// HALBORN

CoreDAO - Chain and BTCPowerMirror

Golang Security Audit

Prepared by: **Halborn**

Date of Engagement: **August 22nd, 2022 - November 28th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	09/20/2022	Gokberk Gulgun
0.2	Document Updates	10/05/2022	Emiliano Carmona
0.3	Document Updates	10/06/2022	Gustavo Dutra
0.4	Document Review	10/10/2022	Gabi Urrutia
1.0	Remediation Plan	11/28/2022	Gokberk Gulgun
1.1	Remediation Plan Review	11/29/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Gustavo Dutra	Halborn	Gustavo.Dutra@halborn.com
Emiliano Carmona	Halborn	Emiliano.Carmona@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

CoreDAO engaged Halborn to conduct a security audit on their chain and btc power mirror repositories beginning on August 22nd, 2022 and ending on November 28th, 2022. The security audit was scoped to the repositories provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided nearly six weeks for the engagement and assigned three full-time security engineers to audit the security of the **chain and btc power mirror** implementation. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit to achieve the following:

- Ensure that module Implementation functions as intended.
- Identify potential security issues with the CoreDAO team.

In summary, Halborn identified few security risks that were mostly addressed by the CoreDAO Team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the chain and power mirror implementation. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of structures and can quickly identify items that do not follow security best practices.

The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Static Analysis of security for scoped repository, and imported functions. (staticcheck, gosec, unconvert, LGTM, ineffassign and semgrep).
- Manual Assessment for discovering security vulnerabilities on code-base.
- Ensuring correctness of the codebase.
- Dynamic Analysis on chain Implementation functions and data types.

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.

1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

The security assessment was scoped to the following repositories.

Core Chain

- Commit ID : 6219caa1de6182643e136e27c79355addaee1fc5

BTC Power Mirror

- Commit ID : 52b70ff629216ce8b40ad05652631da3efb95da3

BTC Power Mirror Second Phase Audit

- Commit ID : ec22e535a2e2d441d853795309c8f67c8e495509

Second Phase Audit Branch - Core Chain

- Commit ID : 9b9088a6a99121e983fd1a955a3d1209bed1b66a

FIX COMMIT ID :

cad6814457999e2de97ad653f91f3288b342f222
9b9088a6a99121e983fd1a955a3d1209bed1b66a

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	1	3	2

LIKELIHOOD

IMPACT

		(HAL-01)		
(HAL-03) (HAL-04) (HAL-05)		(HAL-02)		
(HAL-06) (HAL-07)				

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - SYSTEM CONTRACT UPGRADES ARE DISABLED	High	SOLVED - 11/28/2022
HAL-02 - CHAIN ID COLLISION	Medium	SOLVED - 10/31/2022
HAL-03 - USING VULNERABLE PACKAGE LEADS TO DOS	Low	SOLVED - 10/31/2022
HAL-04 - MATH RAND IS NOT SAFE FOR CONCURRENT USE	Low	RISK ACCEPTED
HAL-05 - MEMORY LEAK ON THE HANDLER	Low	SOLVED - 10/31/2022
HAL-06 - OUT-OF-DATE PACKAGES	Informational	SOLVED - 10/31/2022
HAL-07 - MISSING GO COMPILER BUILD DIRECTIVES	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) SYSTEM CONTRACT UPGRADES ARE DISABLED - HIGH

Description:

In the **CoreDAO** chain, system contract upgrades can be done via the **upgrade.go** file. This gives you the ability to upgrade system contracts. However, the feature is disabled in the **CoreDAO** chain.

Proof Of Concept:

1. Go to upgrade.go#L41.
2. Upgrades to the system contract are commented out in the code base.
3. With the following parameters, the system contract can be enabled.

Listing 1

```
Configs: []*UpgradeConfig{
            {
                ContractAddr: common.HexToAddress(TokenHubContract
↳ ),
                CommitUrl:    "CoreDAO genesis commit id",
                Code: "System Smart Contract Code"
            }
```

4. By calling the following function, the system contracts can be triggered.

Listing 2

```
systemcontracts.UpgradeBuildInSystemContract(config, b.header.
↳ Number, statedb)
```

5. In the BSC, the system contract upgrades can be done using the fork number. The implementation can be seen from the code.

6. On the other hand, the System Burn contract should not be updated during the upgrades. All funds burned will be located at the Burn address. In the BSC, the BurnContract address is deleted from the constant contract addresses.

Code Location:

The following code files indicate that system contract upgrades are disabled.

upgrade.go#L41

Listing 3

```
1 func UpgradeBuildInSystemContract(config *params.ChainConfig,
↳ blockNumber *big.Int, statedb *state.StateDB) {
2     /*
3         apply system upgrades
4     */
5 }
```

Risk Level:

Likelihood - 3
Impact - 5

Recommendation:

Make sure that the functionality is correctly implemented in the code base. Without this functionality, system contracts could not be updated in the code base.

Remediation Plan:

SOLVED: The CoreDAO Team solved the issue in the following commit v1.1.0_-fix - 9b9088a by enabling system upgrades.

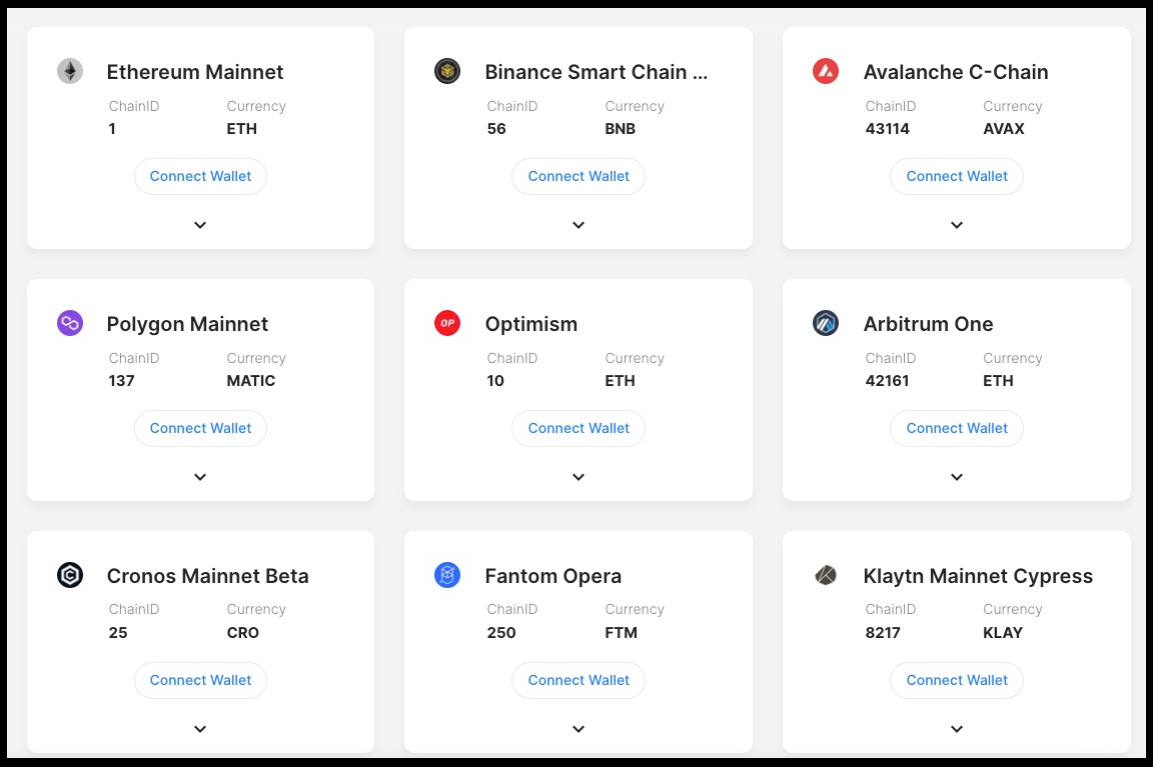
3.2 (HAL-02) CHAIN ID COLLISION - MEDIUM

Description:

Ethereum's networks have two identifiers, a network ID and a chain ID. Although they usually have the same value, they have different uses. Peer-to-peer communication between nodes uses the network ID, while the transaction signature process uses the chain ID. EIP 155 introduced the use of the chain ID as part of the transaction signing process to protect against replay attacks of transactions. In CoreDAO Chain, ChainId is defined as 56.

Proof Of Concept:

1. Navigate to BSC code base.
2. In the BSC code base, ChainID is defined as 56.
3. From the following code location, you can see that there is a conflict with the BSC config.



4. With the conflict, **CoreDAO** can suffer from the transaction replay attack.

Code Location:

config.go#L224

Listing 4

```
    CoreChainConfig = &ChainConfig{
        ChainID:             big.NewInt(56),
        HomesteadBlock:      big.NewInt(0),
        EIP150Block:         big.NewInt(0),
        EIP155Block:         big.NewInt(0),
        EIP158Block:         big.NewInt(0),
        ByzantiumBlock:      big.NewInt(0),
        ConstantinopleBlock: big.NewInt(0),
        PetersburgBlock:     big.NewInt(0),
        IstanbulBlock:       big.NewInt(0),
        MuirGlacierBlock:    big.NewInt(0),
        Satoshi: &SatoshiConfig{
            Period: 3,
```

```
14              Epoch:  200,
15          },
16      }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Consider changing the ChainID in the parameter definition.

Remediation Plan:

SOLVED: The CoreDAO team solved the issue by changing **testnet/mainnet** chain IDs.

3.3 (HAL-03) USING VULNERABLE PACKAGE LEADS TO DOS - LOW

Description:

During the code review, it was noted that the system uses a vulnerable version of go-ethereum. From the Security Advisory, it can be seen that the malicious crafted P2P message can lead to a Denial of Service (DoS).

Code Location:

peer.go#L326

Listing 5

```
    case msg.Code == discMsg:
        var reason [1]DiscReason
        // This is the last message. We don't need to discard or
        // check errors because, the connection will be closed
↳ after it.
        rlp.Decode(msg.Payload, &reason)
        return reason[0]
    case msg.Code < baseProtocolLength:
        // ignore other base protocol messages
        return msg.Discard()
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to update the system component.

Remediation Plan:

SOLVED: The CoreDAO team solved the issue by updating the dependency.

3.4 (HAL-04) MATH RAND IS NOT SAFE FOR CONCURRENT USE - LOW

Description:

Using math/rand is deterministic random by default, and it is dangerous. The BackOff time depends directly on math.Rand. In shuffling, the math/rand based random number that is used for the shuffle operation. Consider using crypto/rand instead of math/rand.

Code Location:

satoshi.go#L12

Listing 6

```
func backOffTime(snap *Snapshot, val common.Address) uint64 {
    if snap.inturn(val) {
        return 0
    } else {
        idx := snap.indexOfVal(val)
        if idx < 0 {
            // The backOffTime does not matter when a validator is not authorized.
            return 0
        }
        s := rand.NewSource(int64(snap.Number))
        r := rand.New(s)
        n := len(snap.Validators)
        backOffSteps := make([]uint64, 0, n)
        for idx := uint64(0); idx < uint64(n); idx++ {
            backOffSteps = append(backOffSteps, idx)
        }
        r.Shuffle(n, func(i, j int) {
            backOffSteps[i], backOffSteps[j] = backOffSteps[j], backOffSteps[i]
        })
        delay := initialBackOffTime + backOffSteps[idx]*wiggleTime
        return delay
    }
```

```
23 }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to use crypto/rand instead of math/rand.

Remediation Plan:

RISK ACCEPTED: The CoreDAO team accepted the risk of this finding.

3.5 (HAL-05) MEMORY LEAK ON THE HANDLER - LOW

Description:

In the current BSC repository, it has been noticed that on the Drift Protocol side, the memory leak issue has been fixed. However, the issue is not fixed in the CoreDAO chain.

Code Location:

/eth/handler.go#L392

Listing 7

```
1 func (h *handler) runDiffExtension(peer *diff.Peer, handler diff.
↳ Handler) error {
2     h.peerWG.Add(1)
3     defer h.peerWG.Done()
4
5     if err := h.peers.registerDiffExtension(peer); err != nil {
6         peer.Log().Error("Diff extension registration failed", "
↳ err", err)
7         return err
8     }
9     return handler(peer)
10 }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Make sure all recent findings (BSC) are fixed to the CoreDAO chain.

Remediation Plan:

SOLVED: The CoreDAO team solved the issue by changing the code base.

3.6 (HAL-06) OUT-OF-DATE PACKAGES - INFORMATIONAL

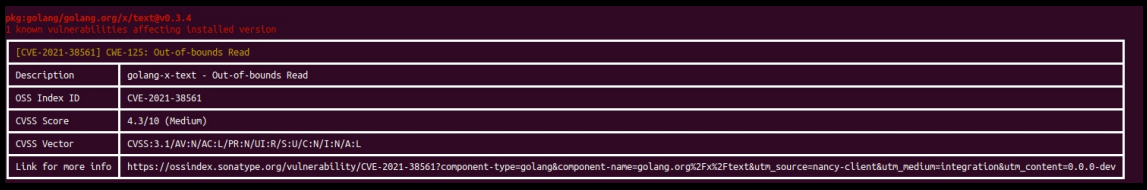
Description:

A software component is part of a system or application that extends the functionality of the application. Component-based vulnerabilities occur when a software component is incompatible, out of date, or vulnerable to a known exploit. You may inadvertently use vulnerable software components in production environments.

Code Location:

go.mod#L63

Vulnerable Text Package:



pkg:golang/golang.org/x/text@v0.3.4
1 known vulnerabilities affecting installed version

[CVE-2021-38561] CWE-125: Out-of-bounds Read	
Description	golang-x-text - Out-of-bounds Read
OSS Index ID	CVE-2021-38561
CVSS Score	4.3/10 (Medium)
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:L
Link for more info	https://ossindex.sonatype.org/vulnerability/CVE-2021-38561?component-type=golang&component-name=golang.org%2Fx%2Ftext&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-dev

Vulnerable InfluxDB Package:

pkg:golang/github.com/influxdata/influxdb@v1.8.3
2 known vulnerabilities affecting installed version

[CVE-2022-36640] CWE-276: Incorrect Default Permissions	
Description	** DISPUTED ** influxData influxDB before v1.8.10 contains no authentication mechanism or controls, allowing unauthenticated attackers to execute arbitrary commands. NOTE: the CVE ID assignment is disputed because the vendor's documentation states "If InfluxDB is being deployed on a publicly accessible endpoint, we strongly recommend authentication be enabled. Otherwise the data will be publicly available to any unauthenticated user. The default settings do NOT enable authentication and authorization."
OSS Index ID	CVE-2022-36640
CVSS Score	9.8/10 (Critical)
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Link for more info	https://ossindex.sonatype.org/vulnerability/CVE-2022-36640?component-type=golang&component-name=github.com%2Finfluxdata%2Finfluxdb&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-dev

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Update the software to the latest version.

Remediation Plan:

SOLVED: The CoreDAO team solved the issue by updating the dependency.

3.7 (HAL-07) MISSING GO COMPILER BUILD DIRECTIVES - INFORMATIONAL

Description:

During testing, it has been observed that the build flags of the Go compiler are not set. The use of compiler flags and compiler sequences can optimize and improve the performance of specific types of applications.

Risk Level:

Likelihood - 1
Impact - 1

Repository Makefile Flags:

Makefile#L16

Listing 8

```
1 geth:
2     $(GORUN) build/ci.go install ./cmd/geth
3     @echo "Done building."
4     @echo "Run \"$(GOBIN)/geth\" to launch geth."
5
6 all:
7     $(GORUN) build/ci.go install
```

Example Flags:

Listing 9: Example Compiler Build Flags

```
-a
    force rebuilding of packages that are already up-to-date.
-ldflags "-s -w"
    The -w turns off DWARF debugging information
    The -s turns off generation of the Go symbol table
-trimpath
    The -trimpath Remove all file system paths from the resulting executable.
-gcflags
    arguments to pass on each go tool compile invocation.
```

Recommendation:

Enabling the compiler build flags could speed up binary build and outputs in a smaller and probably more efficient binary. Therefore, the flags should be reviewed and enabled according to the structure.

Remediation Plan:

SOLVED: The CoreDAO team acknowledged the issue.

FUZZING

Description:

As part of the automated testing techniques, Halborn performed fuzzing using go-fuzz and libfuzzer. This is a technique that involves developing a custom script that injects malformed or unexpected input into the application to reveal possible vulnerabilities in the handling of that input.

For chain testing, most efforts were focused on the Satoshi plus consensus code. For that reason, the targeted fuzzing functions selected from satoshi.go were:
IsRoundEnd, Finalize, FinalizeAndAssemble, VerifyHeader, VerifyHeaders, Delay, BeforeValidateTx, BeforePackTx and Seal

As for the BtcLightMirror structure, our focus was on the Deserialize and CheckMerkle functions, which were more susceptible to potentially handling user input.

Fuzzing Harness:

For fuzzing, the target functions are needed to create a fuzzing harness. This is the script that the fuzzer program will use to inject the generated input into the target functions.
Since targeted functions take complex data structures as input, a combination of structure-aware fuzzing using the gofuzz library and valid objects was used to increase code coverage.

Fuzzing - Harness Code for Satoshi:

Satoshi's harness code is divided into 2 sections.
The first contains the definition of variables and the creation of the necessary objects that will be part of each function input. The second contains the fuzzing functions (one for each targeted function) in which data input is injected into some valid objects and other go objects are populated with the gofuzz library.

Listing 10: Fuzzing Harness for Satoshi

```
package corechain_fuzz

import (
    "math/big"

    "github.com/ethereum/go-ethereum/accounts"
    "github.com/ethereum/go-ethereum/common"
    "github.com/ethereum/go-ethereum/consensus/satoshi"
    "github.com/ethereum/go-ethereum/core"
    "github.com/ethereum/go-ethereum/core/rawdb"
    "github.com/ethereum/go-ethereum/core/state"
    "github.com/ethereum/go-ethereum/core/state/snapshot"
    "github.com/ethereum/go-ethereum/core/types"
    "github.com/ethereum/go-ethereum/crypto"
    "github.com/ethereum/go-ethereum/params"
    fuzz "github.com/google/gofuzz"
)

var (
    //--Fuzz Vars--
    hash      = [32]byte{}
    hash2     = []byte{}
    code      string
    gas       uint64
    timestamp int64
    dif       *big.Int
    //--

    extraVanity = 32
    extraSeal   = 65
    db          = rawdb.NewMemoryDatabase()
    key, _      = crypto.HexToECDSA("
↳ b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f291")
    addr        = crypto.PubkeyToAddress(key.PublicKey)
    chainConfig = &params.ChainConfig{big.NewInt(1337), big.NewInt
↳ (0), nil, false, big.NewInt(0), common.Hash{}, big.NewInt(0), big.
↳ NewInt(0), big.NewInt(0), big.NewInt(0), big.NewInt(0), big.NewInt
↳ (0), big.NewInt(0), big.NewInt(0), nil, nil, nil, nil, nil, &
↳ params.SatoshiConfig{Period: 0, Epoch: 30000, Round: 3}}

    txs      []*types.Transaction
    uncles   []*types.Header
    receipts *[]*types.Receipt
```

```
39      usedgas  uint64 = uint64(1234)
40      state2   *state.StateDB
41      snaps    *snapshot.Tree
42      db2      state.Database
43
44      signFn = func(account accounts.Account, s string, data []byte)
↳  ([]byte, error) {
45          return crypto.Sign(crypto.Keccak256(data), key)
46      }
47      signTxFn = func(accounts.Account, *types.Transaction, *big.Int
↳ ) (*types.Transaction, error) {
48          signer := types.NewEIP155Signer(big.NewInt(18))
49          utx := types.NewTransaction(0, addr, new(big.Int), 0, new(
↳ big.Int).SetUint64(10000000), nil)
50          tx, err := types.SignTx(utx, signer, key)
51          return tx, err
52      }
53      genspec = &core.Genesis{
54          ExtraData: make([]byte, extraVanity+common.AddressLength+
↳ extraSeal),
55          Alloc: map[common.Address]core.GenesisAccount{
56              addr: {Balance: big.NewInt(1)},
57          }, Config: chainConfig,
58      }
59  )
60
61  // IsRoundEnd - OK
62  func Fuzz1(data []byte) int {
63
64      fuzz.NewFromGoFuzz(data).Fuzz(&hash)
65      fuzz.NewFromGoFuzz(data).Fuzz(&code)
66      fuzz.NewFromGoFuzz(data).Fuzz(&gas)
67      fuzz.NewFromGoFuzz(data).Fuzz(&timestamp)
68      fuzz.NewFromGoFuzz(data).Fuzz(&dif)
69      var engine = satoshi.New(chainConfig, db, nil, hash)
70      defer engine.Close()
71
72      engine.Authorize(addr, signFn, signTxFn)
73      copy(genspec.ExtraData, data)
74      _ = genspec.MustCommit(db)
75      tx := types.NewContractCreation(0, big.NewInt(0), gas, big.
↳ NewInt(1), common.FromHex(code))
76      txs = append(txs, tx)
77
```

```
78      header := &types.Header{
79          ParentHash: hash,
80          Number:     big.NewInt(1),
81          Difficulty: big.NewInt(1),
82          Coinbase:   addr,
83          GasLimit:   gas,
84          Extra:      data,
85          Time:       uint64(timestamp),
86      }
87
88      headerchain, _ := core.NewHeaderChain(db, chainConfig, engine,
 ↳  func() bool { return false })
89      engine.IsRoundEnd(headerchain, header)
90      return 1
91  }
92
93  // Finalize - OK
94  func Fuzz2(data []byte) int {
95      fuzz.NewFromGoFuzz(data).Fuzz(&hash)
96      fuzz.NewFromGoFuzz(data).Fuzz(&code)
97      fuzz.NewFromGoFuzz(data).Fuzz(&gas)
98      fuzz.NewFromGoFuzz(data).Fuzz(&timestamp)
99      fuzz.NewFromGoFuzz(data).Fuzz(&dif)
100     fuzz.NewFromGoFuzz(data).Fuzz(&usedgas)
101     var engine = satoshi.New(chainConfig, db, nil, hash)
102     defer engine.Close()
103     engine.Authorize(addr, signFn, signTxFn)
104     copy(genspec.ExtraData, data)
105     _ = genspec.MustCommit(db)
106     tx := types.NewContractCreation(0, big.NewInt(0), gas, big.
 ↳ NewInt(1), common.FromHex(code))
107     txs = append(txs, tx)
108     header := &types.Header{
109         ParentHash: hash,
110         Number:     big.NewInt(1),
111         Difficulty: big.NewInt(1),
112         Coinbase:   addr,
113         GasLimit:   gas,
114         Extra:      data,
115         Time:       uint64(timestamp),
116     }
117     headerchain, _ := core.NewHeaderChain(db, chainConfig, engine,
 ↳  func() bool { return false })
```

```
118     //state generation using High memory consumption but higher
 ↳ code coverage
119     //blockchain, _ := core.NewBlockChain(db, nil, genspec.Config,
 ↳  engine, vm.Config{}, nil, nil)
120     //state, _ := blockchain.State()
121
122     //state generation using lower memory consumption but lower
 ↳ code coverage
123     state, err := state.New(common.Hash{}, state.NewDatabase(rawdb
 ↳ .NewMemoryDatabase()), nil)
124     err = engine.Finalize(headerchain, header, state, &txs, uncles
 ↳ , nil, nil, &usedgas)
125
126     if err != nil {
127         return 0
128     }
129     return 1
130 }
131
132 // FinalizeAndAssemble - OK
133 func Fuzz3(data []byte) int {
134     fuzz.NewFromGoFuzz(data).Fuzz(&hash)
135     fuzz.NewFromGoFuzz(data).Fuzz(&code)
136     fuzz.NewFromGoFuzz(data).Fuzz(&gas)
137     fuzz.NewFromGoFuzz(data).Fuzz(&timestamp)
138     fuzz.NewFromGoFuzz(data).Fuzz(&dif)
139     fuzz.NewFromGoFuzz(data).Fuzz(&usedgas)
140     var engine = satoshi.New(chainConfig, db, nil, hash)
141     defer engine.Close()
142     engine.Authorize(addr, signFn, signTxFn)
143     copy(genspec.ExtraData, data)
144     _ = genspec.MustCommit(db)
145     tx := types.NewContractCreation(0, big.NewInt(0), gas, big.
 ↳ NewInt(1), common.FromHex(code))
146     txs = append(txs, tx)
147     header := &types.Header{
148         ParentHash: hash,
149         Number:     big.NewInt(1),
150         Difficulty: big.NewInt(1),
151         Coinbase:   addr,
152         GasLimit:   gas,
153         Extra:      data,
154         Time:       uint64(timestamp),
155     }
```

```
156     headerchain, _ := core.NewHeaderChain(db, chainConfig, engine,
 ↳  func() bool { return false })
157     //state generation using High memory consumption but higher
 ↳ code coverage
158     //blockchain, _ := core.NewBlockChain(db, nil, genspec.Config,
 ↳  engine, vm.Config{}, nil, nil)
159     //state, _ := blockchain.State()
160
161     //state generation using lower memory consumption but lower
 ↳ code coverage
162     state, err := state.New(common.Hash{}, state.NewDatabase(rawdb
 ↳ .NewMemoryDatabase()), nil)
163     _, _, err = engine.FinalizeAndAssemble(headerchain, header,
 ↳ state, txs, uncles, nil)
164
165     if err != nil {
166         return 0
167     }
168     return 1
169 }
170
171 // VerifyHeader - OK
172 func Fuzz4(data []byte) int {
173     fuzz.NewFromGoFuzz(data).Fuzz(&hash)
174     fuzz.NewFromGoFuzz(data).Fuzz(&code)
175     fuzz.NewFromGoFuzz(data).Fuzz(&gas)
176     fuzz.NewFromGoFuzz(data).Fuzz(&timestamp)
177     fuzz.NewFromGoFuzz(data).Fuzz(&dif)
178     var engine = satoshi.New(chainConfig, db, nil, hash)
179     defer engine.Close()
180
181     engine.Authorize(addr, signFn, signTxFn)
182     copy(genspec.ExtraData, data)
183     _ = genspec.MustCommit(db)
184     tx := types.NewContractCreation(0, big.NewInt(0), gas, big.
 ↳ NewInt(1), common.FromHex(code))
185     txs = append(txs, tx)
186     header := &types.Header{
187         ParentHash: hash,
188         Number:     big.NewInt(1),
189         Difficulty: big.NewInt(1),
190         Coinbase:   addr,
191         GasLimit:   gas,
192         Extra:      data,
```

```
193         Time:       uint64(timestamp),
194     }
195     headerchain, _ := core.NewHeaderChain(db, chainConfig, engine,
 ↳  func() bool { return false })
196     err := engine.VerifyHeader(headerchain, header, true)
197
198     if err != nil {
199         return 0
200     }
201
202     return 1
203 }
204
205 // VerifyHeaders - OK
206 func Fuzz5(data []byte) int {
207     fuzz.NewFromGoFuzz(data).Fuzz(&hash)
208     fuzz.NewFromGoFuzz(data).Fuzz(&code)
209     fuzz.NewFromGoFuzz(data).Fuzz(&gas)
210     fuzz.NewFromGoFuzz(data).Fuzz(&timestamp)
211     fuzz.NewFromGoFuzz(data).Fuzz(&dif)
212     var engine = satoshi.New(chainConfig, db, nil, hash)
213     defer engine.Close()
214
215     engine.Authorize(addr, signFn, signTxFn)
216     copy(genspec.ExtraData, data)
217     _ = genspec.MustCommit(db)
218     tx := types.NewContractCreation(0, big.NewInt(0), gas, big.
 ↳ NewInt(1), common.FromHex(code))
219     txs = append(txs, tx)
220     header := &types.Header{
221         ParentHash: hash,
222         Number:     big.NewInt(1),
223         Difficulty: big.NewInt(1),
224         Coinbase:   addr,
225         GasLimit:   gas,
226         Extra:      data,
227         Time:       uint64(timestamp),
228     }
229     headerchain, _ := core.NewHeaderChain(db, chainConfig, engine,
 ↳  func() bool { return false })
230     headers := []*types.Header{header}
231     _, _ = engine.VerifyHeaders(headerchain, headers, []bool{true
 ↳ })
232
```

```
233     return 1
234 }
235
236 // Delay - OK
237 func Fuzz6(data []byte) int {
238     fuzz.NewFromGoFuzz(data).Fuzz(&hash)
239     fuzz.NewFromGoFuzz(data).Fuzz(&code)
240     fuzz.NewFromGoFuzz(data).Fuzz(&gas)
241     fuzz.NewFromGoFuzz(data).Fuzz(&timestamp)
242     fuzz.NewFromGoFuzz(data).Fuzz(&dif)
243
244     var engine = satoshi.New(chainConfig, db, nil, hash)
245     defer engine.Close()
246     engine.Authorize(addr, signFn, signTxFn)
247     copy(genspec.ExtraData, data)
248     _ = genspec.MustCommit(db)
249     if genspec == nil {
250
251     }
252     tx := types.NewContractCreation(0, big.NewInt(0), gas, big.
 ↳ NewInt(1), common.FromHex(code))
253     txs = append(txs, tx)
254     header := &types.Header{
255         ParentHash: hash,
256         Number:     big.NewInt(1),
257         Difficulty: big.NewInt(1),
258         Coinbase:   addr,
259         GasLimit:   gas,
260         Extra:      data,
261         Time:       uint64(timestamp),
262     }
263     headerchain, _ := core.NewHeaderChain(db, chainConfig, engine,
 ↳  func() bool { return false })
264     _ = engine.Delay(headerchain, header)
265
266     return 1
267 }
268
269 // BeforeValidateTx - OK
270 func Fuzz7(data []byte) int {
271     fuzz.NewFromGoFuzz(data).Fuzz(&hash)
272     fuzz.NewFromGoFuzz(data).Fuzz(&code)
273     fuzz.NewFromGoFuzz(data).Fuzz(&gas)
274     fuzz.NewFromGoFuzz(data).Fuzz(&timestamp)
```

```
275     var engine = satoshi.New(chainConfig, db, nil, hash)
276     defer engine.Close()
277 
278     engine.Authorize(addr, signFn, signTxFn)
279     copy(genspec.ExtraData, data)
280     _ = genspec.MustCommit(db)
281     tx := types.NewContractCreation(0, big.NewInt(0), gas, big.
 ↳ NewInt(1), common.FromHex(code))
282     txs = append(txs, tx)
283     header := &types.Header{
284         ParentHash: hash,
285         Number:     big.NewInt(1),
286         Difficulty: big.NewInt(1),
287         Coinbase:   addr,
288         GasLimit:   gas,
289         Extra:      data,
290         Time:       uint64(timestamp),
291     }
292     headerchain, _ := core.NewHeaderChain(db, chainConfig, engine,
 ↳  func() bool { return false })
293     //state generation using High memory consumption but higher
 ↳ code coverage
294     //blockchain, _ := core.NewBlockChain(db, nil, genspec.Config,
 ↳  engine, vm.Config{}, nil, nil)
295     //state, _ := blockchain.State()
296 
297     //state generation using lower memory consumption but lower
 ↳ code coverage
298     state, _ := state.New(common.Hash{}, state.NewDatabase(rawdb.
 ↳ NewMemoryDatabase()), nil)
299 
300     err := engine.BeforeValidateTx(headerchain, header, state, &
 ↳ txs, uncles, receipts, &txs, &usedgas)
301     if err != nil {
302         return 0
303     }
304 
305     return 1
306 }
307 
308 // BeforePackTx - OK
309 func Fuzz8(data []byte) int {
310     fuzz.NewFromGoFuzz(data).Fuzz(&hash)
311     fuzz.NewFromGoFuzz(data).Fuzz(&code)
```

```
312     fuzz.NewFromGoFuzz(data).Fuzz(&gas)
313     fuzz.NewFromGoFuzz(data).Fuzz(&timestamp)
314     fuzz.NewFromGoFuzz(data).Fuzz(&dif)
315     fuzz.NewFromGoFuzz(data).Fuzz(&usedgas)
316     fuzz.NewFromGoFuzz(data).Fuzz(&receipts)
317     var engine = satoshi.New(chainConfig, db, nil, hash)
318     defer engine.Close()
319
320     engine.Authorize(addr, signFn, signTxFn)
321     copy(genspec.ExtraData, data)
322     _ = genspec.MustCommit(db)
323     tx := types.NewContractCreation(0, big.NewInt(0), gas, big.
 ↳ NewInt(1), common.FromHex(code))
324     txs = append(txs, tx)
325     header := &types.Header{
326         ParentHash: hash,
327         Number:     big.NewInt(1),
328         Difficulty: big.NewInt(1),
329         Coinbase:   addr,
330         GasLimit:   gas,
331         Extra:      data,
332         Time:       uint64(timestamp),
333     }
334     headerchain, _ := core.NewHeaderChain(db, chainConfig, engine,
 ↳  func() bool { return false })
335     //state generation using High memory consumption but higher
 ↳ code coverage
336     //blockchain, _ := core.NewBlockChain(db, nil, genspec.Config,
 ↳  engine, vm.Config{}, nil, nil)
337     //state, _ := blockchain.State()
338
339     //state generation using lower memory consumption but lower
 ↳ code coverage
340     state, err := state.New(common.Hash{}, state.NewDatabase(rawdb
 ↳ .NewMemoryDatabase()), nil)
341     err = engine.BeforePackTx(headerchain, header, state, &txs,
 ↳ uncles, receipts)
342     if err != nil {
343         return 0
344     }
345
346     return 1
347 }
348
```

```
349 // Seal - OK
350 func Fuzz9(data []byte) int {
351     fuzz.NewFromGoFuzz(data).Fuzz(&hash)
352     fuzz.NewFromGoFuzz(data).Fuzz(&code)
353     fuzz.NewFromGoFuzz(data).Fuzz(&gas)
354     fuzz.NewFromGoFuzz(data).Fuzz(&timestamp)
355     fuzz.NewFromGoFuzz(data).Fuzz(&usedgas)
356 
357     var engine = satoshi.New(chainConfig, db, nil, hash)
358     defer engine.Close()
359 
360     engine.Authorize(addr, signFn, signTxFn)
361     copy(genspec.ExtraData, data)
362 
363     _ = genspec.MustCommit(db)
364     tx := types.NewContractCreation(0, big.NewInt(0), gas, big.
 ↳ NewInt(1), common.FromHex(code))
365     txs = append(txs, tx)
366     header := &types.Header{
367         ParentHash: hash,
368         Number:     big.NewInt(1),
369         Difficulty: big.NewInt(1),
370         Coinbase:   addr,
371         GasLimit:   gas,
372         Extra:      data,
373         Time:       uint64(timestamp),
374         Root:       hash,
375     }
376     results := make(chan *types.Block)
377     headerchain, _ := core.NewHeaderChain(db, chainConfig, engine,
 ↳  func() bool { return false })
378     err := engine.Seal(headerchain, types.NewBlockWithHeader(
 ↳ header), results, nil)
379     if err != nil {
380         return 0
381     }
382 
383     return 1
384 }
```

Fuzzing - Harness Code for BtcLightMirror:

For the BtcLightMirror fuzzing, a small program was built to write a valid serialized object to a corpus file. These data will be the initial point to be mutated by gofuzz throughout the fuzzing process.

Listing 11: Small program for Corpus preparation

```
 1 package main
 2
 3 import (
 4     "os"
 5 )
 6
 7 var blmEncoded = []byte{
 8         0x01,0x00,0x00,0x00,0x6f,0xe2,0x8c,0x0a,0xb6,0xf1,0xb3,0
↳ x72,0xc1,0xa6,0xa2,0x46,
 9         0xae,0x63,0xf7,0x4f,0x93,0x1e,0x83,0x65,0xe1,0x5a,0x08,0
↳ x9c,0x68,0xd6,0x19,0x00,
10         0x00,0x00,0x00,0x00,0x3b,0xa3,0xed,0xfd,0x7a,0x7b,0x12,0
↳ xb2,0x7a,0xc7,0x2c,0x3e,
11         0x67,0x76,0x8f,0x61,0x7f,0xc8,0x1b,0xc3,0x88,0x8a,0x51,0
↳ x32,0x3a,0x9f,0xb8,0xaa,
12         0x4b,0x1e,0x5e,0x4a,0x29,0xab,0x5f,0x49,0xff,0xff,0x00,0
↳ x1d,0xf3,0xe0,0x01,0x00,
13         0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0
↳ x00,0x00,0x00,0x00,0x00,
14         0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
↳ x00,0x00,0x00,0x00,0x00,
15         0x00,0x00,0x00,0x00,0x00,0xff,0xff,0xff,0xff,0x07,0x04,0
↳ x31,0xdc,0x00,0x1b,0x01,
16         0x62,0xff,0xff,0xff,0xff,0x02,0x00,0xf2,0x05,0x2a,0x01,0
↳ x00,0x00,0x00,0x43,0x41,
17         0x04,0xd6,0x4b,0xdf,0xd0,0x9e,0xb1,0xc5,0xfe,0x29,0x5a,0
↳ xbd,0xeb,0x1d,0xca,0x42,
18         0x81,0xbe,0x98,0x8e,0x2d,0xa0,0xb6,0xc1,0xc6,0xa5,0x9d,0
↳ xc2,0x26,0xc2,0x86,0x24,
19         0xe1,0x81,0x75,0xe8,0x51,0xc9,0x6b,0x97,0x3d,0x81,0xb0,0
↳ x1c,0xc3,0x1f,0x04,0x78,
20         0x34,0xbc,0x06,0xd6,0xd6,0xed,0xf6,0x20,0xd1,0x84,0x24,0
↳ x1a,0x6a,0xed,0x8b,0x63,
21         0xa6,0xac,0x00,0xe1,0xf5,0x05,0x00,0x00,0x00,0x00,0x43,0
↳ x41,0x04,0xd6,0x4b,0xdf,
```

```
22          0xd0,0x9e,0xb1,0xc5,0xfe,0x29,0x5a,0xbd,0xeb,0x1d,0xca,0
↳ x42,0x81,0xbe,0x98,0x8e,
23          0x2d,0xa0,0xb6,0xc1,0xc6,0xa5,0x9d,0xc2,0x26,0xc2,0x86,0
↳ x24,0xe1,0x81,0x75,0xe8,
24          0x51,0xc9,0x6b,0x97,0x3d,0x81,0xb0,0x1c,0xc3,0x1f,0x04,0
↳ x78,0x34,0xbc,0x06,0xd6,
25          0xd6,0xed,0xf6,0x20,0xd1,0x84,0x24,0x1a,0x6a,0xed,0x8b,0
↳ x63,0xa6,0xac,0x00,0x00,
26          0x00,0x00,0x01,0x9d,0x0e,0x29,0x88,0x3d,0x9b,0xdc,0x34,0
↳ x65,0x5a,0x80,0xe4,0xd1,
27          0x7d,0xb7,0xa1,0x79,0x63,0xa5,0xa0,0x4c,0x16,0x5c,0xe0,0
↳ x8d,0xf3,0x2f,0x52,0x5a,
28          0xd1,0x00,0x01,
29 }
30
31 func main() {
32     f, err := os.Create("corpus/in")
33
34     defer f.Close()
35
36     _, err = f.Write(blmEncoded)
37     if err != nil {
38         panic(err)
39     }
40
41     f.Sync()
42 }
```

This is the fuzz harness that was executed for the Deseralize and CheckMerkle functions, using the initial mutated corpus as data.

Listing 12: Harness Code for BtcLightMirror

```
1  package lightmirror
2
3  import (
4          "bytes"
5  )
6
7  func FuzzDeser(data []byte) int {
8          rbuf := bytes.NewReader(data)
9          var mirror BtcLightMirror
10         err := mirror.Deserialize(rbuf)
```

```
11          if err != nil {
12                  return 0
13          }
14
15          err = mirror.CheckMerkle()
16          if err != nil {
17                  return 0
18          }
19
20          return 1
21  }
```

Fuzzing - Libfuzzer Output Sample for Satoshi:

```
└─$ GOPATH=$(pwd) GO111MODULE=off go-fuzz-build -libfuzzer -o fuzz3.a -func Fuzz3 . && \
clang -o Fuzz.libfuzzer3 fuzz3.a -fsanitize=fuzzer && \
./Fuzz.libfuzzer3 coredao_wd/corpus -rss_limit_mb=15500 -workers=4 -max_len=1000

INFO: Running with entropic power schedule (0×FF, 100).
INFO: Seed: 2520844396
INFO: 65536 Extra Counters
INFO:     2597 files found in coredao_wd/corpus
INFO: seed corpus: files: 2597 min: 1b max: 871065b total: 9988708b rss: 51Mb
#512    pulse  ft: 5545 corp: 314/715b exec/s: 170 rss: 60Mb
#1024   pulse  ft: 5722 corp: 383/1521b exec/s: 146 rss: 65Mb
#2048   pulse  ft: 5920 corp: 469/8287b exec/s: 120 rss: 69Mb
#2598   INITED ft: 6123 corp: 526/36Kb exec/s: 108 rss: 70Mb
#2599   NEW    ft: 6128 corp: 527/36Kb lim: 1000 exec/s: 108 rss: 70Mb L: 17/1000 MS: 1 ChangeByte-
#2615   NEW    ft: 6133 corp: 528/36Kb lim: 1000 exec/s: 108 rss: 70Mb L: 5/1000 MS: 1 CrossOver-
#2631   NEW    ft: 6138 corp: 529/36Kb lim: 1000 exec/s: 109 rss: 70Mb L: 118/1000 MS: 1 InsertRepeatedBytes-
#2647   NEW    ft: 6143 corp: 530/36Kb lim: 1000 exec/s: 110 rss: 70Mb L: 123/1000 MS: 1 ChangeByte-
#2663   NEW    ft: 6154 corp: 531/36Kb lim: 1000 exec/s: 110 rss: 70Mb L: 2/1000 MS: 1 ChangeByte-
#2698   NEW    ft: 6157 corp: 532/36Kb lim: 1000 exec/s: 107 rss: 70Mb L: 90/1000 MS: 4 InsertRepeatedBytes-Ch
#2711   NEW    ft: 6162 corp: 533/37Kb lim: 1000 exec/s: 108 rss: 70Mb L: 395/1000 MS: 3 InsertByte-CopyPart-E
#2747   REDUCE ft: 6162 corp: 533/37Kb lim: 1000 exec/s: 105 rss: 70Mb L: 58/1000 MS: 1 EraseBytes-
#2769   NEW    ft: 6163 corp: 534/37Kb lim: 1000 exec/s: 106 rss: 70Mb L: 6/1000 MS: 2 ShuffleBytes-InsertRepe
#2807   NEW    ft: 6164 corp: 535/37Kb lim: 1000 exec/s: 107 rss: 71Mb L: 24/1000 MS: 3 ChangeBit-ChangeBinInt
#2822   NEW    ft: 6167 corp: 536/37Kb lim: 1000 exec/s: 108 rss: 71Mb L: 1/1000 MS: 5 ChangeByte-CopyPart-Cha
#2823   NEW    ft: 6172 corp: 537/37Kb lim: 1000 exec/s: 108 rss: 71Mb L: 595/1000 MS: 1 PersAutoDict- DE: "\x
#2877   REDUCE ft: 6172 corp: 537/37Kb lim: 1000 exec/s: 106 rss: 71Mb L: 45/1000 MS: 4 CopyPart-ShuffleBytes-
#2887   NEW    ft: 6183 corp: 538/37Kb lim: 1000 exec/s: 106 rss: 71Mb L: 23/1000 MS: 5 CrossOver-ChangeBinInt
```

Fuzzing - gofuzz Output Sample for BtcLightMirror:

```
→ lightmirror go-fuzz-build .
→ lightmirror go-fuzz
2022/10/06 22:59:55 workers: 8, corpus: 68 (3s ago), crashers: 1, restarts: 1/0, execs: 0 (0/sec), cover: 0, uptime: 3s
2022/10/06 22:59:58 workers: 8, corpus: 68 (6s ago), crashers: 1, restarts: 1/0, execs: 0 (0/sec), cover: 586, uptime: 6s
2022/10/06 23:00:01 workers: 8, corpus: 69 (1s ago), crashers: 1, restarts: 1/3247, execs: 64947 (7206/sec), cover: 586, uptime: 9s
2022/10/06 23:00:04 workers: 8, corpus: 69 (4s ago), crashers: 1, restarts: 1/6550, execs: 196522 (16359/sec), cover: 586, uptime: 12s
2022/10/06 23:00:07 workers: 8, corpus: 69 (7s ago), crashers: 1, restarts: 1/7297, execs: 306480 (20414/sec), cover: 586, uptime: 15s
2022/10/06 23:00:10 workers: 8, corpus: 69 (10s ago), crashers: 1, restarts: 1/7802, execs: 413551 (22958/sec), cover: 586, uptime: 18s
2022/10/06 23:00:13 workers: 8, corpus: 69 (13s ago), crashers: 1, restarts: 1/8069, execs: 524523 (24960/sec), cover: 586, uptime: 21s
2022/10/06 23:00:16 workers: 8, corpus: 69 (16s ago), crashers: 1, restarts: 1/8403, execs: 630256 (26246/sec), cover: 586, uptime: 24s
2022/10/06 23:00:19 workers: 8, corpus: 69 (19s ago), crashers: 1, restarts: 1/8553, execs: 718521 (26599/sec), cover: 586, uptime: 27s
2022/10/06 23:00:22 workers: 8, corpus: 69 (22s ago), crashers: 1, restarts: 1/8749, execs: 813742 (27113/sec), cover: 586, uptime: 30s
2022/10/06 23:00:25 workers: 8, corpus: 69 (25s ago), crashers: 1, restarts: 1/8975, execs: 906558 (27460/sec), cover: 586, uptime: 33s
2022/10/06 23:00:28 workers: 8, corpus: 69 (28s ago), crashers: 1, restarts: 1/8889, execs: 1004479 (27892/sec), cover: 586, uptime: 36s
2022/10/06 23:00:31 workers: 8, corpus: 69 (31s ago), crashers: 1, restarts: 1/9099, execs: 1101067 (28223/sec), cover: 586, uptime: 39s
2022/10/06 23:00:34 workers: 8, corpus: 69 (34s ago), crashers: 1, restarts: 1/9136, execs: 1196924 (28489/sec), cover: 586, uptime: 42s
2022/10/06 23:00:37 workers: 8, corpus: 69 (37s ago), crashers: 1, restarts: 1/9188, execs: 1286335 (28577/sec), cover: 586, uptime: 45s
2022/10/06 23:00:40 workers: 8, corpus: 69 (40s ago), crashers: 1, restarts: 1/9204, execs: 1380737 (28757/sec), cover: 586, uptime: 48s
2022/10/06 23:00:43 workers: 8, corpus: 69 (43s ago), crashers: 1, restarts: 1/9270, execs: 1483253 (29076/sec), cover: 586, uptime: 51s
2022/10/06 23:00:46 workers: 8, corpus: 69 (46s ago), crashers: 1, restarts: 1/9286, execs: 1578690 (29228/sec), cover: 586, uptime: 54s
2022/10/06 23:00:49 workers: 8, corpus: 69 (49s ago), crashers: 1, restarts: 1/9347, execs: 1673269 (29349/sec), cover: 586, uptime: 57s
2022/10/06 23:00:52 workers: 8, corpus: 69 (52s ago), crashers: 1, restarts: 1/9417, execs: 1761143 (29346/sec), cover: 586, uptime: 1m0s
2022/10/06 23:00:55 workers: 8, corpus: 69 (55s ago), crashers: 1, restarts: 1/9410, execs: 1844460 (29271/sec), cover: 586, uptime: 1m3s
2022/10/06 23:00:58 workers: 8, corpus: 69 (58s ago), crashers: 1, restarts: 1/9382, execs: 1932747 (29278/sec), cover: 586, uptime: 1m6s
2022/10/06 23:01:01 workers: 8, corpus: 69 (1m1s ago), crashers: 1, restarts: 1/9460, execs: 2015133 (29199/sec), cover: 586, uptime: 1m9s
2022/10/06 23:01:04 workers: 8, corpus: 69 (1m4s ago), crashers: 1, restarts: 1/9509, execs: 2101604 (29184/sec), cover: 586, uptime: 1m12s
2022/10/06 23:01:07 workers: 8, corpus: 69 (1m7s ago), crashers: 1, restarts: 1/9501, execs: 2185396 (29133/sec), cover: 586, uptime: 1m15s
```

Fuzzing Results:

After fuzzing all selected functions, no vulnerabilities were found.

AUTOMATED TESTING

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were staticcheck, gosec, semgrep, unconvert, LGTM and Nancy. After Halborn verified all the contracts and scoped structures in the repository and was able to compile them correctly, these tools were leveraged on scoped structures. With these tools, Halborn can statically verify security related issues across the entire codebase.

Semgrep - Security Analysis Output Sample:

Listing 13: Rule Set

```
semgrep --config "p/dgryski.semgrep-go" x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o dgryski.semgrep
semgrep --config "p/owasp-top-ten"      x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o owasp-top-ten.semgrep
semgrep --config "p/r2c-security-audit" x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o r2c-security-audit.semgrep
semgrep --config "p/r2c-ci"             x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o r2c-ci.semgrep
semgrep --config "p/ci"                 x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o ci.semgrep
semgrep --config "p/golang"             x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o golang.semgrep
semgrep --config "p/trailofbits"        x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o trailofbits.semgrep
```

AUTOMATED TESTING

Semgrep Results:

```
Scanning across multiple languages:
    <multilang> | 52 rules × 34 files
             go | 86 rules × 17 files

  100% |████████████████████████████████████████████████████████████████████████████████████████████████████████████████|51/51 tasks

Findings:

  ethash/ethash.go
     go.lang.security.audit.crypto.math_random.math-random-used
        Do not use `math/rand`. Use `crypto/rand` instead.
        Details: https://sg.run/6nX6

        134┆ temp := path + "." + strconv.Itoa(rand.Int())
           ⋮┆----------------------------------------
     trailofbits.go.questionable-assignment.questionable-assignment
        Should `c` be modified when an error could be returned?
        Details: https://sg.run/qq6y

        261┆ c.dump, c.mmap, c.cache, err = memoryMap(path, lock)
           ⋮┆----------------------------------------
        269┆ c.dump, c.mmap, c.cache, err = memoryMapAndGenerate(path, size, lock, func(buffer []uint32) { generateCache(buffer, c.epoch, seed) })
           ⋮┆----------------------------------------
     trailofbits.go.questionable-assignment.questionable-assignment
        Should `d` be modified when an error could be returned?
        Details: https://sg.run/qq6y

        347┆ d.dump, d.mmap, d.dataset, err = memoryMap(path, lock)
           ⋮┆----------------------------------------
        358┆ d.dump, d.mmap, d.dataset, err = memoryMapAndGenerate(path, dsize, lock, func(buffer []uint32) { generateDataset(buffer, d.epoch, cache) })


  ethash/sealer.go
     trailofbits.go.nondeterministic-select.nondeterministic-select
        Logic executed as a result of ticker `ticker` may execute more times than desired. When both
        `ticker` and `s.requestExit` are written to at the same time, the scheduler randomly picks a
        case to execute. As a result, the `ticker.C` may excute one more time than expected.
        Details: https://sg.run/X8dX

        275┆ ticker := time.NewTicker(5 * time.Second)
        276┆ defer ticker.Stop()
        277┆
        278┆ for {
        279┆     select {
        280┆     case work := <-s.workCh:
        281┆             // Update current work with new received block.
        282┆             // Note same work can be past twice, happens when changing CPU threads.
        283┆             s.results = work.results
        284┆             s.makeWork(work.block)
              [hid 52 additional lines, adjust with --max-lines-per-finding]

Some files were skipped or only partially analyzed.
  Scan skipped: 8 files matching .semgrepignore patterns
```

Gosec - Security Analysis Output Sample:

```
Results:

Golang errors in file: [./...]:



Summary:
  Gosec  : 2.11.0
  Files  : 0
  Lines  : 0
  Nosec  : 0
  Issues : 0
```

THANK YOU FOR CHOOSING

// HALBORN