Upgradeability
Good, Bad, Ugly

Mehdi Salehi

UNIVERSITÉ Concordia UNIVERSITY

Funding & Partners

MADIBA security research group

UNIVERSITÉ Concordia UNIVERSITY

GINA CODY SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

FUNDING & PARTNERS:

NSERC CRSNG

Raymond Chabot Grant Thornton

catallaxy

AUTORITÉ DES MARCHÉS FINANCIERS

Fonds de recherche Nature et technologies Québec

Office of the Privacy Commissioner of Canada

Software Maintenance

1

Correct Faults
Fixing a bug
Small Changes

Updates

2

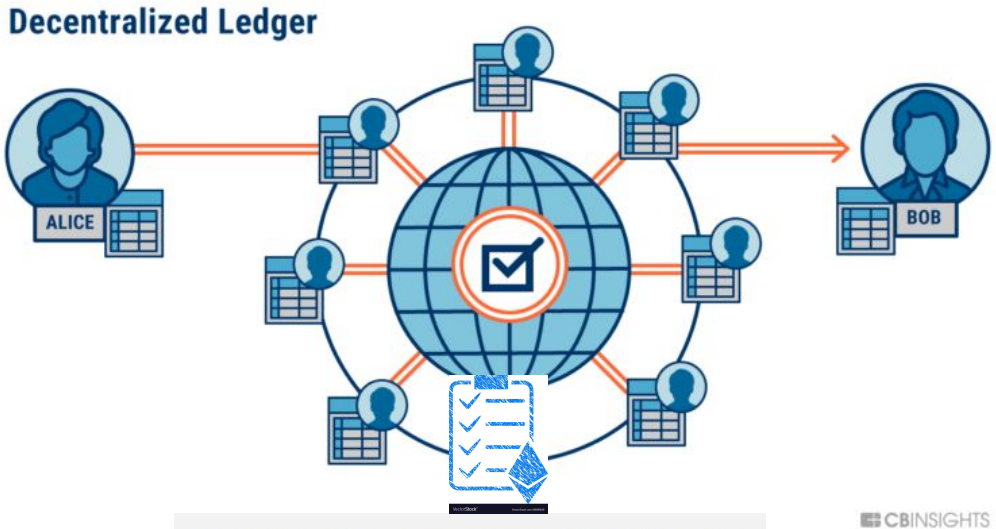
Improve Performance
New Functionalities

Upgrades

UNIVERSITÉ Concordia UNIVERSITY

Smart Contracts

- **Pieces of codes**
  - Better to say Bytecodes
- **Supposed to be immutable**




```
606060405 25b600080fd00a165627a7a7230582012c9bd00152fa1c480f6827f81515
bb19c3e63bf7ed9ffbb5fda0265983ac7980029
```

Immutable

A piece of software that you cannot change it!!!

Immutable

A piece of software that you cannot change it!!!

It is worst in Blockchains Because codes are keeping money here

UNIVERSITÉ Concordia UNIVERSITY

What if the code is Buggy!

Hacker will drain all the fund

UNIVERSITÉ Concordia UNIVERSITY

What if the code is Buggy!

Hacker will drain all the fund

If possible

UNIVERSITÉ Concordia UNIVERSITY

What if the code is Buggy!

DAO Hack

UNIVERSITÉ Concordia UNIVERSITY

Upgradeability is a Bug!

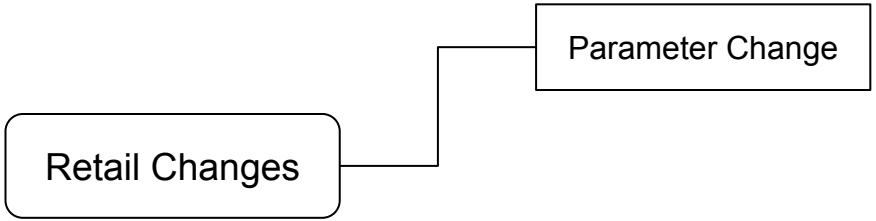
No way!
We need **upgradeability**

How ?!
(immutability)

Who
Is responsible?!

Upgradeability Patterns

Retail Changes



Parameter Change

DAI

Stability Fee

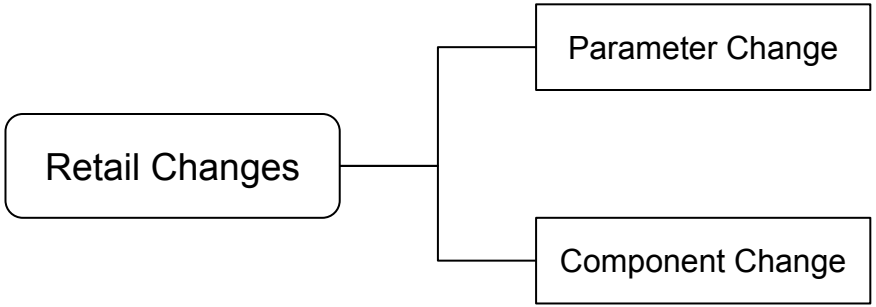
Dai Saving Rate

Collateral Ratio

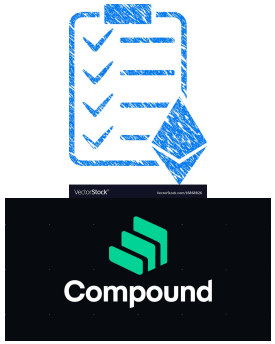
Upgrade to control Demand and Supply of **DAI**

UNIVERSITÉ Concordia UNIVERSITY

Upgradeability Patterns



Component Change

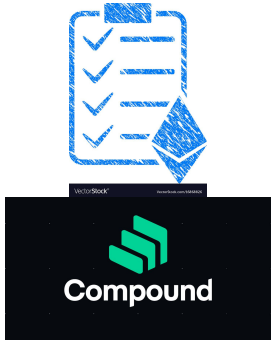


contract for calculations

Assets to Borrow

Need a way to calculate interest rate

Component Change



Assets to Borrow

Need a way to calculate interest rate

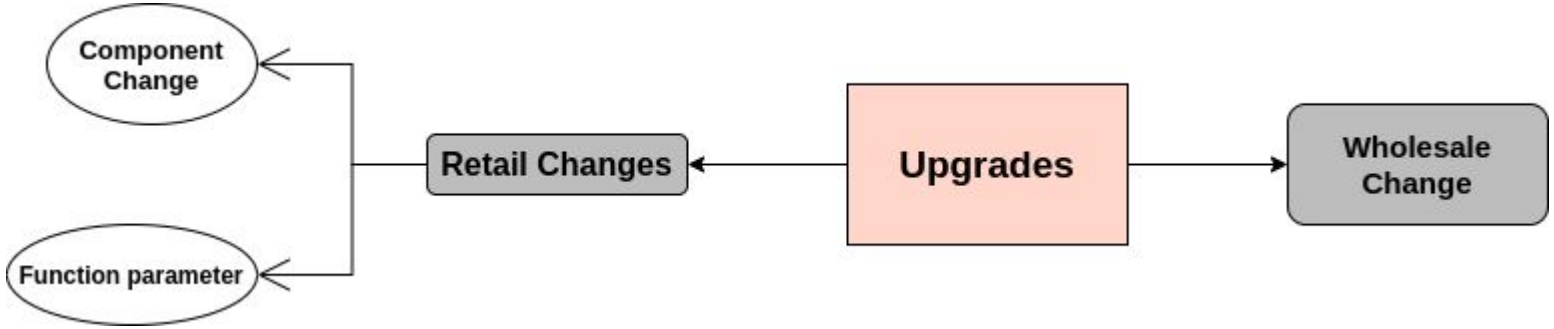
Old contract

New contract

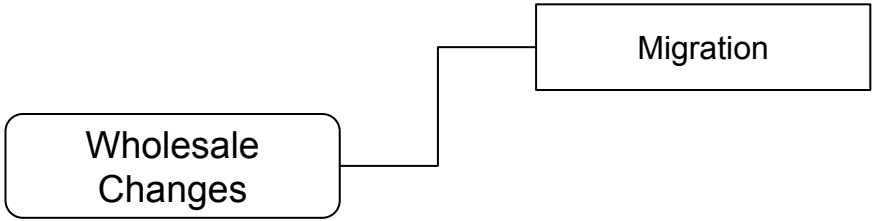
Pros and Cons

- Pros
  - Simple to implement
  - Easy to audit
- Cons
  - Cannot fix a bug
  - Cannot add/change Logic in Parameter Configuration
  - Cannot add new functionality in Component change

Upgradeability Patterns



Upgradeability Patterns



Migration (Social Upgrade)

Deploy a new contract

Migration

Collecting data from old version

Push data in the newer version

Migration (Social Upgrade)

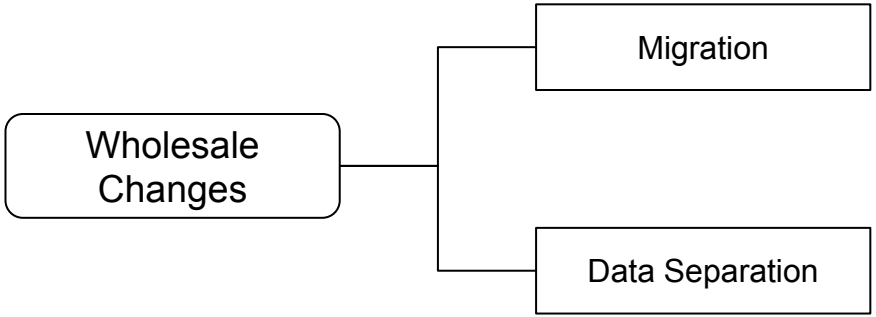
Migration

Deploy a new contract — **Come up with 2 different versions**

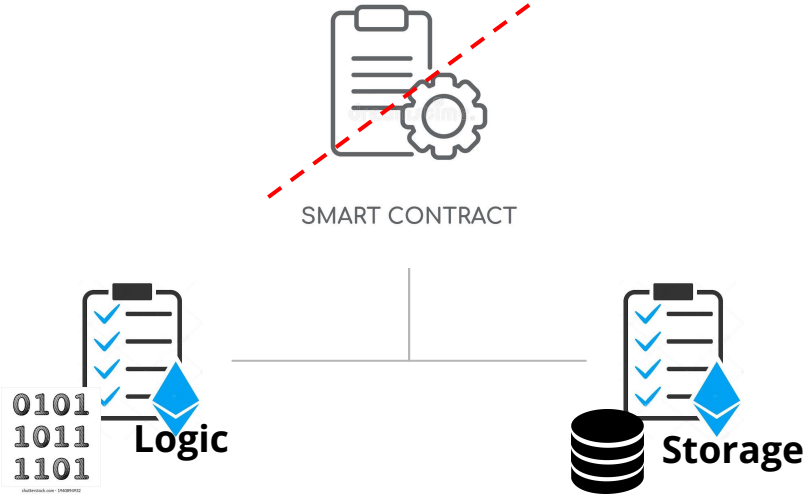
Collecting data from old version — **Cannot Fix a Bug**

Push data in the newer version — **Costly**

Upgradeability Patterns

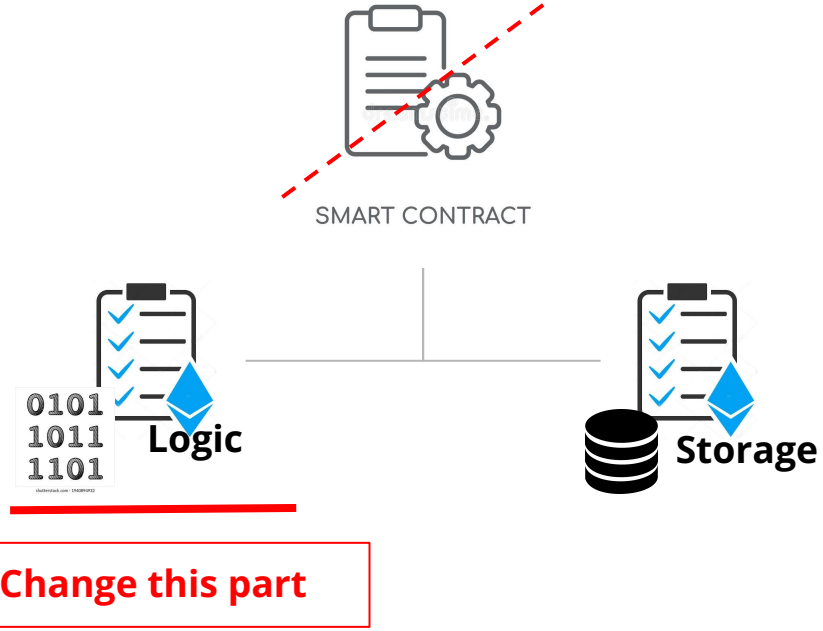


Data Separation



UNIVERSITÉ Concordia UNIVERSITY

Data Separation

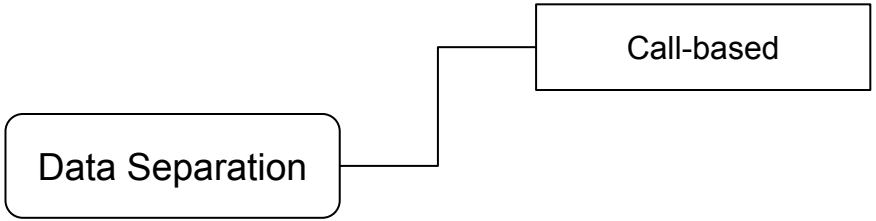


UNIVERSITÉ Concordia UNIVERSITY

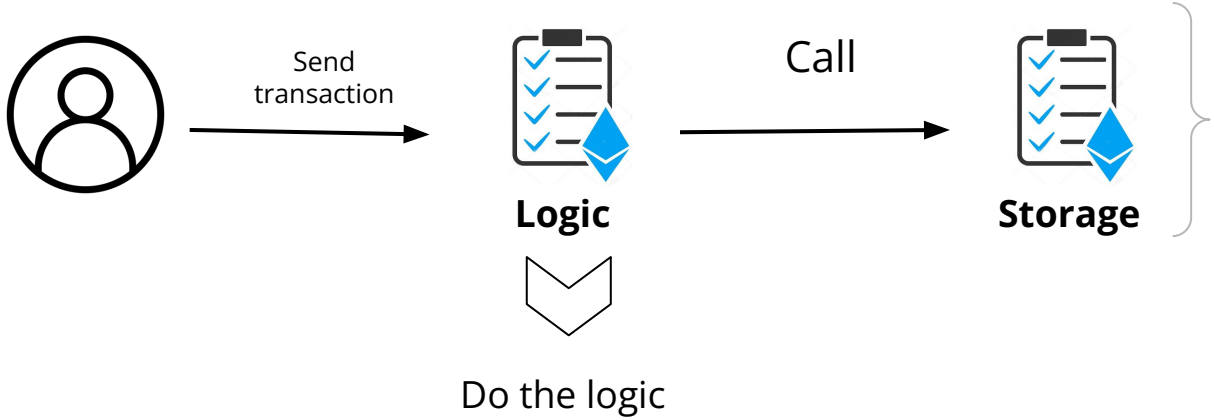
Data Separation

1. **Call** Opcode
2. **DelegateCall** Opcode

Upgradeability Patterns

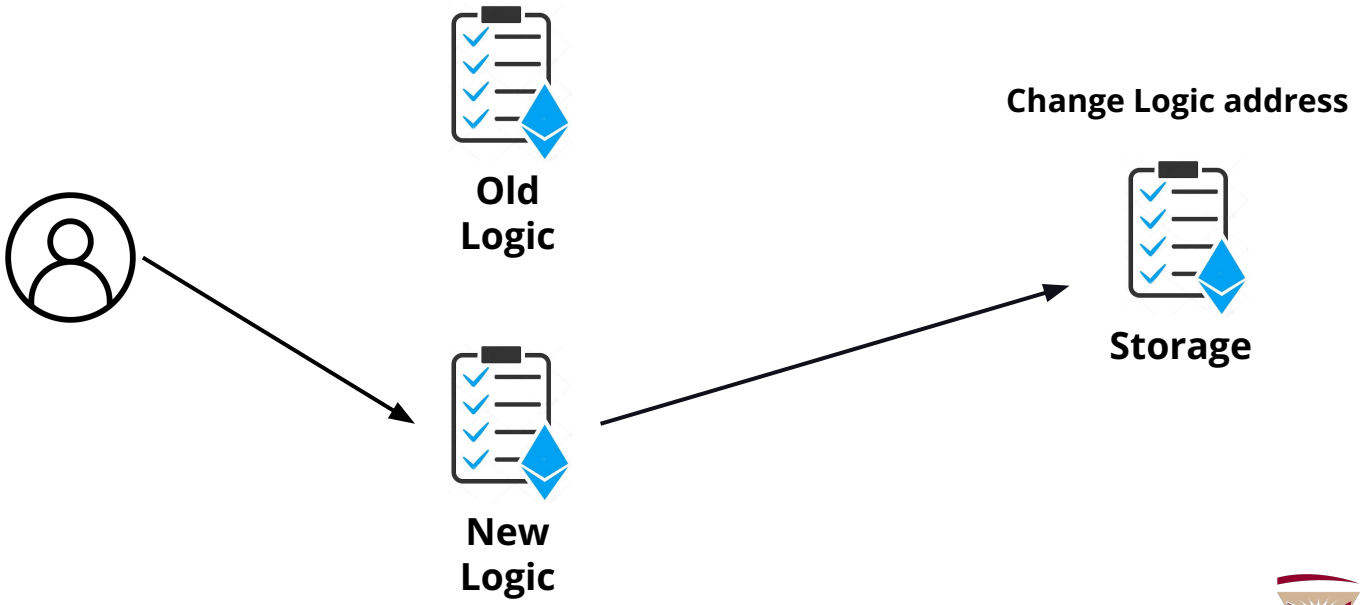


Call based pattern



1. Getter function
2. Setter function
3. Ability to add new variables

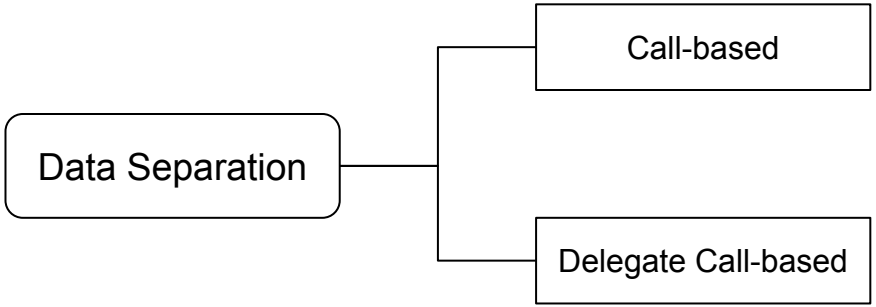
Call based pattern



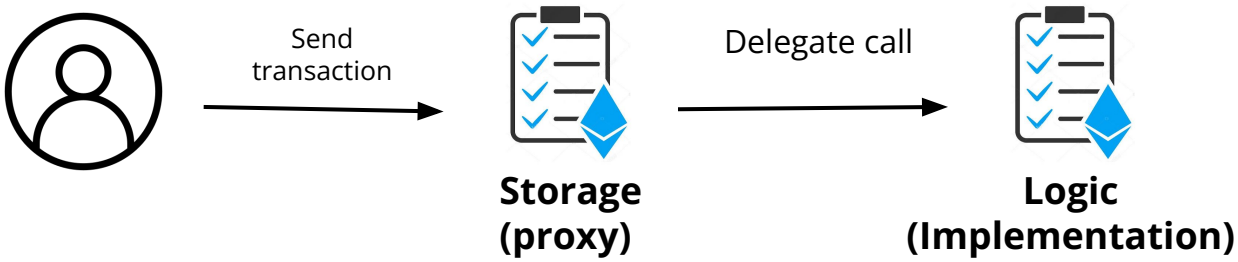
Call based pattern

- No need for Data Migration
- Hard for developers to change their codes and add this pattern
  - Hard to implement for complex data structures
- The address will change
  - Break Composability
  - Not user friendly
  - Need to aware all services about the change
- Deal with the old version

Upgradeability Patterns

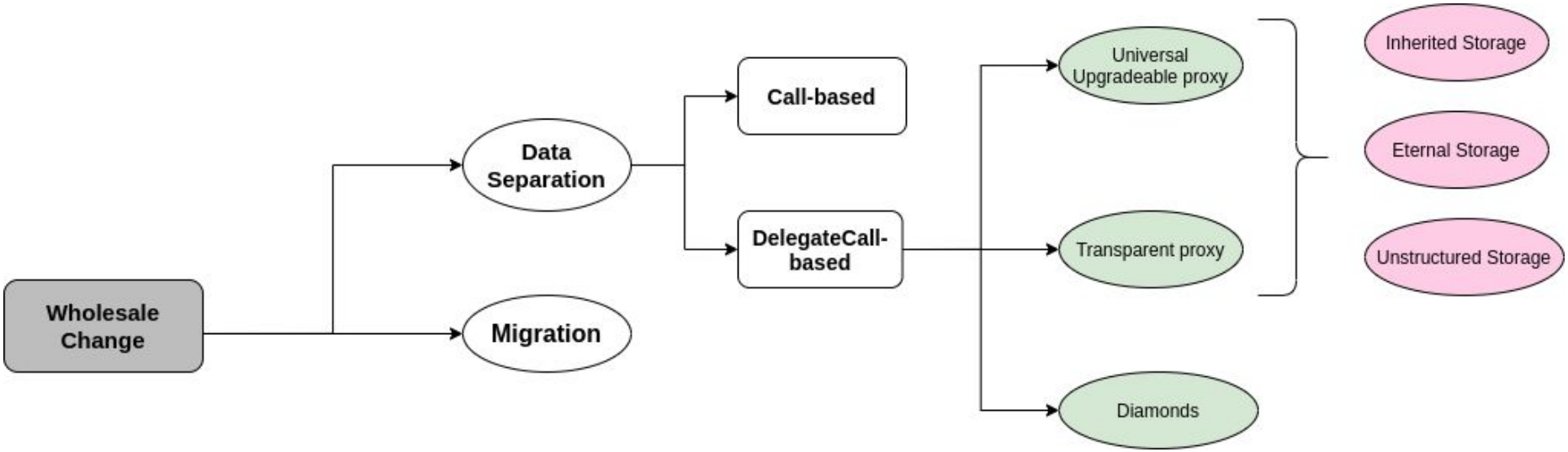


Delegate Call based pattern



- Delegate call preserves the context
- Similar to copy pasting (not the same)
- Needs some consideration
  - Storage structure should be the same
  - Don’t have similar functions

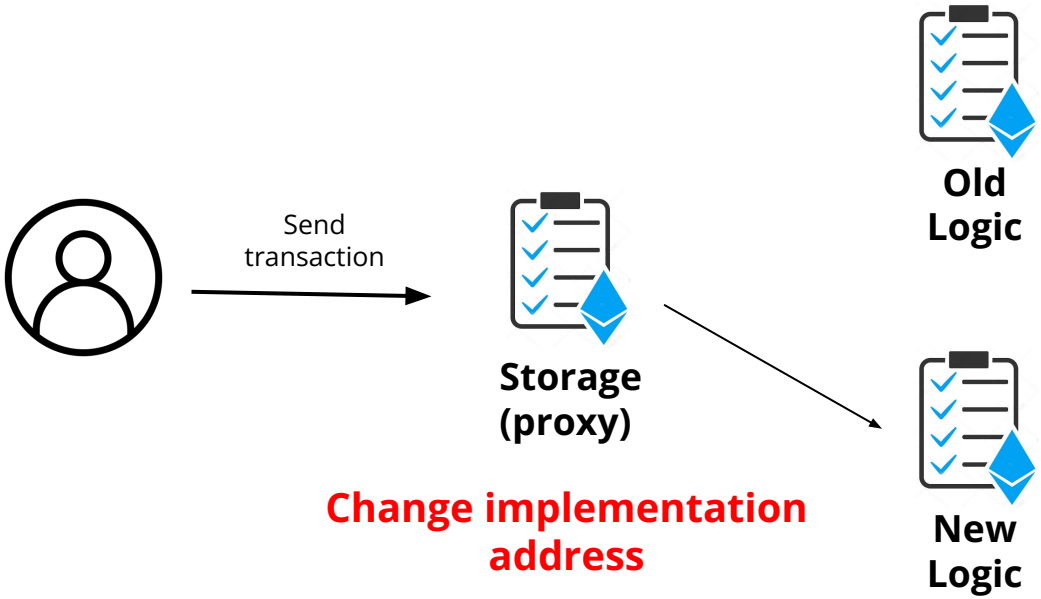
Upgradeability Patterns



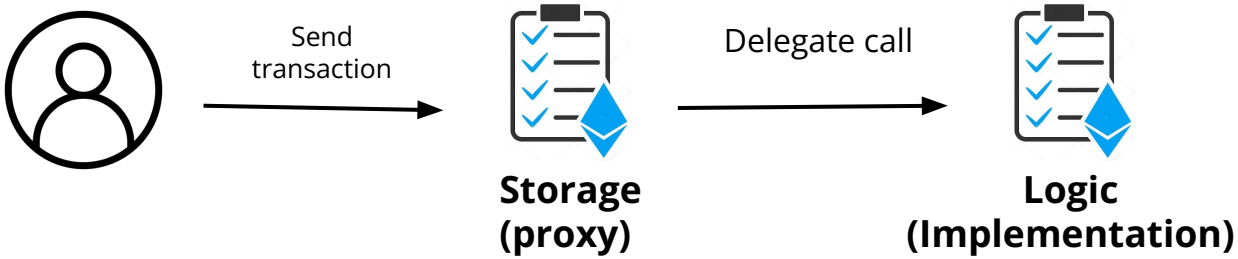
Pros and Cons of Proxy patterns

- Pros
  - Can update the whole logic
  - No need to migrate data
  - Easy to add this pattern to regular code (OpenZepplin OS)
  - User endpoint not changed
  - No downtime during upgrade
    - If contract is unpauseable
  - Very fast in upgrades
  - User friendly
  - No need to deal with two Dapps at the end

Delegate Call based pattern



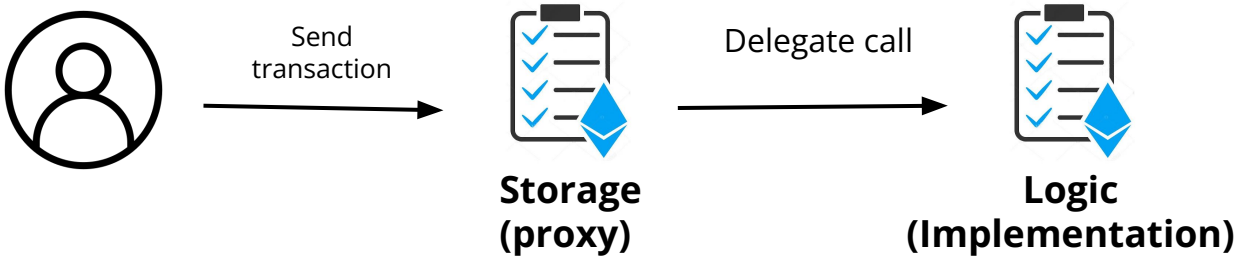
Delegate Call based pattern



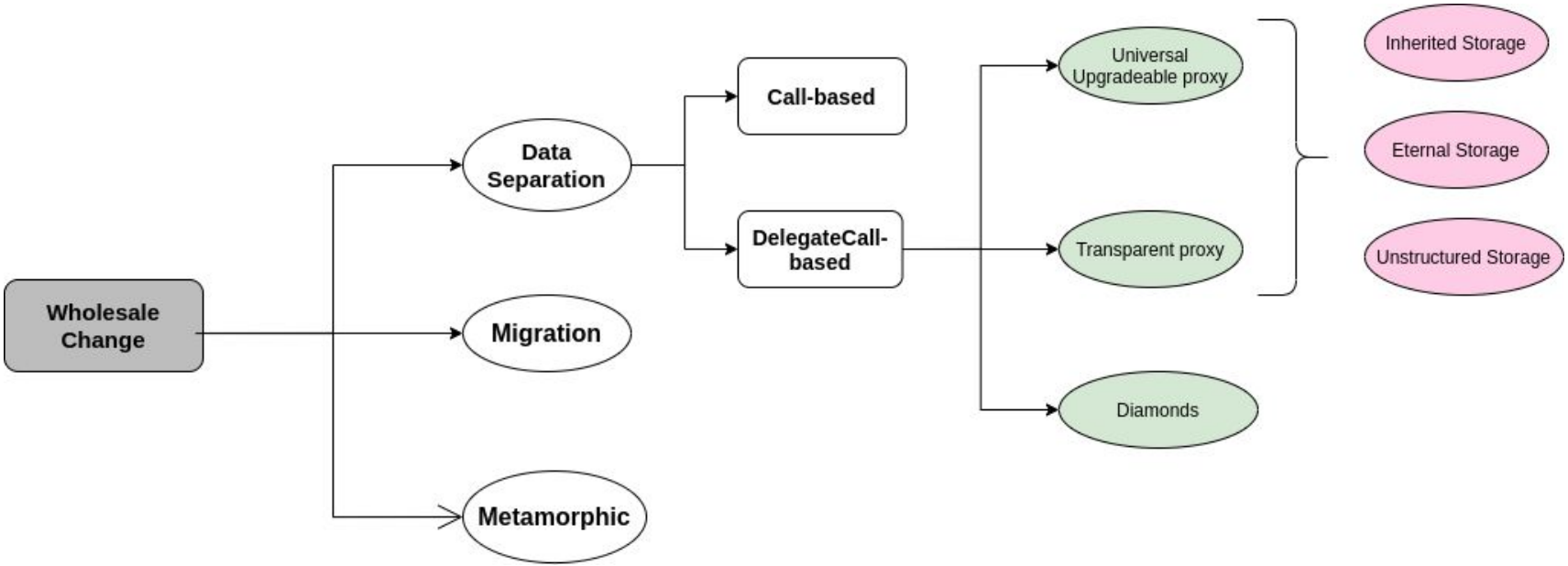
Pros and Cons of Proxy patterns

- Cons
  - Gas overhead for tx
  - Risks due to using Delegate Call opcode
    - Function selector clashes
    - Storage layout clashes
    - No constructor in Implementation contract
  - Cannot change the state layout (data layout)
    - Changing state of contract needs Migration plan

Delegate Call based pattern



Upgradeability Patterns



Metamorphic

Contract

Is there a way to
delete a contract

UNIVERSITÉ Concordia UNIVERSITY

Metamorphic

Contract

Is there a way to
delete a contract

UNIVERSITÉ Concordia UNIVERSITY

Metamorphic

Contract

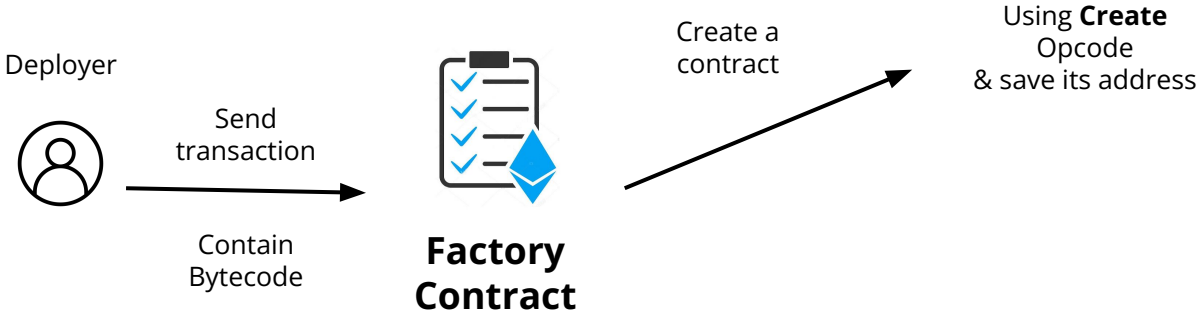
Is there a way to delete a contract

New Contract

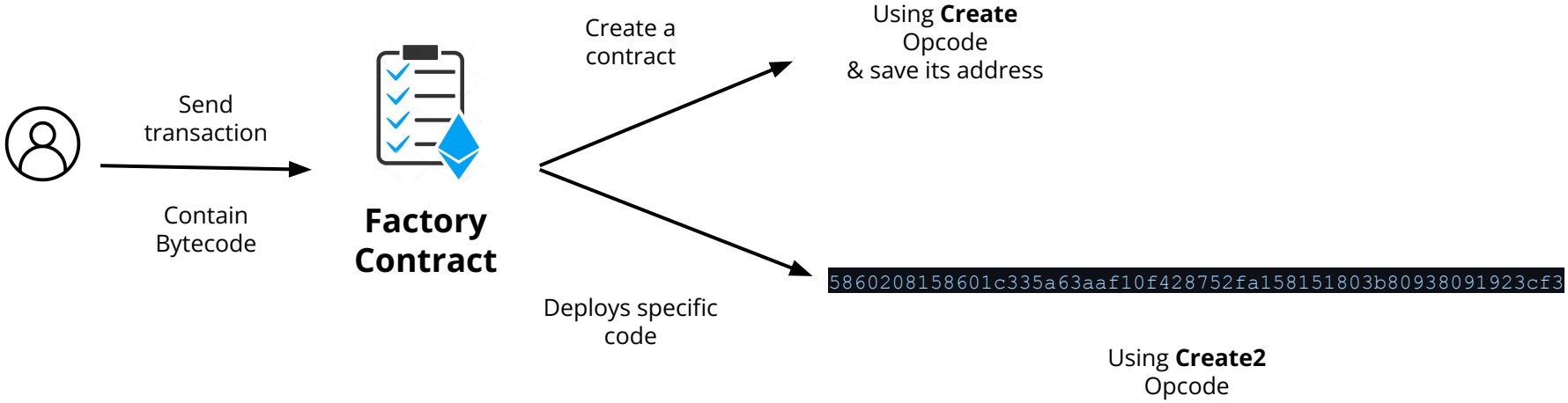
And then deploy another one on the same address

UNIVERSITÉ Concordia UNIVERSITY

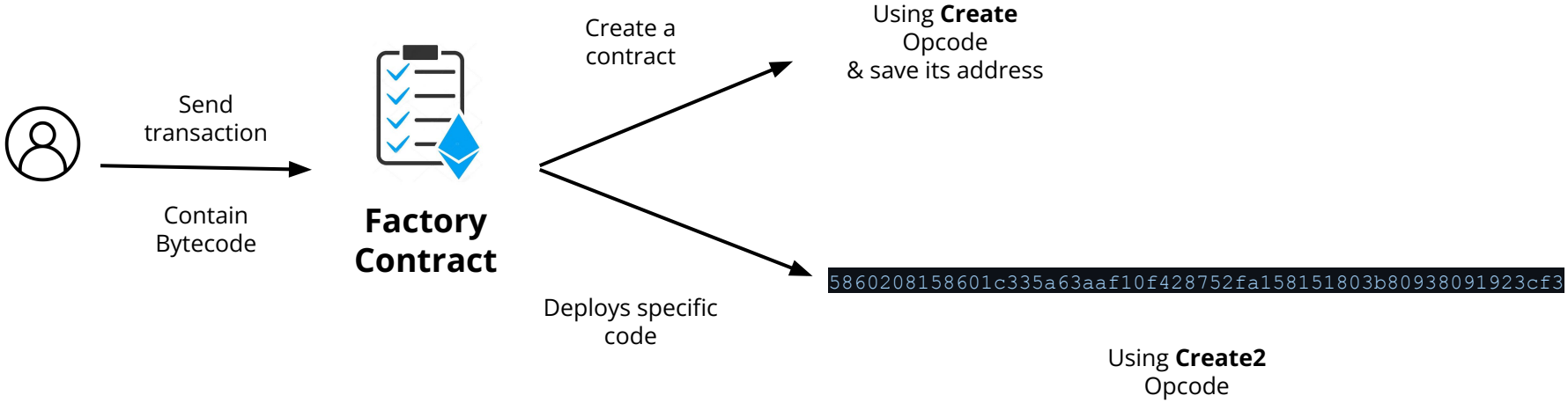
Metamorphic



Metamorphic



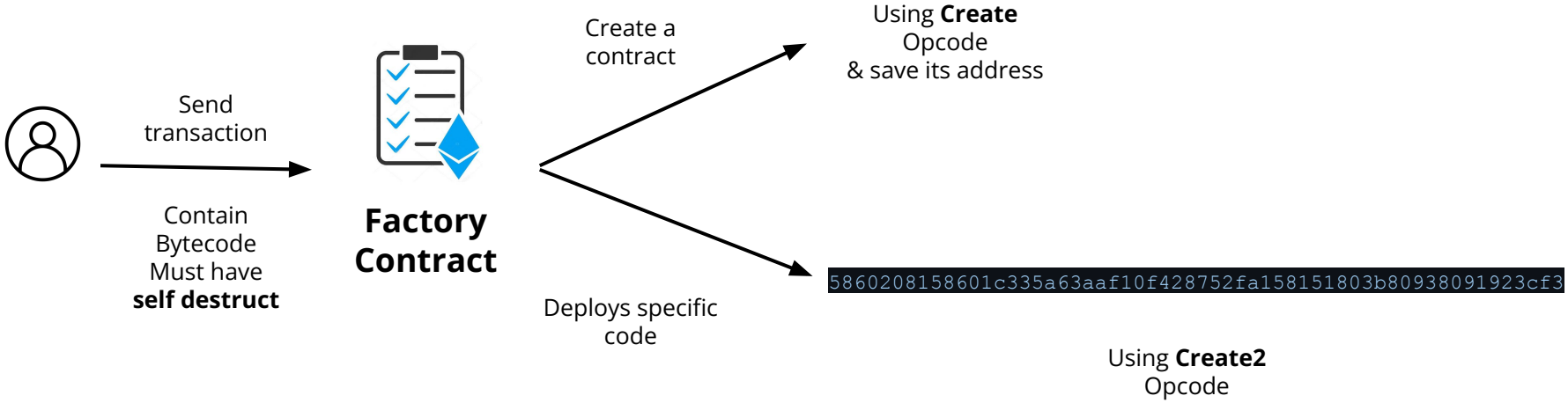
Metamorphic



Why **Create2 ?!** → The deployed address depends on the **deployer address** & **Bytecode**

→ **Re-deploying a contract in specific address**

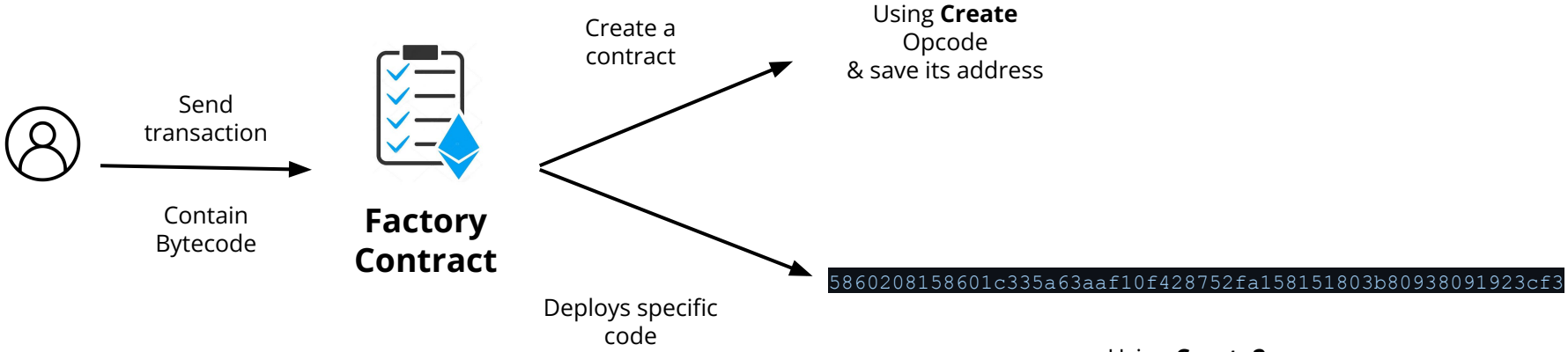
Metamorphic



Why **Create2 ?!** → But we want to change bytecode?!
This way we just deploys the same bytecode?

UNIVERSITÉ Concordia UNIVERSITY

Metamorphic



Why **Create2 ?!** → But we want to change bytecode?!
This way we just deploys the same bytecode?

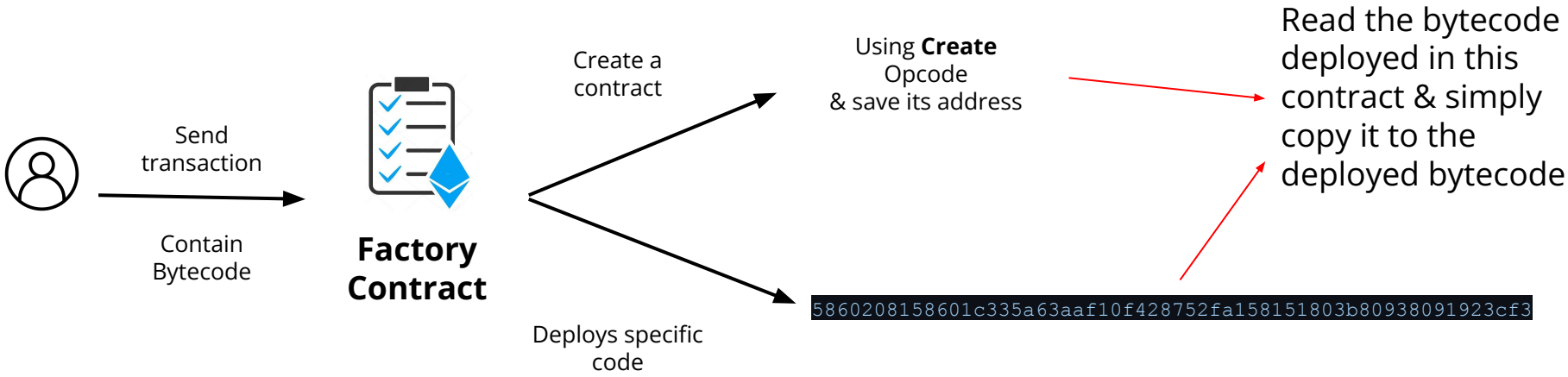
Using **Create2**
Opcode

NO

This bytecode
Is the key

UNIVERSITÉ Concordia UNIVERSITY

Metamorphic



Pros and Cons

- Pros
  - Can change the whole logic
  - Can change the whole state
  - Just using a single address
  - No gas overhead
  - Nothing to change for deployers to add this pattern
  - User endpoint address is not changed
- Cons
  - Self-destruct opcode removal
  - No way to keep state (self destruct deletes the state)
  - Downtime in upgrade events (because need to self destruct then deploy)

Evaluation of different methods

	Can replace entire logic	can replace pre-specified part of logic	Can replace entire state	can change pre-specified state variables	No need to deploy a new contract	No need to migrate state from old contract	No need to separate State and Logic	Function Selector Clashes Risk	Storage Clashes Risk	No indirection	User endpoint address not changed	Downtime in upgrade events	No need to change code to add the upgrade pattern	Need to change a state variable
Parameter change				✓	✓	✓	✓			✓	✓		✓	✓
Component Change		✓				✓	✓			⊠	✓		⊠	✓
Migration	✓		✓				✓			✓			✓	
Call-based	✓					✓								✓
DelegateCall-based	✓					✓		✓	✓		✓		⊠	✓
Diamonds	✓					✓		✓	✓✓		✓		⊠	✓
Metamorphic	✓		✓				✓			✓	✓	✓	✓	

Table 1. Evaluation

UNIVERSITÉ Concordia UNIVERSITY

Upgradeability!

No way!
We need **upgradeability**

How ?!
(immutability)

Who
Is responsible?!

Who is the decision maker?!

- Externally Owned Address (EOA)
  - 1 person decision

- Multi Signature Wallet
  - M out of N person

- Decentralized Governance
  - Governance token
  - Threshold on acceptance of a proposal

UNIVERSITÉ Concordia UNIVERSITY

Who is the decision maker?!

- Externally Owned Address (EOA)
  - 1 person decision
- Multi Signature Wallet
  - M out of N person

Compromised/ Malicious

Rug pools

- Decentralized Governance
  - Governance token
  - Threshold on acceptance of a proposal

UNIVERSITÉ Concordia UNIVERSITY

What’s Next?

Checking to Find the owners of the smart contract.

Measurement Study

- Our Main Goal in this part is to find contracts that uses upgradeability patterns

Results

- Number of Regular Upgradeable proxies:
    - **7470** Unique contracts
- Number of UUPS proxies:
    - **403** Unique Contracts
- Number of Beacon Proxies
    - **352** Unique contracts

UNIVERSITÉ Concordia UNIVERSITY

Thanks!

questions?

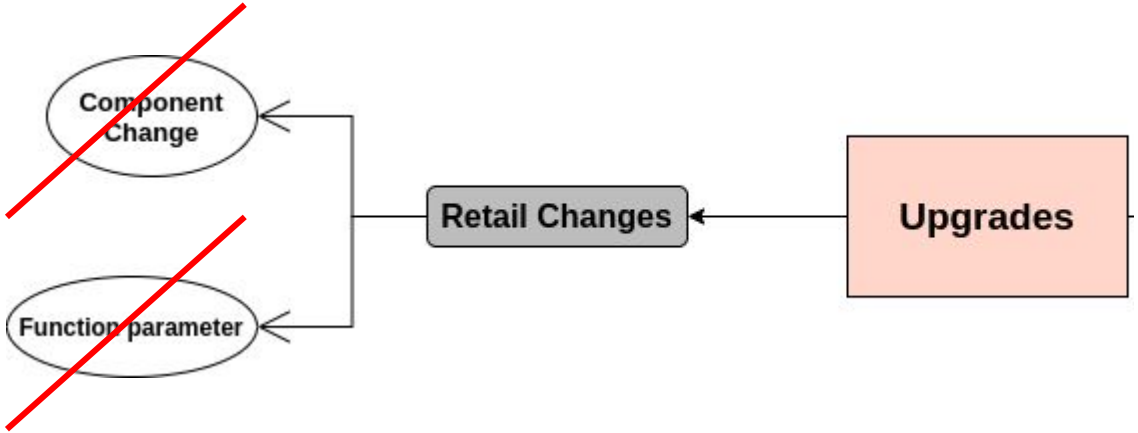
mehdi.salehi@concordia.ca

@Greatsaoshyant

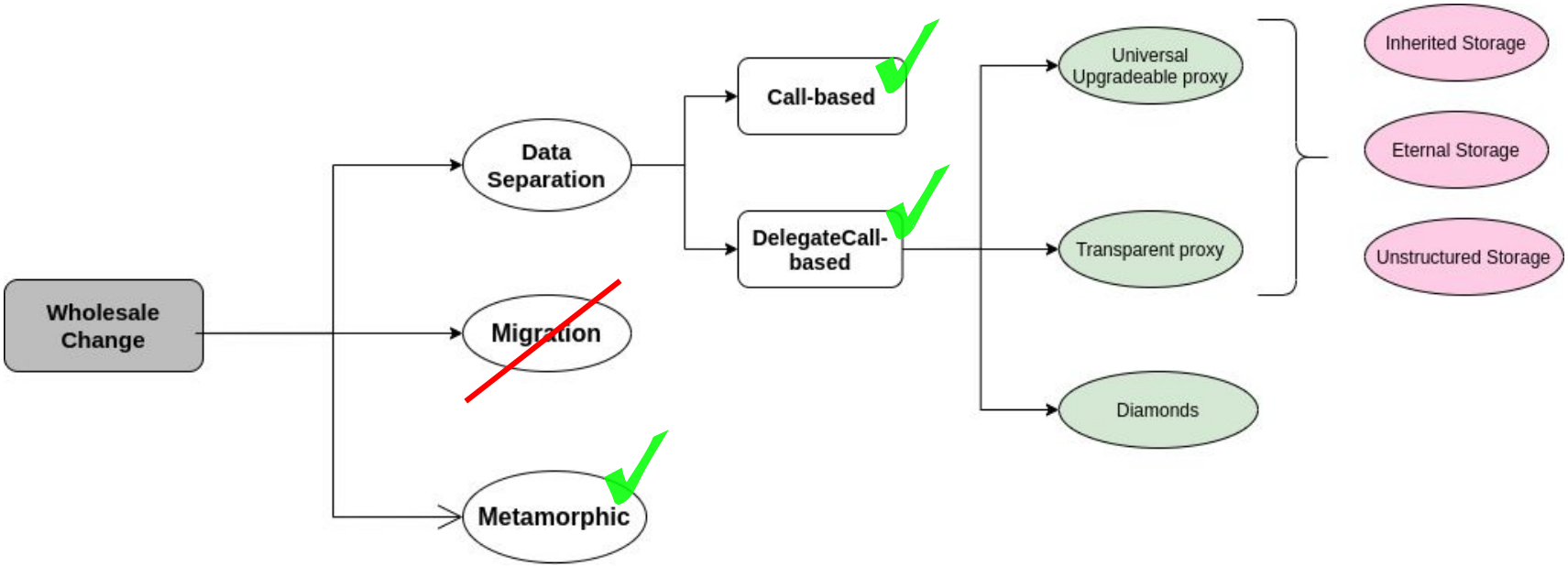
UNIVERSITÉ Concordia UNIVERSITY

Measurement Study

These patterns are more general and not just used for upgradeability



Measurement Study



Measurement Study

- Three main ways of analysis on Ethereum
    - Source Code analysis
        - Limited because we don’t have the HLL code for majority of bytecodes on Ethereum
        - Etherscan Verified Contracts

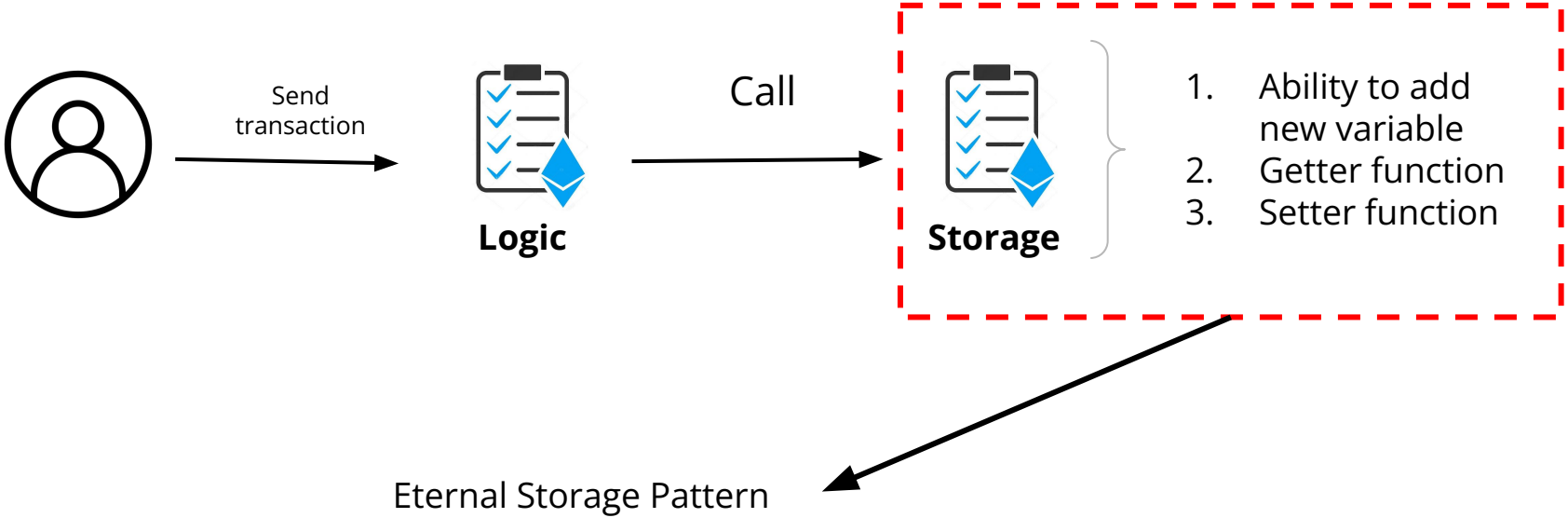
Measurement Study

- Three main ways of analysis on Ethereum
    - Smart Contract Code analysis
        - Etherscan Verified Contracts
    - Transaction-based Analysis
    - Bytecode-based Analysis

Measurement Study

- Try to find contracts that used These patterns:
    - Call-Based
        - This pattern is almost dead. It proposed in early 2016 and devs do not use it right now
    - Delegate-Call Based
        - This is the most favorite pattern that people are using on their contracts
    - Metamorphic
        - This pattern is new and not tested yet. Also because of risks and limitations in this pattern it is not widely used (On the plan to measure them)

Call-based are dead?! Prove it!



Call-based are dead?! Prove it!

1. Ability to add new variable
2. Getter function
3. Setter function

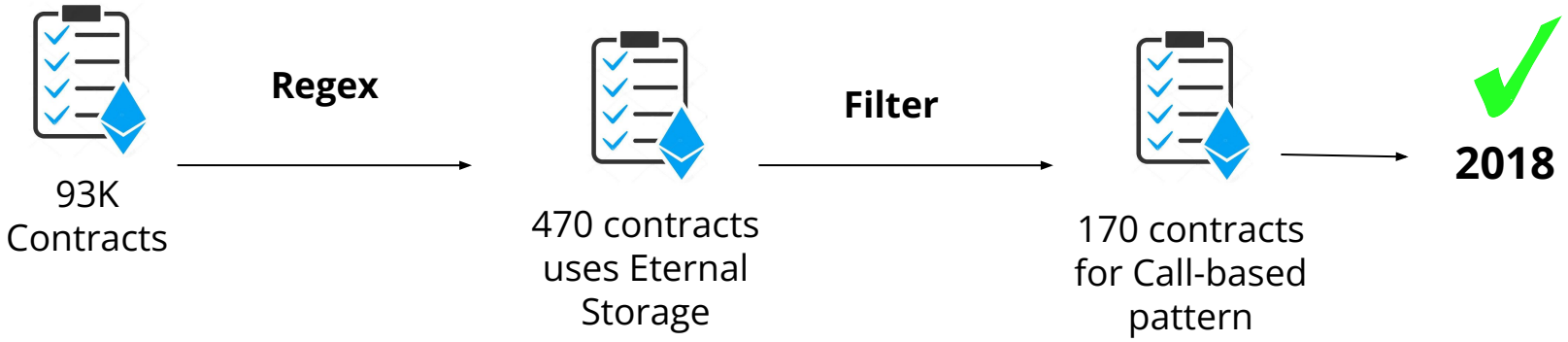
Find Eternal Storage Pattern

Using Verified Contracts on Etherscan

Smart-Contract-Sanctuary Github repo dataset

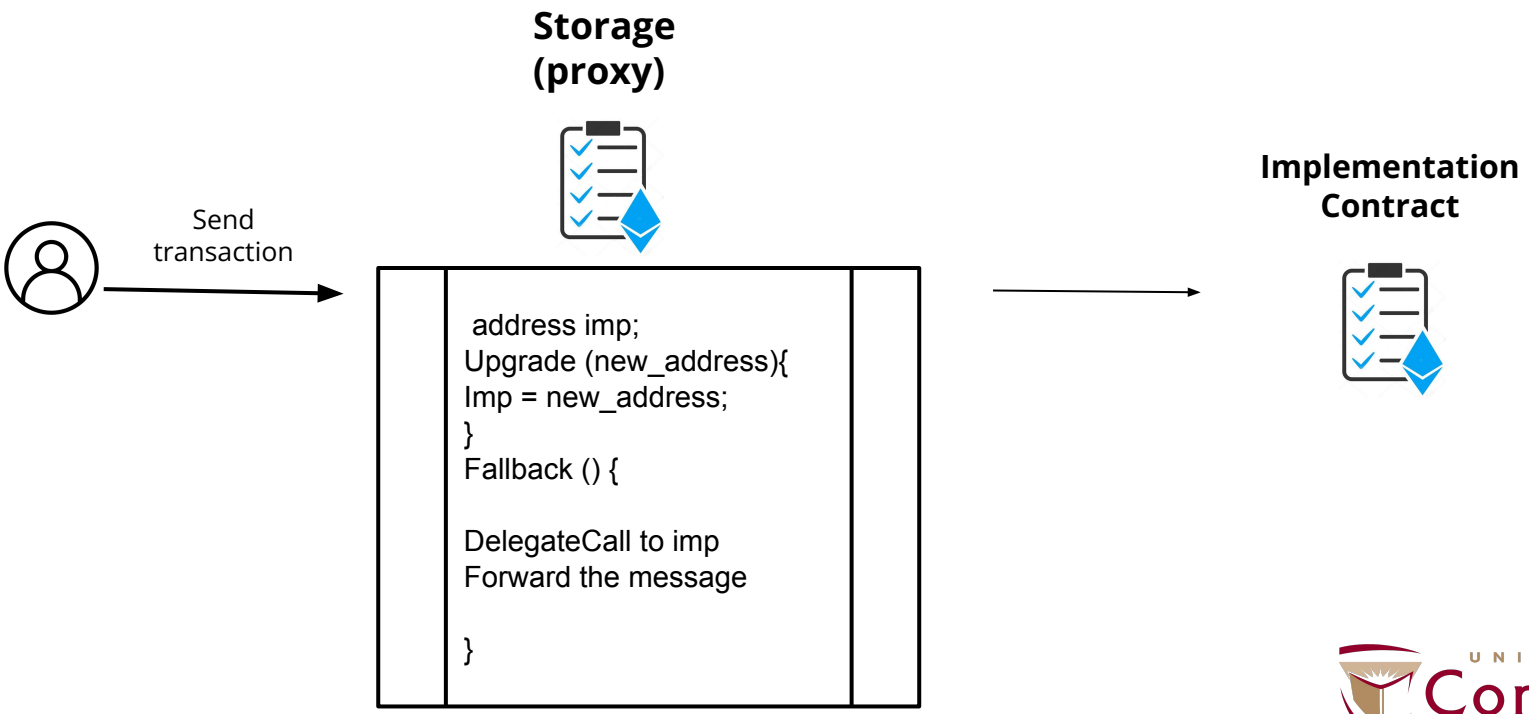
UNIVERSITÉ Concordia UNIVERSITY

Call-based are dead?! Prove it!

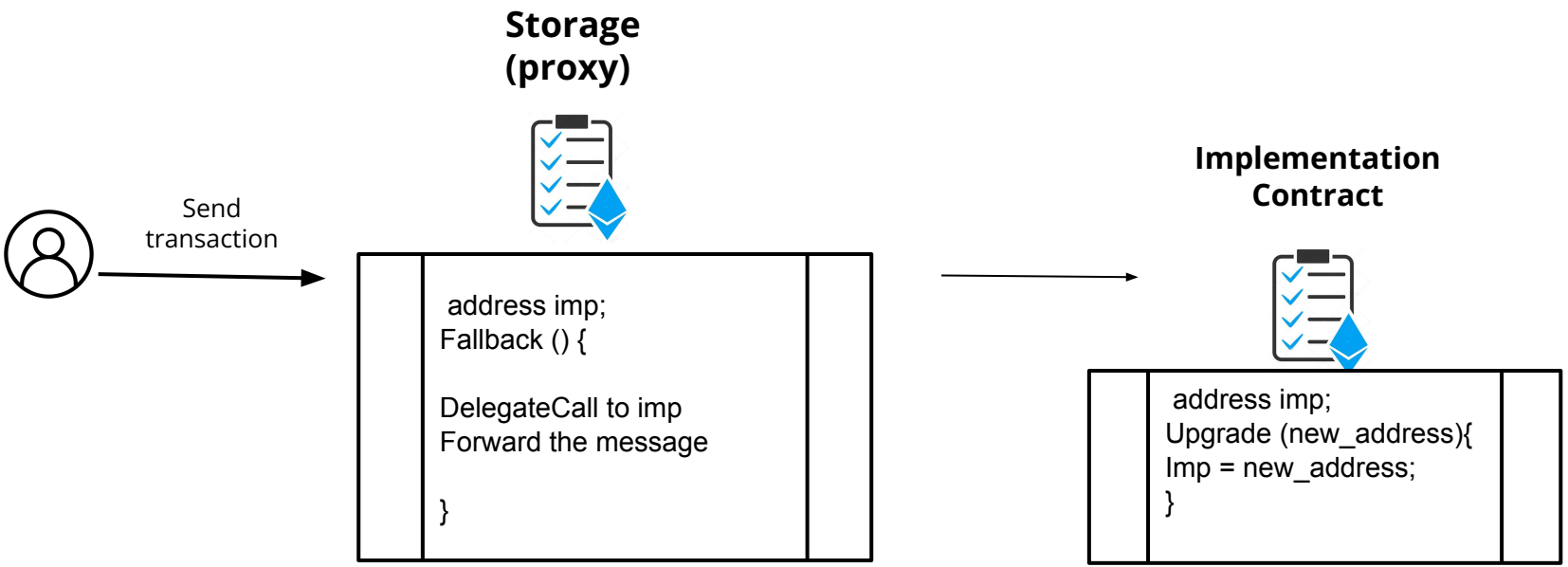


UNIVERSITÉ Concordia UNIVERSITY

Delegate proxy



UUPS Proxies



Measurement Study

- A hybrid analysis:
  - Transaction-based analysis
  - Bytecode-based analysis

Data of Transactions for 1 year (2 million Ethereum Blocks)

Measurement Study

Data of Transactions for 1 year → **Need to access to a full Archival Node** → **Request to replay transactions inside each block** → **Transaction Traces**

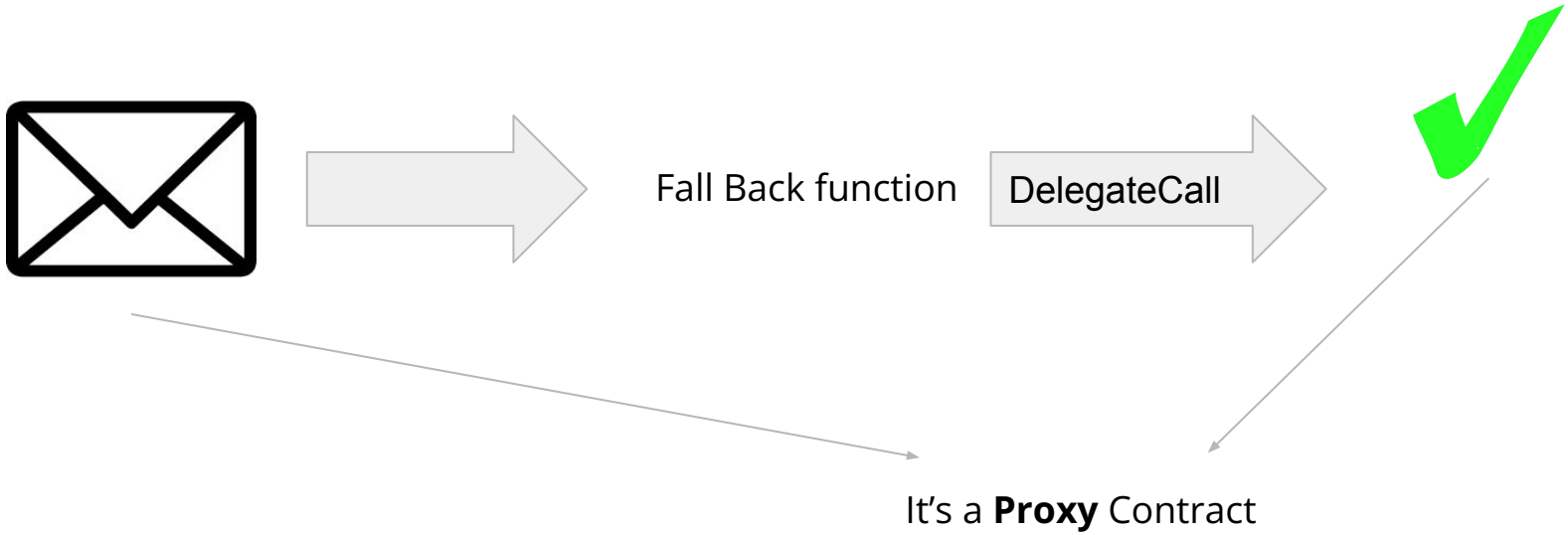
UNIVERSITÉ Concordia UNIVERSITY

Measurement Study

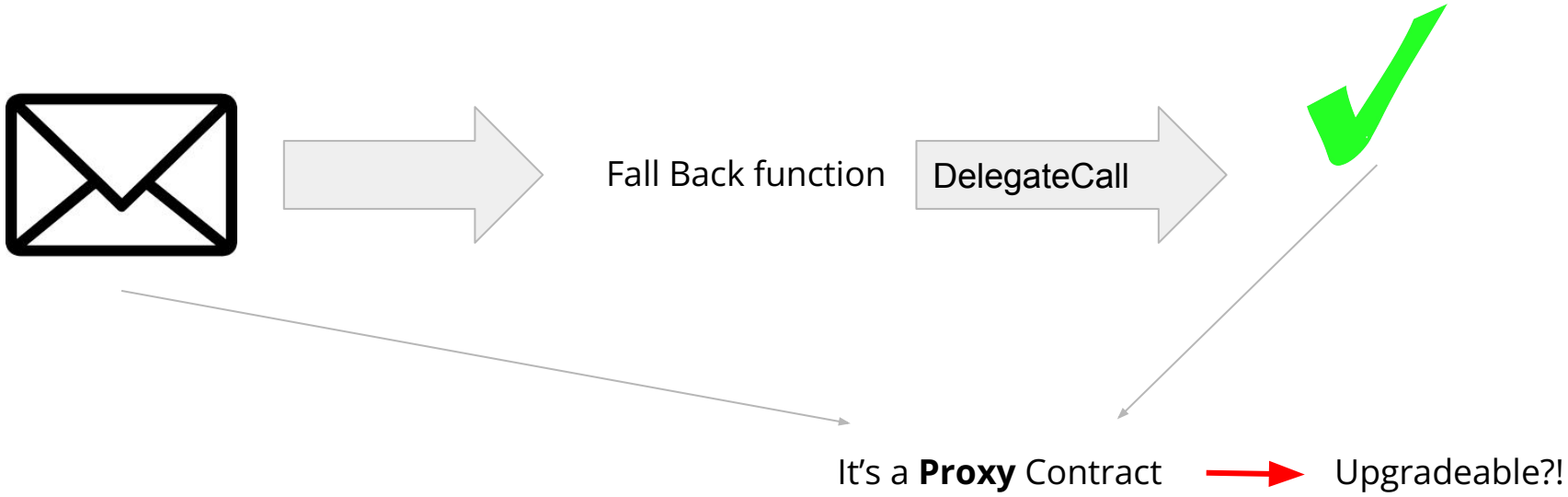
```
Execution trace:

[238519]: [sender] 0x8f891b3de263650668e4354f498b14c6eff418d1
    [214159]: ETH 0.3 [receiver] WyvernExchange.atomicMatch_(addrs=['0x7be8076f4ea4a4a
        [2782]: WyvernProxyRegistry.proxies(0xfc280e2233030025a09fedc5314a241c40283be8)
        [2613]: WyvernProxyRegistry.delegateProxyImplementation() => (AuthenticatedProx
        [2525]: 0x0b7cdd1ee5a92fe2636360820894d2d4bd9a9521.implementation() => (Authent
        [N/A]: ETH 0.0375 0x5b3256965e7c3cf26e11fcaf296dfc8807c01073.fallback() => ()
        [N/A]: ETH 0.2625 0xfc280e2233030025a09fedc5314a241c40283be8.fallback() => ()
        [107181]: (delegate) 0x0b7cdd1ee5a92fe2636360820894d2d4bd9a9521[AuthenticatedPr
            [2553]: WyvernProxyRegistry.contracts([receiver] WyvernExchange) => (True)
            [98284]: NFT.transferFrom(from=0xfc280e2233030025a09fedc5314a241c40283be8, t
```

UNIVERSITÉ Concordia UNIVERSITY

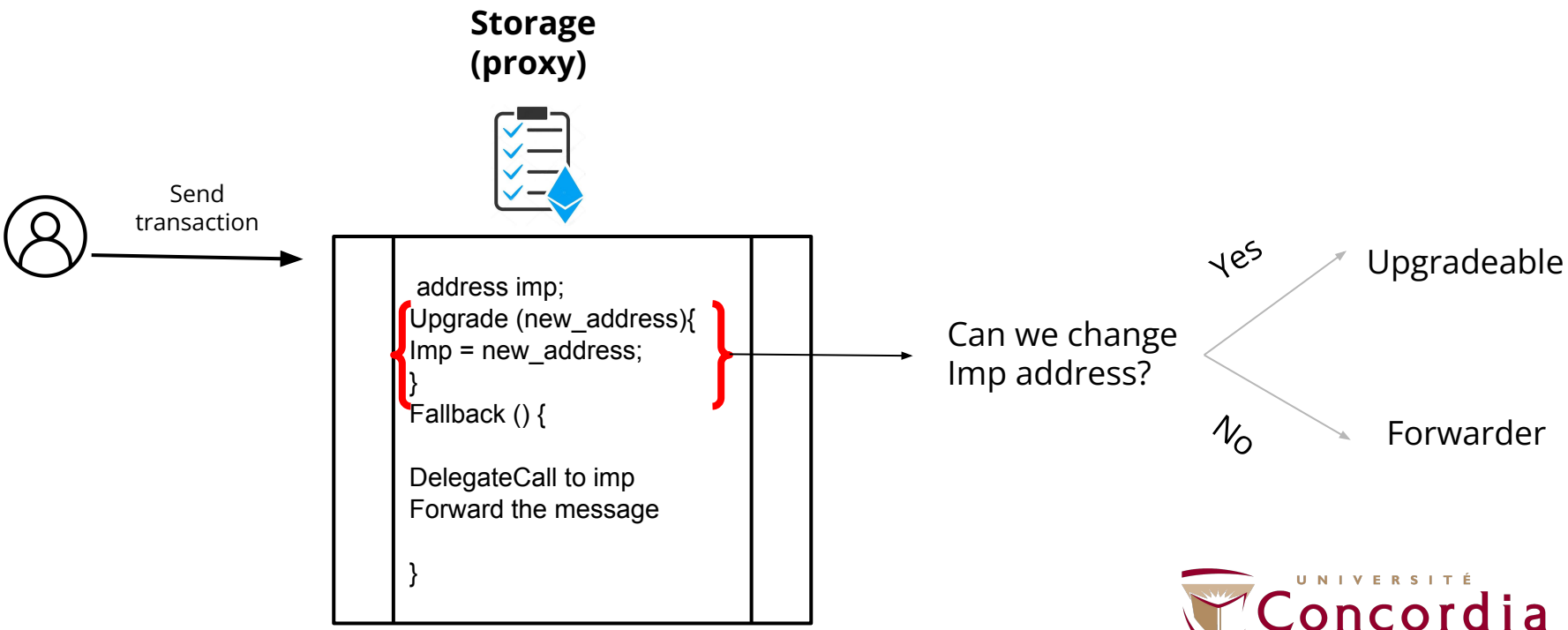
Measurement Study



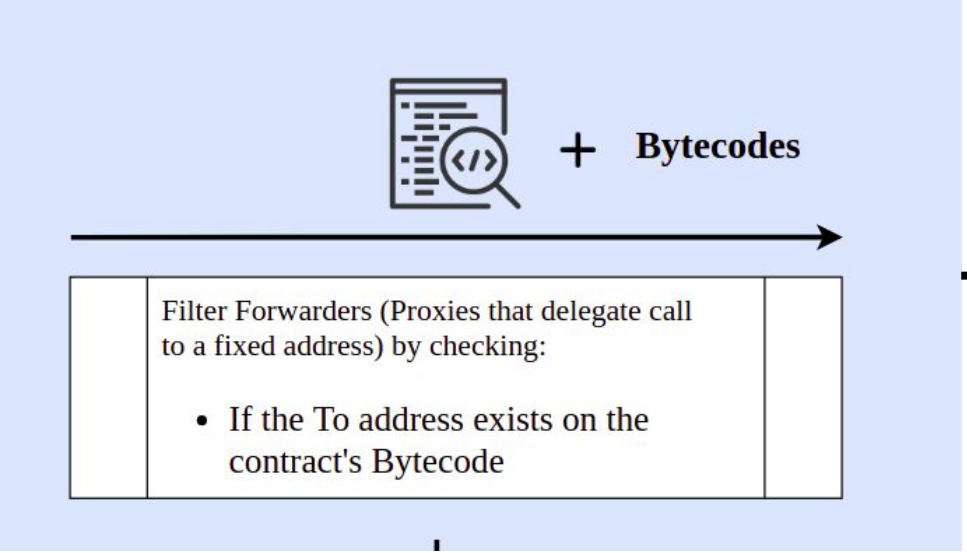
Measurement Study



Delegate proxy



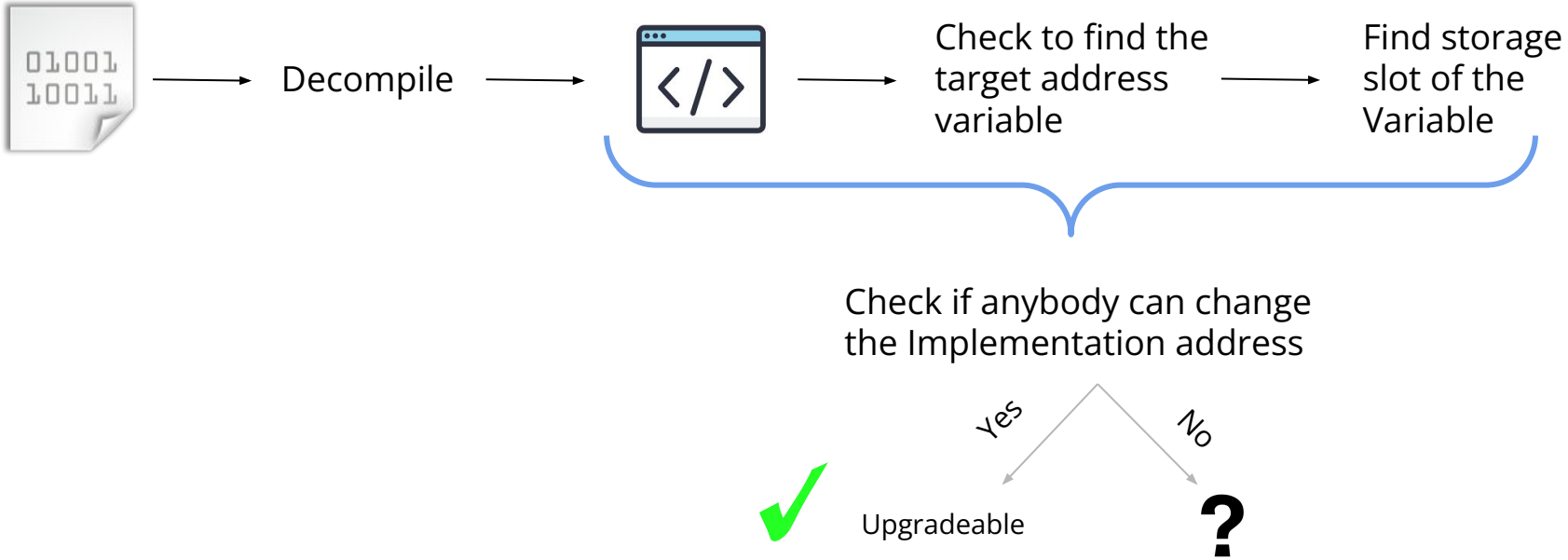
Measurement Study



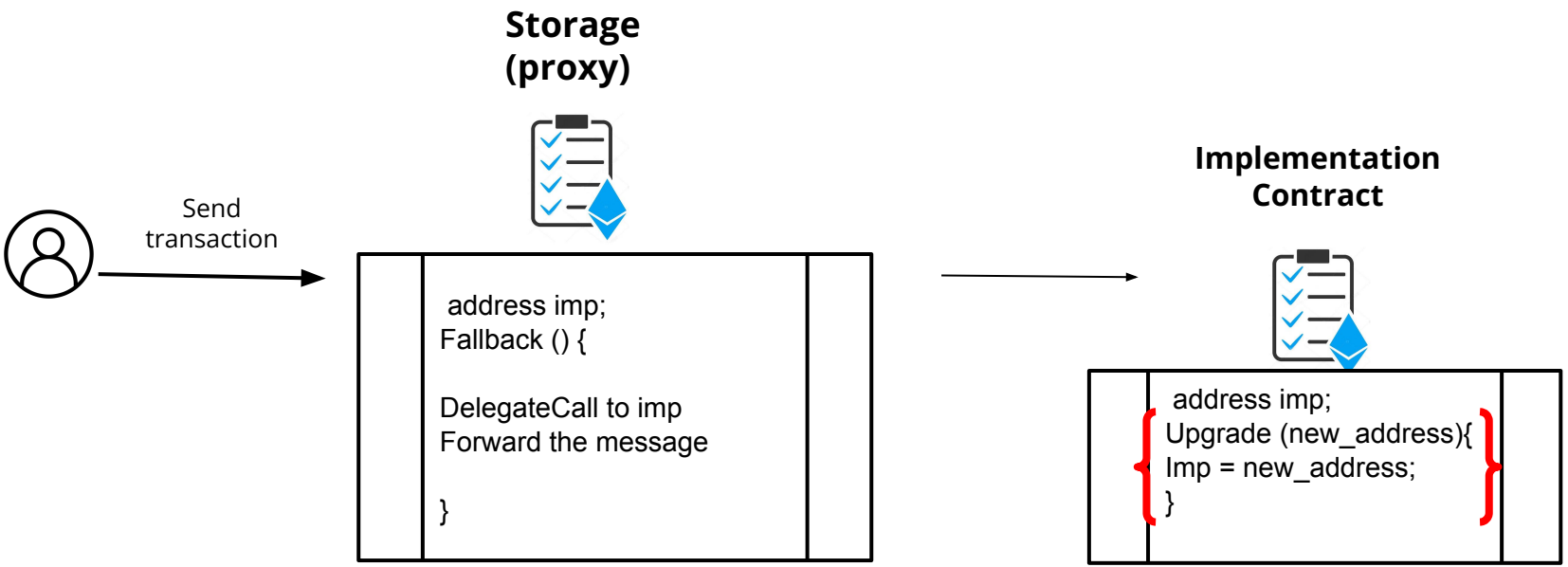
If the target address is hardcoded in the Bytecode

Forwarders

Measurement Study



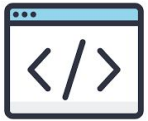
UUPS Proxies



UNIVERSITÉ Concordia UNIVERSITY

UUPS proxies

Decompile



Check if anybody can change the storage slot of Implementation address inside Proxy using Implementation contract

Implementation Contract

Yes

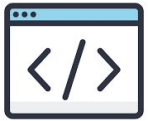
No

UUPS proxy

Beacon Proxies

Proxy Contract

Decompile



Check if the target address comes from an **External Call**

Yes

No

Beacon Proxy

Results

- Number of Regular Upgradeable proxies:
    - **7470** Unique contracts
- Number of UUPS proxies:
    - **403** Unique Contracts
- Number of Beacon Proxies
    - **352** Unique contracts

UNIVERSITÉ Concordia UNIVERSITY

What's Next?

Checks for possible Vulnerability

on UUPS proxies on my dataset

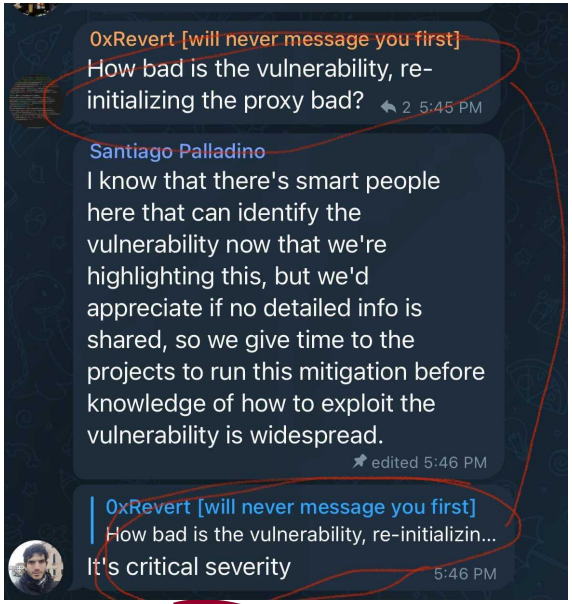
UUPSUpgradeable Vulnerability Post-mortem

General Announcements

spalladino OpenZeppelin Team 99 2 27d

In early September, we received two independent reports of a vulnerability in the UUPSUpgradeable base contract of the OpenZeppelin Contracts library, first released 6 in version 4.0 in April, 2021.

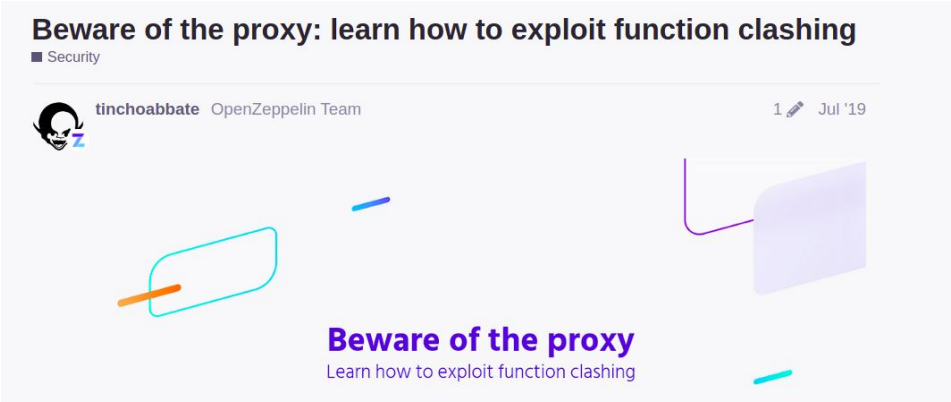
In this post, we share an overview of the contract and the vulnerability, executed mitigation strategies, and the implemented fix. To the best of our knowledge, we successfully executed mitigations and no project suffered loss of funds.



UNIVERSITÉ Concordia UNIVERSITY

What's Next?

Check to find contracts with known vulnerabilities
Regarding the upgradeability patterns



Who is the decision maker?!

- Externally Owned Address (EOA)
  - 1 person decision
- Multi Signature Wallet
  - M out of N person
- Decentralized Governance
  - Governance token
  - Threshold on acceptance of a proposal

UNIVERSITÉ Concordia UNIVERSITY

What’s Next?

Checking to Find the owners of the smart contract.

Thanks!

Do you have any questions?

@Greatsaoshyant

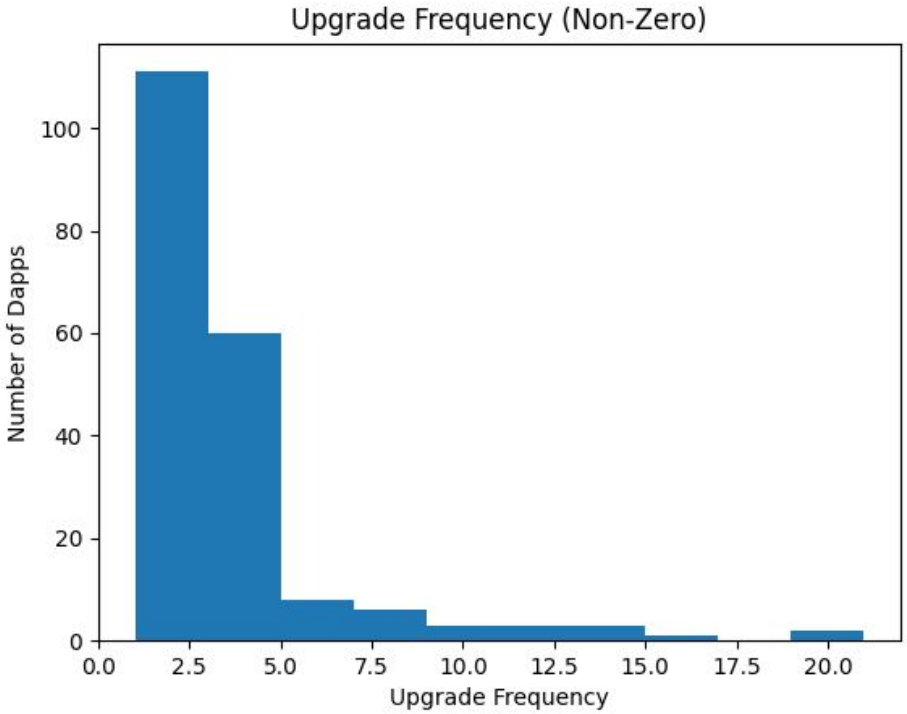
UNIVERSITÉ Concordia UNIVERSITY

Search based on interface

- **Upgrade Events:**
    - When implementation address changed
    - **Upgraded** event will be emitted
    - Find them based on **Ethereum Logs**
    - Querying a full node to get desired logs
- **Admin Change Events:**
    - When Ownership address changed
    - **AdminChanged** event will be emitted
    - Find them based on **Ethereum Logs**

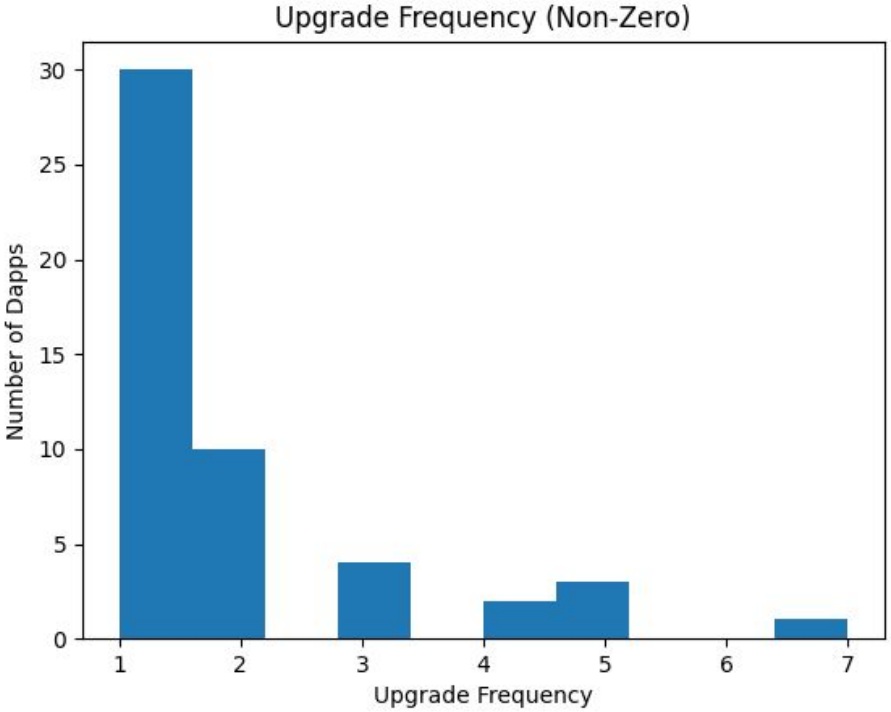
Results for Upgraded Events for V 2.6:

- **110** with **One** Upgrade
- **60** with Two Upgrade
- **2** with **20** upgrades
- 5 with Five upgrade
- 4 with 7 Upgrades
- 3 with 10,12,15,16 Upgrades
- 1 with 16 upgrades



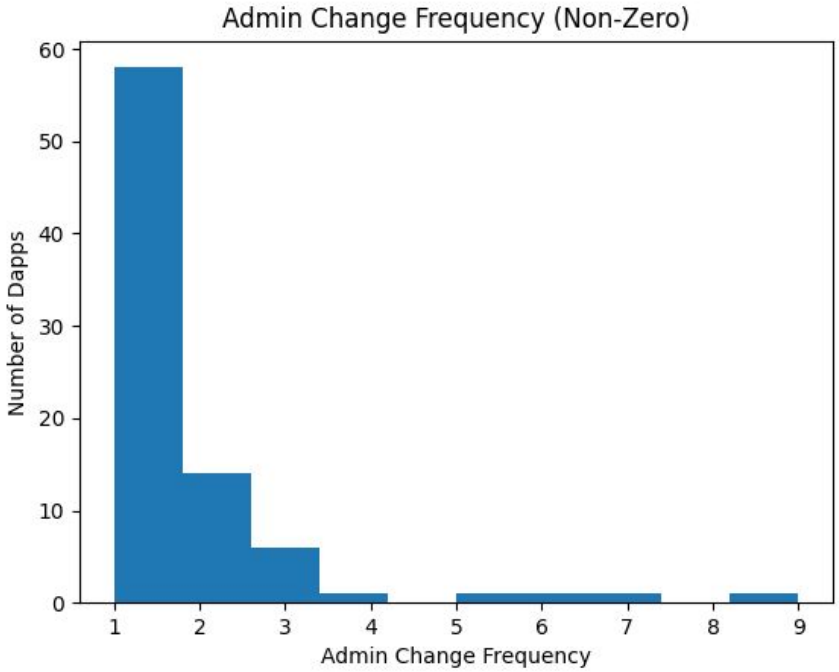
Results for Upgraded Events for V 3:

- **30** with **One** Upgrade
- **10** with **Two** Upgrade
- 4 with **Three** upgrade
- 2 with 4 Upgrades
- 3 with 5 Upgrades
- 2 with 7 Upgrades



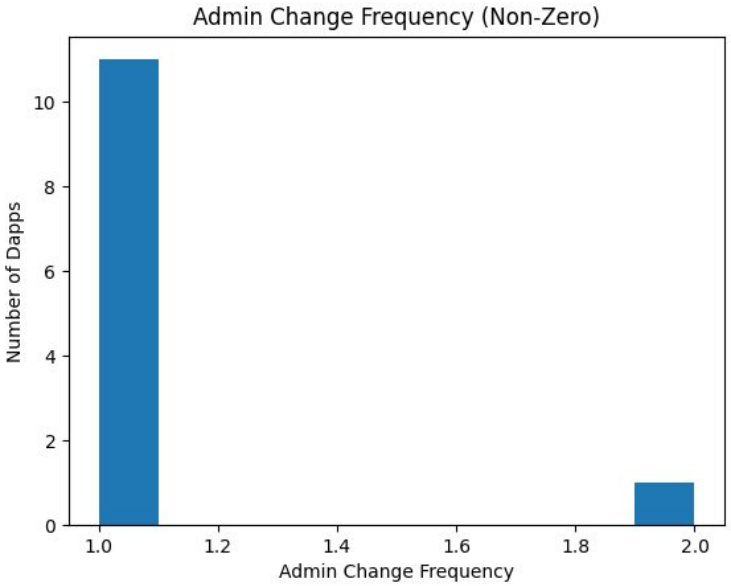
Results for AdminChanged Events for V 2.6:

- **57** with **One** Change
- **14** with **Two** change
- 4 with **Three** change
- 1 with 4 changes
- 1 Dapp with 5, 6,7,9 Changes



Results for AdminChanged Events for V 3:

- **11** with **One** Change
- **1** with **Two** change



Pauseability

- Upgrading a Dapp is like changing Wings of an airplane in the sky
- Because you cannot stop the network

- So, we need a button to push and stop our Dapp?!
- Who can push this button?

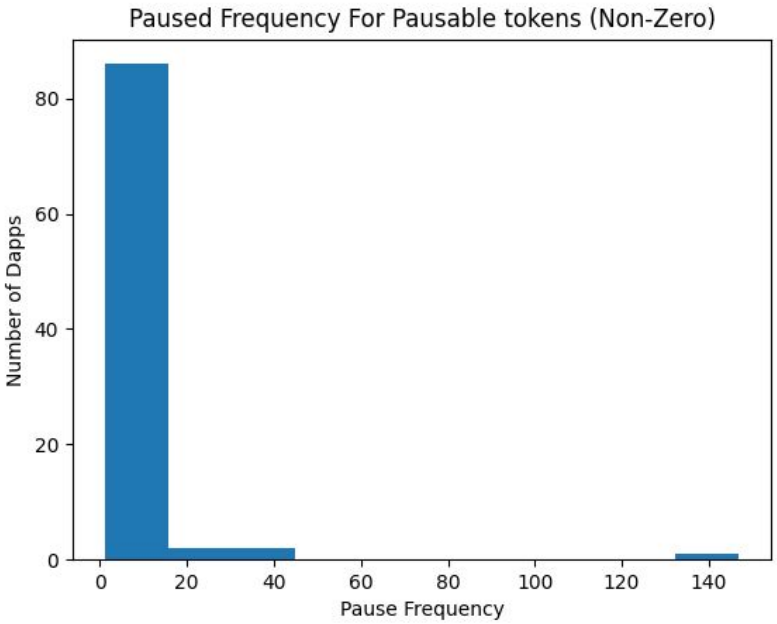
Pauseability

Pauseable Tokens

- **711** tokens that uses **PauseableToken** Library from OpenZepplin
- Using **AST** tool

Results for Pause on PauseableToken Library

- **86** Dapps **One** time paused
- **2** Dapps **Two** time paused
- **2** Dapps **4** time paused
- **1** Dapps **14** time paused



Results for Ownership transfer on PauseableToken

- **170** Dapps **One** time Changed
- **2** Dapps **Two** time changed
- **1** Dapps **3** time changed
- **1** Dapps **6** time changed