Communications of the Association for Information Systems

9-8-2001

Internet Privacy: At Home and At Work

Robert J. Boncella
Washburn University, zzbonc@washburn.edu

Follow this and additional works at: https://aisel.aisnet.org/cais

INTERNET PRIVACY - AT HOME AND AT WORK

Robert J. Boncella
Washburn University
zzbonc@washburn.edu

TUTORIAL

INTERNET PRIVACY - AT HOME AND AT WORK

Robert J. Boncella
Washburn University
zzbonc@washburn.edu

ABSTRACT

This article is a tutorial on Internet privacy.   The objective of the tutorial is to inform users of the Internet how and why their use of the Internet can be monitored.   The focus of the tutorial is Internet privacy at home and in the work place. The tutorial has three major sections: Technical Background, Privacy At Home, and Privacy At Work.

The At Home section is concerned with personal privacy infringement. This section details who would be interested in personal privacy infringement, for what purpose, and how they would accomplish it. The At Home section also discusses techniques on how to avoid privacy infringement.

The At Work section is concerned with employee surveillance. Topics discussed include who would be interested in this surveillance, for what purpose, and how this surveillance would be carried out.   The At Work section also discusses the concept of an Acceptable Use Policy (AUP).

Both of these sections require a bit of technical expertise to understand how Internet activity can be monitored. The Technical Background section and the Appendix review the client/server paradigm of web computing, the important details of the Hypertext Transfer Protocol (HTTP), and presents an overview of Uniform Resource Identifiers (URIs).   These concepts are necessary to understand how Internet activity can be monitored.

Keywords: internet privacy, cookies, web bugs employee surveillance, acceptable use policy

I. INTRODUCTION

In the confines of your office, study or cubicle the Web feels anonymous - IT IS NOT. Everything that one does on the Web can be monitored automatically.

Why monitor? A user's recreational web habits can be useful for marketing purposes. As a user surfs the web it is efficient marketing if the banner ads displayed to the user are targeted for that user. Monitoring a user's "click stream" allows for this "targeting" to take place.

In the workplace, knowing an employee's web behavior can be useful in determining an employee's productivity, maintaining an efficient use of an organization's bandwidth, enhance computer security, and avoid legal complications.

II TECHNICAL BACKGROUND

Three fundamental concepts allow the World Wide Web to be an efficient and effective means to deliver information. These concepts are the client/server method of computing (sometimes called the request/response paradigm); HTTP (Hypertext Transfer Protocol); and the URI (Uniform Resource Identifier) syntax and its related semantics. The request/response paradigm is used to implement the web browser – web server interaction. HTTP is used to coordinate and control this interaction and URIs are used to identify an information resource uniquely on the Web. When a user – through a web browser - requests a web page from a web server these three concepts combine in such a way that the information of who is requesting what from where can be automatically monitored and collected. If a user provides personal information (e.g. name, SSN, phone number) that information may be both monitored and collected as well.

For readers unfamiliar with how these three fundamental concepts (client server, HTTP, and URI) are implemented, Appendix I presents a short, illustrated discussion.

III. INTERNET PRIVACY AT HOME

PRIVATE INFORMATION: WHO WANTS IT

Web merchants, web page advertisers, and market researchers find it useful to know what web sites users visit, what pages they download and what pages they view next. These parties, e.g. DoubleClick (www.doubleclick.net), 24/7 Media (www.247media.com), and Engage (www.engage.com), are interested in users' "click streams". They can use this information to "target" the consumer using the Web and to provide them with advertising relevant to a particular user.

PRIVATE INFORMATION: HOW THEY GET IT

This information can be collected using several techniques: web server log files, cookies, and web bugs.

Server Log Files

Each time a client requests a resource, the server of that resource may record the following in its log files.

- The name and IP address of the client computer.
- The time of the request
- The URL that was requested.
- The time it took to send the resource.
- If HTTP authentication used; the username of the user of the client is recorded.
- Any errors that occurred
- The kind of web browser that was used.
- And most critical - the value of the referer header. This value (information) indicates the URL of the resource being requested and the URL that referred the client (user) to the requested resource.

Cookies

The use of cookies is a technique designed to overcome the "statelessness" of the HTTP protocol. HTTP is stateless because, when a web browser requests a resource from a web server, it establishes an Internet connection with that sever. Once the server sends the response to the browser's request, that connection is terminated. If the same web browser makes a request to the same web server at a later time that web server has no information regarding that browser's last request – *the web server does not remember the state of that web browser*. This failure to remember is a difficulty when, for example, the user is shopping on a web site and filling a shopping cart with items from different pages. Each web page requires a new connection between the browser and server.

To enable the server to remember the state of a browser (e.g. what items are in its shopping cart) the server sends a *cookie* to the browser when the browser makes its first request to that server. The browser then returns the cookie to the server whenever it requests another resource or service from that server.

A cookie can be thought of as a unique ID associated with that browser. This ID can be used to locate a data record stored on that server which may contain current and historical information about the web browser.

Alternatively, the state information maybe stored on the client and returned in the form of a cookie to the server when a new request is made by the web browser to that server.

Cookies are not part of the specification for HTTP version 1.1 but they are an extension of it. The interested reader is referred to the following URL to read about the details regarding the specification and use of cookies:

http://www.w3.org/Protocols/rfc2109/rfc2109.txt

Sidebar 1 shows the header syntax for cookies.

SIDEBAR 1
HTTP HEADER SYNTAX FOR COOKIES

The HTTP header syntax for cookies is:

Set-Cookie: <Name>=<Value>; expires=<Date>; domain=<Domain_Name>; Path=<Path>; secure

Attributes of the Cookie Header

<Name>=<Value> The only required name/value pair is the cookie name and its value e.g. Set-Cookie: custID=12345

expires=<Date> Indicates when cookie is no longer valid. When a cookie expires it should be removed from storage. If no date is specified then cookie expires at end of user session.

domain=<Domain> If the domain of a client request matches the *domain* attribute of a cookie, then the request's path is compared to the cookie's *path* attribute. If there is a match, the cookie is transmitted to the server along with the request.

path=<Path> The *path* attribute indicates the URLs within a domain for which the cookie is valid. If no *path* attribute is set in the Set-Cookie header, the path is assumed to be the same as the resource that is being returned by the server.

secure The *secure* attribute indicates that this cookie should be sent via a secure connection.

Examples of cookie use

A Response Message Example:
HTTP/1.0 200 OK
Server: Netscape-Enterprise/2.01
Content Type: text/html
Content Length: 87
Set-Cookie: userID=1234; domain=mysite.org; path=/cookie_info

A Request Message Example:
GET /login.html HTTP/1.0
User-Agent: Mozilla/4.02 [en] (Win95; I)
Accept: image/gif, image/jpeg, */*

Web Bugs

Given the HTTP protocol and how download web pages are rendered, a third party can exploit the cookie technology to track any user's click steam during a session. This is the idea of "web bug". A web bug can be used to determine a client's browsing profile based on their "clickstream"

As a web page is rendered by a browser, each URL embedded in an HTML tag on the web page causes a request to be sent by the client. Consider the following HTML tag:

<IMG SRC "http://ad.doubleclick.net/ … " WIDTH=1 HEIGHT=1 BORDER=0>

This tag is requesting an image from the server located in the doubleclick.net domain. The image size is specified in the tag to be one pixel wide and one pixel in height. In effect this image is invisible on the rendered web page but it does generate a GET method request message to the server in the domain doubleclick.net. Among other information, the GET method may contain a cookie issued to the client by the server in the domain of that URL and of course the referer header. The effect is the referer header value can be extracted by the server and associated with that cookie. Suppose every page rendered by a client requests the same URL located in the same HTML tag. This is the essence of a web bug - using referer values and associating them with a client's cookie for that server.

Once a browser renders a web page from a server that allows web bugs to be placed on that page it also either accepts a cookie from the server that serves out this web bug if it is the first time the client has requested the web bug or the client returns the cookie received from the web bug server - this can be called the "web bug cookie". In either case the web bug server can associate the value of subsequent referer headers with the cookie it has served to the client. This effect occurs every time the browser requests a resource from a server that allows the web bug from the web bug server to be placed on any of its pages.

That is, the browser will return the web bug cookie to the server that serves web

bugs. Ultimately a user's click stream can be monitored across web servers.

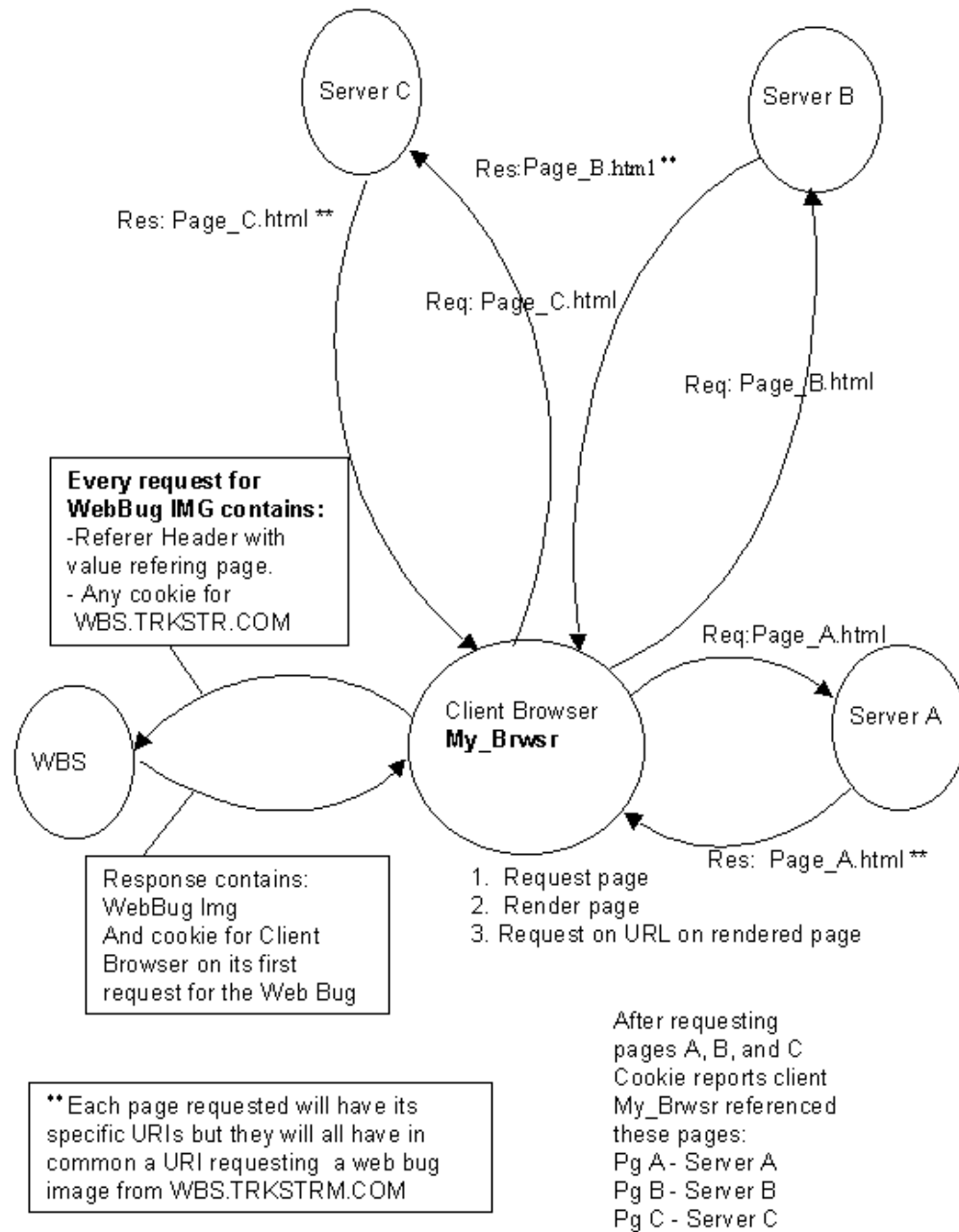This process is depicted in Figure 1. The process proceeds as follows.

Figure 1. Web Bug Process

The client My_Browser requests a web page - *Page_A.html* from server A.

Server A responds with *Page_A.html*. *Page_A.html* contains among other URLs and HTML tags the web bug located at server WBS. As My_Browser renders *Page_A.html* when it encounters the web bug it sends a request to WBS for the image of size one pixel by one pixel. Also as part of the request message will be the referer header whose value will be the URL for *Page_A.html*. The WBS server responds to this request by sending the image and also a cookie for client My_Browser served by WBS.

While viewing the rendered image of *Page_A.html* the user clicks on a link that requests *Page_B.html* located on server B. As My_Browser is rendering this page it encounters the HTML tag that contains the URL for the web bug located at WBS. My_Browser now sends a request for the web bug to WBS, The request message contains the cookie for WBS (provided by first request for the web bug) and the referer header whose value is the URL for *Page_B.html*. The server WBS can now associate *Page_B.html* at server B with the cookie. So far the server WBS has been able to associate with the My_Browser cookie the URLs of *Page_A.html* at server A and *Page_B.html*.

While viewing the rendered image of Page_B.html the user clicks on a link that requests *Page_C.html* located on server C. As My_Browser is rendering *Page_C.html* it encounters the HTML tag that contains the URL for the web bug located at WBS. My_Browser now sends a request for the web bug to WBS, The request message contains the cookie for WBS (provided by first request for the web bug) and the referer header whose value is the URL for *Page_C.html*.

The end result is that the server WBS has been able to track on the basis of the web bug cookie and the value of the referer header the user's sequence of pages visited.

The success of web bugs depend on three factors: the use of cookies, the value the referer header, and the ability of the web bug server to place the web bug on as many web pages that can be visited by a user. The first two are determined by the specification of HTTP. The third factor is determined by the ability of the web bug server to "buy" web bug space on a web pages.

SIDEBAR 2
ARE COOKIES ANONYMOUS?

As explained, a click stream can be associated with a specific cookie on a server. Cookies may be associated with a particular client, based on IP address but are assumed to not be associated with a particular user. As it happens E-mail readers (e.g. MS Outlook, Netscape Messenger) can render web pages.

Suppose a user receives a junk e-mail containing a web bug modified to contain user's e-mail address. An example of the web bug would be:

<img src="http://www.somedomain.com/webbug.gif?email=thisuser@theirdomain.com>

The HTML tag generates the following request to the server of the Web Bug

GET /webbug.gif?e-mail=thisuser@theirdomain.com HTTP/1.1
(other headers and their values)
…
Cookie: userID=1234

The web bug server which receives this request can now extract the e-mail address from the GET method line and associate it with the cookie being returned to the server. This assumes that the user has accepted or will accept a cookie from the web bug server.

Information Monitoring: How It Can Be Prevented

To prevent or at least manage this information collection, the user can use *anonymizing proxy servers* and/or *cookie cutters.*

Anonymizing Proxy Servers

Anonymizing proxy servers behave like regular proxy servers by carrying out the request on behalf of the client. The information sent to the web server is now information about the proxy server not about the requesting client. They work like normal proxy servers but they scrub any identification of the client from the request. Information scrubbed would be: cookies, a referer header, and the IP address of host making the request. In addition no log files are kept regarding a client's use of the proxy server.

The advantages of using an anonymizing proxy server is that they are effective and are transparent to both the client and server.

The disadvantages of using an anonymizing proxy server are slow web response time; cookies are unavailable; and the proxy server may not support secure sockets layer (SSL). The greatest disadvantage is the need for the client to trust the anonymizer

Cookie Cutters

Cookie cutters allow the user to determine if they want to accept a cookie from a specific server. Current versions of web browsers have these features as part of their implementation. Browsers may offer options for accepting cookies. Generally the options are: accept all cookies, reject all cookies, and warn before accepting a cookie and then give the user the option of accepting or rejecting the cookie. Some browsers offer an additional option - only accept cookies that are offered by the server of the rendered page. The intent here is to not accept cookies from a server that serves web bug.

If the "accept all cookies" option is chosen, a user can remain somewhat private by doing a clean sweep and remove files that contain cookies, the history of sites visited, and clear the browser cache which would contain the content of pages most recently visited. This action of a clean sweep is recommended if a user wish to insure privacy regardless of what cookie cutter method the user chooses. There are products available, both shareware and commercial, that perform the "clean sweep" suggested.

IV INTERNET PRIVACY AT WORK

The topic of Internet privacy at work focuses on employee surveillance. An organization conducts surveillance of how their employees use the Web and their e-mail for four reasons:

- employee productivity
- computer security
- wasted bandwidth
- legal liability

WHY CORPORATIONS CONDUCT WEB SURVEILLANCE

It has been estimated that US corporations lose more than $54 billion a year because of non-work related employee use of the web. [Conry-Murray, 2001]. In addition to productivity loss, corporation network bandwidth is reduced when employees use the web and e-mail for non-work related activities.

Computer security can be compromised if an employee either intentionally or unintentionally downloads a virus or opens an e-mail that contains a Trojan Horse program as an attachment..

Finally legal liability can occur when employees improperly use web resources. Improper use ranges from copyright infringement (e.g. downloading and installing copyrighted software) to sexual harassment issues associated with web pages containing pornographic content, and inappropriate e-mail that promote a hostile work environment.

HOW CORPORATIONS CONDUCT WEB SURVEILLANCE
Monitor Browser Activity

Employees of medium to large size corporations will be using Internet access devices attached to a corporate network. This arrangement implies the employee must access the Internet through a proxy server managed by the organization. This arrangement makes it a simple matter both to restrict and to monitor an employee's use of the web. A number of software tools can limit

access to specified web resources. In addition these tools can collect data on a user's "click steam" by means of a proxy server's log file

In smaller size organizations, where employee access is not managed through a proxy sever, an employee's web and e-mail use can be monitored by viewing the local data files associated with the user's web browser and e-mail client. For example, in Netscape Navigator, the cache file, history file, contents of the location bar and cookies file; and the inbox, sent, trash, and drafts file of Netscape Messenger can all be monitored.

E-MAIL

For medium to large size organizations, e-mail use can be monitored and filtered for mail messages that contain malicious mobile code in the message or in the attachments to the message. Examples of malicious code include Trojan Horses, computer viruses, spreadsheet or word processing macros, and executable scripts.

Also e-mail content can be monitored by looking for key words related to racist or sexual harassing remarks. Additional content scanning can be based on key words relating to confidential corporate data.

INTRUSION DETECTION SYSTEMS

An intrusion detection system is software that monitors and logs almost every aspect of a user's interaction with a host. The host being monitored can be stand alone or on a network. Intrusion detection systems, in essence, create an audit trail of computer use. Installed on a stand-alone host they can log user keystrokes. Installed as networked based software, they can record application keystrokes per user

In addition intrusion detection systems can be used to monitor access and use of a organization's intranet. They can be used to detect incoming and outgoing access to the Internet.

REMOTE CONTROL PROGAMS

Remote Control Programs are another method of monitoring an employee's use of the computer system in general and web in particular. Programs of this type allow control of a remote host but for surveillance purposes they can redirect the video display of the remote host to another host. In effect the employer can view in real time a copy of what the employee is viewing. Examples of these programs would be pcAnywhere or Citrix's ICA Client.

HOW CAN EMPLOYEES MANAGE CORPORATION WEB SURVEILLANCE?

Employees will not be able to manage corporate web surveillance but they should be made aware of it. Specifically, employees should know the Acceptable Use Policy (AUP) for computing resources - if one is provide by the organization. Part of that AUP should be a specification of the organization's Internet Access Policy (IAP). If no AUP is provided to an employee then the employee should request the AUP. If the organization does not have an AUP in place a process should be started to develop an AUP for the organization.

Employee web surveillance is effective and accepted if the employee is aware of an Institution's policy for web surveillance in particular and computer use in general. These policies - AUP and IAP - should specify, in plain language what is acceptable and not acceptable. Furthermore the IAP should provide specific examples of Do's and Don'ts

V SUMMARY

In the confines of your office, study or cubicle the Web feels anonymous - IT IS NOT. Everything that you do on the Web can be monitored automatically. If you use the web then you need to be aware of the how and why this activity can be monitored.

Editor's Note: This article is based on a tutorial presented at AMCIS 2001 in Boston, MA on August 3, 2001. The article was received on August 18, 2001 and published on September 8, 2001.

APPENDIX
WEB TECHNOLOGIES

INTRODUCTION

This appendix provides an overview for those readers who are unfamiliar with the web technologies of client/server computing, the Hypertext Transfer Protocol (HTTP), and Uniform Resource Identifier (URI) syntax and semantics.

CLIENT/SERVER METHOD OF COMPUTING

As depicted in Figure A1, client/server computing is a means of implementing a request/respond paradigm of computing. The client initiates the session by requesting a connection with the server. Once this connection is established, the client sends its request using the syntax of the protocol of the service it is requesting. The service type the client is requesting is specified in the URI of the request. The type of service requested is specified by the protocol portion of the URI (see next subsection for details).

UNIFORM RESOURCE IDENTIFIERS

The formal term used to specify a resource on the Internet is Uniform Resource Identifier (URI). However a web resource is generally referred to by a combination of its location – URL (Uniform Resource Locator) and its name – the URN (Uniform Resource Name).

Examples of the general syntax of these URI strings are:

protocol://host[:port]/url-path

protocol://username:password@host/url-path

Where:

"p*rotocol*", names the service being requested

"*host*", specifies either the IP address of a computer or the DNS name of the computer;
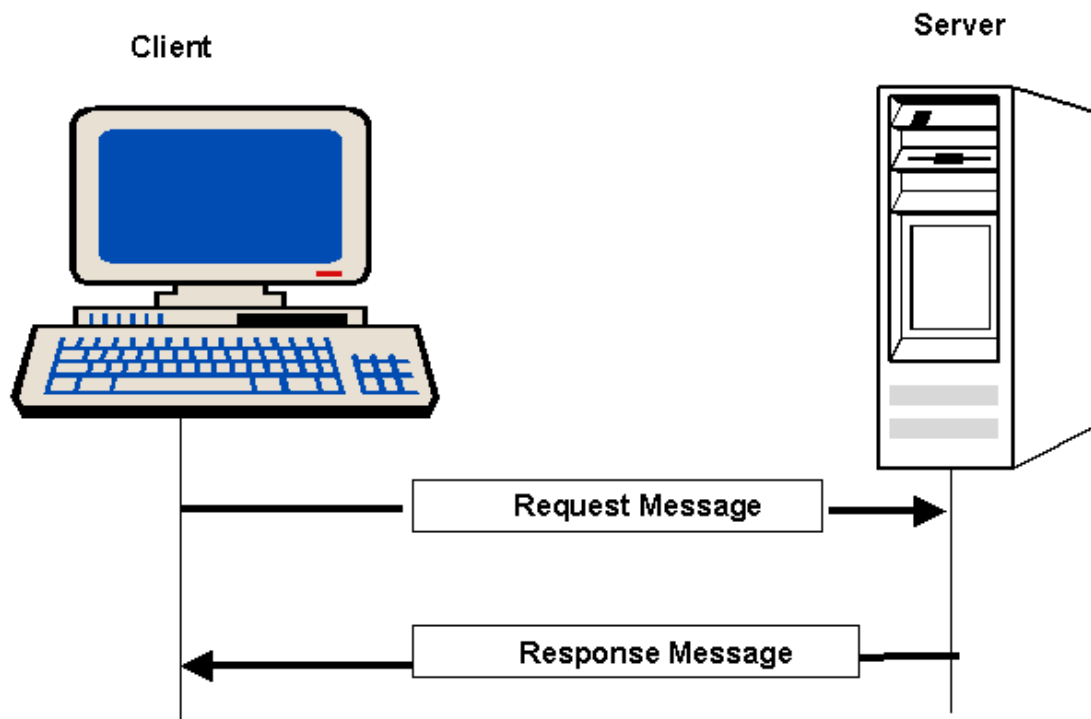
Figure A1. Client/Server Architecture

"*port*", – an optional specification –specifies where the service being requested can befound on the server;

and

"*url-path*" , specifies the name – in terms of directory, subdirectory, file - of the resource on the named host.

Two specific protocol examples are:

http://host[:port]/path/url-path[?query_string]

ftp://username:password@host[:port]/url-path

Examples of using these protocols are:

http://www.webcrawler.com:80/cgi-bin/WebQuery?searchText=privacy

ftp://anonymous:MyEmailAddress@ftp.netscape.com/

HYPERTEXT TRANSFER PROTOCOL – HTTP

HTTP is the most frequently used protocol on the Web. Inherent in its design are the causes of privacy threats. HTTP is a connectionless and stateless protocol where once a connection is established a client sends a request to a server and the server responds to that request.

HTTP is connectionless because when a client opens a connection to a server and sends its request. The sever responds to that request and *CLOSES* the connection.

HTTP is stateless because both the client and the server have no memory of prior connections. In particular a server cannot distinguish one client request from another. For example, if a server receives two consecutive requests from the same client, the server behaves as if they are requests from two different clients.

The request and response messages between the client/server pair are formed according to an accepted standard protocol - the HTTP protocol.

Each message type contains three sections;

- a request or status line;

- a header section, composed of three subsections and

- an optional body depending upon the method in the request line of the request message.

Figure A2 shows the format of HTTP requests and responses.

The specification of the HTTP protocol version 1.1 is given at:

http://www.w3.org/Protocols/rfc2616/rfc2616.html

and here after this specification will be referred to as rfc2616. We present an overview of this protocol that should be sufficient for the reader to appreciate the effect of the protocol specification on Internet privacy
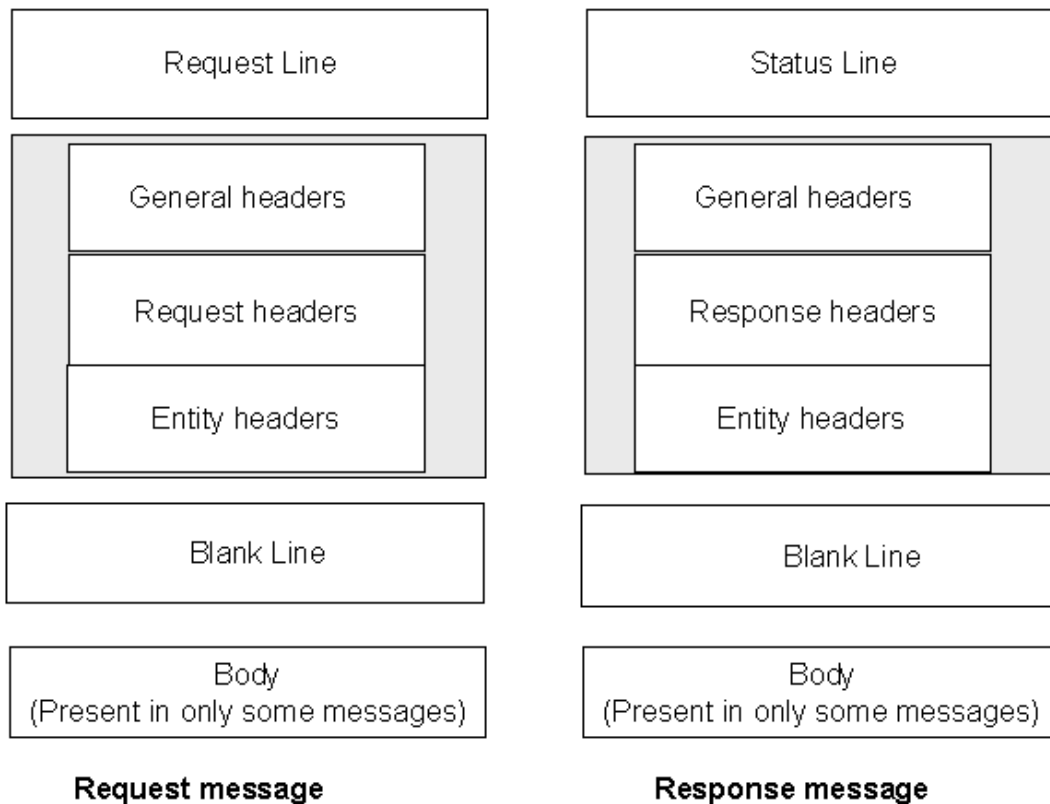
Figure A2. HTTP Protocol General Syntax

Request Line and Response Line

The syntax of the request line and examples of its use are given in Figure A3. Eight different methods or request types are specified for HTTP 1.1. We will only be concerned with the GET, HEAD, and POST methods.
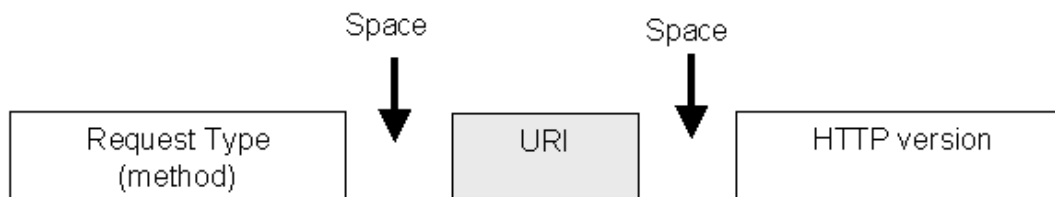
Figure A3 Request Line Syntax

The following are examples of the request line syntax:

GET /login.html HTTP/1.1

POST /login.html HTTP/1.1

GET /login.html?username=zzbonc&password=demo1 HTTP/1.1

The above request line is generated by the URL typed in the location bar of a web browser:

http://www.washburn.edu/login.html?username=zzbonc&password=demo1

The GET method means retrieve whatever information (in the form of a body of the response method) is identified by the Request-URI. The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response. And The POST method is used to request the server accept the body enclosed in the request. The POST method is commonly used to provide user input to the server. For more details about these methods and others the reader is referred to rfc2616.

The syntax of the status line and examples of its use are given Figure A4. The meaning of the response line gives the status of the client's request. There 40 status codes for specified for HTTP version 1.1 For more details about these status codes the reader is referred to rfc2616.
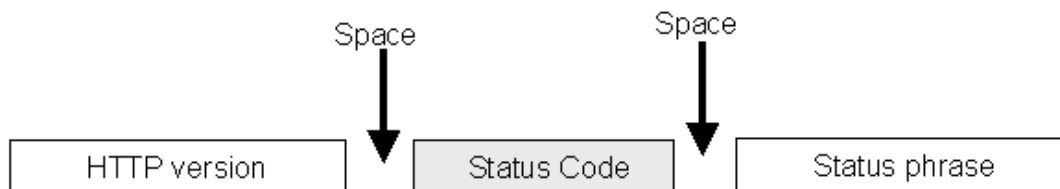
Figure A4 Response Line Syntax

The following are examples of response line syntax:

HTTP/1.1 200 OK

HTTP/1.1 401 Unauthorized

HTTP/1.1 404 Not Found

Header Section

The HTTP protocol specifies several types of headers. These are:

- the *general type* which provide general information about the message.

- the *request type* which specifies the client's configuration and preferred document format,

- the *response type* which specifies the server's configuration and information about the response,   and

- the *entity type* which provides information about the body of the document.

The syntax and examples of the header section is given in Figure A5. For more details on the types of headers the reader is referred to rfc2616.
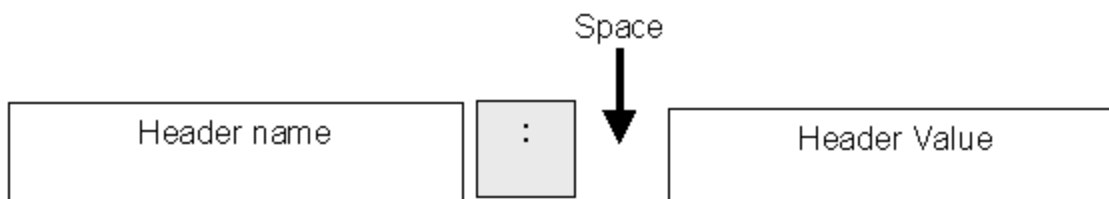
Figure A5. Header Syntax

The following table shows examples of Header syntax:

Type of header	Example	Meaning
General	Date: Sat, 11 Aug 2001 23:59:58 GMT	
Request	*From: zzbonc@washburn.edu*	the e-mail address of the client generating the request.
Referer:	http://www.washburn.edu/myhomepage.html	gives the address that contained (or hyperlink) to the currently requested resource. This header indicates the HTML page that referred the client to the requested resource
Accept:	image/gif .	The Accept request-header field can be used to specify certain media types which are acceptable for the response
Response	*Server: Apache/1.2.5*	the name and version of the server software. This can be useful to someone intent on hacking the server.
Entity	*Content-Type: text/html*	Indicates the MIME type of the request's data
	Content-Length: 1234	Indicates the length in bytes of the body portion of the request or response.

Note that in the case of the referrer, the information is useful in tracking the client's path through the web. The misspelling of referrer is correct. It was originally misspelled in the specification and has remained so.

Body Section

When a GET or HEAD method is used with the request message, the body section is empty. When the POST method is used with the request message, the body of the section contains client input information.

In response to the POST or GET method, the response message body generally contains a document that satisfies the client's request. In response to a HEAD request method, the body of the response is empty.

Examples of Client/Server Computing Using HTTP

Figures A6, A7, and A8 present examples of client/server communication using HTTP.

In the HTTP client/server example in Figure A-6, the client is using the GET method to request an image from the webbug server discussed in Section III. The response message contains all the header information as well as the 1x1 pixel image to be displayed in the browser. Once again an image of this size is invisible to the user.

Client

Server

Request (Get Method)

GET http;//wbs.trkstrm.com/webug.gif HTTP/1.1
Accept: image/gif
Accept: image/jpeg

Response

HTTP/1.1 200 OK
Date: Mon, 08-Jan-01 13:15:14 GMT
Server: WBS
MIME-version: 1.0
Content-length: 2048

(Body of document)
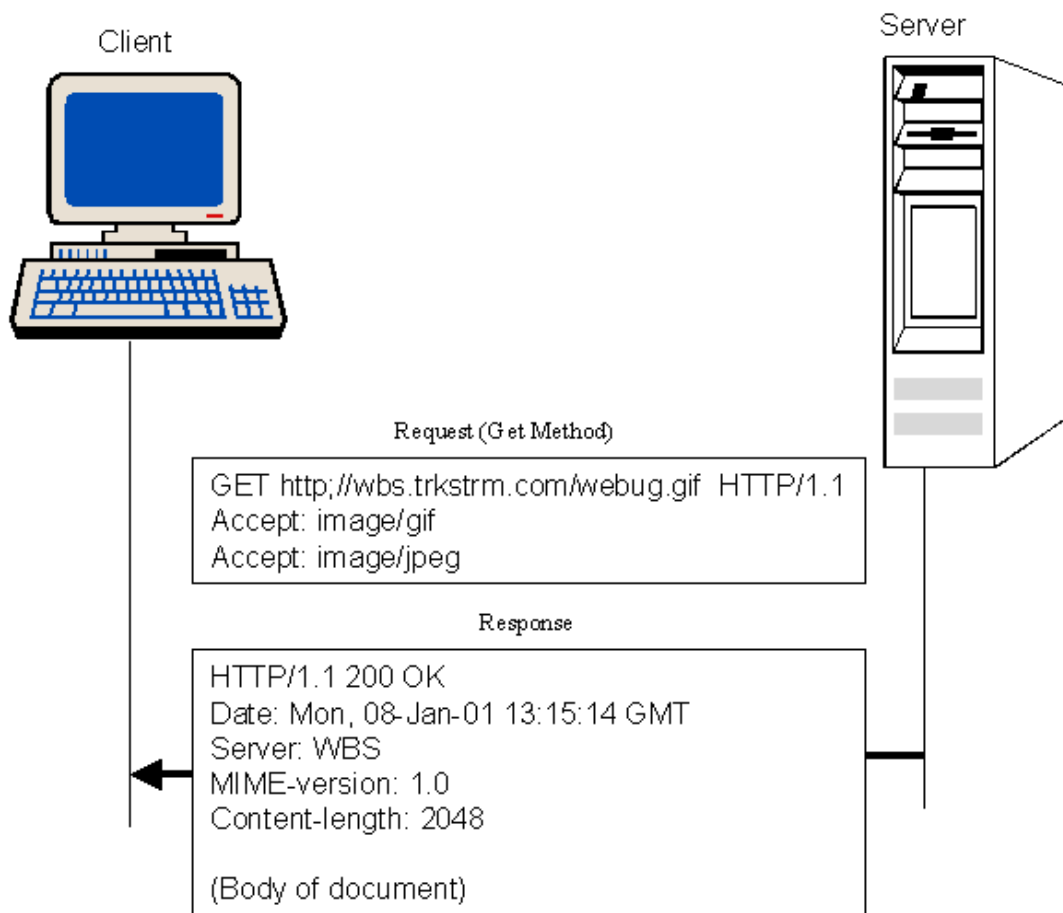
Figure A6. Example of Client/Server Communications Using HTTP
(GET Method)

Client

Server

Request (Head method)

HEAD usr/user/file1.html HTTP/1.1
Accept: */*

Response

HTTP/1.1 200 OK
Date: Mon, 08-Jan-01 13:15:14 GMT
Server: WBS
MIME-version: 1.0
Content-type: text/html
Content-length: 1020
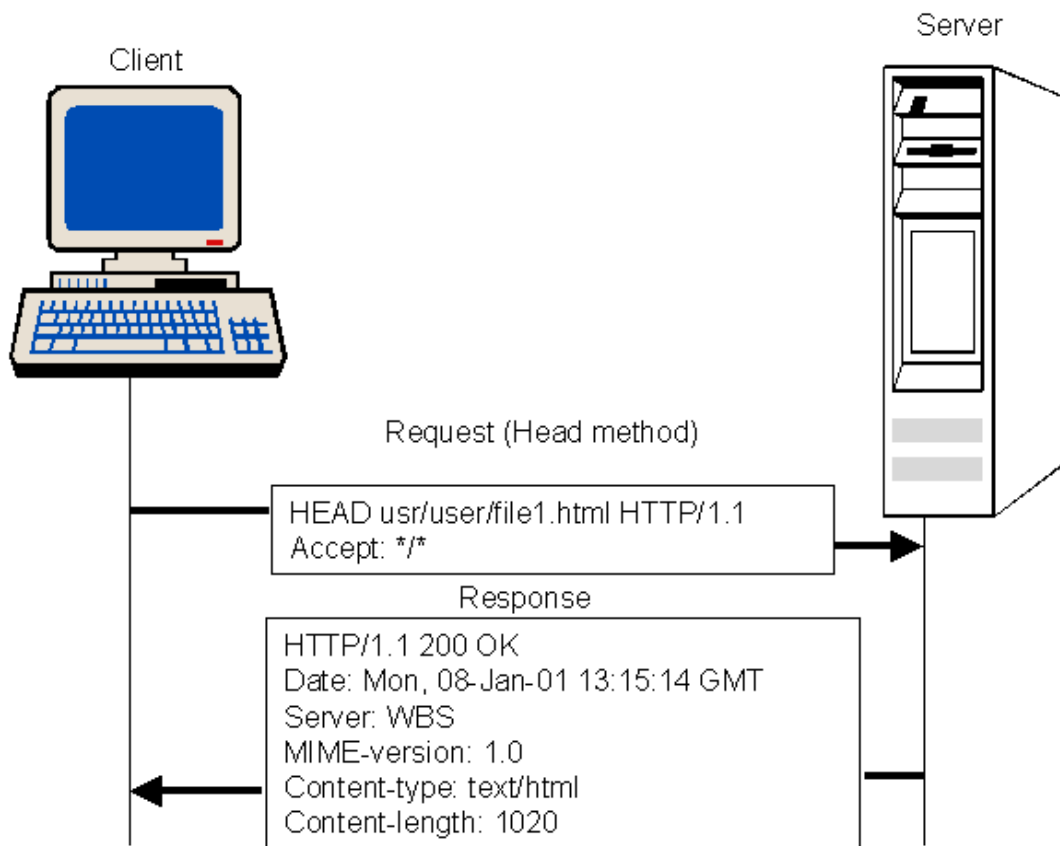
Figure A7. Example of Client/server Communications Using HTTP
(HEAD Method)

In this HTTP client/server example the client is using the HEAD method to retrieve the header information about the resource being requested. The response message contains only this information and not the resource requested.
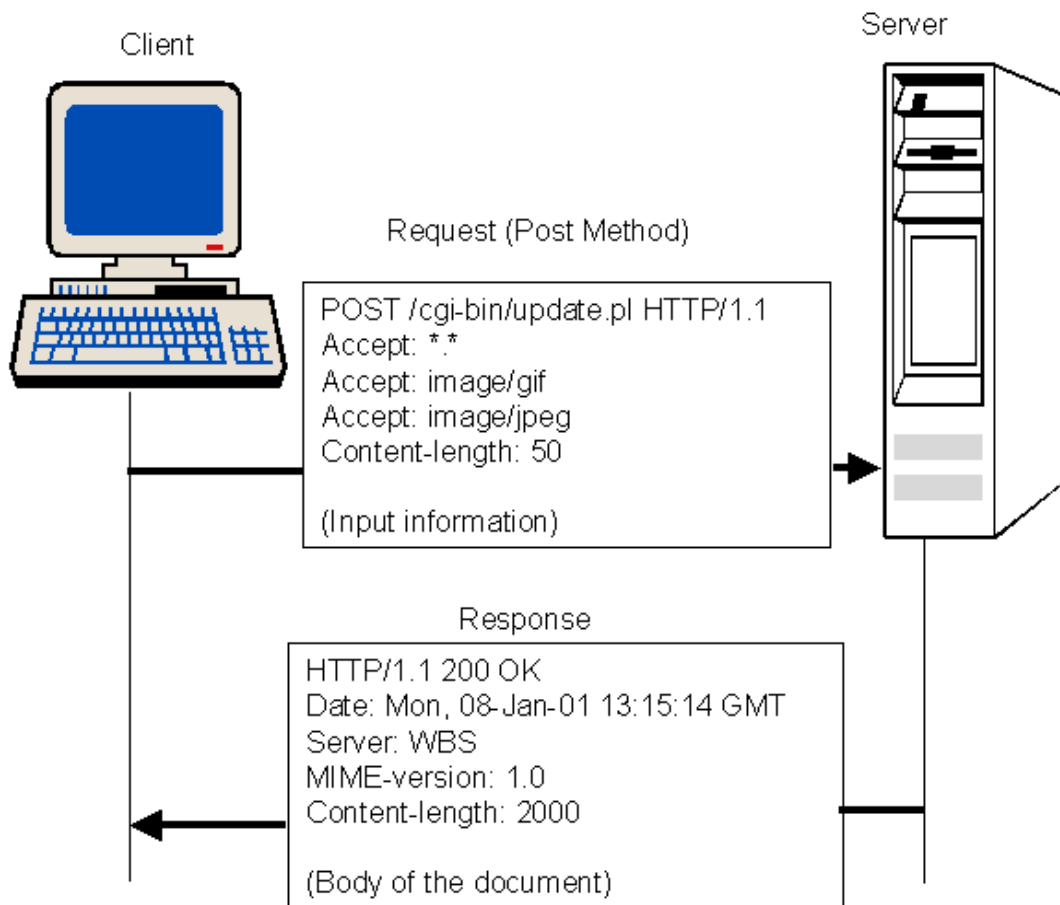
Figure A8. Example of Client/Server Communications Using HTTP
(POST Method)

In this HTTP client/server example the client is using the POST method to provide input to a program (update.pl) specified in the method line of the request message. The request message contains a body section that provides a list of the parameters the server program expects and their associated values. The response message may contain the results of the program if required or perhaps an acknowledgement of receiving the input.

REFERENCE

Conry-Murray, Andrew (2001) "The Pros and Cons of Employee Surveillance", *Network Magazine*, (12)2, pp. 62-66.

BIBLIOGRAPHY

EDITOR'S NOTE: The following bibliography contains the address of World Wide Web pages. Readers who have the ability to access the Web directly from their word processor or are reading the paper on the Web, can gain direct access to these references. Readers are warned, however, that
1. these links existed as of the date of publication but are not guaranteed to be working thereafter.
2. the contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.
3. the authors of the Web pages, not CAIS, are responsible for the accuracy of their content.
4. the author of this article, not CAIS, is responsible for the accuracy of the URL and version information

Angel, J. (2000) "Too Many Cookies Are Bad for You", *Network Magazine*, (15)6, pp.106-112.

Benassi, P. (1999) "TRUSTe: An Online Privacy Seal Program", Communications of the ACM, (42)2, pp.56-59.

Clark, E. (2000) "Privacy on the Internet", *Network Magazine*, (15)6, pp.99-100.

Clarke, R. (1999) "Internet Privacy Concerns Confirm the Case for Intervention", Communications of the ACM, (42)2, pp. 60-67.

Conry-Murray, A.(2001) "The Pros and Cons of Employee Surveillance", *Network Magazine*, (12)2, pp. 62-66.

Cranor, L. F. (1999) "Internet Privacy", Communications of the ACM, (42)2, pp. 29-31.

Dalton, C. E, (2001) "Preventing Corporate Network Abuse Gets Personal", *Network Magazine*, (12)2, pp. 56-60.

Davis, T. and D. Royce (2001) "The Law of the LAN: Monitoring Employees' Electronic Communications", *Network Magazine*, (12)2, pp. 50-54.

Dornan, A. (2000) "Internet Indiscretions", *Network Magazine*, (15)6, pp.100-105.

Garfinkel, S. and G. Spafford. (1997) *Web Security & Commerce*, Cambridge, MA: O'Reilly and Associates.

Goldschlag, D., M. Reed, and P. Syerson (1999) "Onion Routing for Anonymous and Private Internet Connections", Communications of the ACM, (42)2, pp. 39-47.

Mulligan, D. and A. Schwartz (2000) "Your Place or Mine?: Privacy Concerns and Solutions for Server and Client-Side Storage of Personal Information", *Toronto, ON, Canada: Proceedings of the Tenth Conference on Computers, Freedom, and Privacy: Challenging the Assumptions April 4-7, 2000*, pp. 81-84.

Reagle, J. and L.F. Cranor (1999) "The Platform for Privacy Preferences", Communications of the ACM, (42)2, pp. 48-55.

Reiter, M. and A.D. Rubin (1999) "Anonymous Web Transactions with Crowds", Communications of the ACM, (42)2, pp. 32-38.

Stein, L. D. (1988) *Web Security: A Step-by-step Reference Guide*, Reading, MA: Addison-Wesley

Treese, W (2000) "Data Collection and Consumer Privacy", *Networker*, December 2000, pp. 9-11

W3C, (2001) "P3P 1.0: A New Standard in Online Privacy", http://www.w3.org/P3P/brochure.html

Wadlow, T. A. (2000), *The Process of Network Security: Designing and Managing a Safe Network*, , Reading, MA: Addison-Wesley.

ABOUT THE AUTHOR

Robert J. Boncella (http://www.washburn.edu/cas/cis/boncella) is Professor of Computer Information Science at Washburn University, Topeka, KS.. Dr. Boncella has a joint appointment in the Computer Information Sciences Department,where he conducts classes in Data Communications and Computer Networks, and in the School of Business, where he offers instruction on Computer Based Information Systems in the school's MBA program.

He holds a Ph.D. and Masters degrees in Computer Science from the University of Kansas and a Master of Arts in Philosophy from The Cleveland State University. He is a member of ACM, AIS, AAAI, and IEEE.

His current areas of interest are web based information systems, intelligent agents and decision making under uncertainty as well as computer security and privacy.