EIP-3607

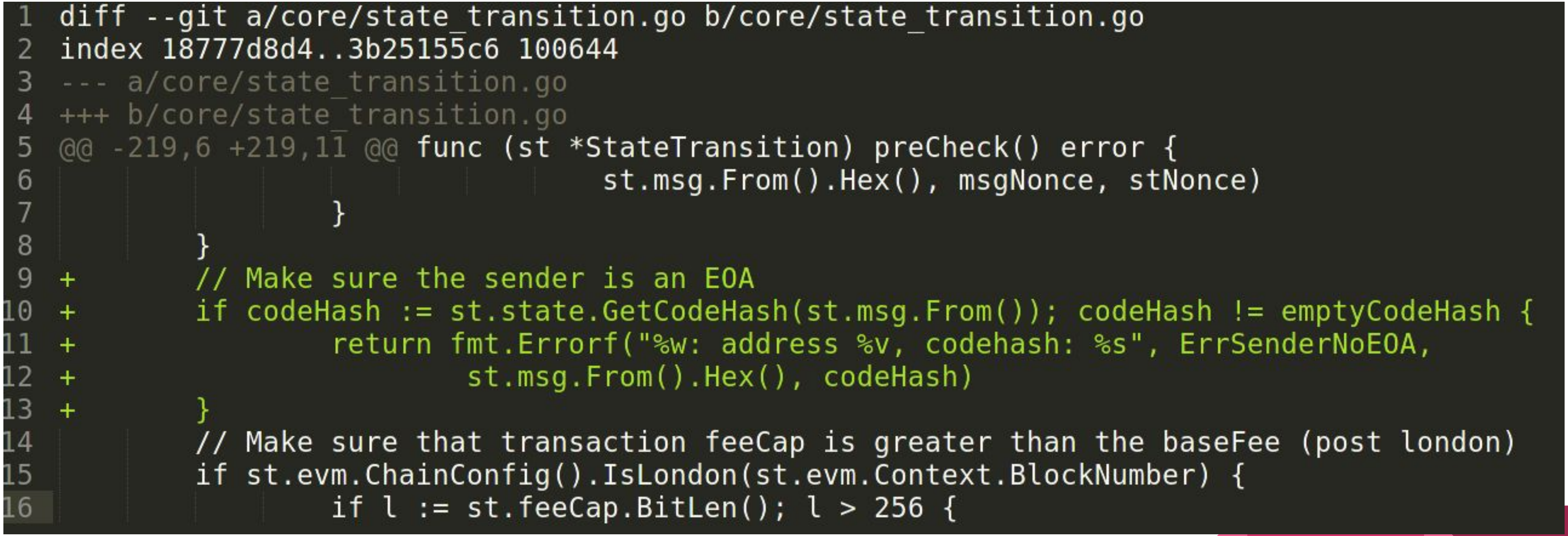
Reject transactions from senders with deployed code

Motivation

- Ethereum address are 20 bytes (160 bits)
- Finding a collision is unlikely (2^80 operations) but not impossible
- Collision between EOA and Contract could be found
- Would allow an attacker to drain a contract

- Can be prevented by disallowing transactions to be send from addresses with deployed contract code

Implementation in geth



```
diff --git a/core/state_transition.go b/core/state_transition.go
index 18777d8d4..3b25155c6 100644
--- a/core/state_transition.go
+++ b/core/state_transition.go
@@ -219,6 +219,11 @@ func (st *StateTransition) preCheck() error {
                                st.msg.From().Hex(), msgNonce, stNonce)
                }
        }
+       // Make sure the sender is an EOA
+       if codeHash := st.state.GetCodeHash(st.msg.From()); codeHash != emptyCodeHash {
+               return fmt.Errorf("%w: address %v, codehash: %s", ErrSenderNoEOA,
+                       st.msg.From().Hex(), codeHash)
+       }
        // Make sure that transaction feeCap is greater than the baseFee (post london)
        if st.evm.ChainConfig().IsLondon(st.evm.Context.BlockNumber) {
                if l := st.feeCap.BitLen(); l > 256 {
```

Testing

- Contracts can be deployed on EOA’s in the genesis
- Testing did rely heavily on EOA’s also containing code
- Many tests had to be rewritten
- EIP-3607 couldn’t be merged into Geth without testing fixed
- An inverse test (testing that sending from contracts fails) could not be merged in before the EIP was merged in Geth
- Update to the yellowpaper was merged to solidify the rule

Gotcha’s

- Old version of gnosis safe uses eth_call to construct the transaction as if it originated from the contract.
- Thus EIP-3607 is not in place if the transaction is not real (eth_call, estimate_gas,...)
- EIP-3607 is implemented on the CodeHash not on CodeSize, etc.
- In Geth GetCodeHash could return emptyCodeHash or common.Hash{}
- In both cases we accept the transaction