AnySwap多链路由V3攻击

原因：随机数使用不当，导致私钥被成功推导

介绍：BSC上的V3路由器MPC账户下存在两个v3 router交易，这两个交易具有相同的R值签名，攻击者可以反推出MPC账户的私钥。

反推：知道两次交易的R值，知道两次签名的原始数据，能反推出随机数种子，可以从地址中推出公钥，通过脚本即可推出MPC地址的私钥

分析：

区块链账户权限是通过数字签名实现的，即用户用私钥对交易数据签名来完成账户资产转移。目前区块链中广泛使用的数字签名是ECDSA，同一数据会有多个合法的签名，这是由于ECDSA在每次签名过程中都会引入随机数k，R值就是通过k计算得到的。

假设有两个签名为（R，s1），（R，s2），那么根据ECDSA算法：

$$s_1 = k^{-1}(m_1 + sk * R)$$
$$s_2 = k^{-1}(m_2 + sk * R)$$
$$R = k\times G$$

m1, m2是交易数据，sk是账户私钥，s1，s2，R是签名数据，G是椭圆曲线上的参考点，因此

$$sk = (s_2 m_1 - s_1 m_2)/(R * s_1 - R * s_2)$$

由于在AnySwap攻击中，黑客发现了两次交易过程中的签名使用了相同的随机数k，所以黑客很容易就可以计算出私钥sk。

产生漏洞的原因：

1. 没有采用真正的MPC（Muti-party Computation），由单一个体产生随机数k，造成k重复出现
2. 代码在实现上出现重大问题，但发生在一个严谨的开发团队看来是不应该出现的
3. AnySwap将R值相同的原因归因于参与的MPC节点数量不足，具有误导性，因为如果签名过程中随机数是多个节点通过多方计算共同决定，那么仅 20 个节点也同样能够产生安全随机数

Crv的业务逻辑缺陷

1.Mochi项目方利用其持有的大量MOCHI打印出4600万枚USDM，然后去Curve上兑换处4600万枚DAI

（Mochi是一个跨链自治算法借贷协议，可以实现无门槛、基于特定条件的抵押资产上市。Mochi 用户可以通过 Mochi Vault 智能合约，利用列出的抵押品来铸造 USDM 稳定币。）

2.项目方使用DAI购买了大量Convex Finance的治理代币CVX，获得投票权

（Convex Finance就是一种旨在帮助Curve LPs和CRV代币持有者实现收益最大化，即该协议试图获得尽可能多的CRV，以便能够将更多的CRV锁定换取veCRV，这样就能获得CRV加速奖励，从而为存款人带来更高的收益。）

3.利用CVX驱使Convex的海量veCRV（治理权）为自己的池子投票，获得大量CRV补贴，这样挖矿的APR就会持续上涨，从而吸引到更多的流动性提供者参与到USDM流动性池中，池子变得深厚

4.项目方轻而易举的通过Curve将原先流动性极差的USDM兑换成USDT、DAI等真金白银，完成收割