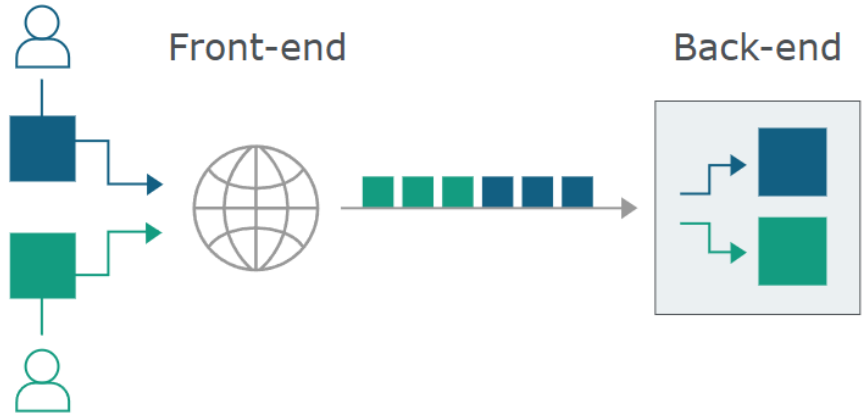
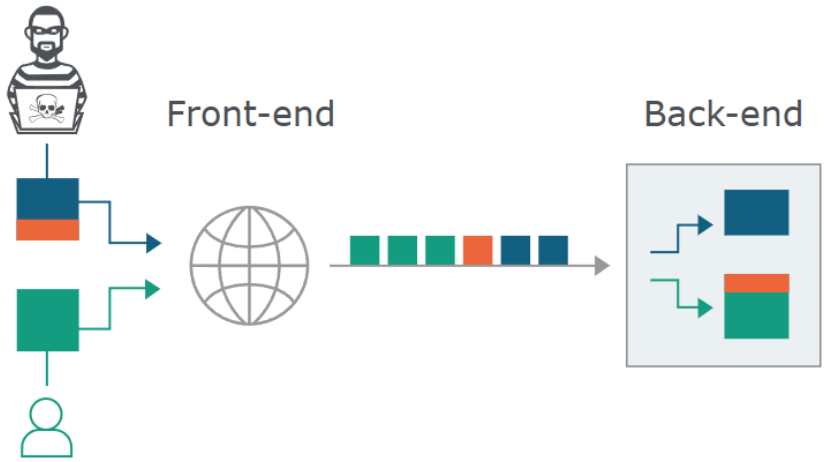
HTTP request smuggling

Внедрение в последовательность http запросов. Может быть критичной для HTTP/1, может работать и для HTTP/2 Критическая уязвимость, позволяющая получить доступ к данным и скомпроментировать пользователей.

При архитектуре proxy/load balancer -> backend, балансировщиком отправляется несколько запросов на бэк. Запросы отправляются последовательно, и бэк обязан определять где заканчивается один и начинается следующий.



Важно, чтобы фронт и бэк согласовали границы между запросами. Иначе можно отправить неоднозначный запрос, который будет по-разному интерпретирован интерфейсной и серверной системами



Это работает, поскольку в HTTP/1 есть два варианта определения завершения запроса: заголовки Content-Length и Transfer-Encoding. Content-Length это длина сообщения в байтах.

```
POST /search HTTP/1.1
Host: normal-website.com
```

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

q=smuggling
```

Transfer-Encoding означает, что текст сообщения содержит один или несколько фрагментов данных. Каждый фрагмент содержит размер фрагмента в байтах, перевод на новую строку, сам фрагмент. Сообщение завершается 0 размером фрагмента.

```
POST /search HTTP/1.1
Host: normal-website.com
Content-Type: application/x-www-form-urlencoded
Transfer-Encoding: chunked

b
q=smuggling
0
```

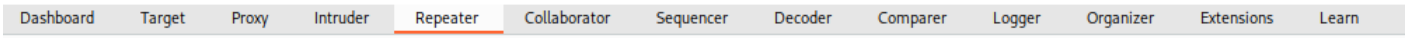
Браузеры не отправляют Transfer-Encoding запросы, это актуально для межсерверного взаимодействия. Так как возможно два варианта, в одном сообщении можно использовать оба заголовка, как будто бы они конфликтуют. Transfer-Encoding приоритетнее, поэтому Content-Length игнорируется. Это эффективно если один бэк, если несколько - проблема:

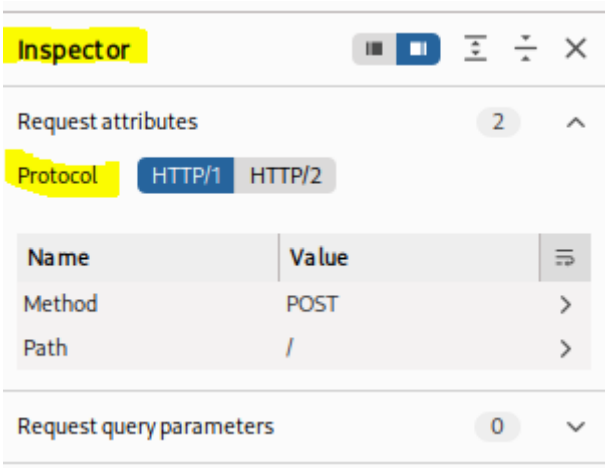
- некоторые серверы не поддерживают заголовок Transfer-Encoding
- могут поддерживать, но не обработать его, если заголовок запутан.

Если фронт и бэк по-разному ведут себя при обработке заголовка Transfer-Encoding, то могут перепутаться границы запросов. Когда фронт HTTP/2, а бэк HTTP/1 (HTTP downgrading), то будет преобразование форматов.

Вектор атаки

Браузеры и другие клиенты, включая Burp, используют http/2 по умолчанию. Поэтому в Burp необходимо включить http/1 (Inspector -> Request attributes).





Классика: отправка http/1 с двумя заголовками. Финальная реализация зависит от фронта и бэка:

- TE.CL: фронт использует Transfer-Encoding, бэк Content-Length
- TE.TE: фронт и бэк используют Transfer-Encoding, но один из них не обрабатывает его в связи с обфускацией

CL.TE: фронт использует Content-Length, бэк Transfer-Encoding

Пример запроса:

```
POST / HTTP/1.1
Host: vulnerable-website.com
Content-Length: 13
Transfer-Encoding: chunked

0

SMUGGLED
```

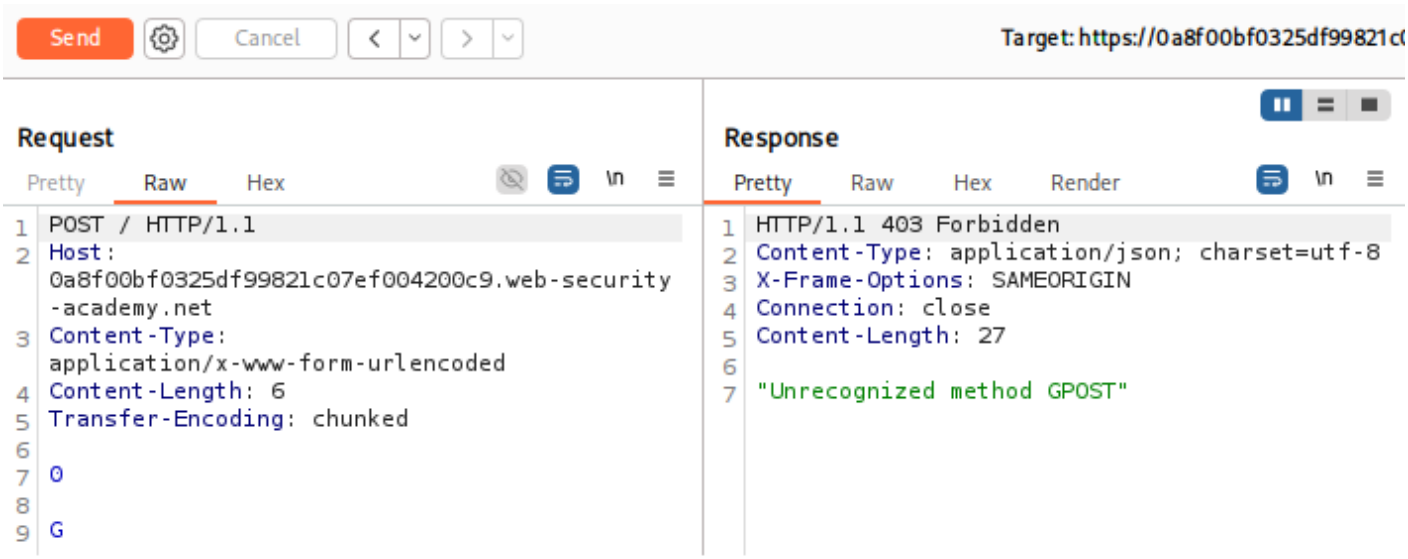
Для решения лабораторной работы предлагается отправить запрос вида

```
POST / HTTP/1.1
Host: YOUR-LAB-ID.web-security-academy.net
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 6
Transfer-Encoding: chunked

0
```

```
G
```

Из-за того, что сервер не поддерживает метод gpost, он должен вернуть ошибку после второй отправки запроса.



Правда почему content-length: 6 - неясно. Возможно перевод строки - 1, 0 - 2, перевод - 1, G - 2

Revision #8
Created 1 September 2025 15:28:55 by Admin
Updated 7 September 2025 14:32:46 by Admin