

projectdiscovery / nuclei Public[Code](#) [Issues](#) 102 [Pull requests](#) 31 [Discussions](#) [Actions](#) [Projects](#)Commit **d221732** **dwiswant0** authored on Mar 16 · ✖ 1 / 11 · Verified`fix(expressions): only eval template-authored expressions (#7221)``* fix(expressions): only eval template-authored expressions`

```
`expressions.Evaluate()` currently replaces  
placeholders first and then scans the substituted  
output for expressions which allows  
response-derived values, including extractor  
output, to be reinterpreted as DSL/helper syntax  
on a second pass, which then becomes more serious  
when env-backed variables are enabled.
```

So making this by collecting expressions from the original template text before placeholder substitution and then evaluating only those original expressions after normal replacement to keep trusted template-authored helpers working, including placeholders nested inside helper arguments while treating substituted response data as literal text.

Fixes [#7210](#)

```
Signed-off-by: Dwi Siswanto <git@dw1.io>
```

```
* test: add resp data literal reuse test
```

```
Signed-off-by: Dwi Siswanto <git@dw1.io>
```

```
-----
```

```
Signed-off-by: Dwi Siswanto <git@dw1.io>
```

[dev](#) (#7221) · [v3.8.0](#)

1 parent [b099852](#) commit [d221732](#)

4 files changed **+105 -1** lines changed

[↑ Top](#)



- v cmd/integration-test
 - http.go
- v integration_tests/protocols/http
 - response-data-literal-reuse.yaml
- v pkg/protocols/common/expressions
 - expressions.go
 - expressions_test.go

4 files changed +105 -1 lines changed


 v cmd/integration-test/http.go ...

```

@@ -88,6 +88,7 @@ var httpTestcases = []TestCaseInfo{
88 88     {Path: "protocols/http/multi-request.yaml", TestCase: &httpMultiRequest{}},
89 89     {Path: "protocols/http/http-matcher-extractor-dy-extractor.yaml", TestCase:
    &httpMatcherExtractorDynamicExtractor{}},
90 90     {Path: "protocols/http/multi-http-var-sharing.yaml", TestCase:
    &httpMultiVarSharing{}},
91 +     {Path: "protocols/http/response-data-literal-reuse.yaml", TestCase:
    &httpResponseDataLiteralReuse{}},
91 92     {Path: "protocols/http/raw-path-single-slash.yaml", TestCase:
    &httpRawPathSingleSlash{}},
92 93     {Path: "protocols/http/raw-unsafe-path-single-slash.yaml", TestCase:
    &httpRawUnsafePathSingleSlash{}},
93 94 }
@@ -102,6 +103,31 @@ func (h *httpMultiVarSharing) Execute(filePath string)
error {
102 103     return expectResultsCount(results, 1)
103 104 }
104 105
106 + type httpResponseDataLiteralReuse struct{
107 +
108 + func (h *httpResponseDataLiteralReuse) Execute(filePath string) error {
109 +     router := httprouter.New()
110 +     router.GET("/", func(w http.ResponseWriter, r *http.Request, _
    httprouter.Params) {

```

```
111 +     _, _ = fmt.Fprint(w, `{{md5("Hello")}}`)
112 + })
113 + router.GET("/echo", func(w http.ResponseWriter, r *http.Request, _
    httprouter.Params) {
114 +     if r.URL.Query().Get("x") != `{{md5("Hello")}}` {
115 +         w.WriteHeader(http.StatusBadRequest)
116 +         return
117 +     }
118 +     w.WriteHeader(http.StatusOK)
119 + })
120 + ts := httptest.NewServer(router)
121 + defer ts.Close()
122 +
123 + results, err := testutils.RunNucleiTemplateAndGetResults(filePath, ts.URL,
    debug)
124 + if err != nil {
125 +     return err
126 + }
127 +
128 + return expectResultsCount(results, 1)
129 + }
```

```
105 131 type httpMatcherExtractorDynamicExtractor struct{}
```

```
106 132
```

```
107 133 func (h *httpMatcherExtractorDynamicExtractor) Execute(filePath string) error {
```



▼ ...ocols/http/response-data-literal-reuse.yaml



```
... @@ -0,0 +1,34 @@
1 + id: response-data-literal-reuse
2 +
3 + info:
4 +   name: Response data literal reuse
5 +   author: dwisiswant0
6 +   severity: info
7 +   description: |
8 +     Make sure response-derived {{...}} content stays literal when reused in a
9 +     later request.
10 +   tags: test,http
11 +
```

```

12 + http:
13 +   - raw:
14 +     - |
15 +       GET / HTTP/1.1
16 +       Host: {{Hostname}}
17 +
18 +     extractors:
19 +       - type: regex
20 +       name: extracted_body
21 +       part: body
22 +       regex:
23 +         - '(?s)(.*)'
24 +       internal: true
25 +
26 +   - raw:
27 +     - |
28 +       GET /echo?x={{extracted_body}} HTTP/1.1
29 +       Host: {{Hostname}}
30 +
31 +     matchers:
32 +       - type: status
33 +       status:
34 +         - 200

```

...protocols/common/expressions/expressions.go

...

↑

```

@@ -43,14 +43,15 @@ func EvaluateByte(data []byte, base
map[string]interface{}) ([]byte, error) {

```

```

43 43   }
44 44
45 45   func evaluate(data string, base map[string]interface{}) (string, error) {
46 +     expressions := FindExpressions(data, marker.ParenthesisOpen,
47 +     marker.ParenthesisClose, base)
47 +
46 48     // replace simple placeholders (key => value) MarkerOpen + key + MarkerClose
47 49     and General + key + General to value
48 50     data = replacer.Replace(data, base)
49 51     // expressions can be:
50 52     // - simple: containing base values keys (variables)
51 53     // - complex: containing helper functions [ + variables]

```

```

52  54      // literals like {{2+2}} are not considered expressions
53  -      expressions := FindExpressions(data, marker.ParenthesisOpen,
      marker.ParenthesisClose, base)
54  55      for _, expression := range expressions {
55  56          // replace variable placeholders with base values
56  57          expression = replacer.Replace(expression, base)

```



...cols/common/expressions/expressions_test.go



```

@@ -62,3 +62,46 @@ func TestEval(t *testing.T) {
62  62          require.Equal(t, item.expected, value, "could not get correct
      expression")
63  63      }
64  64  }
65  +
66  + func TestEvaluateDoesNotReinterpretResolvedValues(t *testing.T) {
67  +     items := []struct {
68  +         name      string
69  +         input      string
70  +         expected   string
71  +         extra      map[string]interface{}
72  +     }{
73  +         {
74  +             name:      "helper syntax in resolved values stays literal",
75  +             input:      "?x={{body}}",
76  +             expected: `?x={{md5("Hello")}}-by-Adelle`,
77  +             extra: map[string]interface{}{
78  +                 "body": `{{md5("Hello")}}-by-Adelle`,
79  +             },
80  +         },
81  +         {
82  +             name:      "resolved values cannot access other variables",
83  +             input:      "Authorization: {{body}}",
84  +             expected: "Authorization: {{secret_token}}",
85  +             extra: map[string]interface{}{
86  +                 "body":      "{{secret_token}}",
87  +                 "secret_token": "top-secret-cia-mi6-kgb-mossad-classified",
88  +             },
89  +         },
90  +     }

```

```
91 +         name:      "template-authored placeholders inside helper expressions
    still resolve",
92 +         input:    "{{base64('{{Host}}')}}",
93 +         expected: "MTI3LjAuMC4x",
94 +         extra: map[string]interface{}{
95 +             "Host": "127.0.0.1",
96 +         },
97 +     },
98 + }
99 +
100 + for _, item := range items {
101 +     t.Run(item.name, func(t *testing.T) {
102 +         value, err := Evaluate(item.input, item.extra)
103 +         require.NoError(t, err)
104 +         require.Equal(t, item.expected, value)
105 +     })
106 + }
107 + }
```

Comments 0



Please [sign in](#) to comment.