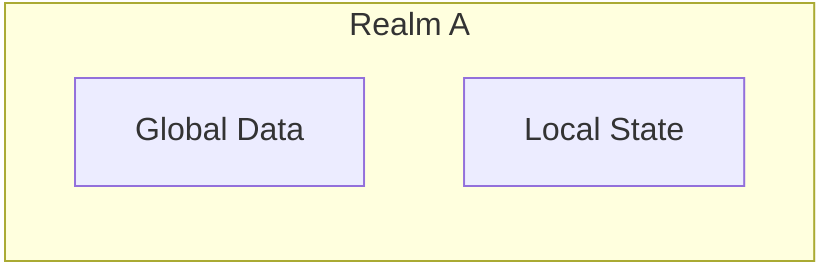
Interrealm in Gno 🌐

A deep dive into Gno’s realm system and rules

What is a Realm?

- A **Realm** is an isolated mutable state space within the GnoVM
- Each realm:
  - Owns its global variables (storage)
  - Controls mutation access through realm context
  - Has a dedicated coin address

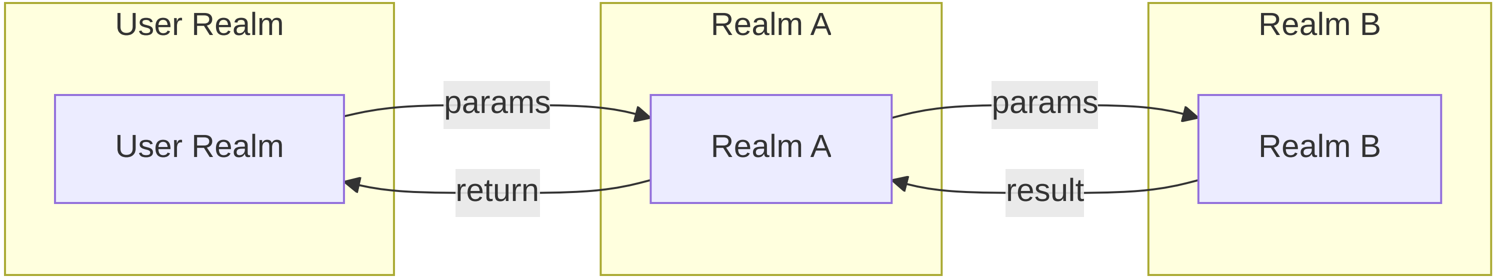

Realm A

Global Data

Local State

Interrealm Flow

Realm method call



Provenance stack

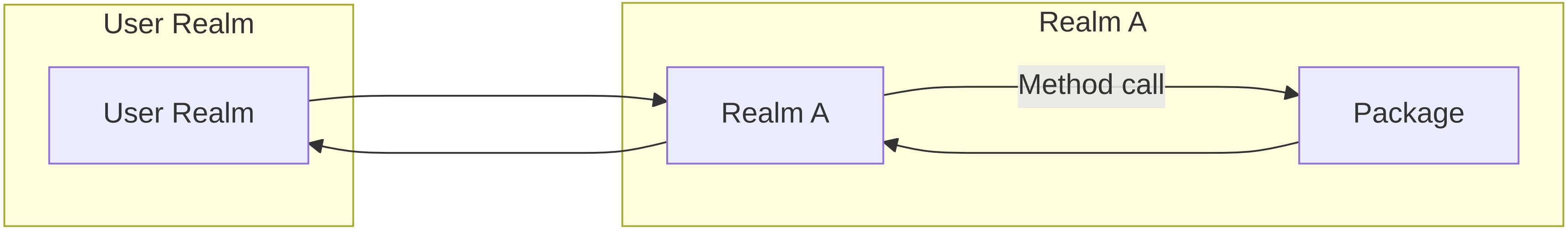
Current Realm: **B**

Called from: **A**

Origin: **User Realm**

Interrealm Flow

Package method call

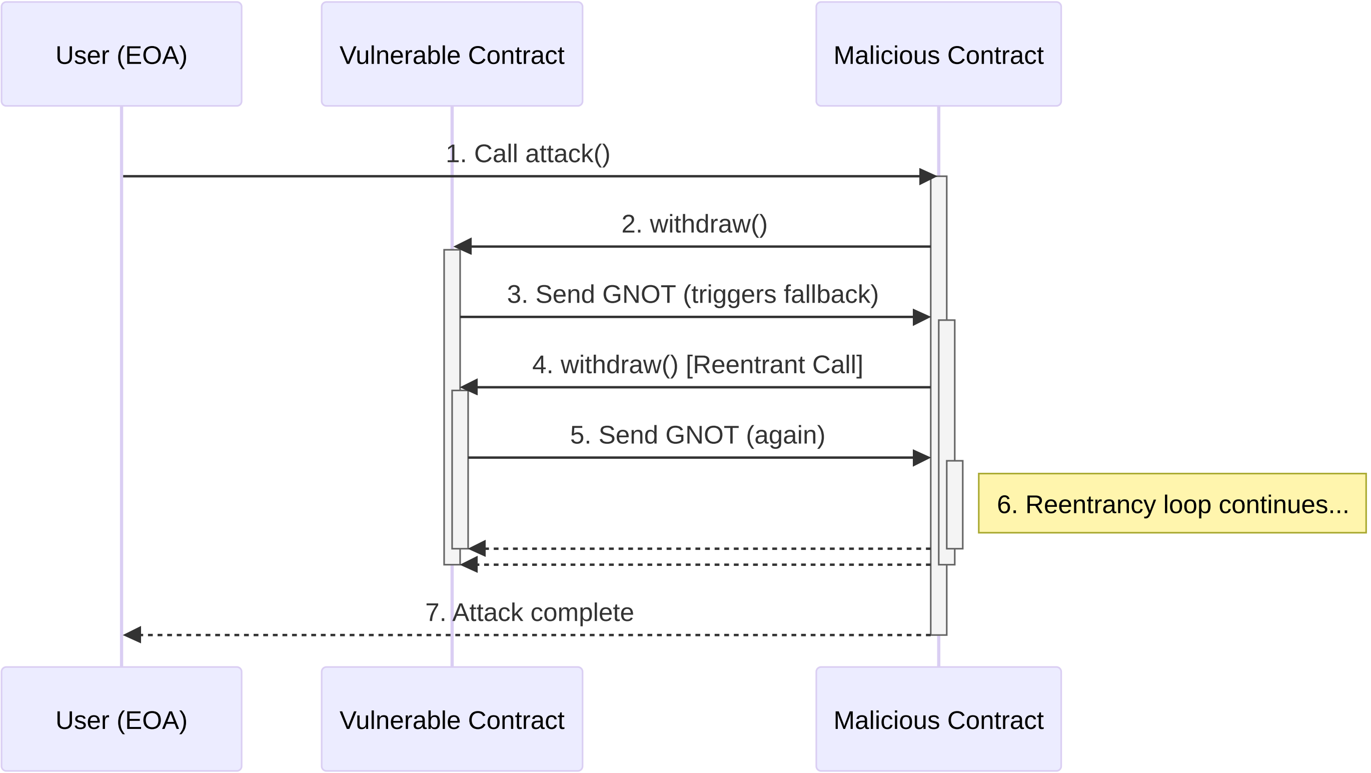


Provenance stack

Current Realm: *A**

Origin: **User Realm**

But it instaure crucial flaw - Reentrency attack

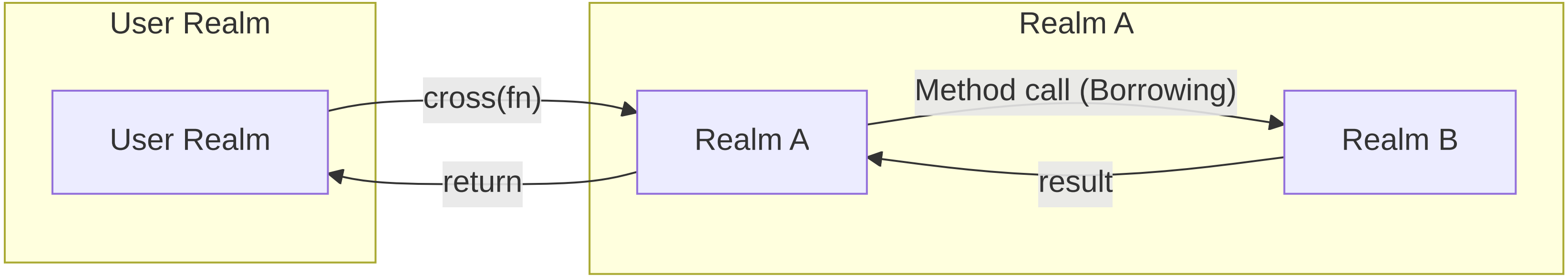


Let’s introduce Borrowing

Less permissive call for a more secure environment.

- **Borrowing** = calling a method on an external realm’s object
- You temporarily enter the object’s storage realm
- You can modify:
  - The receiver object itself
  - Objects reachable from receiver (same realm)
- **Cannot** create new root-level objects

Interrealm Flow (Borrowing)



Provenance stack

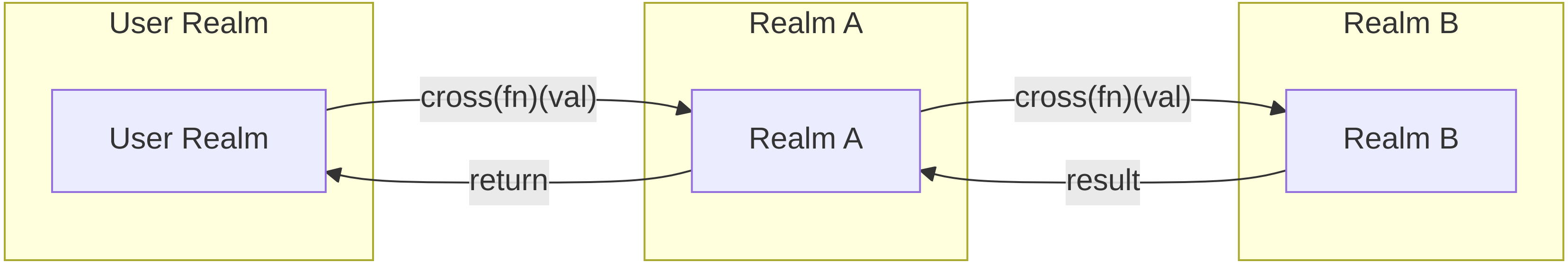
Current Realm: **A**

Origin: **User Realm**

Crossing

- **Explicitly switch** into another realm.
- **Crossing** = `cross(fn)(...)` or functions marked with `crossing()`.
- Gain full **write access** to the realm’s global storage like it used to.
- Use for **creating new objects** or performing realm-specific logic.

Interrealm Flow (Crossing)



Provenance stack

Current Realm: **B**

Called from: **A**

Origin: **User Realm**

Inter-realm Flow (Crossing)

Realm A

```
realmB.CreatePost(cross, "Hello Gno")
```

Realm B

```
func CreatePost(title string) {
    newPost := Post{
        Title: title,
    }
    Posts = append(Posts, *newPost)
}
```

Rules Summary

Action	Borrowing	Crossing
Modify existing object	✅ via method call	✅
Create new unattached object	❌	✅
Implicit realm context change	No (temporary for method)	Yes (permanent inside fn)
Method syntax	`obj.Method()`	`cross(fn)(...)` + `crossing()`

Code Example: Borrowing

```
// In realmB
func (b *Book) SetTitle(new string) {
  b.Title = new  // Allowed: borrowing b’s realm
}

// In realmA
book := &realmB.Book{}
book.SetTitle("Hello Gno")
```

Code Example: Crossing

```
// In realmB
func CreateUser(cur realm, name string) {
  user := &User{Name: name}
  users[name] = user  // Global storage in realmB
}

// In realmA
realmB.CreateUser(cross, "alice")
```

Summary

- **Realms** = isolated code.
- **Borrowing** = temporary, object-scoped access (Object operations)
- **Crossing** = explicit realm switch for full access (Public mutators)
- Choose the right approach for **security** and **clarity**.