

part of [\*Hypertext Transfer Protocol -- HTTP/1.1\*](#)  
*RFC 2616 Fielding, et al.*

## 15 Security Considerations

This section is meant to inform application developers, information providers, and users of the security limitations in HTTP/1.1 as described by this document. The discussion does not include definitive solutions to the problems revealed, though it does make some suggestions for reducing security risks.

### 15.1 Personal Information

HTTP clients are often privy to large amounts of personal information (e.g. the user's name, location, mail address, passwords, encryption keys, etc.), and **SHOULD** be very careful to prevent unintentional leakage of this information via the HTTP protocol to other sources. We very strongly recommend that a convenient interface be provided for the user to control dissemination of such information, and that designers and implementors be particularly careful in this area. History shows that errors in this area often create serious security and/or privacy problems and generate highly adverse publicity for the implementor's company.

#### 15.1.1 Abuse of Server Log Information

A server is in the position to save personal data about a user's requests which might identify their reading patterns or subjects of interest. This information is clearly confidential in nature and its handling can be constrained by law in certain countries. People using the HTTP protocol to provide data are responsible for ensuring that such material is not distributed without the permission of any individuals that are identifiable by the published results.

#### 15.1.2 Transfer of Sensitive Information

Like any generic data transfer protocol, HTTP cannot regulate the content of the data that is transferred, nor is there any a priori method of determining the sensitivity of any particular piece of information within the context of any given request. Therefore, applications **SHOULD** supply as much control over this information as possible to the provider of that information. Four header fields are worth special mention in this context: Server, Via, Referer and From.

Revealing the specific software version of the server might allow the server machine to become more vulnerable to attacks against software that is known to contain security holes. Implementors **SHOULD** make the Server header field a configurable option.

Proxies which serve as a portal through a network firewall **SHOULD** take special precautions regarding the transfer of header information that identifies the hosts behind the firewall. In particular, they **SHOULD** remove, or replace with sanitized versions, any Via fields generated behind the firewall.

The Referer header allows reading patterns to be studied and reverse links drawn. Although it can be very useful, its power can be abused if user details are not separated from the information contained in

the Referer. Even when the personal information has been removed, the Referer header might indicate a private document's URI whose publication would be inappropriate.

The information sent in the From field might conflict with the user's privacy interests or their site's security policy, and hence it **SHOULD NOT** be transmitted without the user being able to disable, enable, and modify the contents of the field. The user **MUST** be able to set the contents of this field within a user preference or application defaults configuration.

We suggest, though do not require, that a convenient toggle interface be provided for the user to enable or disable the sending of From and Referer information.

The User-Agent (section [14.43](#)) or Server (section [14.38](#)) header fields can sometimes be used to determine that a specific client or server have a particular security hole which might be exploited. Unfortunately, this same information is often used for other valuable purposes for which HTTP currently has no better mechanism.

### 15.1.3 Encoding Sensitive Information in URI's

Because the source of a link might be private information or might reveal an otherwise private information source, it is strongly recommended that the user be able to select whether or not the Referer field is sent. For example, a browser client could have a toggle switch for browsing openly/anonymously, which would respectively enable/disable the sending of Referer and From information.

Clients SHOULD NOT include a Referer header field in a (non-secure) HTTP request if the referring page was transferred with a secure protocol.

Authors of services which use the HTTP protocol SHOULD NOT use GET based forms for the submission of sensitive data, because this will cause this data to be encoded in the Request-URI. Many existing servers, proxies, and user agents will log the request URI in some place where it might be visible to third parties. Servers can use POST-based form submission instead

### 15.1.4 Privacy Issues Connected to Accept Headers

Accept request-headers can reveal information about the user to all servers which are accessed. The Accept-Language header in particular can reveal information the user would consider to be of a private nature, because the understanding of particular languages is often

strongly correlated to the membership of a particular ethnic group. User agents which offer the option to configure the contents of an Accept-Language header to be sent in every request are strongly encouraged to let the configuration process include a message which makes the user aware of the loss of privacy involved.

An approach that limits the loss of privacy would be for a user agent to omit the sending of Accept-Language headers by default, and to ask the user whether or not to start sending Accept-Language headers to a server if it detects, by looking for any Vary response-header fields generated by the server, that such sending could improve the quality of service.

Elaborate user-customized accept header fields sent in every request, in particular if these include quality values, can be used by servers as relatively reliable and long-lived user identifiers. Such user identifiers would allow content providers to do click-trail tracking, and would allow collaborating content providers to match cross-server click-trails or form submissions of individual users. Note that for many users not behind a proxy, the network address of the host running the user agent will also serve as a long-lived user identifier. In environments where proxies are used to enhance privacy, user agents ought to be conservative in offering accept header configuration options to end users. As an extreme privacy measure, proxies could filter the accept headers in relayed requests. General purpose user agents which provide a high degree of header configurability SHOULD warn users about the loss of privacy which can be involved.

## 15.2 Attacks Based On File and Path Names

Implementations of HTTP origin servers SHOULD be careful to restrict the documents returned by HTTP requests to be only those that were intended by the server administrators. If an HTTP server translates HTTP URIs directly into file system calls, the server MUST take special care not to serve files that were not intended to be delivered to HTTP clients. For example, UNIX, Microsoft Windows, and other operating systems use ".." as a path component to indicate a directory level above the current one. On such a system, an HTTP server

MUST disallow any such construct in the Request-URI if it would otherwise allow access to a resource outside those intended to be accessible via the HTTP server. Similarly, files intended for reference only internally to the server (such as access control files, configuration files, and script code) MUST be protected from inappropriate retrieval, since they might contain sensitive information. Experience has shown that minor bugs in such HTTP server implementations have turned into security risks.

## 15.3 DNS Spoofing

Clients using HTTP rely heavily on the Domain Name Service, and are thus generally prone to security attacks based on the deliberate mis-association of IP addresses and DNS names. Clients need to be cautious in assuming the continuing validity of an IP number/DNS name association.

In particular, HTTP clients SHOULD rely on their name resolver for confirmation of an IP number/DNS name association, rather than caching the result of previous host name lookups. Many platforms already can cache host name lookups locally when appropriate, and they SHOULD be configured to do so. It is proper for these lookups to be cached, however, only when the TTL (Time To Live) information reported by the name server makes it likely that the cached information will remain useful.

If HTTP clients cache the results of host name lookups in order to achieve a performance improvement, they MUST observe the TTL information reported by DNS.

If HTTP clients do not observe this rule, they could be spoofed when a previously-accessed server's IP address changes. As network renumbering is expected to become increasingly common [24], the possibility of this form of attack will grow. Observing this requirement thus reduces this potential security vulnerability.

This requirement also improves the load-balancing behavior of clients for replicated servers using the same DNS name and reduces the likelihood of a user's experiencing failure in accessing sites which use that strategy.

## 15.4 Location Headers and Spoofing

If a single server supports multiple organizations that do not trust one another, then it MUST check the values of Location and Content-Location headers in responses that are generated under control of said organizations to make sure that they do not attempt to invalidate resources over which they have no authority.

## 15.5 Content-Disposition Issues

RFC 1806 [35], from which the often implemented Content-Disposition (see section 19.5.1) header in HTTP is derived, has a number of very serious security considerations. Content-Disposition is not part of the HTTP standard, but since it is widely implemented, we are documenting its use and risks for implementors. See RFC 2183 [49] (which updates RFC 1806) for details.

## 15.6 Authentication Credentials and Idle Clients

Existing HTTP clients and user agents typically retain authentication information indefinitely. HTTP/1.1 does not provide a method for a server to direct clients to discard these cached credentials. This is a significant defect that requires further extensions to HTTP. Circumstances under which credential caching can interfere with the application's security model include but are not limited to:

- Clients which have been idle for an extended period following which the server might wish to cause the client to reprompt the user for credentials.
- Applications which include a session termination indication (such as a 'logout' or 'commit' button on a page) after which

the server side of the application `knows' that there is no further reason for the client to retain the credentials.

This is currently under separate study. There are a number of work- arounds to parts of this problem, and we encourage the use of password protection in screen savers, idle time-outs, and other methods which mitigate the security problems inherent in this problem. In particular, user agents which cache credentials are encouraged to provide a readily accessible mechanism for discarding cached credentials under user control.

## 15.7 Proxies and Caching

By their very nature, HTTP proxies are men-in-the-middle, and represent an opportunity for man-in-the-middle attacks. Compromise of the systems on which the proxies run can result in serious security and privacy problems. Proxies have access to security-related information, personal information about individual users and organizations, and proprietary information belonging to users and content providers. A compromised proxy, or a proxy implemented or configured without regard to security and privacy considerations, might be used in the commission of a wide range of potential attacks.

Proxy operators should protect the systems on which proxies run as they would protect any system that contains or transports sensitive information. In particular, log information gathered at proxies often contains highly sensitive personal information, and/or information about organizations. Log information should be carefully guarded, and appropriate guidelines for use developed and followed. (Section [15.1.1](#)).

Caching proxies provide additional potential vulnerabilities, since the contents of the cache represent an attractive target for malicious exploitation. Because cache contents persist after an HTTP request is complete, an attack on the cache can reveal information long after a user believes that the information has been removed from the network. Therefore, cache contents should be protected as sensitive information.

Proxy implementors should consider the privacy and security implications of their design and coding decisions, and of the configuration options they provide to proxy operators (especially the default configuration).

Users of a proxy need to be aware that they are no trustworthier than the people who run the proxy; HTTP itself cannot solve this problem.

The judicious use of cryptography, when appropriate, may suffice to protect against a broad range of security and privacy attacks. Such cryptography is beyond the scope of the HTTP/1.1 specification.

### 15.7.1 Denial of Service Attacks on Proxies

They exist. They are hard to defend against. Research continues. Beware.