Information Systems Security

Lectures 7 & 8

Web Security

Dr. En. Bader Ahmad

References

[1] Google Code for Educator: Sample Course Content, Web Security.

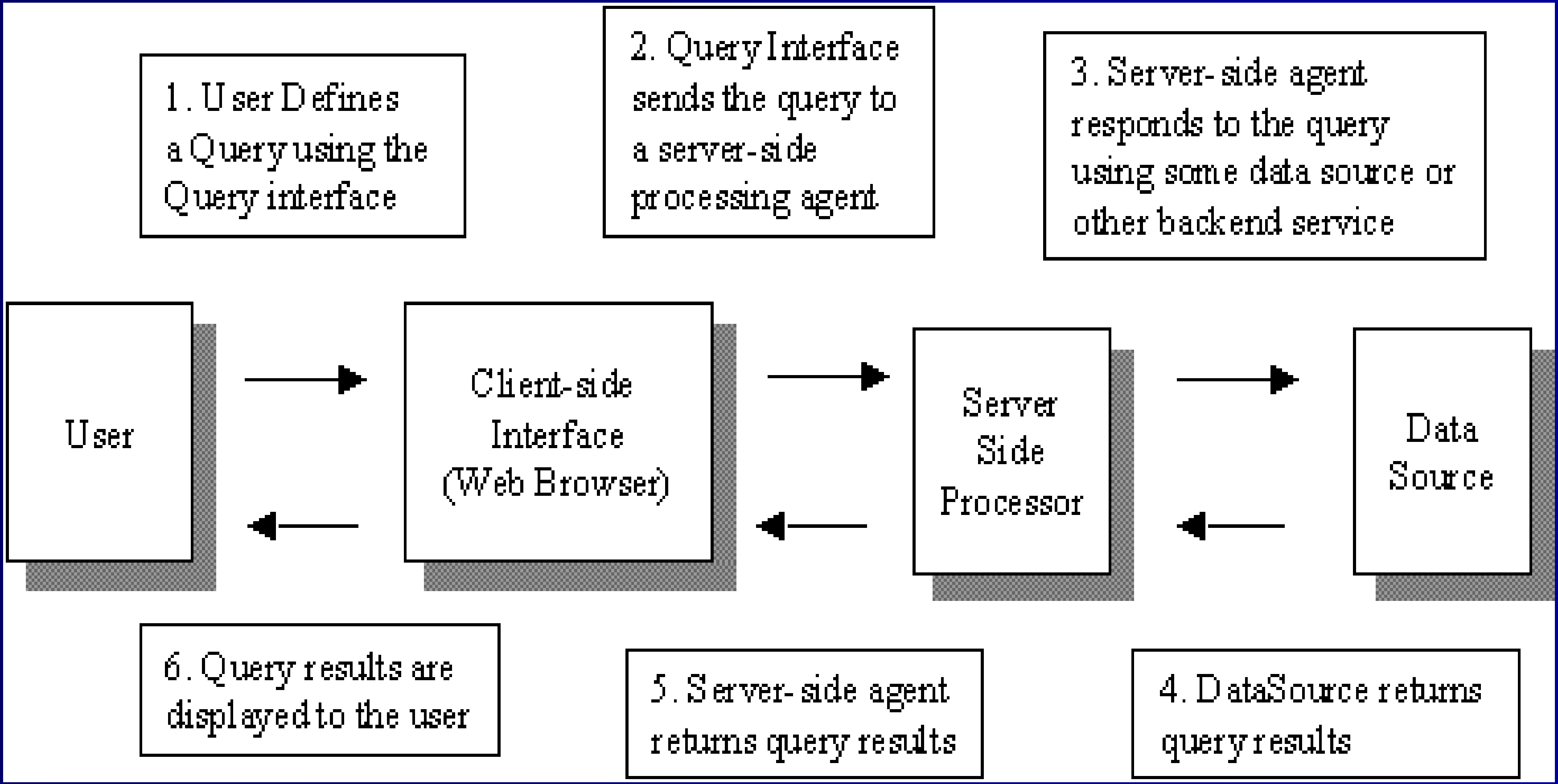
http://code.google.com/edu/content/submissions/web_security/listing.

[2] *Network security, The complete Reference*. R. Bragg, M. Rhodes-Ousley, K. Strassberg. McGraw-Hill Osborne, 2004.

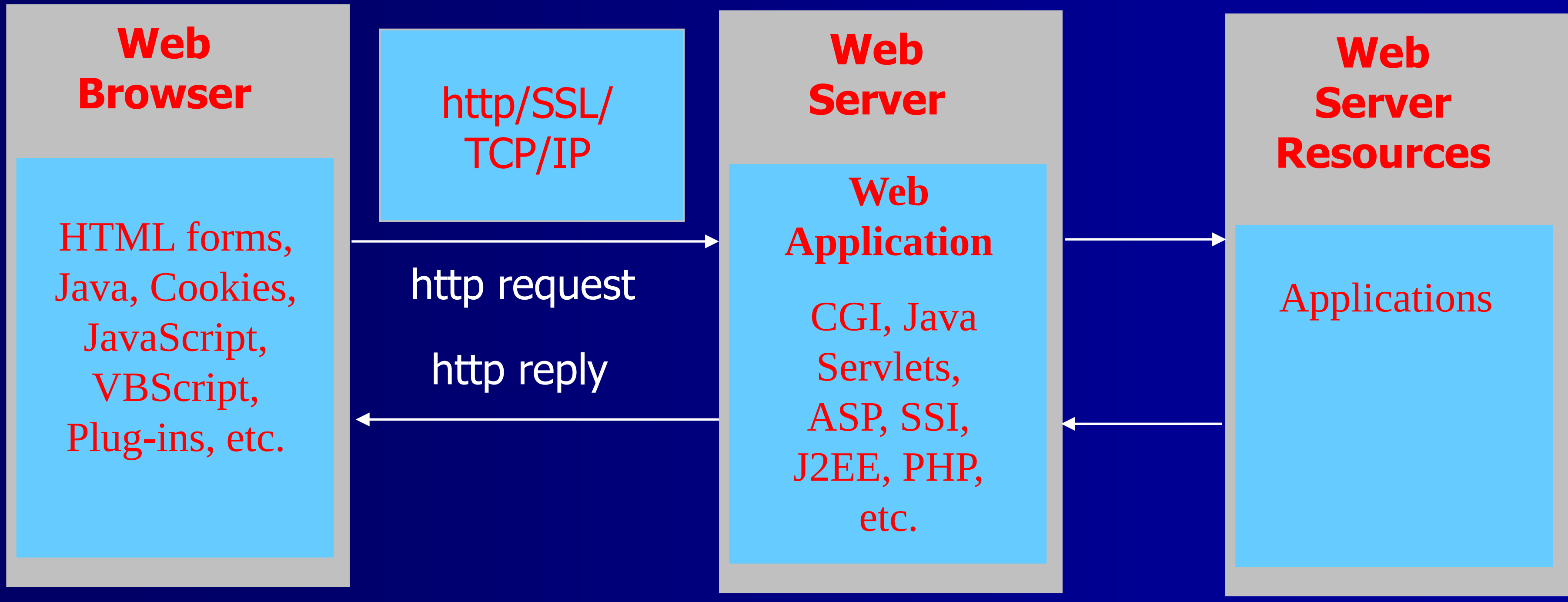
Outline

1. Web System

- Generic web application work flow diagram:


1. User Defines a Query using the Query interface
2. Query Interface sends the query to a server-side processing agent
3. Server-side agent responds to the query using some data source or other backend service
User
Client-side Interface (Web Browser)
Server Side Processor
Data Source
6. Query results are displayed to the user
5. Server-side agent returns query results
4. DataSource returns query results

Web System


Web Browser
HTML forms, Java, Cookies, JavaScript, VBScript, Plug-ins, etc.
http/SSL/ TCP/IP
http request
http reply
Web Server
Web Application
CGI, Java Servlets, ASP, SSI, J2EE, PHP, etc.
Web Server Resources
Applications

2. Web System Security

1. Web Server Security
2. Web Browser Security
3. Web Application Security
4. Channel Security

3. Simple Web Server*

- To illustrate what can go wrong if we do not design for security in our web applications from the start, consider a simple web server implemented in Java.
- All this program does is serve documents using HTTP.
- We will walkthrough the code in the following slides.
- This web server only supports simple HTTP GET requests.

* Slides 7-17 taken from [1]

Some Preliminaries…

- **HyperText Transfer Protocol (HTTP):** The communications protocol used to connect to servers on the Web.
- Its primary function is to establish a connection with a Web server and transmit HTML pages to the client browser or any other files required by an HTTP application.
- **http is stateless (ie, request/reply)**
- Addresses of Web sites begin with an **http://** prefix.

Some Preliminaries…

- A typical HTTP request that a browser makes to a web server:

```
Get / HTTP/1.0
```

- When the server receives this request for filename / (which means the *root* document on the web server), it attempts to load index.html. It sends back:

```
HTTP/1.0 200 OK
```

followed by the document contents.

SimpleWebServer: main()

```
/* This method is called when the program is run from the
   command line. */

public static void main (String argv[]) throws Exception {

   /* Create a SimpleWebServer object, and run it */

   SimpleWebServer sws = new SimpleWebServer();
   sws.run();
}
```

SimpleWebServer Class

```
public class SimpleWebServer {
    /* Run the HTTP server on this TCP port. */
    private static final int PORT = 8080;

    /* The socket used to process incoming
  connections
       from web clients */
    private static ServerSocket dServerSocket;

    public SimpleWebServer () throws Exception {
     dServerSocket = new ServerSocket (PORT);
    }

    public void run() throws Exception {
     while (true) {
        /* wait for a connection from a client */
        Socket s = dServerSocket.accept();

        /* then process the client's request */
        processRequest(s);
   }}
```

SimpleWebServer: processRequest 1

```
   /* Reads the HTTP request from the client, and
      responds with the file the user requested or
      a HTTP error code. */

public void processRequest(Socket s) throws
 Exception {

 /* used to read data from the client */

 BufferedReader br =
      new BufferedReader (new InputStreamReader
 (s.getInputStream()));

 /* used to write data to the client */

 OutputStreamWriter osw =
      new OutputStreamWriter (s.getOutputStream());
```

SimpleWebServer: processRequest 2

```
/* read the HTTP request from the client
*/
String request = br.readLine();

String command = null;
String pathname = null;

/* parse the HTTP request */
StringTokenizer st =
    new StringTokenizer (request, " ");

command = st.nextToken();
pathname = st.nextToken();
```

SimpleWebServer: processRequest 3

```
if (command.equals("GET")) {
    /* if the request is a GET
       try to respond with the file
       the user is requesting */
    serveFile (osw,pathname);
}
else {
    /* if the request is a NOT a GET,
       return an error saying this server
       does not implement the requested
command */
    osw.write ("HTTP/1.0 501 Not
Implemented\n\n");
}

/* close the connection to the client */
osw.close();
```

SimpleWebServer: serveFile 1

```
public void serveFile (OutputStreamWriter osw,
          String pathname) throws Exception {
  FileReader fr=null;
  int c=-1;
  StringBuffer sb = new StringBuffer();

  /* remove the initial slash at the beginning
     of the pathname in the request */
  if (pathname.charAt(0)=='/')
      pathname=pathname.substring(1);

  /* if there was no filename specified by the
     client, serve the "index.html" file */
  if (pathname.equals(""))
      pathname="index.html";
```

SimpleWebServer: serveFile 2

```
/* try to open file specified by pathname */
  try {
      fr = new FileReader (pathname);
      c = fr.read();
  }

  catch (Exception e) {
      /* if the file is not found,return the
         appropriate HTTP response code  */
      osw.write ("HTTP/1.0 404 Not
  Found\n\n");
      return;
  }
```

SimpleWebServer: serveFile 3

```
/* if the requested file can be
successfully opened and read, then return
an OK response code and send the contents
of the file */

osw.write ("HTTP/1.0 200 OK\n\n");
while (c != -1) {
    sb.append((char)c);
    c = fr.read();
}
osw.write (sb.toString());
```

SimpleWebServer Vulnerabilities

- Can you identify any security vulnerabilities in SimpleWebServer? Or what can go wrong?
- Yes: *Denial of Service (DoS):*
  - An attacker makes a web server unavailable, but
  - How?
- DoS on SimpleWebServer:
  - Just send a carriage return as the first message instead of a properly formatted GET message…
  - The web server crashes
  - Service to all subsequent clients is denied until the web server is restarted

4. Web Server Security: Overview

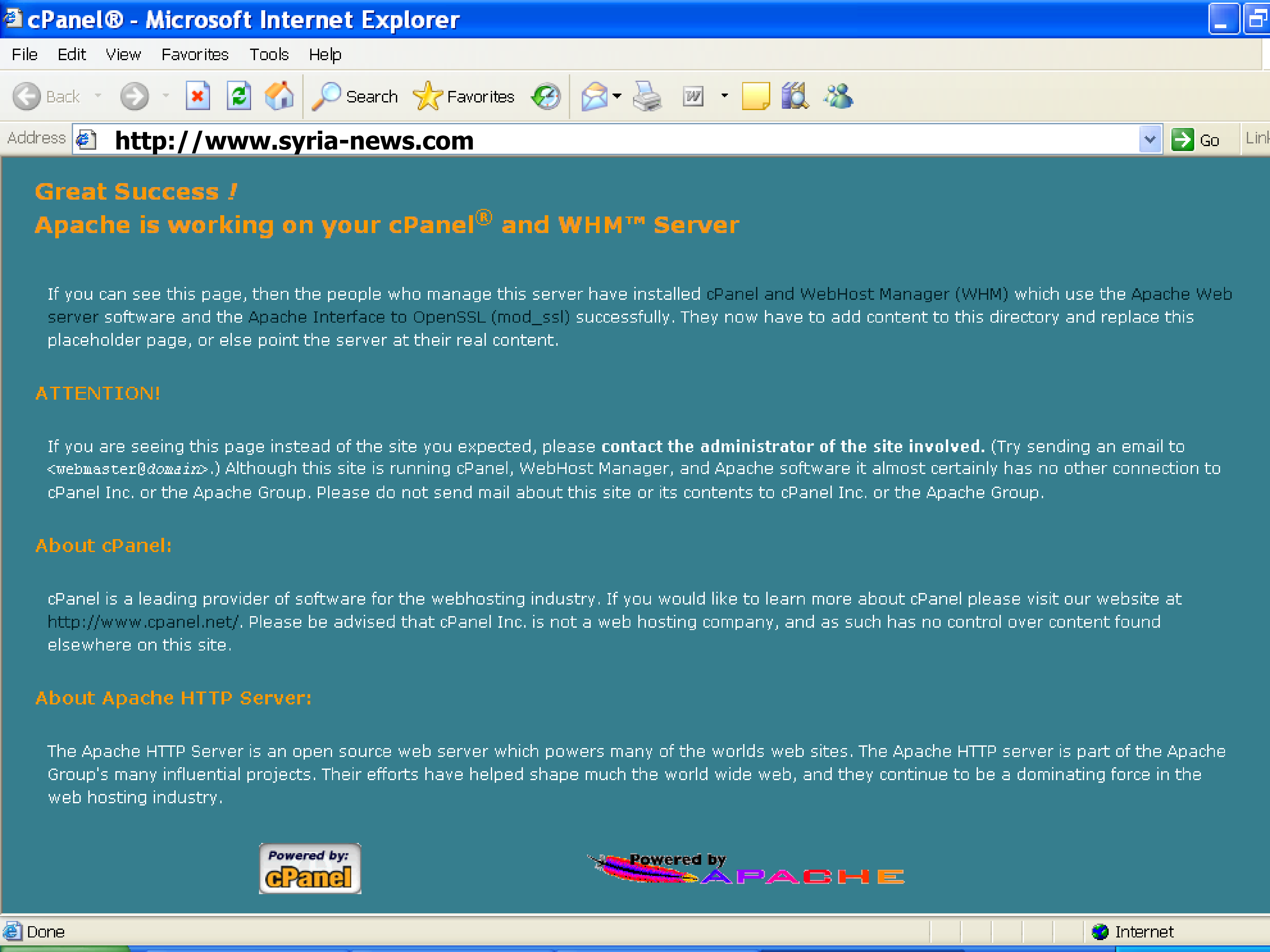
- Consider the following HTML code:

```
<html>
<head>
<title> Hello world </title>
</head>
</html>
```

- Attackers can try 2 strategies to penetrate the web server hosting this HTML code:
  - Exploit web application insecurity
    - there no Exploit in this code
  - Hacking web server itself
    - See the SimpleWebServer : DoS attack

Web Server Security: Goals of server attacks

1. **Web site defacement**
   - Corruption of the HTML code.
   - Example: Next slide
2. **Data Corruption**
   - Any data on the server can be deleted or modified.
3. **Data Theft**
   - eg, credit card number stolen from e-commerce site.
4. **Denial of service**
   - Clients are no more served.


cPanel® - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.syria-news.com
Go
Done
Internet

Great Success !
Apache is working on your cPanel® and WHM™ Server

If you can see this page, then the people who manage this server have installed cPanel and WebHost Manager (WHM) which use the Apache Web server software and the Apache Interface to OpenSSL (mod_ssl) successfully. They now have to add content to this directory and replace this placeholder page, or else point the server at their real content.

ATTENTION!

If you are seeing this page instead of the site you expected, please **contact the administrator of the site involved.** (Try sending an email to <webmaster@*domain*>.) Although this site is running cPanel, WebHost Manager, and Apache software it almost certainly has no other connection to cPanel Inc. or the Apache Group. Please do not send mail about this site or its contents to cPanel Inc. or the Apache Group.

About cPanel:

cPanel is a leading provider of software for the webhosting industry. If you would like to learn more about cPanel please visit our website at http://www.cpanel.net/. Please be advised that cPanel Inc. is not a web hosting company, and as such has no control over content found elsewhere on this site.

About Apache HTTP Server:

The Apache HTTP Server is an open source web server which powers many of the worlds web sites. The Apache HTTP server is part of the Apache Group's many influential projects. Their efforts have helped shape much the world wide web, and they continue to be a dominating force in the web hosting industry.

Web Server Security: Types of attacks

1. Directory traversal
2. Script permissions
3. Directory Browsing
4. Default samples

Web Server Security: Types of attacks

1. **Directory traversal**
   - Is a method for accessing directories other than the allowed ones.
   - In Microsoft's IIS, if the OS XP is installed on drive c: and administrator didn't change the directory name, the default web site directory is c:\inetpub
   - Attackers can read file they are not meant to. For example
     - If the attacker try http://www.somesite.com/../autoexec.bat then the server may return the content of autoexec.bat.

Web Server Security: Types of attacks

2. **Script permissions**
   - In order to run server-side applications (eg, CGI, Perl, etc.), administrator must grant executable permission to the directory where these applications reside.
   - What happens if the admin grand permissions to the wrong directory?
   - Example: if the admin grants executable permission to c: then what happens if the attacker try http://www.somesite.com/../Windows/system32/cmd.exe%20%2fc%20dir

Web Server Security: Types of attacks

- The web server parse the request and execute ../windows/system32/cmd.exe /c dir
  ie, listing all files in the current directory.

– Attacker can execute commands that <u>delete or modify</u> files on the web server.

3. **Directory Browsing**

- If ***Directory Browsing*** is enabled, attacker can browse that directory and its subdirectories.

- Knowledge of the existence of some file can help attacker launching an attack.

Web Server Protection

1. Run web server service with Least privileges.
2. Install most recent security patches of server software.
3. Install most *recent security* patches of OS.
4. Secure other network services running on the same machine.
5. Delete unneeded applications.
6. Grant script permissions only to isolated directory containing the scripts in question.
7. Maintain adequate logs and backups.
8. Secure your web server using third-party security products: antiviruses, Firewalls, vulnerabilities scanners, input validation, etc.

5. Web browser Security

- Browser sends requests
  - May reveal private information (in forms, cookies)
  - Also sends other information that may be damaging:
    - IP address
    - OS
    - Browser version/type, etc.

- Browser receives information, code
  - May corrupt hosts by running unsafe code
  - Information may exercise a bug in the browser allowing arbitrary remote code execution.

Web browser Security

- Cookies
  - Cookie mechanism

- Mobile code
  - Java applet
  - JavaScript
  - VBScript

Web browser Security: Cookies

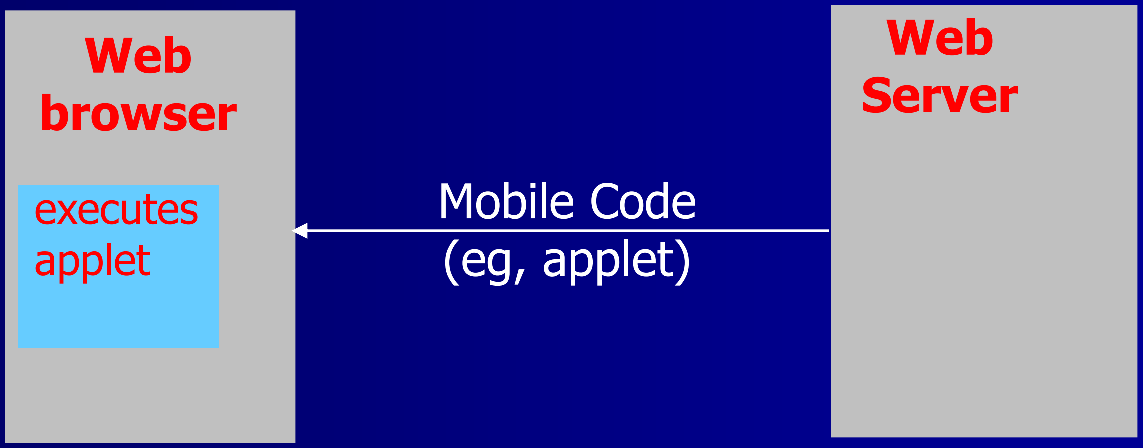
- HTTP is stateless. This causes problems in a lot of transactions that need a concept of a "session":
  - A customer wants to purchase an item online.
  - A customer logs onto their bank to pay bills
  - Sites like Yahoo allow users to customize their view of the portal
  - As the user jumps from web page to web page, the server can't keep track of whether it's the same user, or another user requesting the same page
  - Servers use cookies to keep track of their users.
- A cookie is a file created by an Internet site to store information on your computer
  - Once a cookie is saved on your computer, only the Web site that created the cookie can read it.

Web browser Security: Cookies

- Example: google's cookie
- PREF
  ID=186f76e084b84d56:TM=1193982844:LM=1193982844:S=O8OM9yhkCkr98Ej
  _
  google.co.uk/
  1536 //3081004544 // 30038711 //2452507808 // 29891852
  *
- Problems
  - Cookies maintain record of your browsing habits
    - May include *any* information a web site knows about you
  - Browser attacks could invade your "privacy"
  - Stealing someone's cookies may allow attacker to impersonate the victim:
    - Session hijacking

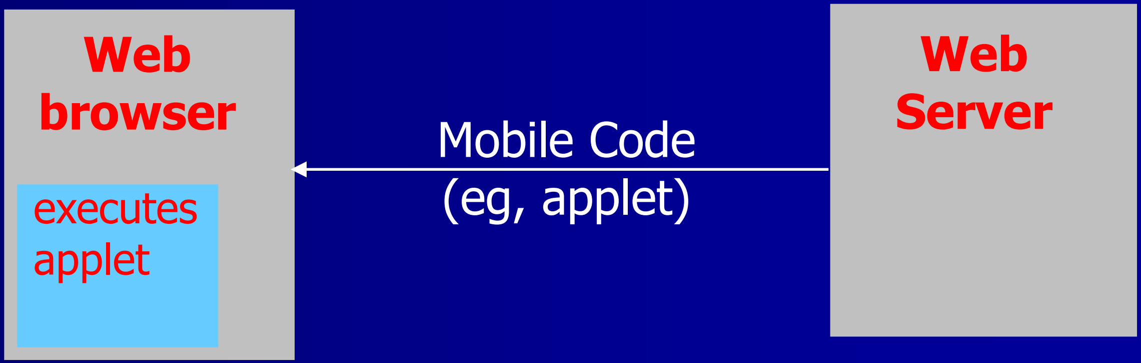
Web browser Security: Mobile Code

- Mobile code runs on clients’ machine.
- It’s an executable content (eg, applets).
- Things to do:
  - Protect machine from downloaded code.
  - Needs protection from content providers.
- Normal users are asked to make security decisions /policies.



Web browser Security: Mobile Code

- Web pages may contain executable code written in java, VB, Javascript, etc.
  - Those downloadable and executable code are called **mobile codes**
    - Run on clients' machines
- Mobile code may contain **malicious code**
- Many types of mobile codes:
  - Java Applets
  - ActiveX
  - JavaScript


Web browser
executes applet
Mobile Code (eg, applet)
Web Server