Moonwell : Base ပေါ်က Lending Market တစ်ခုကနေ ဒေါ်လာ ၈.၇ သန်း ထုတ်ယူသွားစေခဲ့တဲ့ Thin Token နဲ့ Donation Trick

အကျဉ်းချုပ် (Summary)

အသေးစိတ် မဖတ်ခင် ဘာဖြစ်ခဲ့လဲဆိုတာကို အရင် အတိုချုံး ပြောပြပါရစေ။

- **Protocol နဲ့ Code** Moonwell ဆိုတာ Base ပေါ်က Compound v2 ပုံစံ lending protocol တစ်ခုပါ။ ထိသွားတဲ့ market က MAMO Core Market ပါ။ mMAMO contract address က

0x2F90Bb22eB3979f5FfAd31EA6C3F0792ca66dA32 ဖြစ်ပြီး တော့ MAMO token ကတော့ 0x7300B37DfdfAb110d83290A29DfB31B1740219fE ပါ။ ဒီ post ထဲမှာ ကျွန်တော် ကိုးကားထားတဲ့ code အားလုံးက moonwell-fi/moonwell-contracts-v2 repo ရဲ့ commit `3b32d534778b854d31bd3555c90cc787b2072566` ကို အခြေခံထားတာဖြစ်ပါတယ်ဗျ။

Vulnerability အမျိုးအစား Code ကြောင့်ဖြစ်တဲ့ bug မဟုတ်ပါဘူး၊ economic attack ပါ။ အားနည်းချက် သုံးခုကို သုံးသွားတာပါ။ Thin token တစ်ခုပေါ်က spot price oracle၊ နောက်တစ်ခုက contract ထဲကို တိုက်ရိုက် ပို့လိုက်တဲ့ token တွေကိုပါ ရေတွက်တဲ့ exchange rate၊ ပြီးတော့ mint လုပ်တဲ့အခါမှပဲ စစ်တဲ့ supply cap။ တစ်ခုချင်းစီက bug မဟုတ်ပါဘူး။ ဒါပေမယ့် သုံးခုပေါင်းလိုက်တော့ ဒေါ်လာ ၉ သန်းလောက် ဆုံးရှုံးနိုင်တဲ့ hole ဖြစ်သွားပါတယ်။

- **ဆုံးရှုံးမှု။** Attacker က manipulate လုပ်ထားတဲ့ MAMO collateral နဲ့ cbBTC, WETH, USDC, wstETH စုစုပေါင်း ဒေါ်လာ ၁၁.၀၃ သန်းဖိုး တန်ဖိုးရှိတဲ့ asset တွေကို ချေးယူသွားပါတယ်။ CertiK နဲ့ PeckShield တို့ကတော့ ဆုံးရှုံးမှုကို ဒေါ်လာ ၈.၇ သန်းလောက်ရှိမယ်လို့ ခန့်မှန်းထားပါတယ်။ Liquidation တွေ လုပ်ပြီးသွားတဲ့နောက်မှာတောင် အဲဒီ market လေးခုထဲမှာ bad debt ဒေါ်လာ ၉.၁၃ သန်းလောက် ကျန်နေပါသေးတယ်။

- **လက်ရှိအခြေအနေ။** ဒီကိစ္စက ၂၀၂၆ ခုနှစ် ဩဂုတ်လ ၂၇ ရက်နေ့မှာ ဖြစ်ခဲ့တာပါ။ အဲဒီမနက်မှာပဲ Moonwell က Base ပေါ်က borrow cap အားလုံးကို 1 wei အထိလျှော့လိုက်တယ်။ ဩဂုတ်လ ၃၀ ရက်နေ့မှာ Anthias Labs က governance forum ပေါ်မှာ အသေးစိတ် post mortem ကို ထုတ်ပြန်ခဲ့ပါတယ်။ ယနေ့ ဩဂုတ်လ ၃၁ ရက်နေ့အထိတော့ ငွေဆုံးသွားတဲ့ supplier တွေကို ဘယ်လို ပြန်ပေးမယ်ဆိုတဲ့ plan မရှိသေးပါဘူးဗျ။

တစ်ခုကြိုပြောချင်တာက သတင်းခေါင်းစဉ်တော်တော်များများကတော့ “oracle hack” လို့ပဲ ပြောသွားကြတယ်။ တကယ်တော့ Oracle လို့ပြောရင် က တစ်ဝက်ပဲမှန်မယ်။ ကျန်တဲ့ တစ်ဝက်က Compound v2 fork တွေတိုင်းမှာ ရှိတဲ့ ထူးဆန်းတဲ့ Accounting ပုံစံတစ်ခုပါ။ mToken contract ထဲကို token တွေ တိုက်ရိုက် ပို့လိုက်မယ်ဆိုရင် ရှိပြီးသား share တစ်ခုချင်းဆီရဲ့ တန်ဖိုးက ပိုတက်သွားတယ်၊ ဒါပေမယ့် supply cap လည်း မစစ်ဘူး၊ event လည်း မထွက်လာဘူး။ Attacker က အဲ့ဒီအားနည်းချက် နှစ်ခုလုံးကို ရောပြီးသုံးသွားတာ။ အဲ့ဒါကြောင့် oracle တစ်ခုကိုပဲ ပြင်လိုက်မယ်ဆိုရင်တောင် နောက်တစ်ခုဖြစ်တဲ့ accounting ရဲ့အားနည်းချက်က ဆက်ပြီးရှိနေဦးမှာပါပဲ။

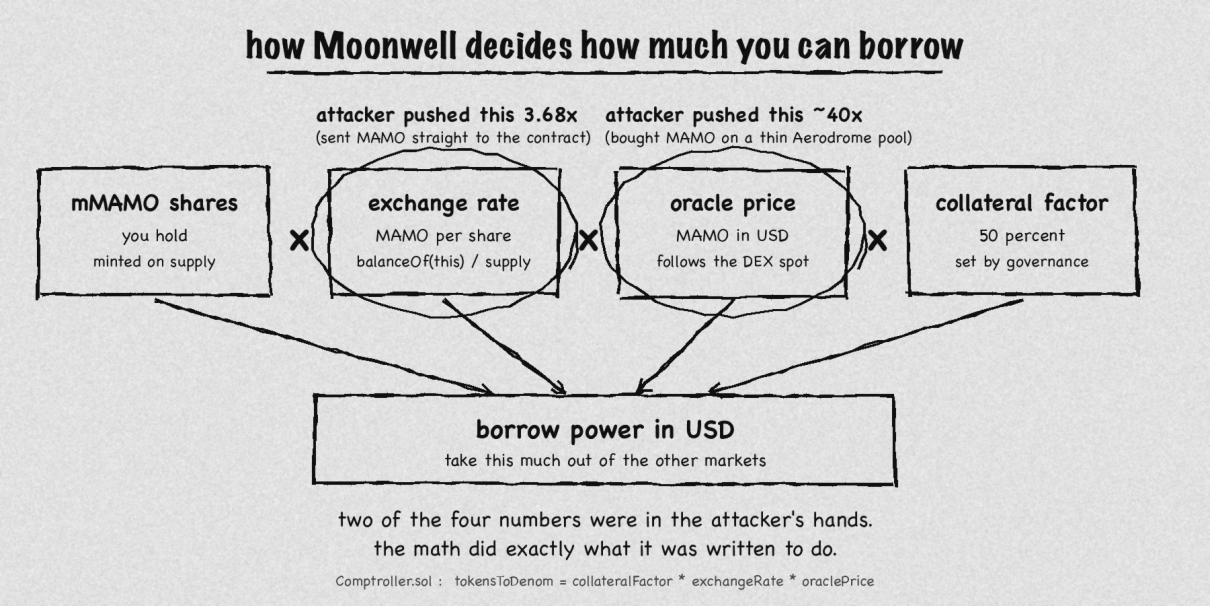
Background

Moonwell က lending protocol အလွယ်ပြောရရင် ချေးငွေထုတ်ပေးတဲ့ Defi protocol တစ်ခုပေါ့ဗျာ။ ခင်ဗျားက Token တစ်ခုကို deposit လုပ်လိုက်ရင် mToken လို့ ခေါ်တဲ့ receipt token တစ်ခု ပြန်ရမယ်။ အဲဒီ deposit ကို အာမခံထားပြီး တခြား token တွေကို လိုက်ချေးလို့ရတယ်။ Compound v2 ရဲ့ fork တစ်ခု ဖြစ်တဲ့အတွက် cToken ဘယ်လို အလုပ်လုပ်လဲ ဆိုတာသိရင် ဒီစနစ်ကို နားလည်ဖို့လွယ်မယ် ထင်တယ်ဗျ။

အလုပ်လုပ်ပုံကတော့ ရိုးရိုးလေးပါပဲ။

1. MAMO ကို mMAMO market ထဲ supply လုပ်တယ်။ မင်းက MAMO ကို mMAMO market ထဲ deposit or supply လုပ်လိုက်ရင် contract က မင်းကို mMAMO shares ထုတ်ပေးတယ်။ မင်းရမယ့် share အရေအတွက်က အဲဒီအချိန်မှာရှိနေတဲ့ **exchange rate** အပေါ်မူတည်ပါတယ်။

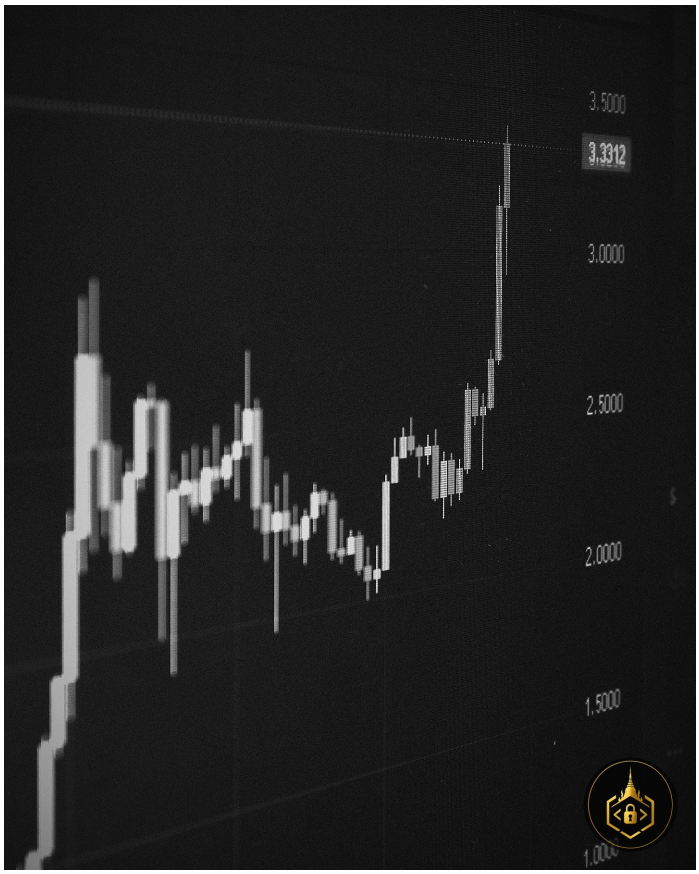
2. Protocol က မင်းရဲ့ share တွေ USD နဲ့ ဘယ်လောက်တန်လဲ တွက်တယ်။ Share အရေအတွက်ကို exchange rate နဲ့ မြှောက်၊ ပြီးရင် oracle price နဲ့ မြှောက်၊ ပြီးရင် collateral factor လို့ ခေါ်တဲ့ safety haircut နဲ့ ထပ် မြှောက်တယ် (Shares x Exchange Rate x Oracle Price x Collateral factor)။ Collateral factor ဆိုတာ safety အတွက် တန်ဖိုးတစ်ခုကို လျှော့ပြီး ချေးခွင့်ပေးတဲ့ ရာခိုင်နှုန်းပါ။ MAMO အတွက် အဲဒီ collateral factor ကတော့ ၅၀ ရာခိုင်နှုန်းပါ။ ဥပမာ MAMO collateral ရဲ့ တန်ဖိုးကို protocol က $100,000 လို့တွက်ရင် $100,000 × 50% = $50,000 အထိပဲ borrow လုပ်နိုင်မယ်ဆိုတဲ့သဘောပါ။
3. အဲဒီ USD တန်ဖိုးအထိ Base ပေါ်က တခြား market တွေကနေ cbBTC, WETH, USDC စသဖြင့် token တွေချေးလို့ရတယ်။



How Moonwell values collateral. Four numbers multiplied together. The attacker controlled two of them.

ဒီတော့ lender တိုင်းရဲ့ deposit တွေလုံခြုံဖို့ဆိုတာက အဲဒီ ဂဏန်း လေးခုလုံးမှန်နေဖို့လိုတာပေါ့နော်။ Share နဲ့ collateral factor ကို protocol က သတ်မှတ်ပေးထားတာပါ။ ကျန်တဲ့နှစ်ခု exchange rate နဲ့ oracle price ကတော့ အပြင်က data ပြန်ယူတာပါ။ အဲဒီနေရာမှာ ပြဿနာက စတာပါ။

MAMO token က Mamo ဆိုတဲ့ Base ပေါ်က personal finance app ရဲ့ token ဗျာ။ Moonwell ကို လုပ်တဲ့ team ကပဲ လုပ်ထားတာ။ trading လုပ်တဲ့လူလည်းနည်းတယ် liquidity လည်းနည်းတယ်။ Attack ဖြစ်တဲ့နေ့မှာ market cap တစ်ခုလုံးက ဒေါ်လာ ၆ သန်းလောက်ပဲ ရှိပြီးတော့ 24 hour trading volume က ဒေါ်လာ ၁.၁၈ သန်းလောက်ပဲရှိတာဗျာ။ Moonwell က MAMO ကို collateral အဖြစ်နဲ့ သုံးခွင့်ပေးထားပြီးတော့ MAMO supply cap ကို 20 million သတ်မှတ်ထားတယ်။ ပြီးတော့ collateral factor ကို ၅၀ ရာခိုင်နှုန်းသတ်မှတ်ထားတယ်။



Docs ကို https://docs.moonwell.fi/ မှာ ဖတ်လို့ရပြီး contract list ကို https://docs.moonwell.fi/moonwell/protocol-information/contracts မှာ ကြည့်လို့ရပါတယ်။

Vulnerable Code

ဒီနေရာမှာ “vulnerable function” တစ်ခုတည်းရှိတာမဟုတ်ဘူး။ တကယ်က အပိုင်းသုံးပိုင်းရှိတယ်လို့ ပြောလို့ရတယ်ပေါ့နော်၊ ပုံမှန်ကြည့်လိုက်ရင်တော့ တစ်ခုချင်းဆီက သူ့အလုပ်သူလုပ်နေတာပဲ။ ဒါပေမယ့် အဲ့ဒီသုံးခု ပေါင်းလိုက်တော့ attacker က ကိုယ့် borrowing power ကိုယ်တိုင် သတ်မှတ်လို့ရတဲ့ အခြေအနေ ဖြစ်သွားတယ်။ အဲ့ဒီ code သုံးပိုင်းလုံး ပြမယ်

အပိုင်း ၁ : Exchange rate က token balance ကို ယုံတယ်

`src/MToken.sol` ထဲမှာ exchange rate ကို ဒီလို တွက်တယ်

```
// src/MToken.sol, lines 440 to 475 (commit 3b32d53)
function exchangeRateStoredInternal() internal view virtual returns (MathError, uint) {
    uint _totalSupply = totalSupply;
    if (_totalSupply == 0) {
        return (MathError.NO_ERROR, initialExchangeRateMantissa);
    } else {
        /*
         * Otherwise:
         *  exchangeRate = (totalCash + totalBorrows - totalReserves) / totalSupply
         */
        uint totalCash = getCashPrior();
        // ... add totalBorrows, subtract totalReserves, divide by totalSupply
    }
}
```

`src/MErc20.sol` ထဲက `getCashPrior` ကတော့ raw ERC20 balance ပဲ။

```
// src/MErc20.sol, lines 205 to 208 (commit 3b32d53)
function getCashPrior() internal view virtual override returns (uint) {
    EIP20Interface token = EIP20Interface(underlying);
    return token.balanceOf(address(this));   // counts EVERY token in the contract
}
```

ဆိုလိုတာက တစ်ယောက်ယောက်က `mint` မလုပ်ဘဲ `MAMO.transfer(mMAMO, hugeAmount)` လုပ်လိုက်ရင် `totalCash` ကသွားတယ်၊ `totalSupply` က တော့အရင် အတိုင်းပဲ၊ ဒီတော့ ရှိပြီးသား mMAMO share တစ်ခုချင်းဆီရဲ့ တန်ဖိုးက ရုတ်တရက် ပိုများလာတယ်။ ဥပမာ ၁၀၀၀ တန်တဲ့ ပစ္စည်းက ရုတ်တရက် ၂၀၀၀ ကိုဈေးတက်သွားတာမျိုးပြောချင်တာ။ Moonwell ဘက်က event လည်း မထွက်ဘူး။ Hook လည်း မ run ဘူး။ ဒါက Compound v2 ရဲ့ ပုံမှန် အလုပ်လုပ်တဲ့ ပုံစံပဲ၊ ပုံမှန်အားဖြင့်တော့ ဘာအန္တရာယ်မှ မရှိဘူး။ ဘာလို့လဲဆိုတော့ shared pool ထဲကို token တွေ donation အနေနဲ့ပို့လိုက်ရင် ကိုယ့်ပိုက်ဆံကို လူတိုင်းကို ဝေပေးလိုက်တာနဲ့ အတူတူပဲ။ ဒါပေမယ့် လူတစ်ယောက်တည်းက share အားလုံးနီးပါး ကိုင်ထားလိုက်ရင်တော့ အန္တရာယ် ရှိလာတာပေါ့။

အပိုင်း ၂ : Supply cap က mint ကိုပဲ ကြည့်တယ်

`src/Comptroller.sol` ထဲက `mintAllowed` ထဲမှာ cap စစ်တဲ့ အပိုင်းပါ။

```
// src/Comptroller.sol, lines 282 to 296 (commit 3b32d53)
uint supplyCap = supplyCaps[mToken];
if (supplyCap != 0) {
    uint totalCash = MToken(mToken).getCash();
    uint totalBorrows = MToken(mToken).totalBorrows();
    uint totalReserves = MToken(mToken).totalReserves();
    uint totalSupplies = sub_(add_(totalCash, totalBorrows), totalReserves);

    uint nextTotalSupplies = add_(totalSupplies, mintAmount);
    require(nextTotalSupplies < supplyCap, "market supply cap reached");
}
```

အဲ့ဒီ mintAllowed ဆိုတာကို `mint` လုပ်တဲ့အခါမှပဲ ခေါ်တာ။ တိုက်ရိုက် transfer လုပ်လိုက်ရင် `mintAllowed` ကိုခေါ်စရာမလိုဘူးဗျာ။ ဆိုတော့ MAMO သန်း ၂၀ရဲ့ 20 million supply cap က attacker ကို contract ထဲမှာ MAMO သန်း ၅၃ ထပ်ပြီးထည့်တာကို မတားနိုင်ခဲ့ဘူး။ Cap ရဲ့ ရည်ရွယ်ချက်က protocol က MAMO ဘယ်လောက်ပမာဏအထိ collateral အဖြစ် လက်ခံမလဲ ကန့်သတ်ဖို့အတွက်ဗျာ။

တကယ်တမ်းတော့ အိမ်အရှေ့တံခါးက ဘယ်လောက်ပမာဏ အထိပဲ ရပါတယ်လို့ ကန်သတ်ထားပေမယ့် အိမ်အနောက်တံခါးကြီးကတော့ ဖွင့်ထားလို့ နောက်က ပတ်ဝင်သွားတာပါ။

အပိုင်း ၃ : Oracle က liquidity နည်းတဲ့ DEX price နောက်ကို လိုက်တယ်

Collateral တန်ဖိုးကို `oracle.getUnderlyingPrice(asset)` ကနေ price ယူပြီးတော့ အောက်ကလိုတွက်တယ်

```
// src/Comptroller.sol, lines 745 to 748 (commit 3b32d53)
// Pre-compute a conversion factor from tokens -> glmr (normalized price value)
vars.tokensToDenom = mul_(
    mul_(vars.collateralFactor, vars.exchangeRate),
    vars.oraclePrice
);
```

MAMO price feed က DEX ပေါ်မှာဖြစ်နေတဲ့ spot trading price ကို လိုက်ပြီး update လုပ်နေတာဗျ။ Post mortem မှာ feed provider နာမည် မဖော်ပြထားပေမယ့် ဂဏန်းတွေတော့ ပေးထားတယ်။ Attack အတွင်း MAMO က $0.010597 ကနေ အမြင့်ဆုံး $0.43127363 အထိ တက်သွားခဲ့တယ်။ Moonwell က valid လို့ လက်ခံခဲ့တဲ့ အမြင့်ဆုံး ဈေးက $0.40248571။ Time weighted average (TWAP) မရှိ၊ price အရမ်းတက်သွားရင် တားမယ့် sanity band မရှိ၊ “ဒီ token က နှစ်နာရီအတွင်း အဆ ၄၀ တက်လို့ မဖြစ်နိုင်ဘူး” လို့ စစ်ပေးမယ့် liquidity check လည်း မရှိဘူး။

သုံးပိုင်း ပေါင်းလိုက်ရင် အပိုင်း ၃ က formula က `0.5 x (attacker က 3.68 ဆ မြှင့်ထားတဲ့ exchange rate) x (attacker က 38 ဆ မြှင့်လိုက်တဲ့ price)`။ exploit တစ်ခုလုံး ရဲ့ အဓိက အကြောင်းအရင်းက ဒါပဲဗျာ။

Root Cause

ရှိသင့်တဲ့ စည်းမျဉ်း

Lending market တိုင်း အတွက် အခြေခံအားဖြင့်တော့ စည်းမျည်းတစ်ခုရှိတယ်ဗျာ။

Protocol က မင်းရဲ့ collateral ကို သတ်မှတ်ပေးတဲ့ USD တန်ဖိုးဟာ မင်း liquidate ခံရရင် အဲ့ဒီ collateral ကို market မှာ ပြန်ရောင်းရင် တကယ်ရနိုင်မယ့် ဈေးနဲ့ အနီးစပ်ဆုံးတော့ဖြစ်ရမယ်ဗျာ။

၂၀၂၆ ဩဂုတ် ၂၇ မှာတော့ အဲဒီစည်းမျဉ်းက တစ်ချိန်တည်းမှာပဲ နှစ်ခါ broke သွားတာပေါ့။

- **Oracle ဘက်။** MAMO price feed က $0.40 လို့ ပြနေတယ်။ ဒါပေမယ့် တကယ့် market depth ကိုကြည့်ရင် MAMO ၉၄ သန်းကို အဲ့ဒီဈေးနဲ့ နည်းနည်းလေးတောင်ရောင်းလို့မရဘူး။ Liquidator တွေ ဒါကို သိသွားတယ်။ Attacker ရဲ့ mMAMO အားလုံးနီးပါး သိမ်းလိုက်နိုင်ပေမယ့် debt အများစုကို ပြန်မရနိုင်ခဲ့ဘူး၊ ဘာလို့လဲဆိုတော့ သိမ်းလိုက်တဲ့ MAMO က oracle ပြထားတဲ့ တန်ဖိုးရဲ့ အနည်းငယ်လောက်ပဲ တန်လို့။
- **Accounting ဘက်။** Exchange rate က mMAMO share တစ်ခုစီကို MAMO 0.0755 နဲ့ backing လုပ်ထားတယ်လို့ ပြောတယ်။ Technically မှန်တယ်၊ attacker က MAMO တွေကို contract ထဲ တကယ် donate လုပ်ထားလို့။ ဒါပေမယ့် အဲဒီ MAMO အပိုတွေက risk manager တွေ သတ်မှတ်ထားတဲ့ သန်း ၂၀ supply cap ကိုကျော်သွားပါတယ်။ Cap ရဲ့ရည်ရွယ်ချက်က ပြောခဲ့သလိုပဲ protocol ထဲမှာ MAMO အများကြီး မကိုင်ထားအောင် တားဖို့ပဲဗျာ။

  Symptom နဲ့ cause ကို ခွဲကြည့်ရင် ပိုရှင်းတယ်။

- Symptom : Attacker က ဒေါ်လာ ၁၁ သန်း တန်ဖိုးရှိတဲ့ asset တွေကို ချေးသွားပြီး MAMO ဝယ်ဖို့အတွက် ဒေါ်လာ ၇.၅ သန်း ဝန်းကျင်ပဲသုံးလိုက်တယ်ဗျာ၊ အဲဒီအထဲက အများစုကိုလည်း နောက်ပိုင်း MAMO ပြန်ရောင်းပြီး ပြန်ရသွားတာပဲ။
- Cause : Collateral formula ရဲ့ input လေးခုထဲက နှစ်ခုက attacker ကိုယ်တိုင် ပြောင်းလို့ရတဲ့ ဟာတွေဖြစ်နေတယ်။ အဲ့ဒီ input နှစ်ခုမှာလည်း ရုတ်တရက်ဖြစ်မယ့် huge moves တွေကို တားမယ့် guard rail တစ်ခုမှ ရှိမနေဘူး။

ဘာလို့ “oracle bug သက်သက်” မဟုတ်လဲ

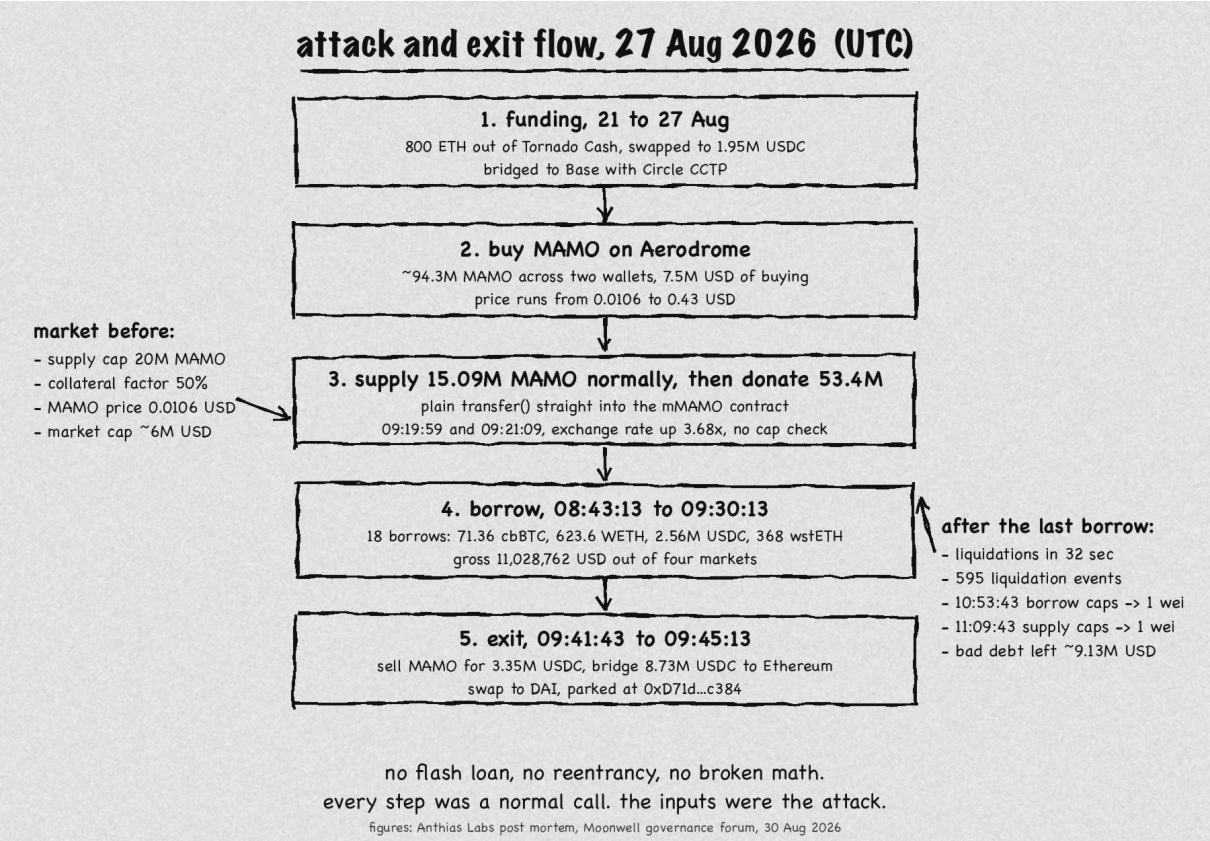
တကယ်လို့ Oracle က လုံးဝ ပြည့်စုံနေပြီးတော့ donation trick တစ်ခုပဲ ရှိခဲ့ရင် attacker က supply cap ကခွင့်ပြုထားတဲ့ collateral ပမာဏ ထက် ၃.၆၈ ဆ ပိုများတဲ့ collateral ကို fair price နဲ့ရမှာပေါ့ဗျာ။ အဲ့တာဆိုရင် အခုလောက်တော့ ကြီးကြီးကျယ်ကျယ်ဖြစ်မသွားမှာ မဟုတ်ဘူးဗျာ အခုထက်ပိုသေးတဲ့ ပြဿနာလေးပဲဖြစ်သွားမှာ။ တကယ်လို့ Oracle ကပဲ ပြဿနာဖြစ်ပြီးတော့ donation trick မရှိခဲ့ရင် attacker က MAMO သန်း ၂၀ supply cap မှာ တင်ရပ်သွားမယ်၊ MAMO ဈေးကို $0.40 နဲ့ collateral factor ၅၀ ရာခိုင်နှုန်း နဲ့တွက်ရင် 20 x 0.04 x 50% = 4 millions ဝန်းကျင်လောက်ပဲ (ဒေါ်လာ ၄ သန်းလောက်ပဲ) ပဲ borrow လုပ်နိုင်မှာဗျာ။ အခုလို ၁၁ သန်းလောက်တော့ မဟုတ်တော့ဘူးပေါ့။ အားနည်းချက် နှစ်ခုက တစ်ခုနဲ့တစ်ခု ပေါင်းပြီးအကြီးကြီးတစ်ခုဖြစ်သွားတာ။ အဲ့တာကြောင့် fix က လည်း နှစ်ခုလုံး ပြင်မှ ရမယ်။

တကယ့် design အမှားက thin token/ liquidity နည်းတဲ့ token ကို collateral အဖြစ် ထားတာ

နောက်တစ်ဆင့်ထပ်ကြည့်ရင် အဓိက ပြဿနာက MAMO ကို collateral အနေနဲ့ ထားတာပဲ။ MAMO ရဲ့ market cap က ဒေါ်လာ ၆ သန်း ဝန်းကျင်လောက်ပဲရှိတာဗျာ၊ တစ်နေ့ကို trading volume က ဒေါ်လာ တစ်သန်းကျော်ပဲ ရှိတယ်။ Moonwell က အဲဒါကို token သန်း ၂၀ အထိ ၅၀ ရာခိုင်နှုန်းနဲ့ collateral အဖြစ် ခွင့်ပြုထားတယ်။ ဒေါ်လာ ၂ သန်းနဲ့ စတဲ့ attacker တစ်ယောက်က တစ်မနက်တည်းနဲ့ ဈေးကို အဆ ၄၀ လောက်ဖြစ်အောင် manipulate လုပ်နိုင်ခဲ့ပါတယ်။ အဲဒီလို လုပ်လို့ရတဲ့ token မျိုးကို oracle ဘယ်လောက်ပဲ ကောင်းကောင်း cbBTC, WETH လို real asset တွေကို အဲ့လောက်ချေးလို့ရတဲ့ collateral အဖြစ်မထားသင့်ဘူး။

Exploit Walkthrough

ဒါက economic attack ဖြစ်တဲ့အတွက် flash loan လည်း မပါဘူး၊ reentrancy လည်း မပါဘူး။ စိတ်ရှည်ရှည်နဲ့ လုပ်သွားတာ စဉ်းစားကြည့်ရင် ပျင်းတော့ ပျင်းစရာကြီးဗျ၊ ရယ်လည်းရယ်ချင်စရာကြီး ပါဗျာ ကိုယ်ဖြစ်တာမှမဟုတ်တာ။



The full attack and exist flow, everything figure is from the Anthias Labs post mortem

1. **Funding, သြဂုတ် ၂၁ ကနေ ၂၇။** Attacker ရဲ့ funding wallet က Tornado Cash ကနေ ETH 800 ကို အကြိမ်များစွာ ခွဲထုတ်ခဲ့တယ်။ ETH 799 ကို CoW Swap မှာ USDC 1,947,391 လဲပြီး Circle ရဲ့ CCTP ကနေ Base ကို bridge လုပ်တယ်။ Ethereum burn : 0x6fd85b1f… 505b7205။ Base mint : 0x79195be2…e12d8852။

2. **MAMO ဝယ်တယ်။** Main wallet 0x719eae70…84BE919D က Aerodrome MAMO/USDC pool မှာ swap ၄၇ ကြိမ် လုပ်ပြီး MAMO သန်း ၈၆.၉ လောက် စုတယ်။ ဒုတိယ wallet 0xD71dd9b6…e338c384 က ထပ်ဝယ်တယ်။ နှစ်ခုပေါင်း MAMO သန်း ၉၄.၃ လောက် ရသွားတယ်၊ gross ဝယ်ငွေ ဒေါ်လာ ၇.၅ သန်းလောက်။ Pool က အရမ်း thin ဖြစ်တဲ့အတွက် အဲဒီဝယ်တဲ့အားက ဈေးကို တစ်ဆ ကနေ အဆ ၄၀ လောက်ထိ တင်လိုက်သလိုဖြစ်သွားတယ်။ Oracle က လည်း နောက်က လိုက်တက်တာပေါ့ဗျာ။

3. **ပုံမှန်နည်းနဲ့ supply လုပ်တယ်။** Attacker က MAMO သန်း ၁၅.၀၉ လောက်ကို mMAMO market ထဲ supply လုပ်ပြီး share တွေ ရတယ်။ Market က တခြားသူတွေရဲ့ deposit တွေနဲ့ သန်း ၂၀ cap နားကို ရောက်နေပြီဆိုတော့ အပေါ်မှာပြောခဲ့သလိုပဲ အိမ်ရှေ့တံခါးက ခွင့်ပြု သလောက် အကုန်နီးပါး ထည့်လိုက်တာပဲ။

4. **နောက်ကနေ donate တယ်။** 09:19:59 UTC နဲ့ 09:21:09 UTC မှာ attacker က MAMO 53,393,290 ကို ရိုးရိုး `transfer` call နဲ့ mMAMO contract ဆီ တိုက်ရိုက် ပို့လိုက်တယ်။ Transaction တွေက 0x056597f4…de109ce9 နဲ့ 0xaf284d7f…61de3cd9။ Mint မရှိ၊ cap check မရှိ။ Exchange rate က share တစ်ခုကို MAMO 0.020513 ကနေ 0.075460 အထိ ခုန်

တက်သွားတယ်၊ ၃.၆၈ ဆ။ Attacker က share အများစု ကိုင်ထားတဲ့အတွက် အဲဒီ တက်တာရဲ့ အများစုက သူ့ဆီပဲ ရောက်သွားတယ်။

5. **ချေးတယ်။** 08:43:13 ကနေ 09:30:13 UTC အတွင်း attacker က transaction ၁၇ ခုနဲ့ borrow ၁၈ ကြိမ် လုပ်တယ်။ ပထမ borrow : 0x09687d74…a395593e (cbBTC 0.5 test လေး)။ 09:30:13 မှာ နောက်ဆုံး borrow : 0xee2b7564…c18c88b18 (USDC 990,000)။ စုစုပေါင်း cbBTC 71.36, WETH 623.60, USDC 2,560,000, wstETH 368၊ အဲဒီအချိန်က တန်ဖိုး $11,028,762။

6. **ထွက်တယ်။** 09:41:43 ကနေ 09:45:13 UTC အတွင်း attacker က MAMO ကို USDC 3,345,134 နဲ့ ရောင်းတယ်၊ ပြီးတော့ Base ပေါ်မှာ USDC 8,729,453 ကို Wormhole ရဲ့ CCTPv2 bridge ကနေ burn လုပ်တယ် (0x9cd2fbe0…567f1dec နဲ့ 0xfcb2ff81…e40ecac03)၊ Ethereum ပေါ်မှာ USDC 8,728,318 ရပြီး Velora မှာ DAI လဲလိုက်တယ် (0xd8fa6fbe…676ca8622)။ DAI တွေက Ethereum ပေါ်က 0xD71d wallet ထဲမှာ ရှိနေတယ်။

7. **Liquidation တွေ။** နောက်ဆုံး borrow ပြီး ၃၂ စက္ကန့်အကြာမှာ liquidator တွေ စဝင်တယ်။ Transaction ၅၉၄ ခုမှာ liquidation event ၅၉၅ ခု ဖြစ်ပြီး attacker ရဲ့ mint လုပ်ထားတဲ့ mMAMO ၁၀၀ ရာခိုင်နှုန်းနီးပါး သိမ်းလိုက်တယ်၊ cbBTC 17.76, WETH 38.57, USDC 215,841, wstETH 25.31 ပြန်ရယူနိုင်တယ်။အဲ့လိုလုပ်ယုံနဲ့တော့ လုံးဝ မလောက်ဘူးဗျာ ဘာလို့လဲဆိုတော့ pump ပြန်ကျသွားတာနဲ့ သိမ်းထားတဲ့ MAMO က ဘာမှ မတန်တော့ဘူး။

8. **အရေးပေါ်ဘရိတ်။** 10:53:43 UTC မှာ Base ပေါ်က Core Market အားလုံးရဲ့ borrow cap 1 wei ဖြစ်သွားတယ်။ 11:09:43 UTC မှာ MAMO နဲ့ WELL supply cap တွေလည်း 1 wei ဖြစ်သွားတယ်။

စိတ်ပါရင် အောက်က address တွေတိုင်းကို Basescan ဒါမှမဟုတ် Etherscan မှာ ကြည့်ကြည့်ပေါ့ဗျာ။

Role	Address	မှတ်ချက်
mMAMO market	0x2F90Bb22eB3979f5FfAd31EA6C3F0792ca66dA32	docs ထဲမှာ “Moonwell MAMO”
MAMO token	0x7300B37DfdfAb110d83290A29DfB31B1740219fE	thin collateral
Comptroller	0xfBb21d0380beE3312B33c4353c8936a0F13EF26C	cap နဲ့ collateral factor တွေ ကိုင်ထားတဲ့နေရာ
Moonwell price oracle	0xEC942bE8A8114bFD0396A5052c36027f2cA6a9d0	Basescan မှာ “Moonwell: Chainlink Oracle”
Attacker main wallet	0x719eae70d4A83f35bF82A2740699F5db84BE919D	MAMO ဝယ်၊ supply၊ donate၊ ချေး
Attacker ဒုတိယ wallet	0xD71dd9b6e634412713c47fe7ae02c628e338c384	Ethereum ပေါ်မှာ DAI ကိုင်ထားတယ်

Proof of Concept

လက်ရှိ bad debt ဒေါ်လာ ၉ သန်း လောက်ကျန်နေသေးတဲ့ live protocol ဖြစ်တဲ့ အတွက် အလုပ်လုပ်တဲ့ exploit script ကို ထုတ်မပြပါဘူး။ ဒါပေမယ့် Foundry test တစ်ခုရဲ့ အခြေခံပုံစံ shape ပေါ့နော် အဲ့တာကတော့ ရိုးရိုးပဲဆိုတော့ mechanics နားလည်အောင် ရေးပြထားတယ်။ ဒါကို လေ့လာဖို့အတွက် sketch လို့ပဲ သဘောထားလိုက်ပေါ့၊ copy paste လုပ်ပြီး run လို့ရတာ တော့မဟုတ်ဘူးပေါ့ဗျာ။

```
// ILLUSTRATIVE ONLY. Not tested, not runnable as is.
// Idea: on a Base fork before block ~50.5M, show that a plain transfer
// inflates the exchange rate and that the supply cap does not stop it.

function test_donationInflatesExchangeRate() public {
    uint256 before = mMAMO.exchangeRateStored();

    // Step 1: supply up to the cap the normal way. This passes mintAllowed.
    mamo.approve(address(mMAMO), 15_000_000e18);
    mMAMO.mint(15_000_000e18);

    // Step 2: a second mint past the cap reverts, as designed.
    vm.expectRevert("market supply cap reached");
    mMAMO.mint(1_000_000e18);

    // Step 3: a plain transfer of the same tokens does NOT revert.
    mamo.transfer(address(mMAMO), 53_000_000e18);

    // Step 4: every existing share is now worth ~3.68x more MAMO.
    uint256 after_ = mMAMO.exchangeRateStored();
    assertGt(after_, before * 3);

    // Step 5: account liquidity, and so borrow power, went up with it.
    (, uint256 liquidity, ) = comptroller.getAccountLiquidity(address(this));
    // liquidity is now roughly 0.5 * 68M MAMO * oraclePrice, not 0.5 * 20M
}
```

Oracle အပိုင်းကတော့ unit test နဲ့ reproduce လုပ်လို့ရမှာ မဟုတ်ဘူး။ အဲဒါက တကယ့် pool ပေါ်မှာ ဒေါ်လာ ၇.၅ သန်း တကယ်ဝယ်လိုက်တာ။ အပေါ်က onchain data တွေကပဲ အဲဒီအပိုင်းအတွက် သက်သေလို့ပြောလို့ရတယ်။

Impact

- **ဆုံးသွားတဲ့ ငွေ။** စုစုပေါင်း assets တွေကို ချေးသွားတာ $11,028,762 လောက်တန်ဖိုးရှိတယ်။ Liquidation လုပ်ပြီးတဲ့နောက်မှာတောင် bad debt $9,131,342 လောက် ကျန်တယ်။ cbBTC 53.60, WETH 585.17, USDC 2,345,281, wstETH 342.75။ Attacker ရဲ့ MAMO ဝယ်တဲ့ နေရာမှာ ကုန်တာကိုနုတ်ပြီး trace လုပ်လို့ရတဲ့ stablecoin အမြတ်က ဒေါ်လာ ၆.၇၈ သန်း

လောက်ထိရှိတယ်။ CertiK နဲ့ PeckShield ရဲ့ headline number ကတော့ဒေါ်လာ ၈.၇ သန်း ဝန်းကျင်လောက်ရှိတယ်။

- **ဘယ်သူတွေ ထိခိုက်လဲ။** Base ပေါ်က Moonwell မှာ cbBTC, WETH, USDC ဒါမှမဟုတ် wstETH supply လုပ်ထားတဲ့ လူတိုင်း ထိခိုက်နိုင်ပါတယ်။ Compound v2 market ထဲက bad debt ဆိုတာ တစ်ဦးတစ်ယောက်တည်းက ပိုင်တာမဟုတ်ဘူး အဲ့တော့ ဘယ်သူ့ဆီက ပြန်ယူမလဲဆိုတဲ့ သတ်မှတ်ထားတာမျိုးလည်းရှိဘူး၊ အဲ့တာကြောင့် protocol က treasury ထဲကနေ ပြန်မဖြည့်ရင် နောက်ဆုံး withdraw လုပ်တဲ့ supplier တွေက အဲ့ဒီ ဆုံးရှုံးမှုကို ခံရနိုင်ပါတယ်။ Mamo app သုံးတဲ့သူတွေလည်း ခဏလောက် USDC withdraw လုပ်လို့မရခဲ့ဘူး၊ ဘာလို့လည်းဆို app က deposit တွေကို Moonwell ထဲ ထည့်ထားလို့ပါ။

- **တစ်နှစ်စာ revenue ထက်တောင် ပိုကြီးတဲ့ ဆုံးရှုံးမှု။** Moonwell ရဲ့ တစ်နှစ်စာ fee ကိုတွက်ရင်က ဒေါ်လာ ၈.၆ သန်းဝန်းကျင်လောက်ပဲရှိတာဗျာ။ attacke တစ်ခု တစ်မနက်တည်းနဲ့ အဲဒါထက် ပိုကုန်သွားတယ်။

- **၁၁ လအတွင်း တတိယအကြိမ်။** ၂၀၂၅ နိုဝင်ဘာ၊ wrsETH oracle ရဲ့ ချွတ်ယွင်းမှုကြောင့်လို့ ပြောလို့ရမယ် အဲ့တာက bad debt ဒေါ်လာ ၃.၇ သန်းလောက် ကျန်ခဲ့တယ်။ ၂၀၂၆ ဖေဖော်ဝါရီ၊ governance proposal တစ်ခုက cbETH oracle ကို cbETH/ETH rate တစ်ခုတည်း သုံးအောင် မှားပြီး configure လုပ်မိလို့ cbETH ဈေးက $1.12 လောက် ဖြစ်သွားသေးတယ် liquidation bot တွေ ကကြိတ်သွားတာပေါ့။ အဲ့မှာလည်း bad debt ဒေါ်လာ ၁.၇၈ သန်းလောက် ကျန်ခဲ့တယ်။ အဲဒီ incident က pull request တွေကို AI coding assistant နဲ့ ရေးထားလို့ ပိုဝေဖန်ခံခဲ့ရတယ်။ ၂၀၂၆ ဩဂုတ်လ အခုဟာဖြစ်တာပေါ့နော်။ ဖေဖော်ဝါရီ incident မှာ ဆုံးရှုံးခဲ့တဲ့ supplier တွေ

က ဘယ်သူက သူတို့ကိုပြန်ပေးရမလဲဆိုတာ ငြင်းနေတုန်းမှာပဲ MAMO attack က ထပ်ဖြစ်တာ။ မုဆိုးမသွားရာ မိုးလိုက်လို့ရွာပေါ့ဗျာ။

- **Token ဈေးတွေ။** WELL က ၂၄ နာရီအတွင်း ၁၃ ရာခိုင်နှုန်းလောက် ကျသွားတယ်။ MAMO က ၉ ရာခိုင်နှုန်းလောက်ပဲ ကျသွားတယ်၊ attacker က ခုနက အဆ ၄၀ pump လုပ်ပြီး ရောင်းသွားတာနဲ့ ယှဉ်ရင် အဲဒါ များတော့များတယ်ဗျ။

- **အခုထိ exploit လုပ်လို့ ရသေးလား။** အခုချိန်ထိတော့ မရသေးဘူး။ Borrow cap အားလုံး 1 wei ဖြစ်နေတော့ Base ပေါ်မှာ ဘယ်သူမှ ဘာမှ ချေးလို့ မရဘူး၊ attacker ပဲဖြစ်ဖြစ် ဘယ်သူပဲဖြစ်ဖြစ် ဘာမှ လုပ်လို့မရတော့ပါဘူး။ ဒါပေမယ့် အဲဒါက သွေးတိတ်အောင် ကြိုးစည်းထားတာလောက်ပဲ၊ ပျောက်အောင်ဆေးကုတာ မဟုတ်ဘူး။ အဓိက အားနည်းတဲ့ Code နဲ့ oracle design ကတော့ ဘာမပြောင်းသေးဘူး အခုထိ တက်ရှိနေတုန်းပဲ။

The Fix

၂၀၂၆ ဩဂုတ် ၃၁ အထိ Moonwell က post mortem ထုတ်ပြီးပေမယ့် remediation plan မထုတ်သေးဘူး။ ဆိုတော့ အခုအောက်မှာ ရေးထားတာက ကျွန်တော့် အမြင်အရ ဘယ်လို fix လုပ်သင့်လည်းဆိုတာကို ၊ နှစ်ပိုင်း ခွဲထားပြီးပြောထားတာပဲဗျာ။ ကျွန်တော့်အမြင် သက်သက်ပါပဲ။

တစ်ဝက် : Donation trick က collateral ဖန်တီးတာကို တားမယ်

Clean အဖြစ်ဆုံး fix က protocol က ဘယ်တော့မှ လက်မခံခဲ့တဲ့ token တွေကို ထည့်မတွက်တော့ဖို့ပဲ။ Compound v2 fork တွေက ဒီပြဿနာအတွက် နည်း၂ နည်း သုံးတယ်။ ပထမတစ်ခုက `balanceOf` ကို ဖတ်ပြီးတော့ underlying token amount ကိုတွက်မယ့်အစား protocol အတွင်းထဲမှာ ကိုယ်ပိုင် underlying token (internal ledger) ကိုသုံးတာဖြစ်ပါတယ်။

```
// ILLUSTRATIVE. Track what was actually deposited, not what happens to be in the
contract.
- function getCashPrior() internal view virtual override returns (uint) {
-     return EIP20Interface(underlying).balanceOf(address(this));
- }
+ uint256 internal trackedCash;
+
+ function getCashPrior() internal view virtual override returns (uint) {
+     return trackedCash;                    // donations do not move this
+ }
+ // update trackedCash inside doTransferIn / doTransferOut only
```

ဒုတိယတစ်ခုက supply cap ကို mint လုပ်တဲ့အချိန်မှာပဲ စစ်တာ မဟုတ်ဘဲ valuation ဆိုတဲ့ တန်ဖိုးတွက်တဲ့အချိန် မှာပါ အလုပ်လုပ်အောင် လုပ်တာ။ Market ထဲမှာရှိတဲ့ underlying token ပမာဏက သတ်မှတ်ထားတဲ့ cap ခွင့်ပြုတာထက် ပိုများနေတယ်ဆိုရင် ပိုနေတဲ့ ပမာဏကို collateral အဖြစ် ထည့်မတွက်သင့်တော့ဘူး။

```
// ILLUSTRATIVE. In getHypotheticalAccountLiquidityInternal, cap the value a market can
back.
```

```
+ uint cap = supplyCaps[address(asset)];
+ uint held = asset.getCash() + asset.totalBorrows() - asset.totalReserves();
+ if (cap != 0 && held > cap) {
+     // scale this market's collateral down so the whole market is worth at most `cap`
tokens
+     vars.exchangeRate = mul_(vars.exchangeRate, Exp({mantissa: cap * 1e18 / held}));
+ }
```

ဒီနှစ်ခုထဲက တစ်ခုခု ပြင်လိုက်ရင်တောင် ၃.၆၈ ဆ multiplier ကို တားနိုင်ခဲ့မှာပေါ့ဗျာ။

နောက်တစ်ခု : Thin token က ကိုယ့်ဈေးကိုယ် ပြန်သတ်မှတ်တာကို ခွင့်မပြုဖို့

- **Thin asset တွေအတွက် အလွယ် manipulate လုပ်မလို့မရတဲ့ ဈေး သုံးသင့်တယ်။**

  time weighted average ဖြစ်ဖြစ်၊ တစ်နာရီကို သတ်မှတ်ရာခိုင်နှုန်းထက် ပိုမတက်/ မကျအောင် ကန့်သတ်ထားတဲ့ feed ဖြစ်ဖြစ်၊ နှစ်ခုလုံး ဖြစ်ဖြစ်လုပ်သင့်တယ်။ နှစ်နာရီအတွင်း အဆ ၄၀ တက်သွားတာကို လက်ခံမယ့်အစား wrong price အဖြစ် reject သင့်တယ်။

- **Collateral ကို မျှော်လင့်ချက်နဲ့ မဟုတ်ဘဲ liquidity ပေါ်မူတည်ပြီး သတ်မှတ်သင့်တယ်။**

  တစ်နေ့ trading volume ဒေါ်လာ တစ်သန်းလောက်ပဲ ရှိတဲ့ token က borrow ceiling ဒေါ်လာ သိန်းဂဏန်း အနိမ့်ပိုင်းလောက်ပဲ ထားသင့်တယ်၊ ဒါမှမဟုတ် collateral factor ကို 0 ထားလိုက်သင့်တယ်။ Aave ရဲ့ isolation mode နဲ့ debt ceiling တွေက ဒီလို risk မျိုးအတွက်ပဲ ရှိနေတာ။

- **Donation တွေကို စောင့်ကြည့်သင့်တယ်။** `Mint` event မပါဘဲ `balanceOf(mToken)` တန်ဖိုး ခုန်တက်လာရင် အချက်ပေးမယ့် monitor တစ်ခုသာ ရှိခဲ့ရင် နောက်ဆုံး borrow မတိုင်ခင် ရှစ်မိနစ် အလိုလောက် မှာကတည်းက 09:19:59 transfer ကို flag လုပ်နိုင်ခဲ့မှာ။

အခုပြောထားတာတွေက Moonwell က တကယ် merge လုပ်ထားတဲ့ code မဟုတ်ပါဘူး။ ဘယ်လို ပုံစံ ပြင်သင့်လည်းဆိုတာ ကျွန်တော့်အမြင်အရ ပြန်ပြောတဲ့သဘောပဲဗျာ။ အဲ့တော့ လိုတာ ပိုတာရှိကောင်းရှိ

မှာပေါ့။ Moonwell က တကယ် သူတို့ code changes တွေ ထုတ်လာရင် ဒီ post ကို ပြန် update လုပ်ပါမယ်။

Takeaways

1. **“Code မ hack ခံရဘူး” ဆိုတာနဲ့ code က အကုန်အဆင်ပြေတယ်လို့ မဆိုလိုဘူး။** Function တိုင်းကတော့ ရေးထားတဲ့အတိုင်း သူ့အလုပ်သူ လုပ်ခဲ့တာပဲ။ ပြဿနာက design မှာဖြစ်တာ။ attacker ကို input တွေ ရွေးခွင့်ပေးထားတာ။ Lending market ကို audit လုပ်တဲ့အခါ collateral formula ကို audit လုပ်သလောက် formula ရဲ့ input တွေကိုလည်း သေချာ audit လုပ်သင့်တယ်။

2. **Compound v2 fork တိုင်းမှာ donation quirk ရှိတယ်။** `getCashPrior` က `balanceOf(address(this))` ဖတ်တာ Compound, Moonwell, Venus, Benqi နဲ့ တခြား compound V2 fork တွေ တော်တော်များများ မှာ တွေ့ရတယ်။ ပုံမှန်အားဖြင့်တော့ အန္တရာယ် မရှိဘူး။ ဒါပေမယ့် Market တစ်ခုမှာ account တစ်ခုတည်းက share အများစု ကိုင်ထားတဲ့အချိန်ကျရင်တော့ အန္တရာယ် ရှိလာပြီ၊ အဲဒါက liquidity နည်းပြီး trading နည်းတဲ့ token တွေမှာ ဖြစ်တတ်တယ်။ getCashPrior နဲ့ balanceOf(address(this)) ကို code ထဲမှာ grep လုပ်ပြီး ရှာကြည့်ပါ။

3. **Mint လုပ်မှပဲ စစ်တဲ့ supply cap က limit လုပ်တာလို့ခေါ်လို့ မရဘူး အကြံပေးတာလို့ပဲပြောရင်မှန်မယ်။** Risk model က “protocol က ဒီ token သန်း ၂၀ ထက် ပိုမကိုင်ဘူး” လို့ limit လုပ်ထားရင် deposit လုပ်တဲ့အချိန်တင်ပဲ စစ်တာ မဟုတ်ဘဲ token ရဲ့ တန်ဖိုး သုံးတဲ့နေရာတိုင်းမှာ အဲ့ဒီ limit ကို enforce လုပ်ရမှာ။

4. **Liquidity နည်းတဲ့ token တွေကို တန်ဖိုးကြီးတဲ့ loan တွေအတွက် collateral မလုပ်သင့်ဘူး။** တကယ့်အမှားက market cap ဒေါ်လာ ၆ သန်းပဲရှိတဲ့ token ကို cbBTC နဲ့ WETH အတွက် ၅၀ ရာခိုင်နှုန်းနဲ့ collateral အဖြစ် list လုပ်ခဲ့တာ။ Underlying asset ကို ဒေါ်လာ ၂ သန်းရှိတဲ့ wallet တစ်ခုက အဆ ၄၀ လောက်ထိ manipulate လုပ်လို့ရရင် oracle quality က သိပ်အရေးမကြီးတော့ဘူး။

5. **၁၁ လအတွင်း oracle incident သုံးခုဆိုတာ process ပြဿနာ။** wrsETH, cbETH, MAMO failure တွေက mechanism ချင်း မတူပေမယ့် အမျိုးအစားချင်း အတူတူပဲ။ Protocol က ယုံလိုက်တဲ့ ဈေးက မှားနေတာ။ အဲ့တာကြောင့် token listing နဲ့ oracle အတွက် risk management ကို Solidity code review လုပ်သလိုပဲ အဲ့တာအတွက် review culture တစ်မျိုး လိုတယ်လို့ထင်တယ်။

6. **ပိုက်ဆံနောက် လိုက်ကြည့်။ You can see the whole story**

   Tornado Cash ကနေ ဝင်တယ်၊ CoW Swap၊ CCTP နဲ့ Base ကို၊ ၄၇ ကြိမ် token ဝယ်တယ်၊ donation နှစ်ခါလုပ်တယ်၊ ၁၈ ခါ ချေးတယ်။ CCTPv2 ကနေငွေပြန်ထုတ်တယ်၊ DAI နဲ့ သိမ်းထားတယ်။ BlockChain ပေါ်က transaction တွေကြည့်တတ်သွားရင် attack တစ်ခုလုံးကို ခဏလေးနဲ့ သိသွားလိမ့်မယ်။

Disclosure Timeline

Date (UTC)	Event
21 to 27 Aug 2026	Attacker funds up: 800 ETH from Tornado Cash, swapped to 1.95M USDC, bridged to Base.
27 Aug 2026, 08:43:13	First borrow, a 0.5 cbBTC test.
27 Aug 2026, 09:19:59 and 09:21:09	53.4M MAMO sent directly to the mMAMO contract. Exchange rate up 3.68x.
27 Aug 2026, 09:28:43	MAMO price feed peaks at $0.4313.
27 Aug 2026, 09:30:13	Last borrow. Gross out: $11,028,762. Liquidations begin 32 seconds later.
27 Aug 2026, 09:41:43 to 09:45:13	8.73M USDC bridged to Ethereum and swapped to DAI.
27 Aug 2026, ~10:00	CertiK and PeckShield flag the exploit publicly. Moonwell posts that it is investigating.
27 Aug 2026, 10:53:43	Borrow caps on all Base Core Markets set to 1 wei.
27 Aug 2026, 11:09:43	MAMO and WELL supply caps set to 1 wei.
27 Aug 2026	Mamo team states “Mamo was not hacked, and no Mamo contracts were compromised.”
30 Aug 2026	Anthias Labs publishes the post mortem on the Moonwell governance forum.
31 Aug 2026	No remediation or repayment plan published yet. Borrowing on Base still frozen.

Responsible disclosure အနေနဲ့ ပြောရရင် ဒီ post ထဲက အချက်အလက်တိုင်းက chain ပေါ်မှာ ဒါမှ မဟုတ် official post mortem ထဲမှာ public ဖြစ်ပြီးသားတွေပါ၊ protocol ကလည်း freeze ဖြစ်ပြီးသား ပါ။ ဆိုတော့ ဒီ post ထဲမှာ live weapon ဆိုတာ ဘာမှ မပါဘူး။

Sources

- Moonwell governance forum, Anthias Labs post mortem: https://forum.moonwell.fi/t/post-mortem-mamo-market-incident-on-base/2208

- Moonwell contract addresses: https://docs.moonwell.fi/moonwell/protocol-information/contracts
- Moonwell contracts at commit 3b32d53: https://github.com/moonwell-fi/moonwell-contracts-v2/tree/3b32d534778b854d31bd3555c90cc787b2072566/src
- Basescan, mMAMO market: https://basescan.org/address/0x2F90Bb22eB3979f5FfAd31EA6C3F0792ca66dA32
- Basescan, attacker main wallet: https://basescan.org/address/0x719eae70d4A83f35bF82A2740699F5db84BE919D
- Etherscan, attacker second wallet holding DAI: https://etherscan.io/address/0xD71dd9b6e634412713c47fe7ae02c628e338c384
- The Block: https://www.theblock.co/news/defi/2026-08-27-moonwell-investigates-base-lending-market-issue-412913
- The Defiant: https://thedefiant.io/news/hacks/moonwell-loses-8-7-million-to-mamo-price-manipulation-on-base
- Yahoo Finance: https://finance.yahoo.com/markets/crypto/articles/moonwell-lost-8-7-million-122819291.html
- CryptoDaily on the $9.13M shortfall: https://cryptodaily.co.uk/2026/08/moonwell-mamo-exploit-base-9-13m-borrower-obligations
- TechTimes on the three incidents: https://www.techtimes.com/articles/325839/20260827/moonwell-oracle-exploit-exceeds-full-annual-revenue-third-failure-11-months.htm
- Protos on the February cbETH incident: https://protos.com/defi-meet-claude-moonwells-vibe-coded-oracle-in-1-8m-blowup/
- Moonwell forum, cbETH remediation counter proposal: https://forum.moonwell.fi/t/counter-proposal-fund-cbeth-remediation-from-the-foundation-treasury-not-a-reserve-residual/2175