SAST - Detailed Findings

dashevo/dapi

lib/config/index.js
severity:error rule:ajinabraham.njsscan.hardcoded_secrets.node_password: A hardcoded password in plain text is identified. Store it properly in an environment variable.
30 DEFAULT_CONFIG[OPTIONS.DASHCORE_RPC_PASS] = 'password';
-> false positive (default configuration)
---
severity:error rule:ajinabraham.njsscan.hardcoded_secrets.node_username: A hardcoded username in plain text is identified. Store it properly in an environment variable.
29 DEFAULT_CONFIG[OPTIONS.DASHCORE_RPC_USER] = 'dashrpc';
-> false positive (default configuration)
---
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via optionName
48 envConfig[optionName] = process.env[optionName];
-> false positive (sets env var)
---
lib/grpcServer/handlers/core/getStatusHandlerFactory.js
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via masternodeStatusResponse.state
53 const masternodeStatus = GetStatusResponse.Masternode.Status[masternodeStatusResponse.state];
-> false positive
---
lib/grpcServer/handlers/createGrpcErrorFromDriveResponse.js
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via code.toString()
64 const CommonErrorClass = COMMON_ERROR_CLASSES[code.toString()];
-> false positive (defined error codes from grpc-common/lib/server/error/GrpcErrorCodes)
---

lib/log/Logger.js
severity:warning rule:ajinabraham.njsscan.error_disclosure.generic_error_disclosure: Error messages with stack traces may expose sensitive information about the application.

```
53        try {
54          appendFileAsync(this.outputFilePath, EOL + log.trim(), { encoding: 'utf8' });
55        } catch (error) {
56          // eslint-disable-next-line no-console
57          console.error(`Error: Logger: ${error}`);
58        }
```

-> false positive (intern log)

---

lib/transactionsFilter/TransactionHashesCache.js
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via i

```
166       const { transaction } = this.transactions[i];

174       const { merkleBlock } = this.merkleBlocks[i];

183       const cachedBlock = this.blocks[i];
```

-> false positive (index)

---

test/functional/grpcServer/handlers/tx-filter-stream/subscribeToTransactionsWithProofsHandlerFactory.spec.js
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via receivedTransactions.length - 1

```
514     const lastReceivedTransaction = receivedTransactions[receivedTransactions.length - 1];
```

-> false positive (last element)

---

severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via historicalTransactions.length - 1

```
515     const lastHistoricalTransaction = historicalTransactions[historicalTransactions.length - 1];
```

-> false positive (last element)

---

dashevo/dashcore-lib

benchmark/script.js
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via i
23: var tx = block.transactions[i];
-> false positive (i belongs to for-loop)
---
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via j
25: var input = tx.inputs[j];
-> false positive (j belongs to for-loop)
---
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via k
31: var output = tx.outputs[k];
-> false positive (k belongs to for-loop)

benchmark/serialization.js
severity:warning rule:ajinabraham.njsscan.crypto_node.node_insecure_random_generator: crypto.pseudoRandomBytes()/Math.random() is a cryptographically weak random number generator.
28: var num = Math.round(Math.random() * 10000000000000);
-> false positive (random test data does not require secure randomness)

lib/block/merkleblock.js
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via options.hashesUsed++
219: const hash = this.hashes[options.hashesUsed++];
-> false positive (value gets checked beforehand if it is too large)

lib/crypto/bn.js
severity:warning
rule:javascript.lang.security.audit.detect-bracket-object-injection.detect-bracket-object-injection: Object injection via bracket notation via buf.length - 1 - i
16: buf2[i] = buf[buf.length - 1 - i];

205: rbuf[rbuf.length - 1 - i] = buf[buf.length - 1 - i];
-> false positives (buf.length comes from buf; i belongs to for-loop)

lib/crypto/hash.js
severity:warning rule:ajinabraham.njsscan.crypto_node.node_sha1: SHA1 is a a weak hash which is known to have collision. Use a strong hashing function.
15: return crypto.createHash('sha1').update(buf).digest();
-> low (while this code is just part of a method for calculating sha1-hashes, usage of that method may pose a security risk)

lib/crypto/random.js
severity:warning rule:ajinabraham.njsscan.crypto_node.node_insecure_random_generator: crypto.pseudoRandomBytes()/Math.random() is a cryptographically weak random number generator.
47:      r = Math.random() * b32;
-> low (part of a function that is supposed to generate insecure randomness, usage of that method may pose a security risk)

Npm-audit (dashcore-lib)

elliptic <6.5.4
Severity: **moderate**
Use of a Broken or Risky Cryptographic Algorithm -
https://github.com/advisories/GHSA-r9p9-mrjm-926w
@dashevo/dashcore-lib <=0.18.11 || >=0.19.0
  Depends on vulnerable versions of **elliptic**
-> false positive (naive implementation in elliptic does not check if point is on curve, but dashcore-lib enforces this validation via point.validate())