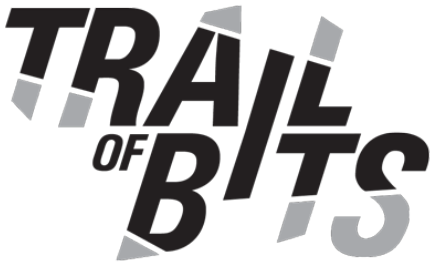

Nomic Cryptography

Security Assessment

April 15, 2020

Prepared For:
Patricio Palladino | *Nomic Labs*
patricio@nomiclabs.io

Prepared By:
Stefan Edwards | *Trail of Bits*
stefan.edwards@trailofbits.com

Brian Glas | *Trail of Bits*
brian.glas@trailofbits.com

Claudia Richoux | *Trail of Bits*
claudia.richoux@trailofbits.com

Will Song | *Trail of Bits*
will.song@trailofbits.com

Executive Summary

From February 18 to March 20, 2020 and April 6–10, 2020, Trail of Bits undertook a design review of the Nomic Labs Javascript Ethereum cryptography library. Three engineers worked over the course of two person-weeks from git commit 7252d02 on the `ethereum/js-ethereum-cryptography` repository. Our goal was to analyze the `js-ethereum-cryptography` package to determine possible design risks, then conduct a cryptographic analysis. Trail of Bits also reviewed the package's dependencies for vulnerabilities.

During the assessment, Trail of Bits found four issues related to the wrapper itself. In two medium-severity issues we identified, the code does not guard against dangerous types of developer misuse, and in two informational-severity issues we identified, the code could undergo some structural changes to be more opaque and make misuse more difficult (or to make the code easier to maintain). Finally, we identified two additional informational-severity issues in which dependency and submodule version management need to ensure that vulnerabilities in outside code are quickly patched.

The `js-ethereum-cryptography` package is intended to be a standard, lightweight cryptographic package for Ethereum development, supporting only the primary cryptographic primitives typically utilized in Javascript and Typescript development related to Ethereum. The developers have put thought and effort into keeping the package as small and self-contained as possible. The wrapper itself is very small and generally unproblematic, but could do much more to ensure that developers do not misuse the cryptographic primitives it exposes.

As all cryptographic primitives provided in this library come from dependencies and submodules, much of the library's security lies in maintaining the submodules in GitHub, and maintaining the dependencies imported by the library. According to the output of `npm audit`, there are **16748** packages in the dependency tree. Ensuring these are actively patched and updated is vital to the security of this library. Also, a full code review of the four primary cryptographic libraries (`bip39-lib`, `hdkey`, `scryptsy`, and `secp256k1-node`) could help determine the amount of risk inherent in this design for a standard Ethereum cryptographic library.

Project Dashboard

Application Summary

Name	js-ethereum-cryptography
Version	`7252d022ab0550b4b2b98ca502dc5e2efa1edbb3`
Type	Javascript
Platforms	Platform-independent

Engagement Summary

Dates	18 February–20 March and 6–10 April 2020
Method	Whitebox
Consultants Engaged	3
Level of Effort	2 person-weeks

Vulnerability Summary

Total High-Severity Issues	0	
Total Medium-Severity Issues	2	■ ■
Total Low-Severity Issues	0	
Total Informational-Severity Issues	4	■ ■ ■ ■
Total Undetermined-Severity Issues	0	
Total	6	

Category Breakdown

Cryptography	2	■ ■
Data Exposure	1	■
Patching	3	■ ■ ■
Total	6	

Engagement Goals

Nomic engaged Trail of Bits to assess its JavaScript cryptography library's cryptographic and API design decisions. As the library is a wrapper around previously existing JavaScript cryptography libraries, we sought to answer the following questions:

1. Does `js-ethereum-cryptography` expose any unsafe APIs?
2. Does `js-ethereum-cryptography` expose any opinionated safe APIs?
3. How easy is it for a developer to misuse this library?
4. How are sensitive dependencies managed and updated?

Coverage

js-ethereum-cryptography. We performed a manual review on this package, which led to four findings. We discovered that the library is a wrapper around other cryptography libraries and tries to present a uniform API across them. As such, it exposed the raw AES API (TOB-NOMIC-001), and its usage of `secp256k1` led to TOB-NOMIC-002. The lack of specific types for arguments provided to the underlying APIs led to TOB-NOMIC-003. Finally, a large amount of code duplication within the wrapper modules themselves was discovered (TOB-NOMIC-004).

js-ethereum-cryptography tests. We ran the tests and reviewed the test vectors to sanity-check them, but found no issues.

Dependencies. We ran various dependency analysis tools against the library and looked at the versions of submodules versus recent releases. Dependency versioning and vulnerability problems led to TOB-NOMIC-005, and submodule versioning issues led to TOB-NOMIC-006.

Recommendations Summary

This section aggregates all the recommendations made during the engagement. Short-term recommendations address the immediate causes of issues. Long-term recommendations pertain to the development process and long-term design goals.

Short Term

Add warnings and guidance in documentation about potentially dangerous AES usage patterns. The AES module is easy to use in an insecure manner. Developers could introduce serious vulnerabilities into their code if they are naive about cryptography. TOB-NOMIC-001

Explicitly state in the `secp256k1` module documentation that keys generated as random bytes must be validated before usage. Developers may not know to do this, as it is only mentioned in the repository for the dependency providing `secp256k1`. They could naively use invalid keys and seriously harm their system's security. TOB-NOMIC-002

Ensure that the code in `ethereum-cryptography/` and `ethereum-cryptography/pure/` get sufficient testing and that both are updated. Duplicated code can lead to bugs if it's not updated reliably. TOB-NOMIC-004

Routinely run an NPM audit to check for vulnerable dependencies, and update current and future vulnerable or out-of-date dependencies regularly. This will protect developers from bugs in cryptographic libraries, as well as other dependencies that could run in any number of sensitive environments. TOB-NOMIC-005

Ensure that `hdkey` should be on a non-release commit, and consider updating `secp256k1-node` to a newer release. These submodules are not on a release commit or are not on the newest release, respectively. Keeping dependencies updated is important for library security, especially when those dependencies provide cryptographic primitives. TOB-NOMIC-006

Long Term

Remove the function that does generic AES encryption and replace it with specific, well-documented functions for useful AES modes. This will prevent developers from misusing the AES function or choosing an insecure AES mode. TOB-NOMIC-001

Implement, document, and expose a function to generate `secp256k1` keys that do validation automatically. This should be based on the code in the `cryptocoinjs secp256k1` library's README to prevent attacks based on invalid keys. TOB-NOMIC-002

Consider adding strong types for functions and data used by the library. This could prevent many classes of bugs caused by developer misuse, and make it easier to add certain features to the library. TOB-NOMIC-003

Consider merging the code in `ethereum-cryptography/` and `ethereum-cryptography/pure/` to entirely avoid code duplication. This will make the library significantly smaller and easier to maintain. TOB-NOMIC-004

Automate dependency updates according to the recommendations below and cut down on the total number of dependencies if possible. Implementing these changes will proactively guard against vulnerabilities in dependencies and submodules. TOB-NOMIC-005

Automate submodule updates and vulnerability notifications. This could be a CI hook to check for new submodule releases. TOB-NOMIC-006

Findings Summary

#	Title	Type	Severity
1	AES modes of operation	Cryptography	Medium
2	secp256k1 interface for key generation	Cryptography	Medium
3	Strong types for security	Data Exposure	Informational
4	Duplicated code	Patching	Informational
5	Dependency management	Patching	Informational
6	Submodule management	Patching	Informational

1. AES modes of operation

Severity: Medium
Type: Cryptography
Target: `packages/ethereum-cryptography/aes.js`

Difficulty: Low
Finding ID: TOB-NOMIC-001

Description
The AES submodule accepts a string for mode of operation, and only checks that the string starts with “aes-” to validate it. This means that the available modes are controlled by the underlying cryptography library. On most systems this will allow the use of ECB mode, or allow developers to use other modes in insecure ways. AES is a cryptographic primitive that can easily be misused, with catastrophic consequences for security. Cryptographic libraries should make it difficult for developers to make these mistakes and provide guidance for writing code that is not controlled by the library.

Exploit Scenario
A developer who does not understand how to safely use AES uses this library’s AES primitives. They use ECB mode in production because they do not know what it is or the risks it entails. Alternatively, they use CBC without a MAC, or implement their error-handling incorrectly with a MAC, and expose themselves to a padding oracle attack.

A developer could also re-use IVs, cycle keys improperly, or use AEAD incorrectly due to lack of documentation for the dangers of these functions’ dangers.

Recommendations
Short term, add warnings in the documentation or code if a developer tries to use an AES mode other than those known to be necessary for Ethereum development. Add guidance for proper usage of those modes, potentially including tips to avoid re-using IVs and leaking information through error messages, and how to use authenticated encryption where necessary.

Long term, remove the function that does generic AES encryption with a string parameter for setting the encryption mode. Add specific and well-documented functions for each necessary AES mode that set the cipher mode, check key and IV lengths, check padding on data if necessary, and have arguments suitable for the mode in question. For example, as it stands now, ECB mode does not use the IV that is provided as an argument to the function. Examples of how to do this are in Appendix D. Having a generic AES function allows developers to do dangerous things with your library, so it should be removed.

2. secp256k1 interface for key generation

Severity: Medium
Type: Cryptography
Target: `packages/ethereum-cryptography/secp256k1.js`

Difficulty: Low
Finding ID: TOB-NOMIC-002

Description
The README says the `secp` library presents the same interface as `cryptocoinjs/secp256k1-node`. This API allows the generation of unsafe keys, and developers may not know to check that the `secp256k1` key is non-zero and not greater than the EC point group's size.

Exploit Scenario
A developer does not know to validate the `secp256k1` key and uses unvalidated bytes from `urandom`. If these bytes are zero, secrets can be leaked and signatures can be forged, depending on how the key is used. If these bytes correspond to an integer greater than the maximum key, computations with this key will be significantly slower, the key distribution will be biased and lead to invalidation of most proofs of ECC's security, and errors may crop up elsewhere when using this key. In all cases, security assumptions about ECC are rendered invalid if the key is not validated after generation.

Recommendations
Short term, explicitly state in the documentation for the `secp256k1` module that keys generated as random bytes must be validated before usage (and regenerated if they fail validation), just like they are in the `cryptocoinjs secp256k1` library.

Long term, implement, document, and expose a function to generate `secp256k1` keys that does generation and validation automatically. This should be based on the code in the `cryptocoinjs secp256k1` library's README.

3. Strong types for security

Severity: Informational
Type: Data Exposure
Target: `N/A, throughout`

Difficulty: N/A
Finding ID: TOB-NOMIC-003

Description
In numerous instances, generic buffers are passed around for IVs, keys, ciphertext, and messages. Although this works without issue, consider using explicit types to differentiate between these use cases. Explicit types make usage clearer for developers and avoid basic (but serious) bugs like misordering arguments.

This development pattern could also allow for the addition of features like zeroing keys from memory when they are garbage-collected with JS's new finalizer API, or changing how data is stored and processed under the hood without breaking backwards compatibility.

Exploit Scenario
A developer switches the IV and key arguments of AES encryption in their code, and the bug escapes their detection during testing. This could cause the key to be rotated every message instead of the IV, and then allow IV-reuse attacks on their code.

Using well-defined types instead of buffers could also prevent a developer from doing encryption with an unvalidated key (as in TOB-NOMIC-002). If the only way they could produce the opaque type for a `secp256k1` keypair was via a function that ensures the key is valid, they couldn't make the mistake of using a zero key that does nothing in elliptic curve operations.

Recommendations
Add types for various arguments to the code (potentially for keys on various curves, shared secrets, encrypted versus unencrypted messages) and converters from and to buffers if needed. These types could constrain things like length, since they are non-zero, and well-formed where needed. Examples of these checks include lengths of keys and nonces for AES, and validating private keys and their corresponding public keys for asymmetric cryptosystems. Cryptographic functions should only accept arguments of the expected types. Work with users to determine what their needs are and whether these solutions would be useful or a hindrance.

4. Duplicated code

Severity: Informational | Difficulty: N/A
Type: Patching | Finding ID: TOB-NOMIC-004
Target: `aes.ts, random.ts, ripemd160.ts, sha256.ts, pbkdf2.ts, scrypt.ts`

Description
We discovered some duplicated code in the wrapper codebase, specifically within the `pure` subdirectory. For example, the difference between `aes.ts` and `pure/aes.ts` is that the pure module uses `browserify-aes`, which utilizes node's crypto when run under node, while the former module can only be run under node, as it directly imports from `crypto`. We see a similar pattern with `randombytes` with respect to `random.ts` and `pure/random.ts`. The `ripemd160`, `sha256`, `pbkdf2`, and `scrypt` modules share similar duplication. Consider refactoring this code so it can be shared to reduce the probability of errors.

Exploit Scenario
A developer changes one of these modules without realizing there is code duplication elsewhere. This may have unintended consequences since both modules are expected to function in the exact same manner, but could be affected by this change.

Recommendations
Short term, ensure that both modules get sufficient testing, and document which modules also need changes if they are modified in any way.

Long term, investigate merging these modules into a single file and avoid code duplication altogether.

5. Dependency management

Severity: Informational
Type: Patching
Target: `package.json`

Difficulty: N/A
Finding ID: TOB-NOMIC-005

Description
Since the cryptographic libraries supported in this package are all dependencies and submodules, it's critical to carefully manage this library's dependencies. Including submodules and sub-dependencies, npm identifies 16748 dependencies of this library.

There were a number of dependencies with an older version defined in the `package.json`, but they were defined to allow for auto-updating to a minor revision. In the optional dependencies for the native packages, two of the four packages (`blake2` and `secp256k1`) were at least one major revision behind.

Running OWASP Dependency-Check on `js-ethereum-cryptography` uncovered a number of dependencies with vulnerabilities in the versions used (see output in Appendix B). The packages on the list below are deeper in the dependency tree, i.e., they are dependencies of dependencies. Some of the packages, like `handlebars:4.1.0`, have CVEs for arbitrary code execution.

`npm audit` was also run against both packages (see output in Appendix C). There were several warnings about deprecated and unmaintained dependencies, and some moderate-severity vulnerabilities in dependencies. Many of these were related to `rollup`, but some were related to cryptographic libraries. This command also revealed that the full dependency tree comprises 16748 packages.

Exploit Scenario
A cryptographic (or other) dependency that is out-of-date and contains known vulnerabilities is distributed to developers due to lack of dependency management. Their code uses this dependency and becomes vulnerable as well.

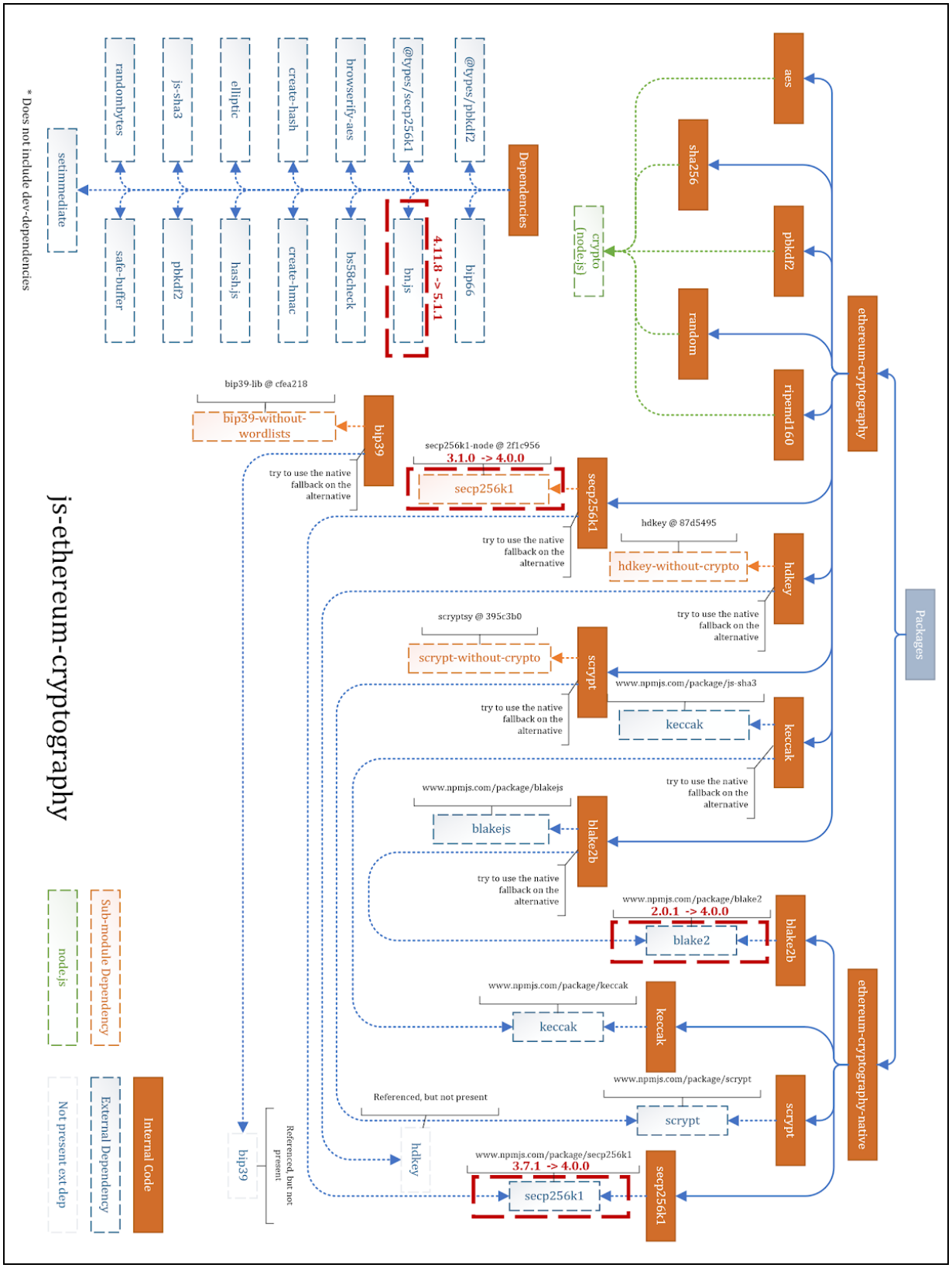

Figure TOB-NOMIC-005.1: Dependency and submodule graph with out-of-date packages highlighted in red.

Recommendations
Short term, run an npm audit regularly against the code, and update package versions to get rid of vulnerable dependencies.

Long term, for cryptographic implementation packages, we recommend keeping the auto-updates at a patch level or completely off, and to build a process of notification for any new releases for those packages so the changes can be reviewed and verified before inclusion into the `js-ethereum-cryptography` package. For supporting dependencies, we recommend minor version or patch -evel auto-updates and a process of notification for larger changes to facilitate review before updating. Add calling `npm audit` and OWASP Dependency Check into the CI process for the repository, so vulnerable dependencies are automatically flagged. Consider ways to cut down on the number of dependencies if possible,so the library has a smaller profile and there are fewer dependencies that can have vulnerabilities. Use GitHub's Dependabot to automatically file PRs to your repository when dependencies are updated.

References
1. https://dependabot.com/

6. Submodule management

Severity: Informational
Type: Patching
Target: `bip39-lib, hdkey, scryptsy, secp256k1-node`

Difficulty: N/A
Finding ID: TOB-NOMIC-006

Description
The current design relies on four submodules (`bip39-lib`, `hdkey`, `scryptsy`, `secp256k1-node`). As mentioned in this post, the following risks are present within each submodule:

- Every time one adds a submodule, changes its remote's URL, or changes the referenced commit for it, they demand a manual update by every collaborator.
- Forgetting this explicit update can result in silent regressions of the submodule-referenced commit.
- Commands such as `status` and `diff` display precious little info about submodules by default.
- Because lifecycles are separate, updating a submodule inside its container project requires two commits and two pushes.
- Submodule heads are generally detached, so any local update requires various preparatory actions to avoid creating a lost commit.
- Removing a submodule requires several commands and tweaks, some of which are manual and unassisted.
- `npm audit` will not alert developers to these issues.

Currently, `bip39-lib` and `scryptsy` are pointed at the most recent releases, which are also the most recent commits to the repositories. `hdkey` is one commit beyond the most recent release (1.1.1), and `secp256k1-node` is at release 3.7.1 which is two releases (about 6 months) behind the most recent one.

Exploit Scenario
Because submodule heads require manual updating, a developer using a library can easily forget to push out the updated submodule head or think that a recursive clone will checkout the latest commit, inadvertently leaving a vulnerable version of a submodule within the library. This will remain there indefinitely because the developer is not alerted by npm.

Recommendations
Short term, ensure that `hdkey` is on a non-release commit, and that `secp256k1-node` does not need to be updated to a newer version.

Long term, find a way to automate submodule updates, either by integrating the submodules as `npm` packages and using the suggestions from TOB-NOMIC-005, or writing a CI hook to check the upstreams for updates. Dependabot can be configured for git submodules.

References

1. https://medium.com/@porteneuve/mastering-git-submodules-34c65e940407
2. https://dependabot.com/submodules/

A. Vulnerability Classifications

Vulnerability Classes	
Class	**Description**
Access Controls	Related to authorization of users and assessment of rights
Auditing and Logging	Related to auditing of actions or logging of problems
Authentication	Related to the identification of users
Configuration	Related to security configurations of servers, devices, or software
Cryptography	Related to protecting the privacy or integrity of data
Data Exposure	Related to unintended exposure of sensitive information
Data Validation	Related to improper reliance on the structure or values of data
Denial of Service	Related to causing system failure
Error Reporting	Related to the reporting of error conditions in a secure fashion
Patching	Related to keeping software up to date
Session Management	Related to the identification of authenticated users
Timing	Related to race conditions, locking, or order of operations
Undefined Behavior	Related to undefined behavior triggered by the program

Severity Categories	
Severity	**Description**
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth
Undetermined	The extent of the risk was not determined during this engagement
Low	The risk is relatively small or is not a risk the customer has indicated is important
Medium	Individual user's information is at risk, exploitation would be bad for

	client's reputation, moderate financial impact, possible legal implications for client
High	Large numbers of users, very bad for client's reputation, or serious legal or financial implications

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploit was not determined during this engagement
Low	Commonly exploited, public tools exist or can be scripted that exploit this flaw
Medium	Attackers must write an exploit, or need an in-depth knowledge of a complex system
High	The attacker must have privileged insider access to the system, may need to know extremely complex technical details, or must discover other weaknesses in order to exploit this issue

B. OWASP Dependency-Check output

We ran the OWASP Dependency-Check tool against the `npm` configuration to determine if there were any known vulnerabilities inherited by the package. We summarize its output below. Because the known vulnerabilities below are not associated with the cryptographic functions of `js-ethereum-cryptography`, no issues were generated.

Dependency	Package	Highest Severity	CVE Count	Evidence Count
acorn:6.4.0	npm/acorn@6.4.0	MODERATE	1	15
bl:0.8.2	npm/bl@0.8.2	HIGH	2	7
handlebars:4.1.0	npm/handlebars@4.1.0	HIGH	10	3
js-yaml:3.13.0	npm/js-yaml@3.13.0	HIGH	2	3
lodash:4.17.11	npm/lodash@4.17.11	HIGH	2	3
minimist:1.2.0	npm/minimist@1.2.0	LOW	1	10
semver:2.3.2	npm/semver@2.3.2	HIGH	2	7
semver:5.7.1	npm/semver@5.7.1	MODERATE	1	7
yargs-parser:13.1.1	npm/yargs-parser@13.1.1	HIGH	1	9
yargs-parser:15.0.0	npm/yargs-parser@15.0.0	HIGH	1	9
yargs:14.2.2	npm/yargs@14.2.2	HIGH	1	7

C. NPM audit output

We ran `npm audit` on several configurations of the repository to get another sense of what sorts of known vulnerabilities were present within the package configuration. `npm audit` can also be configured to automatically install any non-breaking updates with `npm audit fix`. Please consult its documentation to determine how it should be best used within `js-ethereum-cryptography`.

`npm audit` on `packages/ethereum-cryptography` without submodules

```
npm WARN deprecated rollup-plugin-json@4.0.0: This module has been deprecated and is no
longer maintained. Please use @rollup/plugin-json.
npm WARN deprecated rollup-plugin-commonjs@10.1.0: This package has been deprecated and is
no longer maintained. Please use @rollup/plugin-commonjs.
npm WARN deprecated rollup-plugin-alias@2.2.0: This module has moved and is now available at
@rollup/plugin-alias. Please update your dependencies. This version is no longer maintained.
npm WARN deprecated rollup-plugin-replace@2.2.0: This module has moved and is now available
at @rollup/plugin-replace. Please update your dependencies. This version is no longer
maintained.
npm WARN deprecated rollup-plugin-node-resolve@5.2.0: This package has been deprecated and
is no longer maintained. Please use @rollup/plugin-node-resolve.
npm WARN deprecated core-js@2.6.11: core-js@<3 is no longer maintained and not recommended
for usage due to the number of issues. Please, upgrade your dependencies to the actual
version of core-js@3.
npm WARN deprecated request@2.88.2: request has been deprecated, see
https://github.com/request/request/issues/3142
npm WARN deprecated object-keys@0.2.0: Please update to the latest object-keys

npm notice created a lockfile as package-lock.json. You should commit this file.
added 1340 packages from 25 contributors and audited 16719 packages in 11.532s
found 2 moderate severity vulnerabilities
  run `npm audit fix` to fix them, or `npm audit` for details

$ npm audit

                       === npm audit security report ===

┌──────────────────────────────────────────────────────────────────────────────┐
│                                Manual Review                                 │
│           Some vulnerabilities require your attention to resolve             │
│                                                                              │
│         Visit https://go.npm.me/audit-guide for additional guidance          │
└──────────────────────────────────────────────────────────────────────────────┘

┌───────────────┬──────────────────────────────────────────────────────────────┐
│ Moderate      │ Regular Expression Denial of Service                         │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Package       │ semver                                                       │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Patched in    │ >=4.3.2                                                      │
└───────────────┴──────────────────────────────────────────────────────────────┘
```

```
| Dependency of | rollup-plugin-node-builtins [dev]                                |
| Path          | rollup-plugin-node-builtins > browserify-fs > levelup >          |
|               | semver                                                           |
| More info     | https://npmjs.com/advisories/31                                  |

| Moderate      | Memory Exposure                                                  |
| Package       | bl                                                               |
| Patched in    | >=0.9.5 <1.0.0 || >=1.0.1                                        |
| Dependency of | rollup-plugin-node-builtins [dev]                                |
| Path          | rollup-plugin-node-builtins > browserify-fs > levelup > bl       |
| More info     | https://npmjs.com/advisories/596                                 |

found 2 moderate severity vulnerabilities in 16719 scanned packages
  2 vulnerabilities require manual review. See the full report for details.
```

`npm audit` on `packages/ethereum-cryptography-native` without submodules.

```
npm notice created a lockfile as package-lock.json. You should commit this file.
added 160 packages and audited 422 packages in 2.209s
found 0 vulnerabilities
```

`npm audit` on `packages/ethereum-cryptography` with submodules.

```
npm notice created a lockfile as package-lock.json. You should commit this file.
audited 16748 packages in 4.649s

26 packages are looking for funding
  run `npm fund` for details

found 7 moderate severity vulnerabilities
  run `npm audit fix` to fix them, or `npm audit` for details
Brians-MBP-3:ethereum-cryptography Brian$ npm audit

                       === npm audit security report ===

# Run  npm update falafel --depth 6  to resolve 1 vulnerability

| Moderate      | Regular Expression Denial of Service                             |
| Package       | acorn                                                            |
| Dependency of | parcel-bundler [dev]                                             |
```

Path	parcel-bundler > @parcel/logger > grapheme-breaker > brfs > static-module > falafel > acorn
More info	https://npmjs.com/advisories/1488

Run `npm update acorn --depth 6` to resolve 3 vulnerabilities

Moderate	**Regular Expression Denial of Service**
Package	acorn
Dependency of	parcel-bundler [dev]
Path	parcel-bundler > htmlnano > uncss > jsdom > acorn
More info	https://npmjs.com/advisories/1488

Moderate	**Regular Expression Denial of Service**
Package	acorn
Dependency of	parcel-bundler [dev]
Path	parcel-bundler > htmlnano > uncss > jsdom > acorn-globals > acorn
More info	https://npmjs.com/advisories/1488

Moderate	**Regular Expression Denial of Service**
Package	acorn
Dependency of	webpack [dev]
Path	webpack > acorn
More info	https://npmjs.com/advisories/1488

Manual Review
Some vulnerabilities require your attention to resolve

Visit https://go.npm.me/audit-guide for additional guidance

Moderate	**Regular Expression Denial of Service**
Package	semver

```
┌─────────────┬──────────────────────────────────────────────────────────────┐
│ Patched in  │ >=4.3.2                                                      │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Dependency of │ rollup-plugin-node-builtins [dev]                          │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Path        │ rollup-plugin-node-builtins > browserify-fs > levelup >      │
│             │ semver                                                       │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ More info   │ https://npmjs.com/advisories/31                              │
└─────────────┴──────────────────────────────────────────────────────────────┘

┌─────────────┬──────────────────────────────────────────────────────────────┐
│ Moderate    │ Memory Exposure                                              │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Package     │ bl                                                           │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Patched in  │ >=0.9.5 <1.0.0 || >=1.0.1                                    │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Dependency of │ rollup-plugin-node-builtins [dev]                          │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Path        │ rollup-plugin-node-builtins > browserify-fs > levelup > bl   │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ More info   │ https://npmjs.com/advisories/596                             │
└─────────────┴──────────────────────────────────────────────────────────────┘

┌─────────────┬──────────────────────────────────────────────────────────────┐
│ Moderate    │ Regular Expression Denial of Service                         │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Package     │ acorn                                                        │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Patched in  │ >=5.7.4 <6.0.0 || >=6.4.1 <7.0.0 || >=7.1.1                  │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Dependency of │ rollup-plugin-node-globals [dev]                           │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ Path        │ rollup-plugin-node-globals > acorn                           │
├─────────────┼──────────────────────────────────────────────────────────────┤
│ More info   │ https://npmjs.com/advisories/1488                            │
└─────────────┴──────────────────────────────────────────────────────────────┘
found 7 moderate severity vulnerabilities in 16748 scanned packages
  run `npm audit fix` to fix 4 of them.
  3 vulnerabilities require manual review. See the full report for details.
```

`npm audit` on `packages/ethereum-cryptography-native` with submodules.

```
npm notice created a lockfile as package-lock.json. You should commit this file.
added 2 packages and audited 422 packages in 1.221s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

D. Example for new AES module interface

The example below illustrates one way to implement a new AES interface (albeit without authentication in this case), but anything that provides separate and clearly marked functionality for each supported AES mode would also correct the problems in TOB-NOMIC-001. A separate module or pair of functions should exist for each supported mode. Supporting and documenting use of authenticated encryption constructs might also be helpful for users.

```
AES_128_CBC.encrypt(key, data):
   iv = safe_iv_gen()
   ct = AES.encrypt(key, iv, data, mode='AES-128-CBC')
   return (iv, ct)


AES_128_CBC.decrypt(key, iv, data):
   pt = AES.decrypt(key, iv, data, mode='AES-128-CBC')
   return pt
```