ByPassing Withdrawal Approval

```
function approveWithdrawRequest(address user) public onlyOwner nonReentrant {
        PrimeStakedXDCStorage storage s = _getPrimeStakedXDCStorage();
        mapping(address => bool) storage _canWithdraw = s.canWithdraw;
        _canWithdraw[user] = true;
        emit WithdrawRequestApproved(user);
    }
```

Root Cause

- `approveWithdrawRequest()` sets a **per-user boolean** (`canWithdraw[user] = true`).
- This flag is **global for the user**, not linked to:
  - the **amount** staked,
  - the **specific withdrawal request**, or
  - any **time/epoch constraint**.

As a result:

- Once a user is approved **once**, that approval persists until withdrawn.
- The approval can be exploited later with a completely different stake context.

Attack Scenario Walkthrough

1. **Setup**: Attacker stakes a tiny amount (e.g., `1 wei`).
2. **Request Approval**: Calls `requestWithdraw()` → admin/owner approves → `canWithdraw[attacker] = true`.
3. **Upgrade Stake**: Attacker deposits a large amount (e.g., `1000 XDC`).
4. **Exploit**: Calls `withdraw()`. Since approval is still `true`, the attacker is allowed to withdraw the *entire large stake*, without requiring new approval.

Result: **Bypasses intended per-withdrawal approval logic**.

Impact

- Approval no longer reflects the *current* stake or context.
- Admin loses control over withdrawals: attacker can “front-load” approval on a trivial deposit, then later drain a much larger stake.
- Effectively reduces the system’s approval model from “per-withdrawal” to “once per user, forever,” which is a **broken trust assumption**.

Fix Recommendations

1. **Tie approval to request ID or amount**

   ```
   mapping(address => uint256) public approvedAmount;
   ```

   - Store the specific amount (or max) approved.
   - `withdraw()` must not exceed the approved amount.
2. **Time-bound approvals**
   - Expire approval after N blocks or a timestamp.
   - Forces re-approval for later withdrawals.
3. **Invalidate on new stake**
   - Reset `canWithdraw[user] = false` whenever user deposits again.
   - Prevents reuse of old approval.
4. **Event-driven enforcement**
   - Approval should be linked to a **specific request hash** (address + amount + timestamp).
   - Prevents replay or context change.