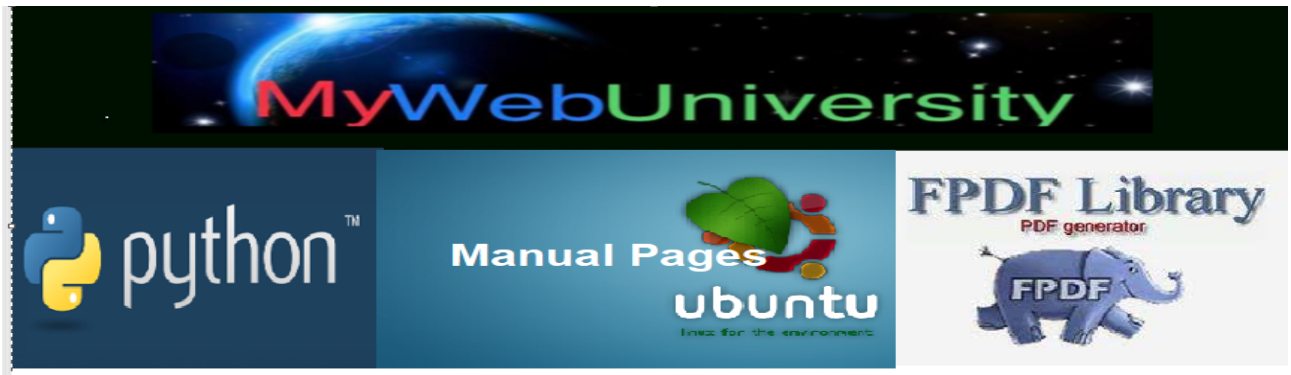

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'npm-audit.1'

$ man npm-audit.1

NPM-AUDIT(1) General Commands Manual NPM-AUDIT(1)

NAME

npm-audit

Synopsis

<!-- AUTOGENERATED USAGE DESCRIPTIONS -->

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. If any vulnerabilities are found, then the impact and appropriate remediation will be calculated. If the fix argument is provided, then remediations will be applied to the package tree.

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

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.

Audit Signatures

To ensure the integrity of packages you download from the public npm registry, or any reg?
istry that supports signatures, you can verify the registry signatures of downloaded pack?
ages using the npm CLI.

Registry signatures can be verified using the following audit command:

```
$ npm audit signatures
```

The npm CLI supports registry signatures and signing keys provided by any registry if the following conventions are followed:

? Signatures are provided in the package's packument in each published version within the dist object:

```
"dist":{
  "..omitted..": "..omitted..",
  "signatures": [{
    "keyid": "SHA256:{{SHA256_PUBLIC_KEY}}",
    "sig": "a312b9c3cb4a1b693e8ebac5ee1ca9cc01f2661c14391917dcb111517f72370809..."
  }]
}
```

See this example of a signed package from the public npm registry.

The sig is generated using the following template: ${package.name}@${package.ver?sion}:${package.dist.integrity} and the keyid has to match one of the public signing keys below.

? Public signing keys are provided at registry-host.tld/-/npm/v1/keys in the following for?mat:

```
{
  "keys": [{
    "expires": null,
    "keyid": "SHA256:{{SHA256_PUBLIC_KEY}}",
    "keytype": "ecdsa-sha2-nistp256",
    "scheme": "ecdsa-sha2-nistp256",
    "key": "{{B64_PUBLIC_KEY}}"
  }]
}
```

Keys response:

? expires: null or a simplified extended <a href="https://en.wikipedia.org/wiki/ISO_8601"

target="_blank">ISO 8601 format: YYYY-MM-DDTHH:mm:ss.sssZ

? keydid: sha256 fingerprint of the public key

? keytype: only ecdsa-sha2-nistp256 is currently supported by the npm CLI

? scheme: only ecdsa-sha2-nistp256 is currently supported by the npm CLI

? key: base64 encoded public key

See this example key's response from the public npm registry.

Audit Endpoints

There are two audit endpoints that npm may use to fetch vulnerability information: the Bulk Advisory endpoint and the Quick Audit endpoint.

Bulk Advisory Endpoint

As of version 7, npm uses the much faster Bulk Advisory endpoint to optimize the speed of calculating audit results.

npm will generate a JSON payload with the name and list of versions of each package in the tree, and POST it to the default configured registry at the path /-/npm/v1/security/advisories/bulk.

Any packages in the tree that do not have a version field in their package.json file will be ignored. If any --omit options are specified (either via the --omit config, or one of the shorthands such as --production, --only=dev, and so on), then packages will be omitted from the submitted payload as appropriate.

If the registry responds with an error, or with an invalid response, then npm will attempt to load advisory data from the Quick Audit endpoint.

The expected result will contain a set of advisory objects for each dependency that matches the advisory range. Each advisory object contains a name, url, id, severity, vulnerable_versions, and title.

npm then uses these advisory objects to calculate vulnerabilities and meta-vulnerabilities of the dependencies within the tree.

Quick Audit Endpoint

If the Bulk Advisory endpoint returns an error, or invalid data, npm will attempt to load advisory data from the Quick Audit endpoint, which is considerably slower in most cases.

The full package tree as found in package-lock.json is submitted, along

with the following pieces of additional metadata:

? npm_version

? node_version

? platform

? arch

? node_env

All packages in the tree are submitted to the Quick Audit endpoint.

Omitted dependency types are skipped when generating the report.

Scrubbing

Out of an abundance of caution, npm versions 5 and 6 would "scrub" any packages from the submitted report if their name contained a / character, so as to avoid leaking the names of potentially private packages or git URLs.

However, in practice, this resulted in audits often failing to properly detect meta-vulnerabilities, because the tree would appear to be invalid due to missing dependencies, and prevented the detection of vulnerabilities in package trees that used git dependencies or private modules.

This scrubbing has been removed from npm as of version 7.

Calculating Meta-Vulnerabilities and Remediations

npm uses the @npmcli/metavuln-calculator module to turn a set of security advisories into a set of "vulnerability" objects. A "meta-vulnerability" is a dependency that is vulnerable by virtue of dependence on vulnerable versions of a vulnerable package.

For example, if the package foo is vulnerable in the range >=1.0.2 <2.0.0, and the package bar depends on foo@^1.1.0, then that version of bar can only be installed by installing a vulnerable version of foo. In this case, bar is a "metavulnerability".

Once metavulnerabilities for a given package are calculated, they are cached in the ~/.npm folder and only re-evaluated if the advisory range changes, or a new version of the package is published (in which case, the new version is checked for metavulnerable status as well).

If the chain of metavulnerabilities extends all the way to the root

project, and it cannot be updated without changing its dependency ranges, then npm audit fix will require the --force option to apply the remediation. If remediations do not require changes to the dependency ranges, then all vulnerable packages will be updated to a version that does not have an advisory or metavulnerability posted against it.

Exit Code

The npm audit command will exit with a 0 exit code if no vulnerabilities were found. The npm audit fix command will exit with 0 exit code if no vulnerabilities are found or if the remediation is able to successfully fix all vulnerabilities.

If vulnerabilities were found the exit code will depend on the audit-level config.

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
```

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

higher:

$ npm audit --audit-level=moderate

Configuration

<!-- AUTOGENERATED CONFIG DESCRIPTIONS -->

See Also

? npm install

? config