UNIVERSITAS INDONESIA

FAKULTAS ILMU KOMPUTER

03. Security, Authentication, Session

Layanan & Aplikasi Web

Outline

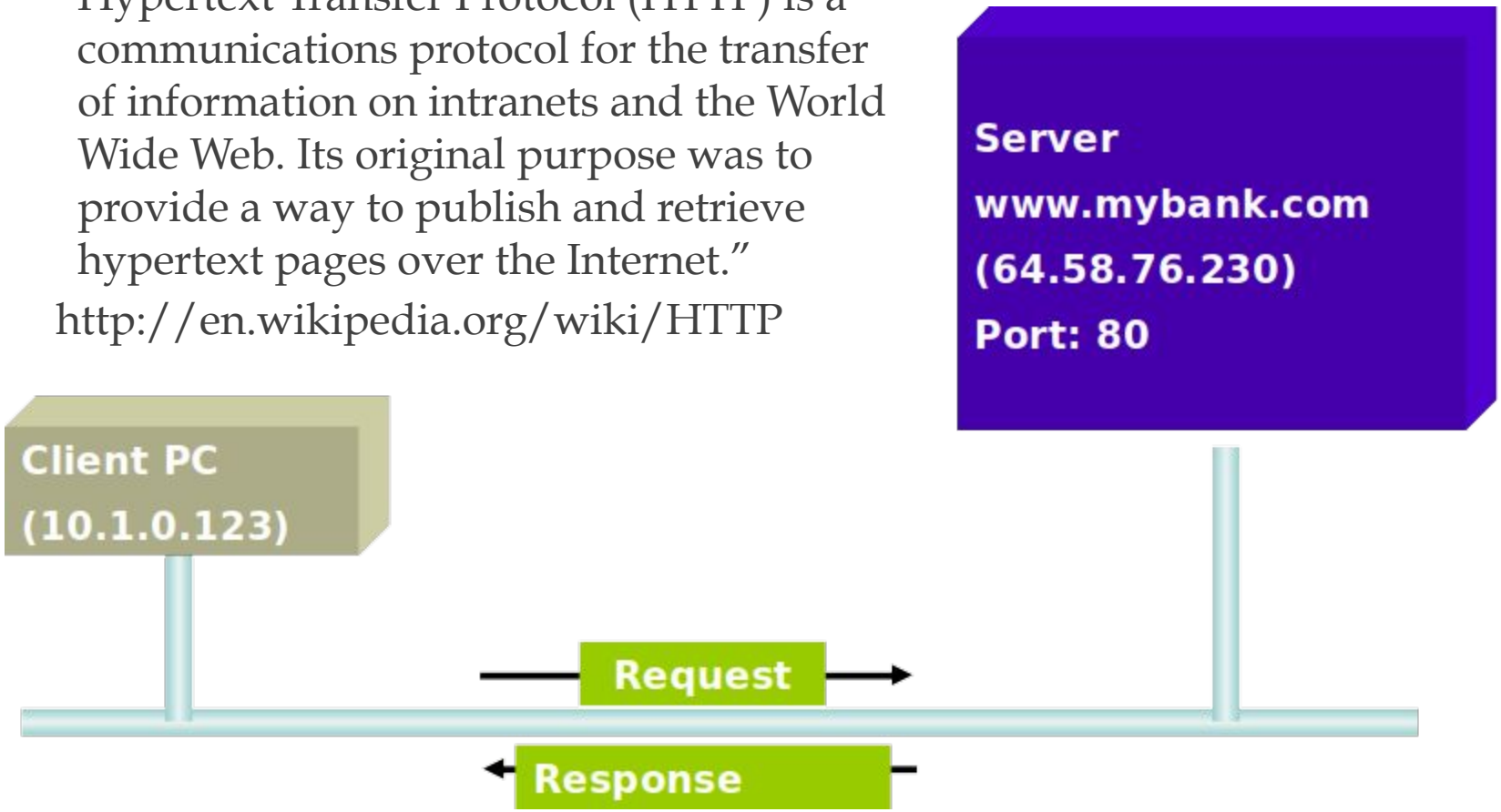
- Introduction
- Web Application Vulnerabilities
- Securing Web Application
- Authentication
- Session

HTTP Protocol

Hypertext Transfer Protocol

"Hypertext Transfer Protocol (HTTP) is a communications protocol for the transfer of information on intranets and the World Wide Web. Its original purpose was to provide a way to publish and retrieve hypertext pages over the Internet."

http://en.wikipedia.org/wiki/HTTP



HTTP Request - GET

- Form data encoded in the URL
- Most common HTTP method used on the web
- Should be used to retrieve information, not for actions that have side-effects

Simple Request with GET arguments


```
GET http://www.mysite.com/kgsearch/search.php?catid=1 HTTP/1.1
Host: www.mysite.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.13)
Gecko/20080311 Firefox/2.0.0.13
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*
;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.mysite.com/
```

Simple Request with lots of GET arguments

http://www.google.com/search?hl=en&lr=&c2coff=1&rls=GGLG%2CGGLG%3A2005-26%2CGGLG%3Aen&q=http%3A%2F%2Fwww.google.com%2Fsearch%3Fhl%3Den%26lr%3D%26c2coff%3D1%26rls%3DGGLG%252CGGLG%253A2005-26%252CGGLG%253Aen%26q%3Dhttp%253A%252F%252Fwww.google.com%252Fsearch%253Fhl%253Den%2526lr%253D%2526c2coff%253D1%2526rls%253DGGLG%25252CGGLG%25253A2005-26%25252CGGLG%25253Aen%2526q%253Dhttp%25253A%25252F%25252Fwww.google.com%25252Fsearch%25253Fsourceid%25253Dnavclient%252526ie%25253DUTF-8%252526rls%25253DGGLG%25252CGGLG%25253A2005-26%25252CGGLG%25253Aen%252526q%25253Dhttp%2525253A%2525252F%2525252Fwww%2525252Egoogle%2525252Ecom%2525252Fsearch%2525253Fsourceid%2525253Dnavclient%25252526ie%2525253DUTF%2525252D8%25252526rls%2525253DGGLG%2525252CGGLG%2525253A2005%2525252D26%2525252CGGLG%2525253Aen%25252526q%2525253Dhttp%252525253A%252525252F%252525252Fuk2%252525252Emultimap%252525252Ecom%252525252Fmap%252525252Fbrowse%252525252Ecgi%252525253Fclient%252525253Dpublic%2525252526GridE%252525253D%252525252D0%252525252E12640%2525252526GridN%252525253D51%252525252E50860%2525252526lon%2525252526lon%252525253D%252525252D0%252525252E12640%2525252526lat%252525253D51%252525252E50860%2525252526search%252525255Fresult%252525253DLondon%25252525252CGreater%252525252520London%2525252526db%252525253Dfreegaz%2525252526cidr%252525255Fclient%252525253Dnone%2525252526lang%2525252526D%2525252526place%252525253DLondon%252525252CGreater%252525252BLondon%2525252526pc%252525253D%2525252526advanced%252525253D%2525252526client%252525253Dpublic%2525252526addr2%252525253D%2525252526quicksearch%252525253DLondon%2525252526addr3%252525253D%2525252526scale%252525253D100000%2525252526addr1%252525253D%2526btnG%253DSearch%26btnG%3DSearch&btnG=Search

Note:

MAX 4096 chars (firefox/chrome), 511 chars in IE

HTTP Request - POST

- Data is included in the body of the request.
- Should be used for any action that has side-effects
  - Storing/updating data, ordering a product, etc…

HTTP Request - POST


```
POST http://www.mysite.com/kgsearch/search.php HTTP/1.1
Host: www.mysite.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.13) Gecko/20080311
Firefox/2.0.0.13
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.mysite.com/

catid=1
```

It's always there ...

"Every program has at least two purposes: the one for which it was written, and another for which it wasn't."

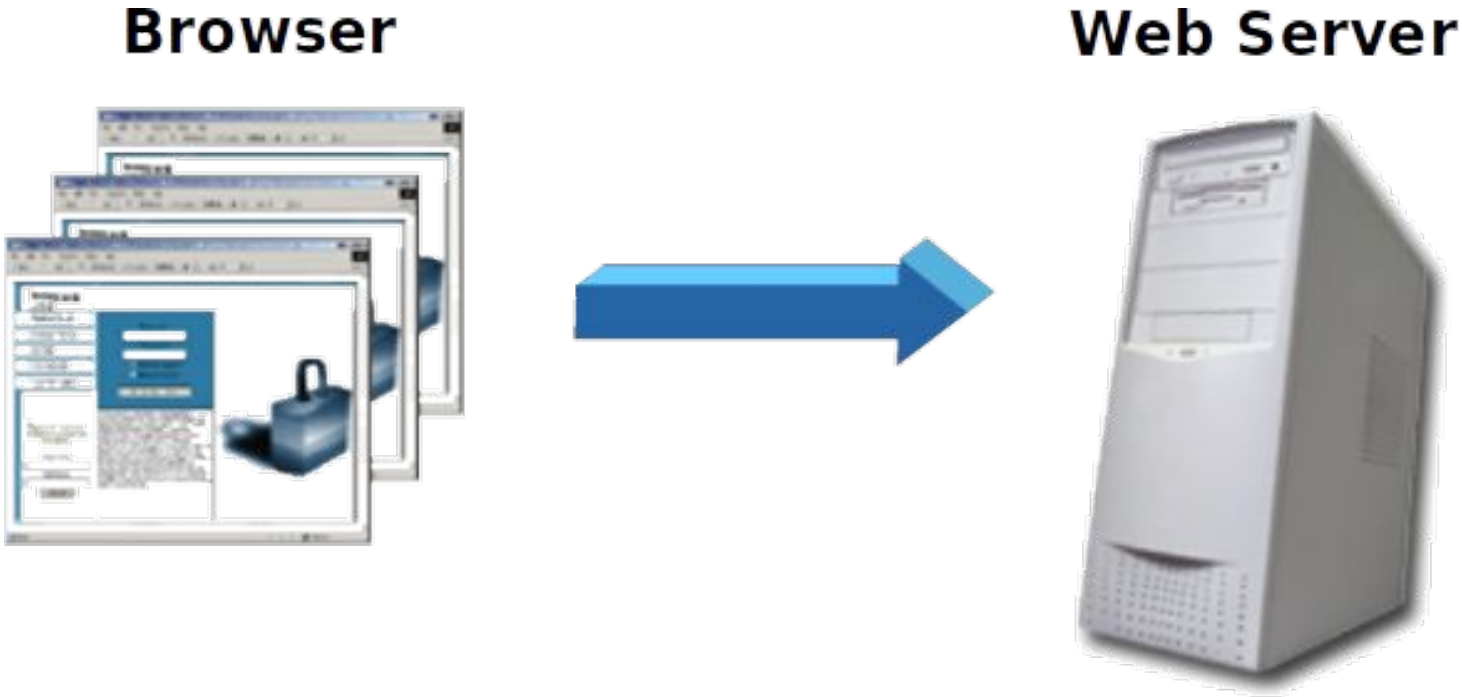
-Alan J. Perlis

GET vs POST Security

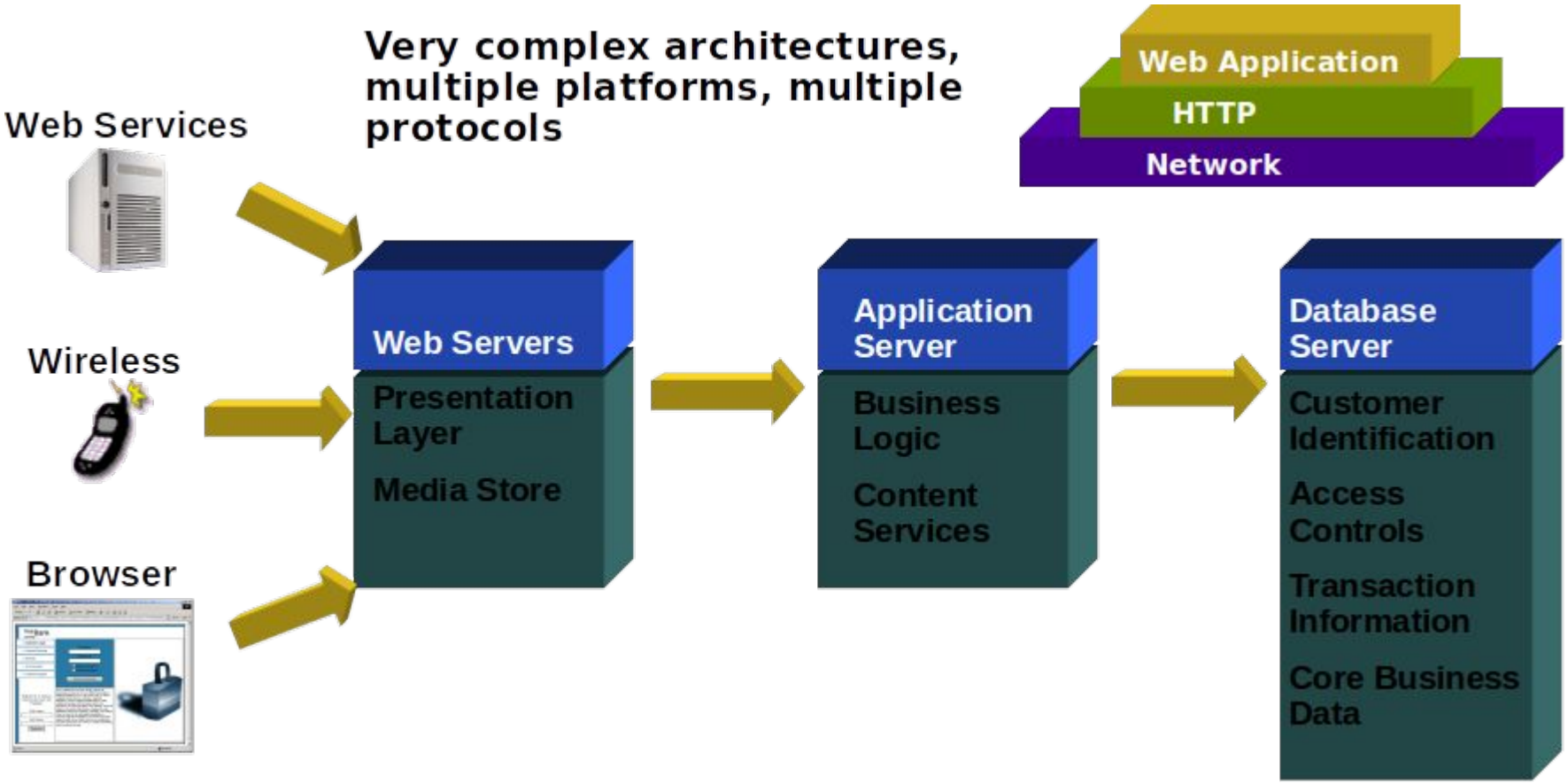
- There information contained in parameters can tell a user a lot about how your application works
- GET parameters are easily visible in the address bar
- POST parameters are hidden from the average user
  - Users can still view source code
  - Users can still view the packets
  - Users can still intercept & modify web requests

Websites (Static)

- No applications
- Static pages
- Hard coded links



Web Applications



Why Web Application Vulnerabilities Occur

Security Professionals Don't Know The Applications

"As a Network Security Professional, I don't know how my companies web applications are supposed to work so I deploy a protective solution…but don't know if it's protecting what it's supposed to."

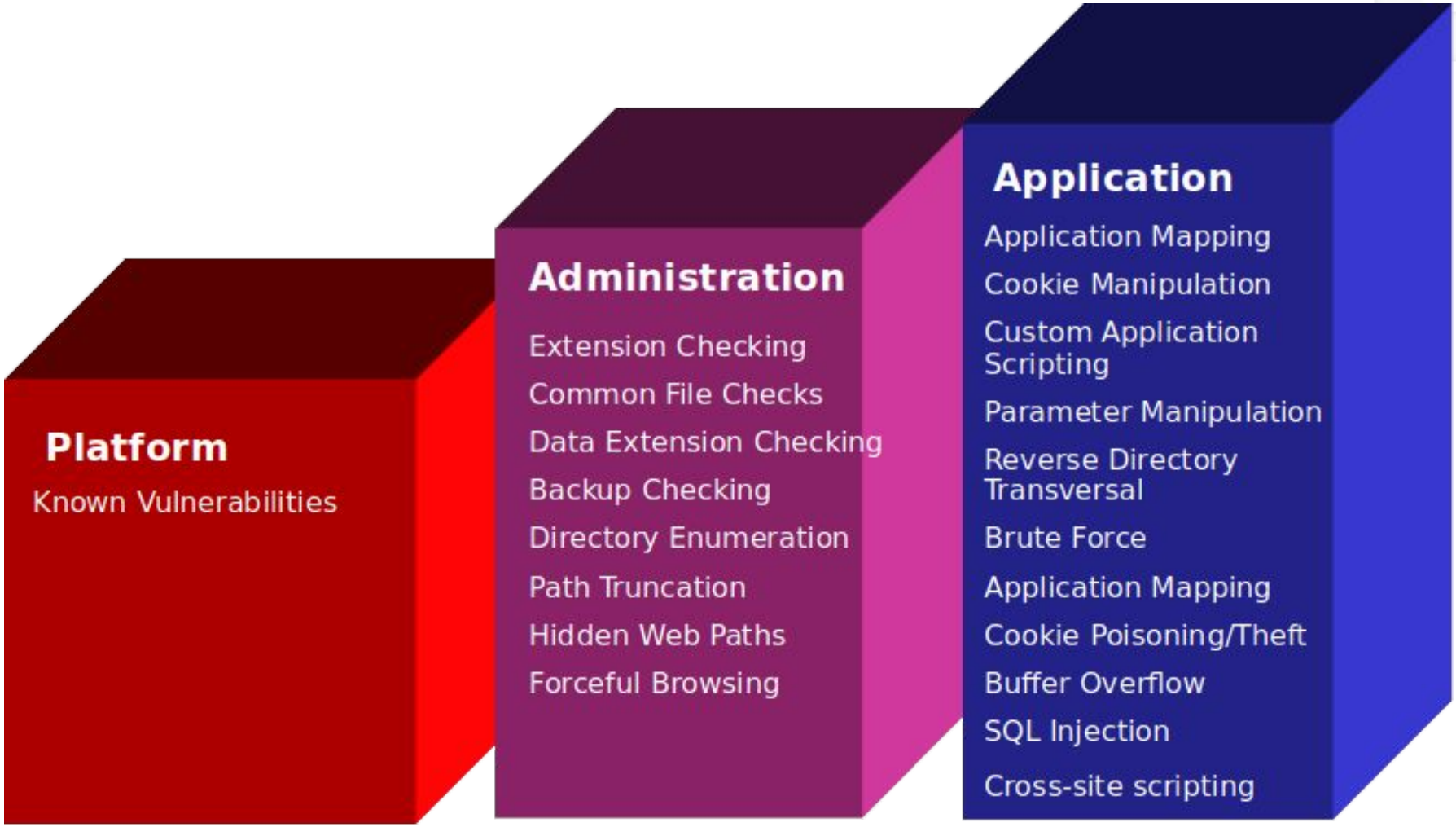
Application Developers and QA Professionals Don't Know Security

"As an Application Developer, I can build great features and functions while meeting deadlines, but I don't know how to develop my web application with security as a feature."

Web Application Vulnerabilities

- Technical Vulnerabilities
  - Result of insecure programming techniques
  - Mitigation requires code changes
  - Detectable by scanners
  - http://example/order.asp?item=<script>alert('p0wned')</script>&price=300.00
- Logical Vulnerabilities
  - Result of insecure program logic
  - Most often to due to poor decisions regarding trust
  - Mitigation often requires design/architecture changes
  - Detection often requires humans to understand the context
  - http://example/order.asp?item=toaster&price=30.00

Web Application Vulnerabilities



Web Application Vulnerabilities

Platform:

- Known vulnerabilities can be exploited immediately with a minimum amount of skill or experience – "script kiddies"
- Most easily defendable of all web vulnerabilities
- MUST have streamlined patching procedures

Web Application Vulnerabilities

Administration:

- Less easily corrected than known issues
- Require increased awareness
- More than just configuration, must be aware of security flaws in actual content
- Remnant files can reveal applications and versions in use
- Backup files can reveal source code and database connection strings

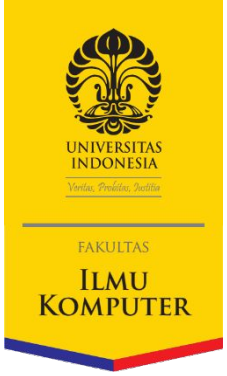
Web Application Vulnerabilities

Application Programming:

- Common coding techniques do not necessarily include security
- Input is assumed to be valid, but not tested
- Unexamined input from a browser can inject scripts into page for replay against later visitors
- Unhandled error messages reveal application and database structures
- Unchecked database calls can be 'piggybacked' with a hacker's own database call, giving direct access to business data through a web browser

How to Secure Web Applications

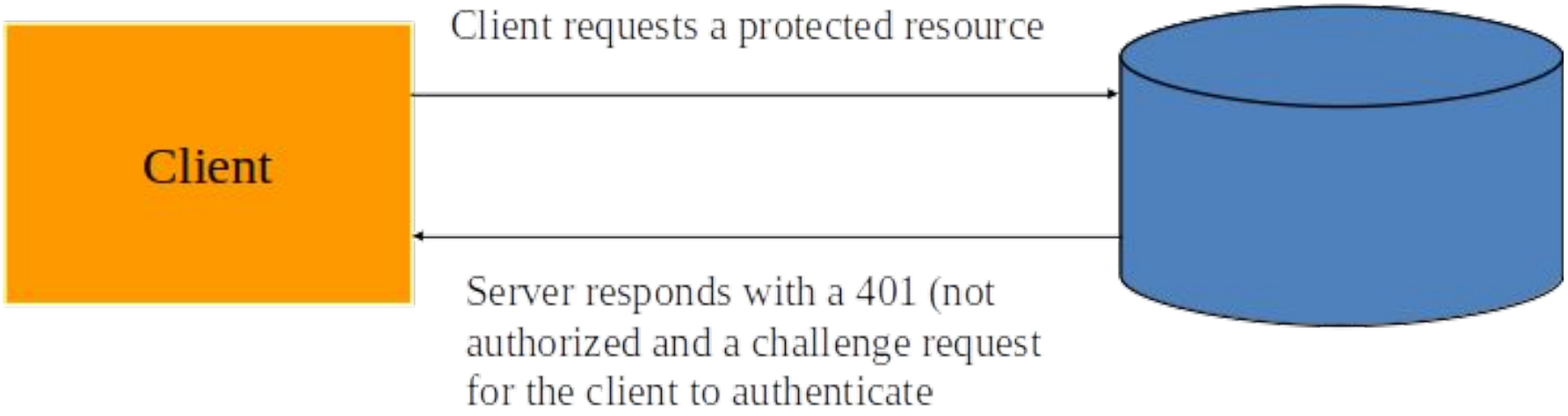
- Incorporate security into the lifecycle
  - Apply information security principles to all software development efforts
- Educate
  - Issue awareness, Training, etc…



Authentication

Web Authentication

- Protect web content from those who don't have a "need to know"
- Require users to authenticate using a userid/password before they are allowed access to certain URLs
- HTTP/1.1 requires that when a user makes a request for a protected resource the server responds with a authentication request header
- WWW-Authenticate
- contains enough pertinent information to carry out a "challenge-response" session between the user and the server



Client Response

- Well established clients like Firefox, Internet Explorer will respond to the challenge request (WWW-Authenticate) by presenting the user with a small pop-up window with data entry fields for
  - userid
  - password
  - a Submit button and a Cancel button
- Entering a valid userid and password will post the data to the server, the server will attempt authentication and if authenticated will serve the originally requested resource.

WWW-Authenticate

- The authentication request received by the browser will look something like:
  - WWW-Authenticate = Basic realm="defaultRealm"
    - Basic indicates the HTTP Basic authentication is requested
    - realm indicates the context of the login
      - realms hold all of the parts of security puzzle:
        - Users
        - Groups
        - ACLs (Access Control Lists)
  - Basic Authentication
    - userid and password are sent base 64 encoded (might as well be plain text)
    - hacker doesn't even need to unencode all he has to do is "replay" the blob of information he stole over and over (this is called a "replay attack")

WWW-Authenticate

- Digest Authentication
  - attempts to overcome the shortcomings of Basic Authentication
  - WWW-Authenticate = Digest realm="defaultRealm" nonce="Server SpecificString"
  - see RFC 2069 for description of nonce, each nonce is different
  - the nonce is used in the browser in a 1-way function (MD5, SHA-1….) to encode the userid and password for the server, this function essentially makes the password good for only one time
- Common browsers don't use Digest Authentication but an applet could as an applet has access to all of the Java Encryption classes needed to create the creation of a Digest.

HTTP Security

- Secure Sockets Layer (SSL)
  - Invented by Netscape and made public domain for everyone's use
  - An additional layer to the TCP/IP stack that sits between the Application and Transport layers
    - ensures that all application data is encrypted but TCP/IP headers are not
    - usually run on port 443 (default HTTPS port)
- Public Key Cryptography
  - owner of a private key sends a public key to all who want to communicate with him (keys are both prime factors of a large (1024 bit) number). Owner keeps the private key secret and uses it to decrypt information sent to him that has been encrypted with the public-key
  - RSA algorithm is most notable public-key cipher algorithm
- Digital Certificates
  - issued by a disinterested third party (ex. Verisign)
  - the Certificate contains the public-key for the specific Web Server and a digital signature of the certifying authority

Secure Socket Layer (SSL)

- Once a secure session is established the source requests the destinations certificate (sent in the http header (uncncrypted)) once the source accepts the authenticity of the certificate it uses the public-key from the certificate to encrypt the generated session key for protecting the conversation between the source and destination.
- Session is encrypted using asymmetric cipher (slow)
- Conversation is encrypted using a symmetric cipher (fast)
- It's done this way to speed up overall communications, asymmetric cipher is used as little as possible while symmetric encryption is used for most exchanges actual cipher algorithms are negotiated on a per-session basis

UNIVERSITAS INDONESIA

FAKULTAS ILMU KOMPUTER

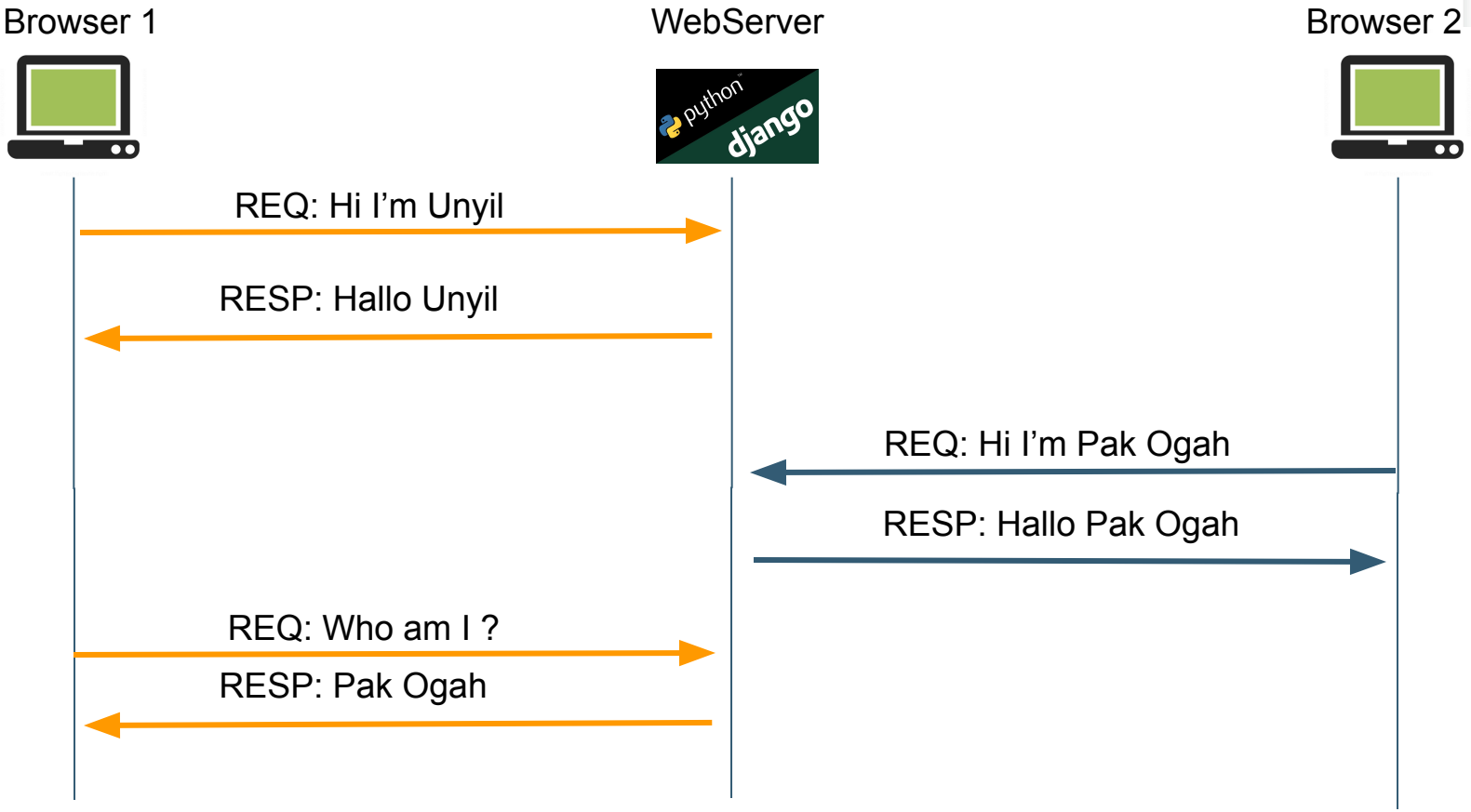
Session

Session

- Session: an abstract concept to represent a series of HTTP request and responses between a specific Web browser and server
  - HTTP doesn't support the notion of a session

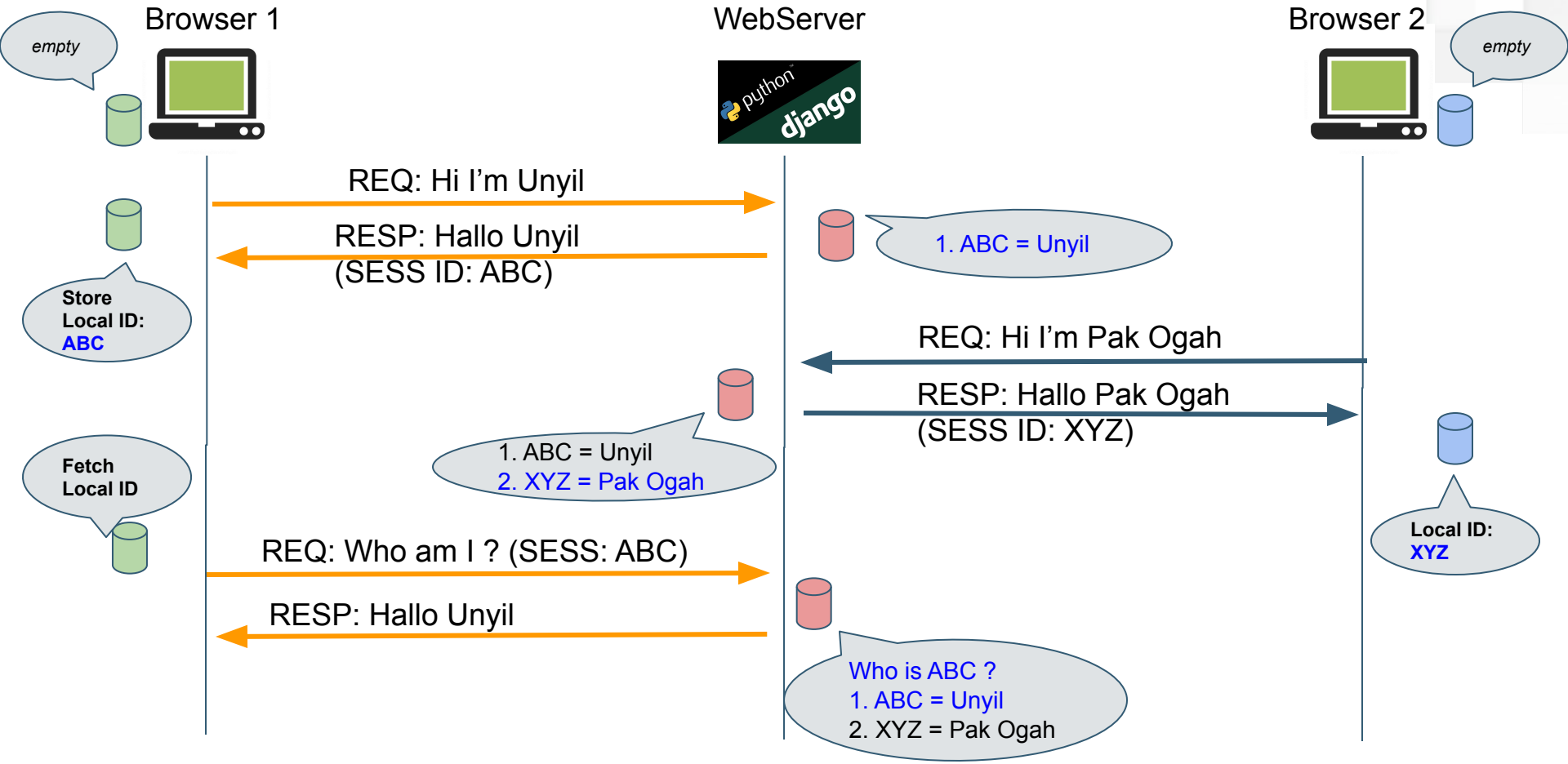
- How to implement Session concept?
  - Implement some code in ServerSide and ClientSide programming
    - ServerSide programming using **Server Session Database**
    - ClientSide programming using **Cookie** or **LocalStorage**
  - Server Session and Cookies need to work together to keep the session alive

Why a webserver need to handle sessions? Why a web user needs to keep their session?



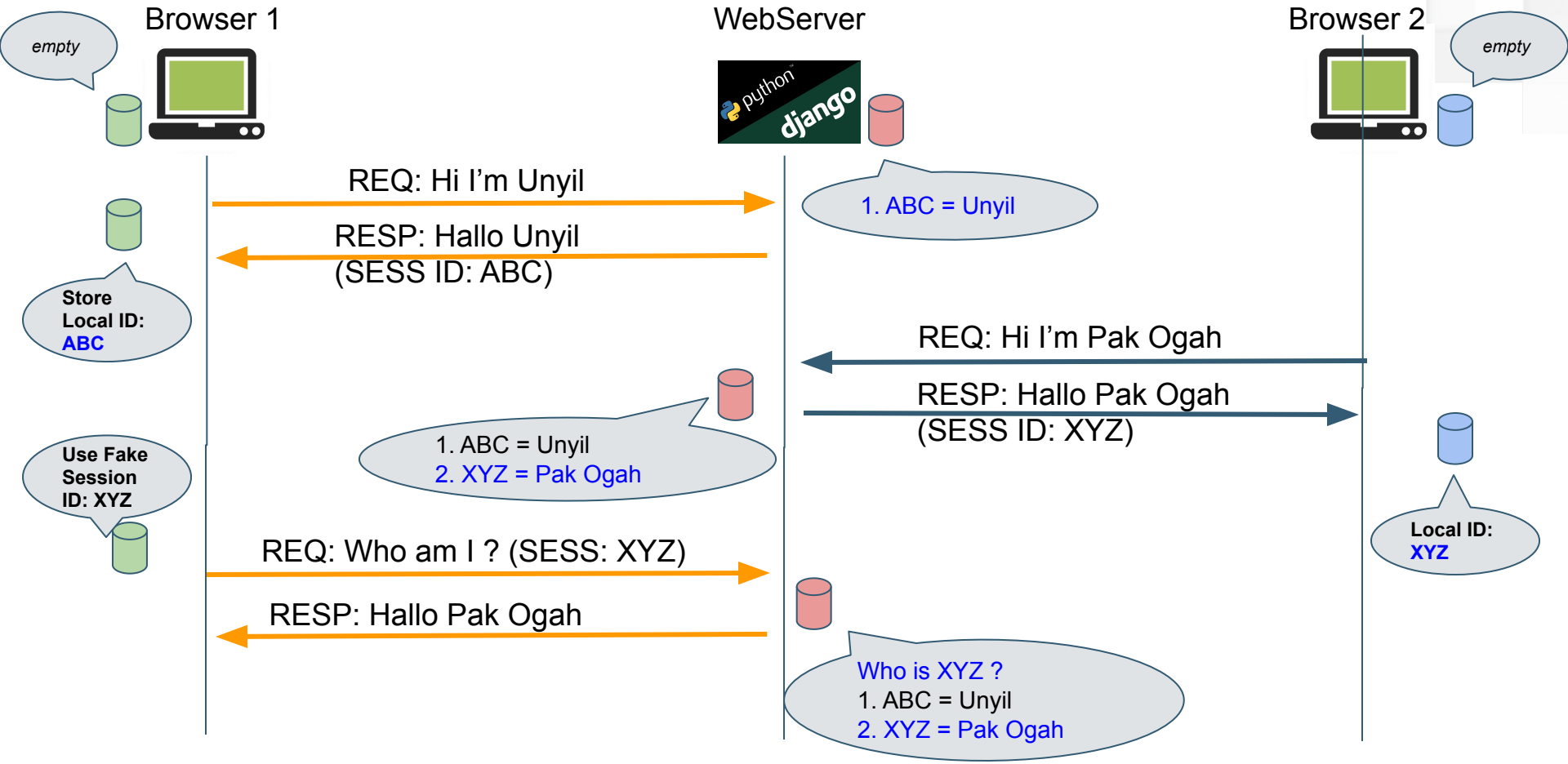
A World **WITHOUT** Session

Why a webserver need to handle sessions? Why a web user needs to keep their session?



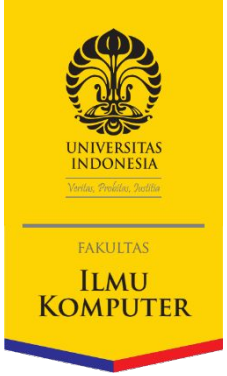
A World **WITH** Session

Why a webserver need to handle sessions? Why a web user needs to keep their session?



Using Fake Session ID (*Session forgery*)

NB: Do not try this at anywhere



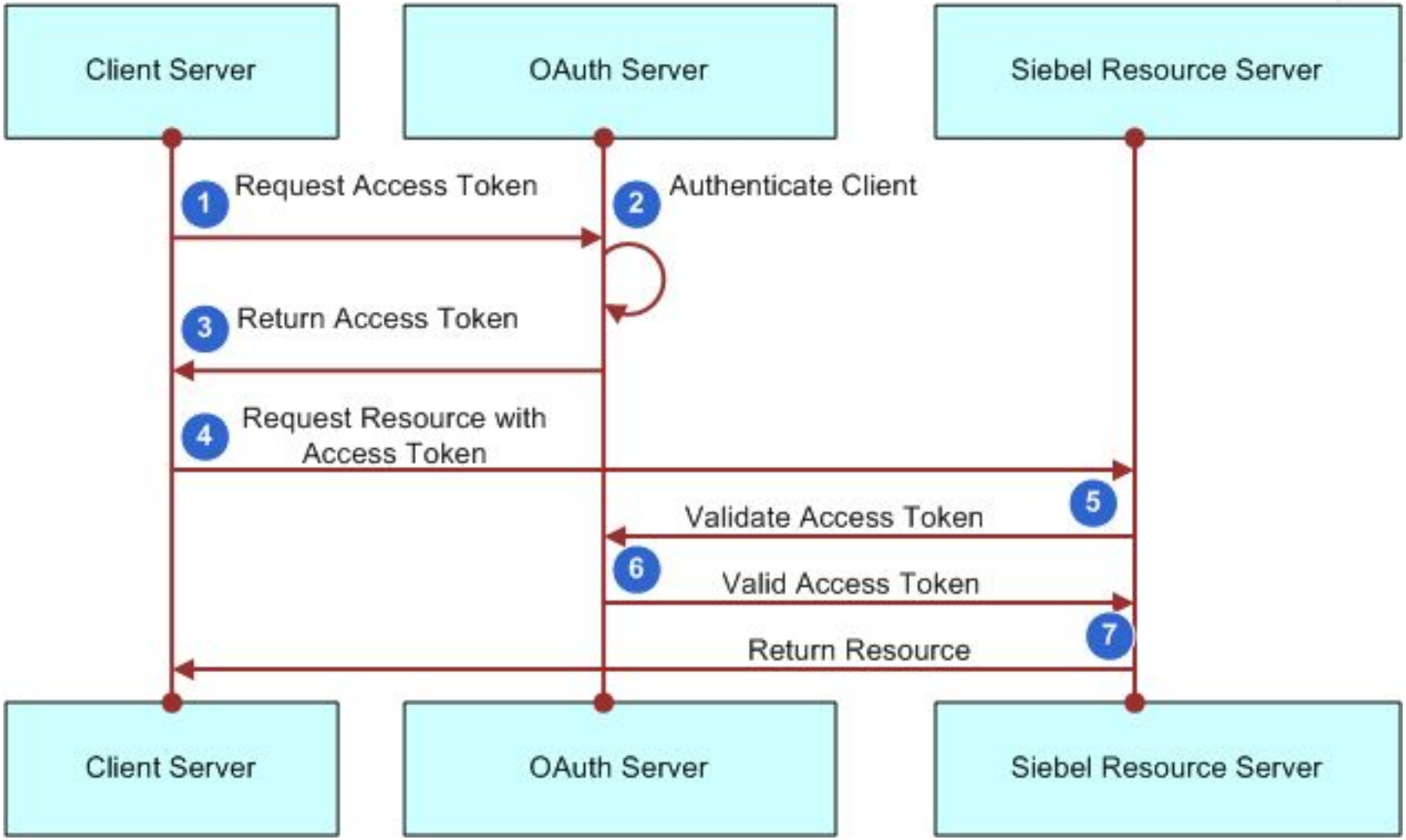
OAuth 2.0 Framework

OAuth 2.0 Framework

RFC 6749 - The OAuth 2.0 Authorization Framework

RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage

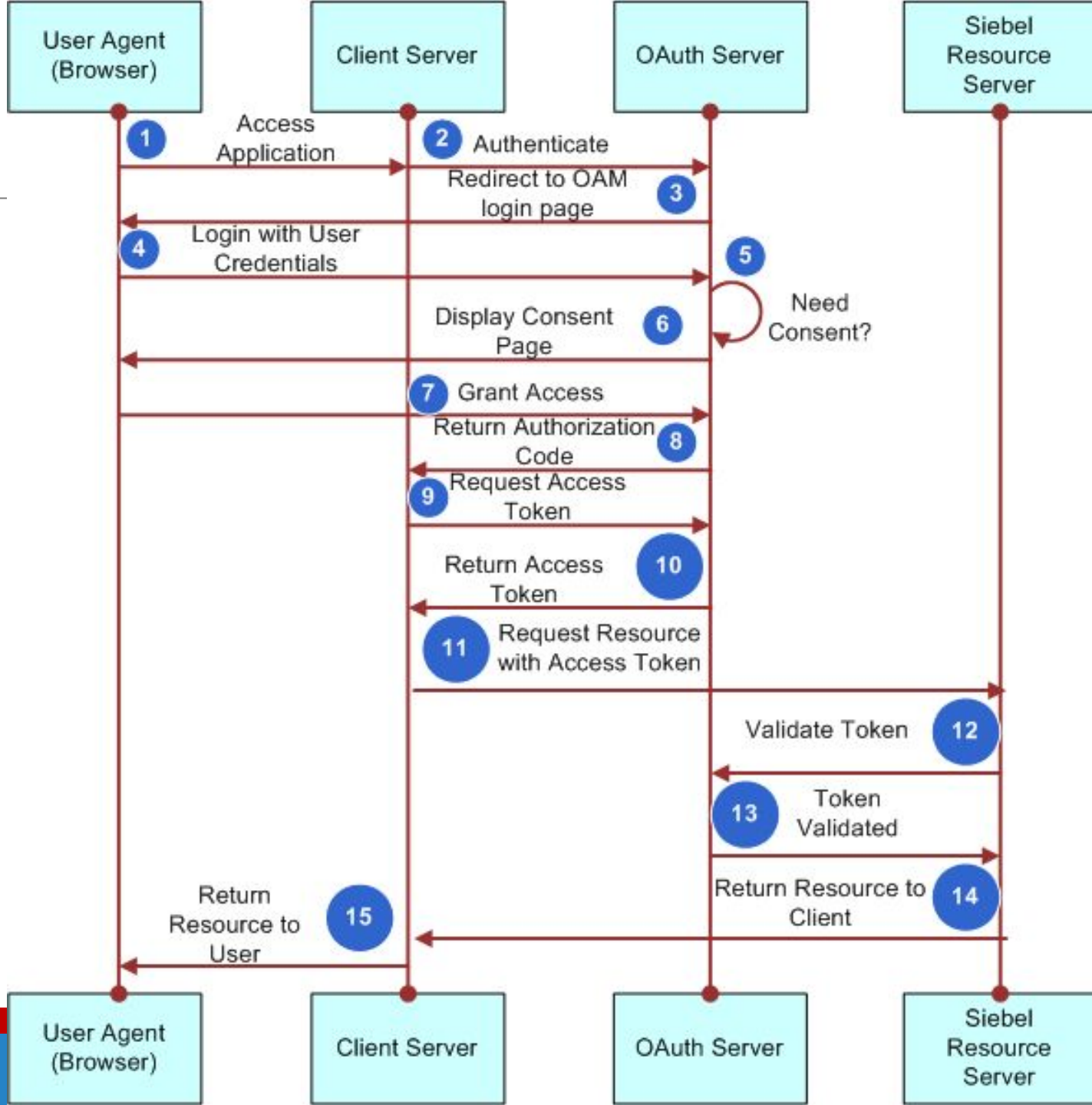
OAuth 2.0 Framework (Client Credentials Grant Auth Flow)



Source: Oracle

OAuth 2.0 Framework

Web Server Auth Flow)



OAuth

Reference:

https://pfelix.files.wordpress.com/2012/11/codebits12-oauth2-slides.pdf