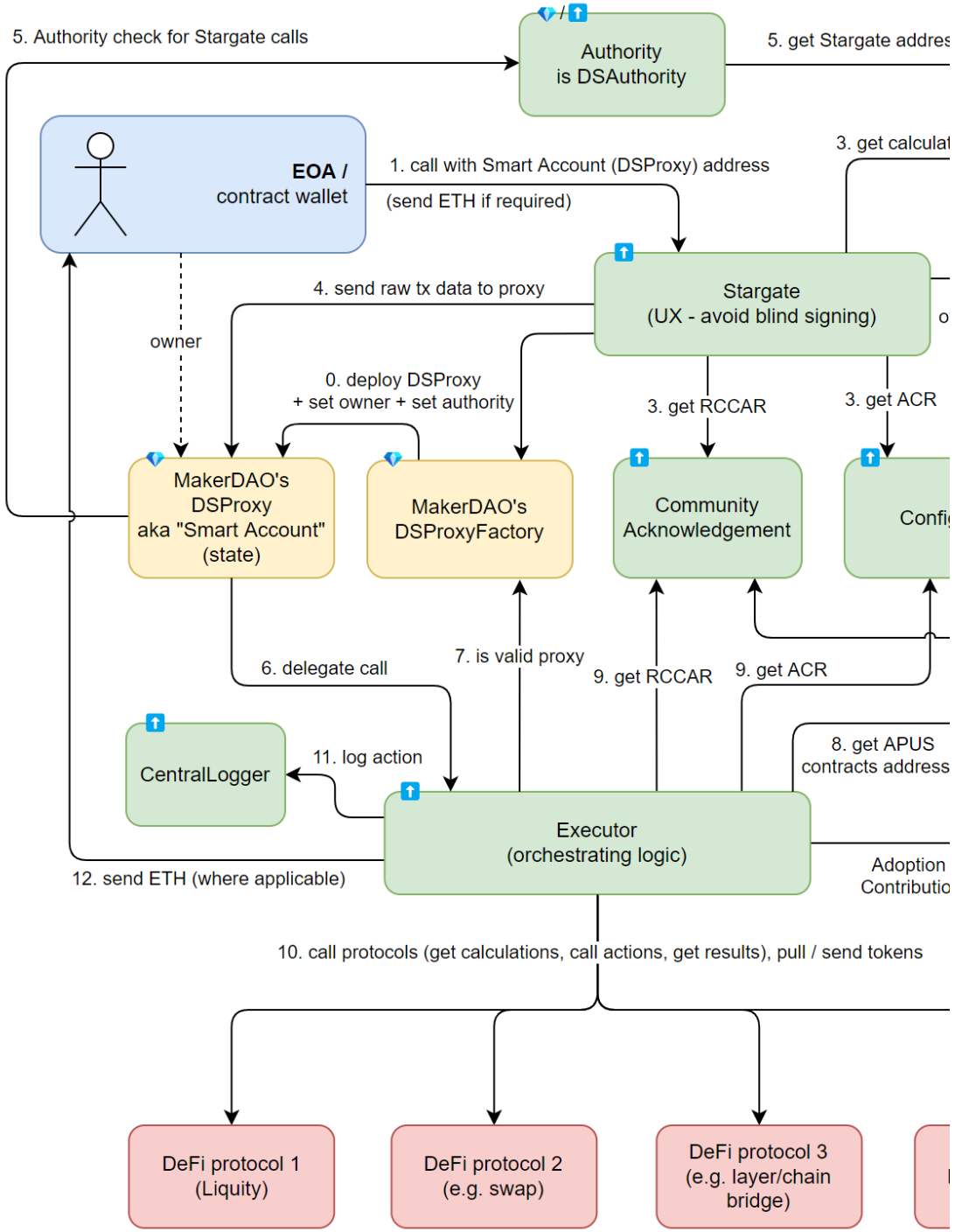
Effectivity of proxies

RADEK ŠVARZ - 🍻 BEERFI 2022-07-26

Some architecture using DS Proxy



@radk

Issue: Gas spent on proxy creation for user

Transaction Info for
0xad423ef00e7de9719bc0a09d7595d6e47e70538a6a70e1d285356698998a628e

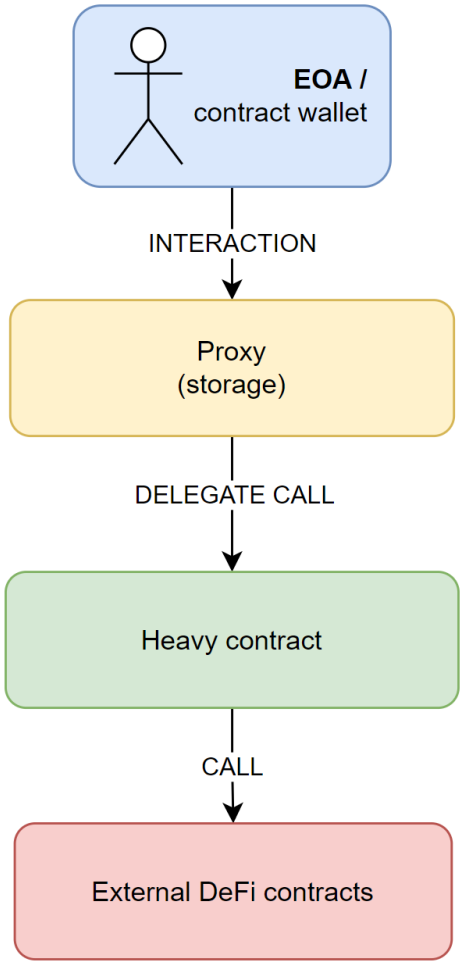
Block Number:	14682732 at Apr-30-2022 01:44:45 AM +UTC
Transaction Cost:	0.022147078891769784 ETH $30.63
Gas Used:	595,976 Gas / 37.161024759 Gwei

Emitted Events:

56	DSProxy.LogSetOwner(owner=DSProxyFactory)
57	DSProxy.0x0000000000000000000000000000000000000000000000000000000000000000 =0x000000000000000000000000a26e15c895efc0616177b7c1e7270a4c7d51c997, = =0x0000000000000000000000000000000000000000000000000000000000000000, = =0000000000000000000000000000000000000000000000000000000000000040, =00
58	DSProxyFactory.Created(sender=[Receiver] ProxyRegistry, owner=[Sender]

Can we do better?

Requirements



Small bytecode footprint to save gas on deploy

Minimum gas for passing execution

Revealing target contract methods (avoid blind signing)

Minimal Proxy - EIP1167

- 45 bytes only

- Probably the most frequently used implementation of EIP 1167 is by **Uniswap V1** in the creation of **their AMM pools**

- 🤔 **Why openzeppelin call them clones ?**

Bytecode:

3d602d80600a3d3981f3363d3d37
3d3d3d363d73bebebebebebebebebe
bebebebebebebebebebebebebebe5af4
3d82803e903d91602b57fd5bf3

Proxy steps

Receive	Receive some data
Forward	Forward the received data to an implementation contract using the DELEGATECALL instruction.
Get	Get the result of the external call (i.e., the result of DELEGATECALL)
Return	Return the external call's result to the caller if step 3 succeeded, or revert the transaction in any other case.

https://blog.openzeppelin.com/deep-dive-into-the-minimal-proxy-contract/

1. Receive what to do (CALLDATACOPY)

3d602d80600a3d3981f3**363d3d37**3d3d3d363d73bebebebebebebebebebebebebebebebebebebebebebe5af43d82803e903d91602b57fd5bf3

Code	Instruction	Stack	Memory
36	CALLDATASIZE	cds	–
3d	RETURNDATASIZE	0 cds	–
3d	RETURNDATASIZE	0 0 cds	–
37	CALLDATACOPY	–	[0, cds] = calldata

https://blog.openzeppelin.com/deep-dive-into-the-minimal-proxy-contract/

2. Forward (DELEGATECALL)

3d602d80600a3d3981f3363d3d37**3d3d3d363d73bebebebebebebebebebebebebebebebebebebebe5af4**3d82803e903d91602b57fd5bf3

Code	Instruction	Stack	Memory
3d	RETURNDATASIZE	0	[0, cds] = calldata
3d	RETURNDATASIZE	0 0	[0, cds] = calldata
3d	RETURNDATASIZE	0 0 0	[0, cds] = calldata
36	CALLDATASIZE	cds 0 0 0	[0, cds] = calldata
3d	RETURNDATASIZE	0 cds 0 0 0	[0, cds] = calldata
73 addr	PUSH20 0x123...	addr 0 cds 0 0 0	[0, cds] = calldata
5a	GAS	gas addr 0 cds 0 0 0	[0, cds] = calldata
f4	DELEGATECALL	success 0	[0, cds] = calldata

https://blog.openzeppelin.com/deep-dive-into-the-minimal-proxy-contract/

3. Get result (RETURNDATACOPY)

3d602d80600a3d3981f3363d3d373d3d3d363d73bebebebebebebebebebebebebebebebebebebebebebebebebebebe5af4**3d82803e**903d91602b57fd5bf3

Code	Instruction	Stack	Memory
3d	RETURNDATASIZE	rds success 0	[0, cds] = calldata
82	DUP3	0 rds success 0	[0, cds] = calldata
80	DUP1	0 0 rds success 0	[0, cds] = calldata
3e	RETURNDATACOPY	success 0	[0, rds] = return data *(there might be some irrelevant leftovers in memory [rds, cds] when rds < cds)*

https://blog.openzeppelin.com/deep-dive-into-the-minimal-proxy-contract/

4. Return / revert

3d602d80600a3d3981f3363d3d373d3d3d363d73bebebebebebebebebebebebebebebebebebebebebebebebebebebe5af43d82803e**903d91602b57fd5bf3**

Code	Instruction	Stack	Memory
90	SWAP1	0 success	[0, rds] = return data
3d	RETURNDATASIZE	rds 0 success	[0, rds] = return data
91	SWAP2	success 0 rds	[0, rds] = return data
60 dest	PUSH1 dest	dest sucess 0 rds	[0, rds] = return data
57	JUMPI	0 rds	[0, rds] = return data

Code	Instruction	Stack	Memory
fd	REVERT	–	[0, rds] = return data
5b	JUMPDEST	0 rds	[0, rds] = return data
f3	RETURN	–	[0, rds] = return data

https://blog.openzeppelin.com/deep-dive-into-the-minimal-proxy-contract/

More minimal proxy

200 less gas to deploy

(1 fewer byte of runtime code)

4 less gas to call

(the SWAP we got rid of costs 3 gas, and RETURNDATASIZE costs 1 less gas than DUP).

```
pc     op / pushdata  opcode              stack (top on the left)
----   -------------  ------------------  -----------------------
0x00   3d             returndatasize      0
0x01   3d             returndatasize      0 0
0x02   3d             returndatasize      0 0 0
0x03   3d             returndatasize      0 0 0 0
0x04   36             calldatasize        cds 0 0 0 0
0x05   3d             returndatasize      0 cds 0 0 0 0
0x06   3d             returndatasize      0 0 cds 0 0 0 0
0x07   37             calldatacopy        0 0 0 0
0x08   36             calldatasize        cds 0 0 0 0
0x09   3d             returndatasize      0 cds 0 0 0 0
0x0a   73bebebebebe.  push20 0xbebebebe   0xbebe 0 cds 0 0 0 0
0x1f   5a             gas                 gas 0xbebe 0 cds 0 0 0 0
0x20   f4             delegatecall        suc 0 0
0x21   3d             returndatasize      rds suc 0 0
0x22   3d             returndatasize      rds rds suc 0 0
0x23   93             swap4               0 rds suc 0 rds
0x24   80             dup1                0 0 rds suc 0 rds
0x25   3e             returndatacopy      suc 0 rds
0x26   602a           push1 0x2a          0x2a suc 0 rds
0x28   57             jumpi               0 rds
0x29   fd             revert
0x2a   5b             jumpdest            0 rds
0x2b   f3             return
```

https://medium.com/coinmonks/the-more-minimal-proxy-5756ae08ee48

@radk

EIP-3448: MetaProxy Standard

Extending clone with extra metadata per deployment.

- a cheap way of storing immutable metadata for each child instead of using storage slots
- inexpensive deployment of clones
- handles error return bubbling for revert messages

<54 bytes metaproxy> <arbitrary data> <length in bytes of arbitrary data (uint256)>

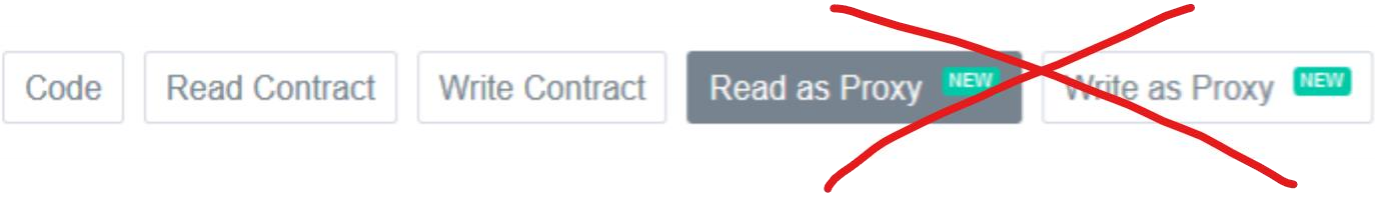
https://eips.ethereum.org/EIPS/eip-3448

But all those (clones) have fixed target address

=> non upgradability of target contracts.

Or chained upgradable proxy is needed as a target.

Chaining proxy **fails** requirement on target methods visibility (Etherscan).



@radk

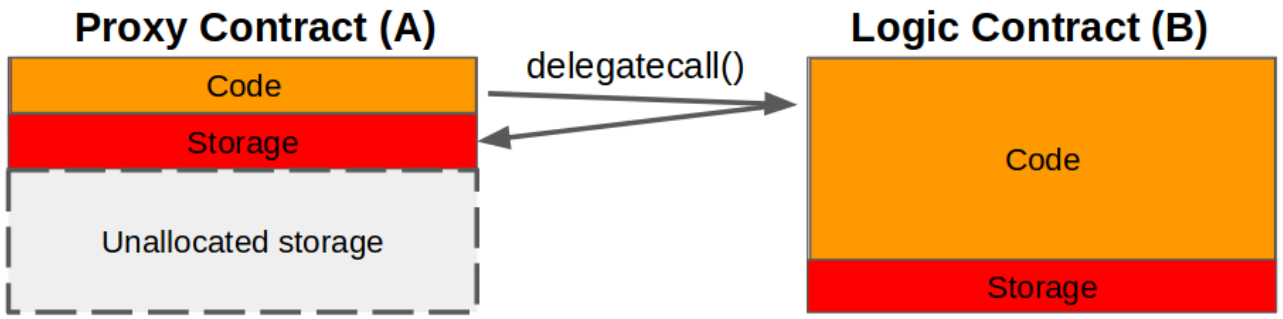
We need proxy with mutable target address

@radk

Mutable target address within proxy

- Change the target address pointer @ proxy storage
- EIP-1822: Universal Upgradeable Proxy Standard (UUPS)
- Uses defined storage position in proxy: keccak256("PROXIABLE")
- 💀 Delegated contract can *"kill"* the proxy.
- 👯 👯 When upgrade => target to be upgraded in each instance (i.e. to ask every user)

```
assembly { // solium-disable-line
    let contractLogic := sload(0xc5f16f0fcc639fa48a6947836d9850f504798523bf8c9a3a87d5876cf622bcf7)
    calldatacopy(0x0, 0x0, calldatasize)
    let success := delegatecall(sub(gas, 10000), contractLogic, 0x0, calldatasize, 0, 0)
    let retSz := returndatasize
    returndatacopy(0, 0, retSz)
    switch success
    case 0 {
        revert(0, retSz)
    }
    default {
        return(0, retSz)
    }
}
```



https://eips.ethereum.org/EIPS/eip-1822

Mutable target address ouside proxy

Manage the target address in Beacon contract

```
contract BeaconProxy is Proxy, ERC1967Upgrade {
    /**
     * @dev Initializes the proxy with `beacon`.
     *
     * If `data` is nonempty, it's used as data in a delegate call to the implementation returned by the beacon. This
     * will typically be an encoded function call, and allows initializing the storage of the proxy like a Solidity
     * constructor.
     *
     * Requirements:
     *
     * - `beacon` must be a contract with the interface {IBeacon}.
     */
    constructor(address beacon, bytes memory data) payable {
        _upgradeBeaconToAndCall(beacon, data, false);
    }

    /**
     * @dev Returns the current beacon address.
     */
    function _beacon() internal view virtual returns (address) {
        return _getBeacon();
    }

    /**
     * @dev Returns the current implementation address of the associated beacon.
     */
    function _implementation() internal view virtual override returns (address) {
        return IBeacon(_getBeacon()).implementation();
    }

    /**
```

VERY HEAVY

https://docs.openzeppelin.com/contracts/4.x/api/proxy#BeaconProxy

@radk

Proposal – Minimal Beacon Proxy

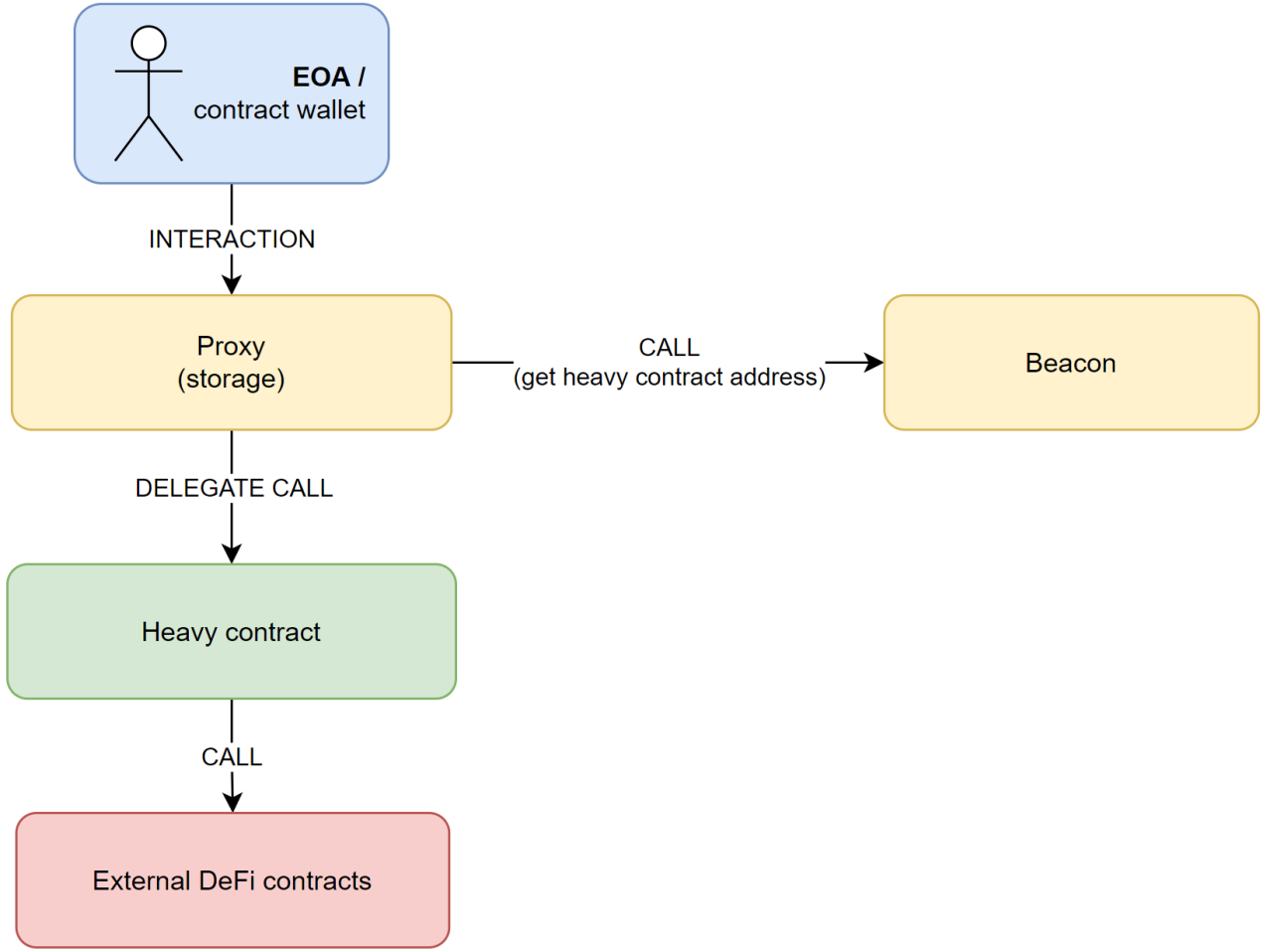
@radk

Proposal – Minimal Beacon Proxy

- Similar to Minimal Proxy (EIP 1167), i.e. low code footprint

And

- Openzeppelin BeaconProxy (i.e. calling Beacon contract to get the target address, but without storing in proxy)



Proposal – Minimal Beacon Proxy

```
assembly { // solium-disable-line
    let contractLogic := sload(0xc5f16f0fcc639fa48a6947836d9850f504798523bf8c9a3a87d5876cf622bcf7)
    calldatacopy(0x0, 0x0, calldatasize)
    let success := delegatecall(sub(gas, 10000), contractLogic, 0x0, calldatasize, 0, 0)
    let retSz := returndatasize
    returndatacopy(0, 0, retSz)
    switch success
    case 0 {
        revert(0, retSz)
    }
    default {
        return(0, retSz)
    }
}
```

- We call Beacon.implementation instead of underlined sload ->
- Cca +2466 gas only for CALL (-100 sload)
- Beacon is centrally manageable – i.e. 1 tx upgrade
- Delegated contracts cannot mess up with proxy target
- No standard, (yet? Wonder, why?)
- 🤔 Etherscan passing methods from target?
- 🤔 Audits?

THANKS

Qs?

@radk