Internet Engineering Task Force (IETF)
Request for Comments: 6265
Obsoletes: 2965
Category: Standards Track
ISSN: 2070-1721

A. Barth
U.C. Berkeley

April 2011

Механизм управления состоянием HTTP

HTTP State Management Mechanism

Аннотация

Этот документ определяет поля Cookie и Set-Cookie заголовка HTTP. Эти поля могут использоваться серверами HTTP для сохранения состояний (называемых cookie) на стороне пользовательских агентов HTTP, что позволяет серверам поддерживать состояния для протокола HTTP, который не поддерживает состояний. Хотя с переменными cookie исторически связано множество недостатков, снижающих уровни безопасности и приватности, поля заголовков Cookie и Set-Cookie широко используются в сети Internet. Данный документ отменяет действие RFC 2965.

Статус документа

Документ является проектом стандарт Internet (Internet Standards Track).

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. Не все одобренные IESG документы претендуют на статус Internet Standard (см. раздел 2 в RFC 5741).

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке http://www.rfc-editor.org/info/rfc6265.

Оглавление

[1]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

Этот документ определяет поля Cookie и Set-Cookie в заголовках HTTP. Используя поле Set-Cookie, сервер HTTP может передать пару «имя-значение» (name/value) и связанные метаданные (cookie) пользовательскому агенту. При последующем запросе этого агента к данному серверу, агент будет использовать метаданные и другую информацию, чтобы определить, нужно ли возвращать в заголовке Cookie пары «имя-значение».

Несмотря на кажущуюся простоту cookie, с ними связан целый ряд сложностей. Например, сервер указывает область действия для каждого cookie при отправке их пользовательскому агенту. Область действия определяет максимальный интервал времени, в течение которого пользовательскому агенту следует возвращать это значение, серверы, которым следует возвращать cookie, и схемы URI, для которых применимы cookie.

Исторически с cookie связано множество вопросов конфиденциальности и приватности. Например, сервер может указать, что данное значение cookie предназначено для «защищённых» соединений, но атрибут Secure не обеспечивает защиты целостности от активных сетевых атак. Аналогично, значения cookie для данного хоста являются общими для всех портов хоста, хотя обычная политика «одного источника (same-origin policy), используемая браузерами, изолирует содержимое, полученное через разные порты.

Данная спецификация адресована двум категориям читателей - разработчикам серверов, создающих cookie, и разработчикам потребляющих cookie пользовательских агентов.

Для обеспечения максимального взаимодействия с пользовательскими агентами серверам следует при генерации cookie ограничиваться хорошо зарекомендовавшим себя профилем, который определён в разделе 4.

Пользовательские агенты должны реализовать более мягкие правила обработки, определённые в разделе 5, для обеспечения максимального взаимодействия с имеющимися серверами, которые не соответствуют профилю, определённому в разделе 4.

Этот документ задаёт синтаксис и семантику заголовков, которые реально применяются в Internet. В частности, этот документ не определяет какого-либо нового синтаксиса или семантики. Рекомендации по созданию cookie в разделе 4 представляют предпочтительное подмножество свойств современных серверов и даже алгоритм более мягкой обработки cookie, описанный в разделе 5 не включает всех современных синтаксических и семантических вариаций. В тех случаях, когда существующие программы значительно отличаются от рекомендуемого протокола, документ даёт соответствующие разъяснения.

До выпуска этого документа существовало по крайней мере три описания cookie - Netscape cookie specification [Netscape], RFC 2109 [RFC2109] и RFC 2965 [RFC2965]. Однако ни один из этих документов не описывал реального использования заголовков Cookie и Set-Cookie в сети Internet (см. [Kri2001]). В отношении предыдущих спецификаций IETF для механизмов управления состояниями HTTP данный документ запрашивает указанные ниже действия.

1. Перевести [RFC2109] в состояние Historic (документ уже отменен [RFC2965]).

2. Перевести [RFC2965] в состояние Historic.
3. Указать, что [RFC2965] отменен данным документом.

В частности, вместе с отменой RFC 2965 и переводом документа в состояние Historic, данный документ отменяет использование в заголовках полей Cookie2 и Set-Cookie2.

2. Соглашения

2.1. Критерии соответствия

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе должны интерпретироваться в соответствии с [RFC2119].

Требования, представленные в повелительном наклонении, как часть алгоритма (например, «вырезать все пробельные символы в начале» или «возвратить false и прервать этот этап»), трактуются со значением ключевого слова (**должно**, **следует**, **возможно** и т. п.) использованным в спецификации алгоритма.

Требования совместимости, выраженные как алгоритмы или конкретные шаги, могут быть реализованы любым способом, дающим эквивалентный результат. В частности, алгоритмы, определённые в этой спецификации, предназначены для упрощения понимания, а не для реализации.

2.2. Описание синтаксиса

В этой спецификации используются обозначения ABNF[1] [RFC5234].

Используются определённые в Приложении B.1 [RFC5234] определения: ALPHA (буквы), CR (возврат каретки), CRLF (CR LF), CTL (элемент управления), DIGIT (десятичные цифры 0-9), DQUOTE (двойные кавычки), HEXDIG (шестнадцатеричные цифры 0-9/A-F/a-f), LF (перевод строки), NUL (октет со значением 0), OCTET (любая 8-битовая последовательность кроме NUL), SP (пробел), HTAB (горизонтальная табуляция), CHAR (любой символ [USASCII]), VCHAR (любой видимый (печатный) символ [USASCII]), WSP (пробел).

Правило OWS (необязательные пробелы) применяется в тех случаях когда **может** (но не обязательно) появляться некоторое число пробелов (linear whitespace).

```
OWS            = *( [ obs-fold ] WSP ) ; «необязательный» пробел
obs-fold       = CRLF
```

OWS **следует** создавать в виде одного символа SP или не создавать совсем.

2.3. Терминология

Термины пользовательский агент, клиент, сервер, прокси и сервер-источник используются в том же смысле, который принят в спецификации HTTP/1.1 ([RFC2616], параграф 1.3).

Request-host указывает имя хоста, известное пользовательскому агенту, по которому этот агент отправляет запрос HTTP или с которого он получает отклик HTTP (т. е., хоста, которому он отправил соответствующий запрос HTTP).

Термин request-uri определён в параграфе 5.1.2 [RFC2616].

О двух последовательностях октетов говорят, что они совпадают без учёта регистра, тогда и только тогда, когда они эквивалентны в сопоставлении i;ascii-casemap, определённом в [RFC4790].

Термин «строка» (string) обозначает последовательность октетов, отличающихся от NUL.

3. Обзор

В этом разделе описан способ, которым сервер-источник отправляет информацию о состоянии пользовательскому агенту, а тот возвращает информацию о состоянии серверу.

Для сохранения состояния сервер-источник включает заголовок Set-Cookie в отклик HTTP. В последующих запросах пользовательский агент возвращает заголовок запроса Cookie серверу-источнику. Заголовок Cookie содержит значения cookie, которые пользовательский агент получил в предшествующих заголовках Set-Cookie. Сервер-источник может игнорировать заголовок Cookie или использовать его содержимое в своих целях, определяемых приложением.

Серверы-источники **могут** передавать заголовок Set-Cookie в любом отклике. Пользовательские агенты **могут** игнорировать заголовки Set-Cookie, содержащиеся в откликах с кодом уровня 100, но **должны** обрабатывать заголовки Set-Cookie, содержащиеся в других откликах (включая отклики с кодами уровней 400 и 500). Сервер-источник может включить в один отклик множество полей заголовка Set-Cookie. Наличие полей Cookie или Set-Cookie в заголовках не препятствует хранению откликов в кэше HTTP и их повторному использованию.

Серверам-источникам **недопустимо** помещать множество полей Set-Cookie в одно поле заголовка. Обычный механизм «упаковки» полей заголовков HTTP (определённый в [RFC2616]) может менять семантику полей заголовка Set-Cookie, поскольку символ %x2C (,) используется в Set-Cookie так, что может вызывать конфликт с такой «упаковкой».

3.1. Примеры

Используя заголовок Set-Cookie, сервер может передать пользовательскому агенту короткую строку в отклике HTTP, которую этот агент будет возвращать в последующих запросах HTTP, попадающих в область действия cookie. Например, сервер может передать пользовательскому агенту идентификатор сессии с именем SID и значением 31d4d96e407aad42. Пользовательский агент будет возвращать этот идентификатор в последующих запросах.

```
== Сервер -> Пользовательский агент ==
Set-Cookie: SID=31d4d96e407aad42
```

[1] Augmented Backus-Naur Form - расширенная форма Бэкуса-Наура.

```
== Пользовательский агент -> Сервер ==
Cookie: SID=31d4d96e407aad42
```

Сервер может поменять принятую по умолчанию область действия cookie, используя атрибуты Path и Domain. Например, сервер может дать пользовательскому агенту инструкцию возвращать cookie для каждого пути и каждого субдомена в example.com.

```
== Сервер -> Пользовательский агент ==
Set-Cookie: SID=31d4d96e407aad42; Path=/; Domain=example.com

== Пользовательский агент -> Сервер ==
Cookie: SID=31d4d96e407aad42
```

Как показано в следующем примере, сервер может сохранить множество cookie на стороне пользовательского агента. Например, сервер может сохранить идентификатор сессии, а также предпочитаемый пользователем язык, возвращая два поля заголовка Set-Cookie. Обратите внимание на то, что сервер использует атрибуты Secure и HttpOnly для обеспечения дополнительной защиты идентификатора сессии (см. параграф 4.1.2).

```
== Сервер -> Пользовательский агент ==
Set-Cookie: SID=31d4d96e407aad42; Path=/; Secure; HttpOnly
Set-Cookie: lang=en-US; Path=/; Domain=example.com

== Пользовательский агент -> Сервер ==
Cookie: SID=31d4d96e407aad42; lang=en-US
```

Обратите внимание, что заголовок Cookie содержит два значения cookie с именами SID и lang. Если сервер хочет, чтобы пользовательский агент сохранял значения cookie на протяжении нескольких «сессий» (например, при перезапуске пользовательского агента), он может задать срок действия с помощью атрибута Expires. Отметим, что пользовательский агент может удалить cookie до истечения срока действия, если в хранилище cookie не будет достаточно места или пользователь вручную удалит cookie сервера.

```
== Сервер -> Пользовательский агент ==
Set-Cookie: lang=en-US; Expires=Wed, 09 Jun 2021 10:18:14 GMT

== Пользовательский агент -> Сервер ==
Cookie: SID=31d4d96e407aad42; lang=en-US
```

В заключение отметим, что сервер может удалить cookie, возвращая заголовок Set-Cookie с уже завершившимся сроком действия. Сервер сможет удалить cookie только в том случае, когда атрибуты Path и Domain в заголовке Set-Cookie соответствуют значениям, использованным при создании cookie.

```
== Сервер -> Пользовательский агент ==
Set-Cookie: lang=; Expires=Sun, 06 Nov 1994 08:49:37 GMT

== Пользовательский агент -> Сервер ==
Cookie: SID=31d4d96e407aad42
```

4. Требования к серверу

В этом разделе описаны синтаксис и семантика хорошо себя зарекомендовавшего профиля заголовков Cookie и Set-Cookie.

4.1. Заголовок Set-Cookie

Заголовок Set-Cookie в откликах HTTP служит для передачи cookie от сервера к пользовательскому агенту.

4.1.1. Синтаксис

Неформально заголовок отклика Set-Cookie содержит имя заголовка (Set-Cookie), за которым следует символ двоеточия (:) и cookie. Каждое поле cookie начинается с пары «имя-значение», за которым могут следовать дополнительные пары «имя-значение». Серверам **недопустимо** передавать заголовки Set-Cookie, не соответствующие приведённым ниже правилам.

```
set-cookie-header = "Set-Cookie:" SP set-cookie-string
set-cookie-string = cookie-pair *( ";" SP cookie-av )
cookie-pair       = cookie-name "=" cookie-value
cookie-name       = token
cookie-value      = *cookie-octet / ( DQUOTE *cookie-octet DQUOTE )
cookie-octet      = %x21 / %x23-2B / %x2D-3A / %x3C-5B / %x5D-7E
                      ; символы US-ASCII, включая CTL, пробельный
                      ; DQUOTE, запятую, точку с запятой и \
token             = <token, определённый в параграфе 2.2 [RFC2616]>

cookie-av         = expires-av / max-age-av / domain-av /
                    path-av / secure-av / httponly-av /
                    extension-av
expires-av        = "Expires=" sane-cookie-date
sane-cookie-date  = <rfc1123-date, определённый в параграфе 3.3.1 [RFC2616]>
max-age-av        = "Max-Age=" non-zero-digit *DIGIT
                      ; На практике expires-av и max-age-av ограничены датами,
                      ; которые может представить пользовательский агент.
non-zero-digit    = %x31-39
                      ; цифры от 1 до 9
domain-av         = "Domain=" domain-value
domain-value      = <subdomain>
                      ; определено в параграфе 3.5 [RFC1034] и
                      ; расширено в параграфе 2.1 [RFC1123]
path-av           = "Path=" path-value
path-value        = * <любой CHAR, кроме CTL и ";">
```

```
secure-av       = "Secure"
httponly-av     = "HttpOnly"
extension-av    = * <любой CHAR, кроме CTL и ";">
```

Отметим, что в некоторых грамматических терминах в упомянутых выше документах применяются грамматические обозначения, отличающиеся от принятых в этом документе (ABNF из [RFC5234]).

Семантика cookie-value не определяется этим документом.

Для обеспечения максимальной совместимости с пользовательскими агентами серверам, которые хотят сохранять в cookie-value произвольные значения, **следует** кодировать эти данные (например, с использованием Base64 [RFC4648]).

Части set-cookie-string, создаваемые элементами cookie-av, называют атрибутами. Для обеспечения максимальной совместимости с пользовательскими агентами серверам **не следует** создавать два атрибута с одинаковыми именами в одной set-cookie-string (см. параграф 5.3, где описана обработка этого случая пользовательскими агентами).

Серверам не следует включать в один отклик более одного поля Set-Cookie с тем же значением cookie-name см. параграф 5.2, где описана обработка этого случая пользовательскими агентами (параграф 5.2).

Если сервер одновременно передаёт пользовательскому агенту множество откликов с заголовками Set-Cookie (например, при взаимодействии с пользовательским агентом через несколько сокетов), между откликами возникает «конкуренция» (race condition), которая может давать непредсказуемые результаты.

> Примечание. Некоторые из имеющихся пользовательских агентов по разному интерпретируют обозначение года двумя цифрами. Для предотвращения проблем совместимости серверам **следует** использовать формат rfc1123-date, где год указывается 4 цифрами.
>
> Примечание. Некоторые пользовательские агенты сохраняют и обрабатывают даты в cookie, как 32-битовые значения UNIX time_t. Ошибки реализации в некоторых библиотеках, поддерживающих обработку time_t могут приводить к некорректной обработке дат после 2038 года.

4.1.2. Семантика (не нормативно)

В этом параграфе описана упрощённая семантика заголовка Set-Cookie. Описание содержит достаточное число деталей для понимания общих вопросов использования cookie на серверах. Полное описание семантики приведено в разделе 5.

При получении пользовательским агентом заголовка Set-Cookie он сохраняет cookie вместе с атрибутами. Позднее, когда пользовательский агент делает запрос HTTP, он включает применимые cookie, для которых не истёк срок действия, в заголовок Cookie.

Если пользовательский агент получает новое «значение» cookie с такими же полями cookie-name, domain-value и path-value, какие уже сохранены, существующая запись cookie заменяется новой. Отметим, что серверы могут удалять cookie, передавая клиенту новое значение cookie с атрибутом Expires, указывающим уже прошедшее время.

Если в атрибутах cookie не указано иное, значение cookie возвращается только серверу источнику (а не в какой-нибудь из субдоменов) и срок действия cookie завершается при закрытии текущей сессии (с точки зрения пользовательского агента). Пользовательские агенты игнорируют нераспознанные атрибуты cookie (но не cookie целиком).

4.1.2.1. Атрибут Expires

Атрибут Expires указывает срок действия cookie датой и временем его завершения. Пользовательский агент не обязан сохранять cookie до наступления указанного времени. На практике пользовательские агенты зачастую удаляют cookie для освобождения памяти или по соображениям защиты приватности.

4.1.2.2. Атрибут Max-Age

Атрибут Max-Age указывает срок действия cookie, представленный число оставшихся секунд. Пользовательский агент не обязан сохранять cookie до истечения указанного времени. На практике пользовательские агенты зачастую удаляют cookie для освобождения памяти или по соображениям защиты приватности.

> Примечание. Некоторые из имеющихся пользовательских агентов не поддерживают атрибут Max-Age и просто игнорируют его.

Если в cookie присутствуют оба атрибута Max-Age и Expires, атрибут Max-Age является предпочтительным и будет определять срок действия cookie. Если в cookie не указано ни одного из атрибутов Max-Age и Expires, пользовательский агент будет сохранять cookie «до завершения текущей сессии» (с его точки зрения).

4.1.2.3. Атрибут Domain

Атрибут Domain указывает хосты, для которых было передано значение cookie. Например, если указан атрибут Domain со значением example.com, пользовательский агент будет включать cookie в заголовки Cookie при отправке запросов HTTP на серверы example.com, www.example.com и www.corp.example.com (отметим, что при наличии в начале значения символа точки - %x2E - он будет игнорироваться, поскольку не является разрешённым, но точка в конце будет приводить к игнорированию атрибута целиком). Если сервер не указал атрибута Domain, пользовательский агент будет возвращать cookie только серверу-источнику.

> Предупреждение. Некоторые реализации пользовательских агентов трактуют отсутствие атрибута Domain, как атрибут Domain с текущим именем хоста. Например, если example.com возвращает заголовок Set-Cookie без атрибута Domain, такие агенты будут ошибочно отправлять cookie также серверу www.example.com.

Пользовательский агент будет отвергать cookie, в которых атрибут Domain задаёт область действия, не включающую сервер-источник. Например, пользовательский агент будет воспринимать cookie с атрибутом Domain, имеющим значение example.com или foo.example.com, от сервера foo.example.com, но не будет воспринимать от него cookie с атрибутом Domain, имеющим значение bar.example.com или baz.foo.example.com.

Примечание. Из соображений безопасности многие пользовательские агенты настроены отвергать атрибуты Domain, соответствующие «публичным суффиксам». Например, некоторые пользовательские агенты будут отвергать атрибуты Domain со значением com или co.uk (см. параграф 5.3).

4.1.2.4. Атрибут Path

Область действия cookie ограничена путями, контролируемыми с помощью атрибута Path. Если сервер не указывает атрибут Path, пользовательский агент будет использовать по умолчанию компонент пути из request-uri (см. параграф 5.1.4).

Пользовательский агент будет включать cookie в запрос HTTP только в тех случаях, когда относящаяся к пути часть request-uri совпадает (или является подкаталогом) со значением атрибута Path в cookie, где символ %x2F (/) интерпретируется, как разделитель в иерархии каталогов.

Несмотря на кажущуюся привлекательность разделения cookie по путям в рамках данного хоста, атрибут Path не может служить основой для защиты (см. раздел 8).

4.1.2.5. Атрибут Secure

Атрибут Secure ограничивает область действия cookie «защищёнными» каналами («защищенность определяется пользовательским агентом). При наличии в cookie атрибута Secure пользовательский агент будет включать cookie в запросы HTTP только при их передаче через защищённые каналы (обычно HTTP на основе TLS[1] [RFC2818]).

Несмотря на кажущуюся привлекательность cookie для защиты от активных сетевых атак, атрибут Secure обеспечивает защиту лишь для конфиденциальности cookie. Активный атакующий может переписать атрибуты Secure в незащищённом канале, нарушая их целостность (см. параграф 8.6).

4.1.2.6. Атрибут HttpOnly

Атрибут Secure ограничивает область действия cookie запросами HTTP. В частности, этот атрибут предлагает пользовательскому агенту опускать cookie в случае предоставления доступа к cookie через не относящиеся в HTTP интерфейсы API[2] (например, API браузера, раскрывающий cookies для сценариев).

Отметим, что атрибут HttpOnly не связан с атрибутом Secure и в cookie могут включаться оба эти атрибута.

4.2. Заголовок Cookie

4.2.1. Синтаксис

Пользовательские агент передаёт сохранённые значение cookie серверу-источнику в заголовке Cookie. Если сервер соответствует требованиям параграфа 4.1 (а пользовательский агент соответствует требованиям раздела 5), агент будет передавать заголовок Cookie, соответствующий приведённой ниже форме.

```
cookie-header = "Cookie:" OWS cookie-string OWS
cookie-string = cookie-pair *( ";" SP cookie-pair )
```

4.2.2. Семантика

Каждая пара cookie-pair представляет экземпляр cookie, сохранённый пользовательским агентом. Пара cookie-pair включает поля cookie-name и cookie-value, которые агент получил в заголовке Set-Cookie.

Отметим, что атрибуты cookie не возвращаются. В частности, сервер не может определить из заголовка Cookie срок завершения действия cookie, хосты, для которых cookie действует или наличие в cookie атрибута Secure или HttpOnly.

Семантика отдельных cookie в заголовке Cookie не определяется данным документом. Предполагается, что серверы используют cookie с определяемой приложением семантикой.

Хотя cookie указываются в заголовке Cookie последовательно, серверам **не следует** полагаться на порядок их размещения. В частности, если заголовок Cookie содержит пару одноимённых cookie (например, полученных с разными атрибутами Path или Domain), серверу не следует принимать решения, основанные на порядке размещения cookie в заголовке.

5. Требования к пользовательским агентам

В этом разделе заголовки Cookie и Set-Cookie рассмотрены более подробно, чтобы пользовательские агенты, в точности реализующие эти требования, были совместимы с имеющимися серверами (включая те, которые не соответствуют профилю, описанному в разделе 4).

Пользовательские агенты могут вносить дополнительные ограничения (например, из соображений безопасности), однако эксперименты показывают, что такая «строгость» снижает уровень совместимости с имеющимися серверами.

5.1. Алгоритмы обработки субкомпонент

В этом разделе определены некоторые алгоритмы, применяемые пользовательскими агентами для обработки конкретных субкомпонент заголовков Cookie и Set-Cookie.

5.1.1. Даты

Для разбора cookie-date пользовательский агент **должен** использовать алгоритм, эквивалентный описанному ниже. Отметим, что различные логические флаги, определённые как часть алгоритма (т. е., found-time, found-day-of-month, found-month, found-year) изначально сброшены (0).

1. На основе приведённых ниже правил cookie-date разбивается на date-token.

[1]Transport Layer Security - защита транспортного уровня.
[2]Application programming interface - интерфейс для прикладных программ. *Прим. перев.*

```
cookie-date     = *delimiter date-token-list *delimiter
date-token-list = date-token *( 1*delimiter date-token )
date-token      = 1*non-delimiter

delimiter       = %x09 / %x20-2F / %x3B-40 / %x5B-60 / %x7B-7E
non-delimiter   = %x00-08 / %x0A-1F / DIGIT / ":" / ALPHA / %x7F-FF
non-digit       = %x00-2F / %x3A-FF

day-of-month    = 1*2DIGIT [ non-digit *OCTET ]
month           = ( "jan" / "feb" / "mar" / "apr" /
                    "may" / "jun" / "jul" / "aug" /
                    "sep" / "oct" / "nov" / "dec" ) *OCTET
year            = 2*4DIGIT [ non-digit *OCTET ]
time            = hms-time [ non-digit *OCTET ]
hms-time        = time-field ":" time-field ":" time-field
time-field      = 1*2DIGIT
```

2. Маркеры date-token последовательно обрабатываются в порядке их указания в cookie-date.
    1. Если флаг found-time не установлен и маркер соответствует времени создания, устанавливается флаг found-time, а также значения hour-value, minute-value и second-value в соответствии с числами в date-token. Остальные этапы пропускаются и обрабатывается следующий маркер date-token.
    2. Если флаг found-day-of-month не установлен и маркер соответствует дате (число месяца) создания, устанавливается флаг found-day-of-month, а также значение day-of-month-value в соответствии с числом в date-token. Остальные этапы пропускаются и обрабатывается следующий маркер date-token.
    3. Если флаг found-month не установлен и маркер соответствует месяцу создания, устанавливается флаг found-month, а также значение month-value в соответствии с числом в date-token. Остальные этапы пропускаются и обрабатывается следующий маркер date-token.
    4. Если флаг found-year не установлен и маркер соответствует году создания, устанавливается флаг found-year, а также значение year-value в соответствии с числом в date-token. Остальные этапы пропускаются и обрабатывается следующий маркер date-token.
3. Если значение year-value находится в интервале от 70 до 99 (включительно), к year-value добавляется 1900.
4. Если значение year-value находится в интервале от 0 до 69 (включительно), к year-value добавляется 2000.
    1. Примечание. Некоторые пользовательские агенты иначе интерпретируют представление года двумя цифрами.
5. Обработка cookie-date прерывается отказом, если выполняется любое из указанных ниже условий:
    - не установлен хотя бы один из флагов found-day-of-month, found-month, found-year, found-time;
    - значение day-of-month-value меньше 1 или больше 31;
    - значение year-value меньше 1601;
    - значение hour-value больше 23;
    - значение minute-value больше 59;
    - значение second-value больше 59

    (отметим, что в этом синтаксисе не могут быть представлены високосные секунды).
6. Пусть parsed-cookie-date будет дата, в которой поля day-of-month, month, year, hour, minute и second (в UTC) будут иметь значения day-of-month-value, month-value, year-value, hour-value, minute-value second-value, соответственно. Если такой даты не существует, разбор cookie-date прерывается отказом.
7. Возврат значения parsed-cookie-date в качестве результат работы алгоритма.

5.1.2. Канонические имена хостов

Канонические имена хостов создаются с помощью приведённого ниже алгоритма.

1. Имя хоста преобразуется в последовательность отдельных меток доменного имени.
2. Каждая метке, которая не является NR-LDH[1], преобразуется в A-метку (см. параграф 2.3.2.1 в [RFC5890]) или метку punycode (преобразование ToASCII в соответствии с разделом 4 в [RFC3490]), в зависимости от ситуации (см. параграф 6.3).
3. Полученные метки собираются в одну строку с разделением точками (%x2E).

5.1.3. Соответствие домена

Строка соответствует домену, если выполняется хотя бы одно из двух приведённых ниже условий.

- Домен идентичен строке (отметим, что здесь предполагается приведение обеих строк к нижнему регистру).
- Выполняются все приведённые ниже условия:
    - домен является суффиксом строки;
    - последним символом строки, не входящим в имя домена, является точка (%x2E);

[1]Non-Reserved LDH.

- строка является именем хоста (не IP-адресом).

5.1.4. Пути и path-Match

Для расчёта default-path в cookie пользовательский агент **должен** применять алгоритм, эквивалентный приведённому ниже.

1. Пусть uri-path будет частью request-uri, если такая часть существует и пустым в остальных случаях. Например, если request-uri содержит путь (и необязательную строку запроса), uri-path будет этим путём (без %x3F (?) и строки запроса), а если request-uri содержит полный идентификатор absoluteURI, uri-path будет компонентой пути этого URI.
2. Если значение uri-path пусто или первый символ отличается от / (%x2F), возвращается %x2F (/) и остальные этапы пропускаются.
3. Если uri-path содержит не более одного символа / (%x2F), возвращается %x2F (/) и остальные этапы пропускаются.
4. Возвращаются символы uri-path, начиная с первого и заканчивая (но не включая) последним символом %x2F (/).

Строка request-path соответствует пути в данном cookie-path, если выполняется хотя бы одно из приведённых ниже условий.

- Строки cookie-path и request-path идентичны. Отметим, что это отличается от правил RFC 3986 для эквивалентности компоненты path и, следовательно, два эквивалентных пути могут иметь разные cookie.
- Строка cookie-path является префиксом request-path и последним символом cookie-path является , (%x2F).
- Строка cookie-path является префиксом request-path и первым символом request-path, не включённым в cookie-path, является / (%x2F).

5.2. Заголовок Set-Cookie

Получив поле заголовка Set-Cookie в отклике HTTP, пользовательский агент **может** игнорировать поле Set-Cookie целиком. Например, пользовательский агент может выбрать блокирование установки cookie на основании откликов на запросы «третьих сторон» (см. параграф 7.1).

Если пользовательский агент не игнорирует Set-Cookie целиком, он **должен** разобрать содержимое заголовка Set-Cookie, как set-cookie-string (см. определение ниже).

Примечание. Приведённый ниже алгоритм более мягок, нежели правила параграфа 4.1. Например, алгоритм вырезает начальные и конечные пробелы из имени и значения cookie (но сохраняет пробелы внутри), тогда как параграф 4.1 запрещает пробелы в этих позициях. Пользовательские агенты применяют этот алгоритм для обеспечения совместимости с серверами, не выполняющими рекомендаций раздела 4.

Для разбора set-cookie-string пользовательский агент **должен** применять алгоритм, эквивалентный приведённому ниже.

1. Если set-cookie-string содержит символ %x3B (;),

   строка name-value-pair будет включать символы вплоть (но не включая) до первого символа %x3B (;), а строка unparsed-attributes будет содержать оставшуюся часть set-cookie-string (включая упомянутый символ %x3B).

   В противном случае

   строка name-value-pair будет включать все символы set-cookie-string, а unparsed-attributes будет пустой.
2. Если строка name-value-pair не содержит знака равенства (%x3D), set-cookie-string игнорируется целиком.
3. Строка name (возможно пустая) будет содержать символы вплоть (но не включая) до первого знака равенства (%x3D), а строка (возможно пустая) будет содержать символы после первого знака равенства (%x3D).
4. Удаляются все символы WSP (пробелы) в начале и в конце строк name и value.
5. Если строка name пуста, set-cookie-string игнорируется целиком.
6. Строка cookie-name будет строкой name, а cookie-value - value.

Для разбора unparsed-attributes пользовательский агент **должен** использовать алгоритм, эквивалентный приведённому ниже.

1. Если строка unparsed-attributes пуста, остальные шаги пропускаются.
2. Отбрасывается первый символ unparsed-attributes (%x3B - ;).
3. Если оставшаяся часть unparsed-attributes содержит символ %x3B (;),

   извлекаются символы unparsed-attributes вплоть (но не включая) до первого символа %x3B (;).

   В противном случае

   извлекается оставшаяся часть unparsed-attributes.

   Извлечённые на этом этапе символы составляют значение cookie-av.
4. Если cookie-av включает символ %x3D (=),

Строка (возможно пустая) attribute-name будет содержать символы вплоть (но не включая) до первого знака равенства (%x3D), а строка (возможно пустая) attribute-value будет содержать символы после знака равенства (%x3D).

В противном случае

Строка attribute-name будет содержать cookie-av целиком, а attribute-value будет пустой.

5. Удаляются все символы WSP (пробелы) в начале и в конце строк attribute-name и attribute-value.
6. Строки attribute-name и attribute-value обрабатываются в соответствии с требованиями последующих параграфов (отметим, что атрибуты с нераспознанными именами attribute-names игнорируются).
7. Возврат на этап 1 данного алгоритма.

По завершении разбора set-cookie-string пользовательский агент «получает cookie» из request-uri с именем cookie-name, значением cookie-value и атрибутами cookie-attribute-list (дополнительные требования, связанные с получением cookie, рассмотрены в параграфе 5.3).

5.2.1. Атрибут Expires

Если attribute-name без учёта регистра символов совпадает со строкой Expires, пользовательский агент **должен** обработать cookie-av, как показано ниже.

Пусть expiry-time будет результатом разбора attribute-value, как cookie-date (см. параграф 5.1.1).

Если attribute-value не удаётся разобрать, как дату cookie, значение cookie-av игнорируется.

Если expiry-time указывает дату позже последней даты, которую может представить пользовательский агент, он **может** заменить expiry-time последней представимой датой.

Если expiry-time указывает дату раньше самой ранней даты, которую может представить пользовательский агент, он **может** заменить expiry-time самой ранней представимой датой.

В конец списка cookie-attribute-list добавляется attribute-name со значением Expires и attribute-value со значением expiry-time.

5.2.2. Атрибут Max-Age

Если attribute-name без учёта регистра символов совпадает со строкой Max-Age, пользовательский агент **должен** обработать cookie-av, как показано ниже.

Если первый символ attribute-value не является цифрой (DIGIT) или символом «-», значение cookie-av игнорируется.

Если оставшаяся часть attribute-value содержит нецифровые символы (не DIGIT), значение cookie-av игнорируется.

Пусть delta-seconds будет значением attribute-value, преобразованным в целое число.

Если delta-seconds не больше 0, значение expiry-time будет самой ранней представимой датой. В противном случае expiry-time будет текущей датой с добавлением delta-seconds секунд.

В конец списка cookie-attribute-list добавляется attribute-name со значением Max-Age и attribute-value со значением expiry-time.

5.2.3. Атрибут Domain

Если attribute-name без учёта регистра символов совпадает со строкой Domain, пользовательский агент **должен** обработать cookie-av, как показано ниже.

Если строка attribute-value является пустой, поведение не определено, однако пользовательскому агенту **следует** игнорировать cookie-av целиком.

Если первым символом attribute-value является точка (%x2E),

тогда cookie-domain будет attribute-value без начальной точки (%x2E).

В противном случае

cookie-domain будет полным значением attribute-value.

Строка cookie-domain приводится к нижнему регистру.

В конец списка cookie-attribute-list добавляется attribute-name со значением Domain и attribute-value со значением cookie-domain.

5.2.4. Атрибут Path

Если attribute-name без учёта регистра символов совпадает со строкой Path, пользовательский агент **должен** обработать cookie-av, как показано ниже.

Если строка attribute-value является пустой или первый символ отличается от %x2F (/),

тогда cookie-path будет принимать значение default-path.

В противном случае

cookie-path будет принимать значение attribute-value.

В конец списка cookie-attribute-list добавляется attribute-name со значением Path и attribute-value со значением cookie-path.

5.2.5. Атрибут Secure

Если attribute-name без учёта регистра символов совпадает со строкой Secure, пользовательский агент **должен** добавить в конец списка cookie-attribute-list attribute-name со значением Secure и пустым attribute-value.

5.2.6. Атрибут HttpOnly

Если attribute-name без учёта регистра символов совпадает со строкой HttpOnly, пользовательский агент **должен** добавить в конец списка cookie-attribute-list attribute-name со значением HttpOnly и пустым attribute-value.

5.3. Модель хранения

Пользовательский агент сохраняет для каждого cookie поля name, value, expiry-time, domain, path, creation-time, last-access-time, persistent-flag, host-only-flag, secure-only-flag и http-only-flag.

Когда пользовательский агент из request-uri «получает cookie» с именем cookie-name, значением cookie-value и атрибутами cookie-attribute-list, этот агент **должен** обработать cookie в соответствии с приведённым ниже алгоритмом.

1. Пользовательский агент **может** игнорировать полученный cookie целиком. Например, он может блокировать получение cookie из «чужих откликов» или отказаться от сохранения cookie по причине экономии места в хранилище.
2. Создаётся новая запись cookie с именем cookie-name и значением cookie-value. Для creation-time и last-access-time указывается текущее время и дата.
3. Если cookie-attribute-list содержит атрибут с именем (attribute-name) Max-Age,

   устанавливается флаг persistent-flag = true;

   в качестве expiry-time для cookie устанавливается attribute-value последнего атрибута cookie-attribute-list с именем (attribute-name) Max-Age.

   В противном случае, если cookie-attribute-list содержит атрибут с именем (attribute-name) Expires (и не содержит атрибута с именем Max-Age),

   устанавливается флаг persistent-flag = true;

   в качестве expiry-time для cookie устанавливается attribute-value последнего атрибута cookie-attribute-list с именем (attribute-name) Expires.

   В остальных случаях

   устанавливается флаг persistent-flag = false;

   в качестве expiry-time для cookie устанавливается последняя представимая дата.
4. Если cookie-attribute-list содержит атрибут с именем (attribute-name) Domain,

   domain-attribute будет attribute-value последнего атрибута в списке cookie-attribute-list с именем (attribute-name) Domain.

   В противном случае

   domain-attribute будет пустой строкой.
5. Если пользовательский агент настроен отвергать «публичные суффиксы» (public suffix), а domain-attribute является таким суффиксом,

   если domain-attribute идентичен канонизированному значению request-host,

   domain-attribute будет пустой строкой.

   В противном случае

   cookie игнорируется целиком и обработка прерывается.

   Примечание. «Публичным суффиксом» считается домен, контролируемый публичным регистратором (например, com, co.uk или pvt.k12.wy.us). Выполнение этого этапа имеет важное значение для предотвращения нарушения целостности домена example.com со стороны злоумышленника attacker.com путём организации cookie с атрибутом Domain = com. К сожалению множество публичных суффиксов (их называют также «доменами, контролируемыми регистратором») изменяется с течением времени. По возможности пользовательским агентам следует применять актуальный список публичных суффиксов типа поддерживаемого проектом Mozilla на странице <http://publicsuffix.org/>.
6. Если значение domain-attribute не пусто,

   если канонизированное значение request-host не соответствует domain-attribute:

   cookie игнорируется целиком и обработка прерывается.

   В противном случае

   устанавливается флаг host-only-flag = false;

   для domain-attribute в качестве домена для cookie.

   Иначе

   устанавливается флаг host-only-flag = true;

   в качестве домена для cookie устанавливается канонизированное значение request-host.

7. Если cookie-attribute-list содержит атрибут с именем (attribute-name) Path, в качестве пути для cookie устанавливается attribute-value последнего атрибута в cookie-attribute-list с именем (attribute-name) Path. В противном случае в качестве пути для cookie устанавливается default-path из the request-uri.
8. Если cookie-attribute-list содержит атрибут с именем (attribute-name) Secure, устанавливается флаг secure-only-flag = true. В противном случае устанавливается secure-only-flag = false.
9. Если cookie-attribute-list содержит атрибут с именем (attribute-name) HttpOnly, устанавливается флаг http-only-flag = true. В противном случае устанавливается secure-only-flag = false.
10. Если cookie был получен от API, не являющегося HTTP, и установлен флаг http-only-flag, обработка прерывается и cookie игнорируется целиком.
11. Если хранилище cookie содержит cookie с таким же именем, доменом и путём, как у вновь созданного cookie:
    1. Пусть old-cookie будет существующий cookie с тем же именем, доменом и путём (отметим, что этот алгоритм предполагает не более одного такого cookie).
    2. Если новый cookie был получен от API, не являющегося HTTP, и для old-cookie установлен флаг http-only-flag, обработка прерывается и вновь созданный cookie игнорируется целиком.
    3. Значение creation-time для нового cookie устанавливается равным creation-time из old-cookie.
    4. old-cookie удаляется из хранилища.
12. Вновь созданный cookie помещается в хранилище.

Cookie считается недействительным (expired) если срок завершения уже наступил.

Пользовательский агент **должен** удалять просроченные cookie из хранилища при их обнаружении.

Пользовательский агент **может** в любой момент «удалить лишние cookie» из хранилища, если число cookie для одного домена превышает заданный реализацией верхний предел (например, 50).

Пользовательский агент **может** в любой момент «удалить лишние cookie» из хранилища, если число cookie в хранилище превышает некое предопределённое значение (например, 3000).

При удалении избыточных cookie из хранилища пользовательский агент **должен** руководствоваться приведённым ниже порядком удаления:

1. устаревшие cookie;
2. cookie с одинаковым доменом в количестве, превышающем заданное значение;
3. прочие cookie.

При наличии двух cookie с одинаковым приоритетом удаления пользовательский агент **должен** удалять cookie с более ранним временем последнего обращения (last-access date).

Когда задан срок действия до конца текущей сессии (the current session is over), пользовательский агент **должен** удалить из хранилища все cookie, для которых не установлен флаг persistent-flag (false).

5.4. Заголовок Cookie

Пользовательский агент включает сохранённые cookie в заголовки Cookie своих запросов HTTP.

При генерации запроса HTTP пользовательскому агенту **недопустимо** включать в запрос более одного поля Cookie.

Пользовательский агент **может** не включать в запрос заголовок Cookie. Например, пользовательский агент может блокировать отправку cookie при запросе «третьей стороны» на установку cookie (см. параграф 7.1).

Если пользовательский агент включает заголовок Cookie в свой запрос HTTP, он **должен** передать в качестве такого заголовка описанную ниже строку cookie-string.

Для создания cookie-string из сохранённого cookie и request-uri пользовательский агент **должен** применять алгоритм, эквивалентный приведённому ниже.

1. Пусть cookie-list обозначает набор cookie из хранилища, удовлетворяющих приведённым ниже требованиям.
   - Или

     флаг host-only-flag для cookie имеет значение true и канонизированное значение request-host идентично домену cookie.

     Или

     флаг host-only-flag для cookie имеет значение false и канонизированное значение request-host соответствует домену cookie.
   - Путь в request-uri соответствует пути в cookie.
   - Если флаг secure-only-flag для cookie имеет значение true, схема request-uri должна указывать «защищённый» протокол (с точки зрения пользовательского агента).

     Примечание. Понятие «безопасного» протокола не определяется в данном документе. Обычно пользовательский агент считает протокол безопасным, если тот обеспечивает защиту на транспортном уровне (типа SSL или TLS). Например, большинство пользовательских агентов считают схему https обозначением безопасного протокола.

- Если флаг http-only-flag для cookie имеет значение true, исключаются cookie, где cookie-string создана с использованием отличного от HTTP API (по усмотрению пользовательского агента).

2. Пользовательскому агенту **следует** отсортировать cookie-list в описанном ниже порядке.

   - Cookie с более длинными путями размещаются впереди cookie с короткими путями.

   - Среди cookies с одинаковым размером пути первыми размещаются cookie с более ранним creation-time.

     Примечание. Не все пользовательские агенты сортируют cookie-list в указанном порядке, но этот порядок отражает общепринятую практику во время подготовки этого документа. Исторически сложилось так, что некоторые серверы (ошибочно) зависят от соблюдения этого порядка.

3. Значение last-access-time каждого cookie из списка cookie-list заменяется текущим временем и датой.

4. Список cookie-list преобразуется в строку cookie-string путём обработки каждого cookie в соответствии с порядком их размещения в cookie-list:

   1. Выводится имя cookie, знак равенства (%x3D) и значение cookie.

   2. Если в списке cookie-list остаются необработанные cookie , выводятся символы %x3B и %x20 (;).

Примечание. Несмотря на своё имя, cookie-string в реальности представляет собой последовательность октетов, а не символов. Для преобразования cookie-string (или отдельных компонент) в последовательность символов (например, для представления пользователю) пользовательский агент может предпринять попытку использования кодировки UTF-8 [RFC3629]. При этом могут возникать отказы, поскольку не всякая последовательность октетов является корректной строкой UTF-8.

6. Вопросы реализации

6.1. Ограничения

Практические реализации пользовательских агентов имеют ограничения на число и размер cookie в хранилище. Агентам общего назначения **следует** обеспечивать перечисленные ниже минимальные возможности:

- не менее 4096 байтов на cookie (суммарный размер имени, значения и атрибутов cookie);
- не менее 50 cookie на домен;
- всего не менее 3000 cookie.

Серверам следует использовать небольшое число cookie незначительных размеров, чтобы избежать превышения приведённых выше ограничений и минимизировать расход пропускной способности сети на передачу заголовков Cookie в каждом запросе.

Серверам **следует** аккуратно снижать возможности (деградировать), если пользовательский агент не возвратил один или несколько cookie в заголовке Cookie, поскольку пользовательский агент имеет право по своему усмотрению или запросу пользователя удалять cookie в любой момент.

6.2. Интерфейс прикладных программ

Одна из причин использования в заголовках Cookie и Set-Cookie столь эзотерического синтаксиса заключается в том, что на многих платформах (как серверных, так и пользовательских) для cookie предоставляется основанный на строковых переменных интерфейс (API), требующий от разработчиков прикладных программ генерации и анализа синтаксиса, использованного в заголовках Cookie и Set-Cookie, который многие программисты реализуют некорректно, что приводит к проблемам совместимости.

Вместо предоставления для cookie интерфейса API на базе строковых переменных от платформ хотелось бы получить более осмысленный API. Рекомендации по конкретным API выходят за рамки этого документа, однако восприятие абстрактных объектов Date обеспечит явные преимущества по сравнению со строковым представлением дат.

6.3. Связь с IDNA

IDNA2008 [RFC5890] заменяет собой IDNA2003 [RFC3490]. Тем не менее, между этими спецификациями имеются различия и это может приводить к разной обработке (т. е., преобразованию) меток доменных имён, соответствующих разным спецификациям. В течение некоторого времени метки доменных имён на базе IDNA2003 будут сохраняться в сети. Пользовательским агентам **следует** поддерживать IDNA2008 [RFC5890] и **можно** поддерживать [UTS46] или [RFC5895] для упрощения перехода IDNA. Если в пользовательском агенте не реализуется IDNA2008, он **должен** поддерживать IDNA2003 [RFC3490].

7. Вопросы приватности

Cookie часто критикуют за возможность серверов отслеживать пользователей. Например, множество компаний web-анализа используют cookie для обнаружения возврата пользователя на сайт или его перехода на другой сайт. Хотя cookie является не единственным механизмом, позволяющим серверам отслеживать пользователей по запросам HTTP, cookie упрощают такое отслеживание поскольку они сохраняются в пользовательском агенте и могут совместно использоваться несколькими хостами.

7.1. Сторонние cookie

Особенно тревожат «сторонние cookie» (third-party). При воспроизведении документа HTML пользовательский агент часто запрашивает ресурсы с других серверов (типа рекламных сетей). Эти сторонние серверы могут использовать cookie для отслеживания даже тех пользователей, которые никогда не заходили на сервер напрямую. Например, если

пользователь посетит сайт, содержащий информацию от третьей стороны, а позднее посетит сайт, также содержащий информацию от этой третьей стороны, та сможет отследить посещение этим пользователем двух сайтов.

Некоторые пользовательские агенты ограничивают поведение сторонних cookie. Например, некоторые из таких агентов запрещают передачу заголовков Cookie в запросах к сторонним серверам. Другие агенты запрещают обработку заголовков Set-Cookie в откликах на запросы к сторонним серверам. Отношение к сторонним cookie существенно различается у разных пользовательских агентов. Этот документ предоставляет пользовательским агентам широкие возможности для экспериментов со сторонними cookie для поиска баланса между требованиями пользователей в плане приватности и совместимости. Однако этот документ не задаёт каких-либо конкретных правил для сторонних cookie.

Правила блокировки сторонних cookie зачастую не эффективны в плане сохранения приватности, если серверу нужно обойти ограничения в плане отслеживания пользователей. В частности, два взаимодействующих сервера могут зачастую отслеживать пользователей без применения cookie за счёт вставки идентифицирующих данных в динамические идентификаторы URL.

7.2. Пользовательские элементы управления

Пользовательским агентам **следует** предоставлять пользователю механизм управления cookie в своём хранилище. Например, пользовательский агент может разрешить удаление всех cookie, полученных в заданном интервале времени, или всех cookie, относящихся к определённому домену. Кроме того, многие пользовательские агенты поддерживают в своём интерфейсе элемент, позволяющий пользователю проверить cookie в хранилище.

Пользовательским агентам **следуе**т предоставлять пользователю механизм запрета cookie. При отключённых cookie пользовательскому агенту **недопустимо** включать заголовок Cookie в исходящие запросы HTTP, а также **недопустима** обработка заголовков Set-Cookie во входящих откликах HTTP.

Некоторые пользовательские агенты предоставляют возможность запрета сохранения cookie по завершении сессии. В таком режиме пользовательский агент **должен** трактовать все полученные cookie, как имеющие флаг persistent-flag со значением false. Некоторые популярные пользовательские агенты указывают такую функциональность, как режим private browsing [Aggarwal2010].

Некоторые пользовательские агенты предоставляют пользователю возможность одобрять отдельные записи в хранилище cookie. В большинстве случаев обычной работы такая возможность приводит к слишком частому выводу запросов на подтверждение записи. Однако некоторых пользователей защита приватности беспокоит сильнее, чем создаваемые таким режимом неудобства.

7.3. Сроки действия

Хотя серверы могут указывать срок завершения действия cookie в далёком будущем, большинство пользовательских агентов на практике не будет сохранять cookie в течение десятилетий. Вместо неразумно долгого срока действия cookie серверам **следует** повышать уровень приватности пользователей за счёт выбора разумного срока действия cookie с учётом задач их применения. Например, для типичного сеансового идентификатора разумно установить срок действия две недели.

8. Вопросы безопасности

8.1. Обзор

С cookie связано множество проблем безопасности. В этом разделе рассматриваются некоторые наиболее важные проблемы.

В частности, cookie подталкивают разработчиков полагаться на сомнительные основания при проверке подлинности, зачастую создающие уязвимости для атак типа поддержки кросс-сайтовых запросов [CSRF]. Аналогично, при сохранении сеансовых идентификаторов в cookie разработчики зачастую создают уязвимость, связанную с фиксацией сессии.

Шифрования на транспортном уровне, реализованного в HTTPS, не достаточно для предотвращения сетевых атак, связанных с перехватом и изменением cookie жертвы атаки, поскольку сам протокол cookie имеет множество уязвимостей (см. параграфы 8.5. Недостаточная конфиденциальность и 8.6. Недостаточная защита целостности). Кроме того, по умолчанию cookie не обеспечивают защиты конфиденциальности и целостности от сетевых атак даже при использовании с HTTPS.

8.2. Сомнительные основания

Сервер, использующий cookie для проверки подлинности пользователей, может оказаться уязвимым по причине того, что некоторые пользовательские агенты позволяют удалённой стороне подавать запросы HTTP от пользовательского агента (например, с помощью перенаправления HTTP или форм HTML). При таких запросах пользовательский агент добавляет cookie, даже если удалённая сторона не знает их содержимого, что позволяет той воспользоваться чужими полномочиями на ничего не знающем сервере.

Эту проблему называют по-разному (например, кросс-сайтовый обман (cross-site request forgery) или обманное делегирование (confused deputy)), она связана с использованием cookie в качестве основания для отождествления. Cookie побуждают операторов серверов отделять обозначение (в форме URL) от проверки полномочий (в форме cookie). Следовательно, пользовательский агент может предоставить полномочия для указанного атакующим ресурса и это может приводить к тому, что сервере или его клиенты предпримут заданные атакующим действия как будто они были уполномочены реальным пользователем.

Вместо применения cookie для проверки полномочий операторы могут рассмотреть возможность «объединить» обозначение и проверку полномочий за счёт трактовки идентификатора URL как возможности. Взамен записи секретных значений в cookie эта модель предусматривает их включение в идентификаторы URL, что требует от

удалённой стороны самостоятельно предоставлять секрет. Эта модель не решает всех проблем, но разумное применение этого принципа может привести к существенному росту уровня безопасности.

8.3. Открытый текст

Пока не используется защищённый канал (типа TLS), информация заголовков Cookie и Set-Cookie передаётся в открытом виде.

1. Вся «деликатная» информация из этих заголовков открыта для перехвата.
2. Враждебные промежуточные узлы могут менять содержимое заголовков при их передаче в обоих направлениях и результаты этого непредсказуемы.
3. Враждебный клиент может изменить заголовок Cookie до его передачи и результаты этого непредсказуемы.

Серверам **следует** шифровать и подписывать содержимое cookie (используя любой желаемый для сервера формат) при их передаче пользовательскому агенту (даже при передаче cookie по защищённому каналу). Однако шифрование и подписывание содержимого cookie не препятствует атакующему при «пересадке» cookie из одного пользовательского агента в другой или повторном использовании (replay) cookie.

В дополнение к шифрованию и подписыванию содержимого cookie серверам, которым требуется высокий уровень защиты, **следует** использовать заголовки Cookie и Set-Cookie только через защищённые каналы. В этом случае серверу **следует** устанавливать атрибут Secure (см. параграф 4.1.2.5) для каждого cookie. Если сервер не будет устанавливать атрибут Secure, защита, обеспечиваемая безопасным каналом, становится весьма спорной.

В качестве примера рассмотрим почтовый сервер (webmail), который хранит идентификатор сессии в cookie и обычно предоставляет доступ по протоколу HTTPS. Если сервер не устанавливает атрибут Secure в своих cookie, активный атакующий в сети может перехватить любой исходящий запрос HTTP от пользовательского агента и перенаправить этот запрос серверу webmail по HTTP. Даже если сервер webmail не принимает соединений HTTP, пользовательский агент все равно будет включать cookie в запрос. Активный атакующий в сети может перехватывать эти cookie и использовать их (replay) для контакта с сервером, получая доступ к электронной почте пользователя. Если же сервер установит атрибут Secure для своих cookie, пользовательский агент не будет включать cookie в открытые (незашифрованные) запросы.

8.4. Идентификаторы сессий

Вместо сохранения сеансовых данных в cookie (откуда они могут быть получены и использованы злоумышленником), серверы обычно сохраняют в cookie одноразовый идентификатор (nonce или session identifier). При получении сервером запроса HTTP с nonce, сервер может получить данные о состоянии, связанные с cookie, используя nonce в качестве ключа поиска.

Использование cookie с сеансовыми идентификаторами ограничивает возможности злоумышленника при раскрытии содержимого cookie, поскольку значение nonce полезно лишь для взаимодействия с сервером (в отличие от другого содержимого cookie, которое может быть конфиденциальным). Кроме того, использование одного nonce препятствует «сращиванию» атакующим содержимого cookie из разных взаимодействий с сервером, которое может приводить к непредсказуемому поведению сервера.

Использование сеансовых идентификаторов не избавляет от риска. Например, серверу **следует** принимать меры предотвращения уязвимости, связанной с «фиксацией сессии». Атаки этого типа выполняются в три этапа. Сначала атакующий переносит сеансовый идентификатор со своего пользовательского агента на агент жертвы. После этого жертва использует этот сеансовый идентификатор при взаимодействии с сервером, возможно включающем учётные данные пользователя или иную информацию. И, наконец, злоумышленник использует тот же сеансовый идентификатор для непосредственного взаимодействия с сервером и может при этом получить учётные данные пользователя или иную конфиденциальную информацию.

8.5. Недостаточная конфиденциальность

Доступ к cookie не ограничен каким-либо портом. Если данные cookie доступны для чтения службе, работающей на одном порту, они будут доступны и службе, работающей на другом порту того же сервера. Если cookie доступны для записи службе на одном порту, другая служба на другом порту того же сервера также будет иметь возможность записи. По этой причине серверам **не следует** запускать не доверяющие друг другу службы на разных портах одного хоста и использовать cookie для хранения связанной с безопасностью информации.

Cookie не изолированы по схеме. Хотя обычно они применяются со схемами http и https, cookie для данного хоста могут быть доступны и для других схем типа ftp или gopher. Несмотря на то, что этот недостаток изоляции связан в основном с интерфейсами API, не относящимися к HTTP и разрешающим доступ к cookie (например, интерфейс document.cookie API в HTML), отсутствие изоляции по схеме фактически присутствует в требованиях к обработке самих cookie (например, получение URI со схемой gopher через HTTP).

Cookies не всегда изолированы по пути. Хотя протокол сетевого уровня не передаёт cookie, сохранённые для одного пути в другой путь, некоторые пользовательские агенты раскрывают cookie через API, не относящиеся к HTTP, типа document.cookie API в HTML. Поскольку некоторые из таких пользовательских агентов (например, web-браузеры) не изолируют ресурсы, полученные из других путей, ресурсы полученные из одного пути, могут иметь доступ к cookie, сохраненным для другого пути.

8.6. Недостаточная защита целостности

Cookie не обеспечивают гарантий целостности для родственных (sibling) доменов и их субдоменов. Например, рассмотрим домены foo.example.com и bar.example.com. Сервер foo.example.com может установить cookie с атрибутом Domain = example.com (возможно, переписав существующий cookie example.com от bar.example.com) и пользовательский агент будет включать этот файл cookie в запросы HTTP к серверу bar.example.com. В худшем случае bar.example.com не сможет отличить этот cookie от своего. Сервер foo.example.com может усугубить эту ситуацию, организовав атаку на bar.example.com.

Даже при наличии в заголовке Set-Cookie атрибута Path этот атрибут не обеспечивает какой-либо защиты целостности, поскольку пользовательский агент будет принимать произвольный атрибут Path в заголовке Set-Cookie. Например, отклик HTTP на запрос http://example.com/foo/bar может установить cookie с атрибутом Path = /qux. По этой причине серверам **не следует** запускать не доверяющие друг другу службы на разных путях одного хоста и применять cookie для хранения связанной с безопасностью информации.

При активной сетевой атаке возможна также вставка cookie в заголовок Cookie, переданный по адресу https://example.com/, путём подмены отклика от http://example.com/ и вставки заголовка Set-Cookie. Сервер HTTPS на example.com не сможет отличить эти cookie от установленных им самим в отклике HTTPS. Атакующий сможет использовать это для организации атаки на example.com даже если этот сервер использует только HTTPS.

Серверы могут частично смягчить такие атаки, используя шифрование и подпись для содержимого своих cookie. Однако применение криптографии не решает проблему полностью, поскольку атакующий может воспроизвести (replay) cookie, полученные от настоящего сервера example.com server в сессии пользователя, с непредсказуемым результатом.

Кроме того, атакующий может вынудить пользовательский агент к удалению cookie путём записи большого числа cookie. После того, как пользовательский агент достигнет заданного предела, он будет вынужден удалить часть cookie. Серверам **не следует** полагаться на сохранение cookie пользовательскими агентами.

8.7. Зависимость от DNS

Cookie работают на основе DNS[1] в части обеспечения безопасности. При частичной или полной компрометации DNS протокол cookie может оказаться не способным обеспечить требуемую приложениями защиту.

9. Взаимодействие с IANA

Реестр полей заголовков сообщений (см. [RFC3864]) обновляется приведёнными ниже значениями.

9.1. Cookie

Название поля заголовка: Cookie

Применимый протокол: http

Статус: стандарт

Автор/контроль изменений: IETF

Спецификация: данный документ (параграф 5.4)

9.2. Set-Cookie

Название поля заголовка: Set-Cookie

Применимый протокол: http

Статус: стандарт

Автор/контроль изменений: IETF

Спецификация: данный документ (параграф 5.2)

9.3. Cookie2

Название поля заголовка: Cookie2

Применимый протокол: http

Статус: отменено

Автор/контроль изменений: IETF

Спецификация: [RFC2965]

9.4. Set-Cookie2

Название поля заголовка: Set-Cookie2

Применимый протокол: http

Статус: отменено

Автор/контроль изменений: IETF

Спецификация: [RFC2965]

10. Литература

10.1. Нормативные документы

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[1]Domain Name System - система доменных имён.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

В параграфе 6.3 указано, почему в числе нормативных документов указаны отменённые спецификации.

[RFC4790] Newman, C., Duerst, M., and A. Gulbrandsen, "Internet Application Protocol Collation Registry", RFC 4790, March 2007.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[USASCII] American National Standards Institute, "Coded Character Set -- 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

10.2. Дополнительная литература

[RFC2109] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", RFC 2109, February 1997.

[RFC2965] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", RFC 2965, October 2000.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[Netscape] Netscape Communications Corp., "Persistent Client State – HTTP Cookies", 1999, <http://web.archive.org/web/20020803110822/http://wp.netscape.com/newsref/std/cookie_spec.html>.

[Kri2001] Kristol, D., "HTTP Cookies: Standards, Privacy, and Politics", ACM Transactions on Internet Technology Vol. 1, #2, November 2001, <http://arxiv.org/abs/cs.SE/0105018>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC5895] Resnick, P. and P. Hoffman, "Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008", RFC 5895, September 2010.

[UTS46] Davis, M. and M. Suignard, "Unicode IDNA Compatibility Processing", Unicode Technical Standards # 46, 2010, <http://unicode.org/reports/tr46/>.

[CSRF] Barth, A., Jackson, C., and J. Mitchell, "Robust Defenses for Cross-Site Request Forgery", 2008, <http://portal.acm.org/citation.cfm?id=1455770.1455782>.

[Aggarwal2010] Aggarwal, G., Burzstein, E., Jackson, C., and D. Boneh, "An Analysis of Private Browsing Modes in Modern Browsers", 2010, <http://www.usenix.org/events/sec10/tech/full_papers/Aggarwal.pdf>.

Приложение A. Благодарности

Этот документ включает много заимствований из RFC 2109 [RFC2109]. Мы обязаны David M. Kristol и Lou Montulli за их усилия по спецификации cookie. David M. Kristol, в частности, предоставил неоценимые рекомендации про прохождению процесса IETF. Мы благодарим также Thomas Broyer, Tyler Close, Alissa Cooper, Bil Corry, corvid, Lisa Dusseault, Roy T. Fielding, Blake Frantz, Anne van Kesteren, Eran Hammer-Lahav, Jeff Hodges, Bjoern Hoehrmann, Achim Hoffmann, Georg Koppen, Dean McNamee, Alexey Melnikov, Mark Miller, Mark Pauley, Yngve N. Pettersen, Julian Reschke, Peter Saint-Andre, Mark Seaborn, Maciej Stachowiak, Daniel Stenberg, Tatsuhiro Tsujikawa, David Wagner, Dan Winship и Dan Witte за их полезные отклики на этот документ.

Адрес автора

Adam Barth

University of California, Berkeley

EMail: abarth@eecs.berkeley.edu

URI: http://www.adambarth.com/

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru