CPU 100% receiving an invalid large TLS frame

High **waynebeaton** published **GHSA-26vr-8j45-3r4w** on Apr 1, 2021

Package

org.eclipse.jetty:jetty-io (Maven)

Affected versions	Patched versions
7.2.2-9.4.38, 10.0.0.alpha0-10.0.1, 11.0.0.alpha0-11.0.1	9.4.39, 10.0.2, 11.0.2

Description

Impact

When using SSL/TLS with Jetty, either with HTTP/1.1, HTTP/2, or WebSocket, the server may receive an invalid large (greater than 17408) TLS frame that is incorrectly handled, causing CPU resources to eventually reach 100% usage.

Workarounds

The problem can be worked around by compiling the following class:

```java
package org.eclipse.jetty.server.ssl.fix6072;

import java.nio.ByteBuffer;
import javax.net.ssl.SSLEngine;
import javax.net.ssl.SSLEngineResult;
import javax.net.ssl.SSLException;
import javax.net.ssl.SSLHandshakeException;

import org.eclipse.jetty.io.EndPoint;
import org.eclipse.jetty.io.ssl.SslConnection;
import org.eclipse.jetty.server.Connector;
import org.eclipse.jetty.server.SslConnectionFactory;
import org.eclipse.jetty.util.BufferUtil;
import org.eclipse.jetty.util.annotation.Name;
import org.eclipse.jetty.util.ssl.SslContextFactory;

public class SpaceCheckingSslConnectionFactory extends SslConnectionFactory
{
    public SpaceCheckingSslConnectionFactory(@Name("sslContextFactory") SslContextFactory factory, @Name("next") String nextProtocol)
    {
        super(factory, nextProtocol);
    }

    @Override
    protected SslConnection newSslConnection(Connector connector, EndPoint endPoint, SSLEngine engine)
    {
        return new SslConnection(connector.getByteBufferPool(), connector.getExecutor(), endPoint, engine, isDirectBuffersForEncryption(), isDirectBuffersForDecryption())
        {
            @Override
            protected SSLEngineResult unwrap(SSLEngine sslEngine, ByteBuffer input, ByteBuffer output) throws SSLException
            {
                SSLEngineResult results = super.unwrap(sslEngine, input, output);

                if ((results.getStatus() == SSLEngineResult.Status.BUFFER_UNDERFLOW ||
                    results.getStatus() == SSLEngineResult.Status.OK && results.bytesConsumed() == 0 && results.bytesProduced() == 0) &&
                    BufferUtil.space(input) == 0)
                {
                    BufferUtil.clear(input);
                    throw new SSLHandshakeException("Encrypted buffer max length exceeded");
                }
                return results;
            }
        };
    }
}
```

This class can be deployed by:

- The resulting class file should be put into a jar file (eg sslfix6072.jar)
- The jar file should be made available to the server. For a normal distribution this can be done by putting the file into ${jetty.base}/lib
- Copy the file `${jetty.home}/modules/ssl.mod` to `${jetty.base}/modules`
- Edit the `${jetty.base}/modules/ssl.mod` file to have the following section:

```
[lib]
lib/sslfix6072.jar
```

- Copy the file `${jetty.home}/etc/jetty-https.xml` and `${jetty.home}/etc/jetty-http2.xml` to `${jetty.base}/etc`
- Edit files `${jetty.base}/etc/jetty-https.xml` and `${jetty.base}/etc/jetty-http2.xml` , changing any reference of `org.eclipse.jetty.server.SslConnectionFactory` to `org.eclipse.jetty.server.ssl.fix6072.SpaceCheckingSslConnectionFactory` . For example:

```xml
<Call name="addIfAbsentConnectionFactory">
  <Arg>
    <New class="org.eclipse.jetty.server.ssl.fix6072.SpaceCheckingSslConnectionFactory">
      <Arg name="next">http/1.1</Arg>
      <Arg name="sslContextFactory"><Ref refid="sslContextFactory"/></Arg>
    </New>
  </Arg>
</Call>
```

- Restart Jetty

Severity

High 7.5 / 10

CVSS base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	None
Integrity	None
Availability	High

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE ID

CVE-2021-28165

Weaknesses

CWE-400