Introduction to Network Security

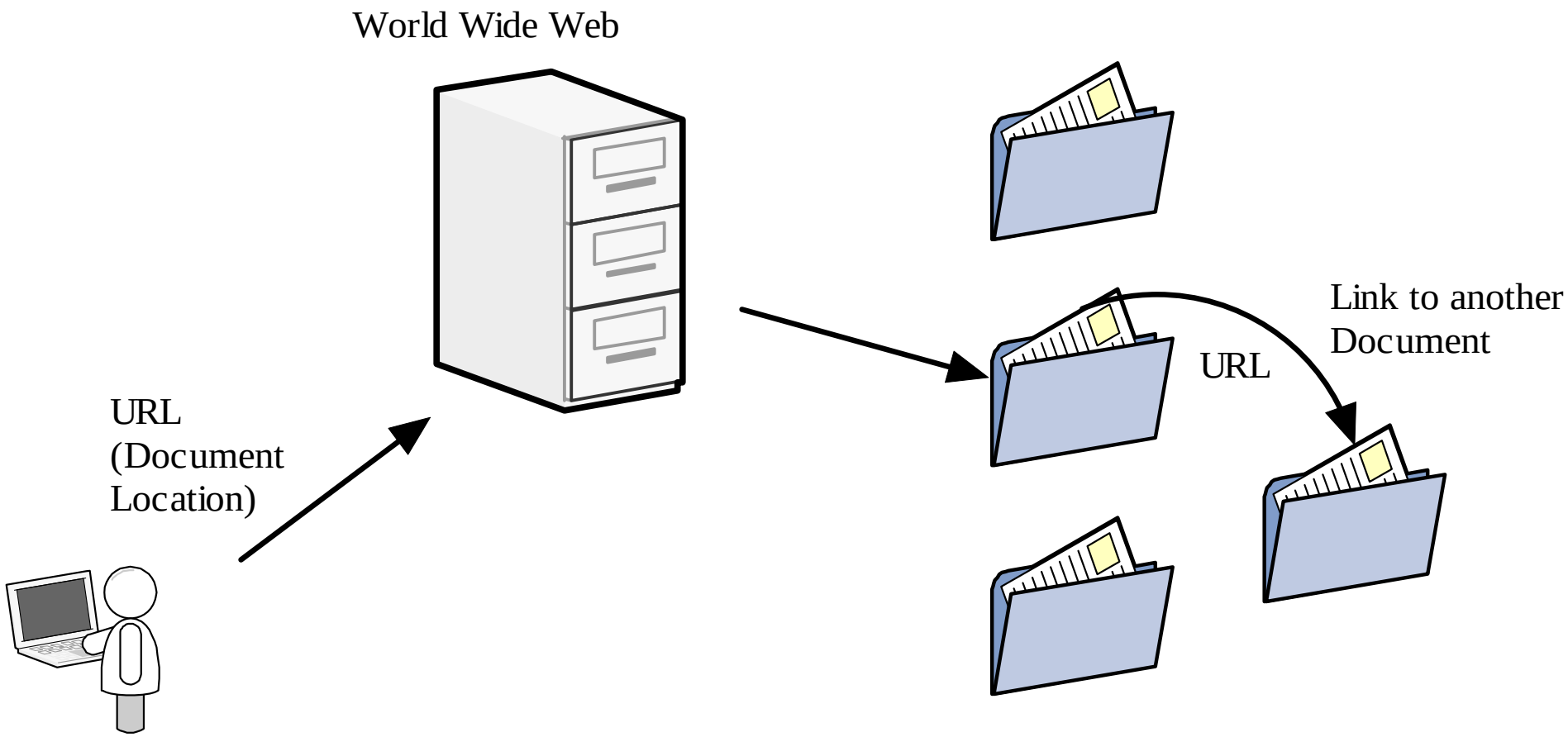
Chapter 10

Web Security

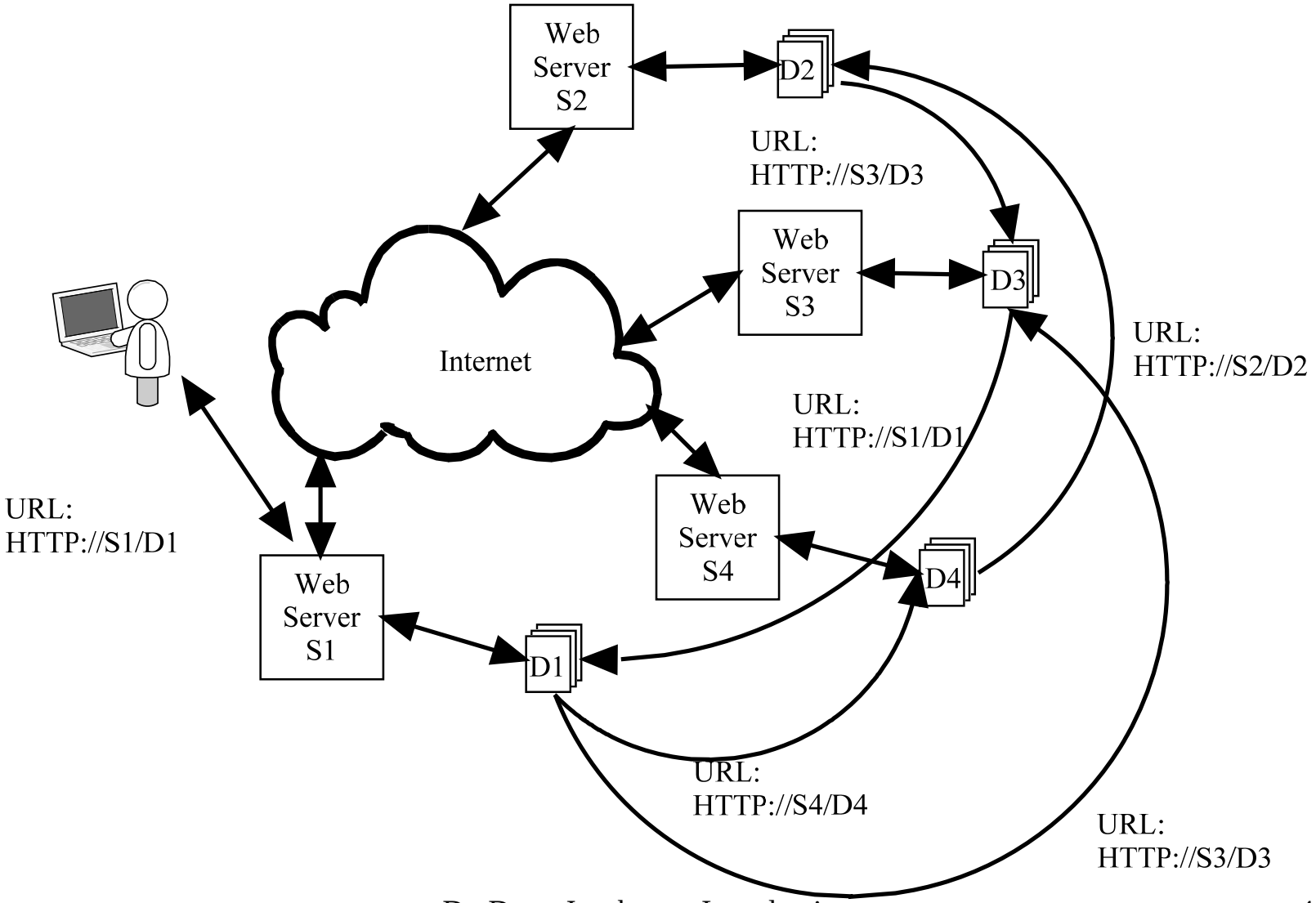
Topics

- WWW
- HTTP: Hyper Text Transfer Protocol
- HTTP Security
- HTML Protocol
- HTML Security
- Server Side Security
- Client Side security
- General Countermeasures

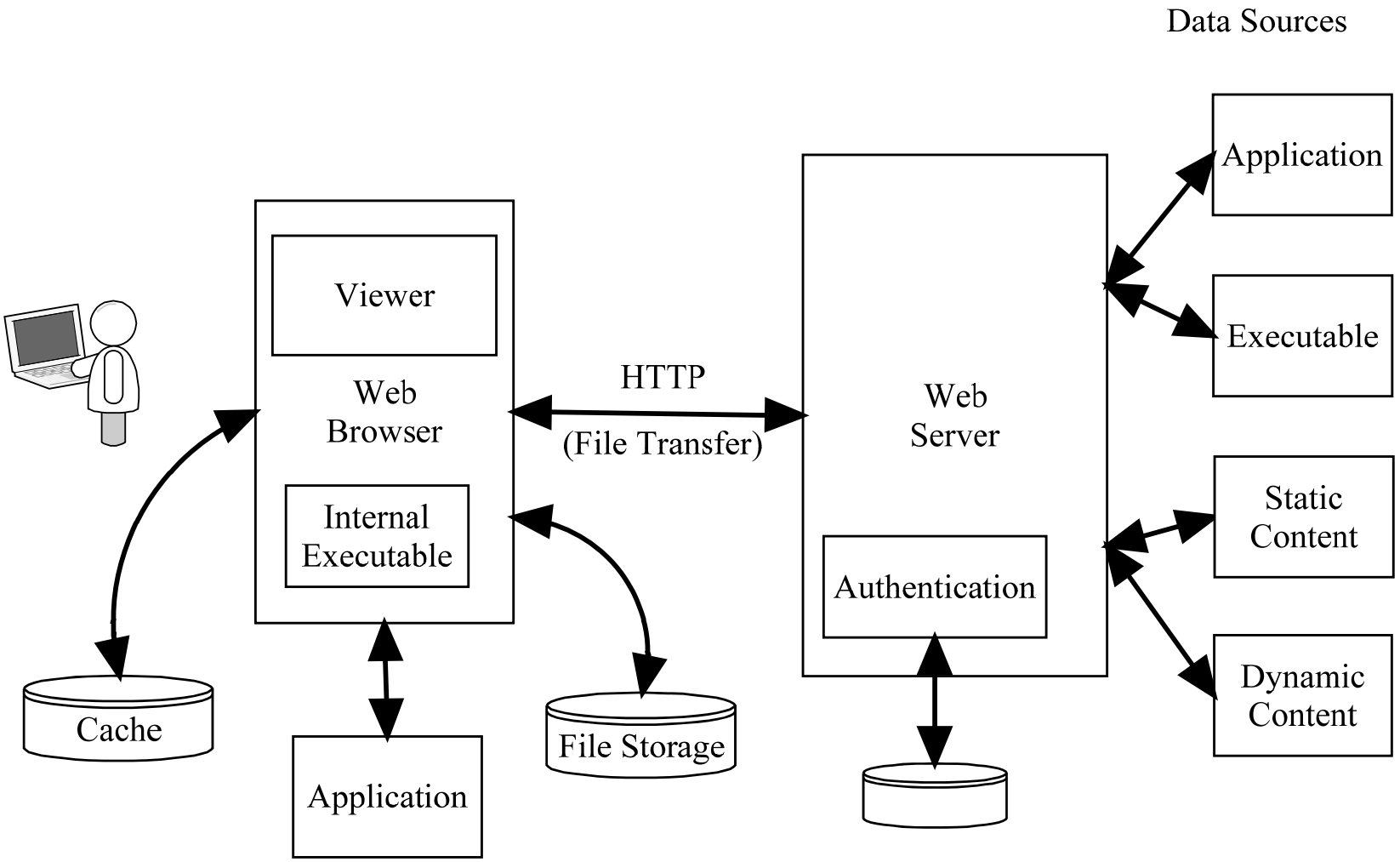
World Wide Web



World Wide Web



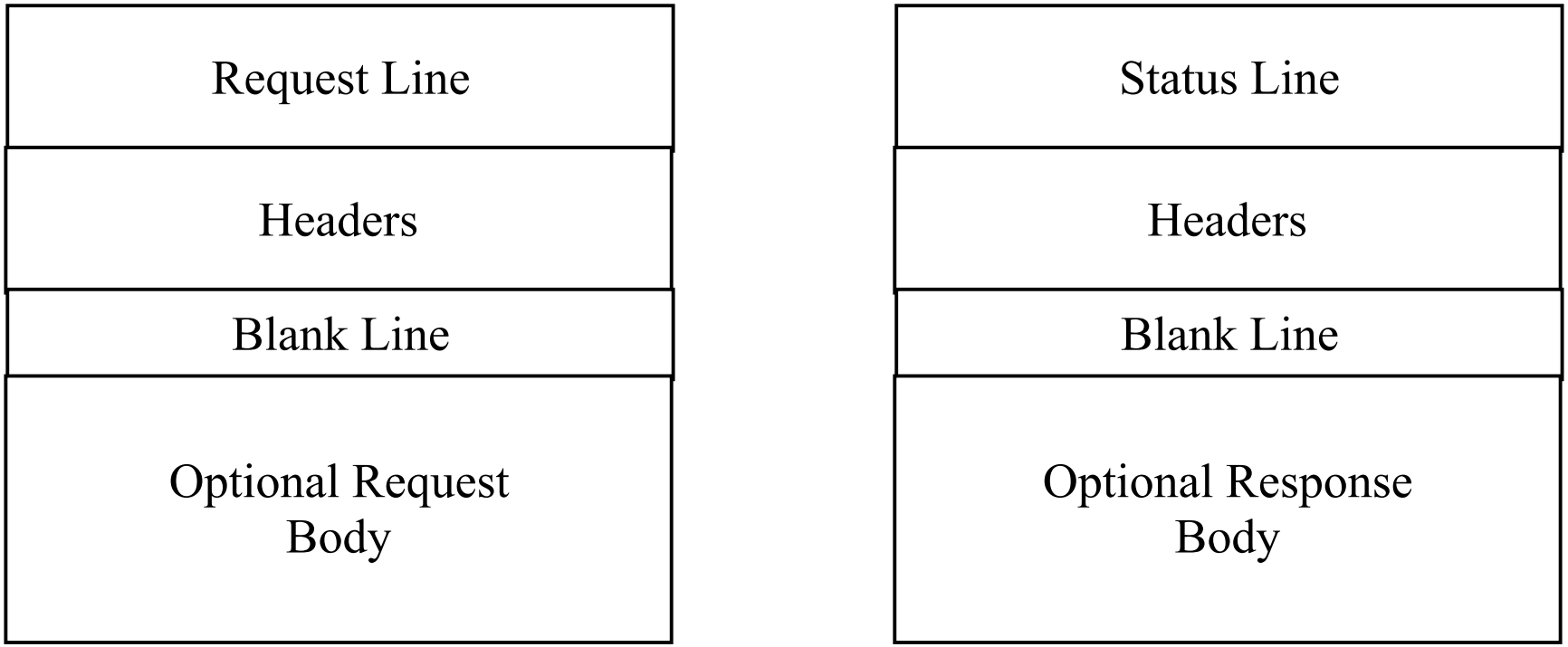
Web Client/Server



HTTP

- Hypertext Transfer Protocol
- Simple command/response protocol
- ASCII based commands
- Typically a new connection for each command/response exchange
- Server runs on port 80 default

HTTP Request & Response



Command Message

Response Message

HTTP Requests

- Three parts:
  - Request line
  - Headers
  - Blank line
  - Body (optional)

- Request line looks like this:

  Request type <sp> URL <sp> HTTP version

  Example: GET http://www.ibm.com HTTP/1.1

  More on request types later

URL

- Uniform Resource Locator
- A URL follows this format: method://host:port/path
- The host can be a machine name or IP address
- The port must be specified if the server is running on a port other than 80.
- The path is the directory where data is stored

Request Types

- GET
- HEAD
- POST
- PUT
- PATCH
- COPY
- MOVE
- DELETE
- LINK
- UNLINK
- OPTION

Many of these types can pose security problems, since they involve modifying or deleting data.

Most servers only implement the first three types: GET, HEAD, POST

Request Types

Type	Action
GET	Retrieve a document specified by the URL.
HEAD	Retrieve the headers from the document specified by the URL. (Response does not contain the body.)
POST	Provide data to the server.
PUT	Provide new or replacement document specified by the URL. (Disabled)
PATCH	Provide differences to document specified by the URL in order to change the document. (Disabled)
COPY	Copy the document specified by the URL to the file specified in the header. (Disabled)
MOVE	Move the document specified by the URL to the file specified in the header. (Disabled)
DELETE	Delete the document specified by the URL. (Disabled)
LINK	Create a link to the document specified in the URL. The name of the link is specified in the header. (Disabled)
UNLINK	Remove the link specified in the URL. (Disabled)
OPTION	Ask the server what options are available.

Response Message

- Four parts:
  - Status line
  - Headers
  - Blank line
  - Body
- The status line looks like this:

  HTTP version <sp> status code <sp> status phrase

  Examples: HTTP/1.1 404 File not found

  HTTP/1.1 200 OK

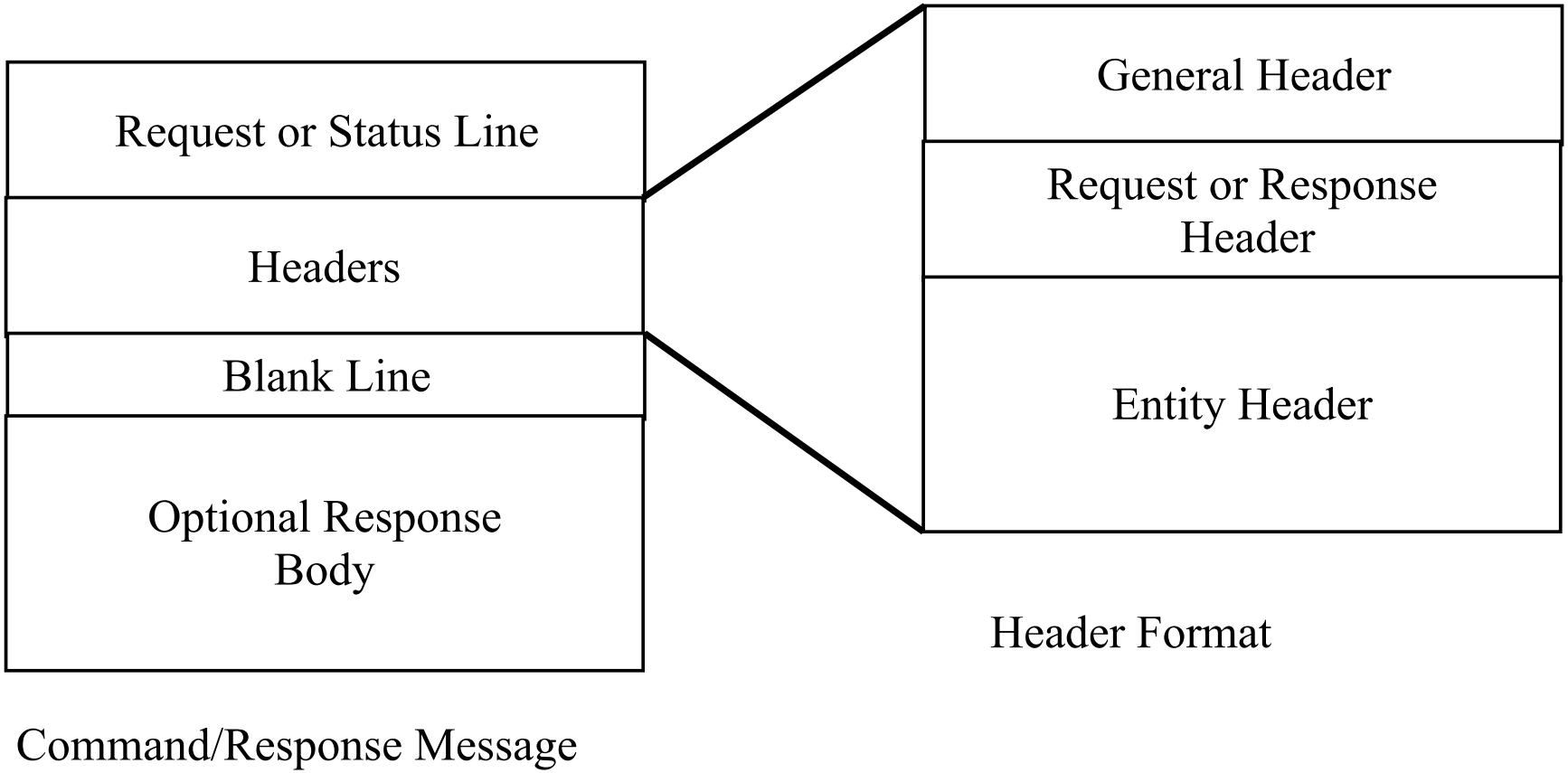
Response Status Codes

- Status codes follow a similar format to FTP and SMTP status codes
- 3 digit ASCII
  - 1xx informational
  - 2xx success
  - 3xx redirection
  - 4xx client error
  - 5xx server error

Example Response Codes

Code	Phrase	Meaning
100	Continue	First part of the request has been received. The client can continue.
200	OK	Successful request
204	No Content	The body contains no content
302	Moved permanently	The document specified by the URL is no longer on the server.
304	Moved temporarily	The document specified by the URL has temporarily moved.
400	Bad request	The request contained a syntax error.
401	Unauthorized	The authentication failed for the requested document.
403	Forbidden	The service requested is not allowed.
404	Not found	The document requested is not found.
405	Method not allowed	The method requested in the URL is not allowed.
500	Internal server error	The server failed.
501	Not implemented	The requested action can not be preformed by the server.
503	Service unavailable	The request cannot be accomplished right now, try again later.

HTTP Headers



HTTP Headers

- Headers have three parts:
  - General header
  - Request or response header, depending on whether the header precedes a request or a response
  - Entity header
- The general header contains the following fields:

Header	Function
Cache-control	Used to specify information about the client side cache.
Connection	Indicates whether the connection should be closed.
Date	Provides the current date.
MIME-version	Indicated the MIME version being used.
Connection	Use to determine connection type.
Keep-Alive	Used to manage keep-alive connection.

HTTP Headers

- The Request header may contain the following fields (all are optional):

Header	Function
Accept	Indicates which data formats the browser can accept.
Accept-charset	Indicates the character set(s) the browser can accept.
Accept-encoding	Indicates what encoding methods the browser can process.
Accept-language	Indicates what language the browser can accept.
From	Provides the e-mail of the user on the browser.
Host	Provides the host and ephemeral port of the browser.
Referrer	Provides the URL of the linked document.
User-agent	Provides information about the browser software.

HTTP Headers

- The response header may contain the following fields

Header	Function
Accept-range	Indicates the server accepts the range requested by the browser.
Retry-after	Indicates the date when the server will be available.
Server	Provides the server application name and version.

HTTP Headers

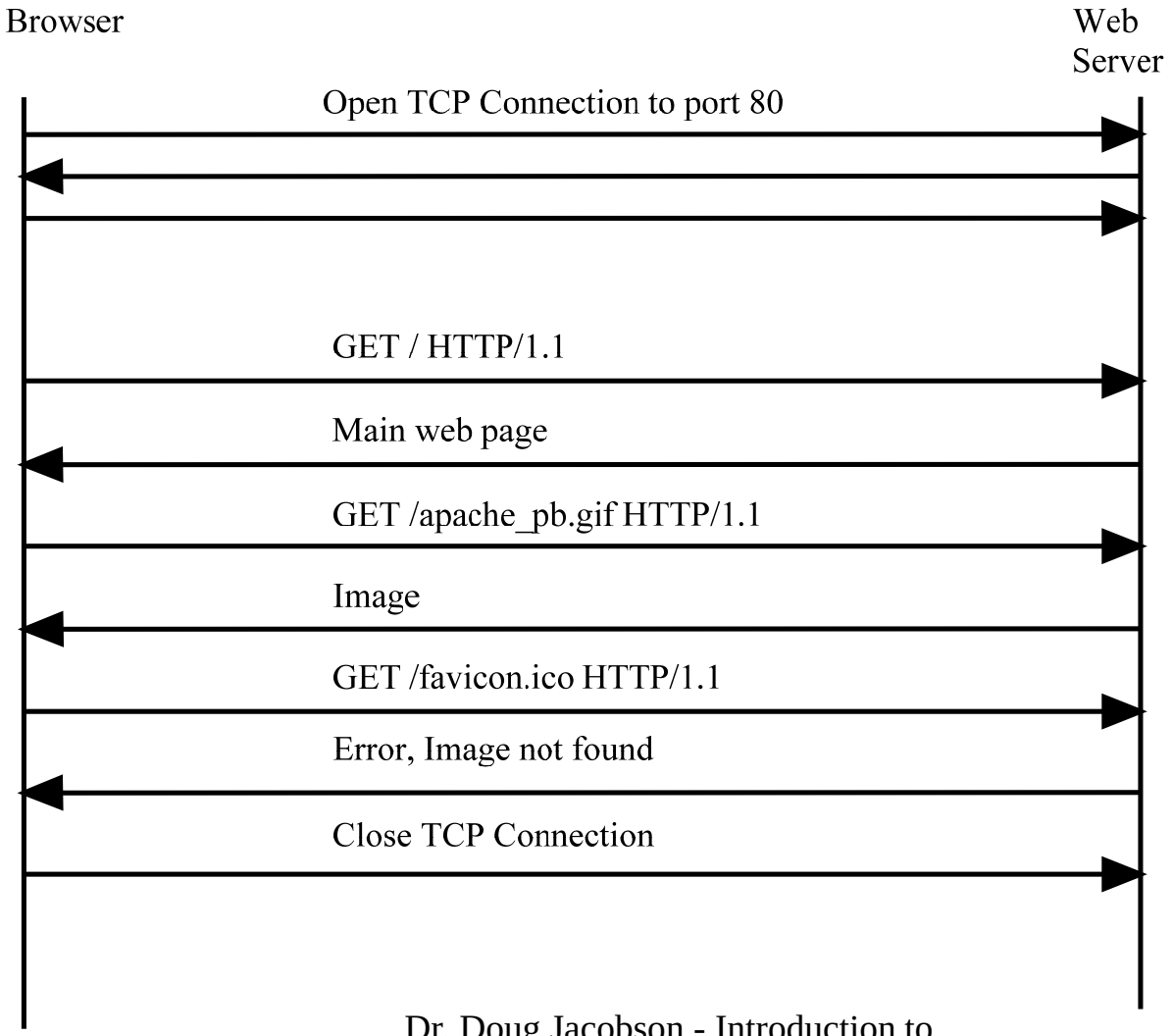
- The entity header may contain the following fields:

Header	Function
Allow	Provides a list of methods allowed for the URL.
Content-encoding	Indicates the encoding method for the document.
Content-language	Indicates the language of the document.
Content-length	Indicates the length of the document.
Content-location	Real name of the document requested.
Content-type	Indicates the media type of the document.
Etag	Provides a tag for the document.
Last-modified	The date the document was last modified.

HTTP Summary

- Request:
  - Request line
  - General Header
  - Request Header
  - Entity header
  - Blank line
  - Optional Body
- Response:
  - Status line
  - General header
  - Request header
  - Entity header
  - Blank line
  - Body
- Note: the entity header does not always appear in the request

HTTP Protocol Exchange



HTTP Request

Request Line

```
GET / HTTP/1.1
```

General Header

```
Keep-Alive: 300
Connection: keep-alive
```

Request Header

```
Host: spock.ee.iastate.edu
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en - US; rv:1.8.0.7)
      Gecko/20060909 Firefox/1.5.0.7
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
```

No Entity Header

Blank Line

No Body

HTTP Response

Status Line

```
HTTP/1.1 200 OK
```

General Header

```
Date: Sat, 28 Oct 2006 16:01:55 GMT
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
```

Response Header

```
Server: Apache/1.3.33 (Unix)
Accept-Ranges: bytes
```

Entity Header

```
Content-Location: index.html.en
Last-Modified: Fri, 04 May 2001 00:00:38 GMT
ETag: "428fd8-5b0-3af1f126;452e43f5"
Content-Length: 1456
Content-Type: text/html
Content-Language: en
```

Blank Line

```
HTML document (1456 bytes long)
```

HTTP Request

Request Line

```
GET /apache_pb.gif HTTP/1.1
```

General Header

```
Keep-Alive: 300
Connection: keep-alive
```

Request Header

```
Host: spock.ee.iastate.edu
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en - US; rv:1.8.0.7)
    Gecko/20060909 Firefox/1.5.0.7
Accept: 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
Referer: http://spock.ee.iastate.edu/
```

No Entity Header

Blank Line

No Body

HTTP Response

Status Line

```
HTTP/1.1 200 OK
```

General Header

```
Date: Sat, 28 Oct 2006 16:01:55 GMT
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
```

Response Header

```
Server: Apache/1.3.33 (Unix)
Accept-Ranges: bytes
```

Entity Header

```
Last-Modified: Wed, 03 Jul 1996 06:18:15 GMT
ETag: "428fd1-916-31da10a7"
Content-Length: 2326
Content-Type: image/gif
```

Blank Line

```
GIF image (2326 bytes long)
```

HTTP Request

Request Line

```
GET /favicon.ico HTTP/1.1
```

General Header

```
Keep-Alive: 300
Connection: keep-alive
```

Request Header

```
Host: spock.ee.iastate.edu
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en - US; rv:1.8.0.7)
      Gecko/20060909 Firefox/1.5.0.7
Accept: 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
```

No Entity Header

Blank Line

No Body

HTTP Response

Status Line

```
HTTP/1.1 404 Not Found
```

General Header

```
Date: Sat, 28 Oct 2006 16:01:55 GMT
Keep-Alive: timeout=15, max=97
Connection: Keep-Alive
```

Response Header

```
Server: Apache/1.3.33 (Unix)
```

Entity Header

```
Content-Type: text/html; charset=iso-8859-1
```

Blank Line

```
HTML Document
```

Header Based

- Buffer overflow problems
- Server can pass HTTP requests to back-end servers and applications so header problems are not just with the WEB server
- Some header-based attacks facilitate authentication-based attacks
- Accessing hidden pages

Protocol Based

- Not many protocol based attacks since it is a command/response protocol

Authentication Based

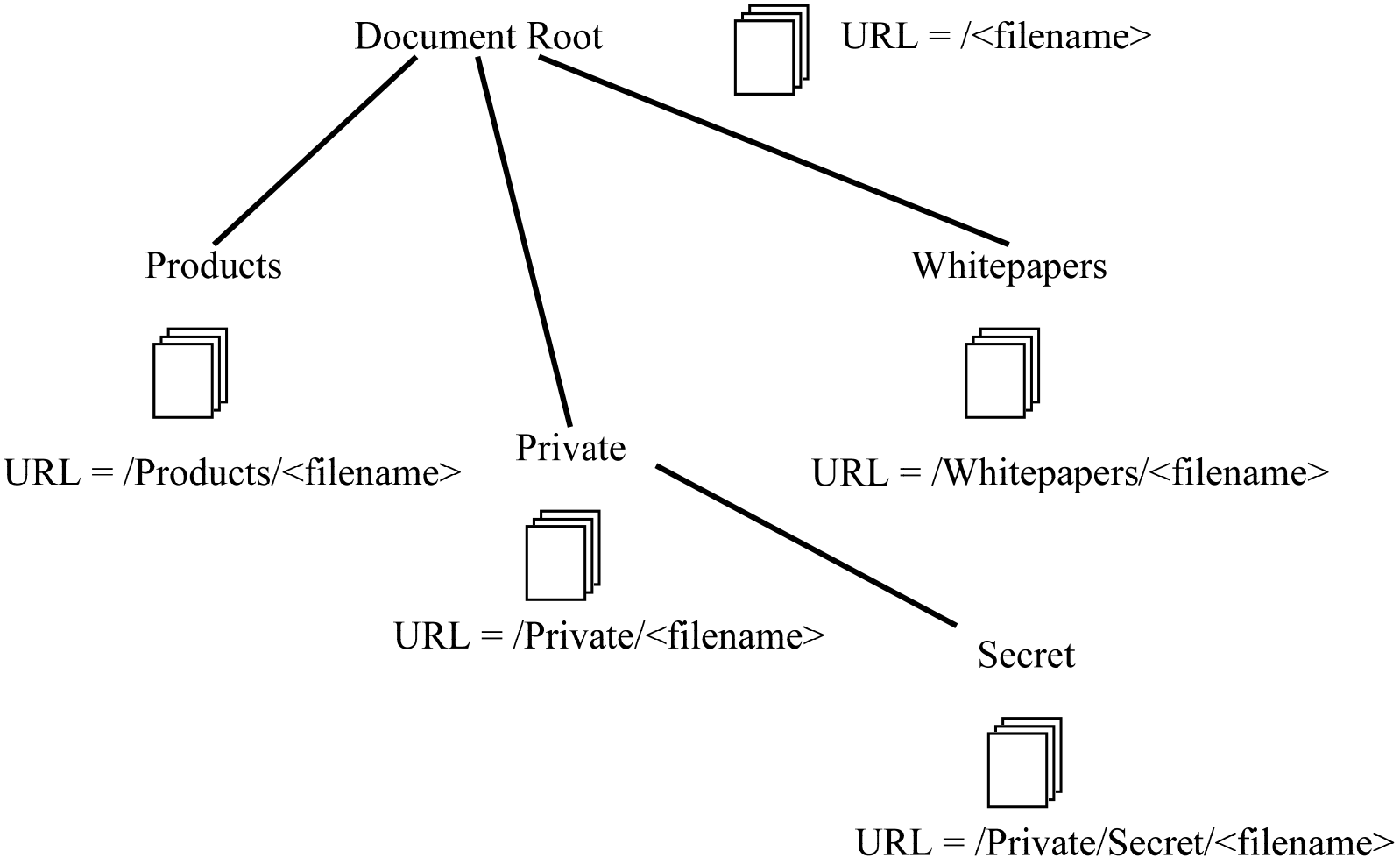
- This is the most common method of attack in the WEB.
- The web server uses HTTP to request user credentials.
- Authentication can also be directly with the server side application (to be discussed later)
- Authentication is used to access pages within a directory on the server

WEB Authentication

- Server challenge:
  - WWW-Authenticate: Basic realm=“Text String"
- Client Challenge:
  - user-ID and password, separated by a single colon (":") character, within a base64 encoded string.

  For example:
  - Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

HTTP Authentication



HTTP Authentication

Request Header

```
GET /~dougj/private /doc.html  HTTP/1.1
Host: spock.ee.iastate.edu
```

Response Header

```
HTTP/1.1 401 Authorization Required
Date: Tue, 14 Nov 2006 22:37:47 GMT
Server: Apache/1.3.33 (Unix)
WWW-Authenticate: Basic realm="Enter Password"
```

Request Header

```
GET /~dougj/private/doc.html HTTP/1.1
Host: spock.ee.iastate.edu
Authorization: Basic bG9yaWVuOmZpcnN0b25l
```

Web Authentication

- Can be sniffed (traffic based attack)
- Can be guessed

- Countermeasures:
  - Encrypted sessions
  - Good passwords

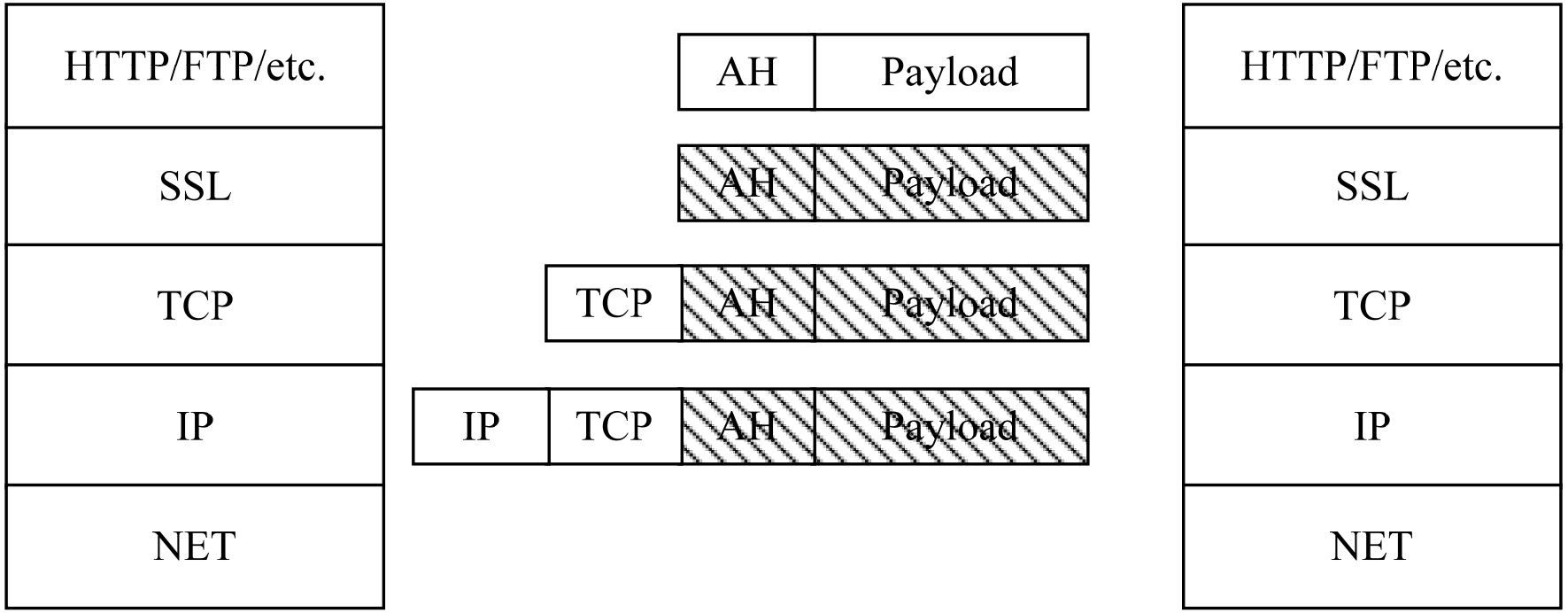
Traffic Based

- Very common attacks
  - Flooding
  - Web Hugging
- HTTP is clear text.
  - HTTP does not support encrypted sessions.
  - Encrypted sessions are supported using transport layer encryption

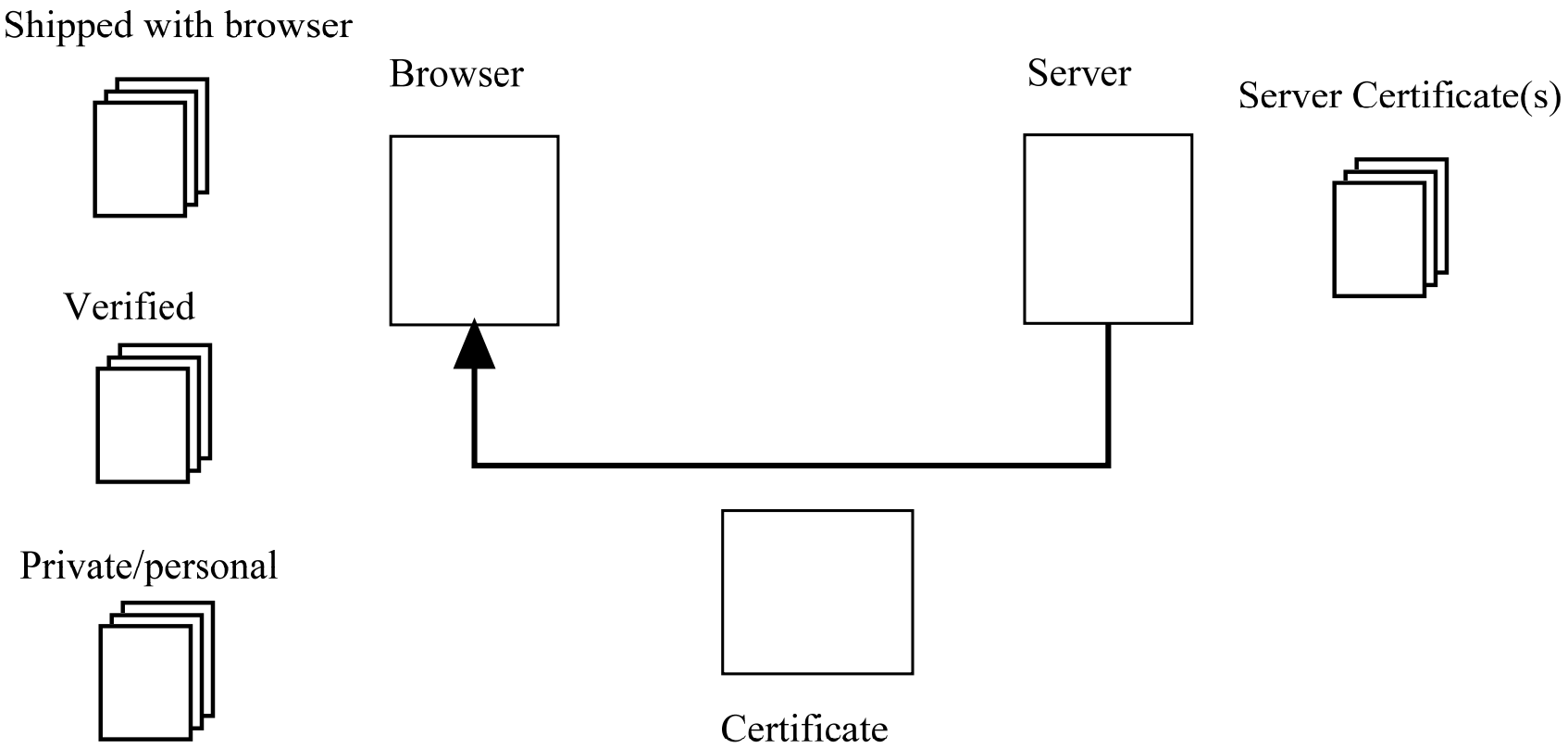
HTTPS

- Uses the Secure Socket Layer SSL
- Port 443
- Uses public key certificates

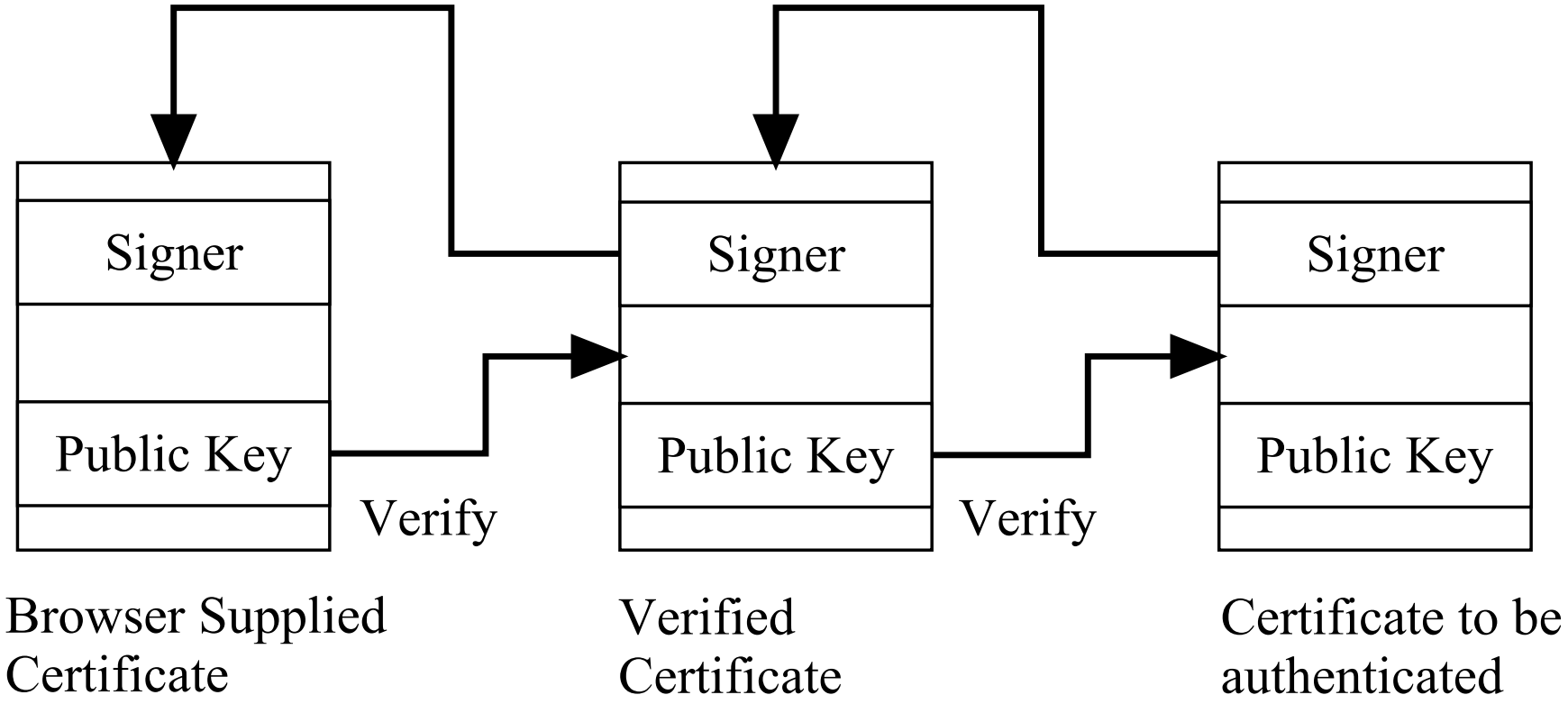
HTTPS


HTTP/FTP/etc.
SSL
TCP
IP
NET
AH
Payload
AH
Payload
TCP
AH
Payload
IP
TCP
AH
Payload
HTTP/FTP/etc.
SSL
TCP
IP
NET

HTTPS Certificates



Certificate chain of trust



HTML

- Hypertext Markup Language
- Two parts
  - Head: contains information for the browser
  - Body: contains information to display on the screen
- Contains markup codes which tell the browser how to display the page
- Each markup code is called an element or a tag
- Tags can be nested:

```
<tag1>
    <tag2>
    </tag2>
</tag1>
```

HTML

Start of an HTML document

```
<HMTL>
```

HEAD section

```
<HEAD>
<TITLE> The page title </TITLE>
</HEAD>
```

BODY section

```
<BODY>
        HTML CODE
</BODY>
```

End of the HTML document

```
</HTML>
```

HTML Tags

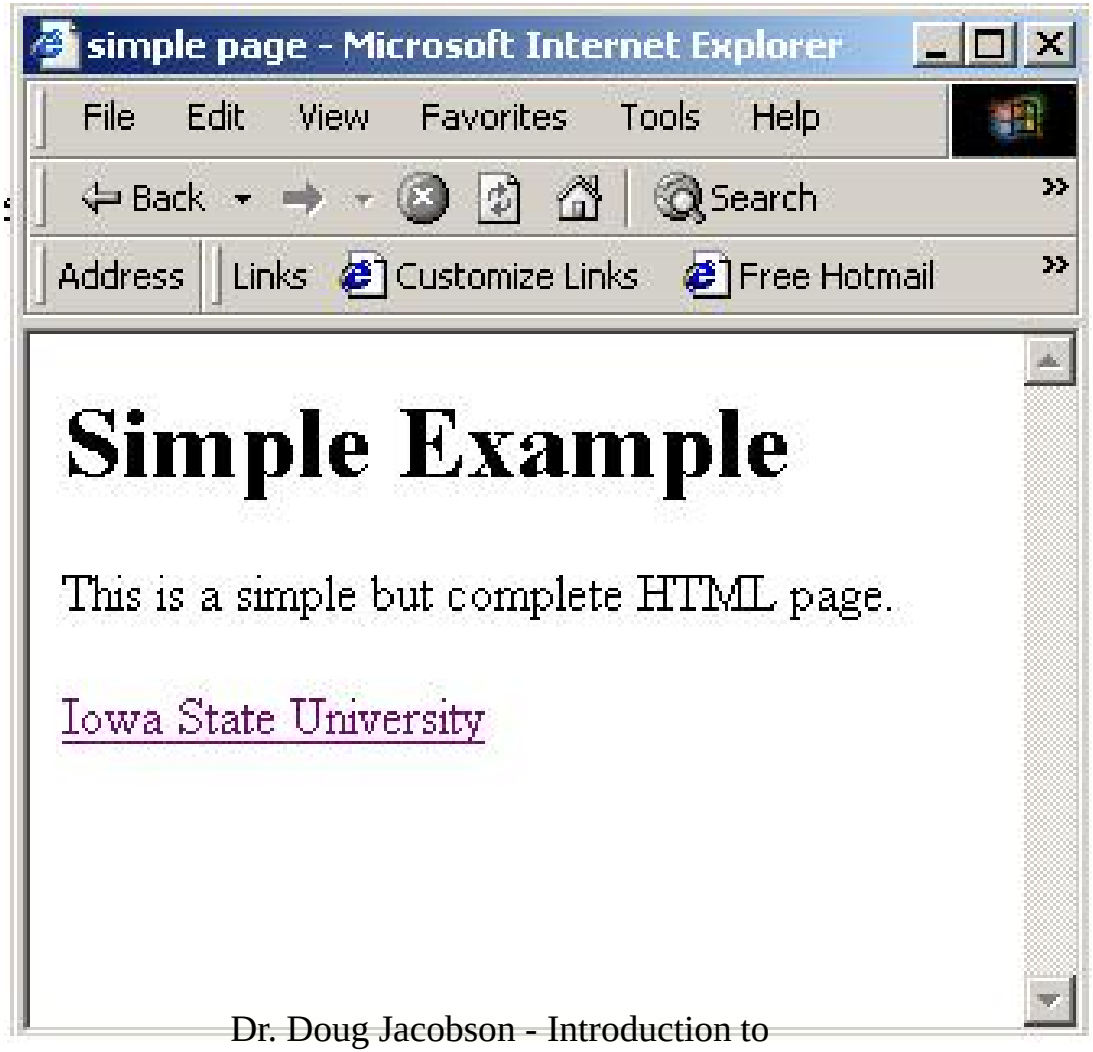
- Basic HTML tags
  - <HTML> - tells browser where page starts
  - <HEAD> - start of head section
  - <TITLE> - text to be displayed in title bar
  - <BODY> - start of body section
  - <H1> - largest header size
  - <P> - paragraph
  -
 - break (new line)
  - <UL> - unordered list
  - <LI> - list item
  - link - hyperlink to abc.com
  - <img src ="red.gif"> - display the image red.gif
  - <APPLET> CODE=XXX </APPLET> - java applet

HTML Example

- Here is a simple HTML page

```
<HTML>
<HEAD><TITLE>simple page</TITLE>
</HEAD>
<BODY>
<H1>Simple Example</H1>
<p>
This is a simple but complete HTML page.
<p>
<a href=http://www.iastate.edu>Iowa State University</a>
</BODY>
</HTML>
```

HTML Example



Header based

- HTML documents with hyperlinks where the text is different than the link
- Pictures can come from anywhere
- Links to rouge code.
- Countermeasures:
  - User education

Protocol Based

- Different that normal protocols (no message exchange)
- Client side downloads can be malicious (viruses, worms, Trojan horses)
- Countermeasures:
  - Scanners, filters
  - Education

Authentication Based

- HTML does not directly support authentication
- HTML can be used to direct you to the wrong site, and since there is no host to user authentication. The site may not be the true site.

- Countermeasures:
  - User education

Traffic Based

- Sniffing

Server Side Security

- HTML documents can cause applications to be run.
- Common method is via a CGI script
- HTML documents can also front end other applications like databases through a CGI script

CGI

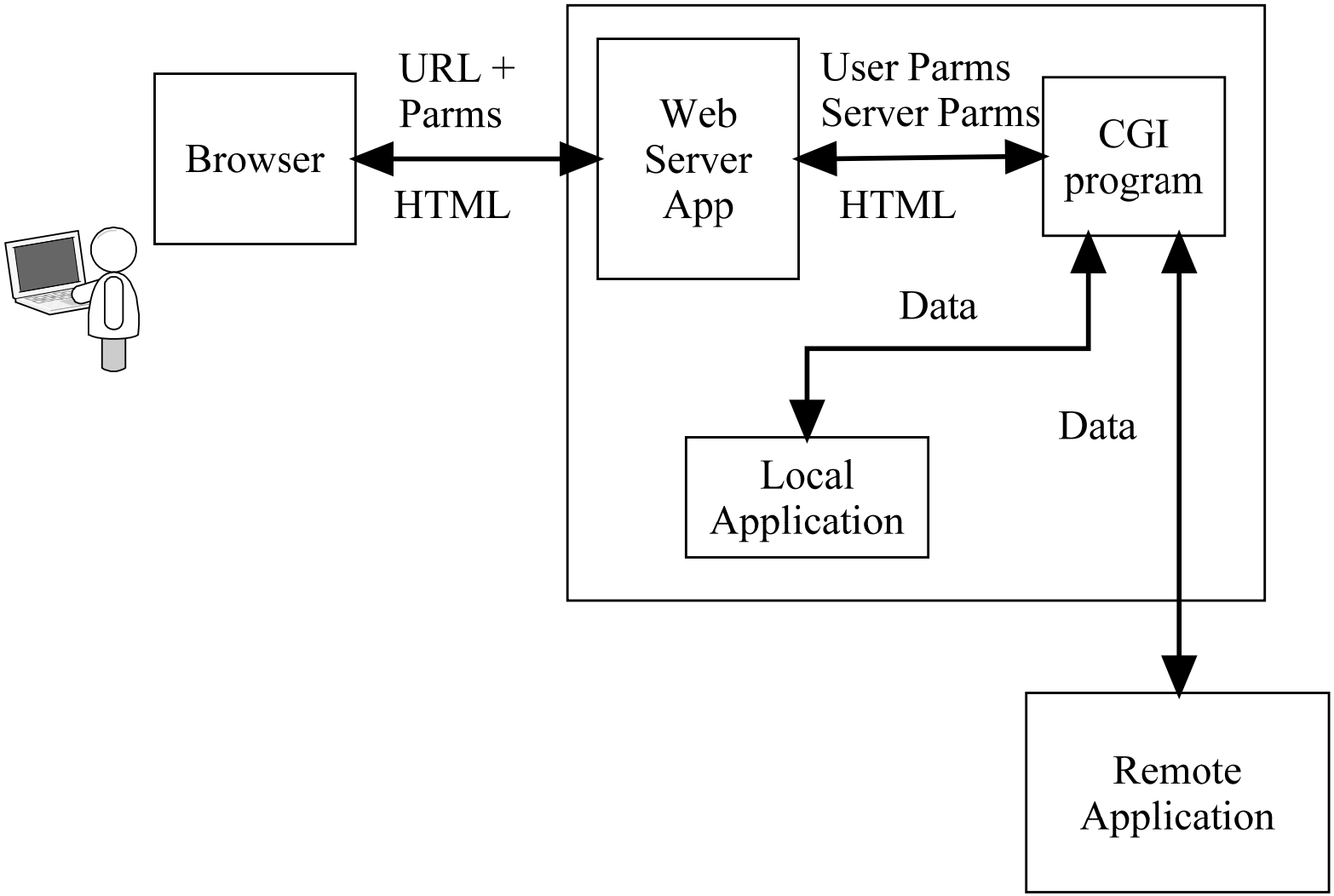
- Common Gateway Interface
- Allows a server to run programs and scripts
- CGI is the method for passing data back and forth between the server and the program or script
- Variables can be passed to the program or script either through a form or after the '?' in the URL
- Examples:

http://HOST/cgi-bin/program.pl?name=bob;state=ia

or

<FORM METHOD=POST ACTION=/cgi-bin/program.pl>

CGI


Browser
URL +
Parms
HTML
Web
Server
App
User Parms
Server Parms
HTML
CGI
program
Data
Data
Local
Application
Remote
Application

CGI

- CGI can access additional information through environment variables
- Environment variables are passed from the server to the program or script
- Environment variables include:

Query_string	HTTP_referrer
Remote_addr	HTTP_user_agent
Remote_host	Path_info
Remote_user	Server_port
Server_name	

Header Based

- Buffer overflow problems on CGI scripts
- Server can pass HTTP requests to back-end servers and applications so header problems are not just with the WEB server
- Some header-based attacks facilitate authentication-based attacks or allow direct access to the web server

Protocol Based

- Not many protocol based attacks since it is not a protocol.

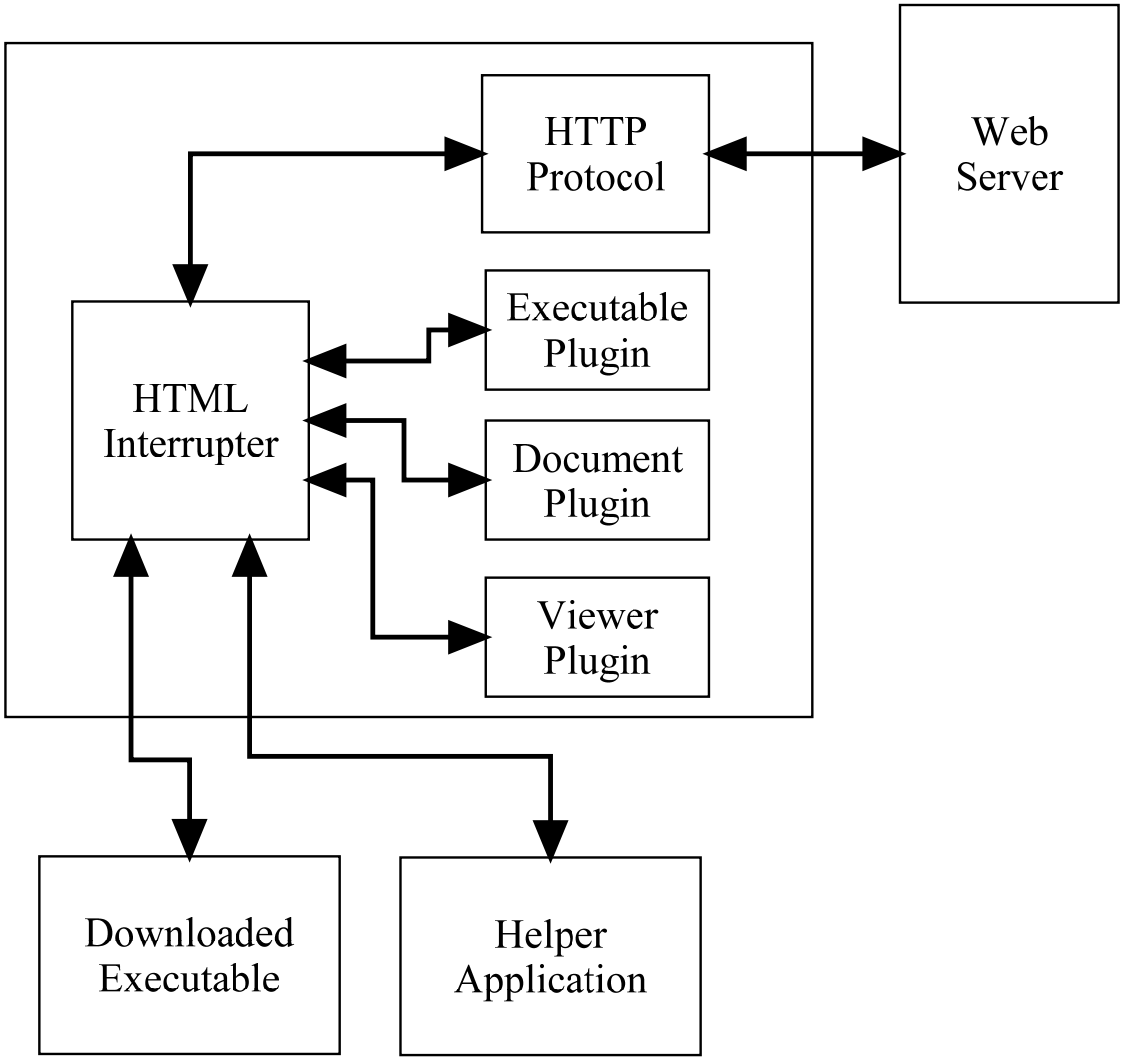
Authentication Based

- Provide access to application authentication methods.

Traffic Based

- No additional attacks due to CGI scripts

Client Side Security



Client Side Security

- Cookies are placed on the client
- Executable programs can be downloaded automatically by the browser.
  - Java Scripts
  - Active X
- They can send information back to the server.

Cookies

- A file on the users computer in which the website can store data
  - Why cookies?
    - HTTP is stateless protocol, websites like to keep state information on your information and habits
- First implementation of cookies allowed any site to read another website's cookie.
- Now only the site the storied the cookie can look at it
- Example of Amazon cookie
- Netscape has one cookie file whereas explorer has a file for each cookie
- Passwords can be in clear text

Clear Gifs

- One pixel gif
- Hyperlink to another site
- This allows people to track documents

Client side Executables

- Plugins: Applications that are part of the browser to help read different file types
- Scripts: Programs run by the browser often to provide inactive graphics or forms
- Downloads: Programs that are downloaded using the browser

Header/Protocol Based

- Not many attacks in these categories since there is not really a separate header or protocol.

Authentication Based

- No authentication of applications leads to malicious code
- Client side executables provide a method for attackers to interject code
  - Trojan horses
  - Spyware
  - Key loggers
- Can be coupled with email attacks (using phishing to direct a user to a web side which downloads code

Authentication based

- Mitigation:
  - Client side protection
  - User awareness

Traffic Based

- Not very common since, however some malicious programs may generate large amounts of network traffic.

General Countermeasures

- Encryption and authentication
- URL Filtering
- Content filtering

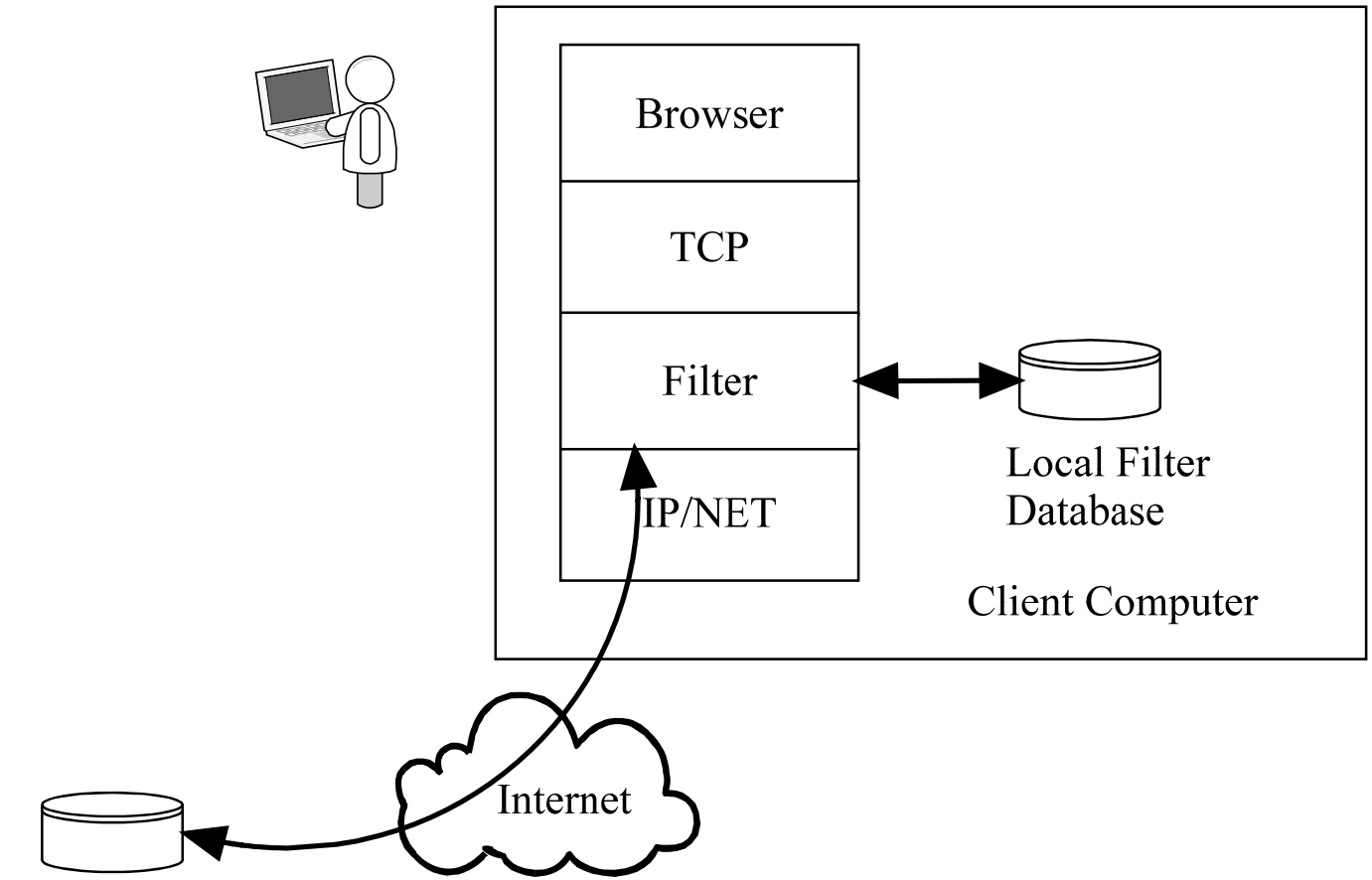
Encrypted Transactions

- SSL
  - Secure Socket Layer
  - Broader application then HTTP
  - Another layer to the mix, creates a secure layer between HTTP and TCP
  - Uses port 443
  - Browser is shipped with certificates for support of this service
  - Communicates through an encrypted channel

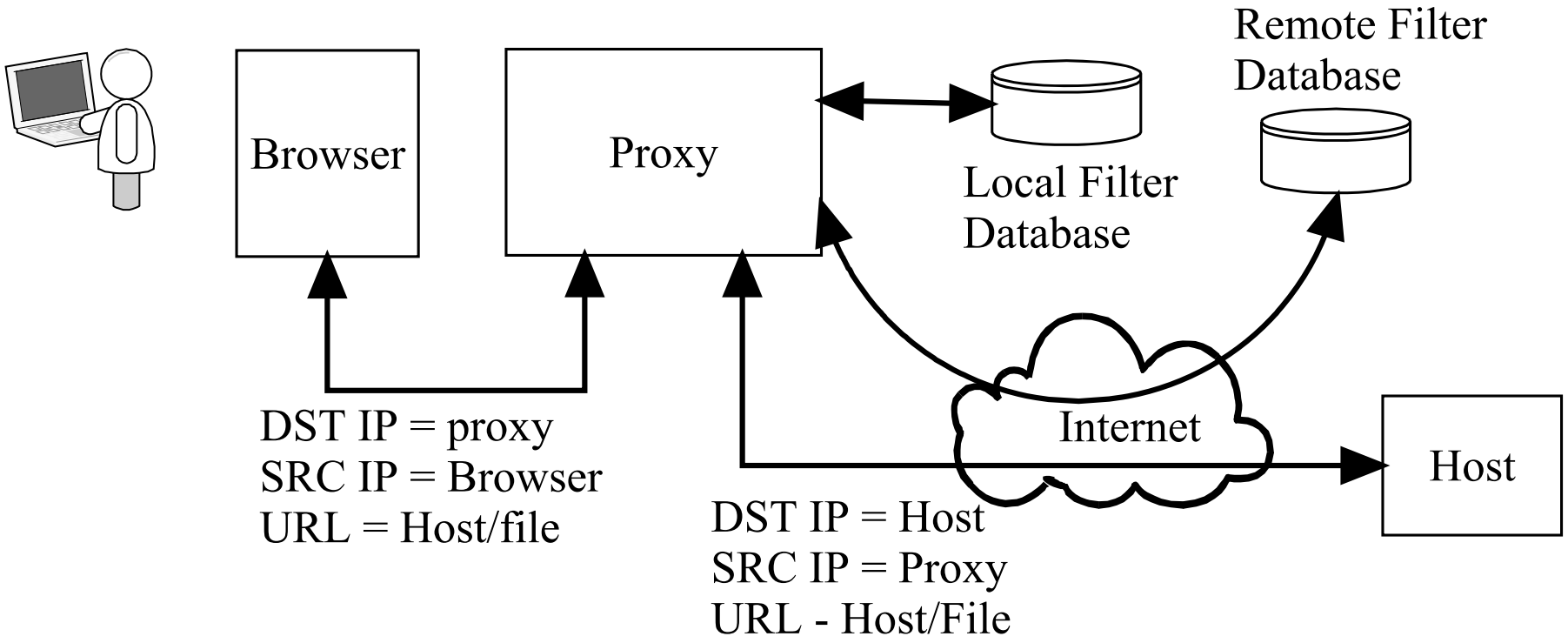
URL Filtering

- Client side
- Proxy based
- Network based

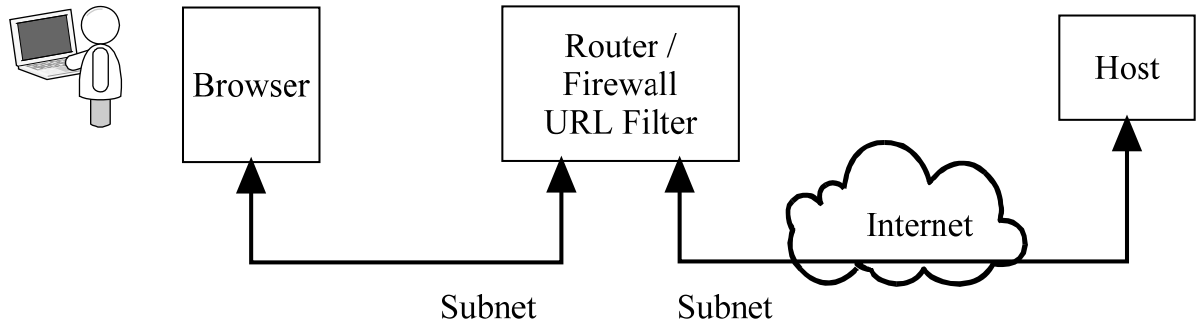
Client Side URL Filter



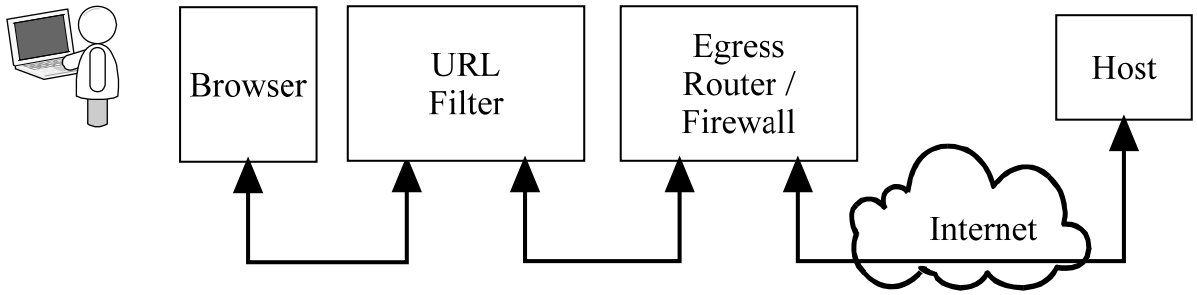
Proxy Based URL Filter



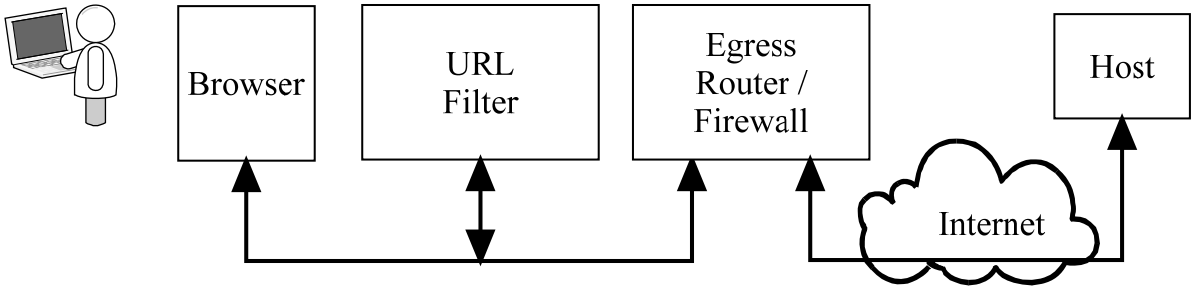
Network Based URL Filter



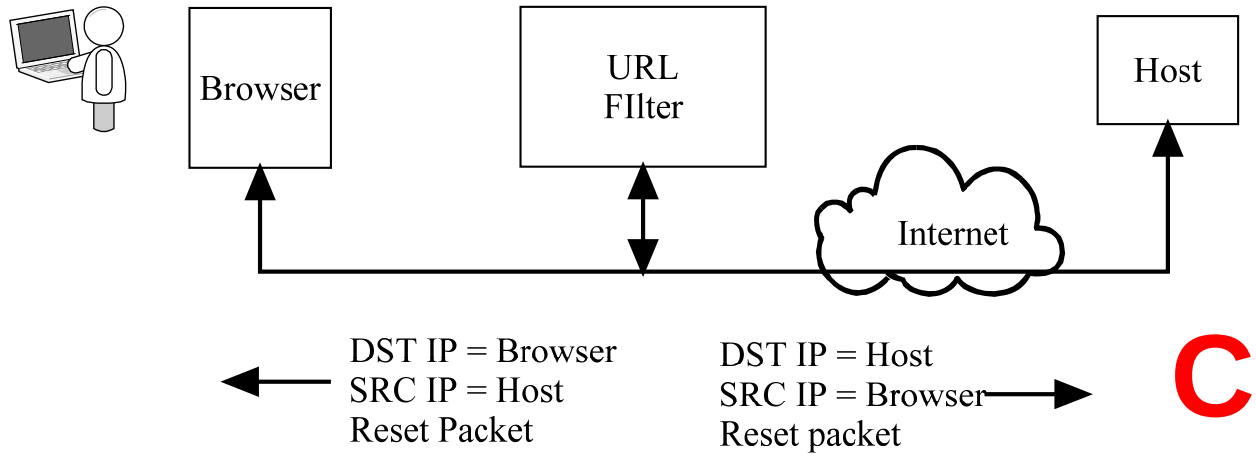
Network device



In-line Transparent

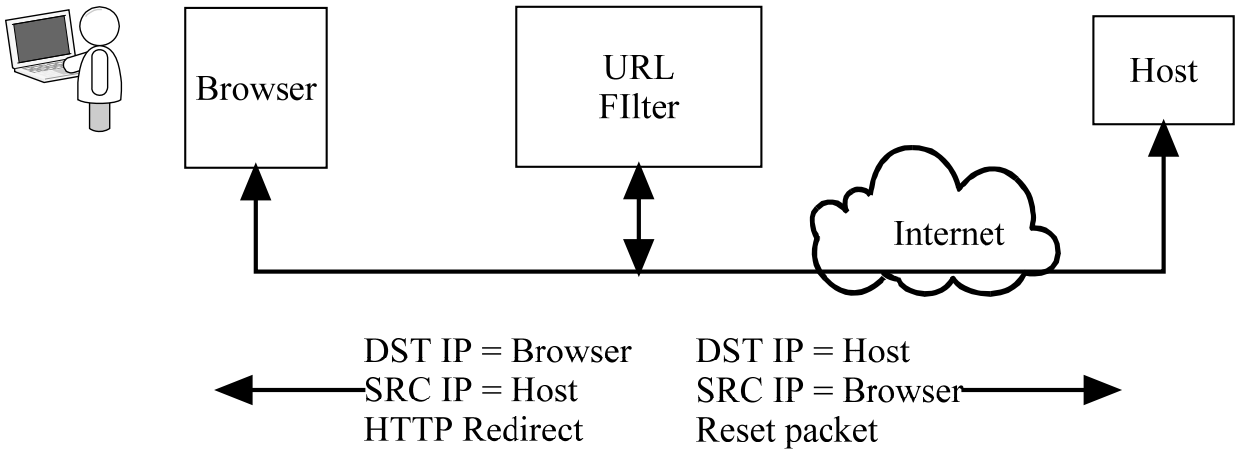


Transparent



Termination Blocking

Connection Blocking



Redirection Blocking

Content Filters

- Proxy based
- Network based

Proxy Based Content Filter