Polyverse Boost Source Analysis Details: ./gomerr/dependency.go

Date Generated: Thursday, September 7, 2023 at 7:15:23 PM PDT

Boost Architectural Quick Summary Security Report

Last Updated: Friday, September 8, 2023 at 1:52:03 PM PDT

Executive Report:

1. **Architectural Impact**: The analysis of this file has not revealed any severe issues.
2. **Risk Analysis**: The analysis of this file has not revealed any severe issues.
3. **Potential Customer Impact**: Based on the analysis, there are no severe issues that could potentially impact customers.
4. **Performance Issues**: Our analysis did not identify any explicit performance issues in the file.
5. **Risk Assessment**: Based on the current analysis of this file, no severe issues have been found. However, this doesn't guarantee that the file is risk-free.

Highlights:

- No severe issues were identified in the current analysis of this file.

Boost Architectural Quick Summary Performance Report

Last Updated: Friday, September 8, 2023 at 1:52:09 PM PDT

Executive Report:

1. **Architectural Impact**: The analysis of this file has not revealed any severe issues.
2. **Risk Analysis**: The analysis of this file has not revealed any severe issues.
3. **Potential Customer Impact**: Based on the analysis, there are no severe issues that could potentially impact customers.

4. **Performance Issues**: Our analysis did not identify any explicit performance issues in the file.
5. **Risk Assessment**: Based on the current analysis of this file, no severe issues have been found. However, this doesn't guarantee that the file is risk-free.

Highlights:

- No severe issues were identified in the current analysis of this file.

Boost Architectural Quick Summary Compliance Report

Last Updated: Friday, September 8, 2023 at 1:52:53 PM PDT

Executive Report: Software Project Analysis

Based on the analysis of the software project, the following key points are highlighted:

1. **Architectural Impact**: The project is a Go library focusing on constraint handling and validation. It follows Go's idiomatic style and structure for a library, providing a clear separation of concerns. However, the presence of data compliance issues in the `gomerr/dependency.go` file could potentially impact the overall architecture if not addressed.
2. **Risk Analysis**: The project has a single file `gomerr/dependency.go` with multiple data compliance issues. These issues span across GDPR, PCI DSS, HIPAA, and Data Exposure categories. This indicates a high risk in terms of data compliance. The risk is further amplified by the fact that the file in question may contain user data, potentially leading to compliance issues if not properly handled.
3. **Potential Customer Impact**: The presence of data compliance issues, particularly GDPR, could have a significant impact on customers, especially those in regions with strict data protection laws. If personal data is not processed according to GDPR regulations, it could lead to legal complications for the customers using this library.
4. **Overall Health of the Project**: The project consists of a single file, which has been flagged with multiple data compliance issues. This suggests that 100% of the project files have issues of 'Warning' severity. This is a serious concern and indicates that the overall health of the project is poor.

5. **Recommendations**: Immediate action is required to address the data compliance issues identified in the `gomerr/dependency.go` file. This includes ensuring that any personal data included in the 'Request' field is processed according to GDPR regulations. This will not only improve the overall health of the project but also reduce the potential customer impact.

In conclusion, while the project follows good architectural practices, the presence of multiple data compliance issues poses a significant risk. Immediate action is required to address these issues and improve the overall health of the project.

Detailed Analysis

./gomerr/dependency.go :

Programming Language: go

```
package gomerr

type DependencyError struct {
        Gomerr
        Service string
        Request interface{} `gomerr:"include_type"`
}
```

Boost Code Explanation

Last Updated: Thursday, September 7, 2023 at 7:13:03 PM PDT

This is a Go programming language code snippet that is defining a custom error type called `DependencyError`. Go is a statically typed, compiled language that is known for its simplicity and efficiency. It is often used in systems programming and for creating web servers.

Here's a breakdown of the code:

- `package gomerr`: This line declares the package name for this file. All Go files in the same directory should declare the same package name. The name `gomerr` suggests that this file is part of a package that handles errors in a Go application.

- `type DependencyError struct {`: This line declares a new type `DependencyError` which is a struct. In Go, a struct is a composite data type that groups together zero or more values with different types. The struct `DependencyError` is meant to represent an error that occurs due to a dependency.

- `Gomerr` : This line is an embedded field of the `DependencyError` struct. It means that all the methods and fields of the `Gomerr` type are directly accessible on the `DependencyError` type. The `Gomerr` type is not defined in this code snippet, but it is likely an interface or a struct defined elsewhere in the `gomerr` package that provides common error handling functionality.

- `Service string` : This line declares a field `Service` of type `string` inside the `DependencyError` struct. This field probably stores the name of the service where the dependency error occurred.

- `Request interface{}` gomerr:"include_type" : This line declares a field Request of type interface{} inside the DependencyError struct. In Go, interface{} is the empty interface that can hold values of any type. The gomerr:"include_type" part is a struct tag that provides metadata about the Request field. Struct tags in Go are used by packages like encoding/json to control the encoding/decoding of the struct to/from JSON. In this case, it seems that the gomerr package uses the struct tag to include the type of the Request` field when handling the error.

This code does not contain any algorithms, it is a simple type definition.

For more information about error handling in Go, you can refer to the official Go blog post: Error handling and Go. For more information about struct tags, you can refer to the Go blog post: The Go Programming Language Specification - Struct types.

Boost Flow Diagram

Last Updated: Thursday, September 7, 2023 at 7:13:52 PM PDT

Boost Source-Level Security Analysis

Last Updated: Thursday, September 7, 2023 at 7:14:10 PM PDT

No bugs found

Boost Source-Level Performance Analysis

Last Updated: Thursday, September 7, 2023 at 7:14:22 PM PDT

No bugs found

Boost Source-Level Data and Privacy Compliance Analysis

Last Updated: Thursday, September 7, 2023 at 7:15:08 PM PDT

1. **Severity**: 8/10

   Line Number: 1

   Bug Type: GDPR

   Description: The code defines a struct 'DependencyError' with a field 'Request' that may contain user data. This might lead to GDPR compliance issues if personal data is included and not properly handled.

   Solution: Ensure that any personal data included in the 'Request' field is processed according to GDPR regulations. This includes obtaining necessary user consent, ensuring data minimization, and providing data protection by design and by default.

2. **Severity**: 7/10

   Line Number: 1

   Bug Type: PCI DSS

   Description: If the 'Request' field in the 'DependencyError' struct contains payment card data, it may violate PCI DSS compliance. Payment card data should never be logged or unencrypted.

   Solution: Ensure that no payment card data is included in the 'Request' field. If it is necessary to include such data, make sure it is properly encrypted and never logged.

3. **Severity**: 8/10

   Line Number: 1

   Bug Type: HIPAA

   Description: The 'Request' field in the 'DependencyError' struct may contain health information, potentially violating HIPAA compliance if not properly protected.

   Solution: Ensure that any health information included in the 'Request' field is properly protected according to HIPAA regulations. This includes ensuring necessary safeguards are in place to protect the data, and that the data is only used and disclosed in ways that are compliant with HIPAA.

./gomerr/dependency.go line 8:

Programming Language: go

```
func Dependency(service string, request interface{}) *DependencyError {
        return Build(new(DependencyError), service, request).(*DependencyError)
}
```

Boost Code Explanation

Last Updated: Thursday, September 7, 2023 at 7:13:37 PM PDT

This is a function written in Go (Golang) that creates and returns a new instance of a `DependencyError`.

Here's a breakdown of the code:

- The function `Dependency` takes two parameters: `service` which is a string, and `request` which is an interface. An interface in Go is a type that defines a set of methods but does not implement them. In this case, `request` can be any type that implements the interface.
- Inside the function, it calls another function `Build` with three arguments: a new instance of `DependencyError`, `service`, and `request`. The `new` keyword in Go creates a new instance of a type. Here it creates a new instance of `DependencyError`.
- The `Build` function presumably constructs an instance of a `DependencyError` with the provided `service` and `request`, and then returns it. The specifics of what `Build` does would depend on its own implementation, which is not provided here.
- The return value of `Build` is type asserted to `*DependencyError` using `.(*DependencyError)`. This means the function is asserting that the returned value is a pointer to an instance of `DependencyError`. If the returned value is not of this type, the program will panic at runtime.

The algorithm used here is quite straightforward. It's essentially a factory function that creates and initializes a new `DependencyError` given a `service` and a `request`.

The function appears to be part of a larger error handling system, possibly for a microservices architecture where dependencies between services may cause errors. This function could be used to create a `DependencyError` whenever a service fails due to a dependency issue.

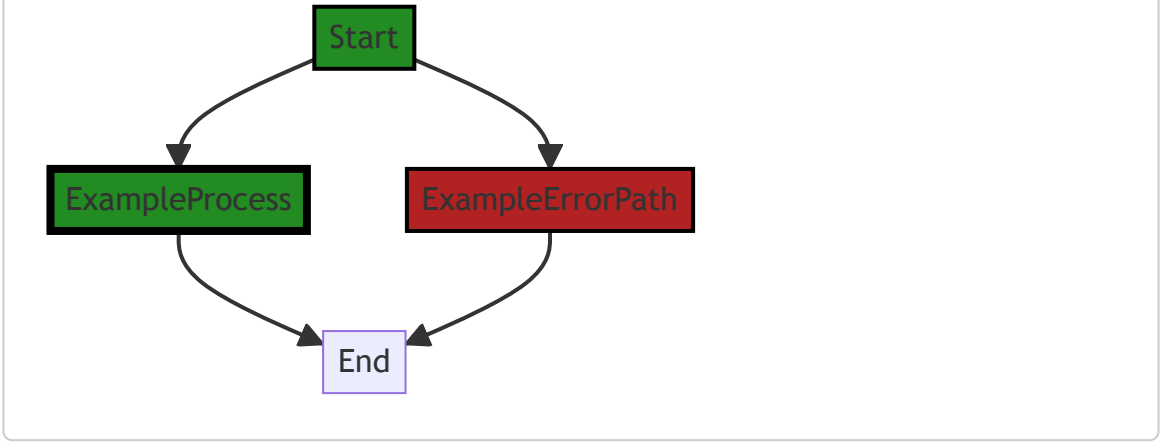
For more information on error handling in Go, you can refer to the following resources:

- Error handling and Go

- Go by Example: Errors

Boost Flow Diagram

Last Updated: Thursday, September 7, 2023 at 7:13:57 PM PDT



The code snippet provided does not have any control flow. It is a simple function definition.

Boost Source-Level Security Analysis

Last Updated: Thursday, September 7, 2023 at 7:14:13 PM PDT

No bugs found

Boost Source-Level Performance Analysis

Last Updated: Thursday, September 7, 2023 at 7:14:26 PM PDT

No bugs found

Boost Source-Level Data and Privacy Compliance Analysis

Last Updated: Thursday, September 7, 2023 at 7:15:23 PM PDT

1. **Severity**: 7/10

   Line Number: 14

   Bug Type: Data Exposure

   Description: The function Dependency() could potentially expose sensitive data. The 'request' parameter, which could contain sensitive data, is passed to the function 'Build()', where it could be logged or otherwise exposed.

   Solution: Ensure that sensitive data in 'request' is properly sanitized or encrypted before it is passed to the 'Build()' function. Implement strict access controls and logging policies to prevent unauthorized access to logs. Consider using a data masking or tokenization solution to replace sensitive data with non-sensitive equivalents.