

[Skip to site navigation \(Press enter\)](#)

[ANNOUNCE] HTTP/2 vulnerabilities from 2.0 to 2.5-dev

Willy Tarreau Tue, 17 Aug 2021 08:14:13 -0700

Hi everyone,

HAProxy is affected by 4 vulnerabilities in its HTTP/2 implementation in recent versions (starting with 2.0). Three of them are considered as having a moderate impact as they only affect the interpretation of the authority (Host header field) in H2->H2 communications in versions 2.2 and above. One only affects a risk of misinterpretation from lenient HTTP/1 backend servers, and affects version 2.0 and above, though at the time of writing we're not aware of any such vulnerable server among the mainstream ones that are commonly found behind HAProxy (Apache, NGINX, Varnish, etc).

Background: on Aug 05 a research article was published about flaws affecting some HTTP/2 to HTTP/1 proxies or gateways:

<https://portswigger.net/research/http2>

A first analysis of the article compared to some selected pieces of code concluded that haproxy was safe. This was actually wrong as only older versions are safe (2.0 in "legacy" mode and 1.8). Tim Dusterhus conducted some additional tests and found some problems, which after digging for a few days, revealed to be significantly more embarrassing. In practice, version 2.0 in the default "HTX" mode and all later versions are affected to some degrees.

1) Spaces in the ":method" field

The first problem is based on the ":method" field. By passing a space in the method, it is possible to build an invalid HTTP/1 request on the backend side, which some lenient servers might possibly interpret as valid, resulting in a different request between the one seen by haproxy and by the server. This might be abused to circumvent some switching rules for example, and get a request to be routed to a wrong server. Example:

```
H2 request
:method: "GET /admin? HTTP/1.1"
:path:   "/static/images"
```

HAProxy would route all "/static" requests to the static server farm, but once the request is reassembled it would become this:

```
GET /admin? HTTP/1.1 /static/images HTTP/1.1
```

This is not valid but if a server fails to properly validate this input, it might be fooled into thinking this is a request for /admin.

Please note that HTTP/2 backend servers are not affected as the request is sent as a new ":method" field there. Additionally, dangerous characters like CR, LF or NUL are always blocked on input so it is not possible to perform a request smuggling attack, and the risks are limited to HTTP/1 servers which fail to properly parse the request line (note that all major server implementations are safe against this).

A workaround for this issue for those having to rely on possibly unsafe servers is to reject invalid characters in the method by placing such a filtering rule on the request path either in the frontend or the backend:

```
http-request reject if { method -m reg ['A-Z0-9'] }
```

A second workaround that may only be used on version 2.0 consists in disabling the HTX internal representation in the affected backends and the frontends that route to them:

```
no option http-use-htx
```

This will have for effect to transform the HTTP/2 requests to HTTP/1 that will then be submitted to the internal HTTP/1 parser which will reject the poorly formatted request. This older representation called "legacy" is not available any more in version 2.1 and above, and is not compatible with HTTP/2 nor FastCGI backend servers.

This issue affects all versions from 2.0 and above, in HTX mode, with HTTP/1 on the server side.

2) Domain parts in ":scheme" and ":path"

The ":scheme" HTTP/2 header field contains the scheme that prefixes the request URI, typically "http" or "https". ":path" contains the part that is local to the target host, and that usually starts with the "/". By appending a part of a request to ":scheme" or by prepending a part of a domain name to ":path", it is possible to make haproxy and a backend server see a different authority or URL prefix. Please note that this only affects HTTP/2 servers on versions 2.2 and above. These versions are indeed capable of passing an absolute URI from end to end, while earlier versions were limited to origin URIs. In addition, HTTP/2 requests are always forwarded in origin form to HTTP/1 backend servers (i.e. they do not have a scheme nor authority parts). As such HTTP/1 servers are safe and only HTTP/2 servers are exposed.

What happens is that on the entry point, the :scheme, :authority and :path fields are concatenated to rebuild a full URI that is then passed along the chain, but the Host header is set from :authority before this concatenation is performed. As such, the Host header field used internally may not always match the authority part of the recomposed URI. Examples:

```
H2 request
:method: "GET"
:scheme: "http://localhost?orig=example.org";
:authority "example.org"
:path:    "/"
```

or:

```
H2 request
:method: "GET"
:scheme: "http"
:authority "example.org"
:path:    ".local/"
```

An internal Host header will be built with "example.org" then the complete URI will become "<http://localhost?orig=example.org/>"; in the first example, or "<http://example.org.local/>"; in the second example, and this URI will be used to build the HTTP/2 request on the server side, dropping the unneeded Host header field. In HTTP/1 there is no such issue as the URI is dropped and the Host is kept. Thus if the configuration contains some routing rules based on the Host header field, a target HTTP/2 server might receive a different :authority than the one that was expected to be routed there.

A workaround consists in rewriting the URI as itself before processing the Host header field, which will have for effect to resynchronize the Host header field with the recomposed URI, making sure both haproxy and the backend server will always see the same value:

```
http-request set-uri %[url]
```

3) Mismatch between ":authority" and "Host"

The HTTP/2 specification (RFC7540) implicitly allows the "Host" header and the ":authority" header field to differ and further mentions that the contents of ":authority" may be used to build "Host" if this one is missing. This results in an ambiguous situation analogue to the one above, because rules built based on the "Host" field will match against a possibly different "Host" header field that will be dropped when the request is forwarded to an HTTP/2 backend server. An HTTP/1 server will not be affected since HTTP/2 requests are forwarded to HTTP/1 in origin form, i.e. without the authority part. Example:

```
H2 request
:method:   "GET"
:scheme:   "http"
:authority "victim.com"
:path:     "/"
Host:      "example.org"
```

Internal switching rules using the "Host" header field will see "example.org" but when the request is passed to an H2 server, "Host" will be dropped and "victim.com" will be used by this server to fill the missing "Host" header.

The new H2 specification in progress ("http2bis") addresses this issue by proposing that "Host" is always ignored on input in favor of ":authority" which remains more consistent with what is done along the chain. This is the solution adopted by the fix here.

A workaround consists in using the same rule as for the previous issue, before the Host header field is used by any switching rule (typically in the frontend), which will have for effect to rewrite the "Host" part according to the contents of the ":authority" field:

```
http-request set-uri %[url]
```

4) Affected versions

- versions 1.7 do not support H2 and are not affected
- versions 1.8 only support H2 legacy mode are not affected
- versions 2.0 prior to 2.0.24 are affected by the :method bug
- versions 2.2 prior to 2.2.16 are affected by all 4 bugs
- versions 2.3 prior to 2.3.13 are affected by all 4 bugs
- versions 2.4 prior to 2.4.3 are affected by all 4 bugs
- versions 2.5 prior to 2.5-dev4 are affected by all 4 bugs

5) Instant remediation

Several solutions are usable against all of these issues in affected versions before upgrading:

- disabling HTTP/2 communication with servers by removing "proto h2" from "server" lines is sufficient to address the ":authority", ":scheme", and ":path" issues if the servers are known *not* to be vulnerable to the issue described in the ":method" attack above. This probably is the easiest solution when using trusted mainstream backend servers such as Apache, NGINX or Varnish, especially since very few configurations make use of H2 to communicate with servers.

- placing the two following rules at the beginning of every HTTP frontend:

```
http-request reject if { method -m reg [^A-Z0-9] }
http-request set-uri %[url]
```

- in version 2.0, disabling HTX processing will force the request to be reprocessed by the internal HTTP/1 parser (but this is not compatible with H2 servers nor FastCGI servers):

```
no option http-use-htx
```

- commenting out "alpn h2" advertisement on all "bind" lines in frontends, and disabling H2 processing entirely by placing the following line in the global section:

```
tune.h2.max-concurrent-streams 0
```

- in versions 2.2 and above it is possible to refine filtering per frontend by disabling "alpn h2" per bind line and by disabling HTTP/1 to HTTP/2 upgrade by placing this option in the respective frontends:

```
option disable-h2-upgrade
```

Many thanks to Tim for helping getting these issues resolved!
Willy

- [Previous message](#)
- [View by thread](#)
- [View by date](#)
- [Next message](#)
- [ANNOUNCE] HTTP/2 vulnerabilities from 2.0 to 2.5-dev Willy Tarreau
- - [Re: \[ANNOUNCE\] HTTP/2 vulnerabilities from 2.0 to 2.5-...](#) Vincent Bernat
 - - [Re: \[ANNOUNCE\] HTTP/2 vulnerabilities from 2.0 to ...](#) Tim Dusterhus
 - [Re: \[ANNOUNCE\] HTTP/2 vulnerabilities from 2.0...](#) Willy Tarreau
 - [Re: \[ANNOUNCE\] HTTP/2 vulnerabilities from 2.0 to 2.5-...](#) Tim Dusterhus
 - [Re: \[ANNOUNCE\] HTTP/2 vulnerabilities from 2.0 to ...](#) Willy Tarreau
 - [Re: \[ANNOUNCE\] HTTP/2 vulnerabilities from 2.0 to 2.5-...](#) James Brown
 - - [Re: \[ANNOUNCE\] HTTP/2 vulnerabilities from 2.0 to ...](#) Lukas Tribus
 - [Re: \[ANNOUNCE\] HTTP/2 vulnerabilities from 2.0 to ...](#) Willy Tarreau


Reply via email to

Willy Tarreau

The Mail Archive



Search the site



- [The Mail Archive home](#)
- [haproxy - all messages](#)
- [haproxy - about the list](#)
- [Expand](#)
- [Previous message](#)
- [Next message](#)



- [The Mail Archive home](#)
- [Add your mailing list](#)
- [FAQ](#)
- [Support](#)

- [Privacy](#)
- 20210817151338.GA27006@lwt.eu