SHERLOCK

Security Review For Hats Protocol

Public Contest Prepared For: **Hats Protocol**
Lead Security Expert: **bughuntoor**
Date Audited: **November 19 - November 23, 2024**

Introduction

HatsSignerGate leverages Hats Protocol to enable DAO-owned Safe multisigs. With HSG, DAOs can safely delegate operations of a Safe multisig to a set of valid Hat-wearing signers without giving up control to the signers.

Contest participants: 166 auditors

Scope

Repository: Hats-Protocol/hats-zodiac

Branch: v2

Audited Commit: 8576776f45d31e1bfde26a72235b2b46a9028b24

Final Commit: a9e3f4f0e968fb332800a468eddcb993fc6d5cd2

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
1	1

Issues Not Fixed or Acknowledged

High	Medium
0	0

Security experts who found valid issues

bughuntoor

Issue H-1: Detaching HSG when there's non-unregistered owners who no longer own the hat would give them control over the multi-sig

Source: https://github.com/sherlock-audit/2024-11-hats-protocol-judging/issues/4

The protocol has acknowledged this issue.

Found by

bughuntoor

Summary

After a user no longer owns a hat, although they are no longer treated as a valid signer, they remain an `owner` within the Safe, until `removeSigner` is called

```
function detachHSG() public {
  _checkUnlocked();
  _checkOwner();
  ISafe s = safe; // save SLOAD

  // first remove as guard, then as module
  s.execRemoveHSGAsGuard();
  s.execDisableHSGAsOnlyModule();
  emit Detached();
}
```

When HSG is detached, it does not check if there's any Safe owners who no longer wear the necessary hat. For this reason, if there's such owners, they'll remain rights within the multi-sig. Depending on their number, they might be able to overturn the multisig, or at least disallow them to reach quorum.

However, this attack can further be weaponized if one of the signer hats has admin rights over another signer hat. If the said admin hat wants to overturn the multisig and gain full access of it upon detaching, they can simply front-run the `detachHSG` call and do a loop of 1) transferring the lower hat to a new address 2) claiming it as a signer. Then, when the `detachHSG` executes, all of these addresses that the attacker had looped through would be owners of the safe and in most cases that should be enough to fully overturn the multi-sig and claim full custody of it.

Root Cause

`detachHSG` does not check if there are Safe owners who no longer wear the necessary hat.

Attack Path

1. DAO plans to detach from HSG
2. There exists a user who has admin hat over a signer hat which has a set max supply of 1.
3. DAO calls detach from HSG
4. The admin hat owner front-runs the tx and does a loop of transferring the hat and adding it as a signer. This gives a lot of wallets `owner` rights within the Safe, which would otherwise be worthless if HSG remains active
5. The DAO gets detached and all of the wallets the admin hat owner had looped through now are owners within the Safe
6. This would usually give full custody to the attacker, or at the very least guarantee the DAO is not able to execute anything on their own.

Impact

Attacker can gain full custody over a Safe upon HSG detachment

Affected Code

https://github.com/sherlock-audit/2024-11-hats-protocol/blob/main/hats-zodiac/src/HatsSignerGate.sol#L341

Mitigation

Upon detaching HSG, loop through all Safe owners and in case a wallet does not wear the necessary hat, unregister them as a signer.

Discussion

spengrah

This is a valid issue within the specific context of HSG because there is nothing that the HSG owner can do to prevent a malicious admin of (one of) the signer hats from front-running a `detachHSG()` call with this attack. However, Hats Protocol gives organizations sufficient tools to ensure that hat admins are not malicious (or at least sufficiently incentivized as such), so in the general context of HSG + Hats Protocol this is

not a major concern since organizations can protect against it without significant trouble. As a result, we will not be changing the HSG code to address this issue.

Issue M-1: Signer can avoid restrictions and change `safe` state variables

Source: https://github.com/sherlock-audit/2024-11-hats-protocol-judging/issues/3

The protocol has fixed this issue and the fix has been signed-off by the Lead Senior Watson.

Found by

bughuntoor

Summary

In order to make sure that a delegatecall does not change Safe's state, HSG's `checkTransaction` stores the current threshold, owners list and fallback handler. Then, after the call is executed, `checkAfterExecution` is supposed to verify that these variables have not been changed.

```
if (operation == Enum.Operation.DelegateCall) {
  // case: DELEGATECALL
  // We disallow delegatecalls to unapproved targets
  if (!enabledDelegatecallTargets[to]) revert DelegatecallTargetNotEnabled();

  // Otherwise record the existing owners and threshold for post-flight checks to
↪  ensure that Safe state has not
  // been altered
  _existingOwnersHash = keccak256(abi.encode(owners));
  _existingThreshold = threshold;
  _existingFallbackHandler = safe.getSafeFallbackHandler();
```

However, since the `checkTransaction` can be re-entered by a new call, these restrictions can easily be bypassed. If the delegatecall changes the owners and the threshold, the executing signer can then just provide a new transaction to be executed with the new owners being just him and threshold set to 1. This will then override the above stored variables. Because of this the `checkAfterExecution` check will also succeed.

Root Cause

Possible reentrancy within `checkTransaction`

Attack Path

1. Signers sign a tx which would alter the owners list and set the threshold to 1.

2. Within that `delegatecall`, the only remaining owner signs a new transaction and executes it. It doesn't realistically mater what the tx is.
3. `checkTransaction` is entered. `_existingOwnersHash` and `_existingThreshold` are overwritten to their new values.
4. The `checkAfterExecution` on both the inner and the outer call check against the altered values, hence they both succeed.
5. In the end, the intended restrictions are bypassed and the ownerlist and threshold are both overwritten.
6. The user who has remained the only owner has full access over the multi-sig until the other owners re-claim their hats.

Affected Code

https://github.com/sherlock-audit/2024-11-hats-protocol/blob/main/hats-zodiac/src/HatsSignerGate.sol#L471

Impact

Users can bypassed intended restrictions not to be able to overwrite the owner list and threshold variables.

Mitigation

If `checkTransaction` is entered, and the transient variables have already been assigned values, revert if the values differ from the current ones.

Discussion

spengrah

Fix PR: https://github.com/Hats-Protocol/hats-zodiac/pull/81

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/Hats-Protocol/hats-zodiac/pull/81

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.