August 2023

ASSESSING RISK INTRODUCED THROUGH A CODE CHANGE

Dhruv H Raithatha

Girish Sivasubramanian

John A Foley

Chandra Mohan Babu Nadiminti

Randy Birdsall

ASSESSING RISK INTRODUCED THROUGH A CODE CHANGE

AUTHORS:
Dhruv H Raithatha
Girish Sivasubramanian
John A Foley
Chandra Mohan Babu Nadiminti
Randy Birdsall

ABSTRACT

Techniques are presented herein that shift the risk assessment focus during a software development process, away from the traditional end-of-process review (when a new feature is delivered, or an application is deployed) to earlier in the process when developers are actively at work. Such an approach allows a developer to assess the risk that a candidate software change is about to introduce prior to the developer committing that change, providing the developer with time (during the early portion of the process) to revisit the software and eliminate the identified risk. Aspects of the presented techniques leverage elements of a continuous integration (CI) and continuous deployment (CD) facility, the results that are available from existing unit and end-to-end tests, and the collection and analysis of OpenTelemetry (OTEL)-based metrics, events, logs, and traces (MELT) data to deliver security insights.

DETAILED DESCRIPTION

Traditional security monitoring tools, in the form of software bill of material (SBOM) analysis tools or data protection tools, typically encompass insights that are gathered at the cusp of delivering a new feature or deploying an application. While such an approach allows developers and security auditors to identify issues when they are critical, at that late stage of the process the developers themselves are not immediately available as they have typically moved on to other development activities.

Techniques are presented herein that address this situation by shifting a risk assessment focus ‘all the way left,’ that is to earlier in the process when developers are actively at work. Such an approach allows a developer to assess the risk that a candidate software change is about to introduce prior to the developer committing that change,

Published by Technical Disclosure Commons, 2023

providing the developer with time (during the early portion of the process) to revisit the software and eliminate the identified risk.

The presented techniques leverage elements of the modern software development and operations (DevOps)-powered environment, where applications and products are built on the shoulders of comprehensive test coverage across all of the application's code. Such code coverage (through unit tests and end-to-end tests) ensures that changes that are made by a developer do not cause functional or performance regressions or any other operational impacts. Currently, such end-to-end tests are used to obtain functional coverage over new code, typically across every new commit that is submitted. The presented techniques enhance the value that is derived from such coverage by collecting and deriving security insights from published metrics, events, logs, and traces (MELT) data that may be available from the various pieces of the end-to-end tests.

Aspects of the presented techniques support the collection of OpenTelemetry (OTEL)-based data regarding the libraries that are used by an application, a base docker image that is used by a service, third party application programming interfaces (APIs) that are referenced by a service, application logs, container logs, and data flows between services. The collected data may then be correlated with the latest available vulnerability reports, sensitive data policies, information on potential misconfigurations, and reports on vulnerable third-party APIs to establish a potential risk to a service or an application, both without and then with a candidate change. Importantly, all of the above-described activities may occur at a pre-commit stage, before a developer commits a candidate change.

The above-described correlation of data (including metrics, logs, traces, and spans of a service consumed) in an OTEL format, when tied to the latest threat intelligence about third-party and operating system (OS) vulnerabilities along with data sensitivity policies, provides for a real-time indication of a risk that may be introduced by a candidate change, thus allowing such a risk to be proactively mitigated.

As described above, the presented techniques support the collection of a range of data (such as events, metrics, logs, etc.). Illustrative examples of several of those data types are presented below.

Regarding events, by examining the packages that are being used, such data allows for the identification of an existing or newly detected vulnerability that may be introduced through a commit operation. An example of such data may comprise:

Resource attributes:
-> container.id:
Str(5d15e91c2e65d775c56c197da98c6cdab9168a647242cbd1a62dfc7c91d61319)
-> container.runtime: Str(containerd)
-> host.name: Str(ip-10-0-0-160.us-west-2.compute.internal)
-> container.short_id: Str(5d15e91c2e65)
-> container.created_at: Int(1681345980000)
-> container.name: Str(featureflagservice)
-> k8s.pod.name: Str(demo-featureflagservice-7d7b87b59d-rgm75)
-> k8s.namespace.name: Str(otel-demo)
-> container.image.name: Str(ghcr.io/open-telemetry/demo:1.3.0-featureflagservice)
-> container.memory_limit: Int(183500800)
-> telemetry.sdk.name: Str(infra-agent)

ScopeEvents #0
ScopeEvents SchemaURL:
InstrumentationScope
Event #0
packages:[github.com/EscherAuth/escher v0.0.0-20200415232717-f57476610940,
github.com/cloudevents/sdk-go/v2 v2.14.0, github.com/gammazero/workerpool v1.1.3, github.com/go-playground/validator/v10 v10.14.1]
Attributes:
-> event.name: libraries
-> event.description: security
...

Regarding traces, such data can aid in detecting if a service is, as part of a candidate change, about to become Internet-facing, communicate with any third-party APIs that are potentially risky, or have new or existing access to a data store. An example of such data may comprise:

```
Resource attributes:
     -> container.id:
Str(5d15e91c2e65d775c56c197da98c6cdab9168a647242cbd1a62dfc7c91d61319)
     -> container.runtime: Str(containerd)
     -> host.name: Str(ip-10-0-0-160.us-west-2.compute.internal)
     -> container.short_id: Str(5d15e91c2e65)
     -> container.created_at: Int(1681345980000)
     -> container.name: Str(featureflagservice)
     -> k8s.pod.name: Str(demo-featureflagservice-7d7b87b59d-rgm75)
     -> k8s.namespace.name: Str(otel-demo)
     -> container.image.name: Str(ghcr.io/open-telemetry/demo:1.3.0-featureflagservice)
     -> container.memory_limit: Int(183500800)
     -> telemetry.sdk.name: Str(infra-agent)
Span #0
          Trace ID       : 7f891af08cc2702ce78ace295ed07c6b
          Parent ID      : 9583938e1df414e8
          ID             : e8124f459a5a5814
          Name           : POST /getquote
          Kind           : Serve
          Start time     : 2023-04-18 02:02:47.724021454 +0000 UTC
          End time       : 2023-04-18 02:02:47.724588282 +0000 UTC
          Status code    : Unset
          Status message :
Attributes:
          -> code.function: Str(handle)
          -> code.namespace: Str(Slim\App)
          -> code.filepath: Str(/var/www/vendor/slim/slim/Slim/App.go)
          -> code.lineno: Int(213)
          -> http.url: Str(http://stock-service/getquote)
          -> http.method: Str(POST)
          -> http.request_content_length: Str(19)
          -> http.scheme: Str(http)
          -> http.status_code: Int(200)
          -> http.flavor: Str(1.1)
          -> http.response_content_length: Str()
Span #1
Trace ID : 7f891af08cc2702ce78ace295ed07c6b
```

Parent ID :
ID : ce1d326d1d8a112a
Name : HTTP POST
Kind : Client
Start time : 2023-04-18 02:02:43.783612188 +0000 UTC
End time : 2023-04-18 02:02:43.805723017 +0000 UTC
Status code : Unset
Status message :
Attributes:
-> http.method: Str(POST)
-> http.url: Str(http://yahoo-finance/api/cart)
-> http.status_code: Int(200)
Span #2
Trace ID : 2fad14a7ab95d7d560d93d7a6815c979
Parent ID :
ID : b47238e0a74a3c44
Name : DB GET
Kind : Client
Start time : 2023-04-18 02:02:43.805955509 +0000 UTC
End time : 2023-04-18 02:02:43.816366651 +0000 UTC
Status code : Unset
Status message :
Attributes:
-> http.method: Str(GET)
-> http.url: Str(postgresql://127.0.0.1:5432)
...

Regarding metrics, such data about a resource's usage may be tied to the cost that is associated with hosting a service. An example of such data may comprise:

Resource attributes:
-> container.id:
Str(5d15e91c2e65d775c56c197da98c6cdab9168a647242cbd1a62dfc7c91d61319)
-> container.runtime: Str(containerd)
-> host.name: Str(ip-10-0-0-160.us-west-2.compute.internal)
-> container.short_id: Str(5d15e91c2e65)
-> container.created_at: Int(1681345980000)

```
    -> container.name: Str(featureflagservice)
    -> k8s.pod.name: Str(demo-featureflagservice-7d7b87b59d-rgm75)
    -> k8s.namespace.name: Str(otel-demo)
    -> container.image.name: Str(ghcr.io/open-telemetry/demo:1.3.0-featureflagservice)
    -> container.memory_limit: Int(183500800)
    -> telemetry.sdk.name: Str(infra-agent)

ScopeMetrics #0
ScopeMetrics SchemaURL:
InstrumentationScope
Metric #0
Descriptor:
    -> Name: container.memory.limit
    -> Description:
    -> Unit:
    -> DataType: Summary

SummaryDataPoints #0
StartTimestamp: 2023-04-18 02:03:01 +0000 UTC
Timestamp: 2023-04-18 02:03:20 +0000 UTC

Count: 1
Sum: 183500800.000000
QuantileValue #0: Quantile 0.000000, Value 183500800.000000
QuantileValue #1: Quantile 1.000000, Value 183500800.000000
...
```

Figure 1, below, presents elements of an exemplary process flow that is possible according to the techniques presented herein and that is reflective of the above discussion.

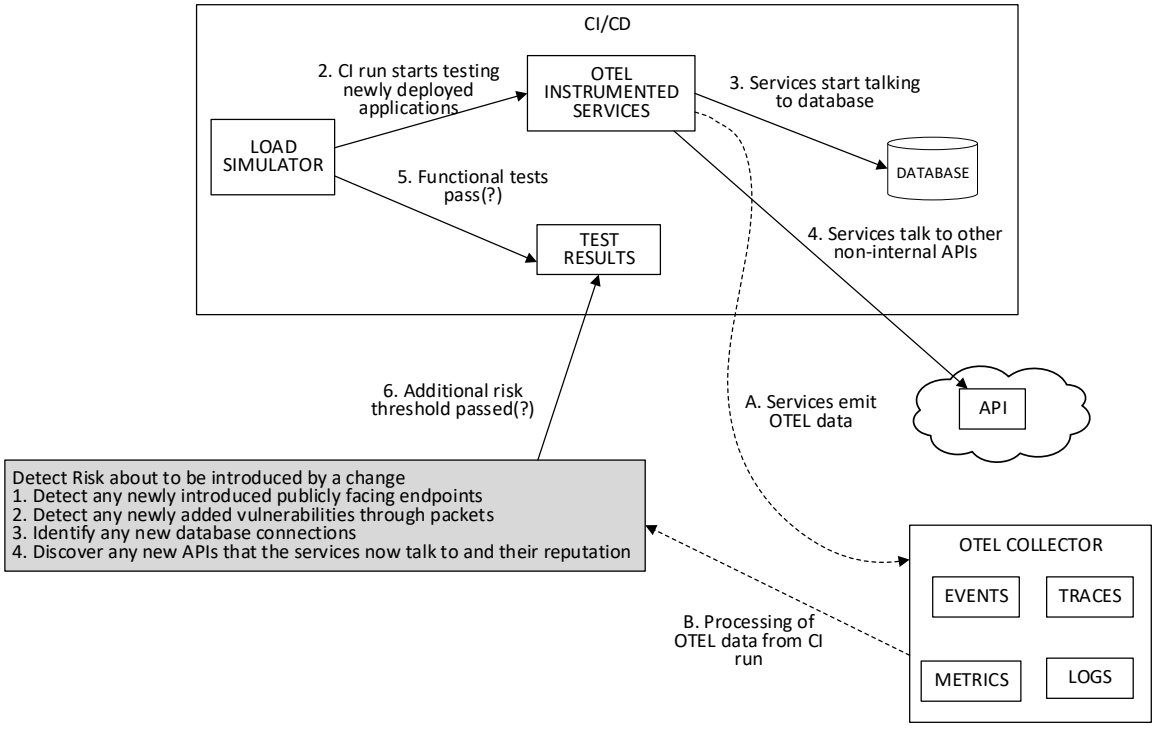

Figure 1: Exemplary Process Flow

As shown in Figure 1, above, the presented techniques encompass, among other things, a continuous integration (CI) and continuous deployment (CD) facility and an OTEL Collector (supporting, as described previously, MELT data). Further, the figure depicts various of the steps that the presented techniques may follow to achieve the above-described functionality.

During a first step (1), a developer submits to a CI/CD facility a change to a code branch on which they have been working. Under a second step (2), a load simulator within the CI/CD facility begins a CI run to test the application within which the code branch resides (since that application will, if the code change is accepted, be updated).

At a third step (3), services (that have been OTEL-instrumented) that are invoked by the application begin communicating with a database and at a fourth step (4) begin communicating with other non-internal (i.e., external) APIs. During those steps, the different services may emit OTEL data (to the OTEL Collector) regarding metrics, events, logs, and traces (i.e., MELT data).

During a fifth step (5), the load simulator issues an indication whether the instant functional tests passed and generates a test results artifact.

Under a sixth step (6), the OTEL data from the CI run (as described above for the third step) is processed and an indication is issued whether the additional risk threshold assessment was passed. That assessment may encompass, among other things, a detection of any newly introduced publicly facing endpoints, a detection of any newly added vulnerabilities through packages, an identification of any new database connections, a discovery of any new APIs that the services now communicate with and an assessment of their reputation, etc. At the end of such a process the test results artifact may be updated.

In summary, techniques have been presented herein that shift the risk assessment focus during a software development process, away from the traditional end-of-process review (when a new feature is delivered, or an application is deployed) to earlier in the process when developers are actively at work. Such an approach allows a developer to assess the risk that a candidate software change is about to introduce prior to the developer committing that change, providing the developer with time (during the early portion of the process) to revisit the software and eliminate the identified risk. Aspects of the presented techniques leverage elements of a CI and CD facility, the results that are available from existing unit and end-to-end tests, and the collection and analysis of OTEL-based MELT data to deliver security insights.