Web Application Hacking

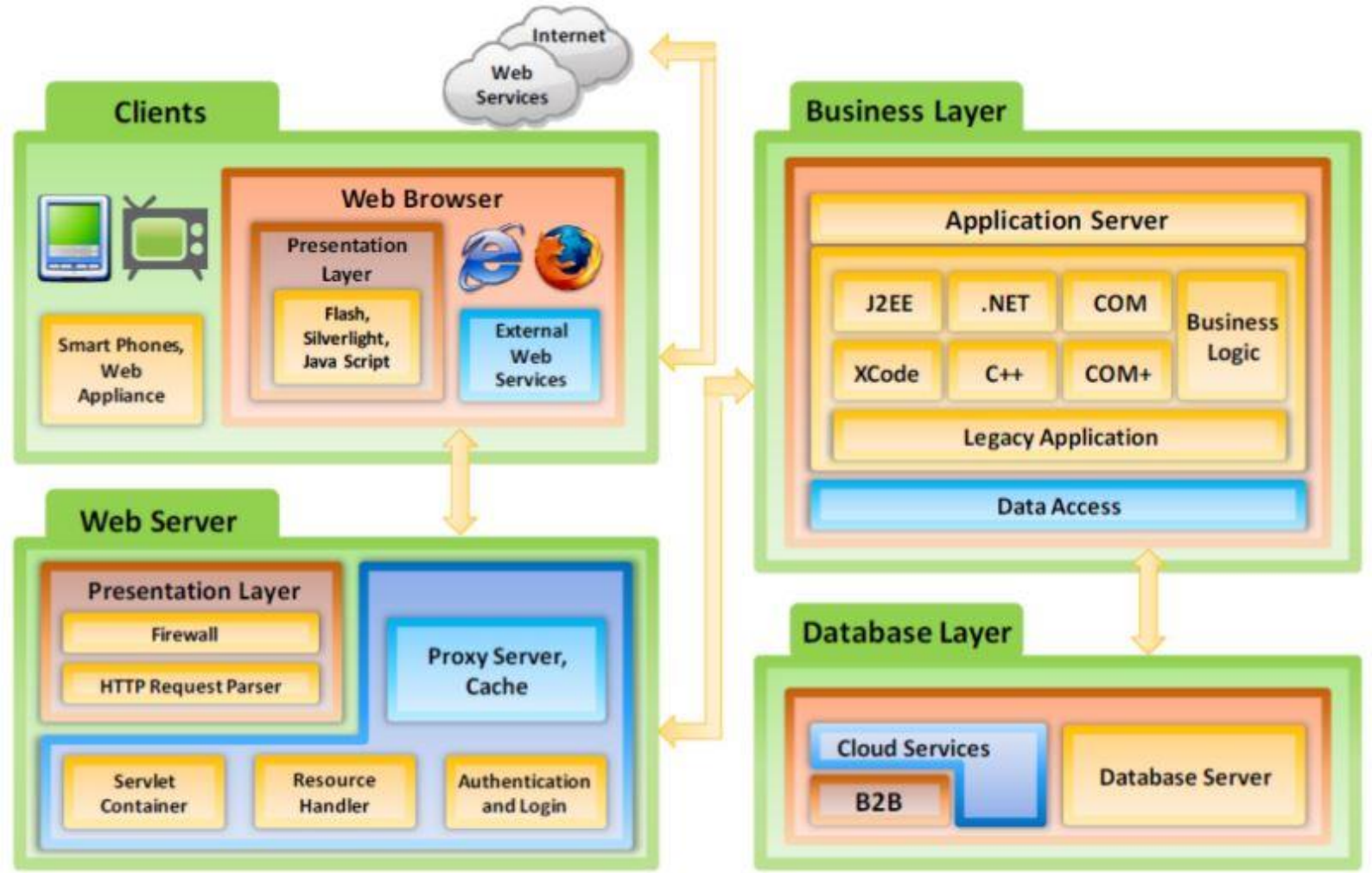
INTRO TO WEB APPLICATIONS

- Provide an interface between users and a web server through web pages or scripts executed on a client browser
- Three layered architecture
  - Presentation (web pages)
  - Logic (Background processing)
  - Data (Underlying databases, files, etc)
- Primarily uses HTTP / HTTPS (stateless)

WEB APPLICATION COMPONENTS

- Web server
  - hardware and software that delivers content
- Application
  - Program that accepts requests and processes them
- Login and Logout
  - Method for starting and ending a session
- Session Tracking
  - Cookies, URL rewriting or SSL info
- User Permissions
  - What users are /are not allowed to access
- Data Store
  - Data maintained by the application

WEB ARCHITECTURE



OWASP TOP 10 WEB VULNERABILTIES FOR 2021

OWASP Top 10 Vulnerabilities

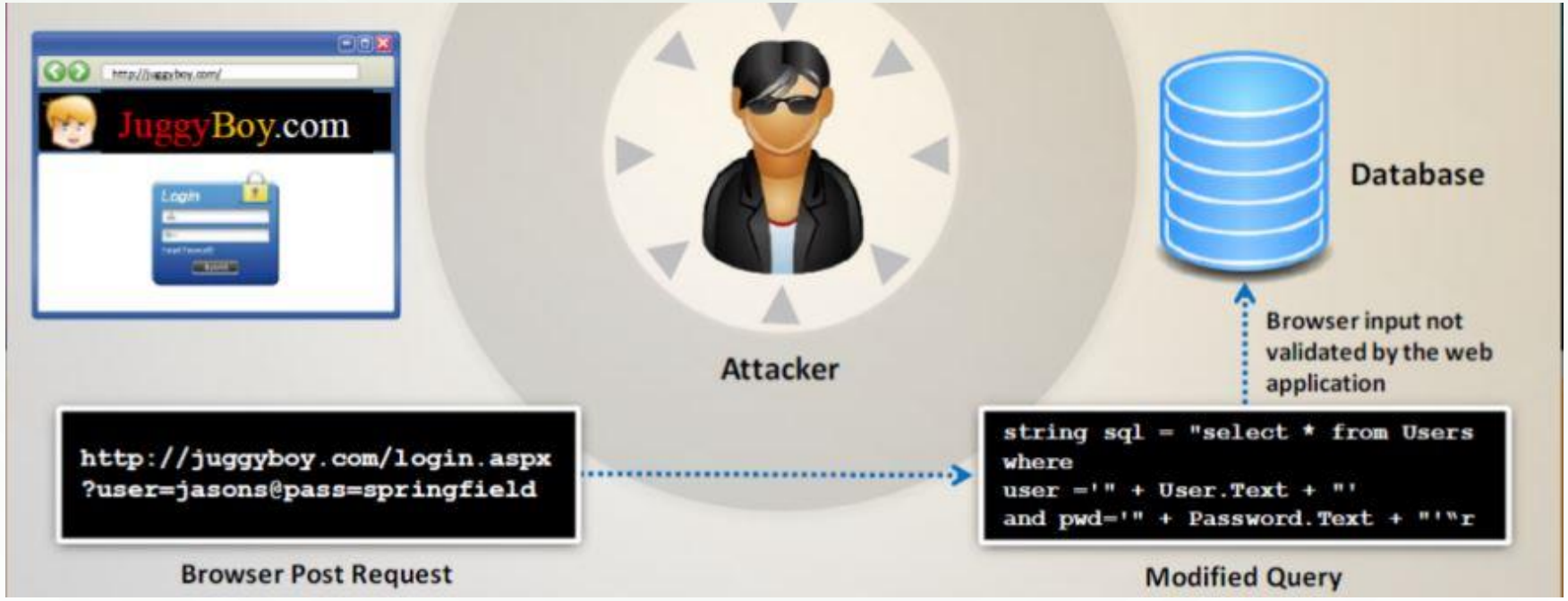
OWASP Top 10 Vulnerabilities 2021	Position in 2017
1. Broken Access Control	**5th**
2. Cryptographic Failures	3rd
3. Injection	1st
4. Insecure Design	New Category
5. Security Misconfiguration	6th
6. Vulnerable and Outdated Components	9th
7. Identification and Authentication Failures	2nd
8. Software and Data Integrity Failures	New Category
9. Security Logging and Monitoring Failures	10th
10. Server-Side Request Forgery	New Category

ATTACK VECTORS

- Defined as path or means by which an attacker can compromise a service / system
- Examples for Web Apps
  - Parameter manipulation
  - Injection
  - Cookie and session manipulation
  - Server misconfiguration
  - Cross site scripting
  - DoS

UNVALIDATED INPUT

- Refers to a flaw in a web application where client input is not validated before processed by the server.
- Allows cross site scripting, buffer overflows and injection attacks



PARAMETER TAMPERING

- Manipulation of parameters exchanged between client and server to modify application data such as credentials, permissions, etc

- Takes advantage of cases when the application programmer uses hidden fields or preselected and locked fields as the only security measures
- Ex.

```
<input type="hidden" name="price" value="100">
```

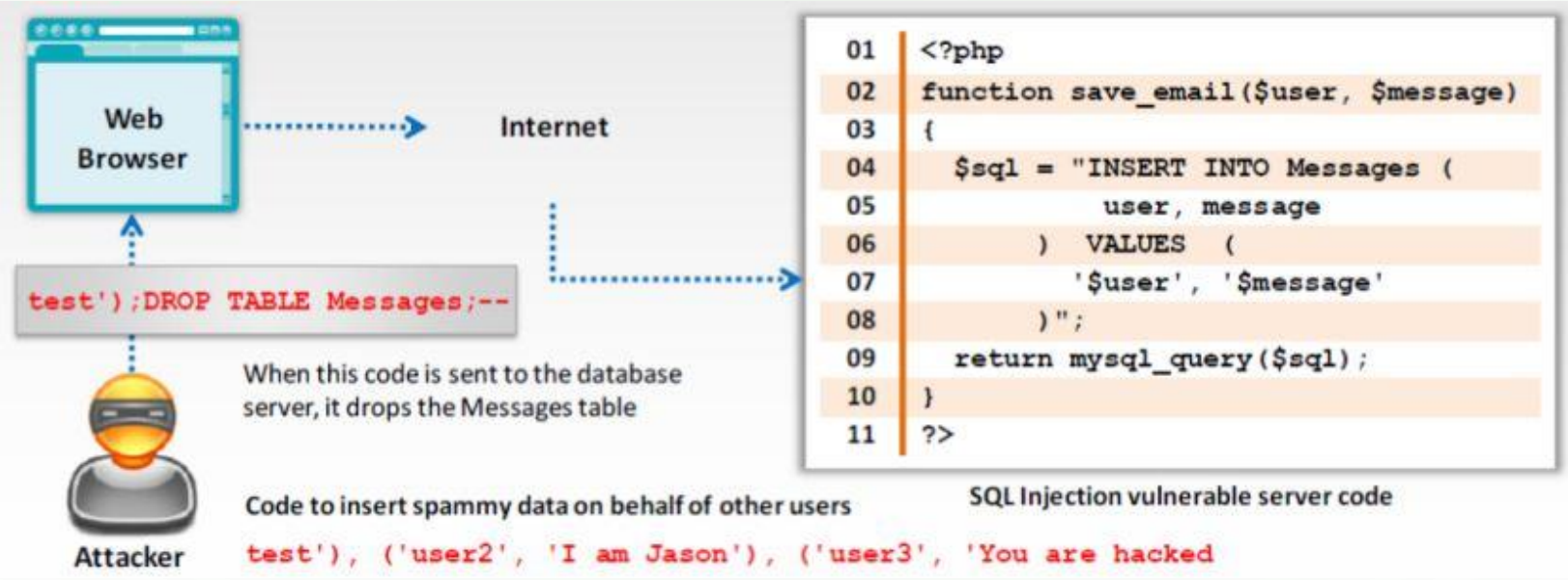
PARAMETER TAMPERING

INJECTION FLAWS

- Refer to vulnerabilities that allow untrusted data to be executed as part of a command or query
- Exploited by attackers through construction of malicious commands or queries
- Example
  - SQL injection – use of malicious SQL queries
  - Command injection – injection of malicious code

INJECTION FLAWS - SQL INJECTION

- Crafting malicious SQL queries to manipulate a database or bypass security measures
- Usually executed from application form fields



INJECTION FLAWS - COMMAND INJECTION

- Passing of malicious code through a web application

Shell Injection	• Crafting an input string to gain shell access to the server
HTML Embedding	• Adding HTML content to deface a website
File Injection	• Upload files such as malicious scripts to a website that is automatically loaded when a page is accessed

INJECTION FLAWS - COMMAND INJECTION

- Attacker injects the URL of a malicious script

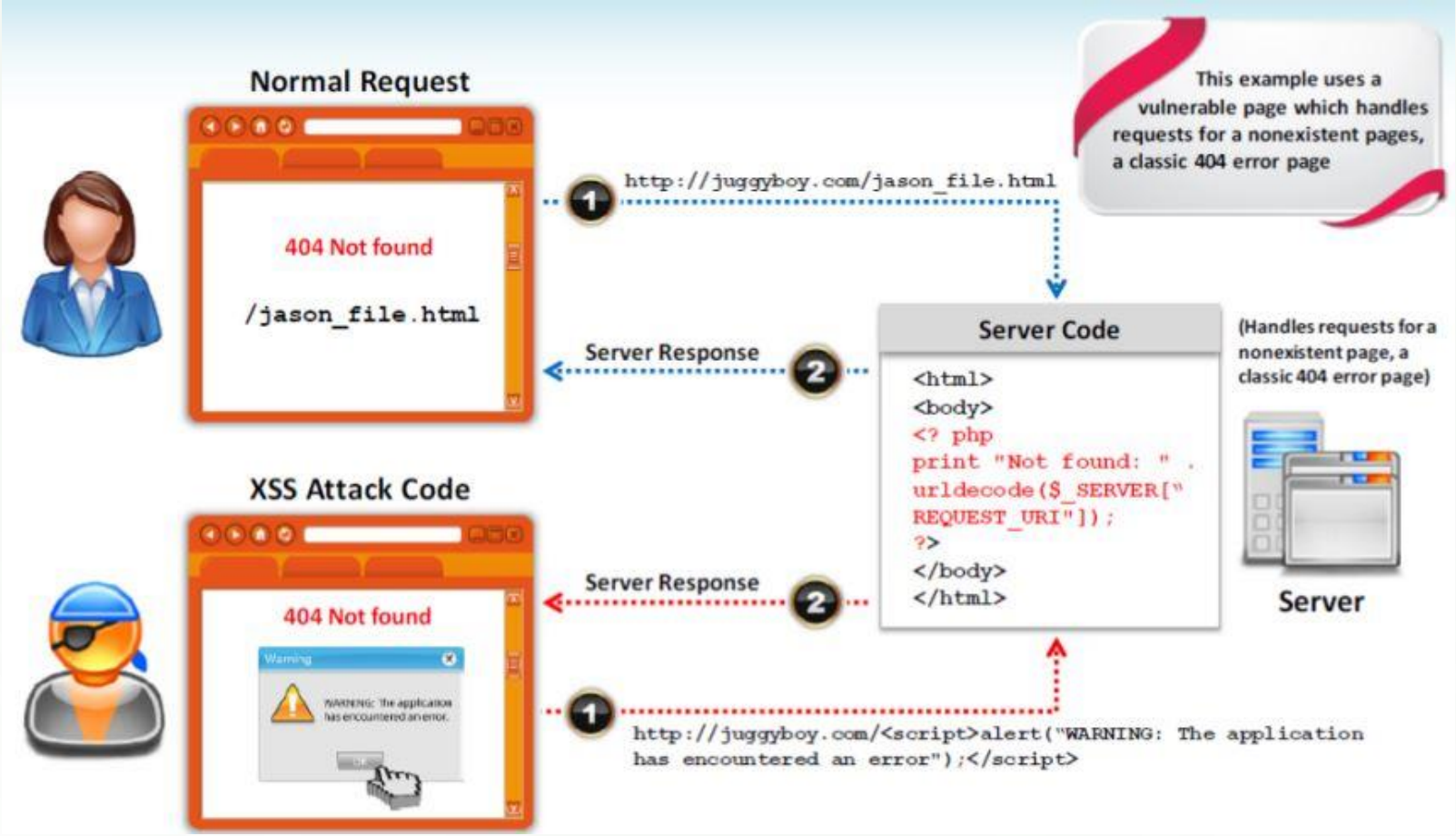
CROSS SITE SCRIPTING (XSS)

- Exploit vulnerabilities in dynamically generated webpages
- Occurs when unvalidated input is included in dynamic content shown on user browsers
- Attacker injects malicious Javascript, VBScript, ActiveX etc.by hiding these as part of legitimate requests

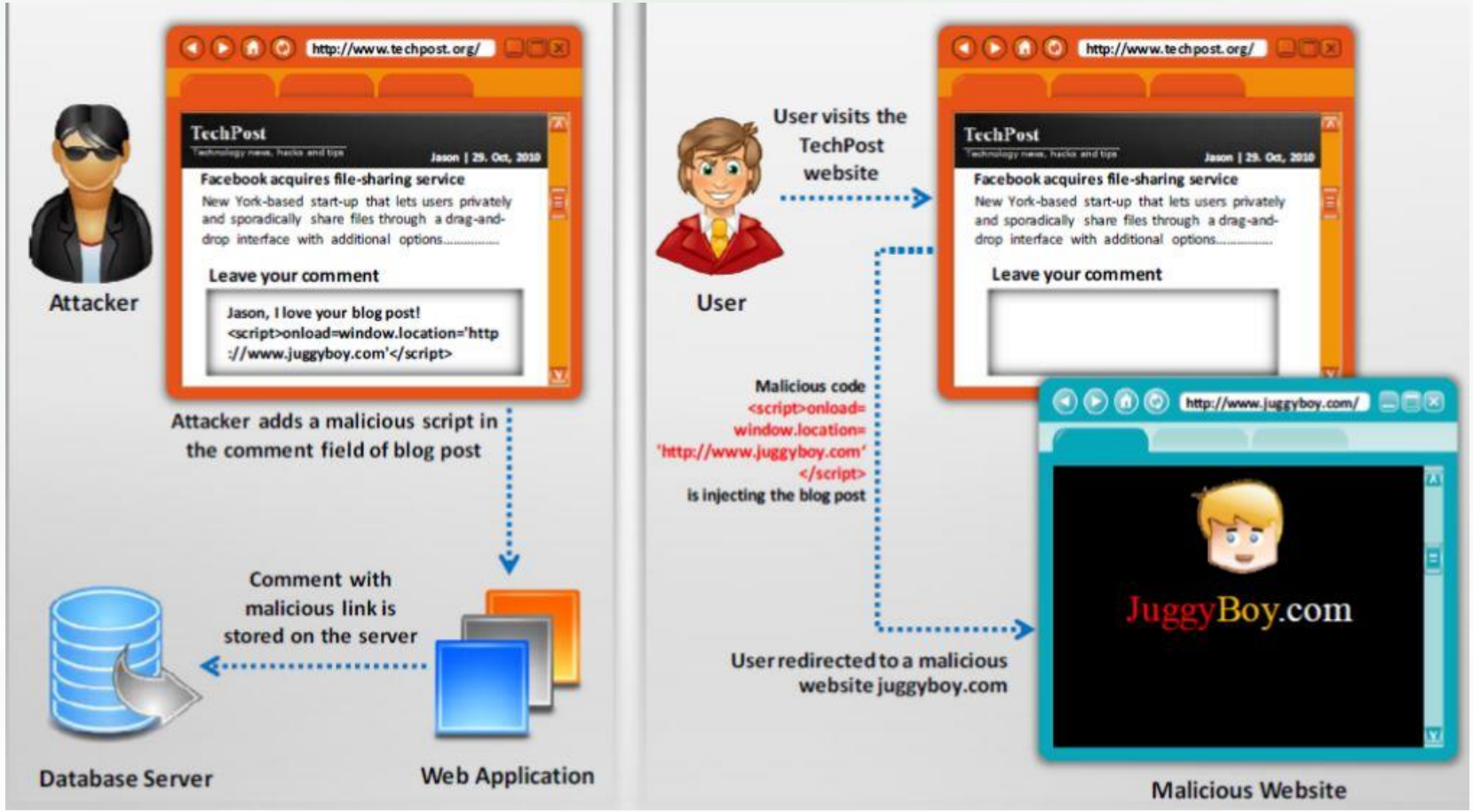
CROSS SITE SCRIPTING (XSS)

- Allows
  - Injecting pop ups
  - Redirection to other pages
  - Stealing cookies to hijack session
  - Data theft

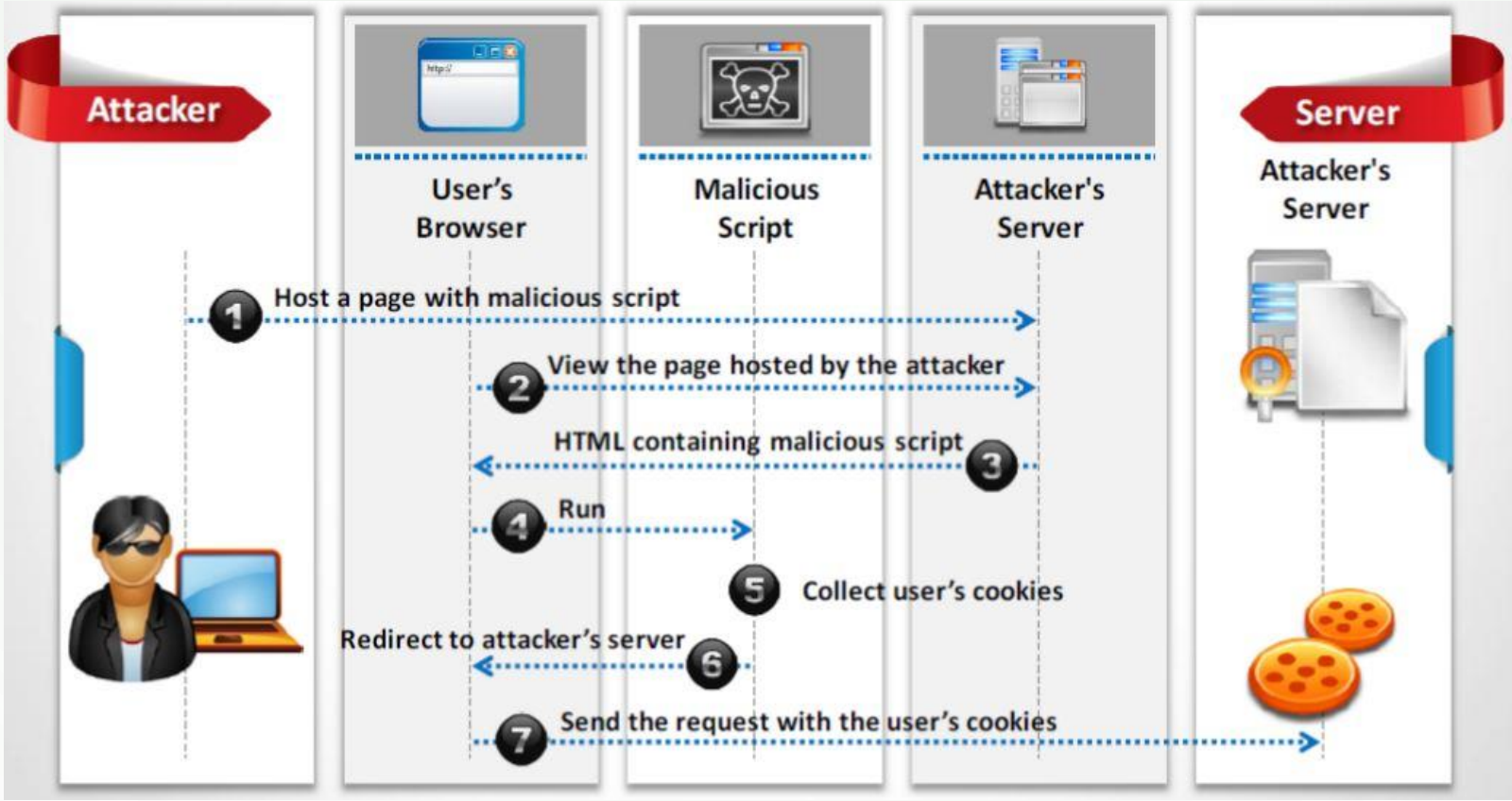
CROSS SITE SCRIPTING (XSS) – A SIMPLE EXAMPLE



CROSS SITE SCRIPTING (XSS) – ANOTHER EXAMPLE



CROSS SITE SCRIPTING (XSS) – YET ANOTHER EXAMPLE



CROSS SITE REQUEST FORGERY (CSRF)

- Exploit vulnerabilities that allow a hacker to force a victim's web browser to send malicious requests on the hacker's behalf using the victim's session

DIRECTORY TRAVERSAL

- Allows hacker to access restricted directories including source code, configuration, and critical files
- Involves manipulation of URLs using '../'
- Ex:
  - http:/some.site.com/home.html/../../../../Users/alice/documents/target.txt

WEB APPLICATION DOS

- Exhaust server resources by generating several resource intensive requests
- Emulate request syntax of legitimate clients so difficult to detect using available DoS detection systems
- Methods
  - Resource starvation – Consume CPU, memory, sockets and bandwidth
  - Exploit programming flaws – Buffer overflows
  - Routing and DNS attack – point to alternate site

WEB APPLICATION DOS – SOME EXAMPLES

- User Registration DoS
  - Create large number of bogus users on an application
- Login Attacks
  - Overload the login process to cause significantly slow response to legitimate users
- User Enumeration
  - Using list of users to check for which are present n the application if it states which between the username or password is incorrect during invalid login
- Lockout DoS
  - Intentionally cause enough failed attempts to lock out multiple user account

BUFFER OVERFLOWS

- Cause an application to write more data into the space allocated for a buffer
- Vulnerable applications are those that don’t check for input length
- Effects
  - Application crash
  - Code execution if function pointers are modified

COOKIE OR SESSION POISONING

- Recall: Cookies are often used to hold sensitive user information or track user sessions

- Attackers can capture user requests containing cookie values and
  - manipulate these values
  - Use the user cookie to authenticate
  - Ride on the user's session by assuming the identity

COOKIE OR SESSION POISONING

GET /index.jsp HTTP/1.1

Host: www.somesite.com

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:36.0) Gecko/20100101 Firefox/36.0

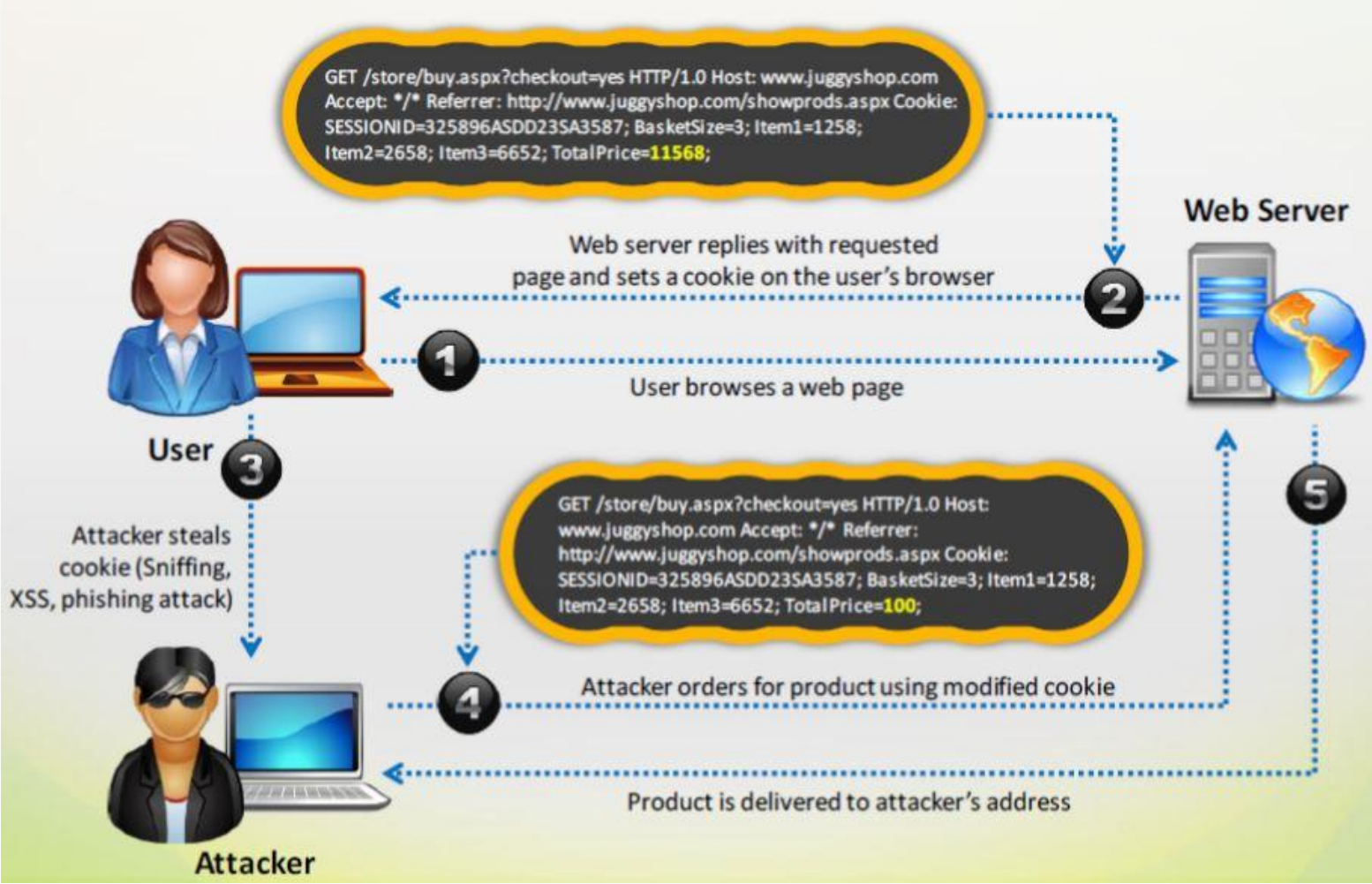
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Cookie: option1=12; JSESSIONID=aaa_RmJlc8o_Yj

COOKIE OR SESSION POISONING



SESSION FIXATION

- Attacker tricks victim into accessing a website using an explicit session ID value

BROKEN AUTHENTICATION AND SESSION MANAGEMENT

- Attacker can use vulnerabilities in authentication and session management functions to impersonate users
- Examples:
  - Plaintext Session IDs in URLs or in HTTP request fields – prone to sniffing
  - Password exploitation – Passwords stored in DB as plaintext
  - Timeout exploitation – session is not timed out when user does not logout properly

SECURITY MISCONFIGURATION

- Includes any form of errors in server configuration that leads to the presence of a vulnerability that can lead to unauthorized access
- Examples
  - Enabled admin console
  - Presence of default accounts
  - Unpatched web server software
  - Enabled unnecessary services

IMPROPER ERROR HANDLING

- Leads to exposure of source code or vulnerability identification from error messages
- Information that can be gathered:
  - Directory hierarchies
  - Stack traces
  - Database information
  - Web server information

IMPROPER ERROR HANDLING

HTTP Status 500 -

type Exception report

message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

```
com.potix.zk.ui.UiException: Recursive import: /test/import.zul
        com.potix.zk.ui.metainfo.Parser.pa
        com.potix.zk.ui.metainfo.Parser.pa
        com.potix.zk.ui.metainfo.PageDefini
        com.potix.web.util.resource.Resourc
        com.potix.util.resource.ResourceCac
        com.potix.util.resource.ResourceCac
        com.potix.util.resource.ResourceCac
```

Object reference not set to an instance of an object. - Windows Internet Explorer

http://warp.senecac.on.ca/bti420_111a01/examples/Week_02_1_Erro

Favorites | Object reference not set to an instance of an obje... | Page | Safety | Tools

Server Error in '/bti420_111a01' Application.

Object reference not set to an instance of an object.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.NullReferenceException: Object reference not set to an instance of an object.

Source Error:

```
The source code that generated this unhandled exception can only be shown when compiled in debug mode.
To enable this, please follow one of the below steps, then request the URL:

1. Add a "Debug=true" directive at the top of the file that generated the error. Example:

  <%@ Page Language="C#" Debug="true" %>

or:

2) Add the following section to the configuration file of your application:
```

COUNTERMEASURES – FOR THE WEB APPLICATION PROGRAMMER

- Limit user input length
- Validate input including hidden fields for data type correctness and presence of special characters
- Sanitize input sent to database
- Use parameterized queries (i.e. prepared statements)
- Use low privileged account for DB connection

COUNTERMEASURES – FOR THE WEB APPLICATION PROGRAMMER

- Disable the admin interface
- Implement a session timeout after certain time of inactivity
- Use a custom error page
- Set secure flag on sensitive cookies so that they are encrypted
- Store sensitive info in hashed form

COUNTERMEASURES – FOR THE WEB APPLICATION USER

- Avoid using the ‘remember me’ option
- Always logout. Don’t just close your browser
- Clear browsing history