New issue

Jump to bottom

Uncaught Exception in Parser #60

Closed GanbaruTobi opened this issue on Feb 22, 2021 · 8 comments

Labels duplicate

GanbaruTobi commented on Feb 22, 2021 • edited

The parser fails to throw the ParseException when the parser expects the input to be of the float number type AND the input not being a valid number. This can lead to uncaught exceptions by unexpected input, which may lead to Denial-of-Service (DoS).

json-smart-v2/json-smart/src/main/java/net/minidev/json/parser/JSONParserBase.java
Lines 139 to 147 in 4402bae

```
139     protected Number extractFloat() throws ParseException {
140             if (!acceptLeadinZero)
141                     checkLeadinZero();
142             if (!useHiPrecisionFloat)
143                     return Float.parseFloat(xs);
144             if (xs.length() > 18) // follow JSonIJ parsing method
145                     return new BigDecimal(xs);
146             return Double.parseDouble(xs);
147     }
```

Parser Input of "-." or "2e+" or "[45e-" will crash with a NumberFormatException.

```
== Java Exception: java.lang.NumberFormatException: For input string: "-."
        at java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2054)
        at java.base/jdk.internal.math.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
        at java.base/java.lang.Double.parseDouble(Double.java:549)
        at net.minidev.json.parser.JSONParserBase.extractFloat(JSONParserBase.java:141)
```

👍 3

This was referenced on Feb 23, 2021

Findings CodeIntelligenceTesting/jazzer#19

Closed

Possible fix for Exception handling #61

Merged

pronovic commented on Mar 12, 2021

Note that this is tied to CVE-2021-27568, categorized as base score 9.1 (critical).

UrielCh added duplicate in progress and removed in progress labels on Apr 1, 2021

UrielCh commented on Apr 4, 2021 Contributor

CVE-2021-27568 is now fully fixed in

- json-smart(v2) for java 1.8 +
- json-smart(v1) for java 1.6 +
- json-smart-mini for java 1.6 +

UrielCh closed this as completed on Apr 4, 2021

ounsworth commented on Apr 8, 2021

Can someone explain why this was given a base score of 9.1?
That seems excessively alarmist given that the invalid input is detected and an exception thrown ...

👍 1

UrielCh commented on Apr 9, 2021 Contributor

I think that the issue was more about the vanilla java exception, I think it can disclose the java virtual machine type.

I do not know how they measure the score.

GanbaruTobi commented on Apr 9, 2021 Author

The score was created on MITREs side. I don't know how they calculated it.

ounsworth commented on Apr 10, 2021 • edited

Ok, I dug a bit. MITRE rated it like this:

> CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:H
> Attack Vector: None, Attack Complexity: Low, Privileges Required: None, User Interaction: None, Scope: Unchanged, Confidentiality: High, Integrity: None, Availability: High.

I would have scored it the same except for Confidentiality: High. I guess that's because of the potential for a stack trace getting returned to the end user? But this bug does not guarantee that a stack trace is returned to the user, for that the library would have to be used by an application returns unexpected stack traces to the user in a 500 body, which well-behaved apps should not be doing anyways. If that understanding is correct, I would have called this Confidentiality: Low or None, which would have given a Base Score of 8.2 or 7.5 respectively -- which are out of the Critical range.

There may also be wiggle-room on Availability: High. This bug does not guarantee that the application crashes; that would only be the case if the calling application does not have a generic `catch (Exception e)` anywhere in the call stack and this NumberFormatException gets all the way back to the JRE, which again is probably uncommon in well-behaved apps that parse user input. Availability: Low might be a better rating?

I'm making a fuss because 9.0 - 10.0 is Critical, and having those show up can trigger emergency patching policies or delays to release schedules. Is it possible to ask MITRE to re-evaluate the scoring of this issue?

GanbaruTobi commented on Apr 10, 2021 Author

Yes it is, but so far I didn't receive any answer to any comment I gave
them. So in reality, I wouldn't expect anything to happen soon.

Mike Ounsworth ***@***.***> schrieb am Sa., 10. Apr. 2021,
16:21:
...

pcy190 mentioned this issue on Apr 23, 2021

ArrayIndexOutOfBoundsException in parser #67

Closed

dpeger mentioned this issue on Apr 23, 2021

[#60][#62] Unchecked Exception in Parser #72

Merged

dpeger commented on Apr 26, 2021 • edited Contributor

@GanbaruTobi out of curiosity: Would you consider this issue fixed if `NumberFormatException` was declared in the throws-clause of `extractFloat()`:

```
protected Number extractFloat() throws ParseException, NumberFormatException {
   ...
}
```

And the corresponding public methods that use `extractFloat()`.

UrielCh added a commit that referenced this issue on Apr 30, 2021

`Merge pull request #72 from dpeger/fixes/v2.3/parse-numberformatexcep…` ··· `7304d1e`

StephenWikeTBCT mentioned this issue on May 5, 2021

jFrog Xray detects critical violation for json-smart flyway/flyway-docker#53

Closed

This was referenced on May 16, 2021

Denial of Service (DoS) SNYK-JAVA-NETMINIDEV-1078499 RADAR-base/radar-output-restructure#193

Closed

Denial of Service (DoS) SNYK-JAVA-NETMINIDEV-1078499 RADAR-base/radar-output-restructure#194

Closed

Denial of Service (DoS) SNYK-JAVA-NETMINIDEV-1078499 RADAR-base/radar-output-restructure#195

Closed

Assignees

No one assigned

Labels

duplicate

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

5 participants