rekt.news

Rekt - Cream Finance - REKT

3-4 minutos

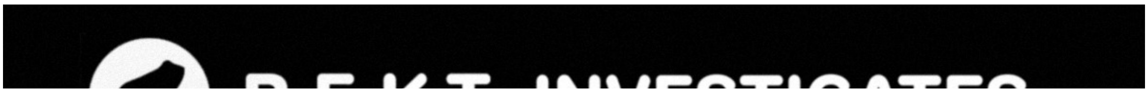
Inspector rekt back once again.

$18.8M lost to a ghostface killer, this time from an old school DeFi protocol.

Cream Finance was audited by Trail of Bits (one of the few auditors absent from our leaderboard) on Jan 28th 2021.

However, even the strongest audit becomes irrelevant once the protocol is changed.

On Feb 10th 2021, the Cream proposal to add the AMP token came into effect, and the loophole opened up.

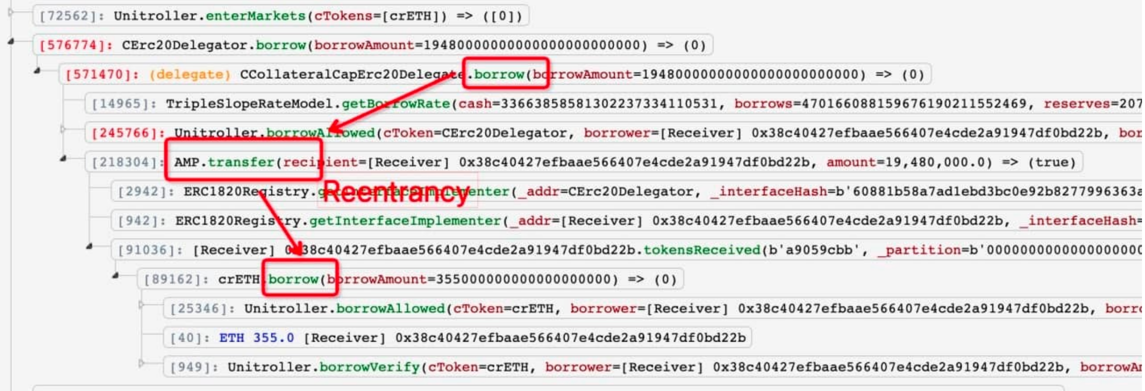


Credit: @peckshield

418,311,571 AMP tokens and 1,308.09 ETH were lost on the Cream Finance AMP token contract.

The AMP token contract implements ERC77-based ERC1820, which has the **_callPreTransferHooks** for reentrancy.

The reentrancy vulnerability within the AMP token contract allowed the exploiter to nest a second borrow() function inside the token transfer() before the initial borrow() has been updated:



```
[72562]: Unitroller.enterMarkets(cTokens=[crETH]) => ([0])
[576774]: CErc20Delegator.borrow(borrowAmount=19480000000000000000000000) => (0)
  [571470]: (delegate) CCollateralCapErc20Delegate.borrow(borrowAmount=19480000000000000000000000) => (0)
    [14965]: TripleSlopeRateModel.getBorrowRate(cash=33663858581302237334110531, borrows=470166088159676190211552469, reserves=207
    [245766]: Unitroller.borrowAllowed(cToken=CErc20Delegator, borrower=[Receiver] 0x38c40427efbaae566407e4cde2a91947df0bd22b, bor
    [218304]: AMP.transfer(recipient=[Receiver] 0x38c40427efbaae566407e4cde2a91947df0bd22b, amount=19,480,000.0) => (true)
      [2942]: ERC1820Registry.getInterfaceImplementer(_addr=CErc20Delegator, _interfaceHash=b'60881b58a7ad1ebd3bc0e92b8277996363a
      [942]: ERC1820Registry.getInterfaceImplementer(_addr=[Receiver] 0x38c40427efbaae566407e4cde2a91947df0bd22b, _interfaceHash=
      [91036]: [Receiver] 0x38c40427efbaae566407e4cde2a91947df0bd22b.tokensReceived(b'a9059cbb', _partition=b'0000000000000000000
        [89162]: crETH.borrow(borrowAmount=355000000000000000000) => (0)
          [25346]: Unitroller.borrowAllowed(cToken=crETH, borrower=[Receiver] 0x38c40427efbaae566407e4cde2a91947df0bd22b, borro
          [40]: ETH 355.0 [Receiver] 0x38c40427efbaae566407e4cde2a91947df0bd22b
          [949]: Unitroller.borrowVerify(cToken=crETH, borrower=[Receiver] 0x38c40427efbaae566407e4cde2a91947df0bd22b, borrowAr
```

Example exploit transaction **(one of 17)**

Attack contracts: A, B and exploiter wallet.

In the above example, the hacker:

1: Uses contract A to take a flash loan of 500 WETH to use as collateral on Cream, minting 24.17k crETH

2: Borrows 19.48M AMP against crETH

3: Exploits the reentrancy bug by inserting a further borrow() function into the token transfer, taking a further

355 ETH before the initial borrow() has been updated.

4: Creates contract B, which is funded with half (9.74M) of A's borrowed AMP

5: Contract B then liquidates part of A's loan, redeeming 187 WETH and transferring it back to contract A.

6: Contract A then uses the ETH borrowed via reentrancy to repay the remainder of the flashloan, leaving a surplus of 41 ETH and 9.74 AMP as profit for this transaction.

A similar process was used over 17 transactions, accumulating a total of almost 6k ETH.

At the time of writing, the stolen ETH (currently worth just over $18M) remains in the exploiter's address: 0xce1f4b4f17224ec6df16eeb1e3e5321c54ff6ede

Cream has never had a great reputation. Perhaps this is why many believed this to be their second or third time getting rekt

Although they were involved with the Alpha Finance incident, this is actually the first direct attack to hit Cream Finance.

Regardless of reputation, even time-tested protocols can be undermined by the integration of a vulnerable token.

However, as @muditgupta pointed out;

> ...seems like [Cream] would have been safe had they just added reentrancy protection on their borrow/lend function.

Can it all be so simple?

REKT serves as a public platform for anonymous authors, we take no responsibility for the views or content hosted on REKT.

donate (ETH / ERC20):

0x3C5c2F4bCeC51a36494682f91Dbc6cA7c63B514C

disclaimer:

REKT is not responsible or liable in any manner for any Content posted on our Website or in connection with our Services, whether posted or caused by ANON Author of our Website, or by REKT. Although we provide rules for Anon Author conduct and postings, we do not control and are not responsible for what Anon Author post, transmit or share on our Website or Services, and are not responsible for any offensive, inappropriate, obscene, unlawful or otherwise objectionable content you may encounter on our Website or Services. REKT is not responsible for the conduct, whether online or offline, of any user of our Website or Services.