Commit eaedaf5

Browse files

gorhill committed 3 weeks ago · 3 / 3 · Verified

Fix regexes with potential catastrophic backtracking

The quoted email below was sent to ubo-security at raymondhill dot net:

=====
Dear Raymond,

I am writing to report a potential Regular Expression Denial of Service (ReDoS) vulnerability in the 1p-filters.js script of uBlock Origin. The vulnerability occurs due to the use of the regular expression /\s+$/, which is used to remove trailing whitespace. This issue can lead to a denial of service when processing strings with a large number of trailing spaces, potentially causing a browser to freeze.

Affected file(s)

    js/1p-filters.js

Vulnerable pattern(s)

    Lines 131 and 167: /\s+$/

Description of the issue

The regular expression /\s+$/ is applied to remove trailing whitespace in user-provided content. However, when the content has a large number of spaces (e.g., ~100,000), this pattern causes excessive backtracking in the regular expression engine, resulting in performance degradation and UI freezing. This is a classic ReDoS attack vector.

Steps to reproduce

1. Open the uBlock Origin dashboard and navigate to the My filters tab.
2. Run the following code in the browser's DevTools Console or as a bookmarklet.
3. Observe the UI freezing for several seconds or even longer, depending on the number of spaces used.

PoC (Proof of Concept)

```
/**
 * poc.js — triggers ReDoS in 1p-filters.js
 * Expected: <1 ms; Actual: several seconds – UI freeze
 */
(() => {
  const payload = " ".repeat(100000) + "!";  // 100,000 spaces + sentinel
  const run = () => {
    if (!window.cmEditor) {
      console.error("cmEditor not ready");
      return;
```

```
    }
    // Inject payload into the editor
    cmEditor.setValue(payload);

    console.time("ReDoS");
    // Call the vulnerable function (mirroring getEditorText)
    cmEditor.getValue().replace(/\s+$/, '');
    // Alternatively, simulate a realistic user flow:
    // document.querySelector('#userFiltersApply').click();
    console.timeEnd("ReDoS");
  };

  if (document.readyState === "complete") {
    run();
  } else {
    window.addEventListener("load", run, { once: true });
  }
})();

Impact

This issue can significantly degrade the user experience, causing the page to
become unresponsive. If an attacker can inject this malicious string into the
page (for example, through XSS or other attacks), it could lead to a denial of
service (DoS). This vulnerability can be triggered repeatedly, causing the
browser to hang indefinitely.

Suggested fix

The issue can be mitigated by replacing /\s+$/ with a more efficient solution,
such as a look-behind assertion /(?<=\S)\s+$/ (available in modern browsers)
which ensures no backtracking occurs, or using trimEnd() for legacy support:

// Example of using look-behind:
cmEditor.setValue(text.replace(/(?<=\S)\s+$/, '') + '\n\n');

// Alternatively, using trimEnd():
cmEditor.setValue(text.trimEnd() + '\n\n');

Additional information

If required, I am happy to assist in testing or provide more information.
Please feel free to contact me for further clarification.

Best regards,
[redacted]
=====
```

master · 1.63.3rc2 ••• 1.63.3b17 1 parent 87007e6 commit eaedaf5

Filter files...

src/js

1p-filters.js

arglist-parser.js

static-filtering-parser.js

whitelist.is

4 files changed +9 -9 lines changed Top Search within code

src/js/1p-filters.js +2 -2

```
@@ -111,12 +111,12 @@ function currentStateChanged() {
111 111     }
112 112
113 113     function getEditorText() {
114     -       const text = cmEditor.getValue().replace(/\s+$/, '');
    114 +       const text = cmEditor.getValue().trimEnd();
115 115         return text === '' ? text : `${text}\n`;
116 116     }
117 117
118 118     function setEditorText(text) {
119     -       cmEditor.setValue(text.replace(/\s+$/, '') + '\n\n');
    119 +       cmEditor.setValue(`${text.trimEnd()}\n\n`);
120 120     }
121 121
122 122     /******************************************************************************/
```

src/js/arglist-parser.js +2 -2

```
@@ -32,7 +32,7 @@ export class ArglistParser {
32 32              this.transform = false;
33 33              this.failed = false;
34 34              this.reWhitespaceStart = /^\s+/;
35     -            this.reWhitespaceEnd = /\s+$/;
   35 +            this.reWhitespaceEnd = /(?:^|\S)(\s+)$/;
36 36              this.reOddTrailingEscape = /(?:^|[^\\])(?:\\\\)*\\$/;
37 37              this.reTrailingEscapeChars = /\\+$/;
38 38          }
@@ -90,7 +90,7 @@ export class ArglistParser {
90 90          }
91 91          rightWhitespaceCount(s) {
92 92              const match = this.reWhitespaceEnd.exec(s);
93     -            return match === null ? 0 : match[0].length;
   93 +            return match === null ? 0 : match[1].length;
94 94          }
95 95          indexOfNextArgSeparator(pattern, separatorCode) {
96 96              this.argBeg = this.argEnd = separatorCode !== this.separatorCode
```

src/js/static-filtering-parser.js +2 -2

```
@@ -785,7 +785,7 @@ export class AstFilterParser {
785 785             this.selectorCompiler = new ExtSelectorCompiler(options);
786 786             // Regexes
787 787             this.reWhitespaceStart = /^\s+/;
788     -           this.reWhitespaceEnd = /\s+$/;
    788 +           this.reWhitespaceEnd = /(?:^|\S)(\s+)$/;
789 789             this.reCommentLine = /^(?:!|#\s|####|\[adblock)/i;
790 790             this.reExtAnchor = /(#@?(?:\$\?|\$|%|\?)?#).{1,2}/;
791 791             this.reInlineComment = /(?:\s+#).*?$/;
@@ -2786,7 +2786,7 @@ export class AstFilterParser {
2786 2786
2787 2787         rightWhitespaceCount(s) {
2788 2788             const match = this.reWhitespaceEnd.exec(s);
2789     -           return match === null ? 0 : match[0].length;
    2789 +           return match === null ? 0 : match[1].length;
2790 2790         }
2791 2791
2792 2792         nextCommaInCommaSeparatedListString(s, start) {
```

src/js/whitelist.js +3 -3

```
@@ -99,12 +99,12 @@ uBlockDashboard.patchCodeMirrorEditor(cmEditor);
99  99  /******************************************************************************/
100 100
101 101 function getEditorText() {
102     -   let text = cmEditor.getValue().replace(/\s+$/, '');
103     -   return text === '' ? text : text + '\n';
    102 +   const text = cmEditor.getValue().trimEnd();
    103 +   return text === '' ? text : `${text}\n`;
104 104 }
105 105
106 106 function setEditorText(text) {
107     -   cmEditor.setValue(text.replace(/\s+$/, '') + '\n');
    107 +   cmEditor.setValue(`${text.trimEnd()}\n`);
108 108 }
109 109
110 110 /******************************************************************************/
```

Comments 0

Please sign in to comment.