Искусство написания сценариев HTTP-запросов с использованием Curl

Похожие:
curl man page
Руководство
curlЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ И ответ

Предыстория

В этом документе предполагается, что вы знакомы с HTML и общими сетевыми технологиями.

Растущее количество приложений, переходящих в Интернет, сделало "HTTP Scripting" более часто запрашиваемым и востребованным. Иметь возможность автоматически извлекать информацию из Интернета, подделывать пользователей, публиковать или загружать данные на веб-серверы - все это важные задачи на сегодняшний день.

Curl - это инструмент командной строки для выполнения всевозможных манипуляций с URL и передач, но в этом конкретном документе основное внимание будет уделено тому, как использовать его при выполнении HTTP-запросов для развлечения и получения прибыли. В этих документах предполагается, что вы знаете, как вызвать `curl --help` или `curl --manual` получить базовую информацию об этом.

Curl написан не для того, чтобы делать все за вас. Он выполняет запросы, получает данные, отправляет данные и извлекает информацию. Вероятно, вам нужно склеить все вместе, используя какой-нибудь язык сценариев или повторяющиеся вызовы вручную.

Протокол HTTP

HTTP - это протокол, используемый для получения данных с веб-серверов. Это простой протокол, построенный на TCP / IP. Протокол также позволяет отправлять информацию на сервер от клиента с использованием нескольких различных методов, как будет показано здесь.

HTTP - это простые текстовые строки в формате ASCII, отправляемые клиентом серверу для запроса определенного действия, а затем сервер отвечает на несколько текстовых строк, прежде чем фактический запрошенный контент отправляется клиенту.

Клиент, curl, отправляет HTTP-запрос. Запрос содержит метод (например, GET, POST, HEAD и т.д.), Несколько заголовков запроса, А иногда и тело запроса. HTTP-сервер отвечает строкой состояния (указывающей, все ли прошло хорошо), заголовками ответов и чаще всего также телом ответа. Основная часть - это обычные данные, которые вы запросили, такие как фактический HTML или изображение и т.д.

Смотрите протокол

При использовании опции curl `--verbose` (`-v` в качестве краткой опции) будет отображаться, какие команды curl отправляет на сервер, а также несколько других информационных текстов.

`--verbose` это единственная наиболее полезная опция, когда дело доходит до отладки или даже понимания взаимодействия curl<-> с сервером.

Иногда даже `--verbose` этого недостаточно. Затем `--trace` и `--trace-ascii` предлагают еще больше деталей, поскольку они показывают **все**, что curl отправляет и получает. Используйте его следующим образом:

```
curl --trace-ascii debugdump.txt http://www.example.com/
```

Посмотрите на сроки

Часто вы можете задаться вопросом, что именно занимает все время, или вы просто хотите знать количество миллисекунд между двумя точками передачи. Для этих и других подобных ситуаций `--trace-time` опция - это то, что вам нужно. Он будет добавлять время к каждой строке вывода трассировки:

```
curl --trace-ascii d.txt --trace-time http://example.com/
```

Смотрите ответ

По умолчанию curl отправляет ответ в стандартный вывод. Вам нужно перенаправить его куда-нибудь, чтобы избежать этого, чаще всего это делается с помощью -o или -O.

URL

Спецификация

Формат Uniform Resource Locator - это способ указания адреса конкретного ресурса в Интернете. Вы знаете это, вы видели URL-адреса, подобные`https://curl.se` или `https://example.com` миллион раз. RFC 3986 является канонической спецификацией. И да, официальное имя - не URL, это URI.

Хост

Имя хоста обычно преобразуется с помощью DNS или вашего файла / etc / hosts в IP-адрес, и именно с ним curl будет взаимодействовать. В качестве альтернативы вы указываете IP-адрес непосредственно в URL вместо имени.

Для разработки и других ситуаций тестирования вы можете указать другой IP-адрес для имени хоста, чем тот, который использовался бы в противном случае, используя `--resolve` опцию curl:

```
curl --resolve www.example.org:80:127.0.0.1 http://www.example.org/
```

Номер порта

Каждый протокол, поддерживаемый curl, работает с номером порта по умолчанию, будь то через TCP или, в некоторых случаях, UDP. Обычно вам не нужно принимать это во внимание, но иногда вы запускаете тестовые серверы на других портах или аналогичных. Затем вы можете указать номер порта в URL-адресе с двоеточием и цифрой, следующей сразу за именем хоста. Например, при выполнении HTTP на порт 1234:

```
curl http://www.example.org:1234/
```

Номер порта, который вы указываете в URL, - это номер, который сервер использует для предоставления своих услуг. Иногда вы можете использовать прокси-сервер, и тогда вам может потребоваться указать номер порта этого прокси-сервера отдельно от того, который требуется curl для подключения к серверу. Например, при использовании HTTP-прокси на порту 4321:

```
curl --proxy http://proxy.example.org:4321 http://remote.example.org/
```

Имя пользователя и пароль

Некоторые службы настроены так, чтобы требовать HTTP-аутентификации, а затем вам необходимо предоставить имя и пароль, которые затем передаются на удаленный сайт различными способами в зависимости от используемого протокола аутентификации.

Вы можете либо вставить пользователя и пароль в URL-адрес, либо указать их отдельно:

```
curl http://user:password@example.org/
```

или

```
curl -u user:password http://example.org/
```

Вы должны обратить внимание, что этот вид HTTP-аутентификации - это не то, что обычно выполняется и запрашивается веб-сайтами, ориентированными на пользователя, в наши дни. Вместо этого они, как правило, используют формы и файлы cookie.

Часть пути

Часть пути просто отсылается на сервер с запросом, чтобы он отправил обратно соответствующий ответ. Путь - это то, что находится справа от косой черты, которая следует за именем хоста и, возможно, номером порта.

Извлеките страницу

ПОЛУЧИТЬ

Самый простой и наиболее распространенный запрос / операция, выполняемая с использованием HTTP, заключается в ПОЛУЧЕНИИ URL. URL-адрес сам по себе может ссылаться на веб-страницу, изображение или файл. Клиент отправляет запрос GET на сервер и получает документ, который он запросил. Если вы запустите командную строку

```
curl https://curl.se
```

вы получаете веб-страницу, возвращенную в окне вашего терминала. Весь HTML-документ, который содержит этот URL.

Все HTTP-ответы содержат набор заголовков ответов, которые обычно скрыты, используйте опцию curl `--include` (`-i`), чтобы отобразить их, а также остальную часть документа.

HEAD

Вы можете запросить у удаленного сервера ТОЛЬКО заголовки, используя `--head` (`-I`) опцию, которая заставит curl выдать запрос HEAD. В некоторых особых случаях серверы отказывают в использовании метода HEAD, в то время как другие все еще работают, что является особым видом раздражения.

Метод HEAD определен и выполнен таким образом, что сервер возвращает заголовки точно так, как это было бы сделано для GET, но без тела. Это означает, что вы можете видеть `Content-Length:` в заголовках ответа, но в ответе HEAD не должно быть фактического тела.

Несколько URL-адресов в одной командной строке

Одна командная строка curl может содержать один или несколько URL-адресов. Вероятно, наиболее распространенным случаем является использование только одного, но вы можете указать любое количество URL-адресов. Да, любой. Никаких ограничений. Затем вы будете получать запросы, повторяющиеся снова и снова для всех заданных URL.

Например, отправьте два запроса GET:

```
curl http://url1.example.com http://url2.example.com
```

Если вы используете `--data` для отправки по URL-адресу, использование нескольких URL-адресов означает, что вы отправляете одно и то же СООБЩЕНИЕ по всем указанным URL-адресам.

Например, отправить два сообщения:

```
curl --data name=curl http://url1.example.com http://url2.example.com
```

Несколько HTTP-методов в одной командной строке

Иногда вам нужно работать с несколькими URL-адресами в одной командной строке и использовать разные HTTP-методы для каждого. Для этого вам понравится `--next` опция. По сути, это разделитель, который отделяет набор опций от следующего. Все предыдущие URL-адреса `--next` будут получены одним и тем же методом и объединят все данные POST в один.

Когда curl достигает `--next` в командной строке, он как бы сбрасывает метод и данные POST и разрешает новый набор.

Возможно, это лучше всего показать на нескольких примерах. Для отправки сначала HEAD, а затем GET:

```
curl -I http://example.com --next http://example.com
```

Сначала отправить POST, а затем GET:

```
curl -d score=10 http://example.com/post.cgi --next http://example.com/results.html
```

HTML-формы

Объясненные формы

Формы - это общий способ, которым веб-сайт может представить HTML-страницу с полями, в которые пользователь может ввести данные, а затем нажать что-то вроде кнопки "OK" или "Отправить", чтобы отправить эти данные на сервер. Затем сервер обычно использует отправленные данные, чтобы решить, как действовать. Например, использование введенных слов для поиска в базе данных или для добавления информации в систему отслеживания ошибок, отображение введенного адреса на карте или использование информации в качестве приглашения для входа, подтверждающего, что пользователю разрешено видеть то, что он собирается увидеть.

Конечно, на стороне сервера должна быть какая-то программа для получения отправляемых вами данных. Вы не можете просто изобрести что-то из воздуха.

ПОЛУЧИТЬ

GET-форма использует метод GET, как указано в HTML, например:

```
<form method="GET" action="junk.cgi">
  <input type=text name="birthyear">
  <input type=submit name=press value="OK">
</form>
```

В вашем любимом браузере появится эта форма с текстовым полем для заполнения и кнопкой с надписью "OK". Если вы введете '1905' и нажмете кнопку OK, ваш браузер создаст новый URL, который будет доступен для вас. URL будет `junk.cgi?birthyear=1905&press=OK` добавлен к части пути предыдущего URL.

Если на странице была видна исходная форма `www.example.com/when/birth.html`, вторая страница, которую вы получите, станет `www.example.com/when/junk.cgi?birthyear=1905&press=OK`.

Большинство поисковых систем работают таким образом.

Чтобы заставить curl выполнить отправку формы GET за вас, просто введите ожидаемый созданный URL:

```
curl "http://www.example.com/when/junk.cgi?birthyear=1905&press=OK"
```

Публикация

Метод GET позволяет отображать имена всех полей ввода в поле URL вашего браузера. Обычно это хорошо, когда вы хотите иметь возможность добавлять в закладки эту страницу с заданными вами данными, но это очевидный недостаток, если вы ввели секретную информацию в одно из полей или если существует большое количество полей, создающих длинный и нечитаемый URL.

Затем протокол HTTP предлагает метод POST. Таким образом, клиент отправляет данные отдельно от URL, и, таким образом, вы ничего из этого не увидите в поле URL-адреса.

Форма будет выглядеть аналогично предыдущей:

```
<form method="POST" action="junk.cgi">
  <input type=text name="birthyear">
  <input type=submit name=press value=" OK ">
</form>
```

И чтобы использовать curl для публикации этой формы с теми же данными, что и раньше, мы могли бы сделать это следующим образом:

```
curl --data "birthyear=1905&press=%20OK%20" http://www.example.com/when/junk.cgi
```

Этот тип ПУБЛИКАЦИИ будет использовать Content-Type application/x-www-form-urlencoded и является наиболее широко используемым типом публикации.

Данные, которые вы отправляете на сервер, уже должны быть правильно закодированы, curl не сделает этого за вас. Например, если вы хотите, чтобы данные содержали пробел, вам нужно заменить этот пробел на %20 и т.д. Несоблюдение этого правила, скорее всего, приведет к неправильному получению ваших данных и их искажению.

Последние версии curl фактически могут кодировать POST-данные с URL-адресом для вас, вот так:

```
curl --data-urlencode "name=I am Daniel" http://www.example.com
```

Если вы повторите --data несколько раз в командной строке, curl объединит все заданные фрагменты данных - и поместит & символ между каждым сегментом данных.

Сообщение о загрузке файла

Еще в конце 1995 года они определили дополнительный способ отправки данных через HTTP. Это задокументировано в RFC 1867, почему этот метод иногда называют RFC1867-posting.

Этот метод в основном предназначен для лучшей поддержки загрузки файлов. Форма, которая позволяет пользователю загружать файл, может быть написана следующим образом в HTML:

```
<form method="POST" enctype='multipart/form-data' action="upload.cgi">
  <input type=file name=upload>
  <input type=submit name=press value="OK">
</form>
```

Это ясно показывает, что тип контента, который будет отправлен, является multipart/form-data.

Для отправки в форму, подобную этой, с помощью curl вы вводите командную строку типа:

```
curl --form upload=@localfilename --form press=OK [URL]
```

Скрытые поля

Распространенный способ для приложений на основе HTML передавать информацию о состоянии между страницами - добавлять скрытые поля в формы. Скрытые поля уже заполнены, они не отображаются пользователю и передаются вместе со всеми остальными полями.

Аналогичный пример формы с одним видимым полем, одним скрытым полем и одной кнопкой отправки может выглядеть следующим образом:

```
<form method="POST" action="foobar.cgi">
  <input type=text name="birthyear">
  <input type=hidden name="person" value="daniel">
  <input type=submit name="press" value="OK">
</form>
```

Чтобы ОПУБЛИКОВАТЬ это с помощью curl, вам не придется думать о том, скрыты поля или нет. Для curl все они одинаковы:

```
curl --data "birthyear=1905&press=OK&person=daniel" [URL]
```

Выясните, как выглядит POST

Когда вы собираетесь заполнить форму и отправить ее на сервер, используя curl вместо браузера, вы, конечно, заинтересованы в отправке сообщения именно так, как это делает ваш браузер.

Простой способ увидеть это - сохранить HTML-страницу с формой на вашем локальном диске, изменить "метод" на GET и нажать кнопку отправки (вы также можете изменить URL действия, если хотите).

Затем вы ясно увидите, что данные добавляются к URL-адресу, разделяясь ? буквой, как и положено формам GET.

Загрузка по HTTP

ПОМЕСТИТЬ

Возможно, лучший способ загрузить данные на HTTP-сервер - использовать PUT. Опять же, для этого, конечно, требуется, чтобы кто-то разместил программу или скрипт на стороне сервера, который знает, как получать поток HTTP PUT.

Поместите файл на HTTP-сервер с помощью curl:

```
curl --upload-file uploadfile http://www.example.com/receive.cgi
```

HTTP-аутентификация

Базовая аутентификация

HTTP-аутентификация - это возможность сообщить серверу ваше имя пользователя и пароль, чтобы он мог убедиться, что вам разрешено выполнить выполняемый вами запрос. Базовая аутентификация, используемая в HTTP (которая является типом, используемым curl по умолчанию), основана на **обычном тексте**, что означает, что имя пользователя и пароль отправляются лишь слегка запутанными, но все еще полностью читаемыми любым, кто прослушивает сеть между вами и удаленным сервером.

Чтобы указать curl использовать пользователя и пароль для аутентификации:

```
curl --user name:password http://www.example.com
```

Другая аутентификация

Сайту может потребоваться другой метод аутентификации (проверьте заголовки, возвращаемые сервером), и тогда `--ntlm`, `--digest --negotiate` или даже `--anyauth` могут быть подходящие вам варианты.

Аутентификация через прокси

Иногда ваш HTTP-доступ доступен только с помощью HTTP-прокси. Кажется, это особенно распространено в различных компаниях. HTTP-прокси может потребовать своего собственного пользователя и пароль, чтобы позволить клиенту подключиться к Интернету. Чтобы указать их с помощью curl, запустите что-то вроде:

```
curl --proxy-user proxyuser:proxypassword curl.se
```

Если ваш прокси требует, чтобы аутентификация выполнялась с использованием метода NTLM, используйте `--proxy-ntlm`, если для этого требуется использовать дайджест `--proxy-digest`.

Если вы используете любой из этих параметров user + password, но не указываете часть пароля, curl запросит пароль в интерактивном режиме.

Скрытие учетных данных

Обратите внимание, что при запуске программы ее параметры можно увидеть при перечислении запущенных процессов системы. Таким образом, другие пользователи смогут просматривать ваши пароли, если вы передадите их как простые параметры командной строки. Есть способы обойти это.

Стоит отметить, что, хотя именно так работает HTTP-аутентификация, многие веб-сайты не будут использовать эту концепцию при предоставлении логинов и т.д. Смотрите главу веб-входа ниже для получения более подробной информации об этом.

Больше HTTP-заголовков

Референт

HTTP-запрос может включать поле 'referer' (да, оно написано с ошибкой), которое может использоваться для определения того, с какого URL клиент попал на этот конкретный ресурс. Некоторые программы / скрипты проверяют поле referer запросов, чтобы убедиться, что это не поступило с внешнего сайта или неизвестной страницы. Хотя это глупый способ проверить что-то, что так легко подделать, многие скрипты все еще делают это. Используя curl, вы можете поместить все, что хотите, в referer-field и, таким образом, легче обмануть сервер, заставив его выполнить ваш запрос.

Используйте curl для задания поля refererer с:

```
curl --referer http://www.example.come http://www.example.com
```

Пользовательский агент

Подобно полю refererer, все HTTP-запросы могут устанавливать поле User-Agent. Он определяет, какой пользовательский агент (клиент) используется. Многие приложения используют эту информацию, чтобы решить, как отображать страницы. Глупые веб-программисты пытаются создавать разные страницы для пользователей разных браузеров, чтобы они выглядели наилучшим образом для своих конкретных браузеров. Обычно они также используют различные виды JavaScript и т.д.

Иногда вы увидите, что при получении страницы с помощью curl не будет возвращена та же страница, которую вы видите при получении страницы с помощью вашего браузера. Тогда вы знаете, что пришло время задать поле User Agent, чтобы обмануть сервер, заставив его думать, что вы являетесь одним из этих браузеров.

Чтобы curl выглядел как Internet Explorer 5 в Windows 2000:

```
curl --user-agent "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)" [URL]
```

Или почему бы не выглядеть так, как будто вы используете Netscape 4.73 на старой коробке Linux:

```
curl --user-agent "Mozilla/4.73 [en] (X11; U; Linux 2.2.15 i686)" [URL]
```

Перенаправляет

Заголовок местоположения

Когда ресурс запрашивается с сервера, ответ с сервера может содержать подсказку о том, куда браузеру следует перейти дальше, чтобы найти эту страницу, или новую страницу, содержащую недавно сгенерированный вывод. Заголовок, который сообщает браузеру о перенаправлении, является Location:.

Curl по умолчанию не следует за Location: заголовками, а просто отображает такие страницы таким же образом, как и все HTTP-ответы. Однако в нем есть опция, которая заставит его пытаться следовать Location: указателям.

Чтобы указать curl следовать местоположению:

```
curl --location http://www.example.com
```

Если вы используете curl для ПУБЛИКАЦИИ на сайте, который немедленно перенаправляет вас на другую страницу, вы можете безопасно использовать --location (-L) и --data/--form вместе. Curl будет использовать POST только в первом запросе, а затем вернется к GET в следующих операциях.

Другие перенаправления

Браузеры обычно поддерживают по крайней мере два других способа перенаправления, которых нет в curl: во-первых, html может содержать тег meta refresh, который запрашивает браузер загрузить определенный URL-адрес через заданное количество секунд, или для этого может использоваться JavaScript.

Файлы cookie

Основы использования файлов cookie

Способ, которым веб-браузеры осуществляют "контроль состояния на стороне клиента", заключается в использовании файлов cookie. Cookies - это просто имена с соответствующим содержимым. Файлы cookie отправляются клиенту сервером. Сервер сообщает клиенту, по какому пути и имени хоста он хочет, чтобы cookie был отправлен обратно, а также отправляет дату истечения срока действия и еще несколько свойств.

Когда клиент обменивается данными с сервером с именем и путем, ранее указанными в полученном файле cookie, клиент отправляет обратно файлы cookie и их содержимое на сервер, если, конечно, срок их действия не истек.

Многие приложения и серверы используют этот метод для соединения серии запросов в один логический сеанс. Чтобы иметь возможность использовать curl в таких случаях, мы должны иметь возможность записывать и отправлять обратно файлы cookie так, как их ожидает веб-приложение. Точно так же с ними справляются браузеры.

Параметры файлов cookie

Самый простой способ отправить несколько файлов cookie на сервер при получении страницы с curl - это добавить их в командную строку, например:

```
curl --cookie "name=Daniel" http://www.example.com
```

Файлы cookie отправляются в виде обычных HTTP-заголовков. Это практично, поскольку позволяет curl записывать файлы cookie просто путем записи заголовков. Записывайте файлы cookie с помощью curl, используя --dump-header (-D) опцию, подобную:

```
curl --dump-header headers_and_cookies http://www.example.com
```

(Обратите внимание, что --cookie-jar описанный ниже вариант является лучшим способом хранения файлов cookie.)

В Curl встроен полнофункциональный механизм анализа файлов cookie, который можно использовать, если вы хотите повторно подключиться к серверу и использовать файлы cookie, сохраненные при предыдущем подключении (или созданные вручную, чтобы обмануть сервер и заставить его поверить, что у вас было предыдущее соединение). Чтобы использовать ранее сохраненные файлы cookie, вы запускаете curl как:

```
curl --cookie stored_cookies_in_file http://www.example.com
```

"Механизм cookie" Curl включается при использовании --cookie опции. Если вы хотите, чтобы curl понимал только полученные файлы cookie, используйте --cookie с файлом, который не существует. Например, если вы хотите, чтобы curl понимал файлы cookie со страницы и отслеживал местоположение (и, таким образом, возможно, отправлял обратно полученные файлы cookie), вы можете вызвать его следующим образом:

```
curl --cookie nada --location http://www.example.com
```

Curl имеет возможность считывать и записывать файлы cookie, которые используют тот же формат файла, который когда-то использовали Netscape и Mozilla. Это удобный способ обмена файлами cookie между сценариями или вызовами. Переключатель --cookie (-b) автоматически определяет, является ли данный файл таким файлом cookie, и анализирует его, а с помощью опции --cookie-jar (-c) вы заставите curl записать новый файл cookie в конце операции:

```
curl --cookie cookies.txt --cookie-jar newcookies.txt http://www.example.com
```

HTTPS

HTTPS - это HTTP безопасный

Существует несколько способов безопасной передачи по HTTP. На сегодняшний день наиболее распространенным протоколом для этого является то, что обычно известно как HTTPS, HTTP поверх SSL. SSL шифрует все данные, отправляемые и принимаемые по сети, и, таким образом, затрудняет злоумышленникам слежку за конфиденциальной информацией.

Протокол SSL (или TLS, как называется текущая версия стандарта) предлагает набор расширенных функций для безопасной передачи данных по протоколу HTTP.

Curl поддерживает зашифрованные выборки, когда создан для использования библиотеки TLS, и он может быть создан для использования одной из довольно большого набора библиотек - curl -V покажет, для какой из них был создан ваш curl (если таковой имеется!).). Чтобы получить страницу с сервера HTTPS, просто запустите curl следующим образом:

```
curl https://secure.example.com
```

Сертификаты

В мире HTTPS вы используете сертификаты для подтверждения того, что вы тот, за кого себя выдаете, в дополнение к обычным паролям. Curl поддерживает сертификаты на стороне клиента. Все сертификаты заблокированы парольной фразой, которую вам необходимо ввести, прежде чем сертификат сможет быть использован curl. Фразу-пароль можно указать в командной строке или, если нет, ввести в интерактивном режиме, когда curl запрашивает ее. Используйте сертификат с curl на сервере HTTPS, например:

```
curl --cert mycert.pem https://secure.example.com
```

curl также пытается проверить, является ли сервер тем, за кого он себя выдает, путем сверки сертификата сервера с локально сохраненным пакетом сертификатов CA. Сбой проверки приведет к тому,

что curl откажет в подключении. Затем вы должны использовать --insecure (-k) на случай, если хотите сообщить curl, чтобы он игнорировал, что сервер не может быть проверен.

Подробнее о проверке сертификата сервера и пакетах сертификатов ca можно прочитать в SSLCERTS документе.

Иногда у вас может получиться собственное хранилище сертификатов CA, и тогда вы можете указать curl использовать это для проверки сертификата сервера:

```
curl --cacert ca-bundle.pem https://example.com/
```

Пользовательские элементы запроса

Измените метод и заголовки

Выполняя необычные вещи, вам может потребоваться добавить или изменить элементы одного запроса curl.

Например, вы можете изменить метод POST на PROPFIND и отправлять данные как Content-Type: text/xml (вместо значения по умолчанию Content-Type) вот так:

```
curl --data "<xml>" --header "Content-Type: text/xml"   --request PROPFIND example.cc
```

Вы можете удалить заголовок по умолчанию, предоставив заголовок без содержимого. Например, вы можете испортить запрос, отрезав Host: заголовок:

```
curl --header "Host:" http://www.example.com
```

Вы можете добавлять заголовки таким же образом. Вашему серверу может потребоваться Destination: заголовок, и вы можете его добавить:

```
curl --header "Destination: http://nowhere" http://example.com
```

Подробнее об измененных методах

Следует отметить, что curl выбирает, какие методы использовать самостоятельно, в зависимости от того, какое действие запрашивать. -d будет делать POST, -I будет делать HEAD и так далее. Если вы используете опцию --request / -X, вы можете изменить ключевое слово метода, которое выбирает curl, но вы не измените поведение curl. Это означает, что если вы, например, используете -d "data" для создания POST, вы можете изменить метод на PROPFIND with -X, и curl все равно будет думать, что он отправляет POST . Вы можете изменить обычный метод GET на POST, просто добавив -X POST в командной строке, например:

```
curl -X POST http://example.org/
```

... но curl все равно будет думать и действовать так, как если бы он отправил GET, поэтому он не будет отправлять тело запроса и т.д.

Веб-вход

Некоторые приемы входа в систему

Хотя это не связано строго только с HTTP, оно по-прежнему вызывает у многих людей проблемы, поэтому вот краткое изложение того, как работает подавляющее большинство всех форм входа и как войти в них с помощью curl.

Также можно отметить, что для правильного выполнения этого в автоматическом режиме вам, безусловно, потребуется написать сценарий, выполнить несколько вызовов curl и т.д.

Во-первых, серверы в основном используют файлы cookie для отслеживания статуса входа клиента в систему, поэтому вам нужно будет фиксировать файлы cookie, которые вы получаете в ответах. Кроме того, многие сайты также устанавливают специальный файл cookie на странице входа (чтобы убедиться, что вы попали туда через их страницу входа), поэтому вам следует завести привычку сначала получать страницу формы входа, чтобы зафиксировать установленные там файлы cookie.

Некоторые веб-системы входа в систему содержат различное количество JavaScript, и иногда они используют такой код для установки или изменения содержимого cookie. Возможно, они делают это, чтобы предотвратить запрограммированные входы в систему, как описано в этом руководстве... В любом случае, если чтения кода недостаточно, чтобы позволить вам повторить поведение вручную, захват HTTP-запросов, выполняемых вашими браузерами, и анализ отправленных файлов cookie обычно являются рабочим методом для определения того, как сократить время выполнения JavaScript.

В самом <form> теге для входа в систему множество сайтов заполняют случайные / сессионные или иным образом тайно сгенерированные скрытые теги, и вам, возможно, потребуется сначала захватить HTML-код для формы входа и извлечь все скрытые поля, чтобы иметь возможность опубликовать правильный вход. Помните, что содержимое должно быть закодировано по URL при отправке в обычном POST.

Отладка

Некоторые приемы отладки

Часто, когда вы запускаете curl на сайте, вы замечаете, что сайт, похоже, не реагирует на ваши запросы curl так же, как на запросы вашего браузера.

Затем вам нужно начать делать ваши запросы curl более похожими на запросы вашего браузера:

- Используйте `--trace-ascii` опцию для хранения полностью подробных журналов запросов для упрощения анализа и лучшего понимания
- Убедитесь, что вы проверяете и используете cookies при необходимости (как для чтения с помощью `--cookie`, так и для записи с помощью `--cookie-jar`)
- Установите user-agent (с `-A`) на один, как это делает недавний популярный браузер
- Установите refererer (с `-E`) так, как он установлен браузером
- Если вы используете POST, убедитесь, что отправляете все поля в том же порядке, в каком это делает браузер.

Проверьте, что делают браузеры

Хорошим помощником в том, чтобы убедиться, что вы делаете это правильно, являются инструменты разработчиков веб-браузеров, которые позволяют просматривать все заголовки, которые вы отправляете и получаете (даже при использовании HTTPS).

Более простой подход заключается в том, чтобы перехватывать HTTP-трафик в сети с помощью таких инструментов, как Wireshark или tcpdump, и проверять, какие заголовки были отправлены и получены браузером. (HTTPS вынуждает вас использовать SSLKEYLOGFILE для этого.)