Flash Loan 개요

소개

Flash Loan은 무담보 대출로 트랜잭션이 끝나기 전에 상환해야하며 이더리움 기반 탈중앙화 금융(Defi) 프로토콜로 제공된다. 2019년 마블 프로토콜(Marble Protocol)에 의해 이더리움 블록체인에 도입되었다. 마블은 전통적인 대출 기관들이 취하는 두가지 리스크와 싸우기 위해 플래시 론을 만들었는데 이는 다음과 같다.

1. 돈을 빌린 사람이 돈을 가지고 사라졌을 때
2. 유동성 부족

플래시 론은 이 두가지 리스크를 모두 완화시켜주며 다음과 같은 방식으로 작동한다.

- 대출을 받으려면 한번의 transaction이 필요하다.
- 트랜잭션이 끝날 때까지 대출받은 금액 이상을 대출자에게 상환하면 된다.
- 금액을 지불할 수 없는 경우 트랜잭션이 롤백되며, 이는 원자성 대출의 특성이다.

플래시 론은 블록체인에서만 가능하다. 즉, 중앙화 거래소는 플래시 론을 제공할 수 없다. 스마트 컨트랙트 플랫폼은 순차적으로 거래를 처리하지 않기 때문이다. 대출 기간이 블록 하나가 생성되는 시간인 15초 안에 진행된다. 이는 대출에 대한 트랜잭션이 승인되는 시간이기도 하다. 플래시 론은 암호화폐 가격이 다른 두 시장에서 차익거래 기회로부터 이익을 얻고자 하는 거래자에게 유용하다.

활용 사례

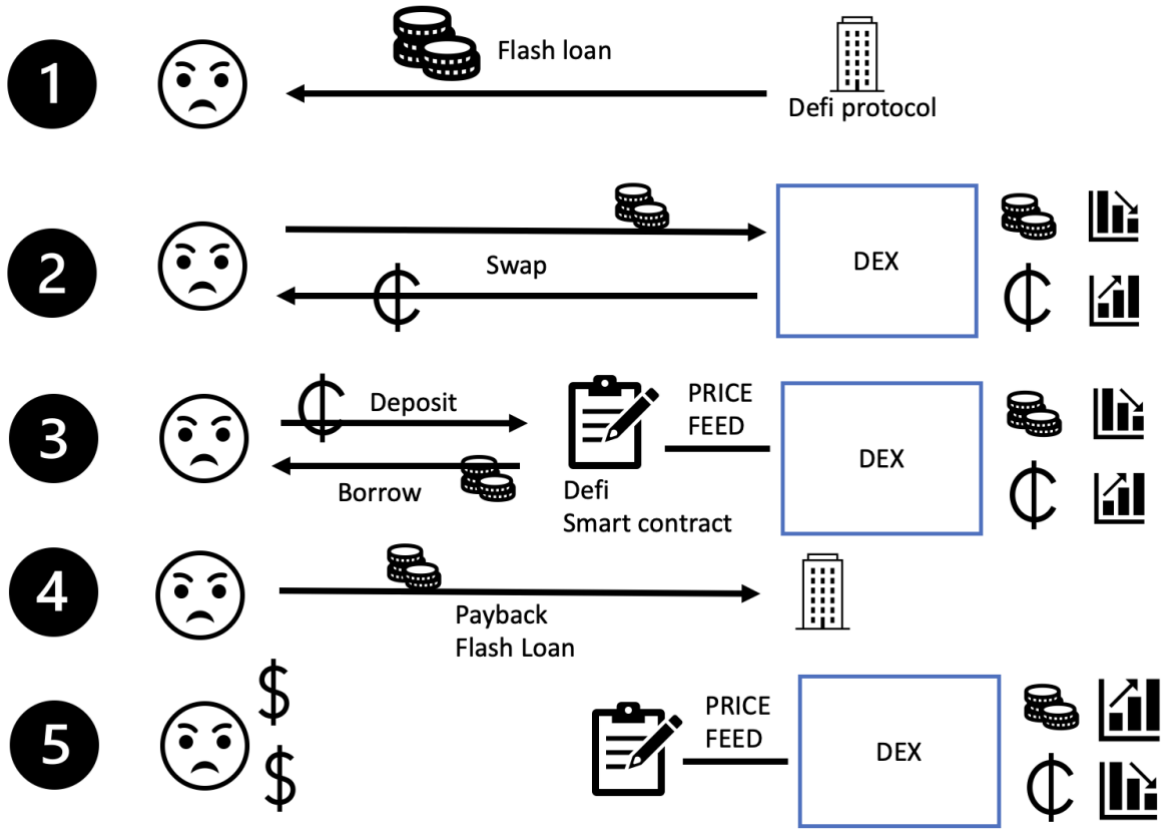
- 담보 스와핑: 담보 거래소를 이용해 다자간 대출 애플리케이션을 통해 대출을 받을 수도 있다. 컴파운드에서 DAI 대출을 받아 ETH 를 담보로 예치한 경우, ETH 담보를 DAI 플래시 대출을 통해 DAI 담보로 거래해야 한다. 이렇게 함으로써 대출한 DAI 금액의 균형을 맞출 수 있다.
- 차익 거래: 암호화폐 투자자들은 서로 다른 시장 간의 가격 차이로 돈을 벌 기회가 많은데, 암호화폐 차익 거래는 수익을 내기 위해서 한 두개의 다른 거래소에서 거래를 하는 과정이다. 코인을 낮은 가격에 산 다음 다른 거래소에서 높은 가격에 파는 것을 말한다.
- 거래 수수료 절감: 플래시 론을 통해 복잡한 트랜잭션 목록을 한 번에 집계할 수 있다. 이는 여러 단계가 필요했던 기존 거래를 개선시킴으로써 플래시 론 개념을 구현하여 최소한의 거래 수수료로 액세스할 수 있도록 해준다.
- 채무재융자: 투자자는 플래시 론을 담보 스왑뿐만 아니라 이렇게 붙은 금리 스왑에도 활용할 수 있다

토큰

이더리움은 채굴에 대한 보상으로 코인을 발행했다. 비슷하게 토큰이라는 개념이 있는데 이는 이더리움 표준인 ERC 에서는 대체 가능한 토큰 ERC-20 이 채택되어 있는데, 이 토큰은 기초 자산에 대한 가치와 권리를 나타냅니다. 스마트 컨트랙트는 이 토큰을 이용해 교환 수단으로

이용한다. 기존 코인과 다른 점은 코인은 자체 블록체인이 있는 반면 토큰은 없다. 이더리움이 코인 대신 토큰을 사용하는 이유는 이더리움이 단순한 암호화폐가 아니라 분산형 애플리케이션(dApp)을 구축하기 위한 플랫폼이기 때문.

Flash Loan Attack



대출받은 암호화폐를 이용해 공격 및 조작을 통해 이익을 취한 뒤 이 거래가 블록에 저장되기 전에 상환하는 방식으로 이루어진다. 대출받은 토큰으로 특정 탈 중앙화 거래소에서 가격을 조작해 예치 자산의 가치를 높인 뒤 거액의 대출을 받아 유동성을 탈취한다. 위의 그림에 따라 다음 과정을 설명한다.

1. 공격자는 Flash Loan을 지원하는 프로토콜에서 대량의 토큰 A를 대출한다.
2. 공격자는 DEX(탈중앙화거래소)에서 토큰 A를 토큰 B로 교환한다.
   이 과정을 통해 토큰 A의 가격은 낮아지고 B의 가격은 높아진다.
3. 공격자는 구매한 토큰 B를 DEX의 현물 가격을 사용하는 DeFi 프로토콜에 담보로 예치하고 조작된 현물 가격을 사용하여 훨씬 더 많은 A를 대출받는다.
4. 3에서 빌린 토큰 A로 1에서 대출한 토큰 A를 상환한다.
5. 차액으로 수익을 창출한다.

DEX에서는 토큰 A와 B의 현물 가격이 실제 시장 가격으로 조정이 되고 Defi 프로토콜은 담보가 부족한 상태로 남게된다.

DFX Finance hack 개요

DFX 소개

DFX는 이더리움 기반의 탈중앙화 거래 프로토콜로, 실제 세계의 환율 정보를 이용하여 USDC, CADC, EURS, XSGD등과 같은 달러 기반 안정화 코인을 최적화한 동적 조정형 결합 곡선을 사용한다. 조정형 결합 곡선은 토큰의 가격을 결정하는 수학적 모델 중 하나. 이 모델에서는 토큰 공급량과 수요에 따라 가격이 결정된다. 즉, 토큰의 가격은 공급과 수요의 상대적인 비율에 따라 결정된다. 이러한 결합 곡선을 동적으로 조정한다는 것은 시장 상황에 따라 토큰의 가격을 적절하게 조정하여 안정성을 유지하려는 것입니다. 예를 들어, 안정화 코인의 경우에는 달러와 1:1로 연동되어 있으므로, 달러 가치의 변동에 따라 안정화 코인의 가격도 함께 조정되어야 한다. 이러한 가격 조정은 동적으로 이루어져야하며, 이를 위해 조정형 결합 곡선을 사용한다. tokenA_balance(p) * tokenB_balance(p) = k

상수 공식은 AMM의 고유한 구성 요소로, 서로 다른 AMM이 작동하는 방식을 결정한다.

사건 개요

2022년 11월에 발생한 DFX Finance Hack은 stable coin을 교환하기 위해 DEX를 공격했다. 총 750만 달러의 토큰을 DEX에서 훔쳤다.

트랜잭션 hash: 0x6bfd9e286e37061ed279e4f139fbc03c8bd707a2cdd15f7260549052cbba79b7

공격자 Contract: 0x14c19962E4A899F29B3dD9FF52eBFb5e4cb9A067

DFX Contract: 0x6cFa86a352339E766FF1cA119c8C40824f41F22D

EtherScan:

https://etherscan.io/tx/0x6bfd9e286e37061ed279e4f139fbc03c8bd707a2cdd15f7260549052cbba79b7

Transaction trace:

https://openchain.xyz/trace/ethereum/0x6bfd9e286e37061ed279e4f139fbc03c8bd707a2cdd15f7260549052cbba79b7

공격방식

Flash Loan은 기본적으로 사용자에게 토큰을 빌려주기 전과 상환한 후의 토큰의 balance를 비교하여 사용자가 토큰을 제대로 상환했는지 확인한다. 토큰을 빌리기 위해서는 DFX의 flash 함수를 호출하여야한다. Flash 함수 내부에 토큰을 빌려주고 flashCallback 함수 즉, 사용자가 정의하는 함수를 호출하게 된다. 여기서 사용자는 빌린 토큰을 어떻게 사용할지 구현하게 된다. 이후 사용자는 토큰을 사용하고 나서 토큰을 갚게 된다. 공격자는 flashCallback 함수 내부에서 deposit을 호출하였다. 이후 공격자는 withdraw를 통해 deposit한 토큰을 인출할 수 있게 된다.

원인

기본적으로 이전에 언급했듯이 flash함수는 토큰의 balance를 비교하여 사용자가 제대로 상환했는지 확인하기에 컨트랙트는 공격자가 토큰을 deposit으로 넣어도 제대로 상환했다고 여기게 된다.

컨트랙트는 토큰을 deposit을 통해서 넣었는지 transfer을 통해서 넣었는지 확인하지 않은 문제가 있으며, 해당 컨트랙트로 재진입할 수 없는 lock이나 modifier를 설정하지 않았다.

Mythril에서 탐지 해결방안

기존 탐지 못한 원인

기존 mythril이 해당 컨트랙트로 다시 들어올 때를 탐지하는 경우는 취약점 Reentrancy가 있다. Flash Loan attack은 같은 함수로 재진입하는 것과 다르게 다른 컨트랙트에 구현된 deposit 함수로 진입하는 것이 문제이다.
또 mythril은 단일 컨트랙트의 취약점을 탐지하는 분석 도구이므로 다중 컨트랙트 사이의 트랜잭션의 정확성은 낮은 편이다.

해결방안

Flash Loan Attack을 해결할 수 있는 새로운 모듈을 개발하거나, 여러 확장이 가능하게 해당 컨트랙트의 다른 함수로도 돌아오는 경우에 Flash Loan Reentrancy 취약점을 도출한다.

Sturdy 개요

Sturdy 소개

Sturdy는 사용자가 높은 안정적인 코인 수익률을 얻거나 무이자 대출을 받을 수 있는 새로운 종류의 Defi 대출 프로토콜이다. Sturdy는 자신의 서비스의 특징으로 편리하게 돈을 무담보로 빌릴 수 있고, 높은 수익률을 가져다 준다고 소개하고 있다.
Sturdy는 사용자가 대출자나 대출자의 역할을 맡을 수 있게 하고, 대출자는 안정적인 코인을 예치에 유동성을 공급하고, 대출자는 이 자금을 빌려 콘벡스, 밸런서 등 화이트리스트 프로토콜에 투자한다. 이러한 투자로 인한 farming rewards 은 대출자와 대출자 사이에 분할된다. 예를 들어, 대출자는 Convex의 FRAXBP를 담보로 예치하고, USDC를 빌리고, 그 USDC를 Convex의 FRAXBP 풀에 투자할 수 있다. 그들은 CRV 보상의 일부를 얻고 나머지는 대출자에게 돌아갈 것이다. 다른 대출 프로토콜에서, 대출자는 대출자들이 지불한 이자로부터 이익을 얻는데 Sturdy는 대출자들에 의해 만들어진 farming profits으로부터 수익을 얻는다.

사건 개요

Defi protocol인 Sturdy Finance는 442 Ether를 손실을 봤는데 이는 거의 800,000 달러의 가치(23년 6월 12일 환율)가 있는 보안 악용으로 착취되었다.
23년 6월 12일 블록체인 보안업체 펙 쉴드는 Sturdy Finance에게 가격조작과 관련된 transaction을 알려주면서 경고하였다. 이후 Sturdy Finance는 해킹을 인정하고 사용자들의 안전을 위해 시장을 일시 중지하였다.

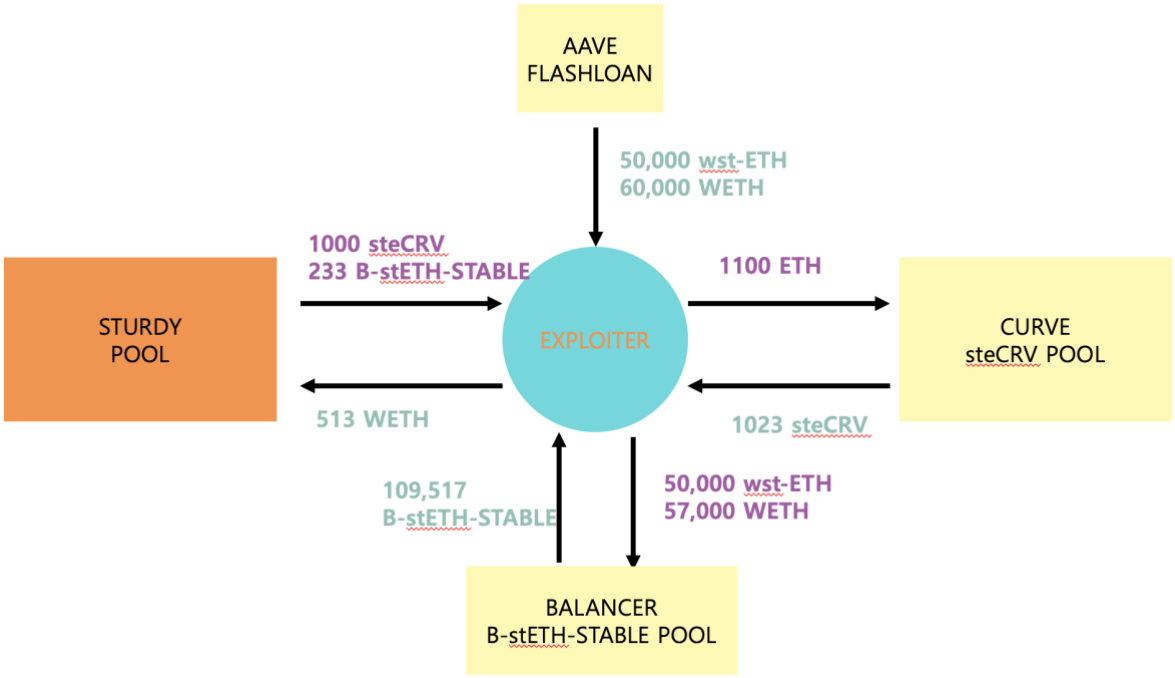
이 공격자는 Defi protocol에서 자금을 인출할 때 흔히 발생할 수 있는 취약점인 Reentrancy의 일부인 Flash loan을 이용한 공격이라고 보안 업체는 밝혔다. 이 방법을 통해 공격자들은 초기 함수 호출이 완료되기 전에 단일 트랜잭션에서 함수를 반복적으로 호출하는 기능을 제공한다.

공격자는 이전 트랜잭션이 실행되기 전에 B-stETH-STABLE 풀을 반복적으로 호출하여 풀의 가격 오라클이 오작동하고 그 가격이 3배 증가되었다. 공격자는 Sturdy에게 빌리기 위해 B-stEth-STABLE를 담보로 사용하였다. 가격이 오르자 공격자는 스트루들의 풀에서 담보물을 회수하여 그들의 담보의 실제 가치는 부풀려진 금액의 3분의 1이므로 해커는 그 차액으로부터 이익을 얻을 수 있다.
결론 - Price oracle이 오작동하도록 하였다.

Tx deatail:
https://etherscan.io/tx/0xeb87ebc0a18aca7d2a9ffcabf61aa69c9e8d3c6efade9e2303f8857717fb9eb7
Tx trace:
https://explorer.phalcon.xyz/tx/eth/0xeb87ebc0a18aca7d2a9ffcabf61aa69c9e8d3c6efade9e2303f8857717fb9eb7
Tx hash: 0xeb87ebc0a18aca7d2a9ffcabf61aa69c9e8d3c6efade9e2303f8857717fb9eb7

Attacker address: 0x1E8419E724d51E87f78E222D935fbbdeb631a08B
Attack Contract: 0x0B09c86260C12294e3b967f0D523B4b2bcdFbeab
Sturdy Exploited Contract: 0x46beA99d977F269399fB3A4637077bB35F075516

공격방식



1. Attacker took a floashLoan of 50,000 wstEth and 60,000 WETH from AAVE.
2. Attacker minted 1023 steCRV tokens using Curve steCRV Pool by depositing 1100ETH
3. 109,517 B-stETH-STABLE tokens were also minted by depositing 50,000 wstETH and 57,000 WETH in Balancer Pool.
4. Attacker borrowed 513 WETH from Sturdy Lending Pool using two tokens as collateral. (1000 steCRV & 233 B-stETH-STABLE)
5. Increase the price of the B-stETH-STABLE, so that steCRV can be set as non collateral. Since now the price of B-stETH-STABLE is increased, PoolBalances.sol/exitPool() is called.

   This function calls PoolBalances._joinOrExit() function which is vulnerable to read only vulnerability and does an external call indirectly.

   further call-graph is as follows:
   PoolBalances._joinOrExit() -> PoolBalances._callPoolBalanceChange() -> AssertTransferHandler._sendAsset() -> Address.sendValue()
   sendValue() is the function using the solidity native call.
6. The call made by the function then used by fallback of the malicious contract to call Sturdy contract LendingPool.setUserUseReserveAsCollateral() to set the steCRV an non-collateral as after the price manipulation, the protocol considers that only 233 B-stETH-STABLE are

able to cover the debt.

This function then calls the read function SturdyOravle.getAssetPrice() which gives false output due to the reentrancy present in the function we discussed above The false output is due to the fact that the value comes directly from the Balancer PoolTokens.getPoolTokens() contract.

7. Withdraw the 1000 steCRV tokens from pool.
8. Now, since the price of B-stETH-STABLE has come to normal, liquidate the position in Sturdy Pool with 236 WETH to get back 233 B-stETH-STABLE
9. Give the FlashLoan back.

Steps 3-8 are repeated 5times and are contained in the exploiter contract as a single function maybe names as "yoink()" This function is called 5 times and contains the logic for steps 3-8.

원인

Github:
https://github.com/balancer/balancer-v2-monorepo/blob/554b907aa49391210db7a6b032c48a7e24455ca9/pkg/vault/contracts/PoolBalances.sol#L132

Balancer의 PoolBalances.sol Smart Contract에 read-only Reentrancy때문인데, 함수 자체에는 noReentracnt modifier로 보호된 것처럼 보이지만 함수 흐름에 있는 _joinOrExit() 함수때문에 외부 호출이 있어 공격이 발생했다.

Fei Rari 개요

Fei Protocol 소개

Fei protocol은 Fei Labs Inc.가 개발한 완전히 분리된 분산된 스테이블 코인 프로젝트이다. 이 프로젝트는 다양한 혁신을 구현하여 안정적인 동전의 개념을 개선하는 것을 목표로 한다. Uniswap(이더리움의 블록체인에서 가장 큰 분산형 거래소) 유동성 풀과 그것의 토종 스테이블 코인인 FEI를 미국 달러 환율에 가깝게 유지하기 위해 PCV(Protocol Controlled Value)라고 불리는 안정성 메커니즘을 사용한다. 또한 TRIBE라는 native Governance token을 사용한다. 이 프로젝트는 유니스왑에 FEI/ETH 유동성 풀을 형성하며, 이는 즉시 유동성 2차 시장이 있는 스테이블 코인을 제공. 페이 프로토콜 자체는 스테이블 코인의 주요 유동성 공급자이며, 토큰의 인플레이션을 이용하여 제 3 자를 끌어들이는 다른 알고리즘 스테이블 코인과 차별화된다.

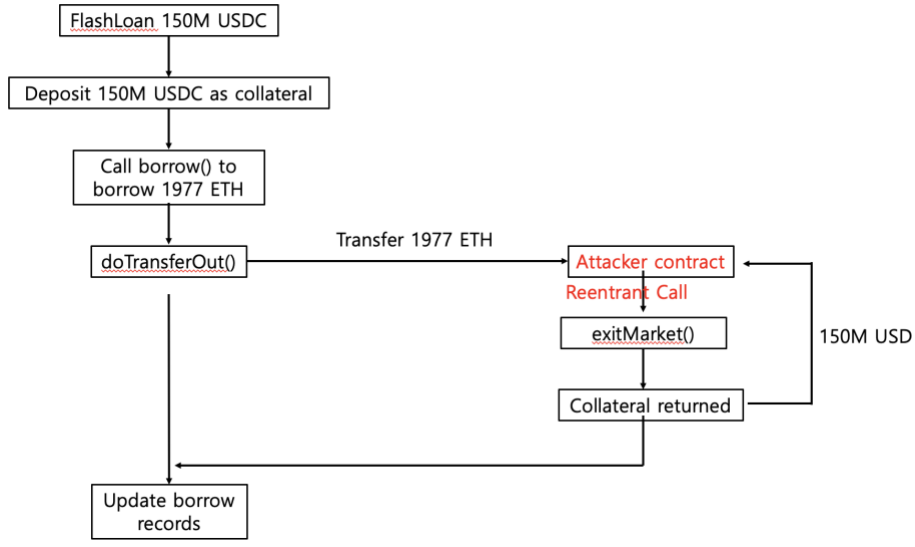
Rari Capital 소개

Rari Capital의 Defi protocol 제품은 기술 사용자와 비기술 사용자, 위험 프로파일이 높거나 낮은 사용자, 초보 사용자 및 전문 사용자 간의 Defi 격차를 해소하고자 합니다. Rari의 서비스는 사용자 커뮤니티에 의해 형성되고 관리됩니다. Rari의 거버넌스 토큰인 RGT는 온체인 투표 및 제안 프로세스에 사용되며 사용자가 유도엉 풀 기여도에 대한 수익률을 생성할 수 있는 주요 수단 역할을 한다.

사건 개요

2022년 4월 30일 Fei Protocol은 다양한 Rari Fuse 풀에 대한 공격을 인지하고 조사 중이라고 발표하였다. 보고된 총 손실은 약 80,000,000달러였다. 그들은 추가 손실을 최소화하기 위해 모든 차입을 중단하고 공격자에게 사용자 자금을 반환하기 위해 1천만 달러를 공개적으로 제공했다. 이로 인해 FEI는 최대 8편만 달러의 Reentrancy 해킹 중 하나가 되었다.

공격방식



1. Attacker flash loaned 150,000,000USDC and 50,000 WETH
2. Deposited 150,000,000 USDC as collateral into fUSDC-127 contract for loans, which is a fork of culnerable smart contract of Compound protocol.
3. The attacker borrows 1,977 ETH via the "borrow()" function
4. However, the "borrow()" function does not follow the check-effect-interaction pattern and transfers ETH to the attacker's contract before updating the attacker's borrow records.
5. Therefore, with the attacker's borrow record not updated, the attacker made a reentrant call to "exitmarket()" that allows the attacker to withdraw his collateral(150M USDC)
6. Attacker repeated the steps on multiple other tokens.
7. Finally, the attacker repaid the flashloan and transferred the rest as profit.

원인

Github:
https://github.com/Rari-Capital/fuse-v1/blob/b48986d8ece25e6a9688449a4e2cb394e58163c4/src/CToken.sol#L712

이 공격은 Fei protocol의 설계 결함으로 인해, 체크 효과 상호 작용패턴을 따르지 못해 공격자가 borrow record가 업데이트되기 전에 reentrant call을 할 수 있게 되었다.

```
812         doTransferOut(borrower, borrowAmount);
813
814         /* We write the previously calculated values into storage */
815         accountBorrows[borrower].principal = vars.accountBorrowsNew;
816         accountBorrows[borrower].interestIndex = borrowIndex;
817         totalBorrows = vars.totalBorrowsNew;
818
819         /* We emit a Borrow event */
820         emit Borrow(borrower, borrowAmount, vars.accountBorrowsNew, vars.totalBorrowsNew);
821
822         /* We call the defense hook */
823         // unused function
824         // comptroller.borrowVerify(address(this), borrower, borrowAmount);
825
826         return uint(Error.NO_ERROR);
827     }
```

doTransferOut()은 borrow record가 업데이트 되기 전에 호출된다. doTransferOut() 기능은 low-level call을 통해 ETH를 receiver에게 전달한다.

```
136     function doTransferOut(address payable to, uint amount) internal {
137         // Send the Ether and revert on failure
138         (bool success, ) = to.call.value(amount)("");
139         require(success, "doTransferOut failed");
140     }
```

따라서 공격자는 fallback() 함수에서 exit()을 만들기 위해 reentrant call을 할 수 있다.