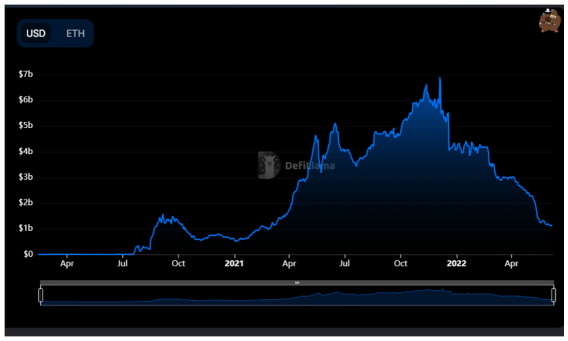
Yield Aggregator 간 Vault Strategy 비교 분석

-Yearn Finance를 중심으로-

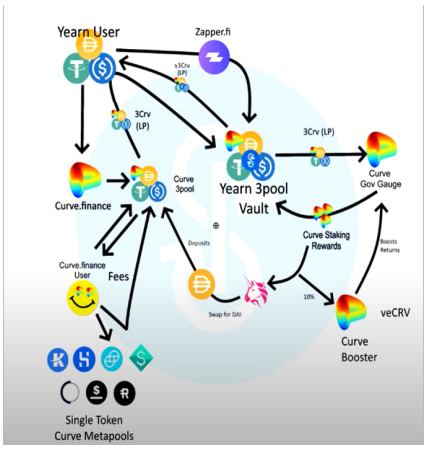
DEFI 6조
김민서 김채린 이건우 이명준

다양한 일드 어그리게이터에 대한 분석을 진행하였고, 특히 yearn finance의 vault 구조에 집중해서 다양한 어그리게이터들의 볼트를 비교하였다. 시중의 많은 수의 이자농사 디파이 프로토콜과 어그리게이터들이 출시되어 있는 반면, 각각의 프로토콜에 대한 분석, 볼트의 전략 , 그리고 각 어그리게이터의 차별점에 대해서는 아직 제대로 짚어진 적이 없다고 판단하여, 이번 프로젝트를 통하여 여러 서비스간의 특색을 파악하고 효율적인 이자농사 방법을 알아보고자 한다.

연파이낸스 개요

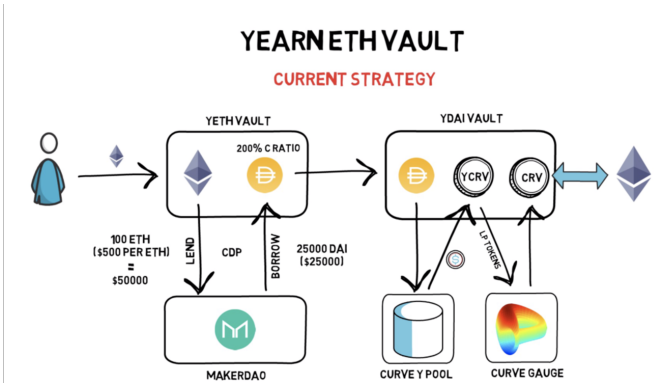


연파이낸스는 다양한 디파이 프로토콜을 효율적으로 사용할 수 있게 해주는 디파이 어그리게이터이다. 현재 해당 프로토콜 내에 잠겨있는 자금의 규모는 약 11억달러 정도로, 지난 12월에 60억이었던 것에 비하면 큰폭으로 하락한 상황이다. 프로토콜 자체의 결함이있어서 예치금액이 줄었다고 보기는 어렵고, 전반적으로 디파이 약세장이 오면서 디파이에 대한 투심이 약해져 하락한 것으로 판단된다.



연파이낸스의 메인 프로덕트는 vault라고 불리우는 스마트 컨트랙트인데, 이용자는 볼트에 자금을 예치해 이자농사를 진행할 수 있다. 볼트에는 다양한 종류의 전략들이 연결되어 있는데, 볼트와 전략은 스트래티지스트에 의해 개발되어 배포될 수 있으며 개발자는 전략의 수익의 일부를 분배받는 구조이다. 이때 개발자의 수익은 전체 수익의 %로 발생해, 좋은 전략을 만들 수록 더 많은 수익을 거둘 수 있는 인센티브 구조가 짜여 있다. 이런 볼트는 납입되는 토큰의 종류에 따라 크게 세가지로 구분된다. 첫째로, 스테이블 코인을 예치받아 운용하는 스테이블 풀은 어그리게이터, 레버리지 일드파밍을 제공하는 플랫폼, 커브등을 활용해 보상으로 주어지는 토큰을 획득하고 스테이블로 다시 바꾸어 펀드의 규모를 키우는 전략을 사용한다. 둘째로, 디파이 볼트가

있다. 예치된 토큰으로 이자농사를 진행할 수 있는 디파이 프로토콜을 활용해 보상 토큰을 획득하며, 보상토큰을 예치토큰의 형태로 바꿔 펀드가 홀딩하는 토큰의 개수를 늘리는 방식으로 운용된다. 셋째로, 커브의 유동성 풀을 활용하는 커브 볼트는 커브와 컨벡스를 활용해 보상 토큰을 활용하고 보상 토큰을 획득하고 다시 원래의 코인으로 바꾸어 볼트에 잠겨있는 토큰의 개수를 늘리는 전략을 취한다. 이때 커브 볼트는 전체 보상량의 10%를 별도로 할당해 부스팅 목적으로 활용한다는 점에서 커브랑 긴밀하게 연동된다. 이러한 특이한 정책 덕분에 기타 어그리게이터 대비 커브 물량을 많이 들고 있다는 점이 차별화되는 점이다.



연파이낸스 볼트와 전략이 작동하는 방식을 보면, 특정한 토큰이 납입되었을때 해당 토큰을 활용할 수 있는 전략에 자금이 투입된다. 위의 그림의 경우 이더를 받아서 메이커다오 프로토콜을 활용해 스테이블 코인인 다이를 빌리고 커브의 유동성 풀을 활용해 이자농사를 진행하는 방식이다. 이때 사용되는 전략은 여러 원칙하게 설계되고 운용된다.

- 토큰의 개수는 증가하여야 함
- 볼트의 기초자산은 한 종류의 토큰이며, 해당 토큰의 개수는 항시 증가하여야 함
- 비영구적 손실이 발생하지 않아야 함
- 출금은 상시 이뤄질 수 있어야 함 (타임락 x)
- 신뢰할 수 있는 프로토콜에만 이자농사를 진행하여야 함.

타 프로토콜 볼트와 차별화 포인트는 다음과 같다.

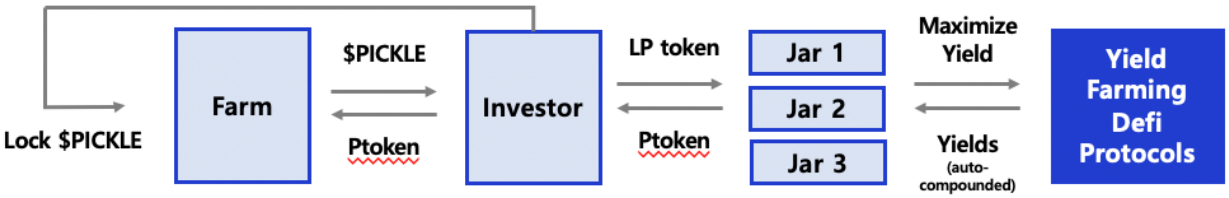
- 이용자는 한 종류의 토큰을 예치하고 해당 토큰의 비율만큼 볼트의 지분을(Vault Shares) 받게 됨. 해당 지분은 자유롭게 전송 가능.
- Strategy에 대한 엄격한 due diligence 절차 존재.
- 오라클을 사용하지 않음. 자체적으로 랜딩, 거래 기능을 활용하지 않기 때문

다른 볼트들과 전략들도 이와 유사한 방식으로 운용되나, 토큰의 특성과 프로토콜의 특징에 약간의 차이가 있다.

피클 파이낸스

피클 파이낸스는 2020년 9월 이더리움 기반으로 출시된 yield aggregator 서비스이다. 다만 스테이블 코인의 페깅 문제를 해결하려는 것을 프로덕트의 목적성으로 잡고 있다는 점이 연파이낸스와 차별화되는 점이다. 피클파이낸스는 크게 두가지, 이자농사와 볼트를 통해 디페깅 문제를 해결하고자 한다. 이자농사의 경우 스테이블 코인 lp 풀을 운영하고, 디페깅된 풀에 추가 보상을 할당해 유동성 공급을 통해 디페깅 문제를 해결하고자 한다. 볼트의 경우 이자농사로 획득한 보상으로 디페깅 된 코인들을 매수하는 구매하는 방식으로 스테이블 코인의 디페깅을 해결하고자 한다.

이 외의 서비스로는 jar 와 farm이라는 내부 서비스가 있는데, jar는 연파이낸스 볼트와 대응대는 개념으로 여타 dex에 유동성을 공급하는 프로토콜이다. farm은 보상 토큰인 피클을 스테이킹하는 컨트랙으로, 예치자는 피클을 예치해 보상에 대한 부스팅을 받을 수 있다.



연파이낸스와 동일하게 다양한 전략들이 jar에 연결되어 있다. 기본적인 전략인 pJar 0은 커브의 기본적인 이자농사 문법위에서 운영된다는 점에서 연 파이낸스와 유사한 모습을 보이나, 보상으로 받은 토큰을 디페깅된 스테이블 코인을 구매하고 이를 재예치해 추가 수익을 노린다는 점에서 차별점을 보인다. pJar 1 은 레버리지 전략이다. 예치된 토큰을 담보로 삼아 시드를 증대시키고, 확대된 시드로 이자농사를 짓는 방식이다.

연파이낸스와 비교한 피클 파이낸스 볼트 전략의 차별점은 다음과 같다. 첫째, 볼트당 예치할 수 있는 코인의 개수에서 차이를 보인다. 연파이낸스의 경우 vault 당 하나의 코인만 운용 가능하며, 1개의 vault 내에서 최대 20개까지 전략을 채택할 수 있어 수익성이 극대화된다. 반면, 피클 파이낸스의 경우, Vault 당 여러 개의 코인 운용 가능하고, Vault 당 1개의 전략만이 사용된다. 서비스 전체에서도 근본적으로는 2개의 전략만이 활용되고 있고, 이로 인하여 연파이낸스에 비해 낮은 수익성을 보인다. 둘째, 토큰의 가치 유지 구조에서 차이를 보인다. 연파이낸스의 경우 프로젝트가 진행될수록 $YFI의 물량이 증가되며 가치가 희석되는 문제점 발생하는 반면, 피클 파이낸스의 경우 수익의 1.5%는 시중 유통되고 있는 $PICKLE을 바이백하기 위해 징수되어 $PICKLE의 가치 하락을 방어한다.

하베스트 파이낸스

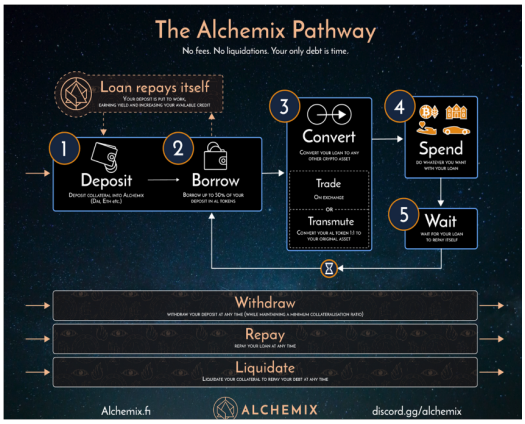
해당 프로토콜은 자동화 이자농사 서비스를 제공하는 일드 어그리게이터로 Ethereum, Polygon, Arbitru, BSC 메인넷 상에서 운영된다. vault와 비슷한 방식으로 이자농사가 이루어지는데, 자금을 투입하면 정해진 전략 (대체로 Yield Farming) 에 따라 운용하고, 운용 수수료 (ETH : 30%, BSC : 8%) 를 제외한 이자를 제공한다. 운용 수수료를 가져가는 방식을 취하고 있으며 오토 컴파운드 기능을 활용해 이용자가 가스비를 아낄 수 있다는 점에서 매우 유사한 점을 보인다. 그러나 보상으로 나오는 farm토큰을 통해 머니레고가 용이하다는 점은 연파이낸스와 차별화되는 점이다.

Yield Farming은 프로젝트 초기에만 높은 수익을 주고, 이후에는 토큰 물량이 풀리며 수익이 급감하는 문제가 있다. 이를 보완하기 위하여 Performance fee로 open market의 FARM 토큰을 구매하고, 이를 Profit Sharing Pool의 스테이커들에게 제공한다. 바이백 시스템이 갖춰져있다는 점에서는 피클파이낸스와 유사한 점을 보이고 있고, 해당 프로토콜도 연 파이낸스와 동일하게 엄격한 듀 딜리전스 과정을 거친다는 점에서도 유사한 점이 있다. 타임락을 걸어 악용을 방지한다는 점에서 차별점을 보인다.

또한 'Council of 69'라는 독특한 방식의 거버넌스를 통해 투자를 진행한다. 기여한 바가 많은 69명의 인원들로 구성된 Council에서, 리저브를 이용한 타 프로젝트에 대한 투자를 발의하고 투표를 진행할 수 있다. 채택된 전략은 제안자가 Dev Team과 함께 진행하며, 얻은 수익의 10%를 수령한다. 이는 매우 유연한 전략을 통해 새로운 자산 운용 루트를 창출할 수 있는 방안으로, 10%의 이익을 제안자에게 부여하기 때문에 언제나 새로운 루트를 찾을 충분한 유인을 제공한다. 더불어 'doHardWork'이라는 간단한 형태의 재투자 로직을 강조한다는 점도 하베스트 파이낸스의 독특한 모습중 하나이다. 재투자 로직을 컨트랙트 상에서 단순하게 유지해 취약점에 대한 공격을 어렵게 만드는 효과를 얻을 수 있다.

알케믹스

연파이낸스와 비슷한 층위에 있는 일드 어그리게이터 뿐만 아니라, 일드 어그리게이터를 통해 파생된 디파이도 간단하게 조사해보았다. 알케믹스는 대출 프로토콜인데, 연파이낸스의 볼트를 사용해 이용자의 부채를 경감하고자 한다. 대부분의 대출 프로토콜은 초과담보를 받아 담보금 대비 적은 양의 금액만 빌려주는 경우가 많다. 알케믹스 같은 경우에는 담보물을 연파이낸스에 보내 이자농사를 진행하고 여기서 얻어지는 이자로 부채를 자동으로 탕감한다. 이자농사라는 형태의 디파이 수익활동을 기반으로 수익 최적화를 위해 어그리게이터 라는 것이 등장했고, 어그리게이터가 다시 랜딩프로토콜에서 활용된다는 점에서 다양한 형태의 레고 블럭이 가능하다는 점을 확인할 수 있었다.



Yearn Vault and Strategy Deployment

추가로, Yearn Vault and Strategy Deployment 에 관련된 코드 리뷰를 진행하였다.

새로운 vault의 deployment를 위해서는 Yearn Vaults 깃허브 레포지토리를 clone 한 뒤, "brownie run scripts/deploy.py --network <network-to-deploy-vault>" 을 run 한다. 해당 brownie account를 선택한다. 이 때, 해당 account는 deploy transaction에 대하여 지불할 수 있는 것으로 선택해야 한다. script가 최신 버전 (v2.registry.ychad.eth) 을 사용하는 것을 확인한 후, vault version 또한 최신 버전으로 사용하도록 선택한다. 파타미터 값들을 설정한다. 해당 address를 governance로, Treasury (treasury.ychad.eth) 를 reward address로, Core Dev multisig (dev.ychad.eth) 을 guardian으로, Strategist multisig (brain.ychad.eth) 을 management로 설정한다. description, vault에 대한 symbol을 설정하거나, 이후 on chain에서 변경할 수 있으므로 default 값을 사용해도 된다. 새로운 vault가 Etherscan위에 ABI setup이 있는지를 체크 한 후, "vault.setDepositLimit(limit)"를 이용하여 해당 vault의 deposit limit을 설정한다.
"vault.setManagementFee(0)" 를 통하여 management fee를 0으로 설정한다. governance가 accept되기 전까지 vault를 수정할 수 있다.

strategy를 추가하기 위해서는 Core Dev strategist의 private repository에 Strategy Review template을 활용하여 새로운 issue를 올린다. 해당 strategy의 review를 진행하여야 하는데, 이때 최소 2명 이상의 strategists의 peer review가 필요하다. 코드 리뷰를 통하여 문제가 있거나, 보안해야할 부분을 수정한다. 이후, 적절한 deposit limit을 비롯한 다른 setting들을 확인하고, strategy를 deploy 하고 해당 코드를 Etherscan에 올린다. depoly된 version으로 깃허브 review issue에 태그를 하고, strategy item에 대한 mainnet address를 private board에 올린다.
이후 해당 vault와 strategy가 같이 동작하도록 하기 위한 작업이 필요하다. 아래는 strategy를 vault에 추가하기 위한 코드 예시이다.

```
strategy = ''
debt_ratio = 9800
minDebtPerHarvest = 0
maxDebtPerHarvest = 2 ** 256 - 1
performance_fee = 1000

vault.addStrategy(
  strategy,
  debt_ratio,
  minDebtPerHarvest,
  maxDebtPerHarvest,
  performance_fee
)
```

strategy에 해당 strategy address를 넣어준뒤. minDebtPerHarvest에 lower limit on debt add를, maxDebtPerHarvest에 Upper limit on debt add을 넣어준다. performance_fee의 경우 strategist performance fee이고 해당 값을 1000으로 설정하면, 10%를 의미한다. vault에서 맨 처음의 strategy인 경우 debt_ratio는 9800으로 설정되어야 하고, 특정한 이유가 있지 않은 한 rate_limit은 0으로 설정되어야 한다.

keeper, health check, rewards를 세팅하는 방법은 다음과 같다.
keep3r_manager = 0x736D7e3c5a6CB2CE3B764300140ABF476F6CFCCF
strategy.setKeeper(keep3r_manager)
health_check = 0xddcea799ff1699e98edf118e0629a974df7df012
strategy.setHealthCheck(health_check)
strategy.setRewards(address)

이후 harvesting을 직접 테스팅해야 한다. vault에 token을 deposit 한뒤, "strategy.harvest( )"를 실행하여 harvest를 진행해 본 뒤, 제대로 작동하는지 확인한다.

strategy에 health check를 위하여 unit test를 추가하는 방법은 다음과 같다.
commonHealthCheck = Contract(web3.ens.resolve(“health.ychad.eth”))
strategy.setHealthCheck(commonHealthCheck)

Health Check를 customize할 수 있는데, 각 strategy에 대한 특정한 profit, loss limit을 체킹하도록 설정할 수 있다. 마찬가지로 'strategy'에는 strategy address를 넣어주면 된다.
strategy = '';
profitLimit = 100
lossLimit = 100
healthcheck.setStrategyLimits(strategy, profitLimit, lossLimit)

customize 된 Health Check contract를 common Health Check contract에 추가하는 방법은 다음과 같다. 'customHealthCheck'에는 해당 customize 된 Health Check address를 넣어준다. 이때, custom Health Check는 정해진 interface를 따르도록 해야한다.
strategy = '';
customHealthCheck = ''
healthcheck.setCheck(strategy, customHealthCheck)

Ethereum Addresses, Fantom Addresses, Arbitrum Addresses는 다음과 같다.

Ethereum Addresses

Identity	ENS	Address
V2 Registry	v2.registry.ychad.eth	0x50c1a2eA0a861A967D9d0FFE2AE4012c2E053804
Yearn multisig (daddy)	ychad.eth	0xFEB4acf3df3cDEA7399794D0869ef76A6EfAff52
Strategist multisig	brain.ychad.eth	0x16388463d60FFE0661Cf7F1f31a7D658aC790ff7
Core Dev multisig	dev.ychad.eth	0x846e211e8ba920B353FB717631C015cf04061Cc9
Treasury	treasury.ychad.eth	0x93A62dA5a14C80f265DAbC077fCEE437B1a0Efde
Health Check	health.ychad.eth	0xDDCea799fF1699e98EDF118e0629A974Df7DF012

Arbitrum Addresses

Identity	Address
Registry	0x3199437193625DCcD6F9C9e98BDf93582200Eb1f
Strategist multisig	0x6346282DB8323A54E840c6C772B4399C9c655C0d
Governance multisig	0xb6bc033D34733329971B938fEf32faD7e98E56aD
Treasury	0x1DEb47dCC9a35AD454Bf7f0fCDb03c09792C08c1
Health Check	0x32059ccE723b4DD15dD5cb2a5187f814e6c470bC

Fantom Addresses

Identity	Address
Registry	0x727fe1759430df13655ddb0731dE0D0FDE929b04
Strategist multisig	0x72a34AbafAB09b15E7191822A679f28E067C4a16
Governance multisig	0xC0E2830724C946a6748dDFE09753613cd38f6767
Treasury	0x89716ad7edc3be3b35695789c475f3e7a3deb12a
Health Check	0xf13Cd6887C62B5beC145e30c38c4938c5E627fe0