Web Issues

WWW

- Web browser: the client interface of WWW. A software application used to locate and display Web pages (I.e. Netscape Navigator, MS Internet Explorer, Opera, Motzilla, etc.)
- Web server: the server that stores and deliver Web page. Each web server has a IP address and possibly a domain name
- Plug-in: a software module that add a specific feature to the browser

WWW

Server

- CGI (Common Gateway Interface):CGI programs are the most common way for Web servers to interact dynamically with users. Many web pages that contain forms, for example, use a CGI program to process the form's data once it's submitted. In C, perl, Java, Visual Basic. They are *not* persistent

- Servlet: a small program that runs on a web server. The term usually refers to a Java applet. They are *persistent*

WWW

Browser

- Applet: A program designed to be executed from within another application. Unlike an application, applets cannot be executed directly from the operating system

- Active X: An ActiveX control can be automatically downloaded and executed by a browser. ActiveX is not a programming language, but rather a set of rules for how applications should share information. They work only with Microsoft and unlike Java applets, ActiveX controls have full access to the Windows operating system

WWW

- HTTP (Hyper Text Transfer Protocol) the underlying protocol used by the WWW.

- URI (Universal Resource Identifier) generic term for all types of names and addresses that refer to object on the WWW.
  - URL (Universal Resource Location) global address of the object
  - URN (Universal Resource Name) global name of the object

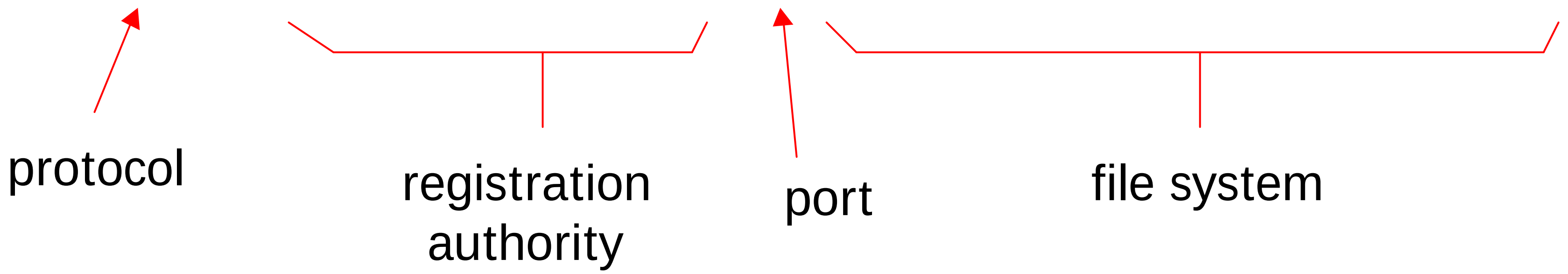
- HTML (HyperText Markup Language) the authoring language used to create documents on the web

  ex. <b><I>hello</b></I> ***Hello***
  - XML can be seen as a generalization of HTML

URLs

- They need to be unique. Partly enforced by domain registration authority partly by file system mechanisms.

http://www.few.vu.nl:80/crispo/teaching/doc.html



- Hierarchical names

URLs

$$\text{ftp://}\underbrace{\text{login:pwd}}_{\substack{\text{login name}\\ \text{password}}}\text{@}\underbrace{\text{www.few.vu.nl/}}_{\text{ftp server address}}$$

- Login and password are in clear!

HTTP

- HTTP is a request/response protocol
- HTTP is *stateless*

- HTTP specifies two authentication mechanisms: Basic and Digest Access Authentication.
  User-to-server authentication

HTTP

- Request header
  - FROM: contains e-mail address of the requestor $\rightarrow$ privacy issue
  - AUTHORIZATION: allows the server to know who the user is. Authorization and authentication as used as synonyms here

Basic Access Authentication

- Browser → Server: GET http://glimworm.cs.vu.nl/a.html

- Server → Browser: ..a.html 401 Unauthorized
  WWW-Authenticate: Basic realm="Restricted File Example"

- Browser → Server : GET http://glimworm.cs.vu.nl/a.html
  Authorization: Basic $\underbrace{\text{c7omsZuk=}}_{\text{Base64-encoded of cleartext}}$

Login and password are cached by the browser for the entire session

Base 64 encoding

- The Base 64 Alphabet Value

A-Z → 0-25 , a-z → 26-51, 0-9 → 52-61,
+ → 62, / → 63, (pad) → =

- 8 bits → 6 bit of a printable char
  - Es. ro = (01110010-01101111) → (011100-100110-1111--)= cmP=

ascii base64

Digest Access Authentication

- Browser → Server: GET http://glimworm.cs.vu.nl/a.html

- Server → Browser: ..a.html 401 Unauthorized
  WWW-Authenticate: Digest realm="Restricted File"
  nonce="012389"

- Browser → Server : GET http://glimworm.cs.vu.nl/a.html
  Authorization: Digest   username= "user1"
  realm = "Restricted File"
  nonce ="012389"
  uri = "a.htlm"
  response ="74yh6us783740ski9"

Digest Access Authentication

Response =

MD5 (MD5(A1):<nonce>:MD5(A2))

A1=<username>:<realm>:<passwd>

A2=<HTTP access method>:<uri>

Limitations: still based on passwords, thus subject to password attacks

Server configuration

Creating passwords file

Usage:

```
        htpasswd [-cmdps] passwordfile username
        htpasswd -b[cmdps] passwordfile username password
        htpasswd -n[mdps] username
        htpasswd -nb[mdps] username password
```

```
 -c  Create a new file.
 -n  Don't update file; display results on stdout.
 -m  Force MD5 encryption of the password (default).
 -d  Force CRYPT encryption of the password.
 -p  Do not encrypt the password (plaintext).
 -s  Force SHA encryption of the password.
 -b  Use the password from the command line rather than prompting for it.
```

Server configuration

Writing server configuration file (httpd.conf or .htaccess)

ex.

```
<Directory "C:/Program Files/Apache Group/Apache2/htdocs">
AuthType Basic
AuthName "Restricted Files Example"
AuthUserFile "C:/Program Files/Apache Group/Apache2/secure/passwords"
Require valid-user
</Directory>
```

Server configuration

- Server does not need to store password in clear $\rightarrow$ robust against breaks-in
- Replay attacks prevented by nonce used as a counter to avoid extra msg
- Server has to store <nonce, nonce count>
- No encryption of data $\rightarrow$ that’s why SSL/TLS. HTTP is tunneled in SSL/TLS

Server Side

- CGI programs are often configured to run with the same permissions of the web server (usually root) so they must be carefully checked.

- Servlet better performance and more secure due to Java native protection mechanisms (i.e. sandboxing)

Client side

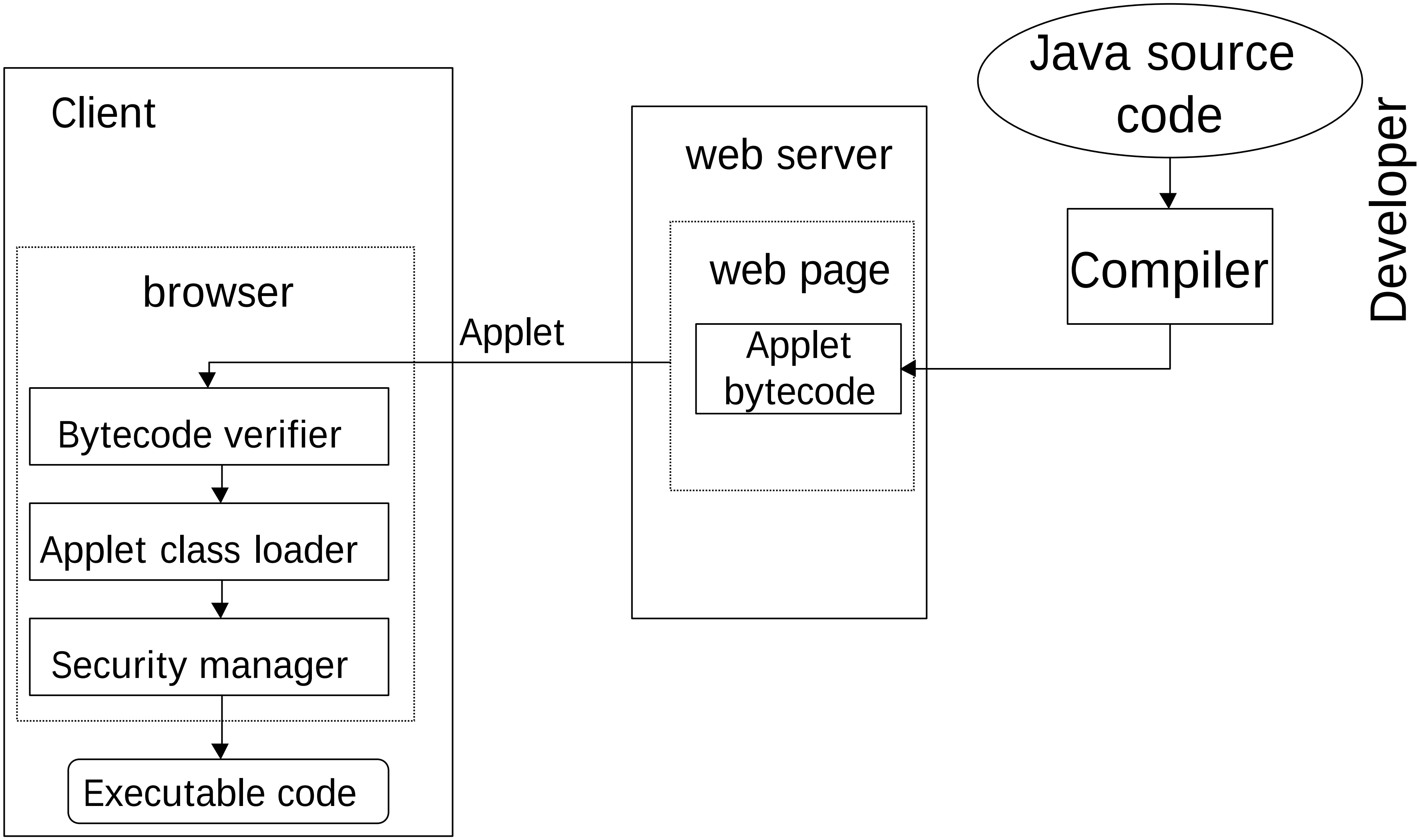
- Plug-ins depend on the browser (manufacturer and version) and the platform (operating system).
- Browsers support automatic downloading and execution of Java applets via web pages. Applets leave no footprint on the user’s workstation. The applet may have to be browser specific in order to access the keys.
- Active-X controls similar but worse because work only with MS and they have access to the entire operating system

Client side

- Plug-ins: user have to trust the author but no way to verify who he is

- Java Applets: sandboxing + signature
-
- Active-X controls: signature

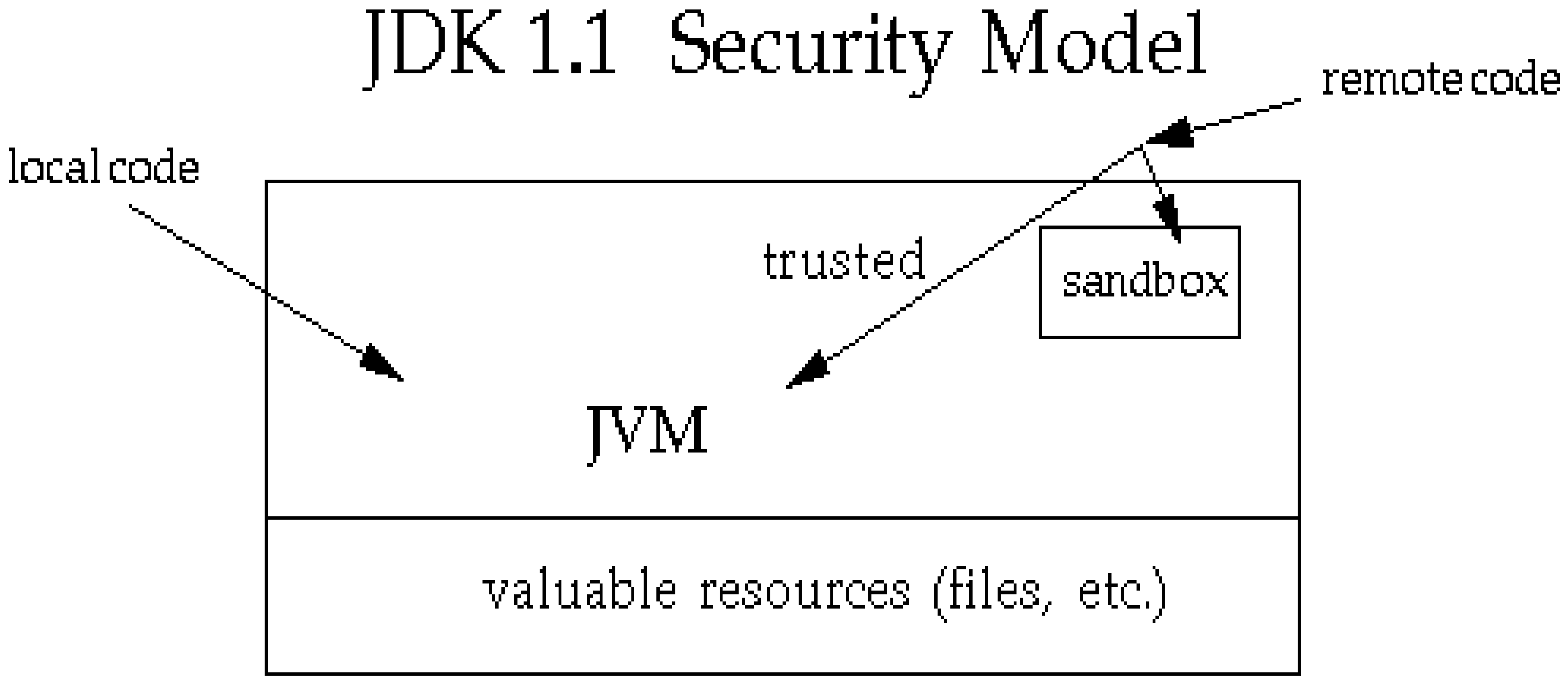
Java Applets



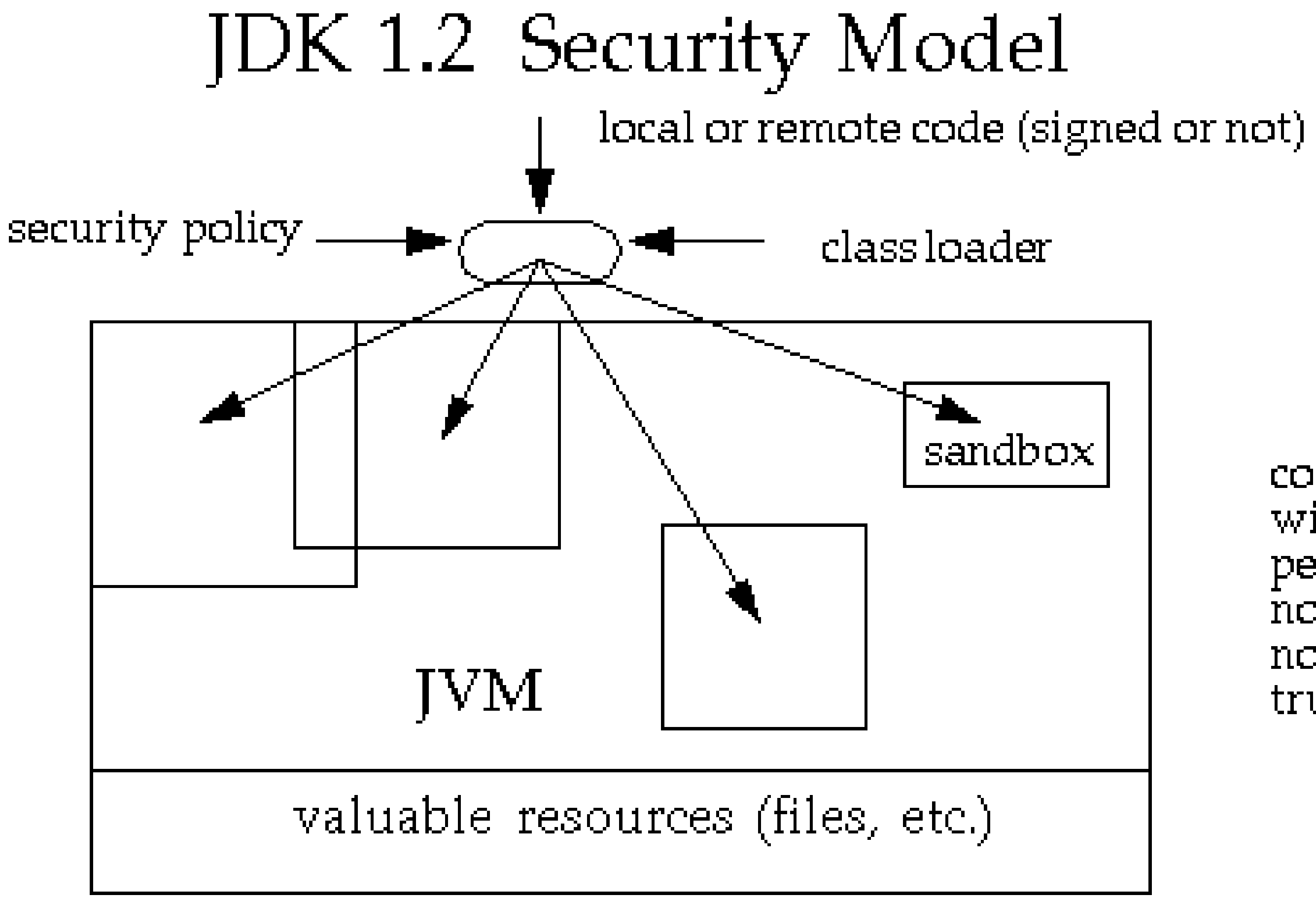
Java Components

- Byte Code Verifier
  - Syntactic check and static type checking (e.g., all operands have argument of correct type. All references to other classes are legal, etc.)
- Applet Class Loader
  - Name space where applet are executed
- Security Manager
  - reference monitor of the java security model

Sandbox


JDK 1.1 Security Model
remote code
local code
trusted
sandbox
JVM
valuable resources (files, etc.)

Sandbox


JDK 1.2 Security Model
local or remote code (signed or not)
security policy
class loader
sandbox
JVM
valuable resources (files, etc.)

codes run with different permissions, no built-in notion of trusted code

Applets Restrictions

- No access to user’s file system
- Cannot gain info about user’s name, e-mail addr, machine config, etc.
- Connecting only with the server of origin
- Can pop-up only “untrusted” windows
- Cannot re-configure the system (e.g., new class loader)

Browser Security Features

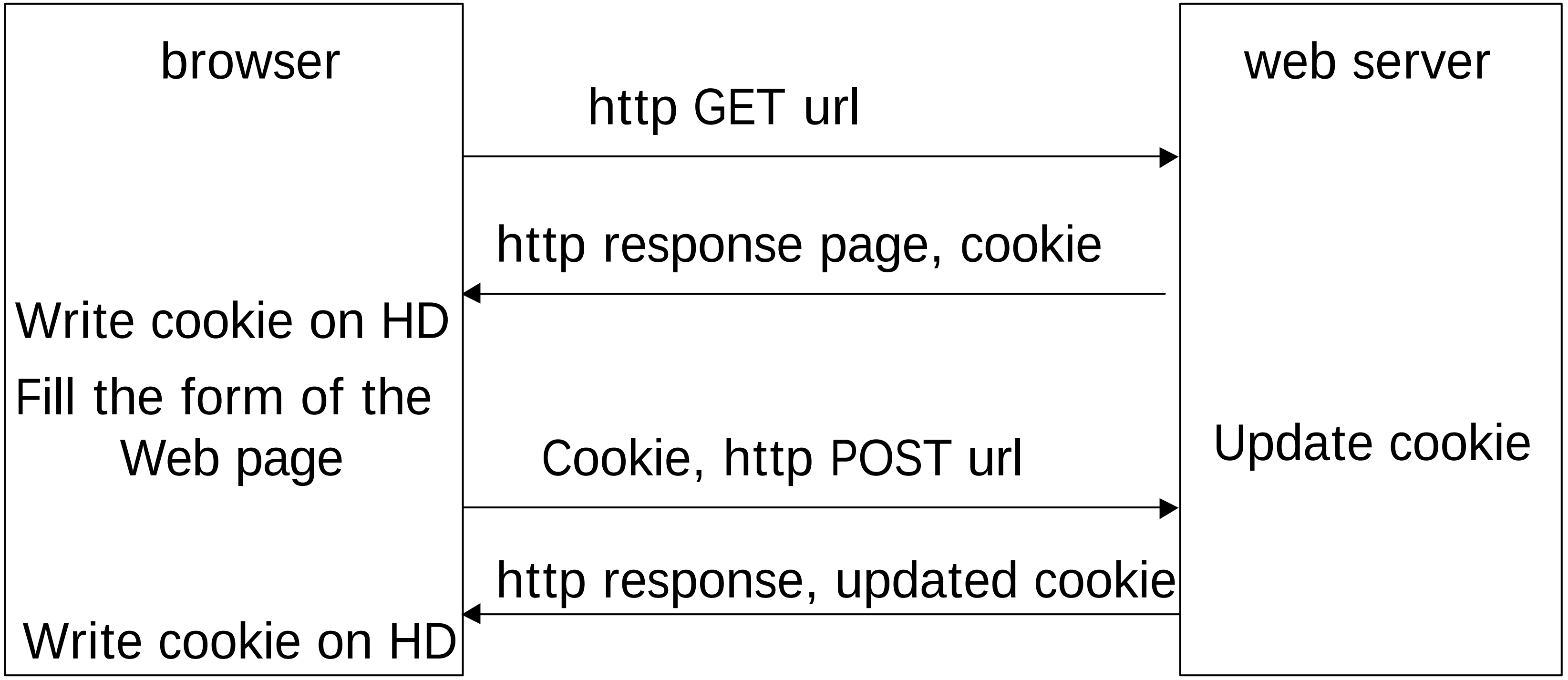
- <u>Browsers can:</u>
  - Validate signatures
  - Generate and store cryptographic keys
  - Enroll certificates
  - Store certificates
  - Retrieve credential from databases (via ldap)

- <u>Cannot</u>
  - Sign
  - Store evidence
  - Check timestamps

Cookies

- Providing a stateful HTTP by “using” the browser
- Enable the server to maintain user’s context across many requests
- A data structure created by the server for that user (browser) and stored at the client
- Save storage memory and searching time to servers

Cookies


browser
web server
http GET url
http response page, cookie
Write cookie on HD
Fill the form of the Web page
Cookie, http POST url
Update cookie
http response, updated cookie
Write cookie on HD

Cookies

- Cannot be bigger than 4Kbytes
- Textual files and not executable $\rightarrow$ safe
- Store information explicitly provided by the user (e.g., address, password, shopping list) or by the browser configuration (i.e. operating system, etc.)
- Encrypted

Cookies

String of text with these 6 attributes:

- The domain and path for which the cookie is valid
- The name of the cookie
- The value of the cookie
- The expiration date of the cookie
- Whether a secure connection needed to use the cookie

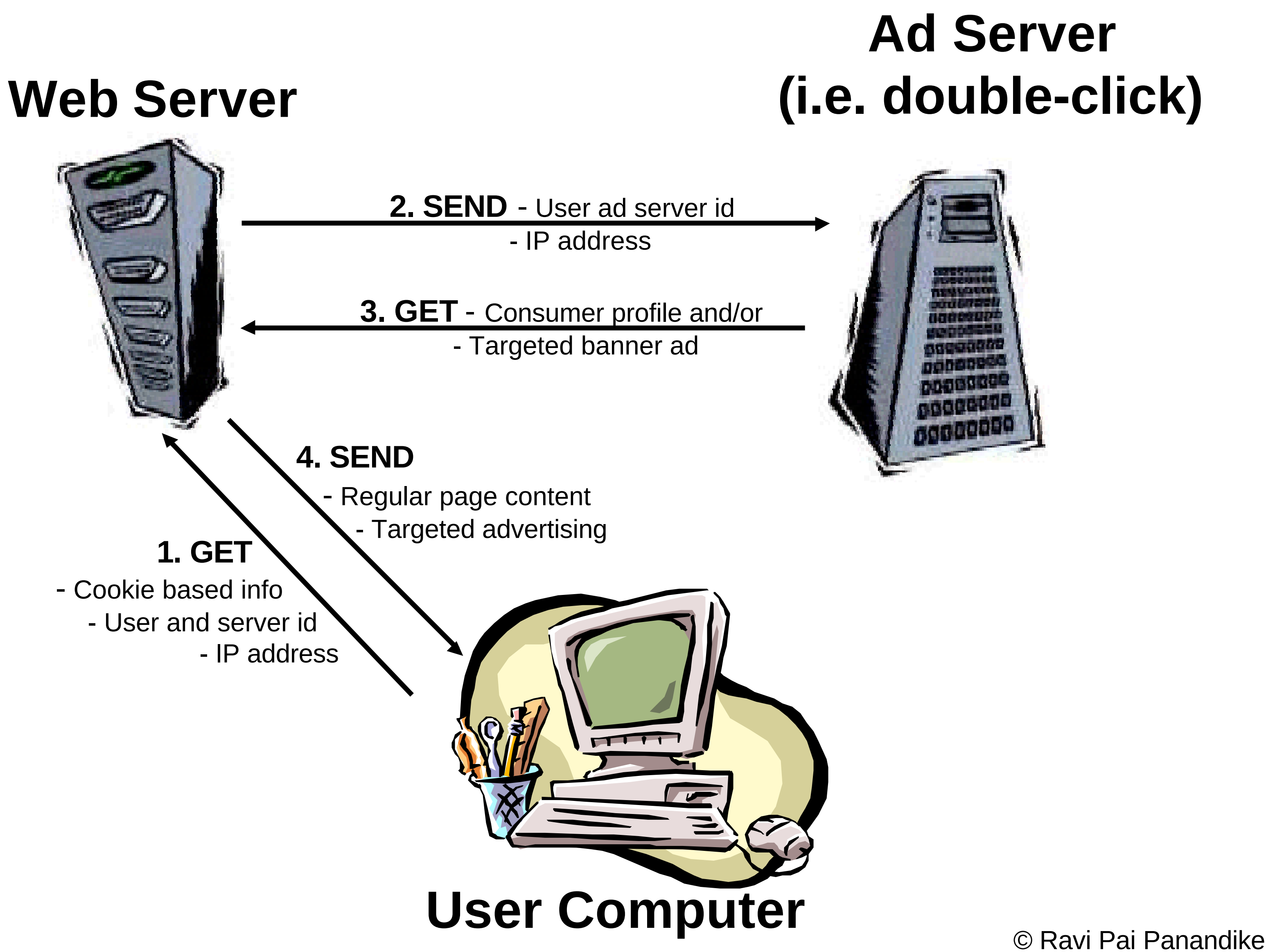
Cookies cannot

- Have automatic access to personal information like name, address, email
- Read or write data to hard disk
- Read or write information in cookies placed by other sites
- Run programs on your computer

Cookies can

- Store and manipulate any information you explicitly provide to a site
- Track your interaction with parent site such as pages visited, time of visits, number of visits
- Use any information available to web server including: IP address, Operating System, Browser Type
- Be read/written by other code (i.e.,Javascript) not only the browser

Privacy Issue



Privacy Issue

- Problem that cookies are only used to save searching time of servers but the data are actually kept in big database
  - Correlation problem
  - Transparent to user

Remedy: possibility to refuse cookies

The privacy problem is not due only to cookies but cookies make it worse