RUSTSEC-2021-0022

Use-after-free in `subscript_next` and `subscript_prev` wrappers

Reported	February 9, 2021
Issued	February 15, 2021 (last modified: October 19, 2021)
Package	yottadb (crates.io)
Type	Vulnerability
Categories	memory-corruption
Keywords	#use-after-free
Aliases	CVE-2021-27377
Details	https://gitlab.com/YottaDB/Lang/YDBRust/-/issues/40
CVSS Score	9.8 CRITICAL

CVSS Details

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Patched	>=1.2.0

Affected Functions	Version
yottadb::Key::sub_next_self_st	<1.2.0
yottadb::Key::sub_prev_self_st	<1.2.0
yottadb::KeyContext::sub_next_self_st	<1.2.0
yottadb::KeyContext::sub_prev_self_st	<1.2.0

Description

Affected versions of this crate had an unsound implementation which could pass a pointer to freed memory to `ydb_subscript_next_st` and `ydb_subscript_prev_st` if the variable and subscripts did not have enough memory allocated on the first call to hold the next variable in the database.

For example, the following code had undefined behavior:

```
let mut key = Key::variable(String::from("a"));
Key::variable("averylongkeywithlotsofletters")
    .set_st(YDB_NOTTP, Vec::new(), b"some val")
    .unwrap();
key.sub_next_self_st(YDB_NOTTP, Vec::new()).unwrap();
```

`yottadb` has no reverse-dependencies on crates.io and there are no known instances of this API being used incorrectly in practice. The fix is backwards compatible.

The flaw was corrected by recalculating the pointer each time it was reallocated.