**Problem**

<https://github.com/satori/go.uuid/issues/73>

A flaw was found in github.com/satori/go.uuid in versions from commit 0ef6afb2f6cdd6cdaeee3885a95099c63f18fc8c to d91630c8510268e75203009fe7daf2b8e1d60c45. Due to insecure randomness in the g.rand.Read function the generated UUIDs are predictable for an attacker.

<https://github.com/satori/go.uuid/commit/d91630c8510268e75203009fe7daf2b8e1d60c45>

**generator\_test.go**

**generator.go**

// NewV4 returns random generated UUID.

func (g \*rfc4122Generator) NewV4() (UUID, error) {

u := UUID{}

if \_, err := g.rand.**Read**(u[:]); err != nil {

return Nil, err

}

u.SetVersion(V4)

<https://pkg.go.dev/io#Reader>

type Reader

type Reader interface {

Read(p []byte) (n int, err error)

}

Read reads up to len(p) bytes into p. It returns the number of bytes read (0 <= n <= len(p)) and any error encountered. Even if Read returns n < len(p), it may use all of p as scratch space during the call. **If some data is available but not len(p) bytes, Read conventionally returns what is available instead of waiting for more**.

Callers should always process the n > 0 bytes returned before considering the error err. Doing so correctly handles I/O errors that happen after reading some bytes and also both of the allowed EOF behaviors.

Implementations of Read are discouraged from returning a zero-byte count with a nil error, except when len(p) == 0. Callers should treat a return of 0 and nil as indicating that nothing happened; in particular it does not indicate EOF.

Implementations must not retain p.

**Fix**

**generator.go**

// NewV4 returns random generated UUID.

func (g \*rfc4122Generator) NewV4() (UUID, error) {

u := UUID{}

if \_, err := io.**ReadFull**(g.rand, u[:]); err != nil {

return Nil, err

}

u.SetVersion(V4)

**generator\_test.go**

import "bytes"

import "testing/iotest"

func (s \*genTestSuite) TestNewV4PartialRead(c \*C) {

g := &rfc4122Generator{

epochFunc: time.Now,

hwAddrFunc: defaultHWAddrFunc,

rand: iotest.OneByteReader(rand.Reader),

}

u1, err := g.NewV4()

zeros := bytes.Count(u1.Bytes(), []byte{0})

mostlyZeros := zeros >= 10

c.Assert(err, IsNil)

c.Assert(mostlyZeros, Equals, false)

}

**func ReadFull**

func ReadFull(r Reader, buf []byte) (n int, err error)

ReadFull reads exactly len(buf) bytes from r into buf. It returns the number of bytes copied and an error if fewer bytes were read. The error is EOF only if no bytes were read. If an EOF happens after reading some but not all the bytes, ReadFull returns ErrUnexpectedEOF. On return, n == len(buf) if and only if err == nil. If r returns an error having read at least len(buf) bytes, the error is dropped.