Package 'rosv'

December 4, 2023

Title Client to Access and Operate on the 'Open Source Vulnerability' API

Version 0.5.1

Description Connect, query, and operate on information available from the 'Open Source Vulnerability' database <https://osv.dev/>. Although 'CRAN' has vulnerabilities listed, these are few compared to projects such as 'PyPI'. With tighter integration between 'R' and 'Python', having an 'R' specific package to access details about vulnerabilities from various sources is a worthwhile enterprise.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Suggests httptest2 (>= 1.0.0), knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports digest (>= 0.6.0), furrr (>= 0.3.0), httr2 (>= 1.0.0), jsonlite (>= 0.9.16), memoise (>= 2.0.0), purrr (>= 1.0.0), R6 (>= 2.0.0), utils

Depends R (>= 2.10)

URL https://al-obrien.github.io/rosv/, https://github.com/al-obrien/rosv

BugReports https://github.com/al-obrien/rosv/issues

VignetteBuilder knitr

NeedsCompilation no

Author Allen OBrien [aut, cre, cph]

Maintainer Allen OBrien <allen.g.obrien@gmail.com>

Repository CRAN

Date/Publication 2023-12-04 17:40:02 UTC

R topics documented:

`check_ecosystem` *Check input against possible ecosystems available*

Description

Internal function that ensures inputs for ecosystem are valid based upon what is available in the OSV database.

Usage

```
check_ecosystem(ecosystem, suppressMessages = TRUE)
```

Arguments

`ecosystem`	Character value for ecosystem(s) to check.
`suppressMessages`	Boolean value whether or not to suppress any messages.

Details

Will attempt to grab latest file and cache for the current R session. If session cannot access the online version, it will use a local copy shipped with the package.

Value

A character vector, the same as input if all are valid ecosystem names.

See Also

`fetch_ecosystems`

---

clear_osv_cache *Reset cached results of OSV calls*

Description

A thin wrapper around `forget` to clear cached results and deletes all cached files under the `ROSV_CACHE_GLOBAL` environment variable location.

Usage

```
clear_osv_cache()
```

Value

Invisibly returns a logical value of `TRUE` if cache cleared without error.

Examples

```
clear_osv_cache()
```

---

copy_rosv *Copy a {rosv} object*

Description

Create a copy of {rosv} R6 class objects to ensure original is not also updated with future changes.

Usage

```
copy_rosv(x, ...)
```

Arguments

`x`	Object to copy.
`...`	Additional parameters sent to R6's clone method.

Details

Since R6 classes have reference semantics, to escape updating original objects a clone can be made with this function.

Value

An R6 class object.

Examples

```
original_obj <- RosvQuery1$new(name = 'readxl', ecosystem = 'CRAN')
new_obj <- copy_rosv(original_obj)
```

`create_osv_list` *List packages identified in the OSV database*

Description

Create a list of package names and versions based upon vulnerabilities discovered in the OSV database using `osv_query`.

Usage

```
create_osv_list(
  rosv_query = NULL,
  as.data.frame = TRUE,
  sort = TRUE,
  delim = "\t",
  NA_value = NULL
)
```

Arguments

`rosv_query`	A table of vulnerabilities (created via `osv_query()`).
`as.data.frame`	Boolean value to determine if a data.frame should be returned.
`sort`	Boolean value to determine if results should be sorted by name and version.
`delim`	The deliminator to separate the package and version details (ignored if `as.data.frame` set to `TRUE`).
`NA_value`	Character value to replace missing versions (typically means all versions impacted).

Details

Requires an object of type `rosv_query` created by `osv_query`. This can be a selection of packages or all vulnerabilities for an ecosystem. Depending on use-case, users may prefer the vector based output with pairs of package names and versions separated by a provided value. Since only name and versions are returned, only one ecosystem can be operated on at a time.

Please note, the default behaviour of `osv_query()` is to return all packages (and versions) across ecosystems associated with discovered vulnerabilities. If a package is discovered across several vulnerabilities it will be listed multiple times, by default, in the returned content. Unlike `osv_query()`, `create_osv_list()` will further sort and return a unique set of packages. In most circumstances, users will create the `rosv_query` (via `osv_query()`) with the `all_affected` parameter set to `FALSE` so that only the package names of interest are returned.

Value

A `data.frame()` or vector object containing the package and version details.

See Also

`osv_query`

Examples

```
# List of a few PyPI packages in data.frame output
pypi_query <- osv_query(c('dask', 'dash', 'aaiohttp'),
                        ecosystem = rep('PyPI', 3),
                        all_affected = FALSE)
pypi_vul <- create_osv_list(pypi_query)
file_name1 <- file.path(tempdir(), 'pypi_vul.csv')
writeLines(pypi_vul, file_name1)

# All CRAN vulns in vector output
cran_query <- osv_query(ecosystem = 'CRAN', all_affected = FALSE)
cran_vul <- create_osv_list(cran_query, as.data.frame = FALSE, delim = ',')
file_name2 <- file.path(tempdir(), 'cran_vul.csv')
writeLines(cran_vul, file_name2)

# Clean up
try(unlink(c(file_name1, file_name2)))
```

`create_ppm_blacklist` *Create blacklist commands for Posit Package Manager*

Description

Use OSV data accessed via `osv_query` to create blacklist (i.e. blocklist) commands for the Posit Package Manager product.

Usage

```
create_ppm_blacklist(rosv_query, flags = NULL)
```

Arguments

`rosv_query`	A table of vulnerabilities (created via `osv_query()`).
`flags`	Global flag to append to commands.

Details

Although OSV has many databases for open source software, this function is only relevant for CRAN/Bioconductor and PyPI. To ensure the blacklist is applied to the appropriate target, it is encouraged to specify the name of the source used in your configuration as an additional flag parameter (see examples). Only one ecosystem can be used at a time to ensure there is not a mix of packages across ecosystems applied to incompatible sources.

Value

Character vector containing blacklist commands.

Examples

```
# Blacklist all CRAN package versions with a listed vulnerability
cran_vul <- osv_query(ecosystem = 'CRAN', all_affected = FALSE)
cmd_blist <- create_ppm_blacklist(cran_vul, flags = '--source=cran')
```

---

create_xref_whitelist *Cross reference a whitelist of packages to a vulnerability database*

Description

Search for package names for vulnerability information and selectively drop packages or define specific versions that should not be used in a curated repository.

Usage

```
create_xref_whitelist(packages, ecosystem, output_format = NULL)
```

Arguments

packages	Character vector of package names.
ecosystem	Character vector of ecosystem(s) within which the package(s) exist.
output_format	Type of output to create (default is NULL for a data.frame).

Details

Note that some version suffixes may have compatibility issues. For example, the use of *-git as a suffix may not be recognized and may need to be dropped. For more details on PyPI package version naming see https://peps.python.org/pep-0440/.

Due to variations in formatting from the OSV API, not all responses have versions associated and are not directly compatible with this function.

Although the default output is a data.frame, for PyPI packages a requirements.txt format can be created that defines which versions should not be allowed based upon the cross-referencing performed. This can be useful when curating repositories in Posit Package Manager.

Value

A data.frame or character vector containing cross-referenced packages.

See Also

PyPI package normalization

Examples

```
# Return xref dataset for CRAN package selection
cran_pkg <- c('readxl', 'dplyr')
cran_xref <- create_xref_whitelist(cran_pkg, ecosystem = 'CRAN')

# Create a requirements.txt with excluded versions
python_pkgs <- c('dask', 'aaiohttp', 'keras')
xref_pkg_list <- create_xref_whitelist(python_pkgs,
                                       ecosystem = 'PyPI',
                                       output_format = 'requirements.txt')
file_name <- file.path(tempdir(), 'requirements.txt')
writeLines(xref_pkg_list, file_name)

# Clean up
try(unlink(file_name))
```

---

fetch_ecosystems *Fetch all available ecosystems*

Description

Internal function used to fetch the available ecosystems in the OSV API.

Usage

```
fetch_ecosystems(offline = FALSE, refresh = FALSE)
```

Arguments

offline	Boolean, determine if using list bundled with package.
refresh	Boolean, force refresh of cache when using online list.

Details

The refresh parameter can be used to force the data to be pulled again even if one is available in the cached location. Since a fresh pull is performed for each R session, it is unlikely that this parameter is required and is primarily reserved for future use if functionality necessitates.

Value

A data.frame containing all the ecosystem names available in the OSV database.

See Also

check_ecosystem

`get_content` *Retrieve contents field from {rosv} R6 object*

Description

Retrieve contents field from {rosv} R6 object

Usage

```
get_content(x)
```

Arguments

x	An object made by {rosv}.

Value

Values contained in the content field of the object (data.frame or list).

Examples

```
test <- RosvQuery1$new(name = 'readxl', ecosystem = 'CRAN')
get_content(test)
```

`is_pkg_vulnerable` *Detect if package within ecosystem has reported vulnerabilities*

Description

Search the OSV database, by package name and its respective ecosystem, to determine if a vulnerability has ever been listed. If a package has been listed as impacted by a vulnerability this may warrant further queries to investigate specific versions that have been affected.

Usage

```
is_pkg_vulnerable(name, ecosystem, ...)
```

Arguments

name	Character vector of package names.
ecosystem	Character vector of ecosystem(s) within which the package(s) exist.
...	Any other parameters to pass to nested functions.

Value

A named vector of logical values indicating vulnerabilities.

Examples

```
is_pkg_vulnerable(c('dask', 'dplyr'), c('PyPI', 'CRAN'))
```

---

is_rosv *Is object made from {rosv} R6 class*

Description

Determine if object is an {rosv} type R6 class

Usage

```
is_rosv(x)
```

Arguments

x	Object to check.

Value

Boolean value based on if x is an R6 class made by {rosv}.

Examples

```
is_rosv(RosvQuery1$new(name = 'readxl', ecosystem = 'CRAN'))
```

normalize_pypi_pkg *Normalize package name to PyPI expectation*

Description

Perform package name formatting as PyPI is case insensitive and long runs of underscore, period, and hyphens are not recognized (- is same as –).

Usage

```
normalize_pypi_pkg(pkg_name)
```

Arguments

pkg_name	Character vector of package names.

Value

Character vector of normalized PyPI package names

See Also

PyPI Package Normalization

Examples

```
normalize_pypi_pkg(c('Dask', 'TenSorFlow'))
```

osv_count_vulns *Count the number of reported vulnerabilities*

Description

Search the OSV database, by package name and its respective ecosystem, and count the number of discovered vulnerabilities listed.

Usage

```
osv_count_vulns(name, ecosystem, ...)
```

Arguments

name	Character vector of package names.
ecosystem	Character vector of ecosystem(s) within which the package(s) exist.
...	Any other parameters to pass to nested functions.

Value

A named vector of numeric values indicating vulnerabilities.

Examples

```
osv_count_vulns(c('dask', 'dplyr'), c('PyPI', 'CRAN'))
```

`osv_download` *Download vulnerabilities from the OSV database*

Description

Use vulnerability IDs and/or an ecosystem name to download vulnerability files from OSV GCS buckets.

Usage

```
osv_download(
  vuln_ids = NULL,
  ecosystem,
  parse = TRUE,
  cache = TRUE,
  download_only = FALSE
)

.osv_download(vuln_ids = NULL, ecosystem, parse = TRUE, download_only = FALSE)

.osv_download_cache(
  vuln_ids = NULL,
  ecosystem,
  parse = TRUE,
  download_only = FALSE
)
```

Arguments

`vuln_ids`	Vector of vulnerability IDs (optional).
`ecosystem`	Ecosystem package lives within (must be set).
`parse`	Boolean value to set if the content field should be parsed from JSON list format.
`cache`	Boolean value to determine if should use a cached version of the function and API results.
`download_only`	Boolean value to determine if only the JSON files should be downloaded to disk.

Details

Although the end-result will be similar to the other API functions, this one specifically downloads .zip or .json files from the OSV GCS buckets. As a result, it has two main benefits. First, it can download the entire set of vulnerabilities listed for an ecosystem. Second, it has options to save the vulnerability files to disk. The files are saved to the R session's temp space, as defined by the environment variable `ROSV_CACHE_GLOBAL`.

Any ecosystems listed here can be downloaded. Only one ecosystem can be provided at a time.

Value

An R6 object containing API query contents.

Functions

- `.osv_download()`: Internal function to run `osv_download` without caching.
- `.osv_download_cache()`: Internal function to run a memoise and cached version of `osv_download`.

Examples

```
vulns <- osv_download("RSEC-2023-8", "CRAN")
get_content(vulns)

# Clean up
try(clear_osv_cache())
```

`osv_query` *Query OSV API for individual package vulnerabilities*

Description

Will connect to OSV API and query vulnerabilities from the specified packages. Unlike the other query functions, `osv_query` will only return content and not the response object. By default all vulnerabilities are returned for any versions of the package flagged in OSV. This can be subset manually or via the parameter `all_affected`.

Usage

```
osv_query(
  name = NULL,
  version = NULL,
  ecosystem = NULL,
  all_affected = TRUE,
  cache = TRUE,
  ...
)
```

Arguments

name	Character vector of package names.
version	Character vector of package versions, NA if ignoring versions.
ecosystem	Character vector of ecosystem(s) within which the package(s) exist.
all_affected	Boolean value, if TRUE return all package results found per vulnerability discovered.
cache	Boolean value to determine if should use a cached version of the function and API results.
...	Any other parameters to pass to nested functions.

Details

Since the query and batchquery API endpoints have different outputs, this function will align their contents to be a list of vulnerabilities. For 'query' this meant flattening the returned list once; for 'batchquery' the returned IDs are used to fetch additional vulnerability information and then flattened to a list.

If only an ecosystem parameter is provided, all vulnerabilities for that selection will be downloaded from the OSV database and parsed into a tidied table. Since some vulnerabilities can exist across ecosystems, all_affected may need to be set to FALSE.

Since the OSV database is organized by vulnerability, the returned content may have duplicate package details as the same package, and possibly its version, may occur within several different reported vulnerabilities. To avoid this behaviour, set the all_affected parameter to FALSE.

Due to variations in formatting from the OSV API, not all responses have versions associated in the response but instead use ranges. Filtering currently does not apply to this field and may return all versions affected within the ranges. If you suspect ranges are used instead of specific version codes, examine the response object using lower-level functions like osv_query_1().

To speed up the process for large ecosystems you can set future::plan() for parallelization; this will be respected via the furrr package. The default will be to run sequentially. There are performance impacts to allow for mixed ecosystems to be queried. For packages with many vulnerabilities, it can be faster to perform those separately so all vulnerabilities can be pulled at once and not individually. Alternative approaches may be implemented in future versions.

Value

A data.frame with query results parsed.

See Also

Ecosystem list

Examples

```
# Single package
pkg_vul <- osv_query('dask', ecosystem = 'PyPI')
```

```
# Batch query
name_vec <- c('dask', 'dash')
ecosystem_vec <- rep('PyPI', length(name_vec))
pkg_vul <- osv_query(name_vec, ecosystem = ecosystem_vec)
```

`osv_querybatch` *Query OSV API for vulnerabilities given a vector of packages*

Description

Using a vector of input information, query the OSV API for any associated vulnerability ID.

Usage

```
osv_querybatch(
  name = NULL,
  version = NULL,
  ecosystem = NULL,
  commit = NULL,
  purl = NULL,
  parse = TRUE,
  cache = TRUE,
  ...
)

.osv_querybatch(
  name = NULL,
  version = NULL,
  ecosystem = NULL,
  commit = NULL,
  purl = NULL,
  parse = TRUE,
  cache = TRUE,
  ...
)

.osv_querybatch_cache(
  name = NULL,
  version = NULL,
  ecosystem = NULL,
  commit = NULL,
  purl = NULL,
  parse = TRUE,
  cache = TRUE,
  ...
)
```

Arguments

name	Name of package.
version	Version of package.
ecosystem	Ecosystem package lives within (must be set if using name).
commit	Commit hash to query against (do not use when version set).
purl	URL for package (do not use if name or ecosystem set).
parse	Boolean value to set if the content field should be parsed from JSON list format.
cache	Boolean value to determine if should use a cached version of the function and API results.
...	Additional parameters passed to nested functions.

Details

The query is constructed from the provided set of vectors. Default will be NULL and thereby empty/null in the JSON request. If some values in the vector are missing, use NA. For many queries, the conversion to a formatted JSON request can be parallelized via {future}.

The returned information are vulnerability IDs and modified fields only, as per API instruction.

Value

An R6 object containing API query contents.

Functions

- .osv_querybatch(): Internal function to run osv_querybatch without caching.
- .osv_querybatch_cache(): Internal function to run a memoise and cached version of osv_querybatch.

See Also

Ecosystem list

Examples

```
osv_querybatch(c("commonmark", "dask"), ecosystem = c('CRAN', 'PyPI'))
```

osv_query_1 *Query OSV API for vulnerabilities based upon an individual package*

Description

Query the OSV API for vulnerabilities that include the individual package of interest. The request is automatically constructed from the provided elements and the returned values are parsed into a `data.frame`.

Usage

```
osv_query_1(
  name = NULL,
  version = NULL,
  ecosystem = NULL,
  commit = NULL,
  purl = NULL,
  parse = TRUE,
  cache = TRUE,
  ...
)

.osv_query_1(
  name = NULL,
  version = NULL,
  ecosystem = NULL,
  commit = NULL,
  purl = NULL,
  parse = TRUE,
  cache = TRUE,
  ...
)

.osv_query_1_cache(
  name = NULL,
  version = NULL,
  ecosystem = NULL,
  commit = NULL,
  purl = NULL,
  parse = TRUE,
  cache = TRUE,
  ...
)
```

Arguments

`name`	Name of package.

version	Version of package.
ecosystem	Ecosystem package lives within (must be set if using name).
commit	Commit hash to query against (do not use when version set).
purl	URL for package (do not use if name or ecosystem set).
parse	Boolean value to set if the content field should be parsed from JSON list format.
cache	Boolean value to determine if should use a cached version of the function and API results.
...	Additional parameters passed to nested functions.

Value

An R6 object containing API query contents.

Functions

- `.osv_query_1()`: Internal function to run `osv_query_1` without caching.
- `.osv_query_1_cache()`: Internal function to run a memoise and cached version of `osv_query_1`.

See Also

Ecosystem list

Examples

```
osv_query_1(commit = '6879efc2c1596d11a6a6ad296f80063b558d5e0f')
```

---

osv_scan *Use OSV database to scan for vulnerabilities*

Description

Scan project based upon specified mode to determine if any vulnerable packages are detected.

Usage

```
osv_scan(mode, ...)
```

Arguments

mode	The kind of scan to perform.
...	Parameters passed to specific underlying functions for mode selected.

Details

The available scanning modes are: 'r_project', 'renv', and 'r_libath'. The 'r_libpath' mode simply performs all R project related scans at once. Emphasis is placed on scans of R related content. Additional parsing and scanning modes will be added over time as needed. If a mode does not exist for a particular purpose, alternate functions such as `is_pkg_vulnerable()` can be used with any list of package names for ecosystems available in the OSV database.

Value

A data.frame specifying which packages are vulnerable or not.

See Also

`is_pkg_vulnerable`

Examples

```
osv_scan('r_libpath')
```

---

osv_vulns *Query OSV API for vulnerability information based on ID*

Description

Use vulnerability IDs to extract more detailed information, usually paired with `osv_querybatch()`.

Usage

```
osv_vulns(vuln_ids, parse = TRUE, cache = TRUE)

.osv_vulns(vuln_ids, parse = TRUE)

.osv_vulns_cache(vuln_ids, parse = TRUE)
```

Arguments

`vuln_ids`	Vector of vulnerability IDs.
`parse`	Boolean value to set if the content field should be parsed from JSON list format.
`cache`	Boolean value to determine if should use a cached version of the function and API results.

Value

An R6 object containing API query contents.

Functions

- `.osv_vulns()`: Internal function to run `osv_vulns` without caching.
- `.osv_vulns_cache()`: Internal function to run a memoise and cached version of `osv_vulns`.

Examples

```
vulns <- osv_vulns("RSEC-2023-8")
get_content(vulns)
```

RosvDownload *R6 Class for OSV Database Downloads*

Description

An R6 class to provide a lower-level interface to download from the OSV database GCS buckets.

Details

If no vulnerability IDs are provided, the entire set is downloaded from the ecosystem's all.zip file. JSON files are downloaded to the R session's temporary folder as dictated by the environment variable `ROSV_CACHE_GLOBAL`. Due to its similarity in parsing process, it simply inherits the method from the parent class `RosvQuery1`.

Any ecosystems listed here can be downloaded.

Value

An R6 object to operate with data downloaded from the OSV GCS buckets.

Super class

`rosv::RosvQuery1` -> `RosvDownload`

Public fields

`osv_cache_dir` Location of cached vulnerability JSON files.

`content` Content from downloading the vulnerabilities.

`time_stamp` Time stamp associated with run.

`date_stamp_hash` Hashed date from time stamp.

`ecosystem` The ecosystem used upon creation.

`vuln_ids` The vulnerability IDs, if provided.

`request` The URLs to request downloaded files.

Methods

Public methods:

- RosvDownload$new()
- RosvDownload$download()
- RosvDownload$run()
- RosvDownload$print()
- RosvDownload$clone()

Method `new()`**:** Set the core request details for subsequent use when called in `run()` method.

Usage:

```
RosvDownload$new(vuln_ids = NULL, ecosystem)
```

Arguments:

`vuln_ids` Character vector of vulnerability IDs.

`ecosystem` Ecosystem package lives within (must be set).

Method `download()`**:** Download vulnerabilities from provided `ecosystem` to disk, the location is recorded under the `osv_cache_dir` field. Will overwrite any existing files in the cache.

Usage:

```
RosvDownload$download()
```

Method `run()`**:** Load vulnerabilities to the R session. The entire contents of each vulnerability file will be loaded. Subsequent use of the `parse()` method will shrink the memory footprint as not all contents will be carried across.

Usage:

```
RosvDownload$run()
```

Method `print()`**:** Print basic details of query object to screen.

Usage:

```
RosvDownload$print(...)
```

Arguments:

`...` Reserved for possible future use.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RosvDownload$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

https://google.github.io/osv.dev/data/#data-dumps

Examples

```
query <- RosvDownload$new(ecosystem = 'CRAN')
query
```

RosvQuery1 *R6 Class for OSV Query Endpoint*

Description

An R6 class to provide a lower-level interface to the query endpoint of the OSV API.

Details

Pageination is implemented via `httr2::req_perform_iterative()` and a private method for extracting tokens automatically. When initialized, the page_token is set to `NULL`; if a token is generated for large results the process is handled internally. The response object will contain a list of all returned responses before any formatting occurred. The content field will contain the list of vulnerabilities which may be further parsed into a table format.

Value

An R6 object to operate with OSV query endpoint.

Public fields

`request` Request object made by `httr2`.

`content` Body contents of response from OSV API.

`response` Response object returned from OSV API.

Methods

Public methods:

- `RosvQuery1$new()`
- `RosvQuery1$run()`
- `RosvQuery1$parse()`
- `RosvQuery1$print()`
- `RosvQuery1$clone()`

Method `new()`**:** Set the core request details for subsequent use when called in `run()` method.

Usage:

```
RosvQuery1$new(
  commit = NULL,
  version = NULL,
  name = NULL,
  ecosystem = NULL,
  purl = NULL
)
```

Arguments:

`commit` Commit hash to query against (do not use when version set).

`version` Version of package.

`name` Name of package.

`ecosystem` Ecosystem package lives within (must be set if using `name`).

`purl` URL for package (do not use if `name` or `ecosystem` is set).

Method `run()`**:** Perform the request and return response for OSV API call.

Usage:

```
RosvQuery1$run()
```

Method `parse()`**:** Parse the contents returned into a tidier format. Can use `future` plans to help parallelize. Not all contents are parsed.

Usage:

```
RosvQuery1$parse()
```

Method `print()`**:** Print basic details of query object to screen.

Usage:

```
RosvQuery1$print(...)
```

Arguments:

`...` Reserved for possible future use.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RosvQuery1$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

https://google.github.io/osv.dev/post-v1-query/

Examples

```
query <- RosvQuery1$new(commit = '6879efc2c1596d11a6a6ad296f80063b558d5e0f')
query
```

RosvQueryBatch *R6 Class for OSV Querybatch Endpoint*

Description

An R6 class to provide a lower-level interface to the querybatch endpoint of the OSV API. Batches are enforced to only process by commit hash, purl, or name+ecosystem. This avoids some confusion as to which is taken preferentially and simplifies query creation.

Details

Pageination is implemented via `httr2::req_perform_iterative()` and a private method for extracting tokens automatically. When initialized, the page_token is set to `NULL`; if a token is generated for large results the process is handled internally. The response object will contain a list of all returned responses before any formatting occurred. The content field will contain the list of results with vulnerabilities which may be further parsed into a table format.

Value

An R6 object to operate with OSV querybatch endpoint.

Super class

`rosv::RosvQuery1` -> `RosvQueryBatch`

Methods

Public methods:

- `RosvQueryBatch$new()`
- `RosvQueryBatch$run()`
- `RosvQueryBatch$parse()`
- `RosvQueryBatch$clone()`

Method `new()`**:** Set the core request details for subsequent use when called in `run()` method.

Usage:

```
RosvQueryBatch$new(
  commit = NULL,
  version = NULL,
  name = NULL,
  ecosystem = NULL,
  purl = NULL
)
```

Arguments:

`commit` Commit hash to query against (do not use when version set).

`version` Version of package.

`name` Name of package.

`ecosystem` Ecosystem package lives within (must be set if using `name`).

`purl` URL for package (do not use if `name` or `ecosystem` is set).

Method `run()`**:** Perform the request and return response for OSV API call.

Usage:

```
RosvQueryBatch$run()
```

Method `parse()`**:** Parse the contents returned into a tidier format.

Usage:

```
RosvQueryBatch$parse()
```

Details: When no result is found, any empty list is returned by the API, which during parsing will be dropped as the list is flattened. However, the index of the list is still accessible and the dropped items can easily be identified from the results column. Not all contents are parsed.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RosvQueryBatch$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

https://google.github.io/osv.dev/post-v1-querybatch/

Examples

```
pkgs <- c('jinja2', 'dask')
ecosystem <- rep('PyPI', length(pkgs))
batchquery <- RosvQueryBatch$new(name = pkgs, ecosystem = ecosystem)
batchquery
```

`RosvVulns` *R6 Class for OSV Vulns Endpoint*

Description

An R6 class to provide a lower-level interface to the vulnerability endpoint of the OSV API.

Value

An R6 object to operate with OSV vulns endpoint.

Super class

`rosv::RosvQuery1` -> `RosvVulns`

Methods

Public methods:

- RosvVulns$new()
- RosvVulns$run()
- RosvVulns$print()
- RosvVulns$clone()

Method `new()`**:** Set the core request details for subsequent use when called in `run()` method.

Usage:

```
RosvVulns$new(vuln_ids)
```

Arguments:

`vuln_ids` Character vector of vulnerability IDs.

Method `run()`**:** Perform the request and return response for OSV API call.

Usage:

```
RosvVulns$run()
```

Method `print()`**:** Print basic details of query object to screen.

Usage:

```
RosvVulns$print(...)
```

Arguments:

`...` Reserved for possible future use.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RosvVulns$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

https://google.github.io/osv.dev/get-v1-vulns/

Examples

```
vulns <- RosvVulns$new(c('RSEC-2023-6', 'GHSA-jq35-85cj-fj4p'))
vulns
```

Index