npm-audit

Run a security audit

SYNOPSIS

```
npm audit [--json|--parseable|--audit-level=(low|moderate|high|critical)]
npm audit fix [--force|--package-lock-only|--dry-run]

common options: [--production] [--only=(dev|prod)]
```

EXAMPLES

Scan your project for vulnerabilities and automatically install any compatible updates to vulnerable dependencies:

```
$ npm audit fix
```

Run **`audit fix`** without modifying **`node_modules`** , but still updating the pkglock:

```
$ npm audit fix --package-lock-only
```

Skip updating **`devDependencies`** :

```
$ npm audit fix --only=prod
```

Have **`audit fix`** install semver-major updates to toplevel dependencies, not just semver-compatible ones:

```
$ npm audit fix --force
```

Do a dry run to get an idea of what **`audit fix`** will do, and *also* output install information in JSON format:

```
$ npm audit fix --dry-run --json
```

Scan your project for vulnerabilities and just show the details, without fixing anything:

```
$ npm audit
```

Get the detailed audit report in JSON format:

```
$ npm audit --json
```

Get the detailed audit report in plain text result, separated by tab characters, allowing for future reuse in scripting or command line post processing, like for example, selecting some of the columns printed:

```
$ npm audit --parseable
```

To parse columns, you can use for example `awk` , and just print some of them:

```
$ npm audit --parseable | awk -F $'\t' '{print $1,$4}'
```

Fail an audit only if the results include a vulnerability with a level of moderate or higher:

```
$ npm audit --audit-level=moderate
```

DESCRIPTION

The audit command submits a description of the dependencies configured in your project to your default registry and asks for a report of known vulnerabilities. The report returned includes instructions on how to act on this information. The command will exit with a 0 exit code if no vulnerabilities were found.

You can also have npm automatically fix the vulnerabilities by running `npm audit fix` . Note that some vulnerabilities cannot be fixed automatically and will require manual intervention or review. Also note that since `npm audit fix` runs a full-fledged `npm install` under the hood, all configs that apply to the installer will also apply to `npm install` – so things like `npm audit fix --package-lock-only` will work as expected.

By default, the audit command will exit with a non-zero code if any vulnerability is found. It may be useful in CI environments to include the `--audit-level` parameter to specify the minimum vulnerability level that will cause the command to fail. This option does not filter the report output, it simply changes the command's failure threshold.

CONTENT SUBMITTED

- npm_version

- node_version
- platform
- node_env
- A scrubbed version of your package-lock.json or npm-shrinkwrap.json

SCRUBBING

In order to ensure that potentially sensitive information is not included in the audit data bundle, some dependencies may have their names (and sometimes versions) replaced with opaque non-reversible identifiers. It is done for the following dependency types:

- Any module referencing a scope that is configured for a non-default registry has its name scrubbed. (That is, a scope you did a `npm login --scope=@ourscope` for.)
- All git dependencies have their names and specifiers scrubbed.
- All remote tarball dependencies have their names and specifiers scrubbed.
- All local directory and tarball dependencies have their names and specifiers scrubbed.

The non-reversible identifiers are a sha256 of a session-specific UUID and the value being replaced, ensuring a consistent value within the payload that is different between runs.

EXIT CODE

The `npm audit` command will exit with a 0 exit code if no vulnerabilities were found.

If vulnerabilities were found the exit code will depend on the `audit-level` configuration setting.