tektoncd / **pipeline** Public

Code | Issues 431 | Pull requests 91 | Discussions | Actions | Projects

Commit b890560

vdemeester committed 3 hours ago Verified

```
fix: strip resolver prefixes and use non-capturing group for pattern anchoring

The previous commit naively prepended ^ and appended $ to patterns,
which broke backward compatibility with the git resolver's SPDX
"git+" URI prefix.  Sources like "git+https://github.com/…" no
longer matched patterns written as "https://github.com/…".

Fix the approach:
- Strip known resolver prefixes ("git+") from the source URI before
  matching, so patterns don't need to account for resolver-specific
  URI schemes.
- Wrap unanchored patterns in ^(?:…)$ (non-capturing group) instead
  of naively prepending/appending anchors, which correctly handles
  patterns that are only partially anchored.
- Preserve already fully-anchored patterns (^…$) unchanged.

Replace the single bypass-control test with comprehensive test
coverage: bypass rejection, prefix stripping, already-anchored
patterns, anchorPattern() unit tests, and stripResolverPrefix() unit
tests.

Fixes: GHSA-rmx9-2pp3-xhcr
```

main

1 parent c8cc73a commit b890560

2 files changed +228 -1 lines changed

Top

Filter files…

- pkg/trustedresources
  - verify.go
  - verify_regex_bypass_control_test.go

2 files changed +228 -1 lines changed

Search within code

pkg/trustedresources/verify.go

```
@@ -23,6 +23,7 @@ import (
23 23      "errors"
24 24      "fmt"
25 25      "regexp"
   26 +    "strings"
26 27
27 28      "github.com/sigstore/sigstore/pkg/signature"
28 29      "github.com/tektoncd/pipeline/pkg/apis/config"
@@ -117,9 +118,14 @@ func VerifyPipeline(ctx context.Context, pipelineObj
    *v1beta1.Pipeline, k8s kube
117 118 // getMatchedPolicies filters out the policies by checking if the resource url
    (source) is matching any of the `patterns` in the `resources` list.
118 119 func getMatchedPolicies(resourceName string, source string, policies
    []*v1alpha1.VerificationPolicy) ([]*v1alpha1.VerificationPolicy, error) {
119 120     matchedPolicies := []*v1alpha1.VerificationPolicy{}
   121 +    // Strip known resolver prefixes before matching so that patterns
   122 +    // written without the prefix (e.g. "https://github.com/…") still
   123 +    // work after we anchor them for full-string matching.
   124 +    normalizedSource := stripResolverPrefix(source)
120 125     for _, p := range policies {
121 126         for _, r := range p.Spec.Resources {
122     -           matching, err := regexp.MatchString(r.Pattern, source)
   127 +           pattern := anchorPattern(r.Pattern)
   128 +           matching, err := regexp.MatchString(pattern, normalizedSource)
123 129             if err != nil {
124 130                 // FixMe: changing %v to %w breaks integration tests.
125 131                 return matchedPolicies, fmt.Errorf("%v: %w", err,
    ErrRegexMatch) //nolint:errorlint
@@ -136,6 +142,42 @@ func getMatchedPolicies(resourceName string, source
    string, policies []*v1alpha1
136 142     return matchedPolicies, nil
137 143 }
138 144
   145 + // anchorPattern wraps a pattern with ^(?:…)$ when it is not already
   146 + // anchored.  This forces full-string matching so that unanchored
   147 + // patterns cannot be bypassed by embedding the trusted URL as a
```

```
148 + // substring of a malicious source URI.
149 + func anchorPattern(pattern string) string {
150 +     hasStart := strings.HasPrefix(pattern, "^")
151 +     hasEnd := strings.HasSuffix(pattern, "$")
152 +     if hasStart && hasEnd {
153 +         return pattern
154 +     }
155 +     if !hasStart {
156 +         pattern = "^(?:" + pattern
157 +     } else {
158 +         // Already has ^, wrap the rest in a non-capturing group.
159 +         pattern = "^(?:" + pattern[1:]
160 +     }
161 +     if !hasEnd {
162 +         pattern = pattern + ")$"
163 +     } else {
164 +         // Already has $, close the group before it.
165 +         pattern = pattern[:len(pattern)-1] + ")$"
166 +     }
167 +     return pattern
168 + }
169 +
170 + // stripResolverPrefix removes known URI scheme prefixes added by
171 + // resolvers before policy matching.  The git resolver prepends "git+"
172 + // per the SPDX convention (see spdxGit in
173 + // pkg/resolution/resolver/git/resolver.go).
174 + func stripResolverPrefix(uri string) string {
175 +     if strings.HasPrefix(uri, "git+") {
176 +         return uri[4:]
177 +     }
178 +     return uri
179 + }
180 +
139 181   // verifyResource verifies resource which implements metav1.Object by provided
          signature and public keys from verification policies.
140 182   // For matched policies, `verifyResource“ will adopt the following rules to do
          verification:
141 183   //  1. If multiple policies match, the resource must satisfy all the "enforce"
          policies to pass verification. The matching "enforce" policies are evaluated
          using AND logic.
```

...sources/verify_regex_bypass_control_test.go

```
@@ -0,0 +1,185 @@
+ package trustedresources
+
+ import (
+     "errors"
+     "testing"
+
+     "github.com/tektoncd/pipeline/pkg/apis/pipeline/v1alpha1"
+ )
+
+ func TestGetMatchedPolicies_RegexBypass(t *testing.T) {
+     policy := &v1alpha1.VerificationPolicy{
+         Spec: v1alpha1.VerificationPolicySpec{
+             Resources: []v1alpha1.ResourcePattern{{
+                 Pattern: "https://github.com/tektoncd/catalog.git",
+             }},
+         },
+     }
+     policies := []*v1alpha1.VerificationPolicy{policy}
+
+     tests := []struct {
+         name        string
+         source      string
+         wantMatch   bool
+         description string
+     }{{
+         name:        "malicious source with trusted URL as query param is rejected",
+         source:      "https://evil.com/?x=https://github.com/tektoncd/catalog.git",
+         wantMatch:   false,
+         description: "substring bypass via query parameter",
+     }, {
+         name:        "malicious source with trusted URL as path component is rejected",
+         source:      "https://evil.com/https://github.com/tektoncd/catalog.git/foo",
```

```
33  +         wantMatch:   false,
34  +         description: "substring bypass via path embedding",
35  +     }, {
36  +         name:        "exact match without prefix succeeds",
37  +         source:      "https://github.com/tektoncd/catalog.git",
38  +         wantMatch:   true,
39  +         description: "exact source matches unanchored pattern",
40  +     }, {
41  +         name:        "git+ prefixed source matches after prefix stripping",
42  +         source:      "git+https://github.com/tektoncd/catalog.git",
43  +         wantMatch:   true,
44  +         description: "git resolver spdx prefix is stripped before matching",
45  +     }}
46  +
47  +     for _, tc := range tests {
48  +         t.Run(tc.name, func(t *testing.T) {
49  +             matched, err := getMatchedPolicies("test-resource", tc.source,
      policies)
50  +             if tc.wantMatch {
51  +                 if err != nil {
52  +                     t.Fatalf("expected match for source %q, got err: %v",
      tc.source, err)
53  +                 }
54  +                 if len(matched) != 1 {
55  +                     t.Fatalf("expected 1 matched policy, got %d", len(matched))
56  +                 }
57  +             } else {
58  +                 if err == nil {
59  +                     t.Fatalf("expected no match for source %q, but got a
      match", tc.source)
60  +                 }
61  +                 if !errors.Is(err, ErrNoMatchedPolicies) {
62  +                     t.Fatalf("expected ErrNoMatchedPolicies, got: %v", err)
63  +                 }
64  +             }
65  +         })
66  +     }
67  + }
68  +
69  + func TestGetMatchedPolicies_AlreadyAnchoredPatterns(t *testing.T) {
```

```
70  +     tests := []struct {
71  +         name      string
72  +         pattern   string
73  +         source    string
74  +         wantMatch bool
75  +     }{{
76  +         name:      "fully anchored pattern still works",
77  +         pattern:   "^https://github\\.com/tektoncd/catalog\\.git$",
78  +         source:    "https://github.com/tektoncd/catalog.git",
79  +         wantMatch: true,
80  +     }, {
81  +         name:      "start-anchored pattern gets end-anchored",
82  +         pattern:   "^https://github.com/tektoncd/.*",
83  +         source:    "https://github.com/tektoncd/catalog.git",
84  +         wantMatch: true,
85  +     }, {
86  +         name:      "end-anchored pattern gets start-anchored",
87  +         pattern:   ".*catalog.git$",
88  +         source:    "https://github.com/tektoncd/catalog.git",
89  +         wantMatch: true,
90  +     }, {
91  +         name:      "wildcard pattern matches everything",
92  +         pattern:   ".*",
93  +         source:    "https://anything.example.com",
94  +         wantMatch: true,
95  +     }, {
96  +         name:      "git+ prefix stripped before matching anchored pattern",
97  +         pattern:   "^https://github\\.com/tektoncd/catalog\\.git$",
98  +         source:    "git+https://github.com/tektoncd/catalog.git",
99  +         wantMatch: true,
100 +     }}
101 +
102 +     for _, tc := range tests {
103 +         t.Run(tc.name, func(t *testing.T) {
104 +             policy := &v1alpha1.VerificationPolicy{
105 +                 Spec: v1alpha1.VerificationPolicySpec{
106 +                     Resources: []v1alpha1.ResourcePattern{{
107 +                         Pattern: tc.pattern,
108 +                     }},
109 +                 },
```

```
110 +            }
111 +            matched, err := getMatchedPolicies("test-resource", tc.source,
    []*v1alpha1.VerificationPolicy{policy})
112 +            if tc.wantMatch {
113 +                if err != nil {
114 +                    t.Fatalf("expected match for pattern %q against source %q,
    got err: %v", tc.pattern, tc.source, err)
115 +                }
116 +                if len(matched) != 1 {
117 +                    t.Fatalf("expected 1 matched policy, got %d", len(matched))
118 +                }
119 +            } else {
120 +                if err == nil {
121 +                    t.Fatalf("expected no match for pattern %q against source
    %q, but got a match", tc.pattern, tc.source)
122 +                }
123 +            }
124 +        })
125 +    }
126 + }
127 +
128 + func TestAnchorPattern(t *testing.T) {
129 +     tests := []struct {
130 +         input string
131 +         want  string
132 +     }{{
133 +         input: "https://github.com/tektoncd/catalog.git",
134 +         want:  "^(?:https://github.com/tektoncd/catalog.git)$",
135 +     }, {
136 +         input: "^https://github.com/tektoncd/.*",
137 +         want:  "^(?:https://github.com/tektoncd/.*)$",
138 +     }, {
139 +         input: ".*catalog.git$",
140 +         want:  "^(?:.*catalog.git)$",
141 +     }, {
142 +         input: "^https://github\\.com/tektoncd/catalog\\.git$",
143 +         want:  "^https://github\\.com/tektoncd/catalog\\.git$",
144 +     }, {
145 +         input: ".*",
146 +         want:  "^(?:.*)$",
```

```
147 +     }}
148 +
149 +     for _, tc := range tests {
150 +         t.Run(tc.input, func(t *testing.T) {
151 +             got := anchorPattern(tc.input)
152 +             if got != tc.want {
153 +                 t.Errorf("anchorPattern(%q) = %q, want %q", tc.input, got,
      tc.want)
154 +             }
155 +         })
156 +     }
157 + }
158 +
159 + func TestStripResolverPrefix(t *testing.T) {
160 +     tests := []struct {
161 +         input string
162 +         want  string
163 +     }{{
164 +         input: "git+https://github.com/tektoncd/catalog.git",
165 +         want:  "https://github.com/tektoncd/catalog.git",
166 +     }, {
167 +         input: "https://github.com/tektoncd/catalog.git",
168 +         want:  "https://github.com/tektoncd/catalog.git",
169 +     }, {
170 +         input: "gcr.io/tekton-releases/catalog/upstream/git-clone",
171 +         want:  "gcr.io/tekton-releases/catalog/upstream/git-clone",
172 +     }, {
173 +         input: "",
174 +         want:  "",
175 +     }}
176 +
177 +     for _, tc := range tests {
178 +         t.Run(tc.input, func(t *testing.T) {
179 +             got := stripResolverPrefix(tc.input)
180 +             if got != tc.want {
181 +                 t.Errorf("stripResolverPrefix(%q) = %q, want %q", tc.input,
      got, tc.want)
182 +             }
183 +         })
184 +     }
```

```
185 + }
```

Comments 0

Please sign in to comment.