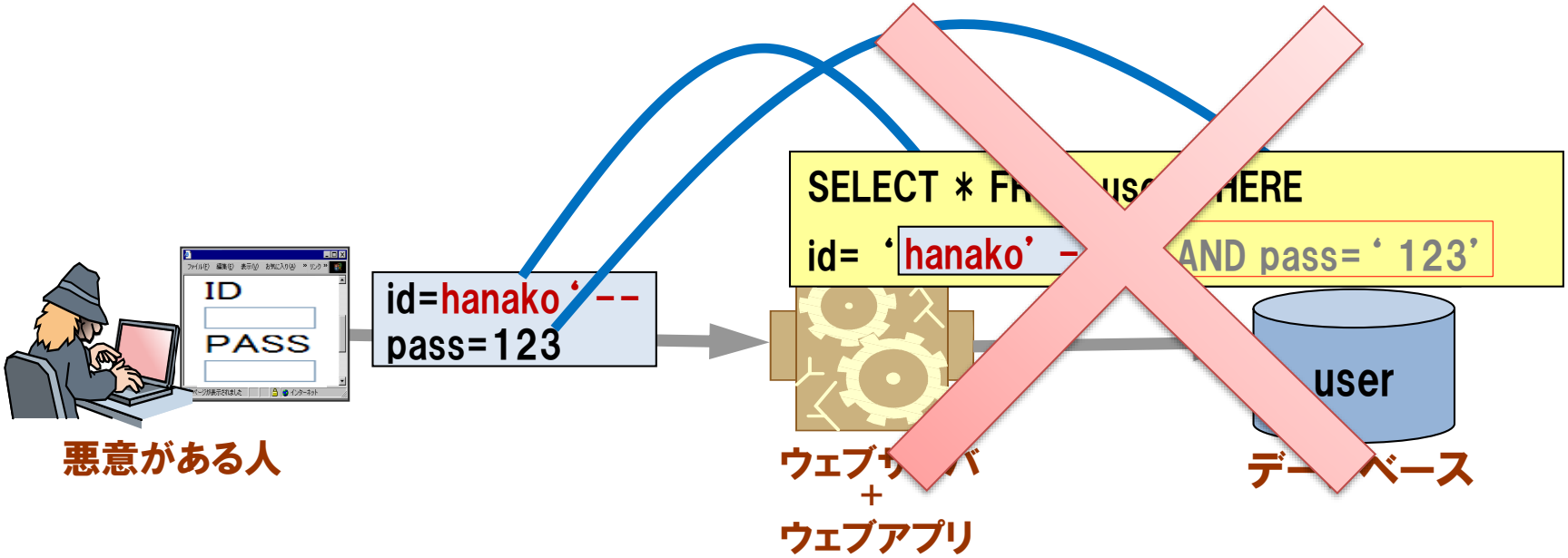
URL /eb/Scenario131/VulSoft/groupware.php?page=4&date=2017/09/03' OR 'a'='a GO

2. その結果、攻撃者が全ユーザのスケジュールを盗み見ることができることを確認します。

対策方法

SQLインジェクションを引起す実装

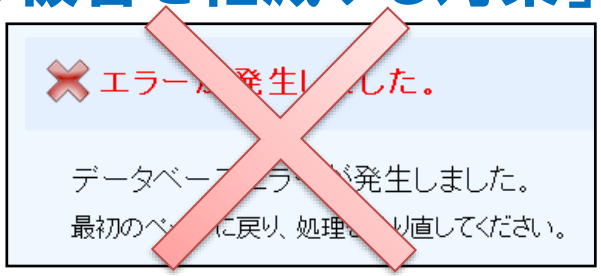
- SQL文の組立てに問題がある実装



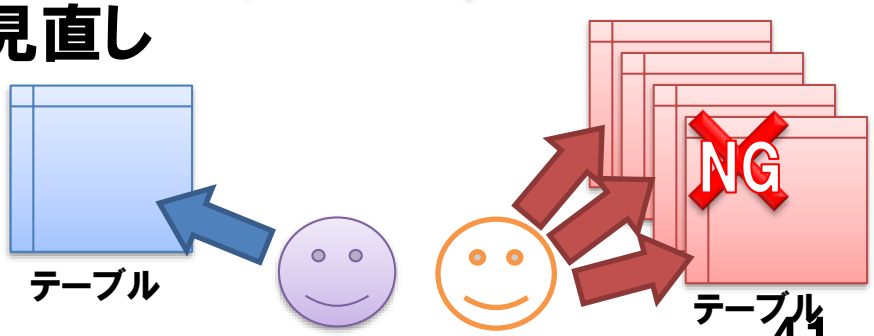
不正なリクエストによって、発行するSQL文の構文を書き換えられないように処理する

SQLインジェクション脆弱性の対策

- **根本的解決**「意図しないSQL文の発行を防止する実装」
  - SQL文の組立ては全てプレースホルダで実装する
  - エスケープ処理を利用する
    （対策の抜け漏れを起こしやすいので注意）
- **保険的対策**「攻撃を抑制、または被害を軽減する対策」
  - 詳細なエラーメッセージの抑止



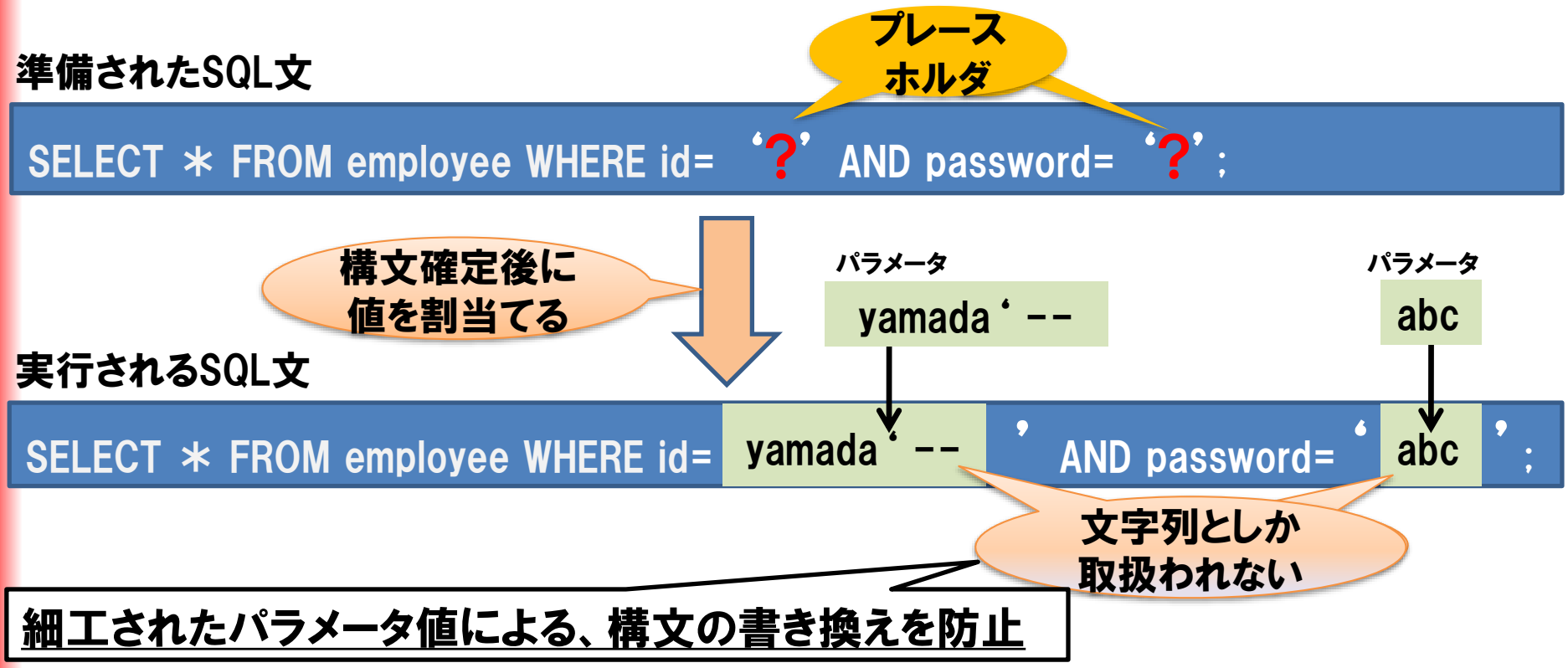
  - データベースアカウントの権限見直し



根本的解決
～プレースホルダを利用した対策例～

- **SQL文の組立てはプレースホルダで実装する**

  プレースホルダは、先にSQL文の構文を確定させ、後からパラメータの値を機械的な処理で割り当てる方式



以上で、
SQLインジェクションの解説は終了です。