CWE-113: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')

Low **dvlato** published **GHSA-6v7p-v754-j89v** on Feb 10, 2020

Package

No package listed

Affected versions

<=1.0.0.beta8

Patched versions

>= 1.0.0-rc1

Description

Vulnerability

Styx is vulnerable to CWE-113: Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting').

Vulnerable Component

The vulnerable component is the `com.hotels.styx.api.HttpHeaders.Builder` due to disabling the HTTP Header validation built into Netty in these locations:

https://github.com/HotelsDotCom/styx/blob/e1d578e9b9c38df9cd19c21dc2eb9b949d85b558/components/api/src/main/java/com/hotels/styx/api/HttpHeaders.java#L145

https://github.com/HotelsDotCom/styx/blob/e1d578e9b9c38df9cd19c21dc2eb9b949d85b558/components/api/src/main/java/com/hotels/styx/api/HttpHeaders.java#L145

`new DefaultHttpHeaders(false)` disables the built-in validation in Netty. Either use the default constructor or `new DefaultHttpHeaders(true` instead.

Additionally, another vulnerable component is the StyxToNettyResponseTranslator due to also disabling the HTTP Header validation built into netty in this location.

https://github.com/HotelsDotCom/styx/blob/8d60e5493e65d0d536afc0b350dcb02d24e0f7a7/components/server/src/main/java/com/hotels/styx/server/netty/connectors/StyxToNettyResponseTranslator.java#L30

`DefaultHttpResponse nettyResponse = new DefaultHttpResponse(version, httpResponseStatus, false);`
`new DefaultHttpResponse(version, httpResponseStatus, false);` disables the built-in validation in Netty. Please use the constructor `new DefaultHttpResponse(version, httpResponseStatus, true);`

Proof of Concept

The following test plugin proves that there is no header validation occurring.

```
static class VulnerablePlugin implements Plugin {

    @Override
    public Eventual<LiveHttpResponse> intercept(LiveHttpRequest request, Chain chain) {
        String header = request.queryParam("header-value").get();
        LiveHttpRequest newRequest = request.newBuilder()
            .header("myRequestHeader", header)
            .build();
        return chain.proceed(newRequest).map(response ->
            response.newBuilder().header("myResponseHeader", header).build()
        ) ;
    }

}

@Test
public void simpleHeaderInjectionVulnerabilityPOC() {
    Plugin vulnerablePlugin = new VulnerablePlugin();
    // a simple way to mock the downstream system
    HttpInterceptor.Chain chain = request -> {
        assertThat(request.header("myRequestHeader").orElse(null), is("test\r\nAnother: CRLF_Injection"));
        return Eventual.of(response(OK).build());
    };

    // an example request you expect your plugin to receive
    String encodedGet = URLEncoder.encode("test\r\nAnother: CRLF_Injection");
    LiveHttpRequest request = get("/foo?header-value=" + encodedGet)
        .build();

    // since this is a test, we want to wait for the response
    LiveHttpResponse response = Mono.from(vulnerablePlugin.intercept(request, chain)).block();

    assertThat(response.header("myResponseHeader").orElse(null), is("test\r\nAnother: CRLF_Injection"));
}
```

Additionally, if you run this LiveHttpResponse from this test through the `StyxToNettyResponseTranslator::toNettyResponse`, ideally, it would have caused an exception to be thrown. In its current state, it does not.

Similar Vulnerabilities

There have been reports of similar vulnerabilities in other popular libraries.

GHSA-35fr-h7jr-hh86 -> CVE-2019-16771
GHSA-mvqp-q37c-wf9j -> CVE-2019-17513

Finding

This vulnerability was found due to this query that Jonathan Leitschuh contributed to the Semmle QL project.
https://lgtm.com/rules/1510696449842/alerts/

Severity

Low

CVE ID

CVE-2020-6858

Weaknesses

No CWEs

Credits


JLLeitschuh