Rectifying Administrated ERC20 Tokens

23rd International Conference on Information and Communications Security (ICICS '21)

Nikolay Ivanov, Hanqing Guo, Qiben Yan

Department of Computer Science and Engineering
Secure and Intelligent Things Lab (SEIT Lab)
Michigan State University, East Lansing, MI, USA

2021

Outline

Outline

Background and Definitions

The person who deploys a smart contract does not receive any privilege or ownership by default.

Background and Definitions

The person who deploys a smart contract does not receive any privilege or ownership by default.

- **Effectively Ungoverned Smart Contracts**
  1. no privileged user(s); **OR**
  2. no privileged power.

Background and Definitions

The person who deploys a smart contract does not receive any privilege or ownership by default.

- **Effectively Ungoverned Smart Contracts**
  1. no privileged user(s); **OR**
  2. no privileged power.

  Example: Binance Token, Market Cap: $\approx$ 48 billion (Jul 12, 2021)

Background and Definitions

The person who deploys a smart contract does not receive any privilege or ownership by default.

- **Effectively Ungoverned Smart Contracts**
  1. no privileged user(s); **OR**
  2. no privileged power.

  Example: Binance Token, Market Cap: $\approx$ 48 billion (Jul 12, 2021)
- **Administrated Smart Contracts**
  1. privileged user(s); **AND**
  2. privileged power.

Background and Definitions

The person who deploys a smart contract does not receive any privilege or ownership by default.

- **Effectively Ungoverned Smart Contracts**
  1. no privileged user(s); **OR**
  2. no privileged power.

  Example: Binance Token, Market Cap: $\approx$ 48 billion (Jul 12, 2021)
- **Administrated Smart Contracts**
  1. privileged user(s); **AND**
  2. privileged power.

  Example: TetherUSD, Market Cap: $\approx$ 62 billion (Jul 12, 2021)

```
1    function issue(uint amount) public onlyOwner {
2        require(_totalSupply + amount > _totalSupply);
3        require(balances[owner] + amount > balances[owner]);
4
5        balances[owner] += amount;
6        _totalSupply += amount;
7        Issue(amount);
8    }
```

Background and Definitions

The person who deploys a smart contract does not receive any privilege or ownership by default.

- **Effectively Ungoverned Smart Contracts**
  1. no privileged user(s); **OR**
  2. no privileged power.

  Example: Binance Token, Market Cap: $\approx$ 48 billion (Jul 12, 2021)
- **Administrated Smart Contracts**
  1. privileged user(s); **AND**
  2. privileged power.

  Example: TetherUSD, Market Cap: $\approx$ 62 billion (Jul 12, 2021)

```
1    function issue(uint amount) public onlyOwner {
2        require(_totalSupply + amount > _totalSupply);
3        require(balances[owner] + amount > balances[owner]);
4
5        balances[owner] += amount;
6        _totalSupply += amount;
7        Issue(amount);
8    }
```

Background and Definitions

The person who deploys a smart contract does not receive any privilege or ownership by default.

- **Effectively Ungoverned Smart Contracts**
  1. no privileged user(s); **OR**
  2. no privileged power.

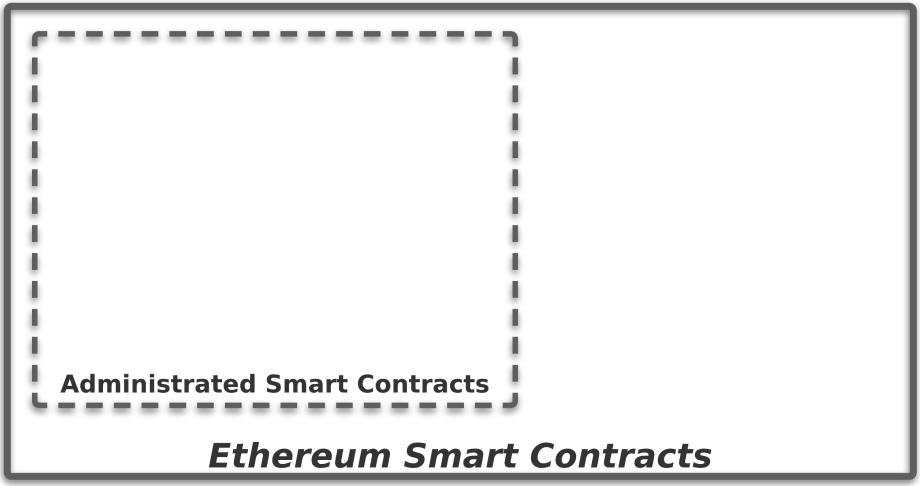
  Example: Binance Token, Market Cap: $\approx$ 48 billion (Jul 12, 2021)
- **Administrated Smart Contracts**
  1. privileged user(s); **AND**
  2. privileged power.

  Example: TetherUSD, Market Cap: $\approx$ 62 billion (Jul 12, 2021)

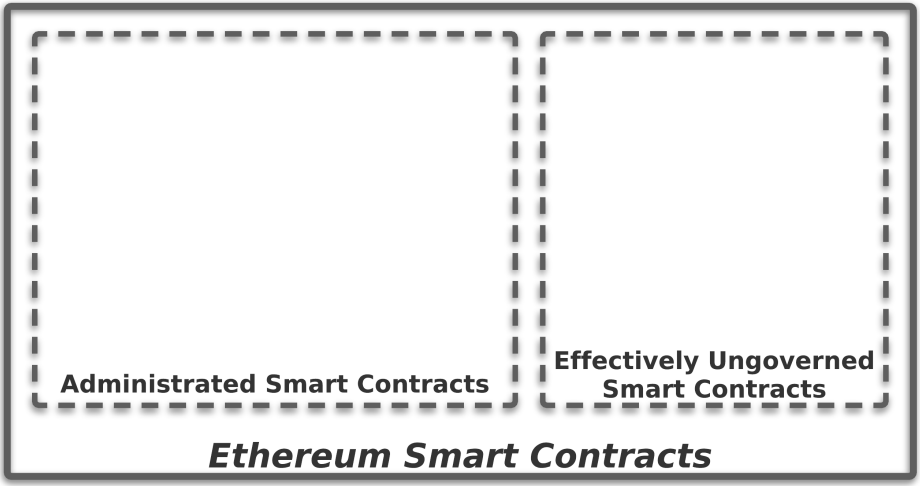
```
1    function issue(uint amount) public onlyOwner {
2        require(_totalSupply + amount > _totalSupply);
3        require(balances[owner] + amount > balances[owner]);
4
5        balances[owner] += amount;
6        _totalSupply += amount;
7        Issue(amount);
8    }
```

Smart Contracts: Management Taxonomy

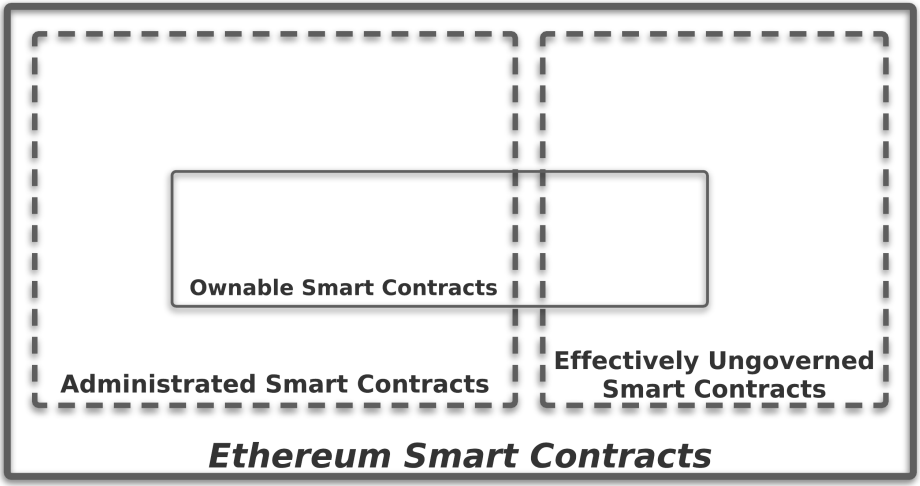
Smart Contracts: Management Taxonomy



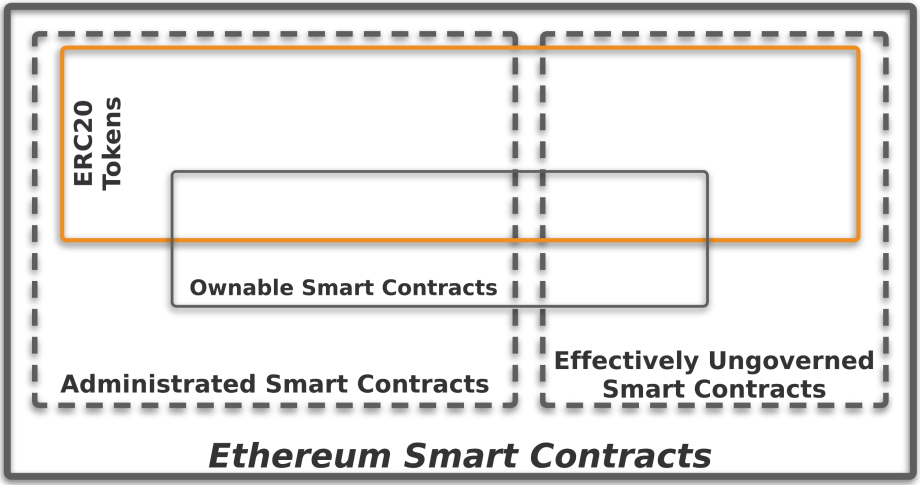
Smart Contracts: Management Taxonomy



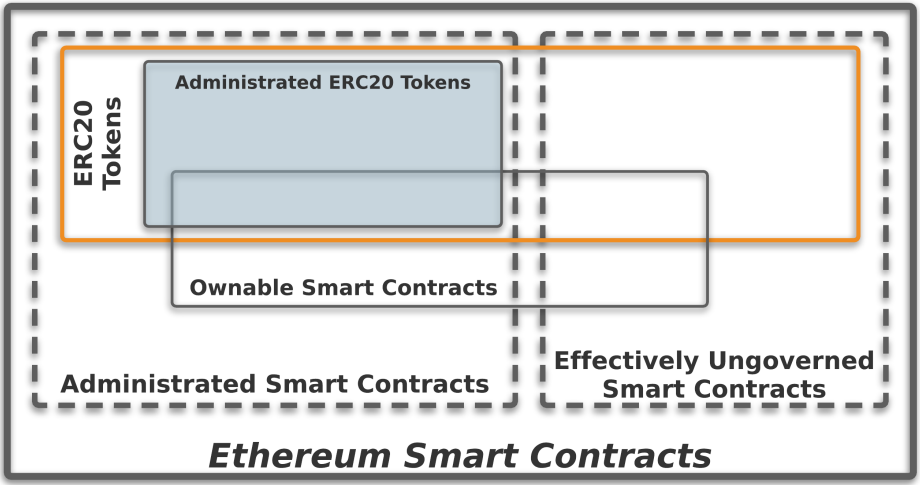
Smart Contracts: Management Taxonomy



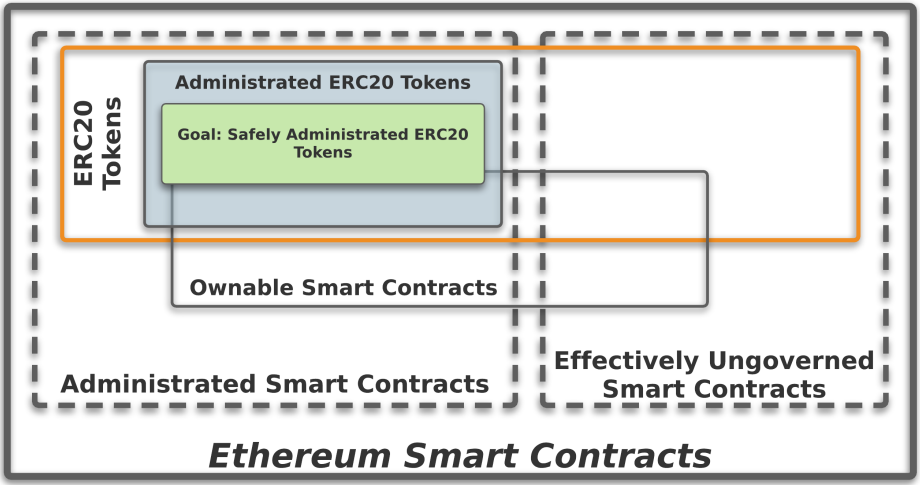
Smart Contracts: Management Taxonomy



Smart Contracts: Management Taxonomy



Smart Contracts: Management Taxonomy



Are Administrated Tokens Safe?

Traditional Financial Services

Administrated ERC20 Tokens

Are Administrated Tokens Safe?

Traditional Financial Services

- Banks, PayPal, VISA/MasterCard, etc.

Administrated ERC20 Tokens

- Deployed on Ethereum Mainnet;

Are Administrated Tokens Safe?

Traditional Financial Services

- Banks, PayPal, VISA/MasterCard, etc.
- Brick-and-mortar business;

Administrated ERC20 Tokens

- Deployed on Ethereum Mainnet;
- Can be created by anyone in minutes;

Are Administrated Tokens Safe?

Traditional Financial Services

- Banks, PayPal, VISA/MasterCard, etc.
- Brick-and-mortar business;
- Licensing and strict regulation;

Administrated ERC20 Tokens

- Deployed on Ethereum Mainnet;
- Can be created by anyone in minutes;
- Loose or no regulation;

Are Administrated Tokens Safe?

Traditional Financial Services

- Banks, PayPal, VISA/MasterCard, etc.
- Brick-and-mortar business;
- Licensing and strict regulation;
- Corporate government with checks and balances;

Administrated ERC20 Tokens

- Deployed on Ethereum Mainnet;
- Can be created by anyone in minutes;
- Loose or no regulation;
- Security and power hinges on a single private key;

Are Administrated Tokens Safe?

Traditional Financial Services

- Banks, PayPal, VISA/MasterCard, etc.
- Brick-and-mortar business;
- Licensing and strict regulation;
- Corporate government with checks and balances;
- Any abuse of power is localized and hard to conduct;

Administrated ERC20 Tokens

- Deployed on Ethereum Mainnet;
- Can be created by anyone in minutes;
- Loose or no regulation;
- Security and power hinges on a single private key;
- The administrator can take all the money in minutes;

Are Administrated Tokens Safe?

Traditional Financial Services

- Banks, PayPal, VISA/MasterCard, etc.
- Brick-and-mortar business;
- Licensing and strict regulation;
- Corporate government with checks and balances;
- Any abuse of power is localized and hard to conduct;
- Attacks are local, limited in scope, and can be remediated.

Administrated ERC20 Tokens

- Deployed on Ethereum Mainnet;
- Can be created by anyone in minutes;
- Loose or no regulation;
- Security and power hinges on a single private key;
- The administrator can take all the money in minutes;
- An attacker only needs to steal a 64-byte secret to steal all the money. No remediation possible.

Are Administrated Tokens Safe?

Traditional Financial Services

- Banks, PayPal, VISA/MasterCard, etc.
- Brick-and-mortar business;
- Licensing and strict regulation;
- Corporate government with checks and balances;
- Any abuse of power is localized and hard to conduct;
- Attacks are local, limited in scope, and can be remediated.

Administrated ERC20 Tokens

- Deployed on Ethereum Mainnet;
- Can be created by anyone in minutes;
- Loose or no regulation;
- Security and power hinges on a single private key;
- The administrator can take all the money in minutes;
- An attacker only needs to steal a 64-byte secret to steal all the money. No remediation possible.

Administrated ERC20 tokens are less safe than the traditional services they are designed to disrupt!

Outline

Self-Destruction

```
1 function kill() public onlyAdmin {
2   selfdestruct(payable(msg.sender));
3 }
```

Contract address: 0xbF3d14995D4A4A719A3B9101DE60baa47De60F39

Self-Destruction

```
1 function kill() public onlyAdmin {
2   selfdestruct(payable(msg.sender));
3 }
```

Contract address: 0xbF3d14995D4A4A719A3B9101DE60baa47De60F39

Self-Destruction

```
1 function kill() public onlyAdmin {
2   selfdestruct(payable(msg.sender));
3 }
```

Contract address: 0xbF3d14995D4A4A719A3B9101DE60baa47De60F39

Self-Destruction

```
1 function kill() public onlyAdmin {
2   selfdestruct(payable(msg.sender));
3 }
```

Contract address: 0xbF3d14995D4A4A719A3B9101DE60baa47De60F39

Deprecation

```
1 // deprecate current contract in favour of a new one
2 function deprecate(address _upgradedAddress) public onlyOwner {
3   deprecated = true;
4   upgradedAddress = _upgradedAddress;
5   Deprecate(_upgradedAddress);
6 }
```

Contract address: 0xdAC17F958D2ee523a2206206994597C13D831ec7

Deprecation

```
1 // deprecate current contract in favour of a new one
2 function deprecate(address _upgradedAddress) public onlyOwner {
3   deprecated = true;
4   upgradedAddress = _upgradedAddress;
5   Deprecate(_upgradedAddress);
6 }
```

Contract address: 0xdAC17F958D2ee523a2206206994597C13D831ec7

Deprecation

```
1 // deprecate current contract in favour of a new one
2 function deprecate(address _upgradedAddress) public onlyOwner {
3   deprecated = true;
4   upgradedAddress = _upgradedAddress;
5   Deprecate(_upgradedAddress);
6 }
```

Contract address: 0xdAC17F958D2ee523a2206206994597C13D831ec7

Deprecation

```
1 // deprecate current contract in favour of a new one
2 function deprecate(address _upgradedAddress) public onlyOwner {
3   deprecated = true;
4   upgradedAddress = _upgradedAddress;
5   Deprecate(_upgradedAddress);
6 }
```

Contract address: 0xdAC17F958D2ee523a2206206994597C13D831ec7

Change of Address

```
1 function setFee(address to) public onlyOwner{
2   fee = to;
3 }
```

Contract address: 0x350BDC46d931712d83ef989725Ba4904C487F360

Change of Address

```
1 function setFee(address to) public onlyOwner{
2   fee = to;
3 }
```

Contract address: 0x350BDC46d931712d83ef989725Ba4904C487F360

Change of Address

```
1 function setFee(address to) public onlyOwner{
2   fee = to;
3 }
```

Contract address: 0x350BDC46d931712d83ef989725Ba4904C487F360

Change of Address

```
1 function setFee(address to) public onlyOwner{
2   fee = to;
3 }
```

Fee address replacement is easily exploitable:

Ivanov et al. *"Targeting the Weakest Link: Social Engineering Attacks in Ethereum Smart Contracts"*, Asia CCS 2021

Contract address: 0x350BDC46d931712d83ef989725Ba4904C487F360

Minting / Burning

```
1 function mint(address account, uint amount) public onlyOwner {
2   _mint(account, amount);
3 }
4 function burn(address account, uint amount) public onlyOwner {
5   _burn(account, amount);
6 }
```

Contract address: 0x82bfdd53dd95efa2c3e92543f28d46c566bf4b8a

Minting / Burning

```
1 function mint(address account, uint amount) public onlyOwner {
2   _mint(account, amount);
3 }
4 function burn(address account, uint amount) public onlyOwner {
5   _burn(account, amount);
6 }
```

Contract address: 0x82bfdd53dd95efa2c3e92543f28d46c566bf4b8a

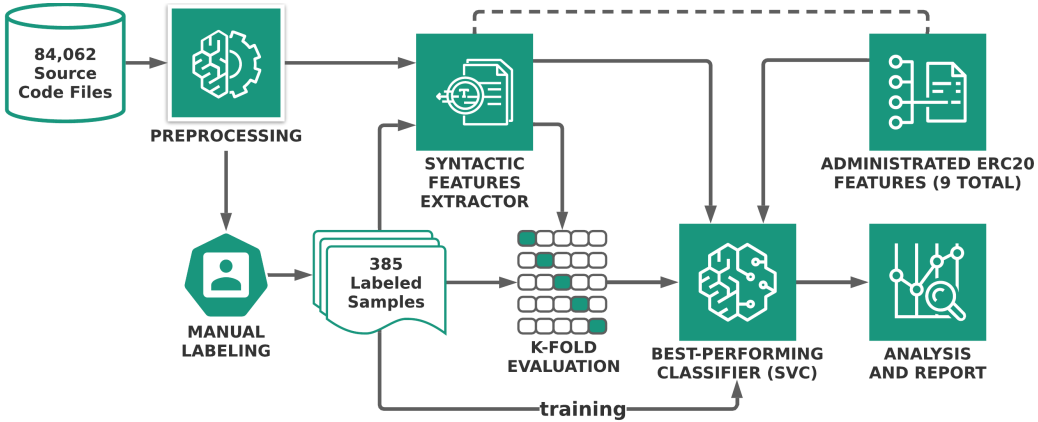
Minting / Burning

```
1 function mint(address account, uint amount) public onlyOwner {
2   _mint(account, amount);
3 }
4 function burn(address account, uint amount) public onlyOwner {
5   _burn(account, amount);
6 }
```

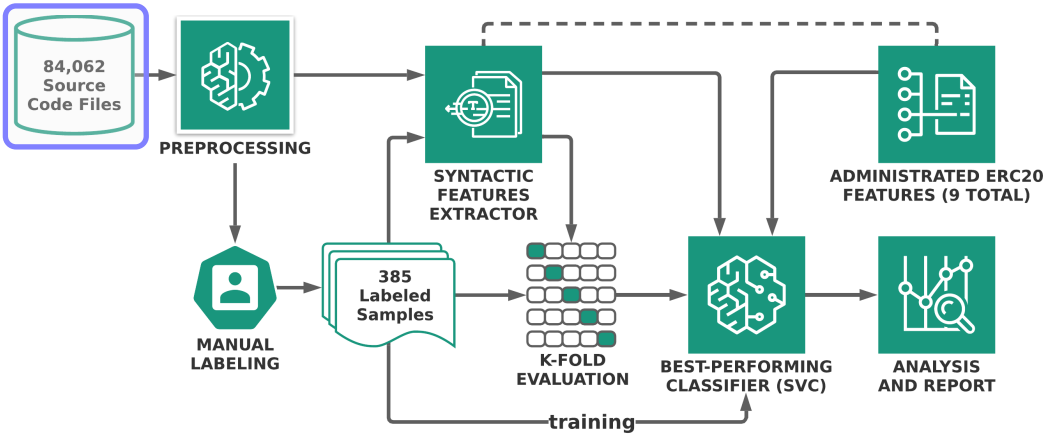
Contract address: 0x82bfdd53dd95efa2c3e92543f28d46c566bf4b8a

Outline

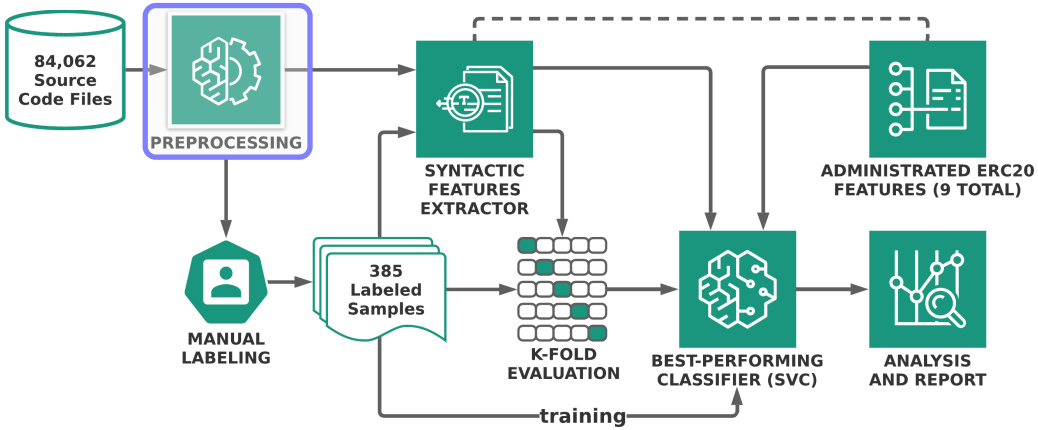
Administrated Tokens on Mainnet



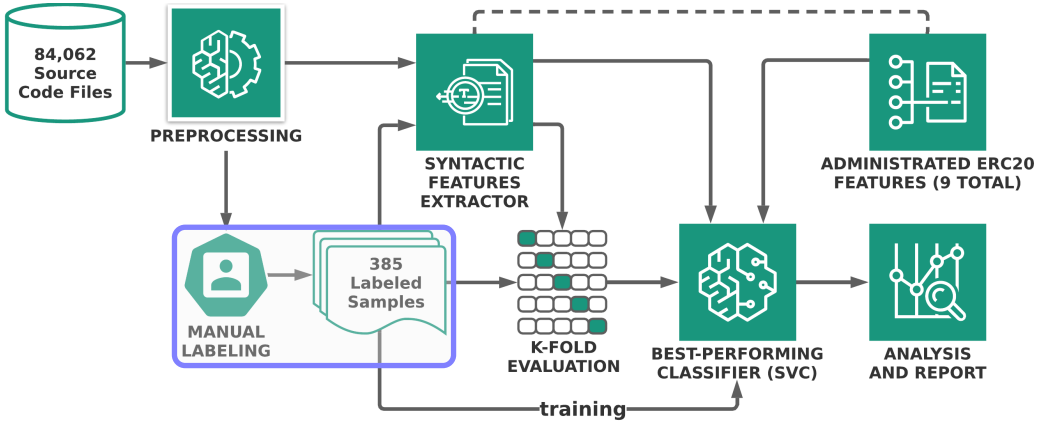
Administrated Tokens on Mainnet



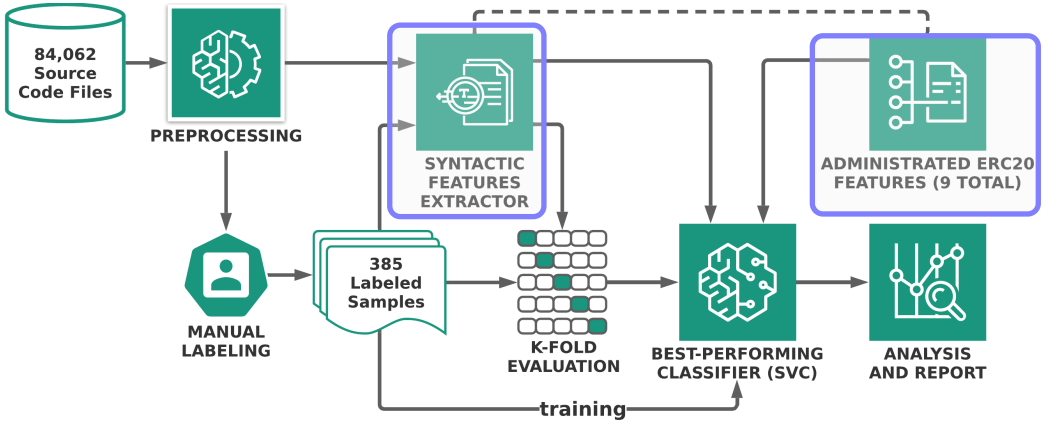
Administrated Tokens on Mainnet



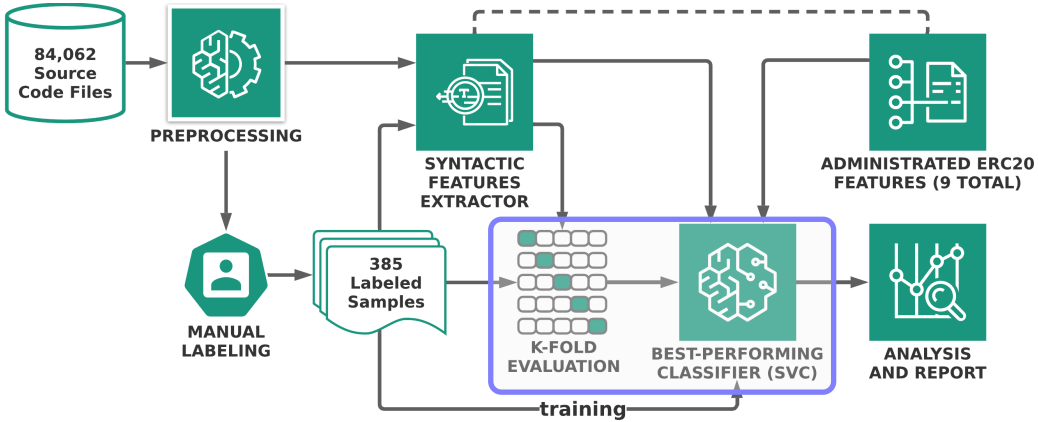
Administrated Tokens on Mainnet



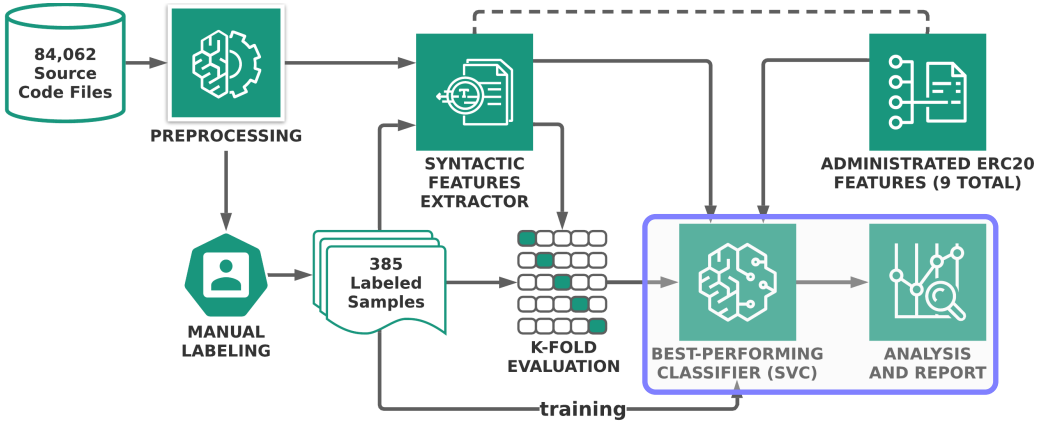
Administrated Tokens on Mainnet



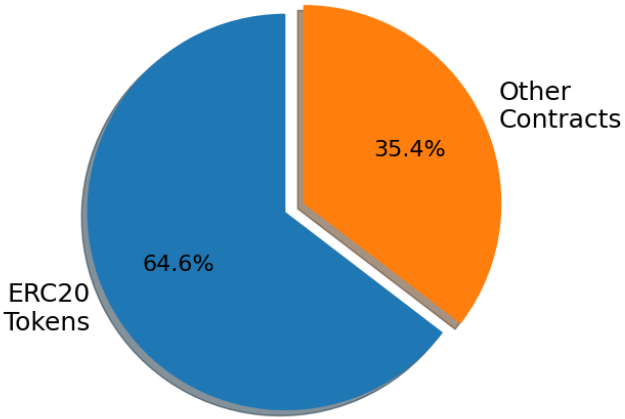
Administrated Tokens on Mainnet



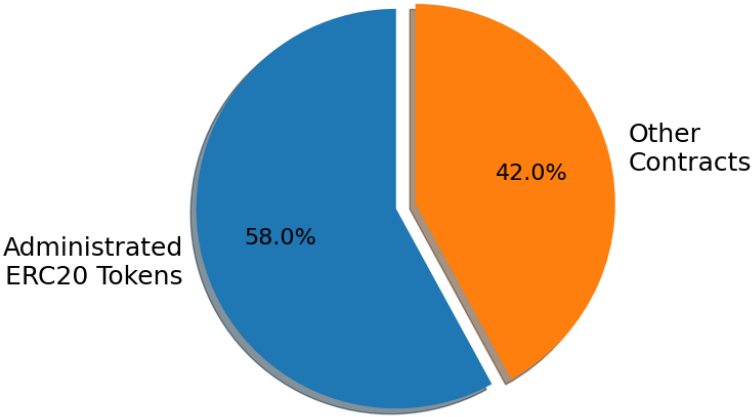
Administrated Tokens on Mainnet



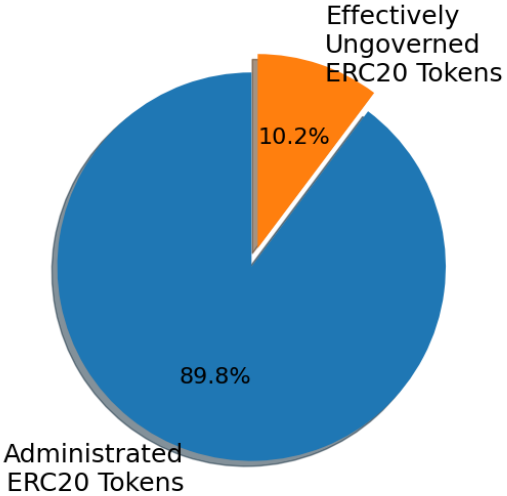
ERC20 Tokens vs Other Contracts



Percentage of Administrated ERC20 Tokens



Percentage of Administrated ERC20 Tokens — Cont'd



Outline

Insight

Should we boycott or ban administrated tokens?

Insight

Should we boycott or ban administrated tokens?

- Administration $\neq$ evil.

Insight

Should we boycott or ban administrated tokens?

- Administration $\neq$ evil.
- Swimming against the tide is counterproductive.

Insight

Should we boycott or ban administrated tokens?

- Administration $\neq$ evil.
- Swimming against the tide is counterproductive.
- **Administrating patterns can be safe!**

Insight

Should we boycott or ban administrated tokens?

- Administration $\neq$ evil.
- Swimming against the tide is counterproductive.
- **Administrating patterns can be safe!**

Insight

Should we boycott or ban administrated tokens?

- Administration $\neq$ evil.
- Swimming against the tide is counterproductive.
- **Administrating patterns can be safe!**

SafelyAdministrated

Deferred Maintenance

Insight

Should we boycott or ban administrated tokens?

- Administration $\neq$ evil.
- Swimming against the tide is counterproductive.
- **Administrating patterns can be safe!**

SafelyAdministrated

Deferred Maintenance

Quorum of Trustees

Insight

Should we boycott or ban administrated tokens?

- Administration $\neq$ evil.
- Swimming against the tide is counterproductive.
- **Administrating patterns can be safe!**

SafelyAdministrated

Deferred Maintenance

Quorum of Trustees

Safe Pause

Deferred Maintenance

- Announce maintenance procedure in advance;

Deferred Maintenance

- Announce maintenance procedure in advance;
- Postpone the execution of the maintenance procedure;

Deferred Maintenance

- Announce maintenance procedure in advance;
- Postpone the execution of the maintenance procedure;
- If users disagree, they can safely quit the contract before the arrival of the maintenance timestamp.

Quorum of Trustees

- Similar to an organization's board of trustees;

Quorum of Trustees

- Similar to an organization's board of trustees;
- Split the authority between multiple people or organizations;

Quorum of Trustees

- Similar to an organization's board of trustees;
- Split the authority between multiple people or organizations;
- The quorum of trustees votes for each maintenance procedure;

Quorum of Trustees

- Similar to an organization's board of trustees;
- Split the authority between multiple people or organizations;
- The quorum of trustees votes for each maintenance procedure;
- If one or a few members' private keys are compromised, the contract remains safe.

Safe Pause

- The pause functionality is important in ERC20 tokens;

Safe Pause

- The pause functionality is important in ERC20 tokens;
- State-of-the-art `Pausable` OpenZeppelin interface allows to freeze the smart contract forever;

Safe Pause

- The pause functionality is important in ERC20 tokens;
- State-of-the-art `Pausable` OpenZeppelin interface allows to freeze the smart contract forever;
- Any trustee can trigger a Safe Pause effective immediately.

Safe Pause

- The pause functionality is important in ERC20 tokens;
- State-of-the-art `Pausable` OpenZeppelin interface allows to freeze the smart contract forever;
- Any trustee can trigger a Safe Pause effective immediately.
- Safe Pause puts a deadline on the pause to allow the trustees to address the issue;

Safe Pause

- The pause functionality is important in ERC20 tokens;
- State-of-the-art `Pausable` OpenZeppelin interface allows to freeze the smart contract forever;
- Any trustee can trigger a Safe Pause effective immediately.
- Safe Pause puts a deadline on the pause to allow the trustees to address the issue;
- Two back-to-back pauses are prohibited.

Outline

Conclusion

- Most administrated ERC20 Tokens are potentially unsafe.

Conclusion

- Most administrated ERC20 Tokens are potentially unsafe.
- The vast majority of ERC20 tokens are administrated, with billions of dollars are in danger.

Conclusion

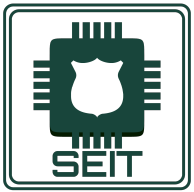
- Most administrated ERC20 Tokens are potentially unsafe.
- The vast majority of ERC20 tokens are administrated, with billions of dollars are in danger.
- *SafelyAdministrated:* make ERC20 tokens safe without boycotting administrating patterns;

Conclusion

- Most administrated ERC20 Tokens are potentially unsafe.
- The vast majority of ERC20 tokens are administrated, with billions of dollars are in danger.
- *SafelyAdministrated:* make ERC20 tokens safe without boycotting administrating patterns;
- **Future work:** Automated translation of unsafely administrated ERC20 tokens into the safely administrated ones.

Conclusion

- Most administrated ERC20 Tokens are potentially unsafe.
- The vast majority of ERC20 tokens are administrated, with billions of dollars are in danger.
- *SafelyAdministrated:* make ERC20 tokens safe without boycotting administrating patterns;
- **Future work:** Automated translation of unsafely administrated ERC20 tokens into the safely administrated ones.



https://seit.egr.msu.edu/