University of BRISTOL

Computer System- B Security

Introduction to Web Security P1

Sanjay Rawat

We will learn about...

We will learn about…

- Basics of Web application and deployment

We will learn about…

- Basics of Web application and deployment
- Web vulnerabilities
  - SQL injection
  - XSS
  - CSRF
  - …

Background

- HTTP – de facto protocol when talking about WEB.
- Historically designed for static contents.
- Security was never a concern.
- Based on a *simple* client-server model.
- This is not what we see in today’s Web Applications.
- Highly technical and complex in nature

Typical web model

Typical web model

- Interaction between a browser and server (+ other stuff)

Typical web model

- Interaction between a browser and server (+ other stuff)
- modern web pages allow personalized dynamic contents.

Typical web model

- Interaction between a browser and server (+ other stuff)
- modern web pages allow personalized dynamic contents.
- web pages may also run client-side scripts that “change” the Internet browser into an interface.

Typical web model

- Interaction between a browser and server (+ other stuff)
- modern web pages allow personalized dynamic contents.
- web pages may also run client-side scripts that “change” the Internet browser into an interface.
- modern web sites allow the capture, processing, storage and transmission of sensitive customer data.

HTTP protocol

HTTP protocol

- HTTP is a stateless protocol

HTTP protocol

- HTTP is a stateless protocol
- HTTP URL: `host/dir/resource`
  - `Host to IP (DNS)`

HTTP protocol

- HTTP is a stateless protocol
- HTTP URL: `host/dir/resource`
  - `Host to IP (DNS)`
- HTTP requests
  - GET (part of the URL)
  - POST (part of the header body)
  - Because of stateless property, a fresh request is made with no memory of the previous interactions.

HTML

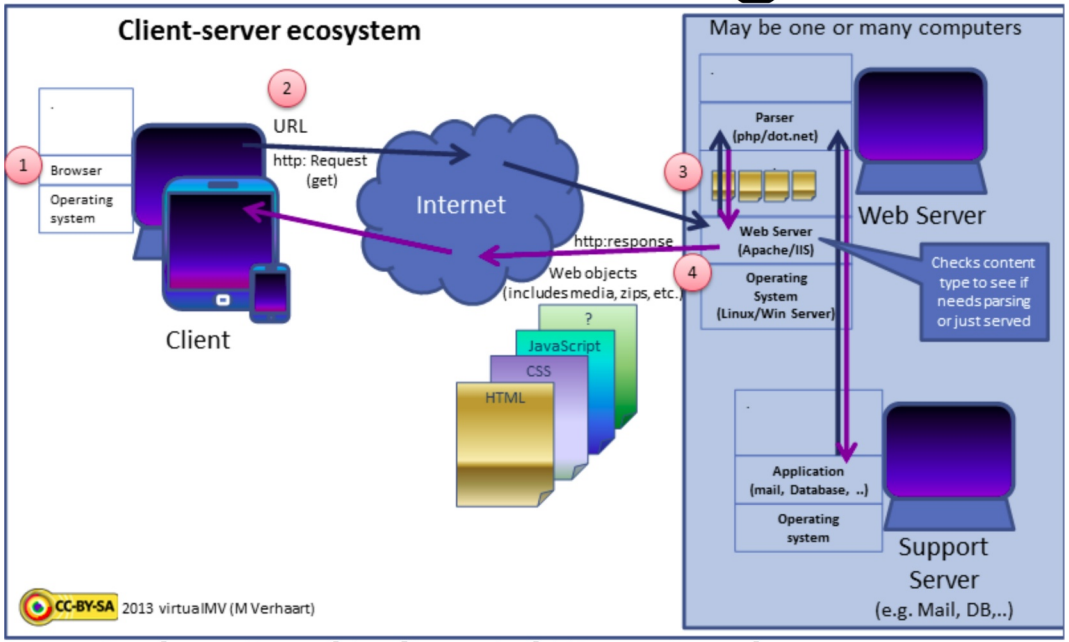
- HTTP request and response are rendered in HTML
- HTML forms
  - Allow for using key-value pairs to be processed by the server
  - Together with other languages (LabaScript/PHP) provide a very powerful interaction mode
  - `img, iframe, href, etc.`

Static vs Dynamic pages

- Static pages
  - Static pages are a typical HTML + CSS assisted
  - Rendered the same content each time
  - Only way to change is to manually change the server side page!
  - Interaction is via hyperlinks on the page.
  - Was good from security point of view!

- Dynamic pages (from https://en.wikipedia.org/wiki/Dynamic_web_page)
  - Are interactive in the sense that based on the request parameters, a new page is rendered (server side).
  - Pages contain other scripting code (JavaScript) that changes rendering on the client-side (client side)
  - It also involves other entities, like application serves, DB etc.

Typical Dynamic webpage rendering



Src: https://en.wikipedia.org/wiki/Dynamic_web_page

So..

So..

- Web applications are computer programs allowing website visitors to submit and retrieve data *to/from a database*, *for example*, *over the Internet* using their preferred web browser.

So..

- Web applications are computer programs allowing website visitors to submit and retrieve data *to/from a database*, *for example*, *over the Internet* using their preferred web browser.
- Web applications query and dynamically generate web documents.

So..

- Web applications are computer programs allowing website visitors to submit and retrieve data *to/from a database*, *for example*, *over the Internet* using their preferred web browser.
- Web applications query and dynamically generate web documents.
- The documents are generated in a standard format to allow support by all browsers (e.g., HTML or XHTML +

- JavaScript is one form of client side script that permits dynamic elements on each page.
- The web browser is key – it interprets and runs all scripts!!
- All requests and responses are nothing but codes written in various languages/scripts.
- And, as we have seen, codes are powerful and dangerous, if not managed!!
-

Some features

- HTML for look and feel
- JavaScript- a powerful language for dynamic content

```
<html>
   …
   <script> javascript code </script>

      …
   </html>
```

- JavaScript can perform many tasks, including:
  - Access to files (e.g. **readAsText**)
  - Access to system resources…. Thus
- Javascript is a language and can do whatever the hosting environment allows it to do.
- Javascript in the **Browser** is sandboxed.

So??

- We also know that now a days, many web sites stores data on local machine, e.g. cookies, user data (auto fill), passwords etc.
- JavaScript can read resources -> we can steal any information???
- There are security mechanisms to take care of it!!

- Wiki says….
  - … Browser authors contain this risk using two restrictions.
  - Scripts run in a sandbox in which they can only perform web-related actions, not general-purpose programming tasks like creating files.
  - Scripts are constrained by the same origin policy:

- scripts from one web site do not have access to information such as usernames, passwords, or cookies sent to another site.

- Most JavaScript-related security bugs are breaches of either the *same origin policy* or the sandbox.

Same Origin Policy

Same Origin Policy

- For absolute URIs, the origin is the triple {protocol, host, port}.

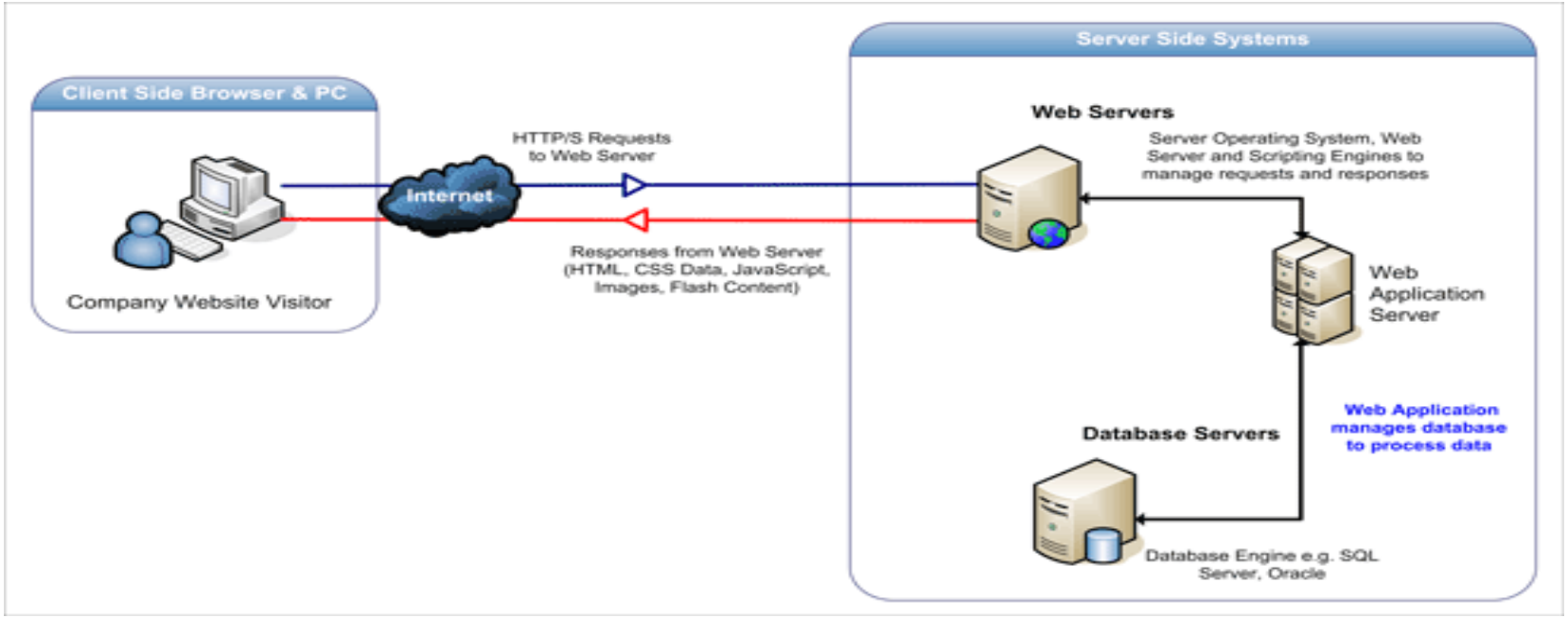
Same Origin Policy

- For absolute URIs, the origin is the triple {protocol, host, port}.
- Two resources are considered to be of the same origin if and only if all these values are exactly the same.

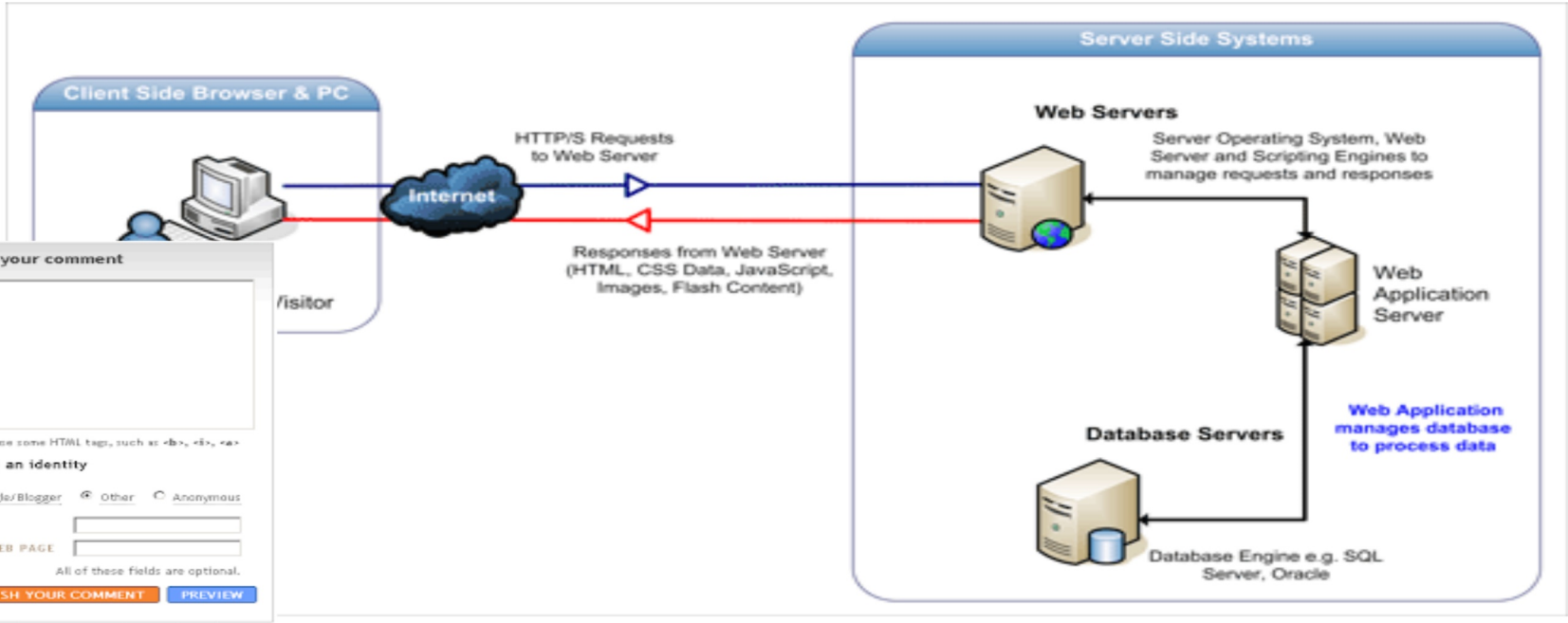
Same Origin Policy

- For absolute URIs, the origin is the triple {protocol, host, port}.
- Two resources are considered to be of the same origin if and only if all these values are exactly the same.
- Example:
  - Allowed: http://www.abc.com/doc1.html & http://www.abc.com/doc2.html
  - Not allowed: http://www.abc.com:8080/doc1.html

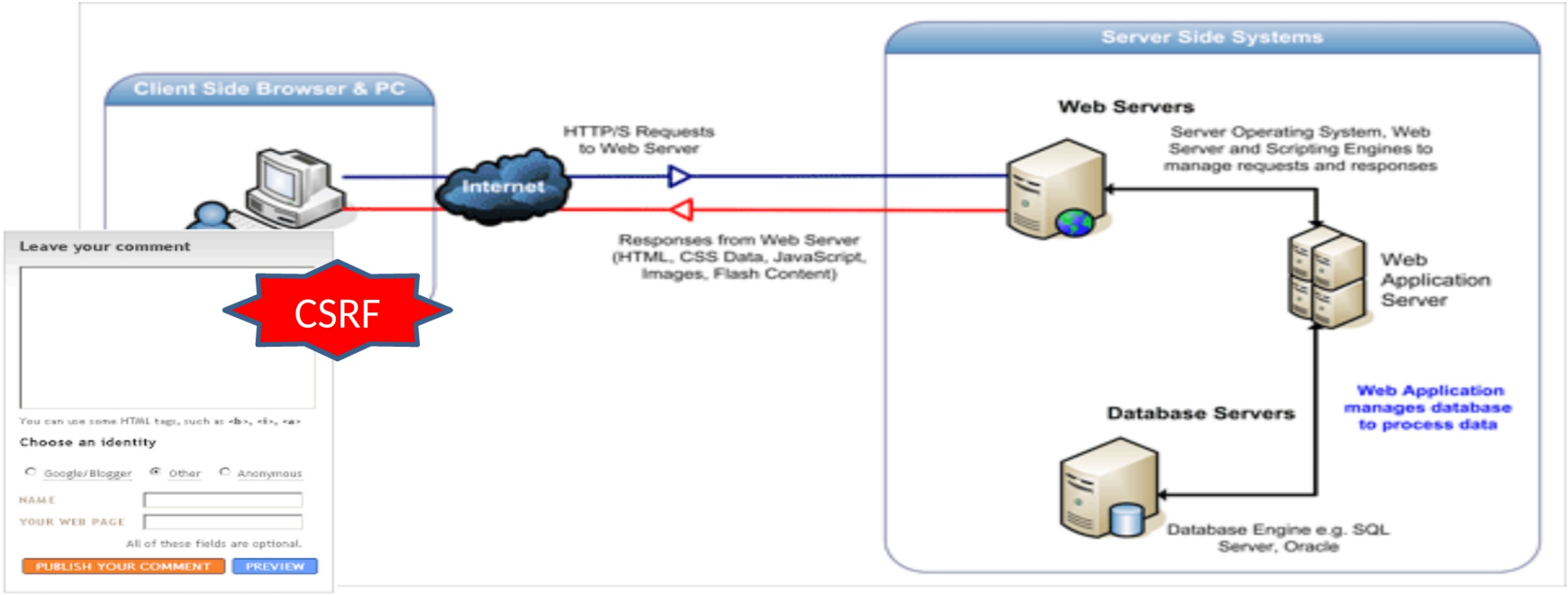
Typical Web Application Vulnerabilities



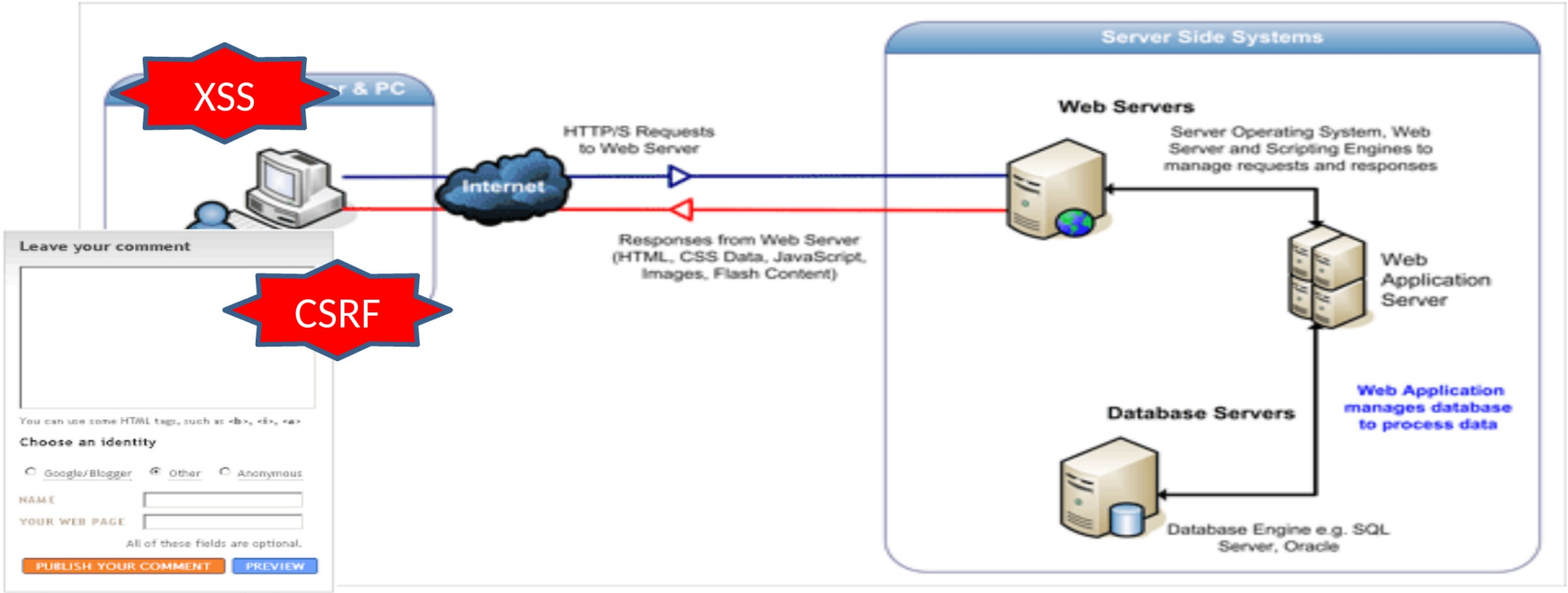
Typical Web Application Vulnerabilities



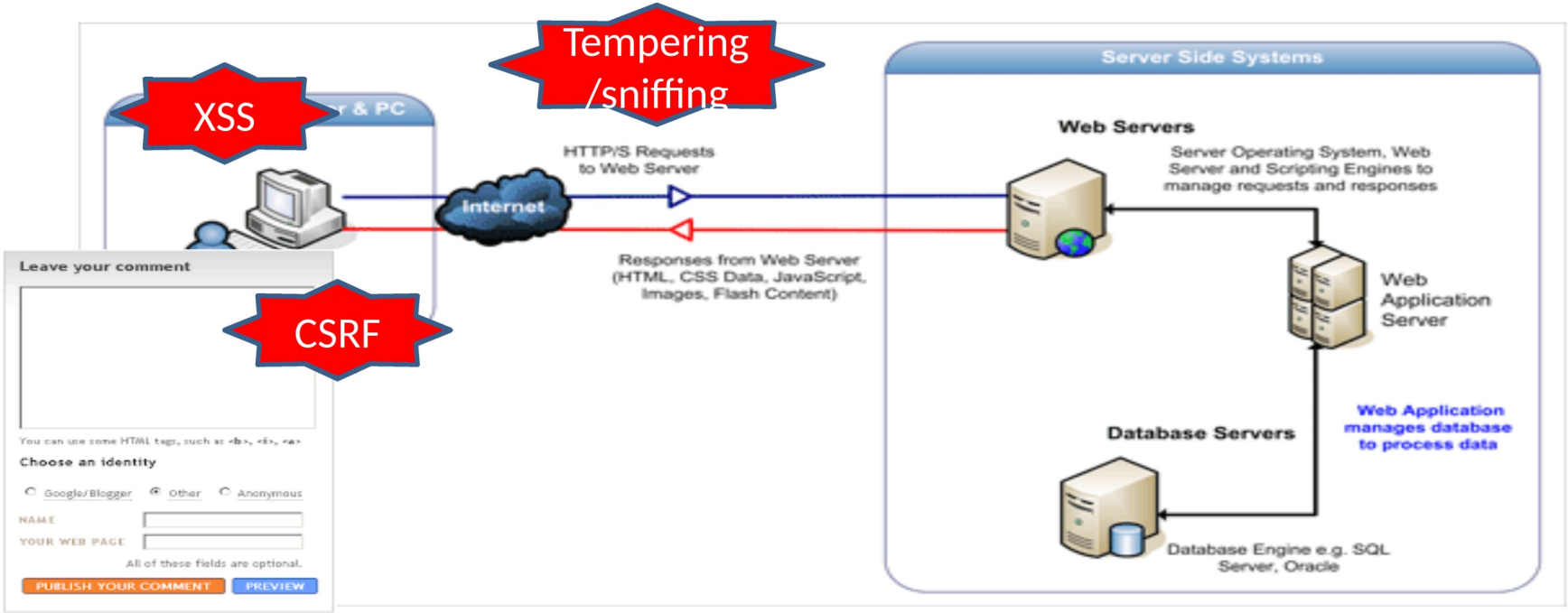
Typical Web Application Vulnerabilities



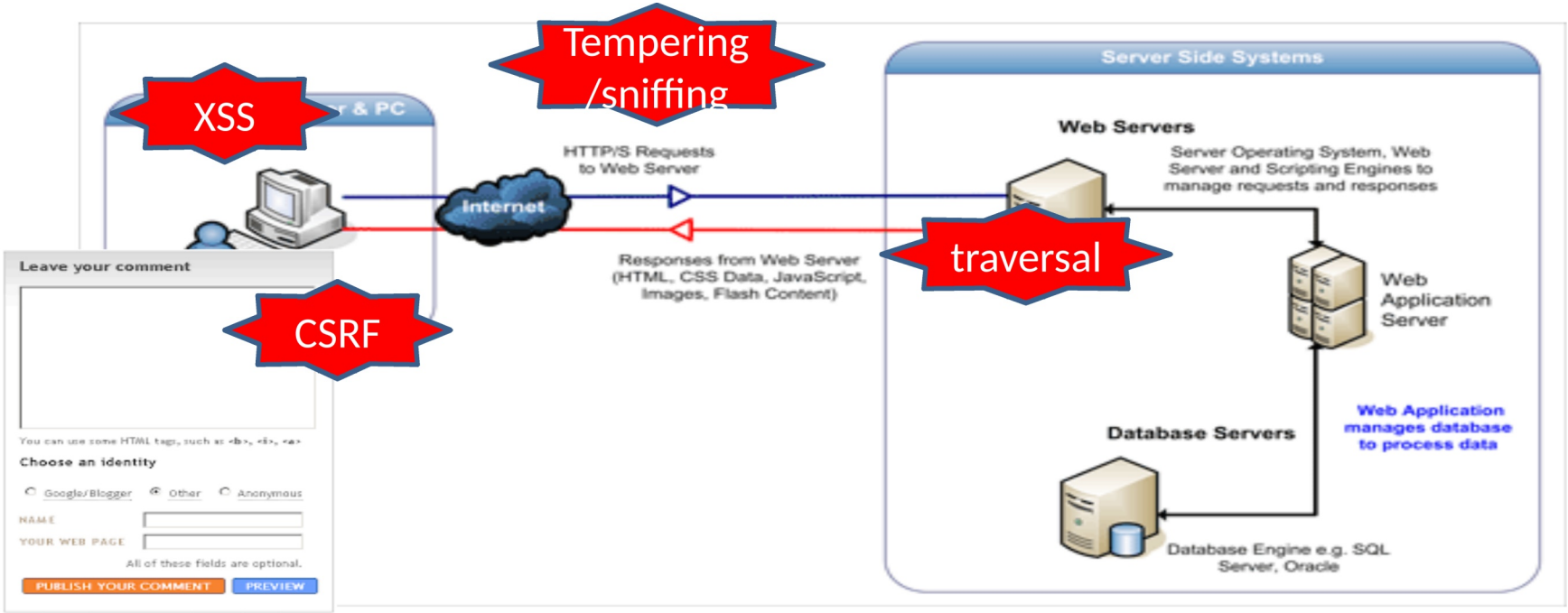
Typical Web Application Vulnerabilities



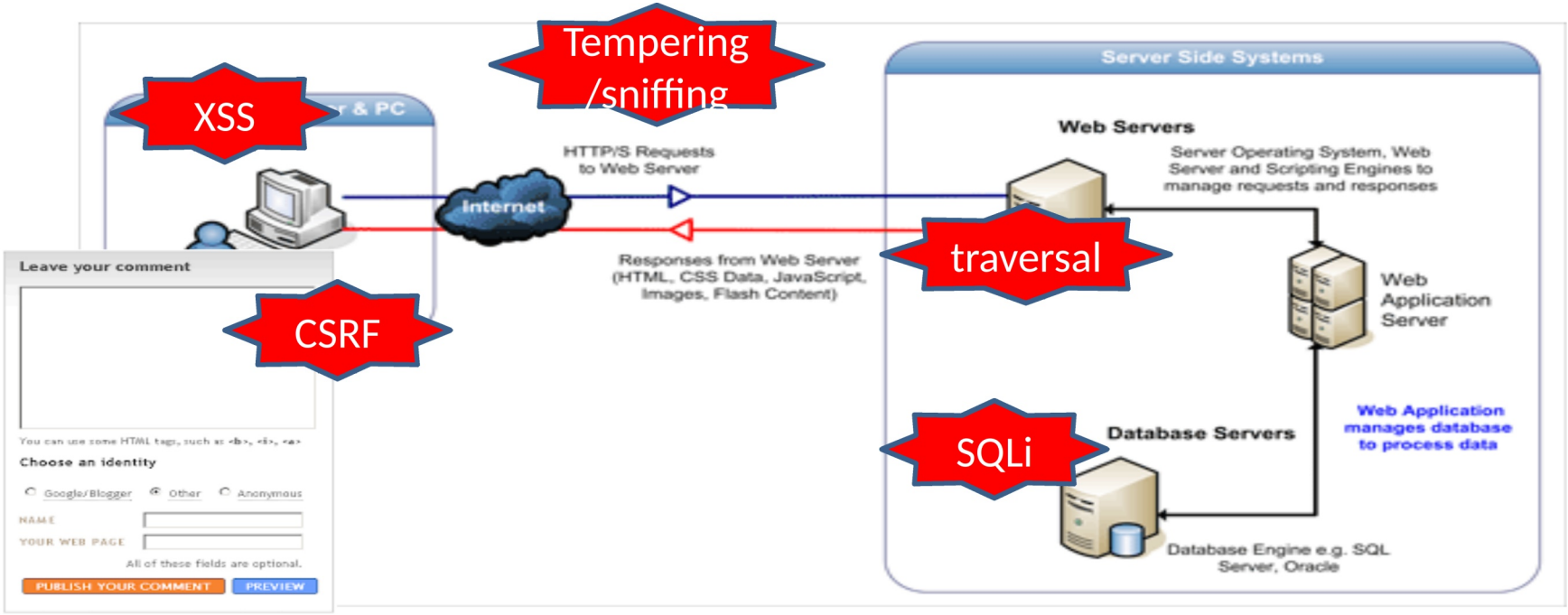
Typical Web Application Vulnerabilities



Typical Web Application Vulnerabilities



Typical Web Application Vulnerabilities



Typical Web Application Vulnerabilities