Slice Memory Allocation with Excessive Size Value in github.com/gagliardetto/binary

Moderate **gagliardetto** published **GHSA-4p6f-m4f9-ch88** on Sep 1

Package

github.com/gagliardetto/binary (Go)

Affected versions	Patched versions
< v0.7.1	v0.7.1

Description

Impact

> *What kind of vulnerability is it? Who is impacted?*

The vulnerability is a memory allocation vulnerability that can be exploited to allocate slices in memory with (arbitrary) excessive size value, which can either exhaust available memory or crash the whole program.

When using `github.com/gagliardetto/binary` to parse unchecked (or wrong type of) data from untrusted sources of input (e.g. the blockchain) into slices, it's possible to allocate memory with excessive size.

When `dec.Decode(&val)` method is used to parse data into a structure that is or contains slices of values, the length of the slice was previously read directly from the data itself without any checks on the size of it, and then a slice was allocated. This could lead to an overflow and an allocation of memory with excessive size value.

Example:

```
package main

import (
	"github.com/gagliardetto/binary" // any version before v0.7.1 is vulnerable
```

```
        "log"
)

type MyStruct struct {
        Field1 []byte // field is a slice (could be a slice of any type)
}

func main() {
        // Let's assume that the data is coming from the blockchain:
        data := []byte{0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0

        var val MyStruct
        // - To determine the size of the val.Field1 slice, the decoder will read the length
        //   of the slice from the data itself without any checks on the size of it.
        //
        // - This could lead to an allocation of memory with excessive size value.
        //   Which means: []byte{0x01, 0x02, 0x03, 0x04} will be read as the length
        //   of the slice (= 67305985) and then an allocation of memory with 67305985 bytes will
        //
        dec := binary.NewBorshDecoder(data)
        err := dec.Decode(&val)  // or binary.UnmarshalBorsh(&val, data) or binary.UnmarshalBin(
        if err != nil {
                log.Fatal(err)
        }
}
```

Patches

> *Has the problem been patched? What versions should users upgrade to?*

The vulnerability has been patched in `github.com/gagliardetto/binary` `v0.7.1`

Users should upgrade to `v0.7.1` or higher.

To upgrade to `v0.7.1` or higher, run:

```
go get github.com/gagliardetto/binary@v0.7.1

# or

go get github.com/gagliardetto/binary@latest
```

Workarounds

> *Is there a way for users to fix or remediate the vulnerability without upgrading?*

A workaround is not to rely on the `dec.Decode(&val)` function to parse the data, but to use a custom `UnmarshalWithDecoder()` method that reads and checks the length of any slice.

References

> *Are there any links users can visit to find out more?*

For more information

If you have any questions or comments about this advisory:

- Open an issue in github.com/gagliardetto/binary
- DM me on twitter

Severity

Moderate

CVE ID

CVE-2022-36078

Weaknesses

CWE-789