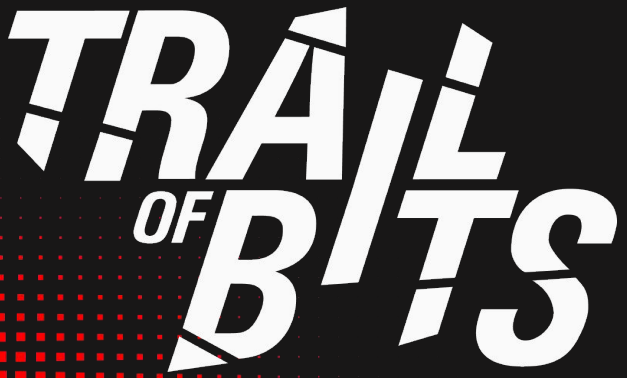


© Trail of Bits 2026. All rights reserved.

Uniswap v4 Hooks

A Security Guide for Builders and Breakers

nisedo | Trail of Bits | EthCC[9] | April 2, 2026

© Trail of Bits 2026. All rights reserved.

Who am I?



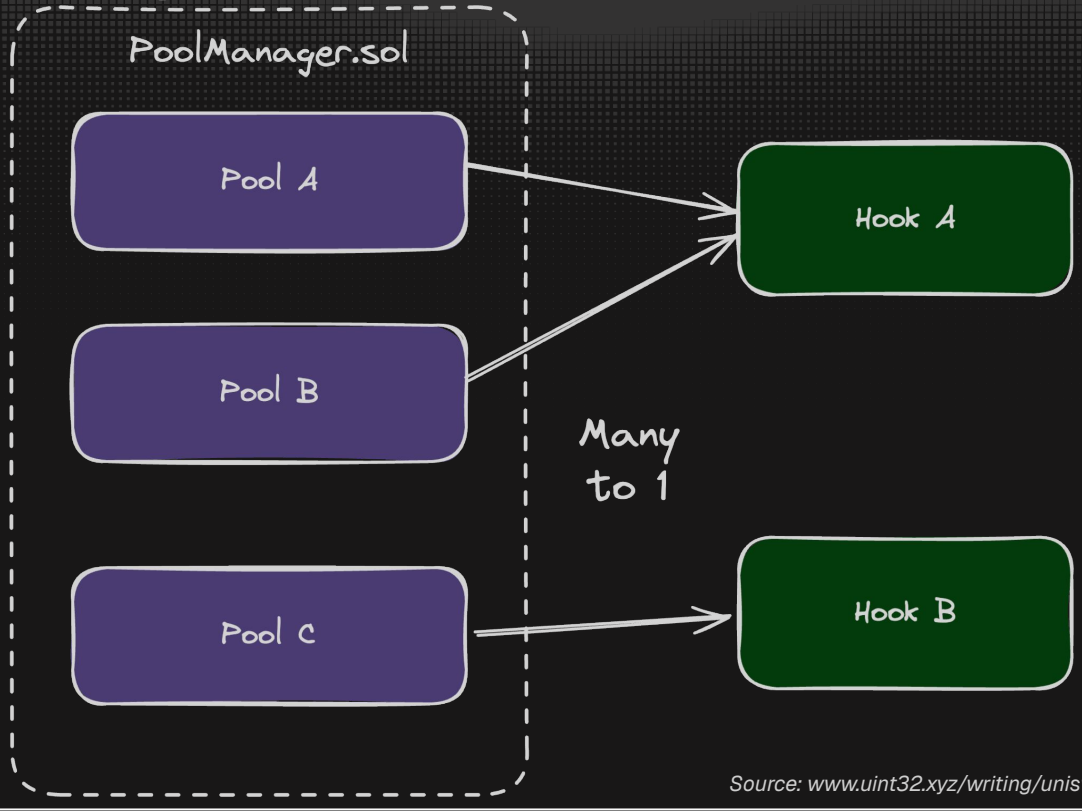
- nisedo (x.com/nisedo_)
- Founder of Soliditors, the French Web3 security community
- Blockchain Security Engineer at Trail of Bits

© Trail of Bits 2026. All rights reserved.

From v3 to v4

© Trail of Bits 2026. All rights reserved.

What Changed from v3 to v4



Source: www.uint32.xyz/writing/uniswap-v4-lock-and-callback-mechanism

© Trail of Bits 2026. All rights reserved.

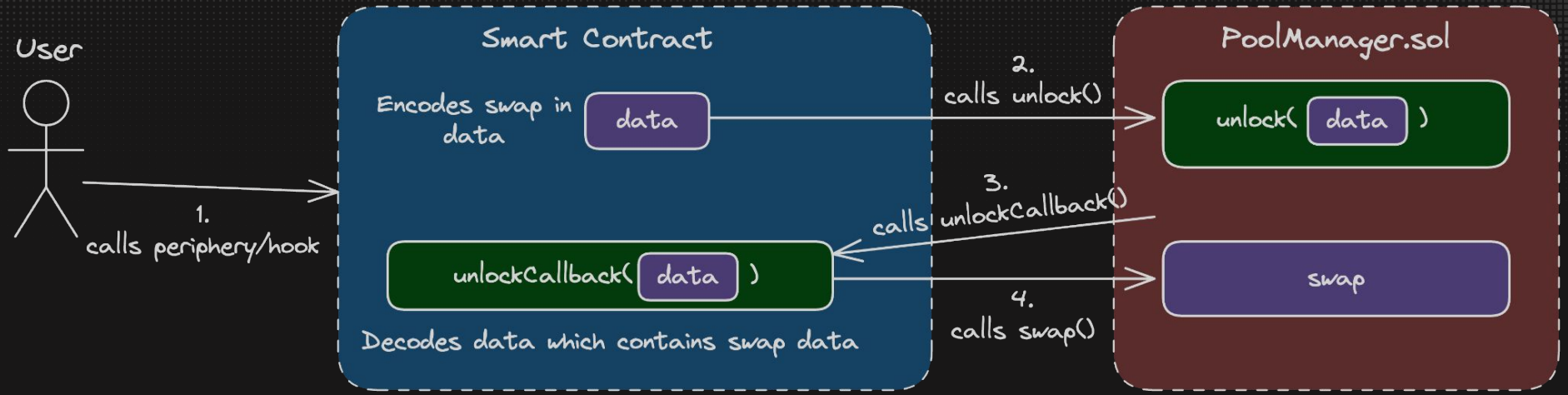
A Pool Now Includes a Hook

```
struct PoolKey {
    Currency currency0;
    Currency currency1;
    uint24 fee;
    int24 tickSpacing;
    IHooks hooks;
}
```

Source: v4-core/src/types/PoolKey.sol

© Trail of Bits 2026. All rights reserved.

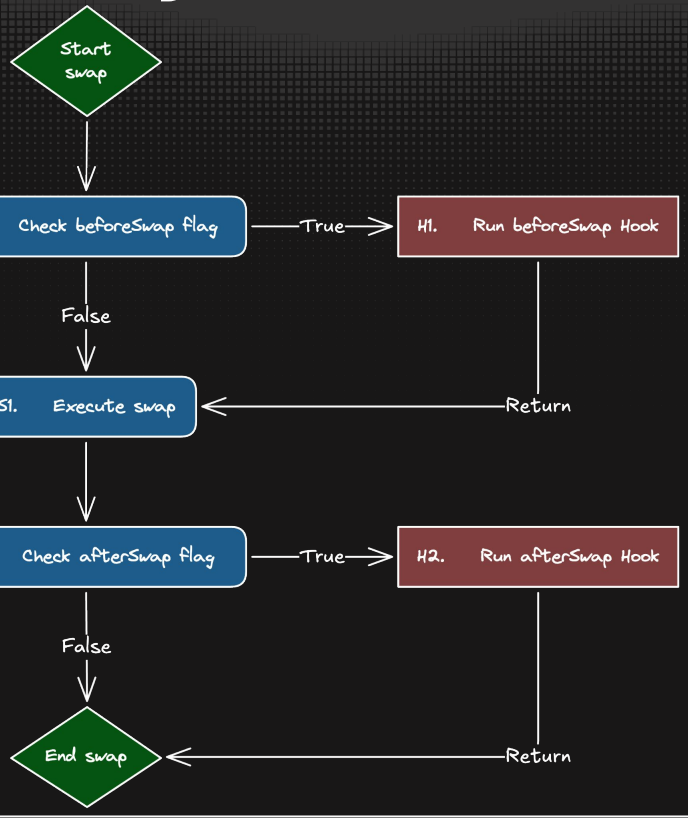
Unlock / Callback Flow



Source: www.uint32.xyz/writing/uniswap-v4-lock-and-callback-mechanism

© Trail of Bits 2026. All rights reserved.

Hook Swap Lifecycle



Source: blocksecteam.medium.com/

© Trail of Bits 2026. All rights reserved.

What PoolManager Won't Save You From

PoolManager Protects

- PoolManager protects settlement and enforces some core swap rules
- If balances and deltas do not settle at the end of unlock, the transaction reverts

PoolManager Does Not Protect

- Who may call your hook
- Which pools may use your hook
- Your custom accounting
- Your external integrations

© Trail of Bits 2026. All rights reserved.

Settlement Enforcement

```
function unlock(bytes calldata data) external returns (bytes memory result) {
    Lock.unlock();

    ...

    if (NonzeroDeltaCount.read() != 0) revert CurrencyNotSettled();

    Lock.lock();
}
```

Source: v4-core/src/PoolManager.sol

© Trail of Bits 2026. All rights reserved.

Five Ways Hooks Break

© Trail of Bits 2026. All rights reserved.

1. Anyone Can Call Your Hook

The Bug

- Hook callbacks are external functions on your contract
- If you do not check the caller, an attacker may call them directly with malicious parameters
- A loose `unlockCallback` path can also expose internal actions you did not mean to expose

Fix + Case Study

- Fix: use `BaseHook` for hook entrypoints and `SafeCallback` for `unlockCallback`
- Validate the caller on every callback path
- Cork exploit ($12M): missing caller check was one link in the exploit chain

© Trail of Bits 2026. All rights reserved.

1. Anyone Can Call Your Hook

```
modifier onlyPoolManager() {

    if (msg.sender != address(poolManager))
    revert NotPoolManager();

    _;

}
```

Source: v4-periphery/src/base/ImmutableState.sol

© Trail of Bits 2026. All rights reserved.

2. Treating Any Pool as Legit

Attack Vector

- Anyone can create a pool with a valid hook address
- If your hook trusts a user-supplied PoolKey, an attacker may route your logic through a malicious pool
- A malicious pool can overwrite your hook's state, or force it to call attacker-controlled token contracts

Fix + Example

- Fix: bind the hook to one canonical PoolKey, or to a strict allowlist, and re-check it on every user-controlled path
- Example: a coordinator contract was drainable because it accepted any pool key without checking the currencies

© Trail of Bits 2026. All rights reserved.

2. Treating Any Pool as Legit

```
if (trustedPoolKey == 0) trustedPoolKey = key;

else if (key != trustedPoolKey) revert InvalidPool();
```

Pseudocode

© Trail of Bits 2026. All rights reserved.

3. Custom Accounting Leaks Value

How Value Leaks

- Once your hook touches deltas, a wrong sign, a rounding error, or a mixed balance bucket can silently leak value

What to Test

- Can a user get free output? Can a user profit from a round trip? Does internal accounting match real token balances?
- The Bunni exploit ($8.4M) showed how badly rounding and accounting mistakes can compound once funds are pooled

© Trail of Bits 2026. All rights reserved.

4. Right Logic, Wrong Hook

Placement

- `before*` and `after*` hooks do not see the same state
- Code that is correct in one hook can be wrong in another
- If your logic needs the final swap result, it belongs in `afterSwap`, not `beforeSwap`

Reverts

- If reward code or dust cleanup can revert during `removeLiquidity`, withdrawals can get blocked

© Trail of Bits 2026. All rights reserved.

5. Address Bits Are Part of the API

Missing Permissions

- In v4, the hook address encodes which hook functions PoolManager may call
- The address is part of the API
- A missing afterSwapReturnDelta bit can make a hook think it charged a fee when PoolManager actually ignored that returned delta

Wrong Permissions

- Permission bit set + function missing → revert
- Function implemented + bit missing → hook not called
- Always inherit BaseHook → it validates bits match code at deploy time

© Trail of Bits 2026. All rights reserved.

5. Address Bits Are Part of the API

```
function hasPermission(IHooks self, uint160 flag) internal
pure returns (bool) {

    return uint160(address(self)) & flag != 0;

}
```

Source: v4-core/src/libraries/Hooks.sol

© Trail of Bits 2026. All rights reserved.

Building Secure Hooks

© Trail of Bits 2026. All rights reserved.

Builder Checklist

- Gate every callback and unlock path
- Allowlist pools, not just tokens
- Label every balance and delta: who owns it, who can move it
- Keep LP funds, fees, and incentives separate
- Keep non-essential code away from the main user flow
- Fuzz with malicious pools, tokens, and dust

© Trail of Bits 2026. All rights reserved.

Auditor Checklist

- Can an attacker call a callback directly?
- Can an attacker route logic through a malicious pool?
- Who owns each balance and each delta?
- What happens if reward or cleanup code reverts during `removeLiquidity`?
- Do the permission bits, implemented functions, and returned values all match?
- What breaks if the hook or the token is malicious?

© Trail of Bits 2026. All rights reserved.

Takeaway

`PoolManager` secures settlement.

Your hook still has to secure **authorization**, **pool selection**, and **value accounting**.

Questions?
x.com/nisedo_

© Trail of Bits 2026. All rights reserved.