SHERLOCK

Security Review For Ethos Network

Public Contest Prepared For: **Ethos Network**
Lead Security Expert: **pkqs90**
Date Audited: **October 25 - November 2, 2024**

Introduction

Ethos Network is an on-chain social reputation platform. This contest focuses on the social elements first: establishing a profile, attesting social accounts, and leaving reviews comments, and votes.

Scope

Repository: trust-ethos/ethos

Branch: main

Audited Commit: 946e931acc97167a9bd932d02b9420cdf37a701e

Final Commit: ad7fa83f40062e0802889bb54ae235da042d293f

For the detailed scope, see the contest details.

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues Found

High	Medium
0	3

Issues Not Fixed or Acknowledged

High	Medium
0	0

Security experts who found valid issues

PNS
pkqs90
onthehunt
KlosMitSoss
LeFy
newspacexyz
dobrevaleri
DigiSafe
shaflow01
Kyosi
x0lohaclohell
heeze
justAWanderKid
durov
0xMosh
s0x0mtee
0xbrivan
056Security
debugging3
TessKimy
Bozho
dany.armstrong90
Dliteofficial
t.aksoy
0xBhumii
y4y
0x37
John44
IvanFitro
sammy

Issue M-1: Compromise check will prevent malicious operations

Source: https://github.com/sherlock-audit/2024-10-ethos-network-judging/issues/23

Found by

056Security, 0x37, 0xBhumii, 0xMosh, 0xbrivan, Bozho, DigiSafe, Dliteofficial, IvanFitro, John44, KlosMitSoss, LeFy, PNS, TessKimy, dany.armstrong90, debugging3, dobrevaleri, durov, heeze, justAWanderKid, newspacexyz, onthehunt, s0x0mtee, sammy, shaflow01, t.aksoy, y4y

Summary

A missing compromise check in `verifiedProfileIdForAddress` will cause unauthorized access for affected contracts as compromised addresses can bypass security measures and perform malicious actions. (e.g: Attacker can steal user's private key, so address is compromised)

Root Cause

In `EthosProfile.sol`, the `verifiedProfileIdForAddress` function is missing a check to ensure `_address` is not compromised, allowing compromised addresses to interact with other contracts without restriction. https://github.com/sherlock-audit/2024-10-ethos-network/blob/979e352d7bcdba3d0665f11c0320041ce28d1b89/ethos/packages/contracts/contracts/EthosProfile.sol#L568-L574

The `verifiedProfileIdForAddress` function used in many contracts.

Here: https://github.com/sherlock-audit/2024-10-ethos-network/blob/979e352d7bcdba3d0665f11c0320041ce28d1b89/ethos/packages/contracts/contracts/EthosAttestation.sol#L228 https://github.com/sherlock-audit/2024-10-ethos-network/blob/979e352d7bcdba3d0665f11c0320041ce28d1b89/ethos/packages/contracts/contracts/EthosDiscussion.sol#L111-L113 https://github.com/sherlock-audit/2024-10-ethos-network/blob/979e352d7bcdba3d0665f11c0320041ce28d1b89/ethos/packages/contracts/contracts/EthosDiscussion.sol#L158-L160 ...

In this project, there are many issues about this compromised address. In almost contracts, it doesn't check `msg.sender` is compromised address. Address is already unregistered from profile by `deleteAddressAtIndex` function, but it is still used in many functions.

Internal pre-conditions

User needs to call `deleteAddressAtIndex` to set `isAddressCompromised` to be true for the target address.

External pre-conditions

No response

Attack Path

1. Attack steal user's private key.
2. User detected it is compromised and calls `deleteAddressAtIndex` for marking `isAddressCompromised` as true for the target address.
3. Attacker can calls `addReview` in `EthorsReview` contract by compromised address. (private key is stolen so attacker can do this operation) It calls `ethosProfile.verifiedProfileIdForAddress(msg.sender);` msg.sender is compromised but it doesn't revert.

Impact

The **protocol** suffers a potential security breach as **compromised addresses** can bypass verification and execute unauthorized actions in dependent contracts, potentially leading to **manipulation of contract functionality.** The attacker gains access to otherwise restricted operations without proper authorization.

PoC

No response

Mitigation

Add modifier `checkIfCompromised` and use `checkIsAddressCompromised` function.

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/trust-ethos/ethos/pull/2204

Issue M-2: Restored addresses will not be able to take any action on behalf of the profile due to still being marked as compromised

Source: https://github.com/sherlock-audit/2024-10-ethos-network-judging/issues/152

Found by

KlosMitSoss, LeFy, PNS, newspacexyz, onthehunt, pkqs90

Summary

When an address is restored after being deleted due to being compromised, it remains marked as compromised in the `isAddressCompromised` mapping. This prevents the address from taking any actions on behalf of the profile, even though it should be able to.

Vulnerability Detail

When an address is compromised, it can be deleted by calling `EthosProfile::deleteAddressAtIndex()`, which marks the address as compromised in the `isAddressCompromised` mapping.

```
function deleteAddressAtIndex(uint256 addressIndex) external whenNotPaused {
  ... ...
  isAddressCompromised[addressStr] = true;
  ... ...
}
```

The purpose of the `isAddressCompromised` mapping is to ensure that no address marked as compromised and deleted can take any action on behalf of the profile. If a previously deleted address is no longer compromised, it can be restored when an address associated with the same profile as the deleted address calls `EthosProfile::registerAddress()` and provides that address.

However, the address is not unmarked in `isAddressCompromised`, meaning that restored addresses will still be unable to take any actions on behalf of the profile as they remain flagged as compromised.

Attack Path

1. AddressA calls `EthosProfile::registerAddress()` to register AddressB.
2. AddressB is compromised.
3. AddressA calls `EthosProfile::deleteAddressAtIndex()` to delete AddressB and mark it as compromised in `isAddressCompromised`.
4. AddressB is no longer compromised.
5. AddressA calls `EthosProfile::registerAddress()` to restore AddressB.
6. AddressB remains marked as compromised in `isAddressCompromised` and cannot act on behalf of the profile, even though it should be able to.

Impact

A previously deleted address will not be able to take any action on behalf of the profile when restored, as it remains marked as compromised.

Code Snippet

EthosProfile#L373-409 EthosProfile#L415-438

Tool Used

Manual Review

Recommendation

Consider unmarking addresses as compromised when they are restored. If it is necessary to store the addresses that were removed in the past, this should be managed with a separate mapping.

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/trust-ethos/ethos/pull/2204

Issue M-3: Corruptible Upgradability Pattern

Source: https://github.com/sherlock-audit/2024-10-ethos-network-judging/issues/206

Found by

DigiSafe, Kyosi, PNS, dobrevaleri, pkqs90, shaflow01, x0lohaclohell

Summary

The EthosContracts (EthosProfile, EthosReview, ...) are UUPSUpgradeable. However, the current implementation has multiple issues regarding upgradability.

Root Cause

Following is the inheritance chain of the EthosContracts.

The Ethos contracts are meant to be upgradeable. However, it inherits contracts that are not upgrade-safe.

The `AccessControl` and `SignatureControl` are both contracts written by Ethos team, both contain storage slots but there are no gaps implemented.

Also, AccessControl inherits the non-upgradeable version Pausable and AccessControlEnumerable from Openzeppelin's library, when it should use the upgradeable version from openzeppelin-contracts-upgradeable lib.

https://docs.openzeppelin.com/contracts/5.x/upgradeable

There is also another issue that in all EthosContract, the constructor does not have initializers disabled for the implementation contract. This is also a best practice for proxy contracts.

```
constructor() {
  _disableInitializers();
}
```

- https://github.com/sherlock-audit/2024-10-ethos-network/blob/main/ethos/packages/contracts/contracts/utils/AccessControl.sol#L15
- https://github.com/sherlock-audit/2024-10-ethos-network/blob/main/ethos/packages/contracts/contracts/utils/SignatureControl.sol#L11

Internal pre-conditions

If admin performs an upgrade and wants to add another storage slot in AccessControl or SignatureControl contract, the storage slot would mess up.

External pre-conditions

N/A

Attack Path

N/A

Impact

Storage of vault contracts might be corrupted during upgrading.

PoC

N/A

Mitigation

1. Add gaps in AccessControl, SignatureControl
2. Use library from Openzeppelin-upgradeable instead, e.g. PausableUpgradeable, AccessControlEnumerableUpgradeable.
3. Disable initializers in EthosContracts constructor.

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/trust-ethos/ethos/pull/2204

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.