Commit 0b62633

adambkaplan committed on Sep 5, 2024

```
CVE-2024-7387: Prevent Build Inputs "Zip-Slip"

BuildConfig-driven builds allow developers to specify additional
`Secrets` and/or `ConfigMaps` whose file contents can be referenced
during the build. For `Docker` strategy builds, this is accoplished
by copying the file contents of the `Secret` or `ConfigMap` directly
to a path within the build's source code using the `cp` Linux command.

Previously, an attacker could maliciously craft their source code and
build `Secrets` or `ConfigMaps` in such a way that they could overwrite
the `cp` command and execute arbitrary code. Due to the way build pods
are constructed, these commands then run as root in a privileged
container, with full Linux capabilies and `unconfined` Seccomp policy.
An attacker can then escalate their privileges through multiple attack
vectors - for example, by obtaining the credentials of the host node's
kubelet.

This change blocks the primary attack vector by requiring the
ultimate destination of the referenced `Secret` or `ConfigMap` to be a
child directory of the source code root, thereby preventing the
overwrite of the `cp` Linux command. This only applies to builds which
use the `Docker` strategy during execution. Builds which use the
`Source` or `Custom` build strategy are not affected.

This patch also uses methods that may be considered out of date or
deprecated in golang 1.22 or later. This is done deliberately to allow
clean cherrypicks to OpenShift 3.11 and other, older versions of
OpenShift 4.x.

Signed-off-by: Adam Kaplan <adam.kaplan@redhat.com>
```

main

1 parent 739f527 commit 0b62633

6 files changed +245 -2

Top

Filter files…

- pkg/build/builder
  - docker.go
  - docker_test.go
  - testdata
    - fakesecret
      - pwned.txt
      - sourceSecret.txt
    - fakesource
      - Dockerfile
      - hello.txt

Search within code

pkg/build/builder/docker.go

```
@@ -238,14 +238,33 @@ func (d *DockerBuilder) copySecrets(secrets []buildapiv1.SecretBuildSource, targ
238 238     }
239 239
240 240     func (d *DockerBuilder) copyLocalObject(s localObjectBuildSource, sourceDir, targetDir string) error {
241     +     if !filepath.IsAbs(sourceDir) {
242     +         return fmt.Errorf("cannot copy local object - source directory %q must be an absolute path", sourceDir)
243     +
244     +     }
245     +     if !filepath.IsAbs(targetDir) {
246     +         return fmt.Errorf("cannot copy local object - target directory %q must be an absolute path", targetDir)
247     +     }
241 248         dstDir := filepath.Join(targetDir, s.DestinationPath())
242 249         if err := os.MkdirAll(dstDir, 0777); err != nil {
243 250             return err
244 251         }
```

```
252 +     // Evaluate symlinks at the destination dir. EvalSymlinks calls filepath.Clean, ensuring the
253 +     // returned path is an absolute path (we checked targetDir and thus dstDir is absolute above).
254 +     dstDir, err := filepath.EvalSymlinks(dstDir)
255 +     if err != nil {
256 +         return err
257 +     }
258 +     // sourceDir and targetDir should always be absolute paths, therefore HasPrefix can verify if
259 +     // dstDir is a subdirectory of targetDir.
260 +     if !strings.HasPrefix(dstDir, targetDir) {
261 +         return fmt.Errorf("destination path %q is outside of the target directory %q", dstDir, targetDir)
262 +     }
263 +
245 264       log.V(3).Infof("Copying files from the build source %q to %q", s.LocalObjectRef().Name, dstDir)
246 265
247     -     // Build sources contain nested directories and fairly baroque links. To prevent extra data being
248     -     // copied, perform the following steps:
    266 +     // Build sources from Secrets or ConfigMaps contain nested directories and fairly baroque links.
    267 +     // To prevent extra data being copied, perform the following steps:
249 268       //
250 269       // 1. Only top level files and directories within the secret directory are candidates
251 270       // 2. Any item starting with '..' is ignored
```

pkg/build/builder/docker_test.go

```
@@ -4,6 +4,7 @@ import (
4   4       "bytes"
5   5       "context"
6   6       "fmt"
    7   +     "io"
7   8       "io/ioutil"
8   9       "os"
9   10      "path/filepath"
```

```
@@ -594,3 +595,215 @@ USER 1001`
594 595             }
595 596         }
596 597     }
    598 +
    599 + // TestCopyLocalObject verifies that we are able to copy mounted Kubernetes Secret or ConfigMap
    600 + // data to the build directory. The build directory is typically where git source code is cloned,
    601 + // though other sources of code may be used as well.
    602 + func TestCopyLocalObject(t *testing.T) {
    603 +     // Test scenario consists of a file tree that simulates the git clone environment
    604 +     // /tmp/source-code-xxxx
    605 +     // ├── git-data
    606 +     // │   ├── Dockerfile
    607 +     // │   └── hello.txt
    608 +     // │   └── linkSrc
    609 +     // │       └── link -> ../linkDst
    610 +     // │   └── linkDst
    611 +     // ├── secrets
    612 +     // │   └── test
    613 +     // │       └── secret.txt
    614 +     // TODO: Update the structure of the `secrets` directory so it more closely matches what
    615 +     // Kubernetes does when it mounts secrets/configmaps.
    616 +     buildDir, err := os.MkdirTemp("", "source-code")
    617 +     t.Logf("buildDir: %s", buildDir)
    618 +     if err != nil {
    619 +         t.Fatalf("failed to create temp dir: %v", err)
    620 +     }
    621 +     defer func() {
    622 +         if err := os.RemoveAll(buildDir); err != nil {
    623 +             t.Fatalf("failed to clean up temp dir: %v", err)
    624 +         }
    625 +     }()
    626 +     // Set up secrets/test directory, with "secret.txt" file
    627 +     secretRoot := filepath.Join(buildDir, "secrets")
    628 +     testSecretFile := filepath.Join(secretRoot, "test", "sourceSecret.txt")
```

```
629 +     err = fsCopyFile("testdata/fakesecret/sourceSecret.txt", testSecretFile)
630 +     if err != nil {
631 +         t.Fatalf("failed to copy secret data: %v", err)
632 +     }
633 +     if _, err := os.Stat(testSecretFile); err != nil {
634 +         t.Fatalf("failed to stat %q: %v", testSecretFile, err)
635 +     }
636 +     // Set up "git-data" directory as fake "git source"
637 +     gitRoot := filepath.Join(buildDir, "git-data")
638 +
639 +     err = filepath.Walk("testdata/fakesource", func(path string, info
      os.FileInfo, err error) error {
640 +         if err != nil {
641 +             return err
642 +         }
643 +         if info.IsDir() {
644 +             return nil
645 +         }
646 +         dstBase := filepath.Base(path)
647 +         return fsCopyFile(path, filepath.Join(gitRoot, dstBase))
648 +     })
649 +     if err != nil {
650 +         t.Fatalf("failed to copy test data: %v", err)
651 +     }
652 +     // Git source may contain symlinks. This is OK as long as the symlink is
      relative and remains
653 +     // within the git source tree.
654 +     linkSrc := filepath.Join(gitRoot, "linkSrc")
655 +     err = os.MkdirAll(linkSrc, os.ModePerm)
656 +     if err != nil {
657 +         t.Fatalf("failed to create test directory %q: %v", linkSrc, err)
658 +     }
659 +     linkDst := filepath.Join(gitRoot, "linkDst")
660 +     err = os.MkdirAll(linkDst, os.ModePerm)
661 +     if err != nil {
662 +         t.Fatalf("failed to create test directory %q: %v", linkDst, err)
663 +     }
664 +     // Create symlink from linkSrc/link to linkDst (directory)
665 +     err = os.Symlink("../linkDst", filepath.Join(linkSrc, "link"))
666 +     if err != nil {
```

```
667 +         t.Fatalf("failed to create symlink to linkDst directory: %v", err)
668 +     }
669 +     dockerBuilder := &DockerBuilder{
670 +         inputDir: buildDir,
671 +     }
672 +     localObj := buildapiv1.SecretBuildSource{
673 +         Secret: corev1.LocalObjectReference{
674 +             Name: "test",
675 +         },
676 +         DestinationDir: "secrets",
677 +     }
678 +     err = dockerBuilder.copyLocalObject(secretSource(localObj), secretRoot,
      gitRoot)
679 +     if err != nil {
680 +         t.Errorf("unexpected error copying local secret: %v", err)
681 +     }
682 +     // sourceSecret.txt should be copied to git-data/secrets/sourceSecret.txt
683 +     expectedFile := filepath.Join(gitRoot, "secrets", "sourceSecret.txt")
684 +     if _, err := os.Stat(expectedFile); err != nil {
685 +         t.Errorf("failed to stat %q: %v", expectedFile, err)
686 +     }
687 +     // Try again, this time copying to a destination that happens to be a
      symlink to another
688 +     // directory
689 +     localObj.DestinationDir = filepath.Join("linkSrc", "link")
690 +     err = dockerBuilder.copyLocalObject(secretSource(localObj), secretRoot,
      gitRoot)
691 +     if err != nil {
692 +         t.Errorf("unexpected error copying local secret: %v", err)
693 +     }
694 +     // sourceSecret.txt should be copied to git-
      data/linkSrc/link/sourceSecret.txt
695 +     expectedFile = filepath.Join(gitRoot, "linkSrc", "link",
      "sourceSecret.txt")
696 +     if _, err := os.Stat(expectedFile); err != nil {
697 +         t.Errorf("failed to stat %q: %v", expectedFile, err)
698 +     }
699 +     // sourceSecret.txt should also show up in git-
      data/linkDst/sourceSecret.txt
700 +     expectedFile = filepath.Join(gitRoot, "linkDst", "sourceSecret.txt")
```

```
701 +     if _, err := os.Stat(expectedFile); err != nil {
702 +         t.Errorf("failed to stat %q: %v", expectedFile, err)
703 +     }
704 +
705 + }
706 +
707 + // TestCopyLocalObjectZipSlip verifies that any copied Secret or ConfigMap is
      not able to be placed
708 + // outside of the target directory. A well-crafted symbolic link can otherwise
      allow secret data
709 + // to be placed in a location outside of our trusted target dirctory.
710 + func TestCopyLocalObjectZipSlip(t *testing.T) {
711 +     // Test scenario consists of a file tree that simulates the git clone
      environment
712 +     // /tmp/source-code-xxxx
713 +     // ├── git-data
714 +     // │   ├── Dockerfile
715 +     // │   └── hello.txt
716 +     // │   └── pwn-link -> /tmp/source-code-xxxx/pwned
717 +     // ├── secrets
718 +     // │   └── test
719 +     // │       └── pwned.txt
720 +     // └── pwned
721 +     //
722 +     // If data ends up in the "pwned" directory, we have a "zip-slip"
      vulnerability.
723 +     buildDir, err := os.MkdirTemp("", "source-code")
724 +     if err != nil {
725 +         t.Fatalf("failed to create temp dir: %v", err)
726 +     }
727 +     defer func() {
728 +         if err := os.RemoveAll(buildDir); err != nil {
729 +             t.Fatalf("failed to clean up temp dir: %v", err)
730 +         }
731 +     }()
732 +     // Set up secrets/test directory, with "pwned.txt" file
733 +     secretRoot := filepath.Join(buildDir, "secrets")
734 +     testPwnedFile := filepath.Join(secretRoot, "test", "pwned.txt")
735 +     err = fsCopyFile("testdata/fakesecret/pwned.txt", testPwnedFile)
736 +     if err != nil {
```

```
737 +         t.Fatalf("failed to copy secret data: %v", err)
738 +     }
739 +     if _, err := os.Stat(testPwnedFile); err != nil {
740 +         t.Fatalf("failed to stat pwned.txt: %v", err)
741 +     }
742 +     pwnedPath := filepath.Join(buildDir, "pwned")
743 +     err = os.MkdirAll(pwnedPath, os.ModePerm)
744 +     if err != nil {
745 +         t.Fatalf("failed to create test directory %q: %v", pwnedPath, err)
746 +     }
747 +     // Set up "git-data" directory as fake "git source"
748 +     gitRoot := filepath.Join(buildDir, "git-data")
749 +     err = filepath.Walk("testdata/fakesource", func(path string, info
      os.FileInfo, err error) error {
750 +         if err != nil {
751 +             return err
752 +         }
753 +         if info.IsDir() {
754 +             return nil
755 +         }
756 +         dstBase := filepath.Base(path)
757 +         return fsCopyFile(path, filepath.Join(gitRoot, dstBase))
758 +     })
759 +     if err != nil {
760 +         t.Fatalf("failed to copy test data: %v", err)
761 +     }
762 +     // Add symlink to "pwned" directory in the git root
763 +     err = os.Symlink(pwnedPath, filepath.Join(gitRoot, "pwn-link"))
764 +     if err != nil {
765 +         t.Fatalf("failed to create symlink to pwned directory: %v", err)
766 +     }
767 +     dockerBuilder := &DockerBuilder{
768 +         inputDir: buildDir,
769 +     }
770 +     localObj := buildapiv1.SecretBuildSource{
771 +         Secret: corev1.LocalObjectReference{
772 +             Name: "test",
773 +         },
774 +         DestinationDir: "pwn-link",
775 +     }
```

```
776 +     err = dockerBuilder.copyLocalObject(secretSource(localObj), secretRoot,
    gitRoot)
777 +     if err == nil {
778 +         t.Error("expected copyLocalObject to fail if the secret tries to write
    its contents outside the destination directory")
779 +     }
780 +     if err != nil && !strings.Contains(err.Error(), "outside of the target
    directory") {
781 +         t.Errorf("expected error message to contain %q, got: %v", "outside of
    the target directory", err)
782 +     }
783 +     // Can we stat the pwned.txt file?
784 +     badPwnedFile := filepath.Join(pwnedPath, "pwned.txt")
785 +     if _, err := os.Stat(badPwnedFile); err == nil {
786 +         t.Errorf("secret file %q was copied outside of the trusted build
    directory to %q", testPwnedFile, badPwnedFile)
787 +     }
788 + }
789 +
790 + func fsCopyFile(srcPath, dstPath string) error {
791 +     srcFile, err := os.Open(srcPath)
792 +     if err != nil {
793 +         return err
794 +     }
795 +     defer srcFile.Close()
796 +     parentDir := filepath.Dir(dstPath)
797 +     if err := os.MkdirAll(parentDir, os.ModePerm); err != nil {
798 +         return err
799 +     }
800 +     dstFile, err := os.Create(dstPath)
801 +     if err != nil {
802 +         return err
803 +     }
804 +     defer dstFile.Close()
805 +     if _, err := io.Copy(dstFile, srcFile); err != nil {
806 +         return err
807 +     }
808 +     return nil
809 + }
```

...build/builder/testdata/fakesecret/pwned.txt

```
@@ -0,0 +1 @@
+ Womp womp - you've been pwned!
```

...uilder/testdata/fakesecret/sourceSecret.txt

```
@@ -0,0 +1,2 @@
+ Top secret data!
+ This should be copied into the same directory as the source code.
```

...uild/builder/testdata/fakesource/Dockerfile

```
@@ -0,0 +1,7 @@
+ FROM registry.access.redhat.com/ubi9/ubi-minimal:latest
+
+ WORKDIR /opt/app-root/src
+
+ COPY hello.txt hello.txt
+
+ CMD ["cat", "hello.txt"]
```

...build/builder/testdata/fakesource/hello.txt

```
@@ -0,0 +1 @@
+ Hello OpenShift!
```

Comments 0

Please sign in to comment.