15 de Enero de 2019

Autores:
Jordi Riulas
Joan Marcos

1. Resolución de problema de ataque masivo a Kickstart en fase de deploy:

Problema:
Si Kickstart realiza el deploy y no tenemos un control sobre este evento, el sistema podría estar sujeto a un ataque de fuerza con solicitudes masivas de despliegue y el coste asociado.
Si el deploy lo realiza el promotor, eventualmente tendría la capacidad de modificar el contrato y diseñar un posible fraude.

Solución propuesta.
El elemento central es que sea Kickstart quien se encargue de deployar los smart contracts de la campaña, pero exige un pago inicial al promotor para que su proyecto sea publicado. Es decir, existe un approval previo a la publicación, mediante pago de un fee.

Diseño funcional:
La propia plataforma Kicksart se gestiona mediante un smart contract específico de plataforma que se publica en el momento de inicio de las operaciones de Kickstar, el kickstart_contract. Este contract da servicio a todas las eventuales campañas que los promotores vayan a lanzar.

kickstart_contract tiene una función que permite la autorización previa y el mapeo @ de promotores, si han realizado un pago mínimo => promoterValidation()

promoterValidation() tiene argumentos: @dirección del promotor, minimum_contribution y es payable con un importe mínimo definido. Este importe, aunque pequeño, será la barrera de entrada a la publicación de campañas.

Si la función promoterValidation() tiene éxito, el promotor podrá llamar a una segunda función del kickstart_contract: launchCampaign(). Si previamente el promotor ha sido validado, esta función launchCampaign() será la que deployará propiamente el contrato de la campaña concreta, con los parámetros de minimum_contribution y de @dirección del promotor. Este segundo parámetro habría que añadirlo al constructor del contrato de campaña, porque ahora mismo el manager de campaña es directamente el msg.sender, cosa que ya no sería así.
En caso contrario no se llegará siquiera a deployar. con lo cual no habría el riesgo de consumo incontrolado de gas.

2. Resolución de problema bug en smart contract.

Problema:
Queremos establecer un mecanismo que permita solucionar bugs de código asociados al smart contract permitiéndonos publicar nuevas versiones del mismo.

Solución:
La solución propuesta es un proxy-contract que apunte siempre al último contrato con la implementación más reciente de la campaña.

Diseño funcional.
El smart contract a desarrollar lo dividiremos en dos smarts contracts.
interfaceContract y implementationContract.

interfaceContract será el que no varía, y el que ofrecerá el interfaz a los usuarios del sistema- inversores en este caso.
ImplementationContract será el contracto no accesible a los usuarios del sistema, que realmente tendrá la implementación de las funciones del interfaceContract.
InterfaceContract debe tener un acceso siempre al implementationContract más reciente.

El interfaceContract tendrá una función restringida al usuario manager: deployImplementationContract(), para hacer deploy del implementationContract. Esta función podrá ser llamada por el manager cada vez que hay una implementación que corrija errores o mejore las prestaciones del sistema. El resultado de este deployImplementationContract() se dejará en una variable global del interfaceContract: currentImplementationContract.

Cada función del interfaceContract, por ejemplo contribute(xxx), realizará internamente una llamada tipo currentImplementationContract.methods.contribute(xxx).send(yyy)