Component Info	golang:golang.org/x/text
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/golang.org\x\text@v0.3.0]
Non Vulnerable Version	[v0.10.0, v0.11.0, v0.12.0, v0.13.0, v0.14.0, v0.15.0, v0.16.0, v0.17.0, v0.18.0, v0.19.0, v0.20.0, v0.21.0, v0.22.0, v0.23.0, v0.24.0, v0.3.8, v0.4.0, v0.5.0, v0.6.0, v0.7.0, v0.8.0, v0.9.0]
CVE-2020-14040	

Vulnerability

Issue

CVE-2020-14040

Severity

CVE CVSS 3: 7.5
CVE CVSS 2.0: 5.0
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 835

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

The x/text package before 0.3.3 for Go has a vulnerability in encoding/unicode that could lead to the UTF-16 decoder entering an infinite loop, causing the program to crash or run out of memory. An attacker could provide a single byte to a UTF16 decoder instantiated with UseBOM or ExpectBOM to trigger an infinite loop if the String function on the Decoder is called, or the Decoder is passed to golang.org/x/text/transform.String.

Explanation

The `x/text` package is vulnerable to Denial of Service (DoS). The `Transform()` function in the `unicode.go` file and the `String()` function in the `transform.go` file do not properly process a single byte string when using the UTF-16 decoder with the `UseBOM` or `ExpectBOM` BOM policy. An attacker can exploit this behavior by supplying a single byte to the UTF-16 decoder which, when decoded, will trigger an infinite loop, causing the application to crash, ultimately leading to a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/golang/go/issues/39491
Third Party: https://groups.google.com/forum/#!topic/golang-announce/bXVe…

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE-2021-38561

Vulnerability

Issue

CVE-2021-38561

Severity

CVE CVSS 3: 7.5
Sonatype CVSS 3: 4.3

Weakness

CVE CWE: 125

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

golang.org/x/text/language in golang.org/x/text before 0.3.7 can panic with an out-of-bounds read during BCP 47 language tag parsing. Index calculation is mishandled. If parsing untrusted user input, this can be used as a vector for a denial-of-service attack.

Explanation

The `golang.org/x/text` package is vulnerable due to an Out-of-bounds Read. The files and functions listed below do not properly handle index calculations when parsing formatted language tags. A remote attacker can exploit this behavior by supplying a specially-crafted language tag to trigger a panic, causing an application crash and ultimately a Denial of Service (DoS) condition.

Vulnerable File(s) and Function(s):

internal/language/language.go

- ParseExtension()
- ParseBase()
- ParseScript()
- ParseRegion()
- ParseVariant()

internal/language/parse.go

- Parse()

language/parse.go

- Parse()
- Compose()
- ParseAcceptLanguage()

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Third Party: https://golangissues.com/issues/1154451
Third Party: https://osv.dev/vulnerability/GO-2021-0113

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE-2022-32149

Vulnerability

Issue

CVE-2022-32149

Severity

CVE CVSS 3: 7.5
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 772

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

An attacker may cause a denial of service by crafting an Accept-Language header which ParseAcceptLanguage will take significant time to parse.

Explanation

The `golang.org/x/text` package is vulnerable to a Denial of Service (DoS) attack. The `ParseAcceptLanguage()` function in the `parse.go` file fails to account for strings containing large numbers of – characters when parsing `Accept-Language` headers. A remote attacker can exploit this vulnerability by submitting a request that leverages the aforementioned header. This will cause the application to expend excessive resources to process the request, resulting in a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Evidence: https://github.com/golang/vulndb/commit/c4e6da635d3b5a5ce387…
Project: https://github.com/golang/go/issues/56152

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	golang:gopkg.in/yaml.v2
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/gopkg.in\yaml.v2@v2.2.1]
Non Vulnerable Version	[v2.2.4, v2.2.5, v2.2.6, v2.2.7, v2.2.8, v2.3.0, v2.4.0]
CVE-2022-3064	**Vulnerability** **Issue** CVE-2022-3064 **Severity** CVE CVSS 3: 7.5 Sonatype CVSS 3: 7.5 **Weakness** CVE CWE: 400 **Source** National Vulnerability Database **Categories** Data **Description** **Description from CVE** Parsing malicious or large YAML documents can consume excessive amounts of CPU or memory. **Explanation** This issue has undergone the Sonatype Fast-Track process. For more information, please see the Sonatype Knowledge Base Guide. **Advisories** Evidence: https://github.com/go-yaml/yaml/releases/tag/v2.2.4 Project: https://github.com/go-yaml/yaml/pull/515 Third Party: https://pkg.go.dev/vuln/GO-2022-0956 **CVSS Details** CVE CVSS 3: 7.5 CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
sonatype-2019-0900	**Vulnerability** **Issue** sonatype-2019-0900 **Severity** Sonatype CVSS 3: 6.5 **Weakness** Sonatype CWE: 400

Source

Sonatype Data Research

Categories

Data

Description

Explanation

The `gopkg.in/yaml` package is vulnerable to Denial of Service (DoS) attacks. The `unmarshal()` function in the `decode.go` file fails to control the number of aliases when unmarshalling values in a YAML document, this allows malicious actors to excessively consume resources by nesting multiple aliases. A remote attacker can exploit this vulnerability by submitting a specially-crafted YAML document to cause a DoS condition in the underlying system.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Evidence: https://github.com/go-yaml/yaml/releases/tag/v2.2.3
Project: https://github.com/go-yaml/yaml/pull/375
Project: https://github.com/go-yaml/yaml/pull/555
Project: https://pkg.go.dev/vuln/GO-2021-0061
Third Party: https://deps.dev/advisory/OSV/GO-2021-0061

CVSS Details

Sonatype CVSS 3: 6.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:google.golang.org/protobuf**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\protobuf@v1.22.0]
Non Vulnerable Version	[v1.33.0, v1.34.0, v1.34.1, v1.34.2, v1.35.0, v1.35.1, v1.35.2, v1.36.0, v1.36.1, v1.36.2, v1.36.3, v1.36.4, v1.36.5, v1.36.6]

CVE-2024-24786

Vulnerability

Issue

CVE-2024-24786

Severity

Sonatype CVSS 3: 7.5

Weakness

Sonatype CWE: 835

Source

National Vulnerability Database

Categories

Data
Functional

Description

Description from CVE

The protojson.Unmarshal function can enter an infinite loop when unmarshaling certain forms of invalid JSON. This condition can occur when unmarshaling into a message which contains a google.protobuf.Any value, or when the UnmarshalOptions.DiscardUnknown option is set.

Explanation

The `protobuf` package is vulnerable to Denial of Service (DoS) attacks due to an Infinite Loop. The `Read()` function within the `decode.go` file fails to account for JSON objects containing keys that lack values (i.e. `{"":}`). A remote attacker can exploit this behavior by supplying malformed JSON to cause affected applications to enter into an infinite loop, ultimately resulting in a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://pkg.go.dev/vuln/GO-2024-2611

CVSS Details

Sonatype CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:google.golang.org/protobuf**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\protobuf@v1.24.0]
Non Vulnerable Version	[v1.33.0, v1.34.0, v1.34.1, v1.34.2, v1.35.0, v1.35.1, v1.35.2, v1.36.0, v1.36.1, v1.36.2, v1.36.3, v1.36.4, v1.36.5, v1.36.6]

CVE-2024-24786

Vulnerability

Issue

CVE-2024-24786

Severity

Sonatype CVSS 3: 7.5

Weakness

Sonatype CWE: 835

Source

National Vulnerability Database

Categories

Data
Functional

Description

Description from CVE

The protojson.Unmarshal function can enter an infinite loop when unmarshaling certain forms of invalid JSON. This condition can occur when unmarshaling into a message which contains a google.protobuf.Any value, or when the UnmarshalOptions.DiscardUnknown option is set.

Explanation

The `protobuf` package is vulnerable to Denial of Service (DoS) attacks due to an Infinite Loop. The `Read()` function within the `decode.go` file fails to account for JSON objects containing keys that lack values (i.e. `{"":}`). A remote attacker can exploit this behavior by supplying malformed JSON to cause affected applications to enter into an infinite loop, ultimately resulting in a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://pkg.go.dev/vuln/GO-2024-2611

CVSS Details

Sonatype CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:google.golang.org/grpc**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\grpc@v1.23.0]
Non Vulnerable Version	[v1.56.3, v1.57.0-dev, v1.57.1, v1.57.2, v1.58.0-dev, v1.58.3, v1.59.0, v1.59.0-dev, v1.60.0, v1.60.0-dev, v1.60.1, v1.61.0, v1.61.0-dev, v1.61.1, v1.61.2, v1.62.0, v1.62.1, v1.62.2, v1.63.0, v1.63.1, v1.63.2, v1.63.3, v1.64.1, v1.65.0, v1.65.1, v1.66.0, v1.66.1, v1.66.2, v1.66.3, v1.67.0, v1.67.0-dev, v1.67.1, v1.67.2, v1.67.3, v1.68.0, v1.68.0-dev, v1.68.1, v1.68.2, v1.69.0, v1.69.0-dev, v1.69.2, v1.69.4, v1.70.0, v1.70.0-dev, v1.71.0, v1.71.0-dev, v1.71.1, v1.72.0-dev, v1.73.0-dev]
sonatype-2023-4386	

Vulnerability

Issue

sonatype-2023-4386

Severity

Sonatype CVSS 3: 7.5

Weakness

Sonatype CWE: 400

Source

Sonatype Data Research

Categories

Data
Operational

Description

Explanation

The `grpc` package is vulnerable to a Denial of Service (DoS) attack. The `serveStreams()` function in the `server.go` file does not set a limit for the maximum concurrent HTTP/2 streams. A remote attacker can exploit this vulnerability by submitting a large number of reset requests in rapid succession in order to establish an unbounded number of concurrent connections. These requests may transition from an open to a closed state before the server processes their cancellation. As a result, this will exhaust the connection pool of affected applications and ultimately result in a DoS condition for their users.

Note: This vulnerability exists under the umbrella of CVE-2023-44487.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/grpc/grpc-go/pull/6703
Third Party: https://blog.cloudflare.com/technical-breakdown-http2-rapid-…
Third Party: https://blog.cloudflare.com/zero-day-rapid-reset-http2-recor…
Third Party: https://github.com/advisories/GHSA-m425-mq94-257g

CVSS Details

Sonatype CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:google.golang.org/grpc**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\grpc@v1.27.0]
Non Vulnerable Version	[v1.56.3, v1.57.0-dev, v1.57.1, v1.57.2, v1.58.0-dev, v1.58.3, v1.59.0, v1.59.0-dev, v1.60.0, v1.60.0-dev, v1.60.1, v1.61.0, v1.61.0-dev, v1.61.1, v1.61.2, v1.62.0, v1.62.1, v1.62.2, v1.63.0, v1.63.1, v1.63.2, v1.63.3, v1.64.1, v1.65.0, v1.65.1, v1.66.0, v1.66.1, v1.66.2, v1.66.3, v1.67.0, v1.67.0-dev, v1.67.1, v1.67.2, v1.67.3, v1.68.0, v1.68.0-dev, v1.68.1, v1.68.2, v1.69.0, v1.69.0-dev, v1.69.2, v1.69.4, v1.70.0, v1.70.0-dev, v1.71.0, v1.71.0-dev, v1.71.1, v1.72.0-dev, v1.73.0-dev]

sonatype-2023-4386

Vulnerability

Issue

sonatype-2023-4386

Severity

Sonatype CVSS 3: 7.5

Weakness

Sonatype CWE: 400

Source

Sonatype Data Research

Categories

Data
Operational

Description

Explanation

The `grpc` package is vulnerable to a Denial of Service (DoS) attack. The `serveStreams()` function in the `server.go` file does not set a limit for the maximum concurrent HTTP/2 streams. A remote attacker can exploit this vulnerability by submitting a large number of reset requests in rapid succession in order to establish an unbounded number of concurrent connections. These requests may transition from an open to a closed state before the server processes their cancellation. As a result, this will exhaust the connection pool of affected applications and ultimately result in a DoS condition for their users.

Note: This vulnerability exists under the umbrella of CVE-2023-44487.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/grpc/grpc-go/pull/6703
Third Party: https://blog.cloudflare.com/technical-breakdown-http2-rapid-…
Third Party: https://blog.cloudflare.com/zero-day-rapid-reset-http2-recor…
Third Party: https://github.com/advisories/GHSA-m425-mq94-257g

CVSS Details

Sonatype CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:google.golang.org/protobuf**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\protobuf@v1.26.0-rc.1]
Non Vulnerable Version	[v1.33.0, v1.34.0, v1.34.1, v1.34.2, v1.35.0, v1.35.1, v1.35.2, v1.36.0, v1.36.1, v1.36.2, v1.36.3, v1.36.4, v1.36.5, v1.36.6]

CVE-2024-24786

Vulnerability

Issue

CVE-2024-24786

Severity

Sonatype CVSS 3: 7.5

Weakness

Sonatype CWE: 835

Source

National Vulnerability Database

Categories

Data
Functional

Description

Description from CVE

The protojson.Unmarshal function can enter an infinite loop when unmarshaling certain forms of invalid JSON. This condition can occur when unmarshaling into a message which contains a google.protobuf.Any value, or when the UnmarshalOptions.DiscardUnknown option is set.

Explanation

The `protobuf` package is vulnerable to Denial of Service (DoS) attacks due to an Infinite Loop. The `Read()` function within the `decode.go` file fails to account for JSON objects containing keys that lack values (i.e. `{"":}`). A remote attacker can exploit this behavior by supplying malformed JSON to cause affected applications to enter into an infinite loop, ultimately resulting in a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://pkg.go.dev/vuln/GO-2024-2611

CVSS Details

Sonatype CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:google.golang.org/protobuf**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\protobuf@v1.23.0]
Non Vulnerable Version	[v1.33.0, v1.34.0, v1.34.1, v1.34.2, v1.35.0, v1.35.1, v1.35.2, v1.36.0, v1.36.1, v1.36.2, v1.36.3, v1.36.4, v1.36.5, v1.36.6]

CVE-2024-24786

Vulnerability

Issue

CVE-2024-24786

Severity

Sonatype CVSS 3: 7.5

Weakness

Sonatype CWE: 835

Source

National Vulnerability Database

Categories

Data
Functional

Description

Description from CVE

The protojson.Unmarshal function can enter an infinite loop when unmarshaling certain forms of invalid JSON. This condition can occur when unmarshaling into a message which contains a google.protobuf.Any value, or when the UnmarshalOptions.DiscardUnknown option is set.

Explanation

The `protobuf` package is vulnerable to Denial of Service (DoS) attacks due to an Infinite Loop. The `Read()` function within the `decode.go` file fails to account for JSON objects containing keys that lack values (i.e. `{"":}`). A remote attacker can exploit this behavior by supplying malformed JSON to cause affected applications to enter into an infinite loop, ultimately resulting in a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://pkg.go.dev/vuln/GO-2024-2611

CVSS Details

Sonatype CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:github.com/prometheus/client_golang**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/github.com\prometheus\client_golang@v1.7.1]
Non Vulnerable Version	[v1.11.1, v1.12.0, v1.12.1, v1.12.2, v1.13.0, v1.13.1, v1.14.0, v1.15.0, v1.15.1, v1.16.0, v1.17.0, v1.18.0, v1.19.0, v1.19.1, v1.20.0, v1.20.1, v1.20.2, v1.20.3, v1.20.4, v1.20.5, v1.21.0, v1.21.0-rc.0, v1.21.1, v1.22.0-rc.0]

CVE-2022-21698

Vulnerability

Issue

CVE-2022-21698

Severity

CVE CVSS 3: 7.5
CVE CVSS 2.0: 5.0
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 400, 770

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

client_golang is the instrumentation library for Go applications in Prometheus, and the promhttp package in client_golang provides tooling around HTTP servers and clients. In client_golang prior to version 1.11.1, HTTP server is susceptible to a Denial of Service through unbounded cardinality, and potential memory exhaustion, when handling requests with non-standard HTTP methods. In order to be affected, an instrumented software must use any of `promhttp.InstrumentHandler*` middleware except `RequestsInFlight`; not filter any specific methods (e.g GET) before middleware; pass metric with `method` label name to our middleware; and not have any firewall/LB/proxy that filters away requests with unknown `method`. client_golang version 1.11.1 contains a patch for this issue. Several workarounds are available, including removing the `method` label name from counter/gauge used in the InstrumentHandler; turning off affected promhttp handlers; adding custom middleware before promhttp handler that will sanitize the request method given by Go http.Request; and using a reverse proxy or web application firewall, configured to only allow a limited set of methods.

Explanation

The `github.com/prometheus/client_golang` package is vulnerable to a Denial of Service (DoS) attack. The `labels()` and `sanitizeMethod()` functions in the `instrument_server.go` file fail to sufficiently sanitize non-standard HTTP methods when processing requests. A remote attacker can exploit this vulnerability to cause the application to consume all available resources by issuing requests leveraging non-standard HTTP methods to any affected endpoint(s).

Advisory Deviation Notice: The Sonatype security research team discovered that this vulnerability was introduced in version v0.9.0-pre1 and therefore does not affect all versions prior to v1.11.1 as stated in the advisory.

Detection

The application is vulnerable by using this component if the application:

- [Uses] any of `promhttp.InstrumentHandler*` middleware except RequestsInFlight.
- [Does] not filter any specific methods (e.g GET) before middleware.
- [Passes] metric with method label name to our middleware.
- [Does not] have any firewall/LB/proxy that filters away requests with unknown method.

Reference: https://github.com/prometheus/client_golang/security/advisories/GHSA-cg3q-j54f-5p7p

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Alternatively, if upgrading is not a viable option, the project has provided the following workaround(s):

- Remove method label name from counter/gauge you use in the `InstrumentHandler`.
- Turn off affected `promhttp` handlers.
- Add custom middleware before `promhttp` handler that will sanitize the request method given by Go `http.Request`.
- Use a reverse proxy or web application firewall, configured to only allow a limited set of methods.

Reference: https://github.com/prometheus/client_golang/security/advisories/GHSA-cg3q-j54f-5p7p

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/prometheus/client_golang/pull/962
Project: https://github.com/prometheus/client_golang/pull/987
Project: https://github.com/prometheus/client_golang/security/advisor…
Third Party: https://bugzilla.redhat.com/show_bug.cgi?id=2045880

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE-2023-45142

Vulnerability

Issue

CVE-2023-45142

Severity

CVE CVSS 3: 7.5
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 770

Source

National Vulnerability Database

Categories

Data
Operational

Description

Description from CVE

OpenTelemetry-Go Contrib is a collection of third-party packages for OpenTelemetry-Go. A handler wrapper out of the box adds labels `http.user_agent` and `http.method` that have unbound cardinality. It leads to the server's potential memory exhaustion when many malicious requests are sent to it. HTTP header User-Agent or HTTP method for requests can be easily set by an attacker to be random and long. The library internally uses `httpconv.ServerRequest` that records every value for HTTP `method` and `User-Agent`. In order to be affected, a program has to use the `otelhttp.NewHandler` wrapper and not filter any unknown HTTP methods or User agents on the level of CDN, LB, previous middleware, etc. Version 0.44.0 fixed this issue when the values collected for attribute `http.request.method` were changed to be restricted to a set of well-known values and other high cardinality attributes were removed. As a workaround to stop being affected, `otelhttp.WithFilter()` can be used, but it requires manual careful configuration to not log certain requests entirely. For convenience and safe usage of this library, it should by default mark with the label `unknown` non-standard HTTP methods and User agents to show that such requests were made but do not increase cardinality. In case someone wants to stay with the current behavior, library API should allow to enable it.

Explanation

The `github.com/prometheus/client_golang` package is vulnerable to Denial of Service (DoS) attacks. The `sanitizeMethod()` function in `instrument_server.go` mishandles unknown request methods. A remote attacker can exploit this vulnerability with requests leveraging arbitrary HTTP methods in order to cause the `InstrumentHandler` to create an unbounded number of partitions and potentially exhaust all of the application's available memory.

Detection

The application is vulnerable by using this component if it meets the following criteria provided by the project:

- The application uses any of `promhttp.InstrumentHandler*` middleware except `RequestsInFlight`.
- The application does not filter any specific methods (e.g GET) before the `InstrumentHandler` middleware.
- The application passes metrics with `method` label name to the `InstrumentHandler` middleware.
- The application does not have any firewall/LB/proxy that filters away requests with unknown methods.

Reference: https://github.com/advisories/GHSA-cg3q-j54f-5p7p

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Alternatively, if upgrading is not a viable option, the project has provided the following mitigation strategy:

- Remove `method` label name from counter/gauge you use in the `InstrumentHandler`.
- Turn off affected `promhttp` handlers.
- Add custom middleware before `promhttp` handler that will sanitize the request method given by Go `http.Request`.

- Use a reverse proxy or web application firewall, configured to only allow a limited set of methods.

Reference: https://github.com/advisories/GHSA-cg3q-j54f-5p7p

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Third Party: https://github.com/advisories/GHSA-cg3q-j54f-5p7p

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:google.golang.org/grpc**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\grpc@v1.25.1]
Non Vulnerable Version	[v1.56.3, v1.57.0-dev, v1.57.1, v1.57.2, v1.58.0-dev, v1.58.3, v1.59.0, v1.59.0-dev, v1.60.0, v1.60.0-dev, v1.60.1, v1.61.0, v1.61.0-dev, v1.61.1, v1.61.2, v1.62.0, v1.62.1, v1.62.2, v1.63.0, v1.63.1, v1.63.2, v1.63.3, v1.64.1, v1.65.0, v1.65.1, v1.66.0, v1.66.1, v1.66.2, v1.66.3, v1.67.0, v1.67.0-dev, v1.67.1, v1.67.2, v1.67.3, v1.68.0, v1.68.0-dev, v1.68.1, v1.68.2, v1.69.0, v1.69.0-dev, v1.69.2, v1.69.4, v1.70.0, v1.70.0-dev, v1.71.0, v1.71.0-dev, v1.71.1, v1.72.0-dev, v1.73.0-dev]

sonatype-2023-4386

Vulnerability

Issue

sonatype-2023-4386

Severity

Sonatype CVSS 3: 7.5

Weakness

Sonatype CWE: 400

Source

Sonatype Data Research

Categories

Data
Operational

Description

Explanation

The `grpc` package is vulnerable to a Denial of Service (DoS) attack. The `serveStreams()` function in the `server.go` file does not set a limit for the maximum concurrent HTTP/2 streams. A remote attacker can exploit this vulnerability by submitting a large number of reset requests in rapid succession in order to establish an unbounded number of concurrent connections. These requests may transition from an open to a closed state before the server processes their cancellation. As a result, this will exhaust the connection pool of affected applications and ultimately result in a DoS condition for their users.

Note: This vulnerability exists under the umbrella of CVE-2023-44487.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/grpc/grpc-go/pull/6703
Third Party: https://blog.cloudflare.com/technical-breakdown-http2-rapid-…
Third Party: https://blog.cloudflare.com/zero-day-rapid-reset-http2-recor…
Third Party: https://github.com/advisories/GHSA-m425-mq94-257g

CVSS Details

Sonatype CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:golang.org/x/text**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/golang.org\x\text@v0.3.2]
Non Vulnerable Version	[v0.10.0, v0.11.0, v0.12.0, v0.13.0, v0.14.0, v0.15.0, v0.16.0, v0.17.0, v0.18.0, v0.19.0, v0.20.0, v0.21.0, v0.22.0, v0.23.0, v0.24.0, v0.3.8, v0.4.0, v0.5.0, v0.6.0, v0.7.0, v0.8.0, v0.9.0]

CVE-2020-14040

Vulnerability

Issue

CVE-2020-14040

Severity

CVE CVSS 3: 7.5
CVE CVSS 2.0: 5.0
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 835

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

The x/text package before 0.3.3 for Go has a vulnerability in encoding/unicode that could lead to the UTF-16 decoder entering an infinite loop, causing the program to crash or run out of memory. An attacker could provide a single byte to a UTF16 decoder instantiated with UseBOM or ExpectBOM to trigger an infinite loop if the String function on the Decoder is called, or the Decoder is passed to golang.org/x/text/transform.String.

Explanation

The `x/text` package is vulnerable to Denial of Service (DoS). The `Transform()` function in the `unicode.go` file and the `String()` function in the `transform.go` file do not properly process a single byte string when using the UTF-16 decoder with the `UseBOM` or `ExpectBOM` BOM policy. An attacker can exploit this behavior by supplying a single byte to the UTF-16 decoder which, when decoded, will trigger an infinite loop, causing the application to crash, ultimately leading to a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/golang/go/issues/39491
Third Party: https://groups.google.com/forum/#!topic/golang-announce/bXVe…

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE-2021-38561

Vulnerability

Issue

CVE-2021-38561

Severity

CVE CVSS 3: 7.5
Sonatype CVSS 3: 4.3

Weakness

CVE CWE: 125

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

golang.org/x/text/language in golang.org/x/text before 0.3.7 can panic with an out-of-bounds read during BCP 47 language tag parsing. Index calculation is mishandled. If parsing untrusted user input, this can be used as a vector for a denial-of-service attack.

Explanation

The `golang.org/x/text` package is vulnerable due to an Out-of-bounds Read. The files and functions listed below do not properly handle index calculations when parsing formatted language tags. A remote attacker can exploit this behavior by supplying a specially-crafted language tag to trigger a panic, causing an application crash and ultimately a Denial of Service (DoS) condition.

Vulnerable File(s) and Function(s):

internal/language/language.go

- ParseExtension()
- ParseBase()
- ParseScript()
- ParseRegion()
- ParseVariant()

internal/language/parse.go

- Parse()

language/parse.go

- Parse()
- Compose()
- ParseAcceptLanguage()

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Third Party: https://golangissues.com/issues/1154451
Third Party: https://osv.dev/vulnerability/GO-2021-0113

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE-2022-32149

Vulnerability

Issue

CVE-2022-32149

Severity

CVE CVSS 3: 7.5
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 772

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

An attacker may cause a denial of service by crafting an Accept-Language header which ParseAcceptLanguage will take significant time to parse.

Explanation

The `golang.org/x/text` package is vulnerable to a Denial of Service (DoS) attack. The `ParseAcceptLanguage()` function in the `parse.go` file fails to account for strings containing large numbers of – characters when parsing `Accept-Language` headers. A remote attacker can exploit this vulnerability by submitting a request that leverages the aforementioned header. This will cause the application to expend excessive resources to process the request, resulting in a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Evidence: https://github.com/golang/vulndb/commit/c4e6da635d3b5a5ce387…
Project: https://github.com/golang/go/issues/56152

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:github.com/prometheus/client_golang**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/github.com\prometheus\client_golang@v1.0.0]
Non Vulnerable Version	[v1.11.1, v1.12.0, v1.12.1, v1.12.2, v1.13.0, v1.13.1, v1.14.0, v1.15.0, v1.15.1, v1.16.0, v1.17.0, v1.18.0, v1.19.0, v1.19.1, v1.20.0, v1.20.1, v1.20.2, v1.20.3, v1.20.4, v1.20.5, v1.21.0, v1.21.0-rc.0, v1.21.1, v1.22.0-rc.0]

CVE-2022-21698

Vulnerability

Issue

CVE-2022-21698

Severity

CVE CVSS 3: 7.5
CVE CVSS 2.0: 5.0
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 400, 770

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

client_golang is the instrumentation library for Go applications in Prometheus, and the promhttp package in client_golang provides tooling around HTTP servers and clients. In client_golang prior to version 1.11.1, HTTP server is susceptible to a Denial of Service through unbounded cardinality, and potential memory exhaustion, when handling requests with non-standard HTTP methods. In order to be affected, an instrumented software must use any of `promhttp.InstrumentHandler*` middleware except `RequestsInFlight`; not filter any specific methods (e.g GET) before middleware; pass metric with `method` label name to our middleware; and not have any firewall/LB/proxy that filters away requests with unknown `method`. client_golang version 1.11.1 contains a patch for this issue. Several workarounds are available, including removing the `method` label name from counter/gauge used in the InstrumentHandler; turning off affected promhttp handlers; adding custom middleware before promhttp handler that will sanitize the request method given by Go http.Request; and using a reverse proxy or web application firewall, configured to only allow a limited set of methods.

Explanation

The `github.com/prometheus/client_golang` package is vulnerable to a Denial of Service (DoS) attack. The `labels()` and `sanitizeMethod()` functions in the `instrument_server.go` file fail to

sufficiently sanitize non-standard HTTP methods when processing requests. A remote attacker can exploit this vulnerability to cause the application to consume all available resources by issuing requests leveraging non-standard HTTP methods to any affected endpoint(s).

Advisory Deviation Notice: The Sonatype security research team discovered that this vulnerability was introduced in version v0.9.0-pre1 and therefore does not affect all versions prior to v1.11.1 as stated in the advisory.

Detection

The application is vulnerable by using this component if the application:

- [Uses] any of `promhttp.InstrumentHandler*` middleware except RequestsInFlight.
- [Does] not filter any specific methods (e.g GET) before middleware.
- [Passes] metric with method label name to our middleware.
- [Does not] have any firewall/LB/proxy that filters away requests with unknown method.

Reference: https://github.com/prometheus/client_golang/security/advisories/GHSA-cg3q-j54f-5p7p

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Alternatively, if upgrading is not a viable option, the project has provided the following workaround(s):

- Remove method label name from counter/gauge you use in the `InstrumentHandler`.
- Turn off affected `promhttp` handlers.
- Add custom middleware before `promhttp` handler that will sanitize the request method given by Go `http.Request`.
- Use a reverse proxy or web application firewall, configured to only allow a limited set of methods.

Reference: https://github.com/prometheus/client_golang/security/advisories/GHSA-cg3q-j54f-5p7p

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/prometheus/client_golang/pull/962
Project: https://github.com/prometheus/client_golang/pull/987
Project: https://github.com/prometheus/client_golang/security/advisor…
Third Party: https://bugzilla.redhat.com/show_bug.cgi?id=2045880

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE-2023-45142

Vulnerability

Issue

CVE-2023-45142

Severity

CVE CVSS 3: 7.5
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 770

Source

National Vulnerability Database

Categories

Data
Operational

Description

Description from CVE

OpenTelemetry-Go Contrib is a collection of third-party packages for OpenTelemetry-Go. A handler wrapper out of the box adds labels `http.user_agent` and `http.method` that have unbound cardinality. It leads to the server's potential memory exhaustion when many malicious requests are sent to it. HTTP header User-Agent or HTTP method for requests can be easily set by an attacker to be random and long. The library internally uses `httpconv.ServerRequest` that records every value for HTTP `method` and `User-Agent`. In order to be affected, a program has to use the `otelhttp.NewHandler` wrapper and not filter any unknown HTTP methods or User agents on the level of CDN, LB, previous middleware, etc. Version 0.44.0 fixed this issue when the values collected for attribute `http.request.method` were changed to be restricted to a set of well-known values and other high cardinality attributes were removed. As a workaround to stop being affected, `otelhttp.WithFilter()` can be used, but it requires manual careful configuration to not log certain requests entirely. For convenience and safe usage of this library, it should by default mark with the label `unknown` non-standard HTTP methods and User agents to show that such requests were made but do not increase cardinality. In case someone wants to stay with the current behavior, library API should allow to enable it.

Explanation

The `github.com/prometheus/client_golang` package is vulnerable to Denial of Service (DoS) attacks. The `sanitizeMethod()` function in `instrument_server.go` mishandles unknown request methods. A remote attacker can exploit this vulnerability with requests leveraging arbitrary HTTP methods in order to cause the `InstrumentHandler` to create an unbounded number of partitions and potentially exhaust all of the application's available memory.

Detection

The application is vulnerable by using this component if it meets the following criteria provided by the project:

- The application uses any of `promhttp.InstrumentHandler*` middleware except `RequestsInFlight`.
- The application does not filter any specific methods (e.g `GET`) before the `InstrumentHandler` middleware.
- The application passes metrics with `method` label name to the `InstrumentHandler` middleware.
- The application does not have any firewall/LB/proxy that filters away requests with unknown methods.

Reference: https://github.com/advisories/GHSA-cg3q-j54f-5p7p

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Alternatively, if upgrading is not a viable option, the project has provided the following mitigation strategy:

- Remove `method` label name from counter/gauge you use in the `InstrumentHandler`.
- Turn off affected `promhttp` handlers.
- Add custom middleware before `promhttp` handler that will sanitize the request method given by Go `http.Request`.
- Use a reverse proxy or web application firewall, configured to only allow a limited set of methods.

Reference: https://github.com/advisories/GHSA-cg3q-j54f-5p7p

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Third Party: https://github.com/advisories/GHSA-cg3q-j54f-5p7p

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:github.com/yuin/goldmark**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/github.com\yuin\goldmark@v1.1.27]
Non Vulnerable Version	[v1.2.0, v1.2.1, v1.3.0, v1.3.1, v1.3.2, v1.3.3, v1.3.4, v1.3.5, v1.3.6, v1.3.7, v1.3.8, v1.3.9, v1.4.0, v1.4.1, v1.4.10, v1.4.11, v1.4.12, v1.4.13, v1.4.14, v1.4.15, v1.4.2, v1.4.3, v1.4.4, v1.4.5, v1.4.6, v1.4.7, v1.4.8, v1.4.9, v1.5.0, v1.5.1, v1.5.2, v1.5.3, v1.5.4, v1.5.5, v1.5.6, v1.6.0, v1.7.0, v1.7.1, v1.7.2, v1.7.3, v1.7.4, v1.7.5, v1.7.6, v1.7.7, v1.7.8]
sonatype-2020-0584	## Vulnerability **Issue** sonatype-2020-0584 **Severity** Sonatype CVSS 3: 6.1 **Weakness** Sonatype CWE: 79 **Source** Sonatype Data Research **Categories** Data ## Description **Explanation** The `goldmark` package is vulnerable to Cross-Site Scripting (XSS). The `renderImage` function in `html.go` does not properly escape user input for the `alt` HTML attribute. An attacker can exploit this by inputting malicious HTML code as the text for the `alt` attribute that would then be parsed and executed. **Detection** The application is vulnerable by using this component. **Recommendation** We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control. **Advisories** Project: https://github.com/yuin/goldmark/pull/145 **CVSS Details** Sonatype CVSS 3: 6.1 CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
Component Info	**golang:github.com/gogo/protobuf**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/github.com\gogo\protobuf@v1.1.1]
Non Vulnerable Version	[v1.3.2]

CVE-2021-3121

Vulnerability

Issue

CVE-2021-3121

Severity

CVE CVSS 3: 8.6
CVE CVSS 2.0: 7.5
Sonatype CVSS 3: 8.6

Weakness

CVE CWE: 129

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

An issue was discovered in GoGo Protobuf before 1.3.2. plugin/unmarshal/unmarshal.go lacks certain index validation, aka the "skippy peanut butter" issue.

Explanation

The `protobuf` package is vulnerable due to Improper Validation of Array Index. The `field()` and `Generate()` functions in the `unmarshal.go` file do not properly validate indices when unmarshalling certain protobuf objects. A remote attacker can exploit this behavior by supplying specially-crafted protobuf messages which, when unmarshalled, would result in Denial of Service (DoS) or other attacks.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Workaround:

Disabling support for protobuf messages may be one possible mitigation for any affected product.

Also, graceful panic handling in message handlers mitigates the bug.

Reference: https://github.com/kubernetes/kubernetes/issues/101435

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Third Party: https://access.redhat.com/security/cve/cve-2021-3121

CVSS Details

CVE CVSS 3: 8.6
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:H

Component Info

golang:google.golang.org/grpc

Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\grpc@v1.29.1]
Non Vulnerable Version	[v1.56.3, v1.57.0-dev, v1.57.1, v1.57.2, v1.58.0-dev, v1.58.3, v1.59.0, v1.59.0-dev, v1.60.0, v1.60.0-dev, v1.60.1, v1.61.0, v1.61.0-dev, v1.61.1, v1.61.2, v1.62.0, v1.62.1, v1.62.2, v1.63.0, v1.63.1, v1.63.2, v1.63.3, v1.64.1, v1.65.0, v1.65.1, v1.66.0, v1.66.1, v1.66.2, v1.66.3, v1.67.0, v1.67.0-dev, v1.67.1, v1.67.2, v1.67.3, v1.68.0, v1.68.0-dev, v1.68.1, v1.68.2, v1.69.0, v1.69.0-dev, v1.69.2, v1.69.4, v1.70.0, v1.70.0-dev, v1.71.0, v1.71.0-dev, v1.71.1, v1.72.0-dev, v1.73.0-dev]
sonatype-2023-4386	

Vulnerability

Issue

sonatype-2023-4386

Severity

Sonatype CVSS 3: 7.5

Weakness

Sonatype CWE: 400

Source

Sonatype Data Research

Categories

Data
Operational

Description

Explanation

The `grpc` package is vulnerable to a Denial of Service (DoS) attack. The `serveStreams()` function in the `server.go` file does not set a limit for the maximum concurrent HTTP/2 streams. A remote attacker can exploit this vulnerability by submitting a large number of reset requests in rapid succession in order to establish an unbounded number of concurrent connections. These requests may transition from an open to a closed state before the server processes their cancellation. As a result, this will exhaust the connection pool of affected applications and ultimately result in a DoS condition for their users.

Note: This vulnerability exists under the umbrella of CVE-2023-44487.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/grpc/grpc-go/pull/6703
Third Party: https://blog.cloudflare.com/technical-breakdown-http2-rapid-…
Third Party: https://blog.cloudflare.com/zero-day-rapid-reset-http2-recor…
Third Party: https://github.com/advisories/GHSA-m425-mq94-257g

CVSS Details

Sonatype CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:github.com/mailru/easyjson**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/github.com\mailru\easyjson@v0.7.1]
Non Vulnerable Version	[v0.7.7, v0.8.0, v0.9.0]
sonatype-2023-3024	

Vulnerability

Issue

sonatype-2023-3024

Severity

Sonatype CVSS 3: 5.3

Weakness

Sonatype CWE: 20

Source

Sonatype Data Research

Categories

Data

Description

Explanation

The `github.com/mailru/easyjson` module is vulnerable to Denial of Service (DoS) attacks. The `Bytes()` function in the `lexer.go` file fails to unmarshall base64 encoded string objects containing a forward slash, `(/)`, resulting in an uncontrolled exception. An attacker with influence over the contents parsed by the lexer can exploit this vulnerability by submitting a base64 encoded string that leverages the aforementioned issue. This will cause a crash and ultimately a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/mailru/easyjson/pull/328

CVSS Details

Sonatype CVSS 3: 5.3
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L

Component Info	**golang:google.golang.org/protobuf**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\protobuf@v1.21.0]

Non Vulnerable Version	[v1.33.0, v1.34.0, v1.34.1, v1.34.2, v1.35.0, v1.35.1, v1.35.2, v1.36.0, v1.36.1, v1.36.2, v1.36.3, v1.36.4, v1.36.5, v1.36.6]
CVE-2024-24786	**Vulnerability** **Issue** CVE-2024-24786 **Severity** Sonatype CVSS 3: 7.5 **Weakness** Sonatype CWE: 835 **Source** National Vulnerability Database **Categories** Data Functional **Description** **Description from CVE** The protojson.Unmarshal function can enter an infinite loop when unmarshaling certain forms of invalid JSON. This condition can occur when unmarshaling into a message which contains a google.protobuf.Any value, or when the UnmarshalOptions.DiscardUnknown option is set. **Explanation** The `protobuf` package is vulnerable to Denial of Service (DoS) attacks due to an Infinite Loop. The `Read()` function within the `decode.go` file fails to account for JSON objects containing keys that lack values (i.e. `{"":}`). A remote attacker can exploit this behavior by supplying malformed JSON to cause affected applications to enter into an infinite loop, ultimately resulting in a DoS condition. **Detection** The application is vulnerable by using this component. **Recommendation** We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control. **Advisories** Project: https://pkg.go.dev/vuln/GO-2024-2611 **CVSS Details** Sonatype CVSS 3: 7.5 CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
Component Info	**golang:golang.org/x/text**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/golang.org\x\text@v0.3.3]

Non Vulnerable Version	[v0.10.0, v0.11.0, v0.12.0, v0.13.0, v0.14.0, v0.15.0, v0.16.0, v0.17.0, v0.18.0, v0.19.0, v0.20.0, v0.21.0, v0.22.0, v0.23.0, v0.24.0, v0.3.8, v0.4.0, v0.5.0, v0.6.0, v0.7.0, v0.8.0, v0.9.0]
CVE-2021-38561	

Vulnerability

Issue

CVE-2021-38561

Severity

CVE CVSS 3: 7.5
Sonatype CVSS 3: 4.3

Weakness

CVE CWE: 125

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

golang.org/x/text/language in golang.org/x/text before 0.3.7 can panic with an out-of-bounds read during BCP 47 language tag parsing. Index calculation is mishandled. If parsing untrusted user input, this can be used as a vector for a denial-of-service attack.

Explanation

The `golang.org/x/text` package is vulnerable due to an Out-of-bounds Read. The files and functions listed below do not properly handle index calculations when parsing formatted language tags. A remote attacker can exploit this behavior by supplying a specially-crafted language tag to trigger a panic, causing an application crash and ultimately a Denial of Service (DoS) condition.

Vulnerable File(s) and Function(s):

internal/language/language.go

- ParseExtension()
- ParseBase()
- ParseScript()
- ParseRegion()
- ParseVariant()

internal/language/parse.go

- Parse()

language/parse.go

- Parse()
- Compose()
- ParseAcceptLanguage()

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Third Party: https://golangissues.com/issues/1154451
Third Party: https://osv.dev/vulnerability/GO-2021-0113

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE-2022-32149

Vulnerability

Issue

CVE-2022-32149

Severity

CVE CVSS 3: 7.5
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 772

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

An attacker may cause a denial of service by crafting an Accept-Language header which ParseAcceptLanguage will take significant time to parse.

Explanation

The `golang.org/x/text` package is vulnerable to a Denial of Service (DoS) attack. The `ParseAcceptLanguage()` function in the `parse.go` file fails to account for strings containing large numbers of – characters when parsing `Accept-Language` headers. A remote attacker can exploit this vulnerability by submitting a request that leverages the aforementioned header. This will cause the application to expend excessive resources to process the request, resulting in a DoS condition.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Evidence: https://github.com/golang/vulndb/commit/c4e6da635d3b5a5ce387…
Project: https://github.com/golang/go/issues/56152

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:google.golang.org/grpc**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/google.golang.org\grpc@v1.19.0]

Non Vulnerable Version	[v1.56.3, v1.57.0-dev, v1.57.1, v1.57.2, v1.58.0-dev, v1.58.3, v1.59.0, v1.59.0-dev, v1.60.0, v1.60.0-dev, v1.60.1, v1.61.0, v1.61.0-dev, v1.61.1, v1.61.2, v1.62.0, v1.62.1, v1.62.2, v1.63.0, v1.63.1, v1.63.2, v1.63.3, v1.64.1, v1.65.0, v1.65.1, v1.66.0, v1.66.1, v1.66.2, v1.66.3, v1.67.0, v1.67.0-dev, v1.67.1, v1.67.2, v1.67.3, v1.68.0, v1.68.0-dev, v1.68.1, v1.68.2, v1.69.0, v1.69.0-dev, v1.69.2, v1.69.4, v1.70.0, v1.70.0-dev, v1.71.0, v1.71.0-dev, v1.71.1, v1.72.0-dev, v1.73.0-dev]
sonatype-2023-4386	## Vulnerability **Issue** sonatype-2023-4386 **Severity** Sonatype CVSS 3: 7.5 **Weakness** Sonatype CWE: 400 **Source** Sonatype Data Research **Categories** Data Operational ## Description **Explanation** The `grpc` package is vulnerable to a Denial of Service (DoS) attack. The `serveStreams()` function in the `server.go` file does not set a limit for the maximum concurrent HTTP/2 streams. A remote attacker can exploit this vulnerability by submitting a large number of reset requests in rapid succession in order to establish an unbounded number of concurrent connections. These requests may transition from an open to a closed state before the server processes their cancellation. As a result, this will exhaust the connection pool of affected applications and ultimately result in a DoS condition for their users. *Note*: This vulnerability exists under the umbrella of CVE-2023-44487. **Detection** The application is vulnerable by using this component. **Recommendation** We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control. **Advisories** Project: https://github.com/grpc/grpc-go/pull/6703 Third Party: https://blog.cloudflare.com/technical-breakdown-http2-rapid-… Third Party: https://blog.cloudflare.com/zero-day-rapid-reset-http2-recor… Third Party: https://github.com/advisories/GHSA-m425-mq94-257g **CVSS Details** Sonatype CVSS 3: 7.5 CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
Component Info	**golang:gopkg.in/yaml.v2**

Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/gopkg.in\yaml.v2@v2.2.2]
Non Vulnerable Version	[v2.2.4, v2.2.5, v2.2.6, v2.2.7, v2.2.8, v2.3.0, v2.4.0]
CVE-2022-3064	**Vulnerability** **Issue** CVE-2022-3064 **Severity** CVE CVSS 3: 7.5 Sonatype CVSS 3: 7.5 **Weakness** CVE CWE: 400 **Source** National Vulnerability Database **Categories** Data **Description** **Description from CVE** Parsing malicious or large YAML documents can consume excessive amounts of CPU or memory. **Explanation** This issue has undergone the Sonatype Fast-Track process. For more information, please see the Sonatype Knowledge Base Guide. **Advisories** Evidence: https://github.com/go-yaml/yaml/releases/tag/v2.2.4 Project: https://github.com/go-yaml/yaml/pull/515 Third Party: https://pkg.go.dev/vuln/GO-2022-0956 **CVSS Details** CVE CVSS 3: 7.5 CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
sonatype-2019-0900	**Vulnerability** **Issue** sonatype-2019-0900 **Severity** Sonatype CVSS 3: 6.5 **Weakness** Sonatype CWE: 400 **Source** Sonatype Data Research

Categories

Data

Description

Explanation

The `gopkg.in/yaml` package is vulnerable to Denial of Service (DoS) attacks. The `unmarshal()` function in the `decode.go` file fails to control the number of aliases when unmarshalling values in a YAML document, this allows malicious actors to excessively consume resources by nesting multiple aliases. A remote attacker can exploit this vulnerability by submitting a specially-crafted YAML document to cause a DoS condition in the underlying system.

Detection

The application is vulnerable by using this component.

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue.

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Evidence: https://github.com/go-yaml/yaml/releases/tag/v2.2.3
Project: https://github.com/go-yaml/yaml/pull/375
Project: https://github.com/go-yaml/yaml/pull/555
Project: https://pkg.go.dev/vuln/GO-2021-0061
Third Party: https://deps.dev/advisory/OSV/GO-2021-0061

CVSS Details

Sonatype CVSS 3: 6.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H

Component Info	**golang:github.com/prometheus/client_golang**
Path	[85/heplify-server-1.59.8.zip/heplify-server-1.59.8/go.sum/github.com\prometheus\client_golang@v0.9.1]
Non Vulnerable Version	[No clean version available]

CVE-2022-21698

Vulnerability

Issue

CVE-2022-21698

Severity

CVE CVSS 3: 7.5
CVE CVSS 2.0: 5.0
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 400, 770

Source

National Vulnerability Database

Categories

Data

Description

Description from CVE

client_golang is the instrumentation library for Go applications in Prometheus, and the promhttp package in client_golang provides tooling around HTTP servers and clients. In client_golang prior to version 1.11.1, HTTP server is susceptible to a Denial of Service through unbounded cardinality, and potential memory exhaustion, when handling requests with non-standard HTTP methods. In order to be affected, an instrumented software must use any of `promhttp.InstrumentHandler*` middleware except `RequestsInFlight`; not filter any specific methods (e.g GET) before middleware; pass metric with `method` label name to our middleware; and not have any firewall/LB/proxy that filters away requests with unknown `method`. client_golang version 1.11.1 contains a patch for this issue. Several workarounds are available, including removing the `method` label name from counter/gauge used in the InstrumentHandler; turning off affected promhttp handlers; adding custom middleware before promhttp handler that will sanitize the request method given by Go http.Request; and using a reverse proxy or web application firewall, configured to only allow a limited set of methods.

Explanation

The `github.com/prometheus/client_golang` package is vulnerable to a Denial of Service (DoS) attack. The `labels()` and `sanitizeMethod()` functions in the `instrument_server.go` file fail to sufficiently sanitize non-standard HTTP methods when processing requests. A remote attacker can exploit this vulnerability to cause the application to consume all available resources by issuing requests leveraging non-standard HTTP methods to any affected endpoint(s).

Advisory Deviation Notice: The Sonatype security research team discovered that this vulnerability was introduced in version v0.9.0-pre1 and therefore does not affect all versions prior to v1.11.1 as stated in the advisory.

Detection

The application is vulnerable by using this component if the application:

- [Uses] any of `promhttp.InstrumentHandler*` middleware except RequestsInFlight.
- [Does] not filter any specific methods (e.g `GET`) before middleware.
- [Passes] metric with method label name to our middleware.
- [Does not] have any firewall/LB/proxy that filters away requests with unknown method.

Reference: https://github.com/prometheus/client_golang/security/advisories/GHSA-cg3q-j54f-5p7p

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Alternatively, if upgrading is not a viable option, the project has provided the following workaround(s):

- Remove method label name from counter/gauge you use in the `InstrumentHandler`.
- Turn off affected `promhttp` handlers.
- Add custom middleware before `promhttp` handler that will sanitize the request method given by Go `http.Request`.
- Use a reverse proxy or web application firewall, configured to only allow a limited set of methods.

Reference: https://github.com/prometheus/client_golang/security/advisories/GHSA-cg3q-j54f-5p7p

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Project: https://github.com/prometheus/client_golang/pull/962
Project: https://github.com/prometheus/client_golang/pull/987
Project: https://github.com/prometheus/client_golang/security/advisor…
Third Party: https://bugzilla.redhat.com/show_bug.cgi?id=2045880

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CVE-2023-45142

Vulnerability

Issue

CVE-2023-45142

Severity

CVE CVSS 3: 7.5
Sonatype CVSS 3: 7.5

Weakness

CVE CWE: 770

Source

National Vulnerability Database

Categories

Data
Operational

Description

Description from CVE

OpenTelemetry-Go Contrib is a collection of third-party packages for OpenTelemetry-Go. A handler wrapper out of the box adds labels `http.user_agent` and `http.method` that have unbound cardinality. It leads to the server's potential memory exhaustion when many malicious requests are sent to it. HTTP header User-Agent or HTTP method for requests can be easily set by an attacker to be random and long. The library internally uses `httpconv.ServerRequest` that records every value for HTTP `method` and `User-Agent`. In order to be affected, a program has to use the `otelhttp.NewHandler` wrapper and not filter any unknown HTTP methods or User agents on the level of CDN, LB, previous middleware, etc. Version 0.44.0 fixed this issue when the values collected for attribute `http.request.method` were changed to be restricted to a set of well-known values and other high cardinality attributes were removed. As a workaround to stop being affected, `otelhttp.WithFilter()` can be used, but it requires manual careful configuration to not log certain requests entirely. For convenience and safe usage of this library, it should by default mark with the label `unknown` non-standard HTTP methods and User agents to show that such requests were made but do not increase cardinality. In case someone wants to stay with the current behavior, library API should allow to enable it.

Explanation

The `github.com/prometheus/client_golang` package is vulnerable to Denial of Service (DoS) attacks. The `sanitizeMethod()` function in `instrument_server.go` mishandles unknown request methods. A remote attacker can exploit this vulnerability with requests leveraging arbitrary HTTP methods in order to cause the `InstrumentHandler` to create an unbounded number of partitions and potentially exhaust all of the application's available memory.

Detection

The application is vulnerable by using this component if it meets the following criteria provided by the project:

- The application uses any of `promhttp.InstrumentHandler*` middleware except `RequestsInFlight`.
- The application does not filter any specific methods (e.g GET) before the `InstrumentHandler` middleware.
- The application passes metrics with `method` label name to the `InstrumentHandler` middleware.
- The application does not have any firewall/LB/proxy that filters away requests with unknown methods.

Reference: https://github.com/advisories/GHSA-cg3q-j54f-5p7p

Recommendation

We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Alternatively, if upgrading is not a viable option, the project has provided the following mitigation strategy:

- Remove `method` label name from counter/gauge you use in the `InstrumentHandler`.
- Turn off affected `promhttp` handlers.
- Add custom middleware before `promhttp` handler that will sanitize the request method given by Go `http.Request`.
- Use a reverse proxy or web application firewall, configured to only allow a limited set of methods.

Reference: https://github.com/advisories/GHSA-cg3q-j54f-5p7p

Note: If this component is included as a bundled/transitive dependency of another component, there may not be an upgrade path. In this instance, we recommend contacting the maintainers who included the vulnerable package. Alternatively, we recommend investigating alternative components or a potential mitigating control.

Advisories

Third Party: https://github.com/advisories/GHSA-cg3q-j54f-5p7p

CVSS Details

CVE CVSS 3: 7.5
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H