DESIGN SECURE WEB APPLICATIONS

OWASP
The Open Web Application Security Project

ASHISH RAO
&
SIDDHARTH ANBALAHAN

About Ashish

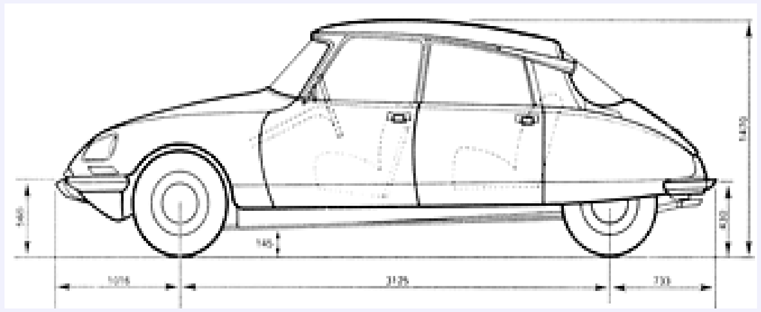
OWASP
The Open Web Application Security Project

- 4 years of IT Security Experience
- Security Consultant and Researcher – Application and Code Security Practice
- Expertise in performing Security Design reviews and Security Code Reviews
- Developed Code Review Checklists and Automation scripts for many platforms
- Conducted Trainings on Secure Development of Web and Mobile applications for different platforms

http://artechtalks.blogspot.in/

Index

What is a design?



Before

After

Design –
A plan or a diagram that translates ideas into models.

What is an application design?

OWASP
The Open Web Application Security Project

Application Design:

- A structure that determines execution flow
- Determines how different components interact with each other

- There are many design frameworks present today
- Most of such designs are based on “MVC”

What is MVC?

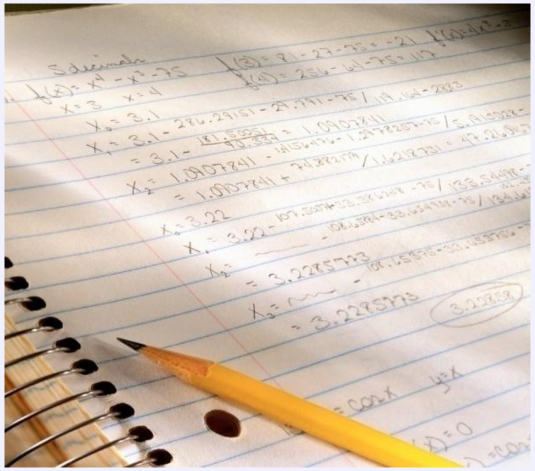
OWASP
The Open Web Application Security Project

John is a developer of an application and he wants to add a new feature that can let the admin user create new users in the system.

How should he code it?

Well, the very first question to ask is, how should he DESIGN it?

What is MVC?



Write the entire code in one file….

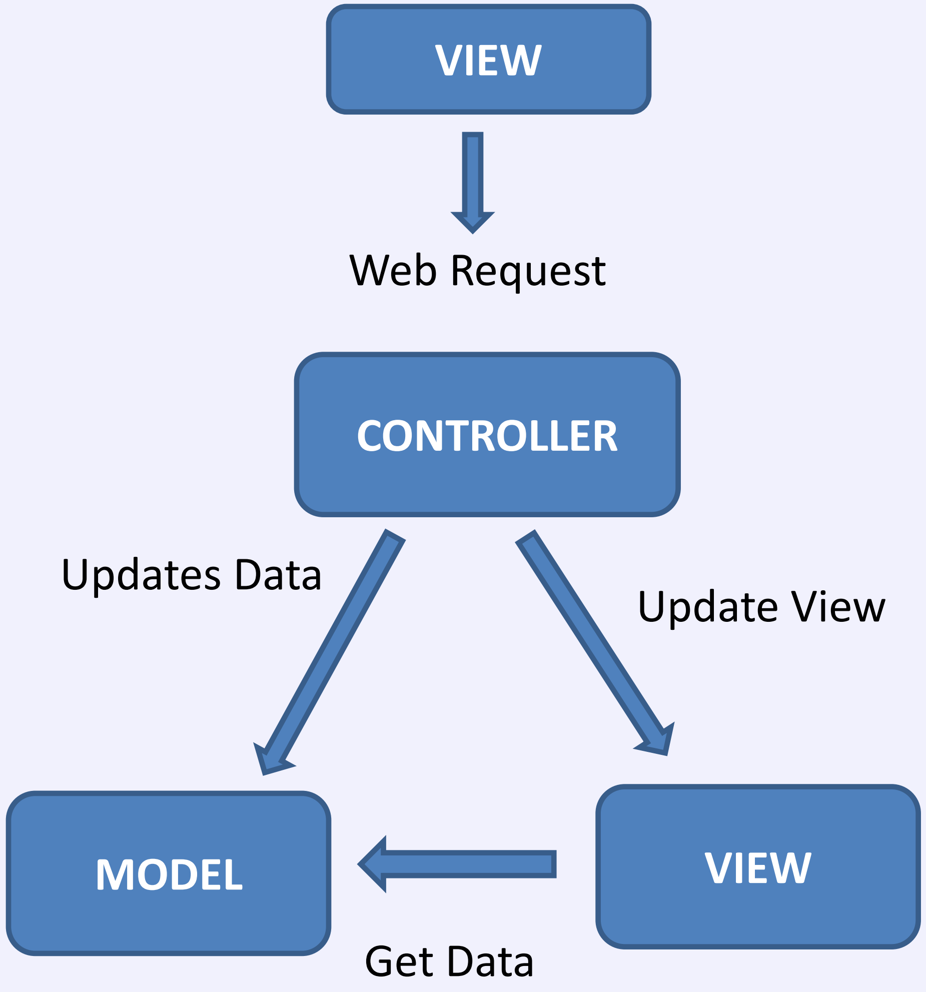
It’s a bad idea.

Design it well

What is MVC?

- Things to develop:
  - Form to add user $\longrightarrow$ View
  - A class to understand and process add user request $\longrightarrow$ Controller
  - A class to hold user data $\longrightarrow$ Model

What is MVC?


VIEW
Web Request
CONTROLLER
Updates Data
Update View
MODEL
VIEW
Get Data

Importance of a design

OWASP
The Open Web Application Security Project

- Segregation of code in logical components
- Makes code maintainable
- Easy to incorporate change
- *Easy to build security controls*

Should we review design?

OWASP
The Open Web Application Security Project

- Can something go wrong in a design?

Why NOT?

- Design reviews are very important
- A flaw in the design can break the entire model

Should we review design?

OWASP
The Open Web Application Security Project

- Insecure designs are big threat to the application
- Design flaws are:
  - Lesser known
  - Invisible
  - Hardly caught by scanners
  - Can lead to many security flaws in the applications

Vulnerable Areas in Design

OWASP
The Open Web Application Security Project

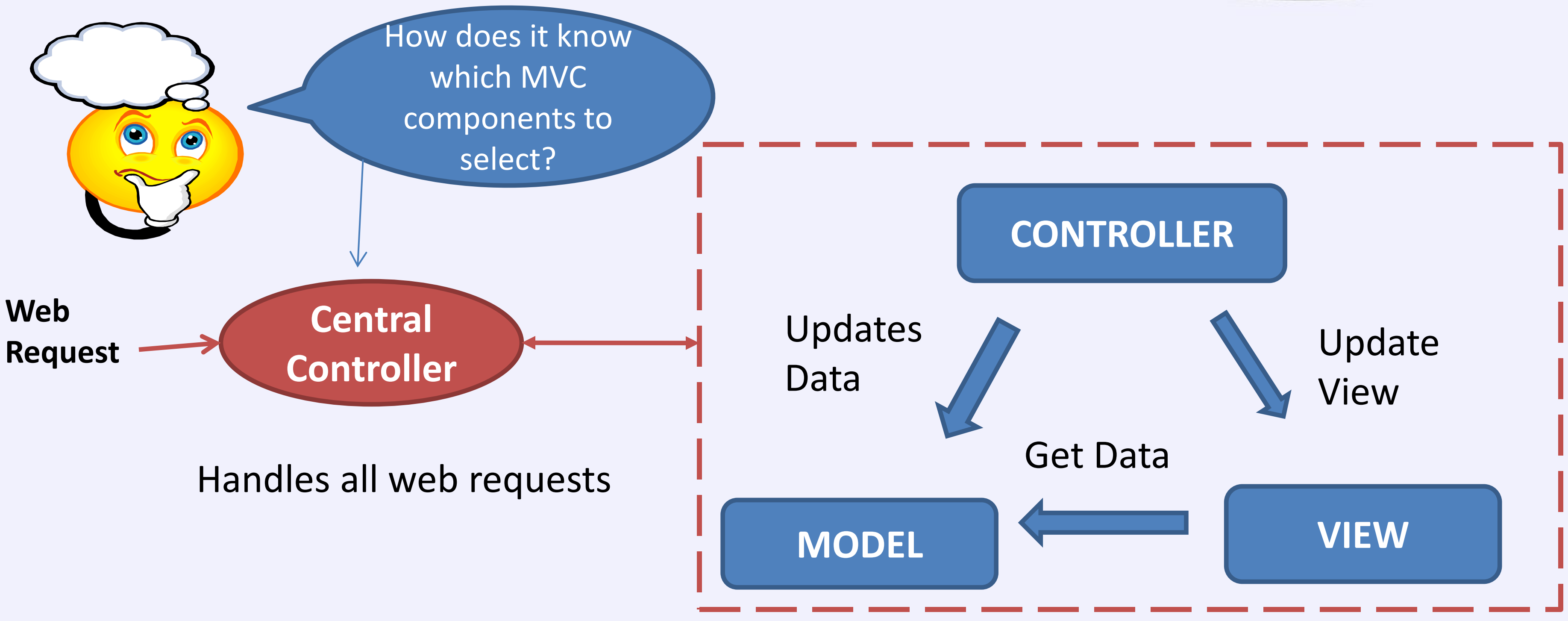
Things can wrong in:

- – Data Flow/Business Logic Invocation
- – Handling Inputs
- – Placement of Checks
- – Inter Application communication

OWASP
The Open Web Application Security Project

INSECURE BUSINESS LOGIC INVOCATION

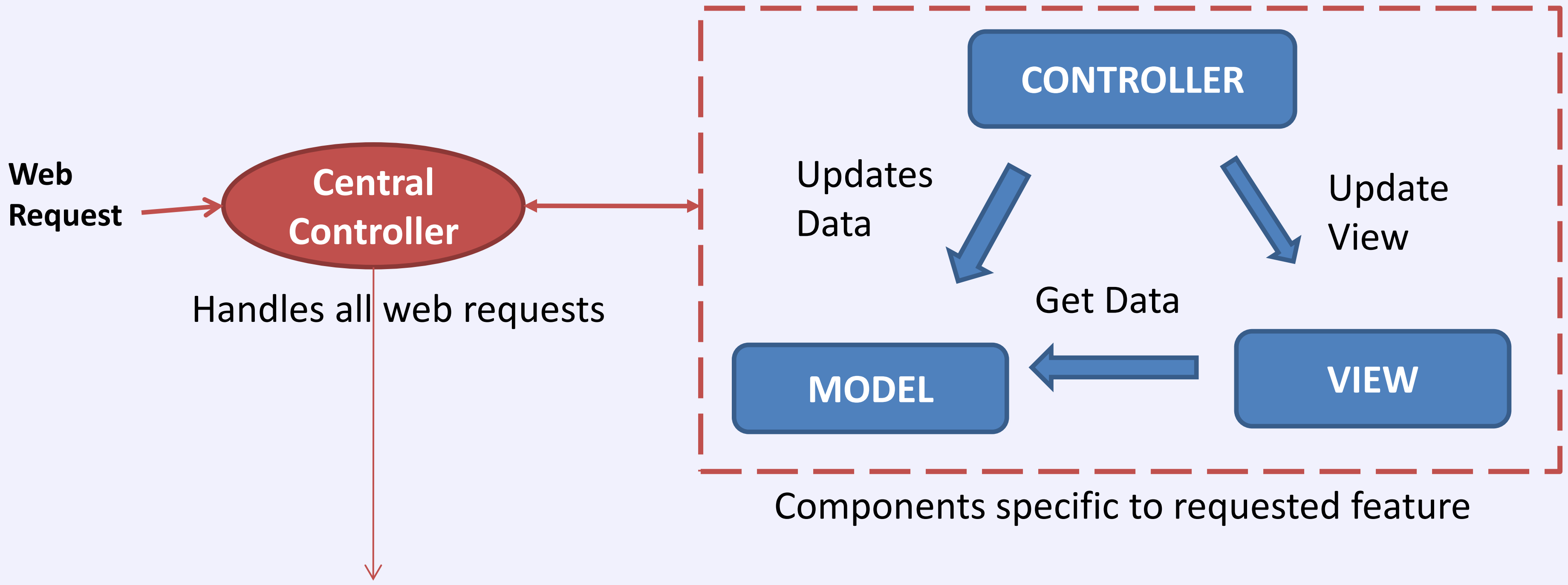
Business Logic Invocation



Handles all web requests

Components specific to requested feature

Business Logic Invocation



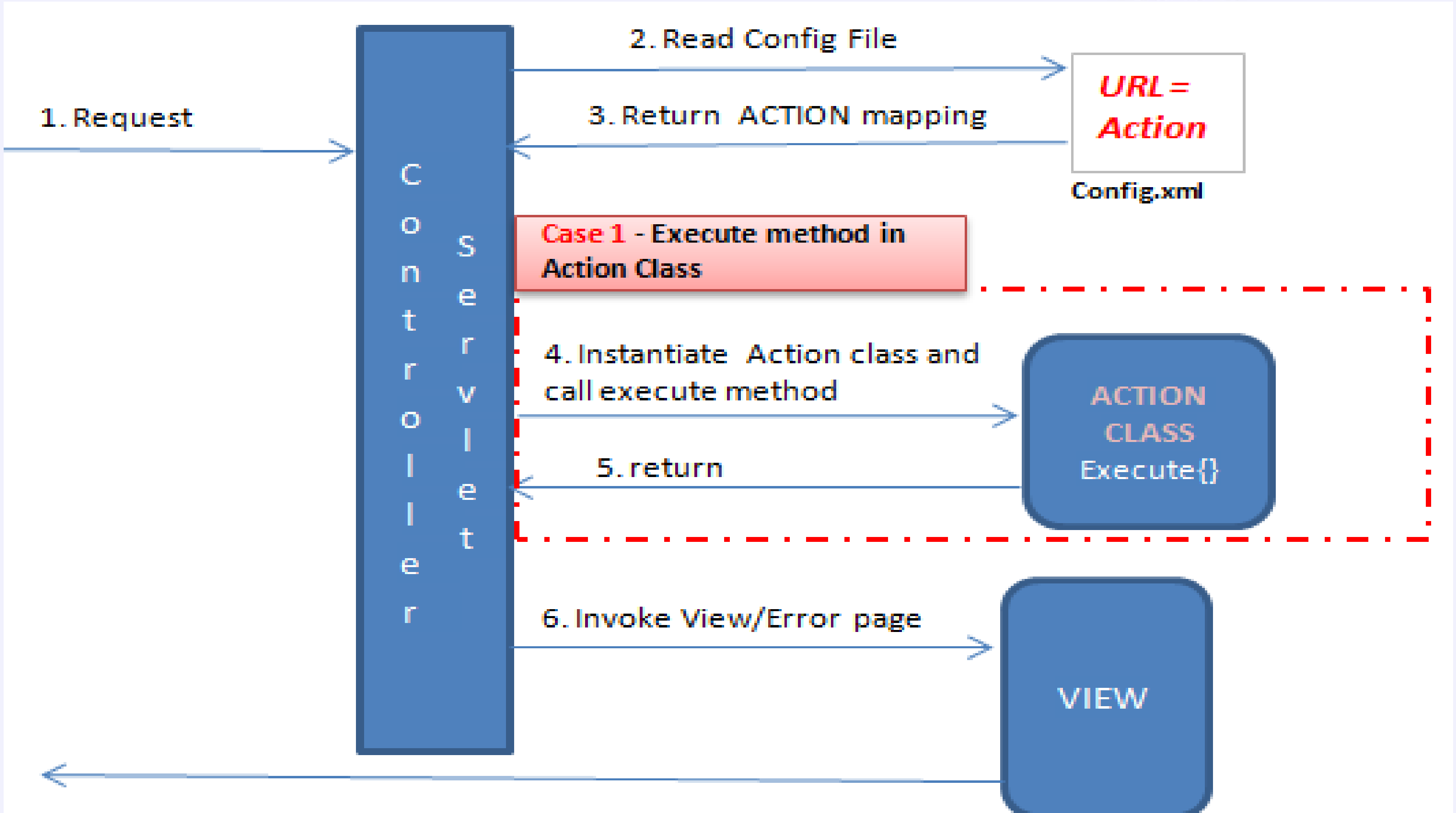
Identifies the business logic class/MVC components based on the request – **URL/Parameters**

Business Logic Invocations

- **Lets understand the design:**
  - The design uses user input to determine:
    - Business logic component
      - Fully qualified class names
      - Method name
      - View component

OWASP
The Open Web Application Security Project

Insecure Business Logic Invocation



Sample Design

Business Logic Invocation

OWASP
The Open Web Application Security Project

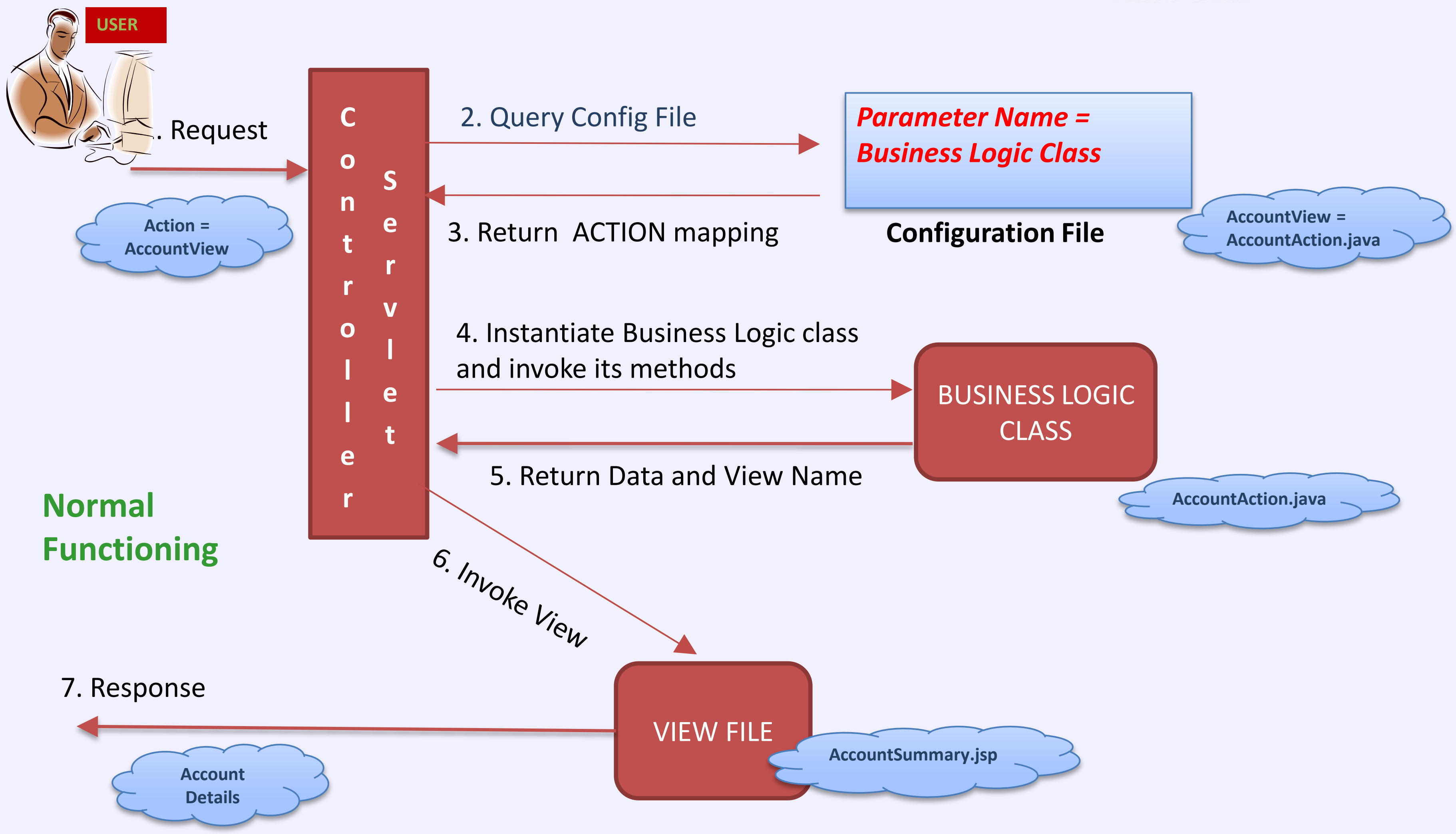
DEMO -

Code Walk through

Insecure Business Logic Invocation

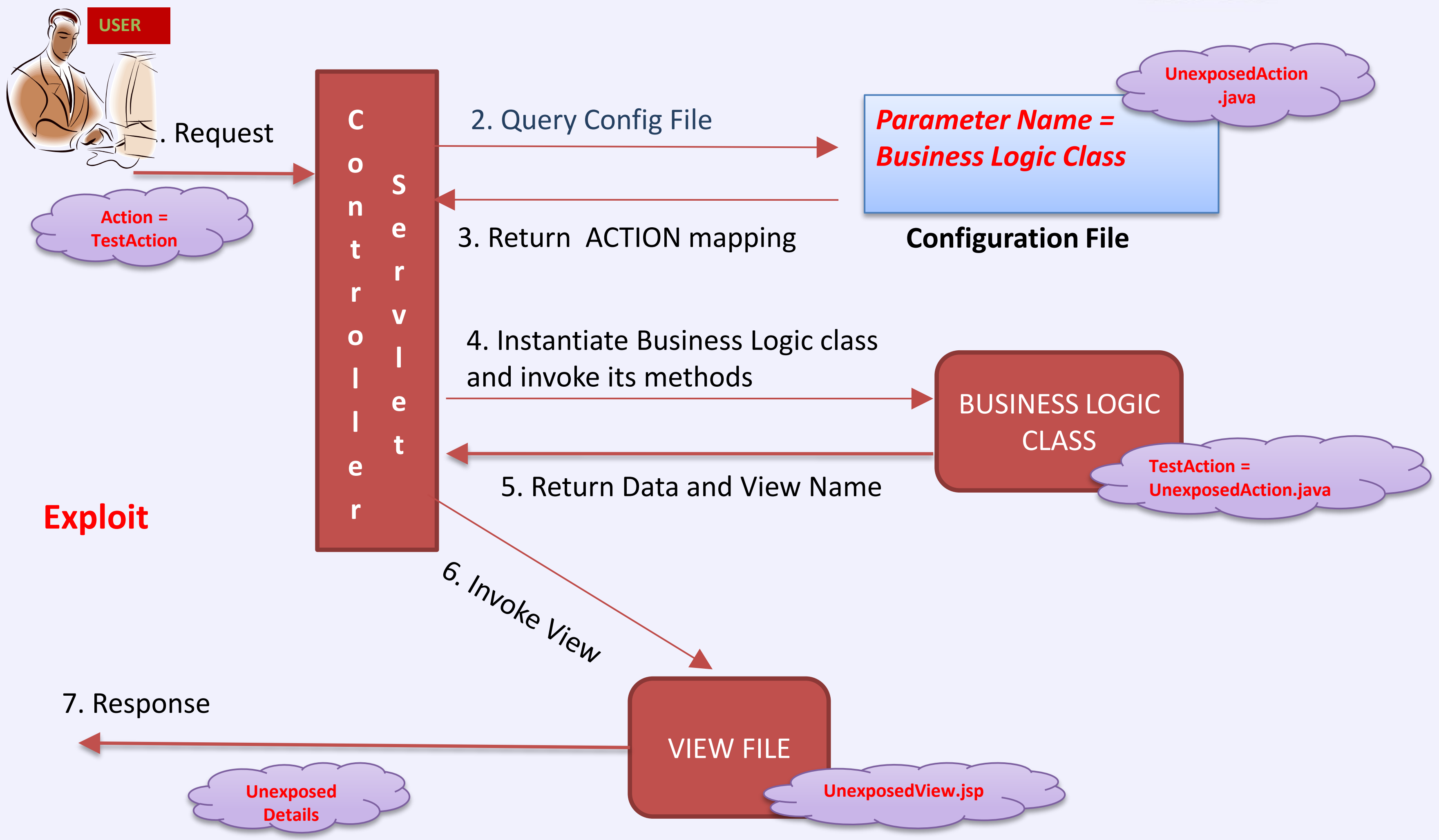
- **What can go wrong in this design:**
  - Unexposed ***Files*** may be accessible to the user

Insecure Business Logic Invocation



Normal Functioning

Insecure Business Logic Invocation



Insecure Business Logic Invocation

DEMO -

Unauthorized Access to Hidden Business Logic Class

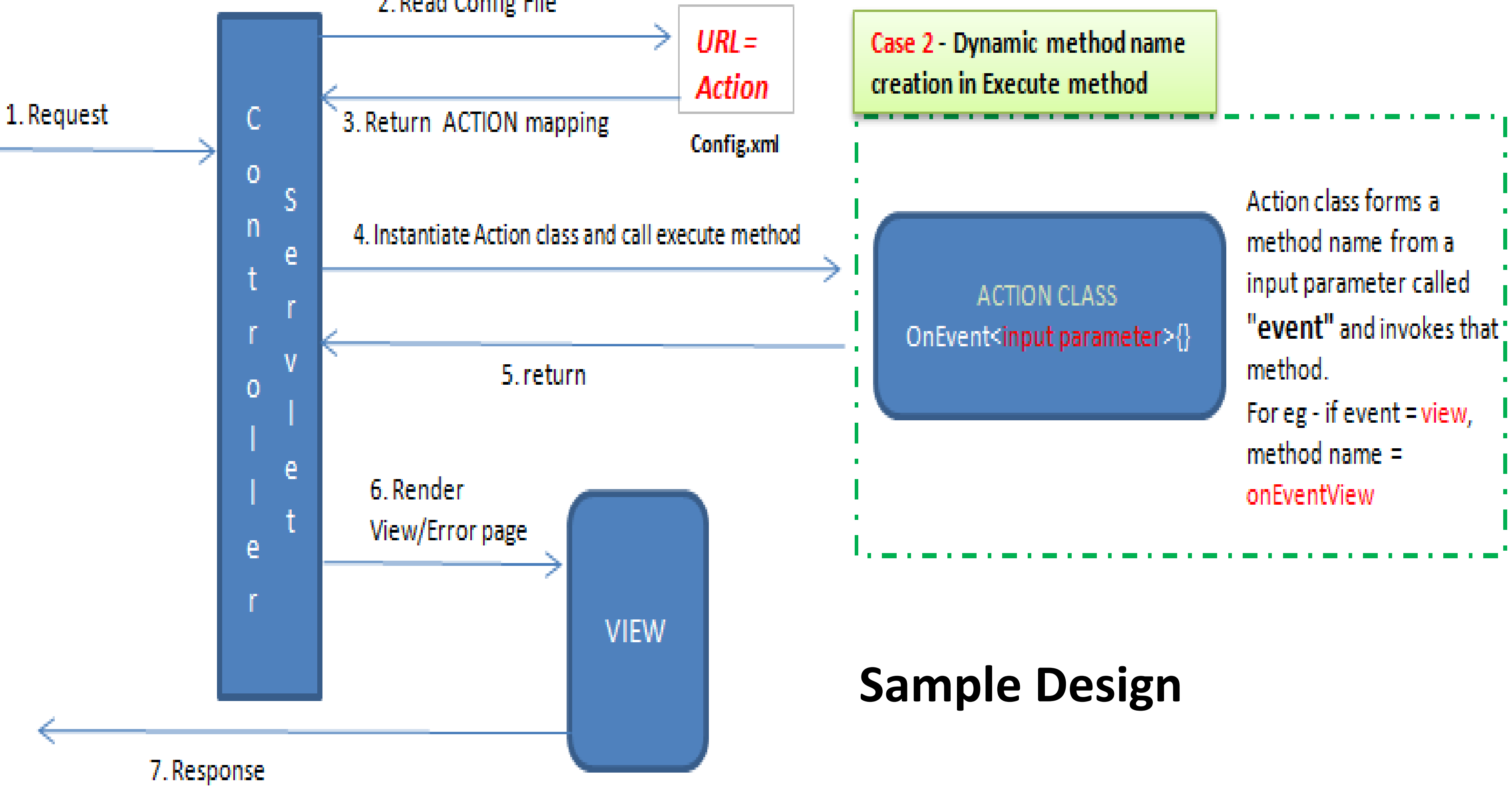
Insecure Business Logic Invocation

OWASP
The Open Web Application Security Project

Another important scenario –

Request parameters used to identify method names of the business logic class

Consider this design



Case 2 - Dynamic method name creation in Execute method

Action class forms a method name from a input parameter called "**event**" and invokes that method.

For eg - if event = view, method name = onEventView

Sample Design

Insecure Business Logic Invocation

- **What can go wrong in this design:**
  - Users can try to perform actions not authorized to them

Insecure Business Logic Invocation

OWASP
The Open Web Application Security Project

DEMO –

Unauthorized Access to unexposed Business Logic Method

Insecure Business Logic Invocation

OWASP
The Open Web Application Security Project

- **Security Measures:**
  - Remove **ALL** redundant/test/unexposed business logic configurations from the file
  - Apply Authorization check before processing business logic
  - Apply a mapping on method/class/view names with the privilege level of the users

Backdoor Parameters

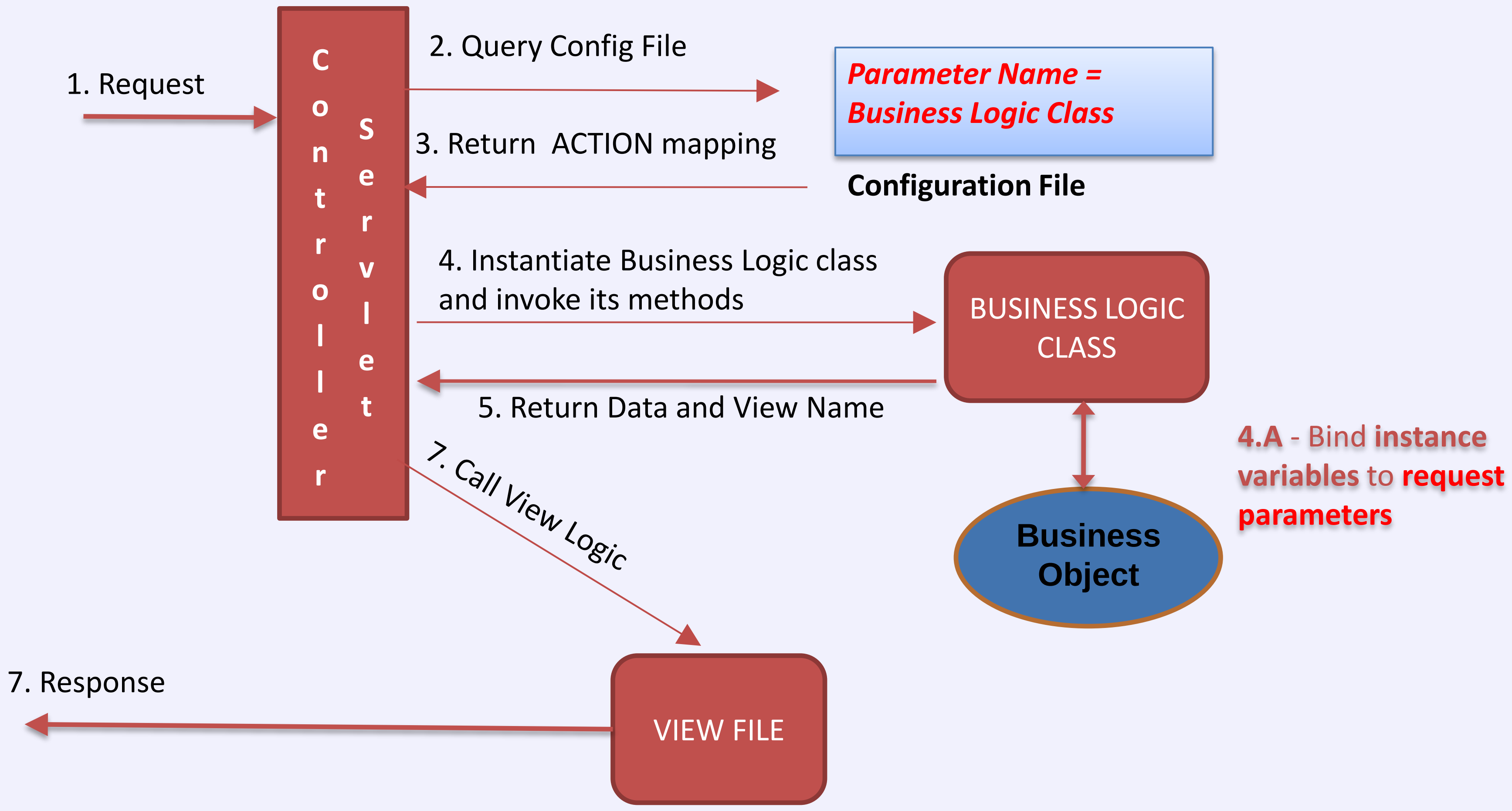
– Insecure Data Binding

Insecure Data Binding

OWASP
The Open Web Application Security Project

- **Lets understand the design:**
  - The design uses a data binding logic to bind user inputs to business/form object variables

What is Data Binding?


1. Request
Controller Servlet
2. Query Config File
Parameter Name = Business Logic Class
Configuration File
3. Return ACTION mapping
4. Instantiate Business Logic class and invoke its methods
BUSINESS LOGIC CLASS
5. Return Data and View Name
Business Object
4.A - Bind instance variables to request parameters
7. Call View Logic
VIEW FILE
7. Response

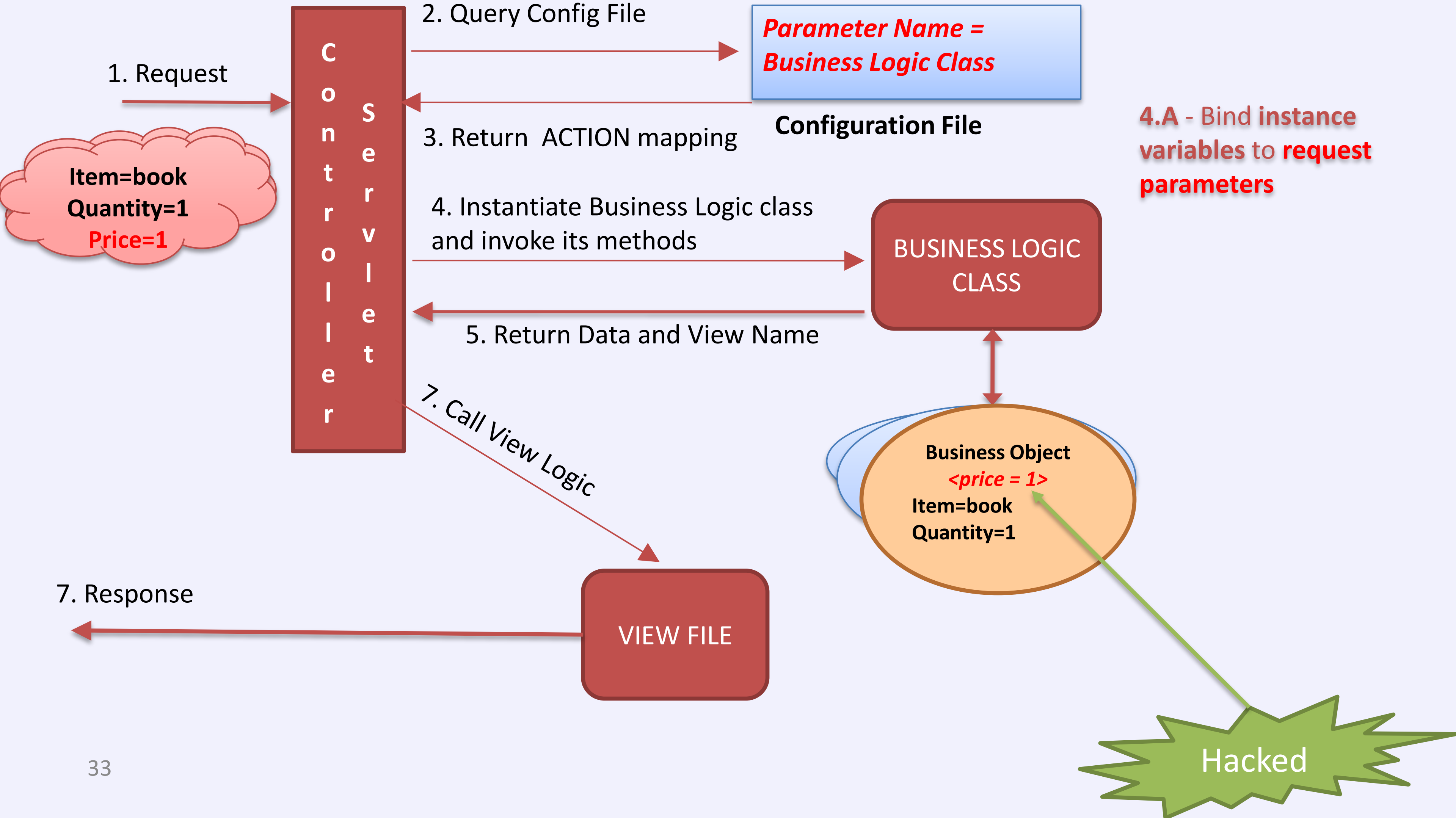
Insecure Data Binding

OWASP
The Open Web Application Security Project

- **What can go wrong in the design:**
  - A user may be able to assign values to unexposed variables of business objects

Insecure Data Binding

OWASP
The Open Web Application Security Project



Insecure Data Binding

OWASP
The Open Web Application Security Project

DEMO –

Unauthorized access by exploiting data binding flaw

Insecure Data Binding

- **Security Measures:**
  - Do not place key variables related to business rules, which are not dependent on user inputs in objects that get bound to request variables
  - Initialize key variables **after** the request to variable binding logic
  - Use "disallow" binding logic for certain variables, if provided by the framework

BACKDOOR PARAMETERS

– **Insecure Decision Logic**

Incorrect Decision Logic

- **Lets understand the design**
  - The application takes business logic decisions based on **presence** or **absence** of a parameter.
    - For instance – isAdmin, isSuccess
  - Menus/input controls are hidden from certain users, generally observed in ASP.NET applications.

Incorrect Decision Logic

OWASP
The Open Web Application Security Project

- **What can go wrong in the design:**
  - The design believes in the concept of – “what is hidden is secure”
  - Server side behavior can be influenced with request parameters
  - Users can perform unauthorized operations in the application.

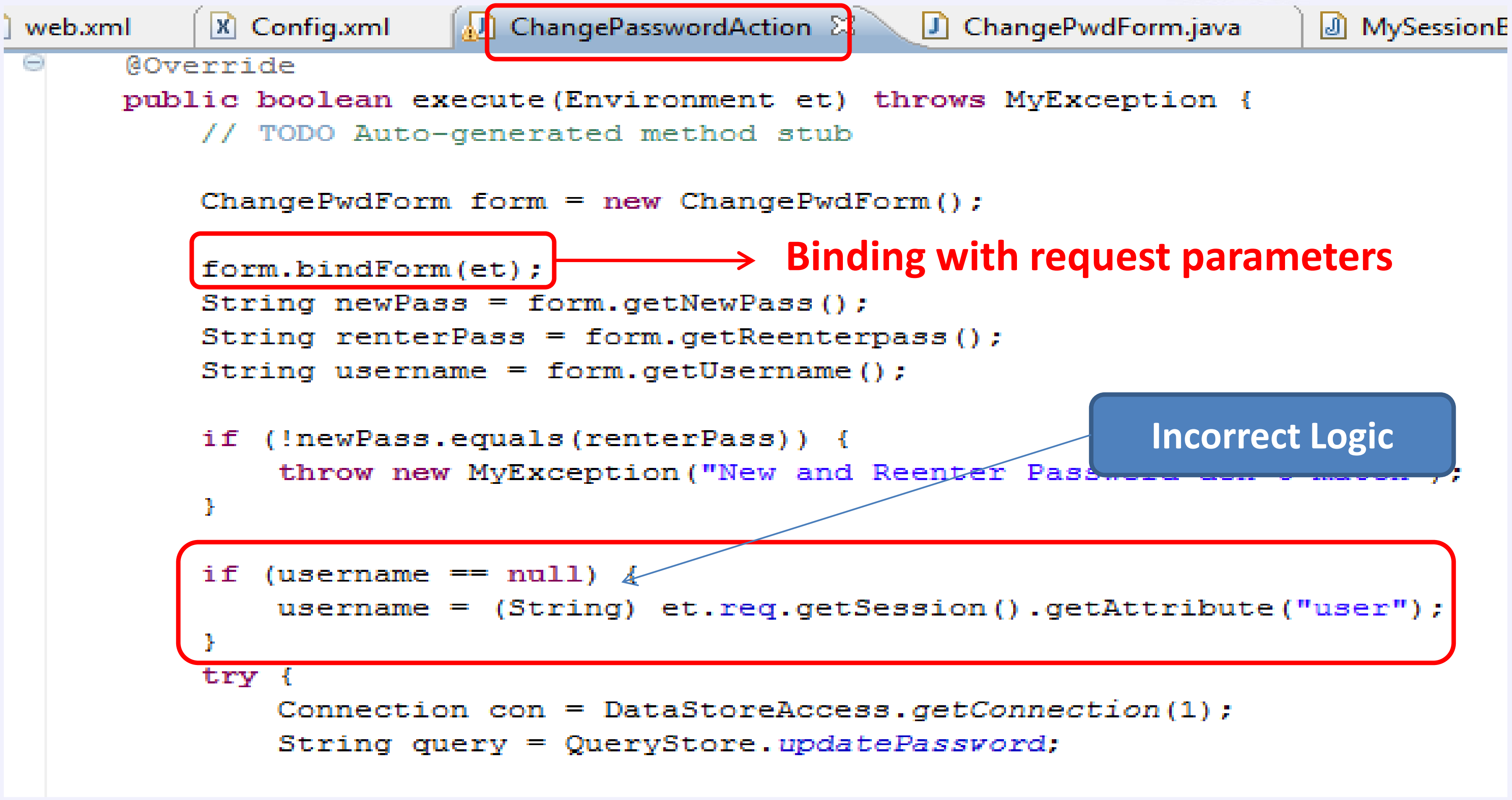
Consider a scenario

OWASP
The Open Web Application Security Project

- Consider a password reset feature of the applications:
  - **Scenario:**
    - Admin users can reset passwords of other users
    - Normal users can reset ONLY their passwords

Lets look at the flaw

OWASP
The Open Web Application Security Project



```
@Override
public boolean execute(Environment et) throws MyException {
    // TODO Auto-generated method stub

    ChangePwdForm form = new ChangePwdForm();

    form.bindForm(et);
    String newPass = form.getNewPass();
    String renterPass = form.getReenterpass();
    String username = form.getUsername();

    if (!newPass.equals(renterPass)) {
        throw new MyException("New and Reenter Password don't match");
    }

    if (username == null) {
        username = (String) et.req.getSession().getAttribute("user");
    }
    try {
        Connection con = DataStoreAccess.getConnection(1);
        String query = QueryStore.updatePassword;
```

Lets understand the flaw

OWASP
The Open Web Application Security Project

Flaw:

- Here, **absence** of username in the request is considered as a request from normal user.

Assumption:

As an option to add username is not given to non-admin users, the username field will always be absent in their request.

Consider a scenario

OWASP
The Open Web Application Security Project

What if a non-admin user sends additional username parameter in the request?

- The server will be fooled to believe that the request coming from the admin user.
- The user will be able to change password of other users

OWASP
The Open Web Application Security Project

DEMO

Unauthorized access to change password of other users

Incorrect Decision Logic

OWASP
The Open Web Application Security Project

- **Security Measures:**
  - Don’t believe in – “If it is hidden it is secure”
  - Apply authorization checks wherever necessary
  - Do not use unvalidated inputs for taking business logic decisions – Use session variables/database values

INCORRECT PLACEMENT OF CHECKS

Incorrect Placement of Checks

OWASP
The Open Web Application Security Project

- **Lets understand the design:**
- The design uses multiple components like MVC.
- The authentication check is implemented on all the views of the application
  - if we try to access any view – For instance, "Adduser.jsp" without authentication, it will be disallowed

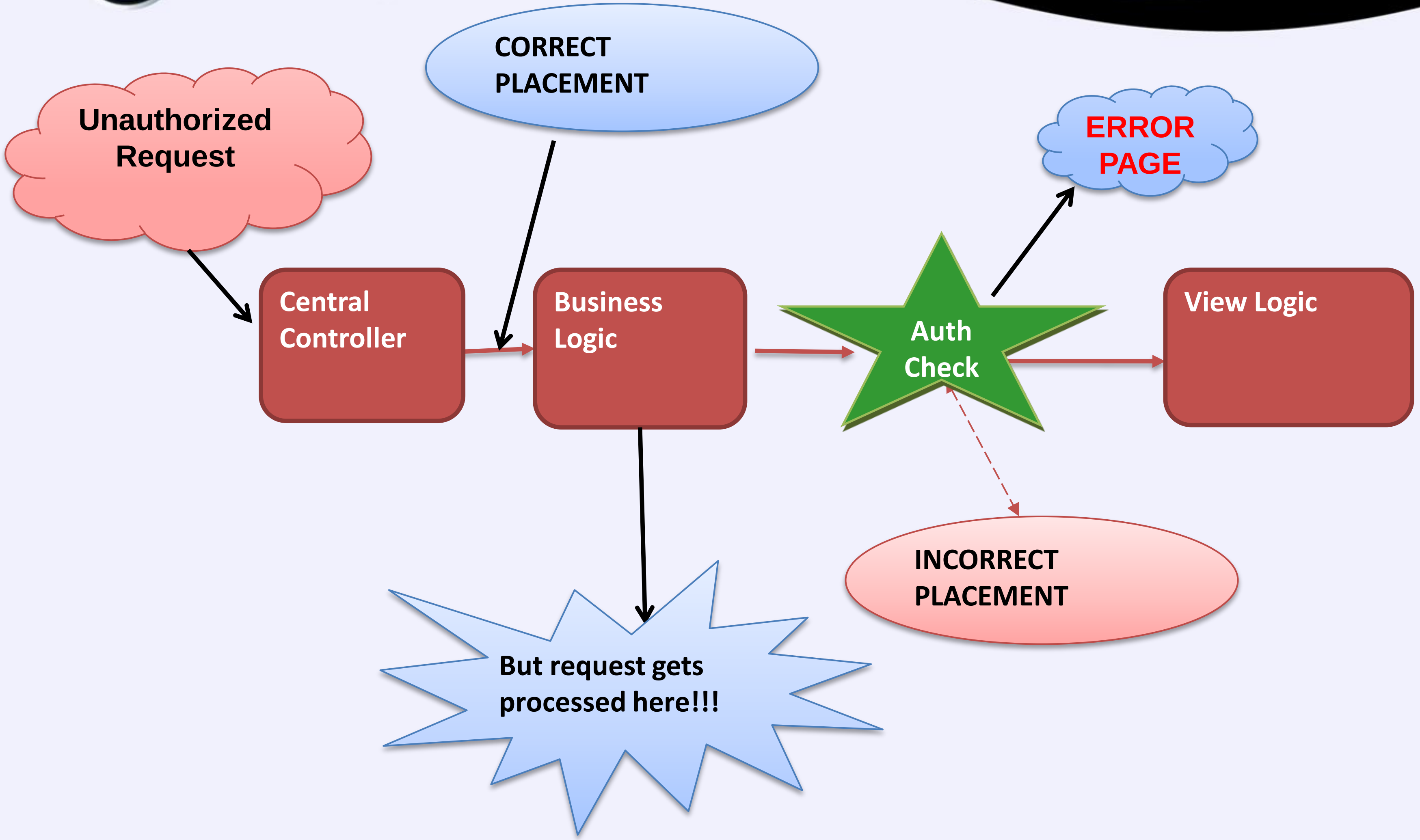
Incorrect Placement of Checks

OWASP
The Open Web Application Security Project

- **What can go wrong in the design:**
  - Placement of checks can be incorrect
  - Business logic components could be placed before the authentication check
  - Users will be able to bypass the authentication or any such security check

Incorrect Placement of Checks

OWASP
The Open Web Application Security Project



Incorrect Placement of Checks

OWASP
The Open Web Application Security Project

DEMO –

Unauthorized access due to incorrect placement of checks

Incorrect Placement of Checks

OWASP
The Open Web Application Security Project

- **Security Measures:**
  - Place all validation checks before request processing logic

OWASP
The Open Web Application Security Project

INTER APPLICATION COMMUNICATION

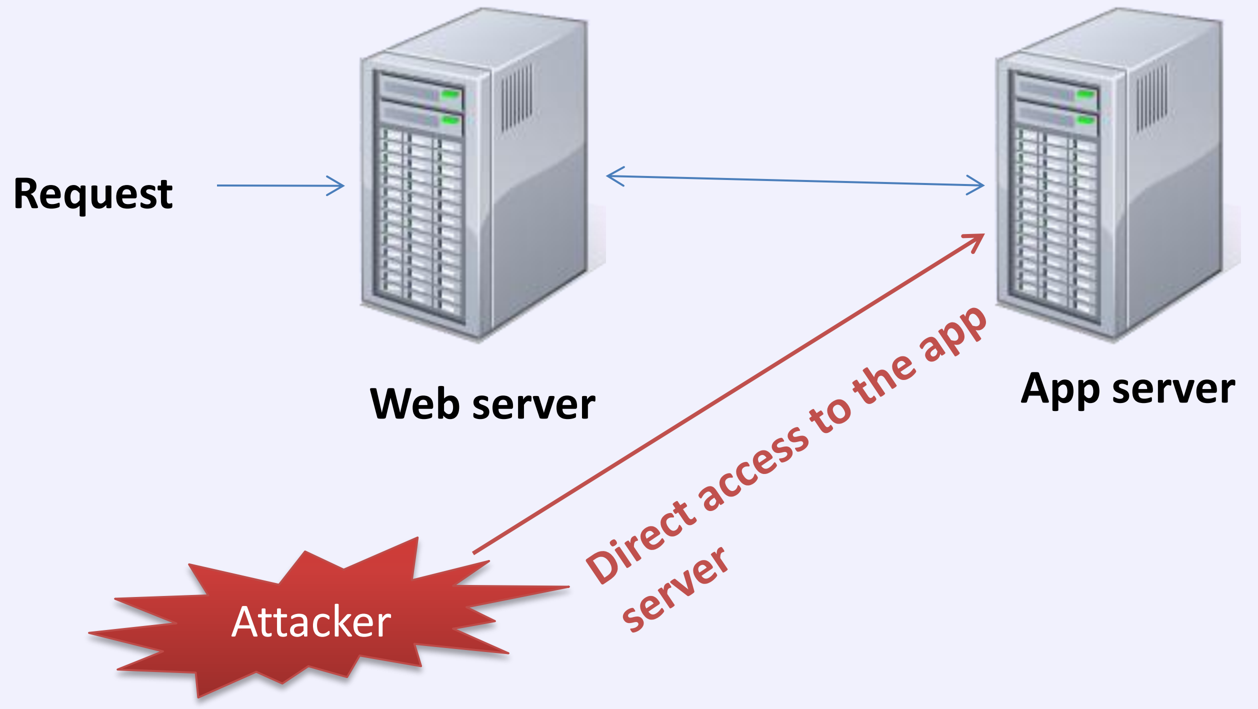
Security Areas in Inter-App Communication

- Verifying the authenticity of the user
- Secure Data transmission
- Tamper proof communication
- Prevention of Replay Attacks

Inter App Communication

OWASP
The Open Web Application Security Project

- Verify authenticity of the user
  - Consider a case of web to app server communication



Inter App Communication

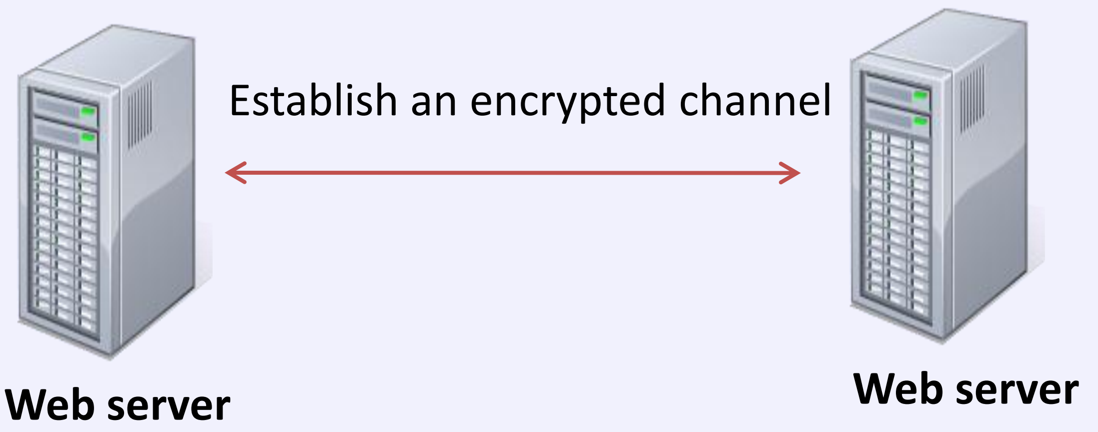
OWASP
The Open Web Application Security Project

- Verify authenticity of the user
  - In the app server, verify the identity of the requesting user using:
    - Declarative access control (container managed)
    - Programmatic access control logic

Inter App Communication

OWASP
The Open Web Application Security Project

- Secure Data transmission
  - Consider a server to server communication



Inter App Communication

OWASP
The Open Web Application Security Project

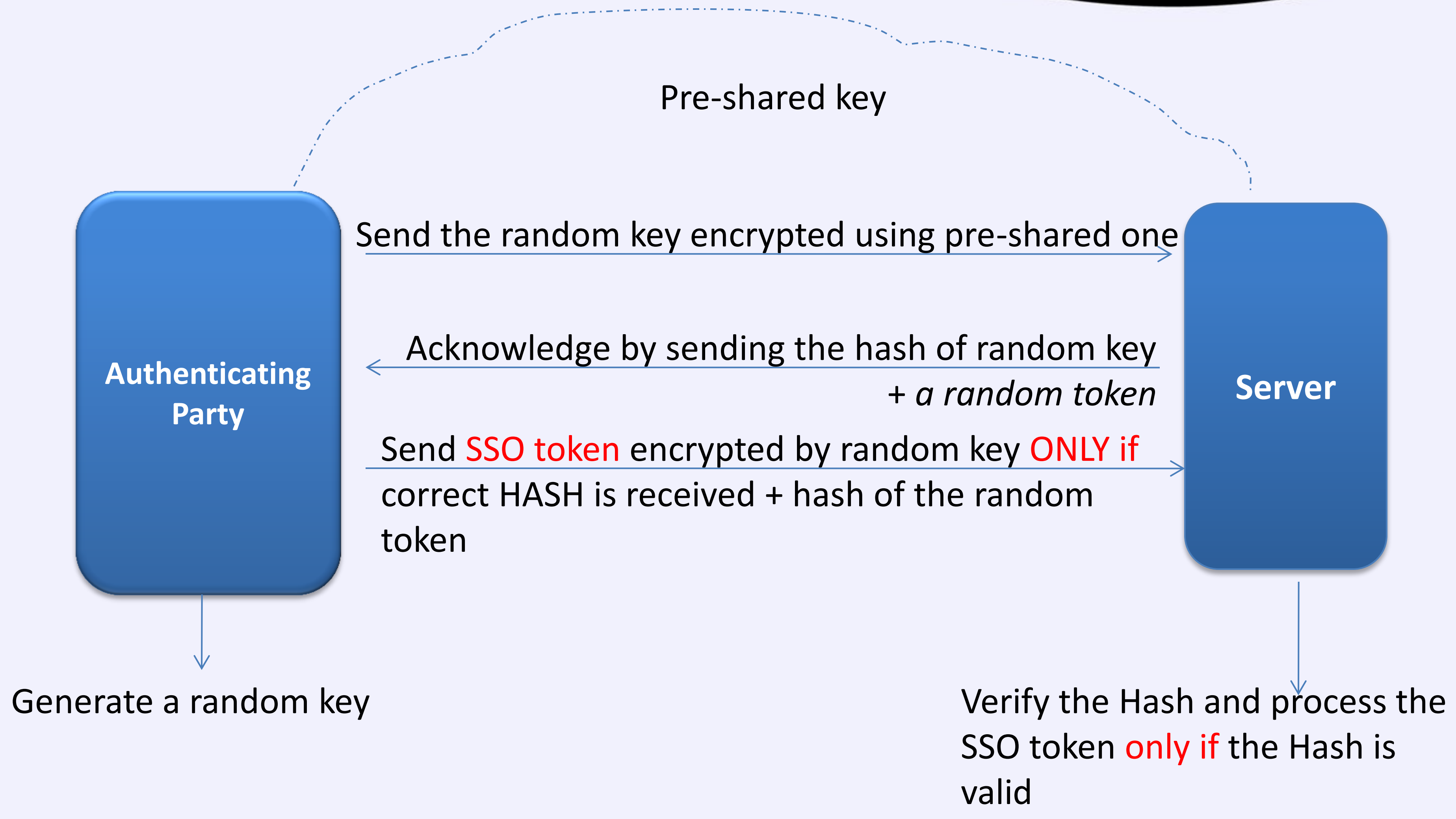
- Secure Data transmission
  - Implement an encrypted channel like SSL or IPSec, wherever possible
  - If the channel cannot be encrypted, encrypt sensitive data like account ID, etc. using a pre-shared key

Inter App Communication

OWASP
The Open Web Application Security Project

- Tamper proof communication
- Prevent Replay Attacks
  - Consider a scenario like SSO or payment gateway transaction

SSO Implementation


Pre-shared key
Authenticating Party
Server
Send the random key encrypted using pre-shared one
Acknowledge by sending the hash of random key + *a random token*
Send SSO token encrypted by random key ONLY if correct HASH is received + hash of the random token
Generate a random key
Verify the Hash and process the SSO token only if the Hash is valid

Inter App Communication

OWASP
The Open Web Application Security Project

- Verify the authenticity of the user
- Send data over an encrypted channel
- Implement HMAC of the request parameter wherever needed
- Use 2 way handshake in cases like SSO
  - Use different pair of pre-shared keys in scenarios where deployment is multiple customer sites

Insecure Design - Recap

OWASP
The Open Web Application Security Project

- Insecure Business Logic Invocations
  - Files
  - Methods
- Backdoor Parameters
  - Insecure Data Binding
  - Incorrect Decision Logic
- Incorrect Placement of Checks
- Inter Application Communication

OWASP
The Open Web Application Security Project

CHECKLIST FOR SECURE DESIGN

OWASP
The Open Web Application Security Project

Questions

OWASP
The Open Web Application Security Project

Thank You
&
Share your feedback with us.

rao.ashish20@gmail.com
AND
sidhanbu@gmail.com