HTTP - Security

HTTP is used for communications over the internet, so application developers, information providers, and users should be aware of the security limitations in HTTP/1.1. This discussion does not include definitive solutions to the problems mentioned here but it does make some suggestions for reducing security risks.

Personal Information Leakage

HTTP clients are often privy to large amount of personal information such as the user's name, location, mail address, passwords, encryption keys, etc. So you should be very careful to prevent unintentional leakage of this information via the HTTP protocol to other sources.

- All the confidential information should be stored at the server in encrypted form.
- 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.
- Proxies that 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.
- 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.
- 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 that use the HTTP protocol should not use GET based forms for the submission of sensitive data, because it will cause the data to be encoded in the Request-URI.

File and Path Names Based Attack

The document should be restricted to the documents returned by HTTP requests to be only those that were intended by the server administrators.

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.

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. So clients need to be cautious in assuming the continuing validity of an IP number/DNS name association.

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 the DNS. If HTTP clients do not observe this rule, they could be spoofed when a previously-accessed server's IP address changes.

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 the responses that are generated under the control of said organizations to make sure that they do not attempt to invalidate resources over which they have no authority.

Authentication Credentials

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 which is a big security risk.

There are a number of work around to the parts of this problem, and so it is recommended to make the use of password protection in screen savers, idle time-outs, and other methods that mitigate the security problems inherent in this problem.

Proxies and Caching

HTTP proxies are men-in-the-middle, and represent an opportunity for man-in-the-middle attacks. Proxies have access to security-related information, personal information about individual users and organizations, and proprietary information belonging to users and content providers.

Proxy operators should protect the systems on which proxies run, as they would protect any system that contains or transports sensitive information.

Caching proxies provide additional potential vulnerabilities, since the contents of the cache represent an attractive target for malicious exploitation. Therefore, cache contents should be protected as sensitive information.